WIRELESS SENSOR NETWORK SECURITY

Cryptology and Information Security Series

The Cryptology & Information Security Series (CISS) presents the latest research results in the theory and practice, analysis and design, implementation, application and experience of cryptology and information security techniques. It covers all aspects of cryptology and information security for an audience of information security researchers with specialized technical backgrounds.

Coordinating Series Editors: R.C.-W. Phan and J. Zhou

Volume 1

ISSN 1871-6431

Wireless Sensor Network Security

Edited by

Javier Lopez

University of Malaga, Spain

and

Jianying Zhou

Institute for Infocomm Research, Singapore

IOS Press

Amsterdam • Berlin • Oxford • Tokyo • Washington, DC

ISBN 978-1-58603-813-7
Library of Congress Control Number: 2008925581

Publisher
IOS Press
Nieuwe Hemweg 6B
1013 BG Amsterdam
The Netherlands
fax: +31 20 687 0019
e-mail: order@iospress.nl

Distributor in the UK and Ireland
Gazelle Books Services Ltd.
White Cross Mills
Hightown
Lancaster LA1 4XS
United Kingdom
fax: +44 1524 63232
e-mail: sales@gazellebooks.co.uk

Distributor in the USA and Canada
IOS Press, Inc.
4502 Rachael Manor Drive
Fairfax, VA 22032
USA
fax: +1 703 323 3668
e-mail: iosbooks@iospress.com

PRINTED IN THE NETHERLANDS

Wireless Sensor Network Security
J. Lopez and J. Zhou (Eds.)
IOS Press, 2008

Contents

Overview of Wireless Sensor Network Security 1
Javier Lopez and Jianying Zhou

Vulnerabilities and Attacks in Wireless Sensor Networks 22
Zinaida Benenson, Peter M. Cholewinski and Felix C. Freiling

Symmetric Primitives 44
Manfred Aigner, Martin Feldhofer and Stefan Tillich

Public-Key Primitives 77
Lejla Batina, Stefaan Seys, Bart Preneel and Ingrid Verbauwhede

Key Management in Wireless Sensor Networks 110
Seyit Ahmet Çamtepe and Bülent Yener

WSN Link-Layer Security Frameworks 142
Ioannis Krontiris, Tassos Dimitriou, Hamed Soroush and Mastooreh Salajegheh

Secure Routing in Wireless Sensor Networks 164
Gergely Ács and Levente Buttyán

Secure Data Aggregation in Wireless Sensor Networks 204
Sanjeev Setia, Sankardas Roy and Sushil Jajodia

Privacy Protection Mechanisms for Sensor Networks 223
Efthimia Aivaloglou, Stefanos Gritzalis and Sokratis Katsikas

Intrusion Detection Techniques in Sensor Networks 251
Aikaterini Mitrokotsa and A. Karygiannis

Remote Attestation – Identification 273
Alec Yasinsac

On the Hardware Implementation Efficiency of Cryptographic Primitives 291
Rodrigo Roman, Cristina Alcaraz and Nicolas Sklavos

Author Index 313

This page intentionally left blank

Wireless Sensor Network Security
J. Lopez and J. Zhou (Eds.)
IOS Press, 2008

Overview of Wireless Sensor Network Security

Javier Lopez[a] and Jianying Zhou[b]
[a]*University of Malaga, Spain*
[b]*Institute for Infocomm Research, Singapore*

1. Background of WSN

The advances on miniaturization techniques and wireless communications have made possible the creation and subsequent development of the *Wireless Sensor Networks* (WSN) paradigm. The main purpose of WSN is to serve as an interface to the real world, providing physical information such as temperature, light, radiation, etc. to a computer system. The major difference between this type of networks and wired networks is their decentralized and specialized nature. In WSN, all its members collaborate towards the common goal of obtaining or deducing certain physical information from their environment. Moreover, WSN is capable of self-organization, thus it can be deployed in a certain context without requiring the existence of a supporting infrastructure.

The functionality and behaviour of WSN are also different from another wireless network paradigm, *Mobile Ad Hoc Network* (MANET). First, all devices in WSN are totally autonomous, not controlled by human users. Also, those devices are much more constrained in terms of battery life and processing power, so it can only offer a simple and predefined set of tasks, whereas a MANET node is usually a PDA-like device with much more functionality and resources. In addition, the density of WSN is usually higher than in MANET.

The infrastructure of WSN can be divided into two parts, the *data acquisition* network and the *data dissemination* network. The data acquisition network contains the sensor network "per se": sensor nodes and base stations. Sensor nodes are a collection of small devices with the task of measuring the physical data of its surroundings, and base stations are powerful devices in charge of collecting data from the nodes and forwarding control information from the users. On the other hand, the data dissemination network is a combination of wired and wireless networks that provides an interface of the data acquisition network to any user.

As aforementioned, the major elements of WSN are the sensor nodes and the base stations. In fact, they can be abstracted as the "sensing cells" and the "brain" of the network, respectively. Sensor nodes get the physical information of the surroundings using its built-in sensors, process the raw information taking advantage of its computational capabilities, and communicate with other nodes in its surroundings using a wireless channel. All sensor nodes are battery-powered; hence, totally independent and able to operate autonomously, if required. Nevertheless, they can also collaborate with other nodes in pursuing a common goal, such as vehicle tracking. In contrast, the base station is a centralized element that is used for accessing the services provided by the sensor network. All data coming from the sensor nodes, as well as all control commands that can be issued to those nodes, will traverse the base station. In the cases where the information obtained by the WSN has to be accessible from more than one point, the coexistence of several base stations is feasible. A base station can be mobile, either autonomously or dependently. In some cases, it positions itself to obtain samples of the environment based on the information supplied by the sensor nodes. In other cases, it is a PDA-like device used by a human in order to access on the spot to the information of the WSN.

There are two basic WSN architectures, *hierarchical* and *flat*, that specify how the sensors group themselves to achieve specific goals. In flat configurations, all the nodes contribute in the decision-making process and participate in the internal protocols (like routing). Conversely, in hierarchical configurations the network is divided into clusters or group of nodes. Organizational decisions, like data aggregation, are made by a single entity called "cluster head". It should be noticed that it is also possible to have a combination of the two previous configurations into the same network; for instance, to avoid situations where the "spinal cord" of the network - the cluster heads - fail and the information must be routed to the base station.

Regarding the services offered by a WSN, they can be classified into three major categories: monitoring, alerting, and provisioning of information "on-demand". As for the first case, sensor nodes can continuously monitor certain features of their surroundings (e.g. measuring the ambient noise level) and timely send such information to the base station. In the second case, sensors can check whether certain physical circumstances (e.g. a fire) are occurring, alerting the users of the system when an alarm is triggered. In the last case, the network can be queried about the actual levels of a certain feature, providing information "on-demand". Due to the computational capabilities of the nodes, it is possible to reprogram the network during its lifetime, or even use it as a distributed computing platform under specific circumstances.

A number of research projects on WSN have been carried out, and some prototypes have been developed, for example, monitoring of aging infrastructures and hazardous materials [WINES2007], management of agricultural scenarios such as vineyards [Beck2004], wireless vital sign sensors [CodeBlue2004].

2. Threats in WSN

As in all computing environments, it is essential to assure the proper functionality of WSN in order to allow the correct provisioning of services. Such networks should comply with certain security requirements, such as confidentiality, integrity, authentication, and others, derived from the application context. However, achieving this goal is not an easy task for WSN as it is especially vulnerable against external and internal attacks due to its peculiar characteristics. The devices of the network, sensor nodes, are highly constrained in terms of computational capabilities, memory, communication bandwidth and battery power. Additionally, it is easy to physically access such nodes because they must be located near the physical source of the events, and they usually are not tamper-resistant due to cost constraints. Furthermore, any internal or external device can access the information exchange because the communication channel is public.

As a result, WSN has to face multiple threats that may easily hinder its functionality and nullify the benefits of using its services. These threats can be categorized as follows:

- Common attacks
- Denial of service attack
- Node compromise
- Impersonation attack
- Protocol-specific attacks

Due to the features of WSN, there are some *specific attacks* targeting the communication channels. An adversary can easily retrieve valuable data from the transmitted packets that are sent (*Eavesdropping*). That adversary can also simply intercept and modify the packets' content meant for the base station or intermediate nodes (*Message Modification*), or re-transmit the contents of those packets at a later time (*Message Replay*). Finally, the attacker can send out false data into the network, maybe masquerading as one of the sensors, with the objectives of corrupting the collected sensors' reading or disrupting the internal control data (*Message Injection*).

Since WSN is a wireless service-oriented infrastructure, one of the most problematic attacks that it may face is the *Denial of Service* (DoS) attack. A DoS attack on WSN may take several forms: *node collaboration*, in which a set of nodes act maliciously and prevent broadcast messages from reaching certain section(s) of the sensor network; *jamming attack*, in which an attacker jams the communication channel and avoids any member of the network in the affected area to send or receive any packet; and *exhaustion of power*, in which an attacker repeatedly requests packets from sensors to deplete their battery life.

A sensor node is considered as being *compromised* when an attacker, through various means, gains control or access to the sensor node itself after it has been deployed. Attacks can be invasive or non-invasive. An invasive physical attack is defined as an attack where the attacker physically breaks into the hardware by modifying its hardware structure (e.g. using focused ion beam, or drilling a hole in the storage media). On the other hand, A non-invasive attack is defined as an attack where the data is taken from the hardware device without any form of structural modification to the device itself (e.g. taking advantage of the JTAG interface [Becher2006]). Various complex attacks can be easily launched from compromised nodes, since the subverted node is a full-fledged member of the network.

The most common attack that can be launched using a compromised node is the *impersonation attack*, in which a malicious node impersonates a legitimate node and uses its identity to mount an active attack such as Sybil or node replication. In a Sybil attack, a single node takes on multiple identities to deceive other nodes. A node that wishes to conduct the Sybil attack can adopt a new identity by creating a new identity or by stealing the identity of an existing node. On the other hand, node or identity replication is the simple duplication of sensor nodes. As sensor nodes tend to be physically unprotected, it is feasible for an attacker to capture, replicate and insert duplicate nodes back into selected regions of the network. Node replication is different from a Sybil attack in that the multiple nodes are duplicates and basically have the same identities.

Complex attacks from subverted nodes can target the internal protocols used in the network, such as routing. Attacks against routing protocols in WSN fall into one of the following categories [Karlof2003]: corruption of the internal control information such as the routing tables (*Spoofed Routing Information*), selective forwarding of the packets that traverse a malicious node depending on some criteria (*Selective Forwarding*), creation of a "wormhole" that captures the information at one location and replays them in another location either unchanged (*Wormhole Attack*) or tampered (*Sinkhole Attack*), creation of false control packets during the deployment of the network (*Hello Flood Attack*), and creation of false acknowledge information (*Acknowledgment Spoofing*).

Other protocols can be attacked as well, such as data aggregation (e.g. by forging the data before, during or after the aggregation process).

Therefore, it is necessary to provide WSN with basic security mechanisms and protocols that can guarantee a minimal protection to the services and the information flow. This means the hardware layer needs to be protected against node compromise, the communication channels should meet certain security goals (like confidentiality, integrity and authentication), and the protocols and services of the network must be robust against any possible interferences.

3. Security on Hardware Layer

One of the most fruitful attacks that can be launched against a sensor node is node compromise. Nodes are easy to access, since they have to be physically near the event they monitor. Once the attacker obtains, subverts, and takes control over the node, he can access its internal information, and also use it for malicious purposes by launching complex or stealthy attacks. Therefore, there should be some kind of protection on the hardware layer to avoid such attacks, like a *Tamper Proof Module* (TPM). Such modules allow the security credentials to be stored securely, preventing an attacker from retrieving the credentials when the sensor node is compromised. Once a tampering of the chip is detected, the TPM will destroy the keys and other information stored in the module.

Although adding a TPM into a sensor node would help to defend against node compromise, the addition of the module will significantly increase its overall cost. Also, using a TPM would mean that the node cannot be reprogrammed after its deployment. As a result, it is necessary to use other software-based mechanisms that are not dependent on any hardware configuration. Examples of those mechanisms are the *code attestation* techniques, which can detect if a node has been compromised, or *code obfuscation* techniques, which can generate different versions of the sensor software for each node.

3.1. Code Attestation

Software based attestation enables a third party to verify the code running on the system to detect any maliciously altered code. Usually code attestation is done through the use of special hardware mechanisms proposed by the Trusted Computing Group (TCG) and Next Generation Secure Computing Base (NGSCB). However, these hardware mechanisms are costly and are not implemented in the current version of the sensors, as of 2007. Thus this kind of

software attestation is designed to provide the detection of malicious code alteration and verify that the sensors are using the correct codes.

A verification procedure is needed to effectively verify that the sensor's code is correct and not maliciously altered. A verifier needs to generate a random challenge to be sent to the sensor. The sensor will then use this challenge to generate a response using the verification procedure. The verifier will then compare the response against an expected value. Any discrepancy would imply that the code in the sensor has been modified. Data used in the code attestation usually include the clock speed, the instruction set architecture, the memory architecture of the microcontroller and the size of the device's memory.

The verification procedure is mostly based on the "pseudorandom memory traversal" concept: using a seed (i.e. the random challenge) provided by the verifier, the sensor must randomly access some positions of its own memory, summarizing them into one report that will be used as a response. If the sensor is malicious, the time used on calculating a valid response will be longer, and such delay can be detected by the verifier [Seshadri2004]. An improvement over this basic scheme is to send an obfuscated attestation routine alongside with the seed, thus delaying even more the efforts of the malicious adversary to reply with a valid response [Shaneck2005].

As software based attestation requires the creation of a response from the challenged node, an attacker may be able to generate a correct response using a more powerful machine. In this way, the attacker is able to obtain the result in a shorter time. A sensor, which is also controlled by the attacker, will then forward the correct result back to the verifier within the expected time frame. It is important to improve upon the current software based attestation schemes to prevent the race condition, which allows attackers to generate the result using a more powerful machine. The attestation method should be able to withstand such attacks and at the same time consume less energy. The design should also prevent the attacker from launching a DoS attack on the sensor by repeating the request to verify the sensor.

3.2. Code Obfuscation

Code obfuscation, or diversification, is a mechanism that allows the protection of a valuable piece of information (e.g. the security credentials) contained inside the node. By obfuscating the code and data, the amount of time needed by the attacker to analyze the compromised nodes will increase, thus it will be more difficult to deduce the secrets from the extracted contents of program flash, the EEPROM or the SRAM. The obfuscation methods must not be equal

for all the nodes. This is to prevent the attacker from using the same method to retrieve the secrets once the attacker is successful in compromising one node.

Those diversification techniques may include stack randomization, instruction set randomization, library randomization, and system call randomization. Also, the diversification can be done through the network level using different OS, application and communication protocol within a network system. In a sensor node, it is possible to hide vital information, such as the secret keys, using a hash function to scramble the information in the data segment [Alarifi2006]. By hiding the keys in a randomized manner, it would be difficult for the attacker to find the keys from the downloaded EEPROM.

In general, code obfuscation can be done in 5 steps [Alarifi2006]. The first stage is code pre-processing, in which comments, standardized loops, control instructions, split variables declarations and format instructions are removed. The next step is intermediate code construction, where the program is re-generated into another semantically equivalent program at the source code level. The new code is flattened to make its control flow harder to be analyzed. The flattened code is then passed through a random code generator to diversify the codes for the sensor nodes. The generator will generate a different version of the same code for each individual node. Lastly, comments and debugging information are removed and the identifiers are changed into meaningless names.

4. Security Primitives

All sensor nodes in WSN use power-efficient radio transceivers for their communications. Most of the existing sensor nodes operate in unlicensed frequency bands (such as the 433 MHz ISM-band in Europe with a maximum capacity of 19.2 Kbps), but some nodes follow the IEEE 802.15.4 standard for Personal Area Networks [IEEE802.15.4], that has a maximum data throughput of 250 Kbps. Regardless of the underlying technology of the transceiver, all communications are done using a wireless channel. As a result, the information flow can be easily accessed by anyone in the vicinity. All packets are then unprotected against any kind of communication attack.

It is indispensable to provide basic security primitives to the sensor nodes in order to give a minimal protection to the information flow and a foundation to create secure protocols. Those security primitives are *Symmetric Key Cryptography* (SKC), *hash primitives*, and *Public Key Cryptography* (PKC). Since sensor nodes are highly constrained in terms of resources, implementing the security primitives in an efficient way (using less energy, computational time and memory space) without sacrificing the strength of their security

properties is one of the major challenges in this area, a challenge that most of the state-of-the-art have managed to achieve.

SKC primitives use the same secret key for both encryption and decryption. Instances of these primitives are able to provide confidentiality to a certain information flow, given that the origin and the destination of the data share the same secret key. They can also provide integrity and authentication if a certain mode of operation is used. These algorithms are usually not very complex, and they can be implemented easily in resource-constrained devices.

The suitability of SKC primitives on sensor nodes was analyzed mainly by Ganesan et al. in 2003 [Ganesan2003], and by Wei Law et al. [Wei2006] and Jun et al. [JunChoi2006] in 2006. The feasibility of the software implementations of SKC algorithms (e.g. RC4, RC5, Skipjack, IDEA, AES) was evaluated, concluding that the most effective algorithms such as RC4 and Skipjack have an overhead (energy, bandwidth, latency, and memory consumption) less than 10%. Other algorithms such as AES impose a higher penalty over the resources of the node (around 20%), but they are still suitable for practical use.

Regarding hardware implementations, nodes with radio chips conforming to the 802.15.4 standard [IEEE802.15.4] do offer a suite of AES-based cryptography operations, which offer encryption, authentication and replay protection. However, not all the suites are actually secure [Sastry2004]. Also, most hardware implementations of the standard do not offer all its functionality. Therefore, system and application designers must take special care on using the standard, providing a workaround in certain implementations for taking advantage of the hardware capabilities.

Cryptographic hash functions or hash primitives are utilized in order to compress a set of data of variable length into a set of bits of fixed length. The result is a "digital fingerprint" of the data, identified as a hash value. A cryptographic hash function must satisfy two properties: i) given a hash value h, it should be hard to find a message m such that hash(m) = h, and ii) it should be hard to find two different messages m_1 and m_2 such that hash(m_1) = hash(m_2).

Hash functions are usually used for assuring the integrity of the information flow, providing a unique fingerprint for every packet in the form of a *Message Authentication Code* (MAC). However, hash functions are resource-heavier compared to SKC primitives: the studies by Ganesan et. al. [Ganesan2003] showed that a hash function is around ten times slower than a SKC algorithm in terms of speed. For that reason, MAC is usually computed using SKC with a special mode of operation called CBC-MAC.

Finally, PKC, also known as asymmetric cryptography, is a form of cryptography that uses two keys: a key called private key, which has to be kept private, and another key named public key, which is publicly known. Any operation done with the private key can only be reversed with the public key, and vice versa. This nice property makes all PKC-based algorithms useful for authentication purposes. Still, the computational cost of calculating their underlying operations had hindered its application in highly-constrained devices, such as sensor nodes.

One of the most promising PKC primitives in the field of WSN security is *Elliptic Curve Cryptography* (ECC), due to the small size of the keys, the memory and energy savings, and the simplicity of its underlying operation, the scalar point multiplication. Thanks to these advantages, ECC has been implemented both in software [Liu2007] [Wang2006] and in hardware [Batina2006]. Software implementations have an acceptable signing and verifying time, smaller than 2 seconds, whereas hardware prototypes achieve a speed of 115ms.

ECC is not the only PKC algorithm that has been implemented for sensor nodes: there are also other hardware prototypes that provide other asymmetric cryptography primitives, such as NTRU [Gaubatz2005], Rabin [Gaubatz2005], and Multivariate [Yang2006]. The primary disadvantage of these primitives, compared with ECC, is their key and signature size. Nevertheless, these prototypes have certain advantages that can make them useful in certain scenarios. For example, most operations done on a NTRU chip are faster than any of the ECC prototypes. Also, the encryption and verification time of Rabin is much smaller, whereas the signature time is optimal when using the Multivariate prototypes.

5. Support for Security Primitives

5.1. Key Management System

The communication channels between any pair of devices inside WSN must be protected to avoid attacks from external parties. Such protection, in terms of confidentiality, integrity and authentication, is provided by the security primitives introduced in the previous section. Nevertheless, those primitives need to store a set of security credentials (i.e. secret keys) inside every node in order to create and share a secure channel. The task of generating and distributing those keys has to be done by a global *Key Management System* (KMS).

There are three basic factors that every key management system for WSN should adequately fulfil: *key storage*, *key distribution*, and *key maintenance*.

- Key storage policies indicate the number of keys that a sensor node needs to store in order to open a secure channel with other peers.
- Key distribution protocols define how the keys are issued to the sensor nodes. A node can receive its keys before the initial deployment of the network or create its keys after the deployment using preloaded information.
- Key maintenance procedures specify how a node can be included into or erased from the network, receiving or nullifying a set of keys in the process.

There are two extreme design cases for a key management system: *global keying* and *pairwise keying*. In global keying, a single key is created for the entire network, and all the secure communications must be encrypted with that key. In the other case, pairwise keying, a node must store a key for every other node inside the network, thus every pair of nodes will have a particular secure channel.

Neither of the cases is feasible in a real scenario. In global keying, any tampered node will reveal the global secret key, thus opening all the communications of the network to any potential attacker. Also, pairwise keying is not a scalable solution due to the memory constraints of the nodes. Therefore, security researchers have been trying to develop more optimal solutions that are scalable and resilient, amongst other properties. The existing systems can be classified into four frameworks: *"key pool" framework*, *mathematical framework*, *negotiation framework*, and *public key framework* [Alcaraz2006].

"Key Pool" Framework. The "key pool" paradigm is the pillar behind one of the most important frameworks to date. The main idea behind this framework is quite simple: the network designer creates a "key pool", a large set of pre-calculated secret keys, and before the network deployment every node in the network is assigned with a unique "key chain", i.e. a small subset of keys from the "key pool" (*Key Pre-distribution Phase*). After the deployment, the nodes can interchange the identification numbers of the keys from their "key chains", trying to find a common key (*Shared-key Discovery Phase*). If they find that they share a common key, that key (or a derivation from that original key) will be used as the pairwise key between the two nodes. Otherwise, if two nodes want to communicate but they do not share any key, they will try to find a "key path" between them in order to negotiate a pairwise key (*Path-key Establishment Phase*).

Mathematical Framework. Some KMSs use mathematical concepts belonging to the fields of Linear Algebra (Blom Scheme), Combinatorics (Generalized Quadrangles) and Algebraic Geometry (Bivariate Polynomials) for calculating the pairwise keys of the nodes, by combining certain information after the deployment phase or by specially crafting a "key pool". All these KMSs belong to the Mathematical Framework, since they share some common properties. On the one hand, their connectivity is very high, since by design almost all nodes can have a pairwise key with any node in the network. Also, the communication overhead is low, since the nodes need to share only small bits of information. On the other hand, the memory footprint is usually high because all nodes need to store certain mathematical information (e.g. rows of a matrix), and the network resilience and the scalability are highly tied to the design of the underlying mechanisms.

Negotiation Framework. It can be possible to let nodes negotiate their keys with their closer neighbours just after the deployment of the network, requiring none or few steps in the pre-distribution phase. This self-configuration approach is usually seen as non-feasible because any potential adversary can attack the network at this point and obtain the messages that contain the secret keys, effectively thwarting the overall security of the network. However, it can be assumed that in several scenarios the nodes and the network itself are not endangered in the first steps after the deployment, because the resources that an attacker has to spend in order to break the security of the network do not justify the benefits obtained from that attack. All the schemes that create their keys like this can be considered part of the Negotiation Framework.

Public Key Framework. One of the multiple uses of asymmetric cryptography has been to securely bootstrap the pairwise key of two nodes over a public communication channel. Two nodes just need to interchange their public keys and some information (e.g. using the ECDH-ECDSA handshake protocol and Elliptic Curve Cryptography) that will be used to create the pairwise secret key and its associated values like initialization vectors. If the public keys are certified by a trusted third party, both peers will also be authenticated.

5.2. Public Key Infrastructure

Public Key Cryptography, although extremely resource-intensive for highly constrained devices, is a viable alternative that can elegantly provide a solution for some of the existing security problems in WSN such as key distribution. Nevertheless, the use of these cryptographic techniques is not enough for providing essential services such as digital signatures. For example, the possession of a public/private key pair is not enough to assure the identity of a certain node. Therefore, it is necessary to have a *Public Key Infrastructure* (PKI) that can establish a trusted identity, amongst other things.

It is not trivial to apply a PKI to WSN which is highly decentralized by nature. However, there is a central system, the base station, that takes the role of initializing the nodes of the network and interacting with the data provided by all these nodes. Therefore, it is clear that the base station can be considered as a *Certification Authority* (CA). Moreover, the base station can also take the role of *Registration Authority* (RA), since it is in charge of assigning the identity of all the nodes before the deployment of the network.

The basic functionality of a PKI, that is, registration, initialization, key generation, certification, and certificate retrieval, can be performed as follows in WSN. The base station generates the public/private key pair of a sensor node, assigns a unique identification to it, and creates the digital certificate that links the unique identification with its public key. The base station also initializes the contents of the sensor node (such as configuration data and internal programming), including the certificate of the node and the certificate of the root CA (i.e. the base station itself). Later, when a node retrieves the certificate of one of its neighbours, it will be able to check the validity of that certificate using the root CA certificate.

6. Core Protocols

It is easy to understand that protecting the communication channel between two nodes does not entirely guarantee the security of WSN. The core protocols of the network, that is, the minimal set of protocols required to provide services must also be secure in order to withstand errors coming from faulty nodes, as well as attacks initiated by malicious elements (from outside and inside the network). They include those protocols for routing, data aggregation, and time synchronization.

6.1. Routing

Designing routing algorithms is a challenging area [AlKaraki2004]. All the nodes inside WSN should be reachable (*Connectivity*) while covering the maximum possible area of environment using their sensors (*Coverage*), even when the nodes inside the network start to fail due to energy issues or other problems (*Fault Tolerance*). The algorithm should also work with any network size and node density (*Scalability*) and provide a certain quality of service. At the same time, designers must try to lower the memory usage and energy consumption of the algorithms.

Security is another factor that cannot be ignored in the design of routing algorithms. Any potential adversary has a wide range of attacks at his disposition [Karlof2004] to manipulate the routing subsystem and take control over the routes, resulting in eavesdropped, altered, spoofed or discarded packets. He can direct traffic over his own nodes by advertising them as nodes with better (real or fake) connectivity or speed. He can alter the routing control packets on his own benefit, and also spoof the identity of the nodes using a Sybil attack [Newsome2004].

The key infrastructure may help in the protection against routing attacks by authenticating nodes and protecting the confidentiality and integrity of the packets, but it is not enough to protect the routing infrastructure. Any adversary can take control of a set of legitimate nodes of the network and modify or discard any control messages on his own benefit. He can also attack the network or certain sections with a denial of service attack, jamming the communication signal or colliding certain control packets. Therefore, it is essential to make the routing algorithm robust against such attacks.

6.2. Data Aggregation

Inside a sensor network, the nodes generate an immense amount of raw data product of their measurements. In most cases these data are needed at the base station, thus there is a great cost, in terms of energy consumption and bandwidth usage, on transporting all the data from the nodes to the base station. However, since nodes are physically near each other, the data likely have some type of redundancy. The role of aggregation is to exploit this redundancy by collecting the data from a certain region and summarizing it into one report, hence decreasing the number of packets sent to the base station.

Aggregated data can be easily attacked by a malicious adversary, even if the communications are protected against any data injection attack or data integrity attack. If an aggregator node is being controlled by an adversary, it can easily ignore the data received from its neighbours and create a false report. Trusted aggregators can still receive false data from faulty nodes or from nodes being controlled by an adversary.

By using strong aggregation functions that are resilient against internal attacks, it is possible to defend the network against false data coming from malicious or faulty nodes. Borrowing ideas from the field of robust statistics, the author in [Wagner2004] developed a theoretical framework for analyzing the resilience of a number of natural aggregation functions (e.g., demonstrating that the minimum, maximum, sum and average functions are insecure) and proposed some tools (such as trimming) for improving the resilience of the aggregation functions, although aggregation is supposed to be carried out in the base station.

There are also solutions that discover whether the reports sent by a malicious aggregator are forged or not. One approach queries the aggregator itself about the data used to create the report. In [Przydatek2003], the aggregator must create a proof of its neighbors' data using a Merkle hash tree, which will be used in a negotiation with the base station to demonstrate the authenticity of the data used to construct the report.

Other approaches take advantage of the density of sensor networks by using the nodes in the neighbourhood of the aggregator as witnesses. They will make the same computation as the aggregator and should obtain a similar result, since they are in the same neighbourhood and measuring the same events. As an example, in [Du2003] nodes can create a proof of result in form of a MAC of their computations by using a shared secret key with the base station, and the aggregator node must send both the aggregated data and a set of the proofs to the base station.

Finally, it is also possible to filter the packets containing the report and the proofs in their way to the base station, hence decreasing the amount of traffic created by false aggregations. In [Ye2004], the proofs created by the witnesses use a key from a key pool, and the aggregator node compresses them all using a Bloom filter in order to decrease the bandwidth usage. On the way, all nodes with a common key from the key pool can recreate a proof and check if it is inside the Bloom filter.

6.3. Time Synchronization

The major task of a sensor network is to collect data from a certain physical environment. But the data itself has, in most cases, no importance for the user if not linked to the time when it was collected. Although sensor nodes are able to know the local time, i.e. the time that has passed after the moment they were born, it is necessary to create a set of protocols that allow the maintenance of a global time. Such protocols are also essential for synchronizing the clocks of the nodes, avoiding problems such as clock drifts. If these issues are not dealt with, services such as tracking and localization may provide erroneous results.

A time synchronization protocol must comply with certain design principles. The use of high-demanding energy devices such as GPS should be limited to powerful nodes (*Energy Efficiency*), the number of transmissions should be kept to a bare minimum (*Transmission Efficiency*), factors such as the latency error and the jitter should be taken into account (*End-to-End Latency*), and the protocols should deal with message loss and problems in message delivery (*Fault Tolerance*) [Sundararaman2005].

Security, while not a design principle but itself, is a vital property that has to be taken into account: any protocol that incorrectly updates the clock of a single node can easily thwart the behaviour of the entire system. Attacks against time synchronization protocols are usually performed by compromised nodes, which provide erroneous clock values or cheat in the negotiation processes. It is also possible for an external node to try to interfere in these protocols if the information is broadcasted through the network.

Possible countermeasures against these problems [Manzo2005] include using authenticated broadcast mechanisms to avoid the existence of malicious external entities or compromised nodes changing the clock information. Also, it is possible to take advantage of the redundancy of the network, allowing many nodes to serve as the root for the global time in case no base station is present or near, or letting one node to get the time information from many sources discouraging the existence of a malicious entity with an abnormal clock drift. At last, the nodes can use approximations to get an upper bound on the error that could be induced by an adversary, and use such approximation in case there is no reliable information around.

7. Services

7.1. Intrusion Detection

An intrusion can be defined as a set of actions (i.e. attacks), either external or internal to a certain system, that can lead to an unauthorized access or alteration of the system. The mission of an *Intrusion Detection System* (IDS) (cf. [Rebecca2000]) is to monitor a certain system in order to detect possible intrusions in the network, alert users after intrusions had been detected and, finally, reconfigure the network provided that is possible. In a WSN environment, such an IDS allows sensor nodes to monitor themselves and react to the different situations in their environment, providing an infrastructure that protects their normal operations and detects and reacts to any possible attacks against network services.

The research of IDS in WSN has not advanced significantly, possibly because the concept of “intrusion” is not clear in a wireless sensor network context. The reason may be the lack of working applications that could give a more exact idea of the real-life attacks that WSN has to face. Despite these problems, at present it is clear what are the challenges that such a system must overcome in order to support auditing and incident registration.

IDS must decide carefully where to locate the detection agents in WSN, due to the distributed nature of the network and the constraints inherent to the nodes.

A probable solution is to have an agent inside every node, but limit its capabilities when dealing with external sources of data. Choosing an adequate set of lightweight detection procedures, like automata, packet analysis, and health monitoring systems, is also of importance. Finally, agents should be able to collaborate in the detection processes, and send the alerts to the base station to inform the user of the system about any abnormal behaviour.

7.2. Trust Management

The concept of trust derives from sociological or psychological environments. Trust is an essential factor in any kind of network, social or computer networks. It becomes an important factor for members of the network to deal with uncertainty about the future actions of other participants. Thus, trust becomes especially important in distributed systems and Internet transactions. Even though there is no consensus on the definition of trust, it is usually defined in terms of a *trustor* (the subject that trusts an entity or a service) and a *trustee* (the entity that is trusted). The term trust has been used with a variety of meanings [McKCher1996]. The Oxford Reference Dictionary defines trust as "*the firm belief in the reliability or truth or strength of an entity*".

Trust management, which models the trust on the behaviour of the network elements, is useful for a sensor network environment. Not only it can help the nodes of the network to self-configure themselves against any change in their neighbourhood, but it can also assist and/or take advantage of the other security protocols. For example, a key management system can use the information provided by the trust system for the purpose of revoking keys. Also, all the protocols of the network can benefit from the existence of a trust management system, either by using the output of the trust system as an assistant in their decision-making process, or by providing useful inputs for the trust system that could be of use for any other protocol or service.

Some aspects should be considered for creating a satisfactory trust system. One is the initialization of the trust model. In most cases, initial trust is not a very important value, since all nodes are usually initialized in a controlled environment. Another is the interpretation of the events that occur during the lifetime of the network [GaneSriv2004]. All events are not equally important, and the existence of a certain event does not indicate that the node is going to misbehave in all its activities. The evolution and density of the events are also of importance. A node having continuous symptoms of malicious behaviour should be completely untrusted. All decisions taken by the nodes regarding untrusted neighbours have to be notified to the base station, although the base station can track its own trust readings. Finally, due to the constraints of the nodes, it is imperative to make the trust and reputation models as lightweight as possible.

7.3. Privacy Protection

Privacy is a security property that is very important in certain scenarios. For example, in a battlefield, it would be necessary to hide the location and identity of the base station and the nodes that generated the information. In contrast, in an earthquake rescue situation locating the source nodes (if the nodes are worn by, for example, dogs) is an absolute must. Note that this property can transcend beyond the technological dimension and affect its social environment, since a sensor network could be used as a surveillance tool to collect data about the behaviour of human beings.

There are three types of privacy threats [Ozturk2004]. If an adversary can determine the meaning of a communication exchange because of the existence of a message and the context of the situation, there is a *content privacy threat*. If an adversary is able to deduce the identities of the nodes involved in a communication, there is an *identity privacy threat*. And if the adversary is able to infer the physical location of a communication entity or to approximate the relative distance to that entity, there is a *location privacy threat*.

Privacy can be addressed in different levels of the network stack and at different points of the information flow. The privacy protection mechanisms are categorised into privacy sensitive information gathering schemes, controlled information disclosure approaches, and mechanisms for the protection of the communications context.

8. Book Outline

The above introduction gives a brief overview of the threats and security mechanisms in WSN. The remainder of this book is structured into the following chapters which will explore the related topics in detail.

Chapter 2 investigates how wireless sensor networks can be attacked in practice. It then develops a generic adversary model that allows for classification of adversaries according to two dimensions of power: presence and intervention, and also provides a framework for realistic security analysis in wireless sensor networks.

Chapter 3 highlights the importance of symmetric cryptographic primitives for providing security in wireless sensor networks. It then outlines the basic goals and primitives and gives a comprehensive overview on the modes of operation. It also provides an extensive survey of the implementation options of *Advanced Encryption Standard* (AES), and discusses the state-of-the-art cryptographic support in today's WSN products.

Chapter 4 describes public-key cryptography based solutions for the security services like encryption, digital signatures, authentication and key establishment. It then evaluates the power consumption of these algorithms and investigates whether they can be used within the power constrained sensor nodes. It also takes a look at hardware implementations of ECC based algorithms and shows that the ECC algorithms that minimize the memory requirements and that require the fewest field operations seem to be suitable for sensor network applications.

Chapter 5 presents a comparative survey of recent key management (key distribution, discovery, establishment and update) solutions for wireless sensor networks. It considers both distributed and hierarchical sensor network architectures where unicast, multicast and broadcast types of communications take place, and presents probabilistic, deterministic and hybrid key management solutions. It then determines a set of metrics to quantify their security properties and resource usage such as processing, storage and communication overheads. It also provides a taxonomy of solutions, and identifies trade-offs in these schemes to conclude that there is no one-size-fits-all solution.

Chapter 6 elaborates on the need for security frameworks at the link-layer and describes what services they provide to the upper layers. It reviews the proposed frameworks in the bibliography and discusses about their pros and cons. Then it presents in more detail the design and implementation of one of the frameworks to show what issues arise in such a process and how they can be solved. Some of these features include providing acceptable resistance against node capture attacks and replay attacks, as well as the run-time composition of security services in a completely transparent way.

Chapter 7 studies how WSN routing protocols can be secured. It describes the adversary model, the objectives of attacks against routing, as well as the different attack methods that may be used in wireless sensor networks. It also describes various countermeasures that can be used to secure the routing protocols in WSN, which include link layer security measures, secure neighbor discovery techniques, authenticated broadcast algorithms, and multi-path routing techniques. Finally, it illustrates the application of some of these countermeasures by presenting and explaining the operation of some secured WSN routing protocols.

Chapter 8 discusses the security vulnerabilities of data aggregation systems which is an essential component in WSN to aggregate the data collected from individual nodes at a base station or even perform in-network aggregation at intermediate nodes enroute to the base station to reduce energy consumption. It

also presents a survey of robust and secure aggregation protocols that are resilient to false data injection attacks.

Chapter 9 addresses privacy issues in wireless sensor networks, by identifying the requirements for privacy preserving deployments, analyzing the challenges faced when designing them, and discussing the main solutions that have been proposed. It also demonstrates that privacy can be addressed in different levels of the network stack and at different points of the information flow, and different approaches can satisfy the diverse privacy aspects and fulfill the complete spectrum of privacy needs in WSN.

Chapter 10 outlines the unique challenges of intrusion detection systems in WSN, and provides a survey of solutions proposed in the research literature which can be classified as four distinct categories: IDS using routing protocols, IDS based on neighbor monitoring, IDS based on innovative techniques and IDS based on fault tolerance. It also reveals that most of the proposed solutions and results are theoretical or based on simulations, and that real-world experience in preventing, detecting, or responding to sensor network attacks has yet to be published.

Chapter 11 investigates sensor network node identity relative to the functional properties reflected in executing code found on small computing sensors. It describes numerous identity properties and identity verification mechanisms. It also shows how wireless sensor networks can establish an important identity property of ensuring code integrity through the process of remote attestation.

Chapter 12 reviews the functionality of the less resource-demanding cryptographic algorithms. It also analyzes the influence on hardware implementation of these primitives over the behavior of the node, and examines their theoretical suitability to the existent resource-constrained sensor node hardware.

References

[AES2002] J. Daemen, V. Rijmen. “The Design of Rijndael”. Springer, ISBN 3-540-42580-2, 2002.

[Alarifi2006] A. Alarifi, W. Du. “Diversifying Sensor Nodes to Improve Resilience against Node Compromise”. 2006 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'06), Alexandria, USA, October 2006.

[Alcaraz2006] C. Alcaraz, R. Roman. “Applying Key Infrastructures for Sensor Networks in CIP/CIIP Scenarios”. 2006 International Workshop on Critical Information Infrastructures Security (CRITIS'06), Samos, Greece, August-September 2006.

[AlKaraki2004] J. N. Al-Karaki, A. E. Kamal. “Routing Techniques in Wireless Sensor Networks: A Survey”. IEEE Wireless Communications, vol. 11, no. 6, pp. 6-28, December 2004.

[Batina2006] L. Batina, N. Mentens, K. Sakiyama, B. Preneel, I. Verbauwhede. “Low-Cost Elliptic Curve Cryptography for Wireless Sensor Networks”. 2006 European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS'06), Hamburg, Germany, September 2006.

[Becher2006] A. Becher, Z. Benenson, M. Dornseif. “Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks”. 2006 International Conference on Security in Pervasive Computing (SPC'06), York, UK, April 2006.

[Beck2004] R. Beckwith, D. Teibel, P. Bowen. “Report from the Field: Results from an Agricultural Wireless Sensor Network”. 2004 IEEE Workshop on Embedded Networked Sensors, Tampa, USA, November 2004.

[CodeBlue2004] David Malan, Thaddeus Fulford-Jones, Matt Welsh, Steve Moulton. “CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care”. 2004 International Workshop on Wearable and Implantable Body Sensor Networks, London, UK, April 2004.

[Du2003] W. Du, J. Deng, Y. S. Han, P. K. Varshney. “A Witness-Based Approach for Data Fusion Assurance in Wireless Sensor Networks”. GLOBECOM'03, San Francisco, USA, December 2003.

[Ganesan2003] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, M. Sichitiu. “Analyzing and Modeling Encryption Overhead for Sensor Network Nodes”. 2003 ACM International Conference on Wireless Sensor Networks and Applications (WSNA'03), San Diego, USA, September 2003.

[GaneSriv2004] S. Ganeriwal, M. B. Srivastava. “Reputation-Based Framework for High Integrity Sensor Networks”. 2004 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04), Washington DC, USA, October 2004.

[Gaubatz2005] G. Gaubatz, J.-P. Kaps, E. Öztürk, B. Sunar. “State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks”. 2005 IEEE International Workshop on Pervasive Computing and Communication Security (PerSec'05), Hawaii, USA, March 2005.

[IEEE802.15.4] IEEE Standard, 802.15.4-2006. “Wireless Medium Access Control and Physical Layer Specifications for Low-rate Wireless Personal Area Networks”. 2006, ISBN 0-7381-4997-7.

[JunChoi2006] K. Jun Choi, J.-I. Song. “Investigation of Feasible Cryptographic Algorithms for Wireless Sensor Networks”. 2006 International Conference on Advanced Communication Technology (ICACT'06), Phoenix Park, Korea, February 2006.

[Karlof2003] C. Karlof, D. Wagner. “Secure Routing in Wireless Sensor Networks: Attacks and Countermeasure”. Ad-Hoc Networks, vol. 1, no. 2-3, pp. 293-315, Elsevier, September 2003.

[Liu2007] A. Liu, P. Kampanakis, P. Ning. “TinyECC: Elliptic Curve Cryptography for Sensor Networks (Version 0.3)”. http://discovery.csc.ncsu.edu/software/TinyECC/, February 2007.

[Manzo2005] M. Manzo, T. Roosta, S. Sastry. “Time Synchronization Attacks in Sensor Networks”. 2005 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'05), Alexandria, USA, November 2005.

[McKCher96] D.H. McKnight, N. L. Chervany. “The Meanings of Trust”. MISRC Working Paper Series, University of Minnesota, Management Information Systems Research Center. 1996.

[Newsome2004] J. Newsome, E. Shi, D. Song, A Perrig. “The Sybil Attack in Sensor Networks: Analysis & Defenses”. 2004 IEEE International Workshop on Information Processing in Sensor Networks (IPSN'04), Berkeley, USA, April 2004.

[Ozturk2004] C. Ozturk, Y. Zhang, W. Trappe, M. Ott. “Source-Location Privacy for Networks of Energy-Constrained Sensors”. 2004 IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (WSTFEUS'04), Vienna, Austria, May 2004.

[Przydatek2003] B. Przydatek, D. Song, A. Perrig. “SIA: Secure Information Aggregation in Sensor Networks”. 2003 International Conference on Embedded Networked Sensor Systems (SenSys'03), Los Angeles, USA, November 2003.

[Rebeca2000] R. Bace. “Intrusion Detection”. New Riders, 2000. ISBN 1-57870-185-6.

[Sastri2004] N. Sastry, D. Wagner. “Security Considerations for IEEE 802.15.4 Networks”. 2004 ACM Workshop on Wireless Security (WiSe'04), Philadelphia, USA, October 2004.

[Seshadri2004] A. Seshadri, A. Perrig, L. van Doorn, P. Khosla. “SWATT: SoftWare-based ATTestation for Embedded Devices”. 2004 IEEE Symposium on Security and Privacy, Oakland, USA, May 2004.

[Shaneck2005] M. Shaneck, K. Mahadevan, V. Kher, Y. Kim. “Remote Software-Based Attestation for Wireless Sensors”. 2005 European Workshop on Security in Ad-hoc and Sensor Networks (ESAS'05), Visegrad, Hungary, July 2005.

[Sundararaman2005] B. Sundararaman, U. Buy, A. D. Kshemkalyani. “Clock Synchronization for Wireless Sensor Networks: a Survey”. Ad-Hoc Networks, vol. 3, no. 3, pp. 281-323, Elsevier, May 2005.

[Wagner2004] D. Wagner. “Resilient Aggregation in Sensor Networks”. 2004 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04), Washington DC, USA, October 2004.

[Wang2006] H. Wang, Q. Li. “Efficient Implementation of Public Key Cryptosystems on MICAz and TelosB Motes”. Technical Report WM-CS-2006-07, College of William & Mary, October 2006.

[Wei2006] Y. W. Law, J. Doumen, P. Hartel. “Survey and Benchmark of Block Ciphers for Wireless Sensor Networks”. ACM Transactions on Sensor Networks, vol. 2, no. 1, pp. 65-93, February 2006.

[WINES2007] Wired and Wireless Intelligent Networked Systems (WINES) - Smart Infrastructure Project. http://www.winesinfrastructure.org/

[Yang2006] B.-Y. Yang, C.-M. Cheng, B.-R. Chen, J.-M. Chen. “Implementing Minimized Multivariate Public-Key Cryptosystems on Low-Resource Embedded Systems”. 2006 International Conference on Security in Pervasive Computing (SPC'06), York, UK, April 2006.

[Ye2004] F. Ye, H. Luo, S. Lu, L. Zhang. “Statistical En-route Filtering of Injected False Data in Sensor Networks”. IEEE INFOCOM'04, Hong Kong, China, March 2004.

Wireless Sensor Network Security
J. Lopez and J. Zhou (Eds.)
IOS Press, 2008

Vulnerabilities and Attacks in Wireless Sensor Networks

Zinaida BENENSON [a,1], Peter M. CHOLEWINSKI [b], Felix C. FREILING [a]
[a] *Laboratory for Dependable Distributed Systems, University of Mannheim, 68131 Mannheim, Germany*
[b] *SAP - Research and Breakthrough Innovation, Germany*

Abstract. We investigate how wireless sensor networks can be attacked in practice. From this we develop a generic adversary model that allows to classify adversaries according to two dimensions of power: presence and intervention. Thus, we provide a framework for realistic security analysis in wireless sensor networks.

Keywords. adversary models, attacks, threat analysis, security evaluation

1. Introduction

Sensor networks provide unique opportunities of interaction between computer systems and their environment. Their deployment can be described at high level as follows: The sensor nodes measure environmental characteristics which are then processed in order to detect events. Upon event detection, some actions are triggered. This very general description applies to extremely security-critical military applications as well as to such benign ones (in terms of security needs) as habitat monitoring.

Considering the Internet as an example, it is extremely difficult to add security to systems which were originally designed without security in mind. The goal of security is to "protect right things in a right way" [1]. Thus, careful analysis is needed concerning which things to protect against which threats and how to protect them. Of course, this analysis is only possible in context of a particular class of applications. However, it makes much sense to provide a set of abstract security requirements and a set of generic attacker models, i.e., a *framework for security analysis in wireless sensor networks*, which can be refined for particular applications.

In this chapter, we present such a framework. It provides concepts to clarify two important aspects of the security analysis in wireless sensor networks:

1. What should be protected? Here we offer a set of generic classes of requirements which can be used to structure and refine a set of concrete security

[1]Zinaida Benenson was supported by Landesstiftung Baden Württemberg as part of Project "ZeuS – Zuverlässige Informationsbereitstellung in energiebewussten ubiquitären Systemen".

requirements. We highlight the main differences between security requirements in classical systems and security requirements in wireless sensor networks.
2. Against what are we protecting the system? Here we offer a set of generic attacker models which can be used to choose and refine particular attacker models for individual systems.

Overall, attacker models in conjunction with security requirements determine the means to achieve security.

In practice it is very important to formulate *realistic* security requirements and *realistic* attacker models. Such choices guarantee that precious resources of wireless sensor networks are invested efficiently. It is therefore useful to evaluate the practicality of certain attacker models. One metric to measure practicality is to evaluate the *effort* an attacker has to invest to perform certain attacks. We contribute to this area by reporting on a number of experiments in which we attacked *real* sensor node hardware.

This chapter is structured as follows: We first give an overview of security goals in sensor networks, i.e., we approach the question "what to protect" (Section 2). We then report on experiments in attacking wireless sensor networks in Section 3. Building on these experiences we develop a generic set of attacker models in Section 4), i.e., we approach the question "against whom to protect". Finally, we briefly discuss protection mechanisms in Section 5, i.e., we approach the question "how to protect". We outline open problems and summarize in Section 6.

2. Security Goals in Sensor Networks

A sensor network can be considered as a highly distributed database. Security goals for distributed databases are very well studied: The data should be accessible only to authorized users (Confidentiality), the data should be genuine (Integrity), and the data should be always available on the request of an authorized user (Availability). All these requirements also apply to sensor networks and their users. Here, the distributed database, as well as the sensor network, are considered as a single entity from the user's point of view. Therefore, we call these security issues *outside security*. To outside security belong, e.g., query processing [22, 35, 46], access control [9] and large-scale anti-jamming services [47].

The internal organization of a distributed database and of a sensor network are quite different. Outside security, as well as all other types of interactions between the user and the corresponding system, is based on the interactions between the internal system components (servers or sensor nodes, respectively). We call security issues for such interactions *inside security*. In sensor networks, inside security realizes robust, confidential and authenticated communication between individual nodes [24, 44]. This also includes in-network processing [18, 48], data aggregation [11, 49], routing [17, 25] and in-network data storage [7, 19].

Aside from necessitating the distinction between inside and outside security, sensor networks differ from conventional distributed databases in other obvious ways: A distributed database consists of a small number of powerful servers, which are well protected from physical capture and from network attacks. The servers

use resource-demanding data replication and cryptographic protocols for inside and outside security. In contrast, a sensor network consists of a large number of resource-constrained, physically unprotected sensor nodes which operate unattended. Therefore, security measures for distributed databases cannot be directly applied to sensor networks. So even if a sensor network can be considered as a distributed system (e.g., as an ad hoc network), sensor networks have some additional constraints which make security mechanisms for distributed systems inapplicable. Apart from the obvious resource constraints, single sensor nodes are relatively unimportant for the properties of the whole system – at least, if the inherent redundancy of sensor networks is utilized in their design.

To summarize, security goals in sensor networks are similar to security goals in distributed databases (outside security) and distributed systems (inside security). So these can be taken as an orientation. While requirements are similar, many standard mechanisms to *implement* security (e.g., public key infrastructures or agreement protocols) are not applicable because they require too many resources or do not scale to hundreds or thousands of nodes. This is the dilemma of sensor networks and forces security mechanisms in wireless sensor networks to spend the "right" amount of effort in the "right" places by exploiting the natural features of sensor networks: inherent redundancy and broadcast communication.

In the remainder of this chapter, we will take a first step towards developing realistic security mechanisms. We evaluate real attacks in practice and from this evaluation derive a fine-grained set of abstract attacker assumptions.

3. Attacking Wireless Sensor Networks

In this section we take the viewpoint of an adversary who wishes to violate one of the security requirements of a WSN which were described in Section 2. From this viewpoint, a WSN is a very interesting target because if offers a large attack surface and an interesting playground for creative attack ideas.

Of course, the many possibilties to attack WSNs include all the classical techniques known from classical system security. An adversary can eavesdrop on the communication, perform traffic analysis of the observed network behavior, he can replay old messages or inject false messages into the network. Possible are also other types of attacks that aim at violating Availability (denial-of-service attacks) like jamming the wireless channel. In WSNs these attacks can be particularly important as they can cause rapid battery draining and effectively disable individual sensor nodes or entire parts of a WSN.

While there are many techniques known from other areas of security, the ability of an attacker to access (and eventually change) the internal state of a sensor node seems particularly characteristic for sensor networks. This type of attack is called *node capture* in the literature [34]. Depending on the WSN architecture, node capture attacks can have significant impact. Thus, most existing routing schemes for WSNs can be substantially influenced even through capture of a minute portion of the network [25]. In the TinySec mechanism [24], which enables secure and authenticated communication between the sensor nodes by means of a network-wide shared master key, capture of a single sensor node suf-

fices to give the adversary unrestricted access to the WSN. Most current security mechanisms for WSNs take node capture into account. It is usually assumed that node capture is "easy". Thus, some security mechanisms are verified with respect to being able to resist capture of 100 and more sensor nodes out of 10,000 [12,23].

In this section, we determine the actual cost and effort needed to attack currently available sensor nodes. We especially concentrate on *physical attacks* which require direct physical access to the sensor node hardware. As sensor nodes operate unattended and cannot be made tamper proof because they should be as cheap as possible, this scenario is more likely than in most other computing environments. Physical attacks of some form are usually considered a prerequisite to perform node capture.

Another possibility to gain access to the internal state of a sensor node is to exploit some bugs in software running on the sensor nodes or on the base stations. These attacks directly correspond to well-known techniques from classical software security. If a software vulnerability is identified by the attacker, an attack can be easily automated and can be mounted on a very large number of nodes in a very short amount of time. Such attacks are relatively well-understood in software security [21]: Possible countermeasures include a heterogenous network design and standard methods from software engineering [20, 21, 27, 33, 43]. As these attacks are not characteristic for WSNs, we do not consider such attacks in detail. This area is however an interesting direction for future work.

In the following, we first give some background on physical attacks on sensor node hardware. Then we report on the effort needed to attack some current sensor nodes. Where appropriate we also discuss countermeasures that increase the effort to mount a successful attack. Overall, this section gives insight into *practical* attacks on WSNs which motivate the abstract attacker models which follow in Section 4.

3.1. Background on Physical Attacks on Sensor Nodes

3.1.1. Tampering vs. Physical Attacks

Tampering attacks on embedded systems, that is, on microcontrollers and smart cards, have been intensively studied, see for example work by Anderson et al. [1,3, 37]. The term "tampering" is well accepted in the research community to designate attacks on components that involve modification of the internal structure of a single chip. At the same time, there are also conceivable attacks on sensor node hardware which are carried out on the circuit board level and therefore, do not really fit this accepted usage. In order to avoid this terminology problem, we use the term "physical attack" to refer to all attacks requiring direct physical access to the sensor node.

Skorobogatov [36] describes in depth tampering attacks on microcontrollers, and classifies them in the three categories of *invasive*, *semi-invasive*, and *non-invasive* attacks. Invasive attacks are those which require access to a chip's internals, and they typically need expensive equipment used in semiconductor manufacturing and testing, as well as a preparation of the chip before the attack can begin. Semi-invasive attacks require much cheaper equipment and less time than the invasive attacks, while non-invasive attacks are the easiest.

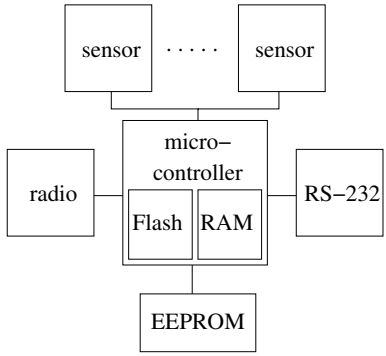


Figure 1. General schematic view of sensor node hardware.

All of these attacks, including the so-called low-cost attacks, if applied to sensor nodes, would require that the nodes be removed from the deployment area and taken to a laboratory. Even if in some cases, the laboratory could be moved into the deployment area in a vehicle, all attacks would require at least disruption of the regular node operation. Most of the invasive and many of the semi-invasive attacks also require disassembly or physical destruction of the sensor nodes.

The existing literature on physical attacks usually assumes that an attacker can gain unsupervised access to the system to be attacked for an extended period of time. This is a sensible assumption for systems such as pay-per-view TV cards, pre-paid electricity tokens, or GSM SIM cards. Attacks which may take days or even weeks to complete present a real threat to the security of these systems.

In wireless sensor networks, however, regular communication with neighboring nodes is usually part of normal network operation. Continuous absence of a node can therefore be considered an unusual condition that can be noticed by its neighbors. This makes time a very important factor in evaluating attacks against sensor nodes, as the system might be able to detect such attacks while they are in progress and respond to them in real-time. If the amount of time needed to carry out various attacks is known, the frequency with which neighbors should be checked can be adapted to the security goals and the anticipated attacker model.

3.1.2. Current Sensor Node Hardware

Currently available sensor nodes typically consist of embedded hardware with low power consumption, and low computing power. A typical sensor node contains some sensors (light, temperature, acceleration etc.), a radio chipset for wireless communication, an EEPROM chip for logging sensor data, a node-to-host communication interface (typically a serial port), and a microcontroller which contains some amount of flash memory for program storage and RAM for program execution. Power is provided by batteries.

Typical choices for the microcontroller are the 8 bit Atmel ATmega 128 or the 16 bit Texas Instruments MSP430 family, with the amount of RAM varying between 2 kB and 10 kB and flash memory ranging from 48 kB to 128 kB. External EEPROM memory can be as small as 8 kB or as large as 1 MB. The speed of radio communications is in the order of 100 $^{\text{kbit}}/_{\text{s}}$.



Figure 2. Current sensor node hardware: Mica 2 by Crossbow, Berkeley [26]; Tmote sky by moteiv, Berkeley [31]; and Embedded Sensor Board by ScatterWeb, Berlin [10]

The most interesting part for an attacker will be the microcontroller, as control over this component means complete control over the operation of the node. However, the other parts might be of interest as well in certain attack scenario.

Figure 1 shows a general schematic view of the hardware of current sensor nodes, while Figure 2 shows photographs of some concrete models available today. The Crossbow Mica2 nodes [15, 16] (Fig. 2, left) use the 8 bit Atmel ATmega 128 microcontroller [5] with 4 kB RAM and 128 kB integrated flash memory, the Atmel AT45DB041B 4 Mbit flash memory chip [4], and the Chipcon CC1000 radio communications chipset [13] with a maximum data rate of 76.8 kbit/s. Programming is done via the Atmel's serial programming interface by placing the node in a special interface board and connecting it to an RS-232 serial port on the host.

The Telos motes [30, 31] (Fig. 2, center) by Moteiv utilize the Texas Instruments MSP430 F1611 microcontroller [41, 42], providing 10 kB of RAM and 48 kB flash memory. The EEPROM used is the 8 Mbit ST Microelectronics M25P80 [38], and the radio chipset is the Chipcon CC2420 [14], whose maximum data rate is 250 kbit/s. Programming is performed by connecting to the USB interface and writing memory with the help of the MSP430 bootloader [39]. A JTAG interface is available as an alternative programming method and can also be used for debugging.

The Embedded Sensor Boards [10] (Fig. 2, right) from ScatterWeb GmbH are built around the Texas Instruments MSP430 F149 microcontroller [40, 42] featuring 2 kB of RAM and 60 kB flash memory, the Microchip 24LC64 [28] 64 kbit EEPROM, and the RFM TR1001 radio chipset [29] with a maximum data rate of 19.2 kbit/s. Programming is done either through a JTAG interface or over-the-air using a gateway.

3.1.3. Possibilities for Physical Attacks

Concrete physical attacks on sensor nodes can be classified in several ways. Our classification takes our previous considerations into account that sensor nodes are more or less in permanent contact with each other. This means that long interruptions of regular operation can be noticed and acted upon. Therefore, attacks which result in a long interruption (e. g. because the node has to be physically removed from the network and taken to a distant laboratory) are not as dangerous as those which can be carried out *in the field*. Therefore, in the following we concentrate on this type of attacks as well as on possible countermeasures to be employed by a sensor network.

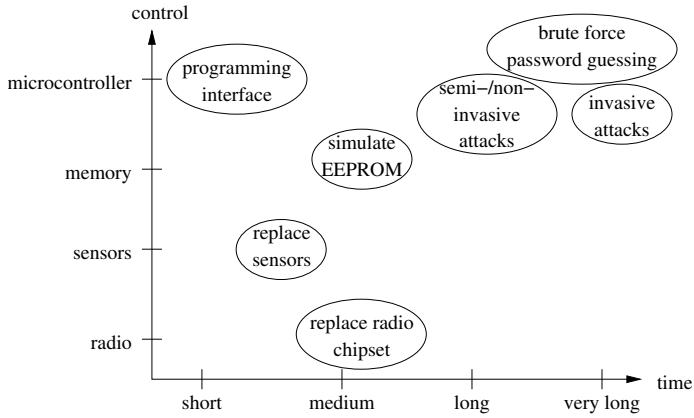


Figure 3. Design space for physical attacks on sensor nodes.

The two main categories that we use for classifying physical attacks are (1) the degree of control over the sensor node the attacker gains; and (2) the time span during which regular operation of a node is interrupted [6]. Figure 3 illustrates this design space and classifies example attacks from the forthcoming Section 3.2 according to its criteria.

3.2. Examples of In-the-Field Attacks and Countermeasures

As explained above, we especially consider attacks which can be mounted without noticeable interruption of the regular sensor node operation. We now discuss some attacks in detail.

3.2.1. Attacks via JTAG

The IEEE 1149.1 JTAG standard is designed to assist electronics engineers in testing their equipment during the development phase. Among other things, it can be used in current equipment for on-chip debugging, including single-stepping through code, and for reading and writing memory.

A JTAG Test Access Port (TAP) is present on both the Atmel and Texas Instruments microcontrollers used on the sensor nodes described above. All sensor nodes examined by us have a JTAG connector on their circuit board allowing easy access to the microcontroller's TAP. While the capabilities offered by JTAG are convenient for the application developer, it is clear that an attacker must not be provided with the same possibilities. Therefore it is necessary to disable access to the microcontroller's internals via JTAG before fielding the finished product.

The MSP430 has a security fuse which can be irreversibly blown (as described in the data sheet) to disable the entire JTAG test circuitry in the microcontroller. Further access to the MSP430's memory is then only possible by using the Bootstrap Loader. The ATmega128 requires the programmer to set the appropriate fuses and lock bits, which effectively disable all memory access via JTAG or any other interface from the outside.

If JTAG access is left enabled, an attacker equipped with an appropriate adapter cable and a portable computer is capable of taking complete control over

the sensor node. Even if there is no JTAG connector provided on the circuit board, attackers can still get access to the JTAG ports by directly connecting to the right pins on the microcontroller which can be looked up in the datasheet. Typical data rates for JTAG access are 1–2 kB/s, so reading or writing 64 kB of data takes between 30 and 70 s. However, there are specialized programming devices on the market which can attain much higher data rates. One such device claims to be able to program 60 kB of memory in a mere 3.4 s.

3.2.2. Attacks via the Bootstrap Loader

On the Telos nodes, the canonical way of programming the microcontroller is by talking to the Texas Instruments specific bootstrap loader (BSL) through the USB interface. The bootstrap loader [39] is a piece of software contained in the ROM of the MSP430 series of microcontrollers that enables reading and writing the microcontroller's memory independently of both the JTAG access and the program currently stored on the microcontroller.

The BSL requires the user to transmit a password before carrying out any interesting operation. Without this password, the allowed operations are essentially "transmit password" and "mass erase", i.e. erasing all memory on the microcontroller.

The BSL password has a size of $16 \cdot 16$ bit and consists of the flash memory content at addresses 0xFFE0 to 0xFFFF. This means in particular that, immediately after a mass erase operation, the password consists of 32 bytes containing the value 0xFF. The memory area used for the password is the same that is used for the interrupt vector table, i.e. the BSL password is actually identical to the interrupt vector table. The interrupt vector table, however, is usually determined by the compiler and not by the user, although Texas Instruments documents describe the password as user-settable.

Finding out the password may be quite time-consuming for an attacker. However, such an investment of time may be justified if a network of nodes all having an identical password is to be attacked. Therefore, an evaluation of the possibility to guess the password follows.

Brute Force. As the password is composed of interrupt vectors, certain restrictions apply to the values of the individual bytes. This section examines the expected size of the key space and estimates the expected duration of a brute force attack on the password.

Initially, the key space has a size of $16 \cdot 16\,\text{bit} = 256\,\text{bit}$. Assuming a typical compiler (mspgcc 3.2 [32] was tested), the following restrictions apply:

- All code addresses must be aligned on a 16 bit word boundary, so the least significant bit of every interrupt vector is 0. This leaves us with a key space of $16 \cdot 15\,\text{bit} = 240\,\text{bit}$.
- The reset vector, which is one of the interrupt vectors, is fixed and points to the start of the flash memory, reducing the key space to $15 \cdot 15\,\text{bit} = 225\,\text{bit}$.
- Interrupt vectors which are not used by the program are initialized to the same fixed address, containing simply the instruction `reti` (return from interrupt). As a worst case assumption, even the most basic program will

still use at least four interrupts, and therefore have a key space of at least $4 \cdot 15\,\text{bit} = 60\,\text{bit}$.

- Code is placed by the compiler in a contiguous area of memory starting at the lowest flash memory address. Under the assumption that the program is very small and uses only $2\,\text{kB} = 2^{11}\,\text{B}$ of memory, we are left with a key space of a mere $4 \cdot 10\,\text{bit} = 40\,\text{bit}$.

We conclude that the size of the key space for every BSL password is at least 40 bit.

A possible brute force attack can be performed by connecting a computer to the serial port (the USB port, in the case of Telos nodes) and consecutively trying passwords. This can be done by executing a modified version of the `msp430-bsl` [32] program that is normally used for communicating with the BSL.

The rate at which passwords can be guessed was measured to be approximately 12 passwords per second when the serial port was set to 9600 baud. However, the MSP430 F1611 used on the Telos nodes is capable of a line speed of 38,400 baud, and at this speed, approximately 31 passwords can be tried per second. Finally, the BSL program normally waits for an acknowledgment from the microcontroller after each message sent over the serial line. If this wait is not performed, the speed of password guessing rises to 83 passwords per second.

The maximum speed of password guessing in practice can therefore be assumed to be 2^7 passwords per second. This is quite close to the theoretical limit of $38,400\,{}^{\text{bit}}/_{\text{s}} \cdot (256\,{}^{\text{bit}}/_{\text{pw}})^{-1} = 150\,{}^{\text{pw}}/_{\text{s}}$.

Recalling that the key space has a size of at least 40 bit, we can now conclude that a brute force attack can be expected to succeed on the average after $2^{40-7-1}\,\text{s} = 2^{32}\,\text{s} \approx 128\,\text{a}$. As 128 years is well beyond the expected life time of current sensor nodes, a brute force attack can be assumed to be impractical.

Knowledge of the Program. One consequence of the fact that the password is equal to the interrupt vector table is that anyone in possession of an object file of the program stored on a sensor node also possesses the password. Worse, even someone who only has the source code of the program still can get the password if he has the same compiler as the developer, since he can use this compiler to produce an image from the source code identical to the one on the deployed nodes.

The secret key in the current TinySec implementation, for example, is contained in the image but does not influence the interrupt vector table. If TinySec were ported to Telos motes, the source code and the compiler used would be sufficient information for an attacker to extract the secret key material. The same holds for any kind of cryptographic mechanism where the key material does not influence the interrupt vectors.

Another way of exploiting the identity of the password and the interrupt vector table is to take one node away from the network and attack the microcontroller on this node with classic invasive or semi-invasive methods. The absence of the node from the network will probably be noticed by the surrounding nodes and its key(s) will be revoked. However, once the attacker succeeds with her long-term attack and learns the BSL password of the one node, it is trivial for her to attack all of the other nodes in the field if they all have the same BSL password.

If an attacker knows the BSL password, reading or writing the whole flash memory takes only about 25 s. In order to avoid these forms of attack, *interrupt*

vector randomization [6] can be used. This significanyly raises the bar for an attacker but is not yet part of standard software distributions.

3.2.3. Attacking the External Flash

Some applications might want to store valuable data on the external EEPROM. For example, the Deluge implementation of network reprogramming in TinyOS stores program images received over the radio there. If these images contain secret key material, an attacker might be interested in reading or writing the external memory.

Probably the simplest form of attack is eavesdropping on the conductor wires connecting the external memory chip to the microcontroller. Using a suitable logic analyzer makes it easy for the attacker to read all data that are being transferred to and from the external EEPROM while she is listening. If a method were found to make the microcontroller read the entire external memory, the attacker would learn all memory contents. This kind of attack could be going on unnoticed for extended periods of time, as it does not influence normal operation of the sensor node.

A more sophisticated attack would connect a second microcontroller to the I/O pins of the flash chip. If the attacker is lucky, the mote microcontroller will not access the data bus while the attack is in progress, and the attack will be completely unnoticed. If the attacker is skillful, she can sever the direct connection between the mote microcontroller and the flash chip, and then connect the two to her own microcontroller. The attacker could then simulate the external memory to the mote, making everything appear unsuspicious.

Of course, instead of using her own chip, the attacker could simply do a "mass erase" of the mote's microcontroller and put her own program on it to read the external memory contents. This operation is even possible without knowledge of the BSL password. While this causes "destruction" of the node from the network's point of view, in many scenarios this might not matter to the attacker.

The exact amount of time required for the attacks proposed above remains to be determined. It should be noted that some of the attacks outlined above require a high initial investment in terms of equipment and development effort. A possible countermeasure could be checking the presence of the external flash in regular intervals, putting a limit on the time the attacker is allowed to disconnect the microcontroller from the external flash.

3.2.4. Sensors

Sensor nodes rely on their sensors for information about the real world, so the ability to forge or suppress sensor data can be classified as an attack. For instance, a surveillance system might be tricked into thinking that the situation is normal while the attacker passes unnoticed through the area under surveillance.

Replacing sensors on the different types of nodes varies in difficulty between child's play and serious electrical engineering, mostly depending on the type of connection between the microcontroller circuit board and the sensors. A pluggable connection—as present on the Mica2 motes—requires an attacker to spend only a few moments of mechanical work. If, on the other hand, the sensors are integrated

into the printed circuit board design, replacing them involves tampering with the conductor wires, cutting them, and soldering new connections. The amount of time required for this will vary with the skill of the attacker, but it can be assumed to be in the order of minutes.

3.2.5. Radio

Finally, the ability to control all radio communications of a node might be of interest to an attacker, e.g., in order to mount an attack using deliberate collisions on the medium access level. We are unaware of any work trying to evaluate the effort necessary to perform such an attack and compare its effort with other attacks that target resource exhaustion and denial-of-service.

3.3. Summary

To summarize, we can classify the above attacks into three categories depending on the effort necessary. We list these classes in order of increasing severity:

1. The class containing the "easy" attacks: Attacks in this class are able to influence sensor readings, and may be able to control the radio function of the node, including the ability to read, modify, delete, and create radio messages without, however, having access to the program or the memory of the sensor node. These attacks are termed "easy" because they can be mounted quickly with standard and relatively cheap equipment.
2. The class containing the "medium" attacks: Attacks in this class allow to learn at least some of the contents of the memory of the node, either the RAM on the microcontroller, its internal flash memory, or the external flash memory. This may give the attacker, e.g., cryptographic keys of the node. These attacks are termed "medium" because they require non-standard laboratory equipment but allow to prepare this equipment elswehere, i.e., not in the sensor field.
3. The class containing the "hard" attacks: Using attacks in this class the adversary complete read/write access to the microcontroller. This gives the attacker the ability to analyze the program, learn secret key material, and change the program to his own needs. These attacks are termed "hard" because they require the adversary to deploy non-standard laboratory equipment in the field.

A different way to classify the above attacks is to look at the time during which the node cannot carry out its normal operation. Here we have the following classes:

1. Short attacks of less than five minutes. Attacks in this class mostly consist of creating plug-in connections and making a few data transfers over these.
2. Medium duration attacks of less than thirty minutes. Most attacks which take this amount of time require some mechanical work, for instance (de-) soldering.
3. Long attacks of less than a day. This might involve a non-invasive or semi-invasive attack on the microcontroller, e.g., a power glitch attack where the

timing has to be exactly right to succeed, or erasing the security protection bits by UV light.
4. Very long attacks which take longer than a day. These are usually invasive attacks on the electronic components with associated high equipment cost.

In thw following section we take these practical insights as the basis for devising a framework of adversary models that can be used for security analysis of WSNs.

4. Adversary Models

Adversary models should be determined with respect to applications. Who are adversaries and what goals do they have? A military sensor network has other security requirements than a network for habitat monitoring. The adversaries can be classified according to the following parameters: goals, intervention, presence, and available resources. We first give an overview over these parameters and then treat the paramaters *intervention* and *presence* in detail later.

4.1. Overview

4.1.1. Goals

When designing security mechanisms in practice, it helps to know the *goals* of the adversary as precisely as possible. Which of the three classical security requirements (Confidentiality, Integrity, Availability) does the adversary try to violate? If the data are valuable (i.e., legitimate users have to pay) or privacy relevant, the adversary would try to gain unauthorized access. If the data are critical (e.g., building or perimeter security), the adversary would try to modify data, such that the alarm is not raised in case of intrusion. Also a denial-of-service attack can successfully disable the network (violating Availability).

Identifying the goals of the adversary is probably the most difficult aspect of security analysis in practice. Therefore the three security requirements Confidentiality, Integrity, and Availability are often treated in a uniform manner (all of them are goals of equal importance).

4.1.2. Presence

The parameter *presence* basically identifies *where* the adversary acts in a wireless sensor network. The basic distinguishing factor is the number and location of the nodes which are in the range of his influence. We distinguish *local*, *distributed* and *global* adversaries.

4.1.3. Intervention

While the presence parameter identifies *where* the adversary can act, the *intervention* parameter identifies *what* the adversary can do in these places. We distinguish *eavesdrop*, *crashing*, *disturbing*, *limited passive*, *passive*, and *reprogramming* adversaries. These classes offer incremental steps of power: Briefly spoken, an eavesdropping adversary can only listen to communication, a crashing adversary

can additionally destroy sensor nodes, a disturbing adversary can additionally upset sensors by manipulating their location or their readings, a limited passive adversary can additionally open a node and steal all its secrets by temporarily removing it from the network, a passive adversary can steal all secrets without displacing the node, and a reprogramming adversary can additionally take full control of a node and cause it to act in arbitrary ways.

4.1.4. Available Resources

There are several resource classes to consider: funding, equipment, expert knowledge, time. In the world of tamper resistance, the adversaries are divided into three classes: *clever outsiders*, *knowledgeable insiders* and *funded organizations* [1]. Commodity sensor networks are likely to be attacked by clever outsiders (who are probably just trying things out) and possibly by knowledgeable insiders, if a substantial gain from the attack can be expected. Available resources are interdependent with the parameters *intervention* and *presence*. However, the precise connection is not always clear. For example, a global adversary need not to be a funded organization. A hacker could subvert the entire sensor network by exploiting some software bug. Or, if a local adversary manages to capture a sensor node and read out its cryptographic keys, he can turn into a distributed or global adversary, depending on how many sensor nodes he is able to buy and deploy as clones of the captured node.

4.1.5. Outlook and Notation

In our view, the presence and intervention parameters are the most relevant ones for basic security analysis. This is why we now look at these two in detail. Before we present them, we need some formal notation.

We model a sensor network as a graph $G = (V, E)$ where the set V is the set of sensor nodes and the set $E \subseteq V \times V$ defines a neighborhood relation. The *number of all sensor nodes* $|V|$ is denoted N. Two sensor nodes v_1 and v_2 are *neighbors* if and only if they are in the neighborhood relation, i.e., $(v_1, v_2) \in E$. A set of nodes $\mathcal{V}$ is *connected* if and only if for all $v_1, v_2 \in \mathcal{V}$ holds that either $(v_1, v_2) \in E$ or $(v_2, v_1) \in E$. Note that in our notation E is *not* necessarily reflexive and transitive because we wish to model sensor networks at a very low layer, i.e., without any routing or transport mechanisms.

4.2. Presence

We now discuss the different and increasingly severe levels of the presence parameter. These levels are defined in an incremental fashion, i.e., every level includes the behavior of the previous one. This implies a total order of parameter values defined by the subset relationship of behaviors. We begin our explanations by starting with the weakest level first.

- local adversary
  A *local adversary* can influence a small localized part of the network [9], for example he has one receiver which can only eavesdrop on several meters,

local → distributed → global

Figure 4. Adversary presence parameter.

or can manipulate only the sensor node which is the closest to him (for example, it is installed in his office).
Formally, an adversary is local if his range of influence contains a small connected subset $S \subset V$ of all sensor nodes. The set S can also be given as a Boolean function $S : V \mapsto \{\text{true}, \text{false}\}$. Such a set is small if its size is several magnitutes smaller than the number of total nodes, i.e., $|S| \ll N$. This set can also change over time, but changes are rather slow.
In general it is always possible to define S in a time-dependent manner, i.e., $S : V \times T \mapsto \{\text{true}, \text{false}\}$ where T is the domain of time values.

- distributed adversary
  A *distributed adversary* models either a mobile adversary (a car with receiver driving around) or managed to install his own sensor nodes in the sensor field and coordinates them [2].
  Formally, an adversary is distributed if its range of influence consists of multiple unconnected small subsets of nodes of the sensor network. As an example of such an adversary, consider the assumption that sensor nodes are attacked randomly following a uniform distribution [49]. In such a case, compromised nodes are distributed over the entire network, but not wide enough to actually see, hear or listen anything.
- global adversary
  A *global adversary* is the most powerful level of presence an adversary can exhibit. Such an adversary can analyze the complete network, hence his area of influence consists of *all* sensor nodes.

We define the relation → to order attacker models in the direction of stronger (i.e., more severe) attacker behavior. Formally, model $A \rightarrow B$ if and only if all behaviors which are possible in A are also possible in B (behavior subsetting). The levels of ability ordered with respect to → are depicted in Figure 4.

4.3. Intervention

Now that presence has been investigated and an ordered set of presence abilities has been given, the same needs to be done for the intervention capabilities an adversary can be ascribed.

The intervention order is constituted of the following levels, starting from the weakest and proceeding towards the strongest. Every level contains the behaviors of all preceding levels.

- eavesdropping adversary
  An eavesdropping adversary can just listen to network traffic, do nothing else, in particular no jamming etc. She can then analyse this traffic either online or offline.
- crashing adversary
  A crashing adversary exhibits the simplest form of failure: The node simply stops to operate (execute steps of the local algorithm). Elsewhere this is

eavesdrop → crash → disturbing → limited passive → passive → reprogramming

Figure 5. Adversary intervention parameter.

also called *failstop adversary* [8]. A *crashing* adversary attacks sensors such that they completely break down. The sensors can be destroyed or drained of energy.

- disturbing adversary
  A *disturbing* adversary can try to partially disturb protocols, even if he does not have full control over any sensors. He can selectively jam the network, or fool some sensors into measuring fake data. For example, he can hold a lighter close to a temperature sensor [46] or re-arrange the topology of the sensor network by throwing sensors around.
- limited passive adversary
  An adversary of this level can retrieve all information on a node, but in order to do so he needs to completely remove the node physically from the network. This means that the danger of being caught is higher.
- passive adversary
  A passive adversary can directly go and open up arbitrary sensor nodes to get the information currently stored in such a node, there is no need to leave the network to do so. Furthermore, such an adversary is assumed to be able to modify the data the node contains.
- reprogramming adversary
  A *reprogramming* adversary can run arbitrary programs on a sensor node. This can be achieved by exploiting some software bug, or by probing out cryptographic keys and other secret information and then cloning the node as described above. The problem of *node capture* is typical for a reprogramming adversary.

The order above defines the different abilities of intervention that can be ascribed to an adversary. These can be ordered according to the relation → as defined above (subsets on behaviors), see Figure 5.

4.4. Adversary Model Lattice

Having defined ordered levels for both presence and intervention, those two abilities are now combined to form the complete set of adversary models. An adversary model $\mathcal{A}$ is a pair (Presence, Intervention), where the first component specifies the level of presence the adversary model assumes and analogously the second component states the level of intervention. A fully specified adversary model thus looks like $\mathcal{A}(local, disturbing)$, in case of an adversary with local presence and disturbing skills. The result is a partial order of adversary models. This partial order is depicted in Figure 6 as a lattice. It can be seen that the most powerful adversary is $\mathcal{A}(global, reprogramming)$ (in the upper right corner) and the weakest is $\mathcal{A}(local, eavesdrop)$ (in the bottom left corner). As it is a partial order not all adversary models are comparable in the sense that one model is truly stronger than another. $\mathcal{A}(global, eavesdropping)$ and $\mathcal{A}(local, reprogramming)$ are such a pair, as neither of them is stronger than the other. However some models can be

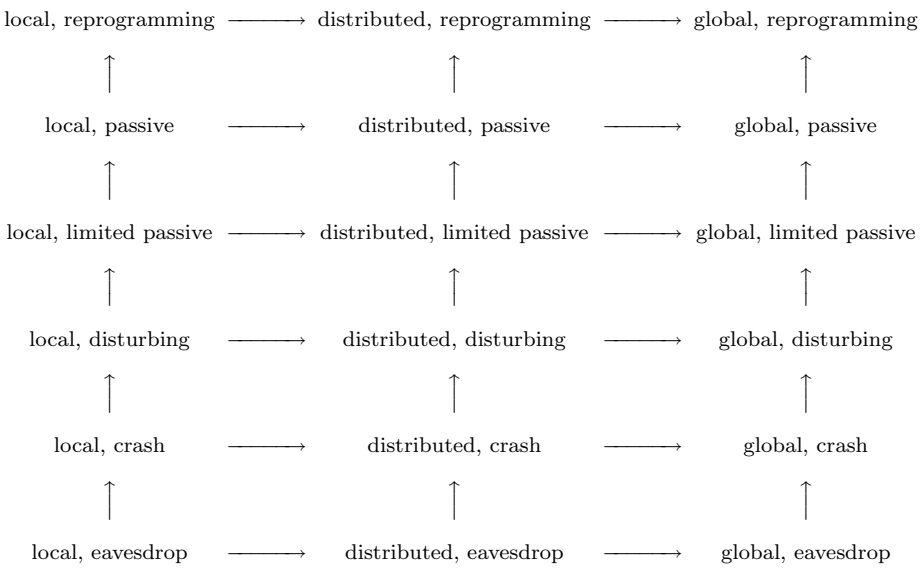


Figure 6. Adversary Model Lattice

put into relation. We take the relation $\rightarrow$ to be the combination of the behavior subsetting ordering defined for presence and intervention parameters above. The resulting partial order is depicted in Figure 6.

There are issues that only concern one ability, either presence or intervention, thus being ignorant about the other ability. For such cases, a $*$ symbol can be inserted for notational convenience into the appropriate component of the model to stand of all different levels possible. For instance, wanting to talk about an adversary that has local presence skills and neglecting completely intervention, the adversary model $\mathcal{A}(local, *)$ would be appropriate to refer to such an adversary.

The ordering of the levels of abilities yields the nice property that a statement ascribing an adversary of a low level (for example a $\mathcal{A}(local, *)$) to perform a malicious action, will automatically ascribe the same for adversaries of higher levels ($\mathcal{A}(local, *)$ and $\mathcal{A}(global, *)$). Similarly, it should be clear that an algorithm that can cope with a $\mathcal{A}(global, passive)$ adversary will be able to sustain all other presented adversaries. However, as the lattice presents only a partial order, an algorithm coping with a $\mathcal{A}(local, \text{limited passive})$ adversary will not necessary work as well with a $\mathcal{A}(global, eavesdrop)$.

4.5. Summary

We have presented an abstract framework for modeling attackers in WSNs. We claim that due to our experiments described in Section 3 these abstract classes model well the critical differences between the hardness of sultions.

5. Discussion of Protection Mechanisms

In Section 3 we systematically investigated physical attacks on current sensor node hardware, paying special attention to attacks which can be executed directly in the deployment area, without interruption of the regular node operation. We found out that most serious attacks, which result in full control over a sensor node (*node capture*), require absence of a node in the network for a substantial amount of time. We also found simple countermeasures for some of the most serious attacks.

Thus, in order to design a WSN secure against those node capture attacks described in this chapter, the following steps should be applied:

- take standard precautions for protecting microcontrollers from unauthorized access;
- choose a hardware platform appropriate for the desired security level, and keep up-to-date with new developments in embedded systems security;
- monitor sensor nodes for periods of long inactivity;
- allow for revocation of the authentication tokens of suspicious nodes.

Standard precautions for protecting microcontrollers from unauthorized access, such as disabling the JTAG interface, or protecting the bootstrap loader password, are an absolute prerequisite for a secure sensor network. We developed a method of protecting the bootstrap loader password by randomization of the interrupt vector table. This allows the developers to make source code of their products public without fearing that their WSN can be taken over by everybody who compiles the source code using the same compiler, thus obtaining the same interrupt vector table, and therefore, the same BSL password.

As security is a process, not a product, system designers should keep up-to-date with the developments in attacks on embedded systems. The security of important systems should be constantly re-evaluated to take new discoveries into account, as newly found attack methods on microcontrollers or previously unknown vulnerabilities might make a previously impossible low-cost attack in the field possible.

The level of security required from the application should also be kept in mind when choosing hardware. In some cases it might make sense to build additional protection, such as a secure housing, around a partially vulnerable microcontroller.

Finally, the removal of a sensor node from the deployment area can be noticed by its neighbors using, e. g., heartbeat messages or topology change notifications, as well as by the sensor node itself using, e. g., acceleration sensors. Appropriate measures can then be taken by the network as well as by the node itself. The network might consider a node that has been removed as "captured" and revoke its authorization tokens or initiate recovery when this node returns to the network [45]. The node itself might react to a suspected physical attack by erasing all confidential material stored on it.

Mechanisms should be developed that allow a sensor node which has been absent for too long from the network to be revoked by its neighbors. This is our future work. Note that depending on the WSN design, local revocation could

be insufficient. For example, if an attacker removes a single sensor node from the network and successfully extracts the node's cryptographic keys, the attacker would be able to *clone* nodes, to populate the network with new sensor nodes which all use the cryptographic keys of the captured sensor node. Thus, a WSN should also be protected from node cloning.

In general, for passive adversaries, encryption techniques often suffice to ensure inside security. Symmetric key encryption techniques will be favored over asymmetric ones because of the computational advantage and the comparatively small key sizes. The problems arise in the setup phase of the network where shared secrets need to be distributed either by the manufacturer at production time or by clever protocols at deployment time [2, 23].

For stronger adversaries, active attacks like impersonation and node capture must be taken into account. Ideally, sensor nodes should be made tamper proof to prevent node capture, e.g., by applying technology known from smart cards or secure processing environments. There, memory is shielded by special manufacturing techniques which makes it more difficult to physically access the stored information [1]. Similarly, sensor nodes could be built in a way that they loose all their data when they are physically tampered with by unauthorized entities.

For large sensor networks, cost considerations will demand that sensor nodes are not tamper proof. Therefore, node capture must be taken into account. A first step to protect a sensor network from node capture against a local or partially present adversary is to use locally distributed protocols that can withstand the capture of a certain fraction of nodes in the relevant parts of the network. For example it is possible to devise access control protocols that work even if up to a certain number t out of any n nodes in the communication range of the adversary are captured [9]. While these protocols can exploit broadcast communication, they are still relatively resource intensive.

A very general approach to counter node capture is to increase the effort of the adversary to run a successful attack. For example, data may be stored at a subset of nodes in the network and continuously be moved around by the sensors to evade the possible access by the adversary [7]. This makes it harder for the adversary to choose which node to capture. In the worst case, the adversary must capture much more than t out of n nodes to successfully access the data in question.

A similar technique that exploits the inherent redundancy of a sensor network and shields against even global adversaries is to devise it as a virtual minefield. A certain fraction of sensors has special alerting software enabled which scans the communication in its vicinity and reacts to local and remote manipulations by issuing a distress signal in its environment or otherwise cause an unpleasant experience to the adversary. Thus, these sensors act as "mines" in the network which the adversary tries to avoid since capturing them needs more resources than capturing a normal sensor node. If it is not possible for the adversary to distinguish these special sensors from normal sensors, the ratio between the fraction of "mines" and the effort to capture them determines the incentive of the adversary to attack the system. For example, if 10% of the sensors are virtual mines and it takes 1000 times more effort to capture a mine than to capture a

normal node, the adversary will waste a lot of resources if he needs to capture even a small subset of sensors without raising an alarm.

6. Summary and Conclusions

We have described security goals, adversary models and protection mechanisms which are relevant and specific for sensor networks. There are a lot of interesting problems and open questions in this area:

- *Realistic* adversary models should be derived with respect to existing and future applications. Here, experiences with GSM and WLAN security (and security failures) can be used as a guideline, but every application needs to define its own adversary model to be able to talk about security.
- What other attack possibilities exist for sensor networks and how much effort do they cost to be pursued? For example, are software-based node capture attacks a real threat? Are side-channel attacks on sensor nodes possible? We believe both to be true, but we are unaware of any work which has tried it.
- As cross-layer integration is especially important for resource-constrained sensor nodes, careful design decisions must be taken concerning which security means to put into which layer. For example TinySec [24], a link layer encryption and message integrity protection mechanism, is integrated into the radio stack of MICA Motes.
- Building secure sensor networks, especially with respect to active adversary, remains a challenge. Can it be done by combining existing solutions, such as random key predistribution, secure routing, secure data aggregation, or would it be too expensive in terms of energy?

Overall, we speculate that probabilistic algorithms which exploit the redundancy of the sensor network to cause high effort for the adversary will be good candidates to establish security in such networks. In a sense, these algorithms mimic guerrilla tactics: evasion and disguise. They can be simple, yet unpredictable. However, their simplicity implies that they are not suitable to establish perfect security. The security goals of sensor networks will be probabilistic and depend on the strength of the adversary.

Acknowledgements

We wish to thank Alexander Becher and Maximillian Dornseif for many helpful discussions and the delightful previous cooperation in breaking sensors [6] which formed the basis of Section 3.

References

[1] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems.* John Wiley & Sons, Inc., 2001.

[2] Ross J. Anderson, Haowen Chan, and Adrian Perrig. Key infection: Smart trust for smart dust. In *ICNP*, pages 206–215, 2004.

[3] Ross J. Anderson and Markus G. Kuhn. Low cost attacks on tamper resistant devices. In *Proceedings of the 5th International Workshop on Security Protocols*, pages 125–136, London, UK, 1998. Springer-Verlag.

[4] Atmel Corp. AT45DB041B datasheet. Atmel document no. 3443, available at `http://www.atmel.com/dyn/resources/prod_documents/doc3443.pdf`.

[5] Atmel Corp. ATmega128 datasheet. Atmel document no. 2467, available at `http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf`.

[6] Alexander Becher, Zinaida Benenson, and Maximillian Dornseif. Tampering with motes: Real-world physical attacks on wireless sensor networks. In *Security in Pervasive Computing, Third International Conference, SPC 2006*, Lecture Notes in Computer Science 3934, pages 104–118. Springer, 2006.

[7] Zinaida Benenson, Peter M. Cholewinski, and Felix C. Freiling. Simple evasive data storage in sensor networks. In *IASTED PDCS*, pages 779–784, 2005.

[8] Zinaida Benenson and Felix C. Freiling. On the feasibility and meaning of security in sensor networks. In *Proceedings 4th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze", Zurich, Switzerland*, March 2005.

[9] Zinaida Benenson, Felix C. Gärtner, and Dogan Kesdogan. An algorithmic framework for robust access control in wireless sensor networks. In *Second European Workshop on Wireless Sensor Networks (EWSN)*, January 2005.

[10] FU Berlin. ScatterWeb Embedded Sensor Board. Online at `http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb_net/esb/`.

[11] Erik-Oliver Blass, Joachim Wilke, and Martina Zitterbart. A Security–Energy Trade-Off for Authentic Aggregation in Sensor Networks. In *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), Extended Abstract*, pages 135–137, Washington D.C., USA, September 2006. ISBN: 1-4244-0732-X.

[12] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, pages 197–213, May 2003.

[13] Chipcon AS. CC1000 datasheet. Available at `http://www.chipcon.com/files/CC1000_Data_Sheet_2_3.pdf`.

[14] Chipcon AS. CC2420 datasheet. Available at `http://www.chipcon.com/files/CC2420_Data_Sheet_1_2.pdf`.

[15] Crossbow, Inc. MICA2 data sheet. Available at `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf`.

[16] Crossbow, Inc. MPR, MIB user's manual. Available at `http://www.xbow.com/Support/Support_pdf_files/MPR-MIB_Series_Users_Manual.pdf`.

[17] J. Deng, R. Han, and S. Mishra. A performance evaluation of intrusion-tolerant routing in wireless sensor networks. In *2nd IEEE International Workshop on Information Processing in Sensor Networks (IPSN 2003)*, April 2003.

[18] Tassos Dimitriou and Dimitris Foteinakis. Secure and efficient in-network processing for sensor networks. In *First Workshop on Broadband Advanced Sensor Networks (BaseNets)*, 2004.

[19] Abhishek Ghose, Jens Grossklags, and John Chuang. Resilient data-centric storage in wireless ad-hoc sensor networks. In *MDM '03: Proceedings of the 4th International Conference on Mobile Data Management*, pages 45–62. Springer-Verlag, 2003.

[20] Mark G. Graff and Kenneth R. van Wyk. *Secure Coding: Principles and Practices*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2003.

[21] Michael Howard and David LeBlanc. *Writing secure code*. Microsoft Press, pub-MICROSOFT:adr, second edition, 2003.

[22] Lingxuan Hu and David Evans. Secure aggregation for wireless networks. In *SAINT-W '03: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, page 384. IEEE Computer Society, 2003.

[23] Joengmin Hwang and Yongdae Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 43–52. ACM Press, 2004.

[24] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A link layer security architecture for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004)*, November 2004.

[25] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's Ad Hoc Network Journal, Special Issue on Sensor Network Applications and Protocols*, September 2003.

[26] Geoff Martin. An evaluation of ad-hoc routing protocols for wireless sensor networks. Master's thesis, University of Newcastle upon Tyne, May 2004.

[27] Gary McGraw and John Viega. *Building Secure Software: How to Avoid Security Problems the Right Way*. Addison-Wesley, September 2001.

[28] Microchip Technology. 24AA64/24LC64 datasheet. Available at `http://ww1.microchip.com/downloads/en/DeviceDoc/21189K.pdf`.

[29] RF Monolithics. TR1001 datasheet. Available at `http://www.rfm.com/products/data/tr1001.pdf`.

[30] moteiv Corp. Telos revision B datasheet. Available at `http://www.moteiv.com/products/docs/telos-revb-datasheet.pdf`.

[31] moteiv Corp. Tmote Sky datasheet. Available at `http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf`.

[32] `http://mspgcc.sourceforge.net/`.

[33] Holger Peine. Rules of thumb for secure software engineering. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 702–703, New York, NY, USA, 2005. ACM Press.

[34] Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, 2004.

[35] Bartosz Przydatek, Dawn Song, and Adrian Perrig. SIA: Secure information aggregation in sensor networks. In *ACM SenSys 2003*, Nov 2003.

[36] Sergei P. Skorobogatov. Semi-invasive attacks - a new approach to hardware security analysis. Technical report, University of Cambridge, Computer Laboratory, April 2005. Technical Report UCAM-CL-TR-630.

[37] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In *CHES '02: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 2–12, London, UK, 2003. Springer-Verlag.

[38] STMicroelectronics. M25P80 datasheet. Available at `http://www.st.com/stonline/products/literature/ds/8495.pdf`.

[39] Texas Instruments. Features of the MSP430 bootstrap loader (rev. B). TI Application note SLAA089B, available at `http://www-s.ti.com/sc/psheets/slaa089b/slaa089b.pdf`.

[40] Texas Instruments. MSP430 F149 datasheet. Available at `http://www-s.ti.com/sc/ds/msp430f149.pdf`.

[41] Texas Instruments. MSP430 F1611 datasheet. Available at `http://www-s.ti.com/sc/ds/msp430f1611.pdf`.

[42] Texas Instruments. MSP430x1xx family: User's guide. TI Application note SLAU049E, available at `http://www-s.ti.com/sc/psheets/slau049e/slau049e.pdf`.

[43] John Viega, Matt Messier, and Gene Spafford. *Secure Programming Cookbook for C and C++*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2003.

[44] Harald Vogt. Exploring message authentication in sensor networks. In *Security in Ad-hoc and Sensor Networks (ESAS), First European Workshop*, volume 3313 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 2004.

[45] Harald Vogt, Matthias Ringwald, and Mario Strasser. Intrusion detection and failure recovery in sensor nodes. In *Tagungsband INFORMATIK 2005, Workshop Proceedings*, LNCS, Heidelberg, Germany, September 2005. Springer-Verlag.

[46] David Wagner. Resilient aggregation in sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 78–87. ACM Press, 2004.

[47] A. Wood, J. Stankovic, and S. Son. JAM: A mapping service for jammed regions in sensor networks. In *In Proceedings of the IEEE Real-Time Systems Symposium*, December 2003.

[48] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. LEAP: efficient security mechanisms for

large-scale distributed sensor networks. In *Proceedings of the 10th ACM conference on Computer and communication security*, pages 62–72. ACM Press, 2003.

[49] Martina Zitterbart and Erik-Oliver Blaß. An Efficient Key Establishment Scheme for Secure Aggregating Sensor Networks. In *ACM Symposium on Information, Computer and Communications Security*, pages 303–310, Taipei, Taiwan, March 2006. ISBN 1-59593-272-0.

Wireless Sensor Network Security
J. Lopez and J. Zhou (Eds.)
IOS Press, 2008

Symmetric Primitives

Manfred AIGNER [a], Martin FELDHOFER [a], Stefan TILLICH [a]
[a] *Graz University of Technology,*
Institute for Applied Information Processing and Communications,
Inffeldgasse 16a, A–8010 Graz, Austria
{Manfred.Aigner,Martin.Feldhofer,Stefan.Tillich}@iaik.tugraz.at

Abstract. This chapter highlights the importance of symmetric cryptographic primitives for providing security in wireless sensor networks. We outline the basic goals and primitives and give a comprehensive overview in regard to modes of operation. We also provide an extensive survey of the implementation options of the Advanced Encryption Standard (AES): In software on processors of different word size, in hardware with different optimization goals, as well as in a hardware/software co-design approach with cryptographic instruction set extensions. An overview of state-of-the-art cryptographic support in today's WSN products concludes this chapter.

Keywords. Wireless sensor networks, WSN, security, symmetric primitives, hash functions, AES, modes of operations.

Introduction

Developers of wireless sensor networks (WSNs) are facing many technological challenges due to the special characteristics of such networks. The autonomous sensor-equipped nodes form a distributed communication network of heterogeneous devices. Possible applications range from medicine, over environmental observation to smart-home applications. Due to the possible large number of nodes, many applications for WSNs require unattended operation of the nodes, as well as remote programming and maintenance on a collective rather than on an individual level.

The network is built up in a heterogeneous way, by including high-performance gateways without any limitations in terms of computation facility or communication capabilities, mid-performance mobile devices like personal digital assistants (PDAs) or mobile phones and a high number of inexpensive sensor equipped nodes, all connected together via wireless communication links.

Mobility of the nodes requires a dynamic network topology and ad-hoc communication. The nodes themselves need to operate in a harsh environment and under very limited energy resources. In order to improve the lifetime and to reduce cost of the nodes their computation capabilities are typically very limited.

Providing security for WSNs is far from trivial. Established approaches from other network types are often hard to transfer due to the special characteristics of WSNs. The wireless link is highly susceptible to eavesdropping and the ad-hoc communication and the dynamic network topology make establishment of secure channels more complicated

than in traditional network architectures. Securing the communication against attackers increases the amount of data that needs to be sent and additional operations need to be computed by every communicating party. Computation of those cryptographic algorithms is typically cost intensive compared to other operations a typical sensor node performs. When designing security measures for WSNs, one therefore needs to take the special requirements of the heterogenous network, i.e. the low cost sensor nodes into account.

Application of cryptographic protocols is the approach to protect a network against threats like eavesdropping, unauthenticated access to the network or illicit change of transmitted messages. Cryptography can provide the following security assurances to ensure protection against attacks:

Confidentiality means to keep data secret for a defined set of recipients during transmission, while the transmission channel can be insecure. Eavesdropping is a logical threat against confidential transmission of data over unprotected RF links. Encryption of data with a key that is shared amongst all authorized parties is typically used to achieve confidentiality in open networks. This implies that pre-shared keys are already established between communicating parties, and that communicating parties are authenticated against each other before confidential communication can be established.

Data Integrity addresses the threat of unauthorized manipulation of data. Data integrity is also linked to authentication, since any modification can also be seen as a change of origin of the data. Providing data integrity in a transmission, usually keyed hash algorithms are used to generate a fingerprint of the message that can be produced or verified only when the corresponding secret key is known.

Authentication is the proof of a claimed identity during communication. Typically this proof is achieved by challenge-response protocols. In such protocols a party that is asked to prove its identity encrypts a random challenge which was generated and sent by the one who claims authentication. The encryption result is sent back as response to the challenge. The verifying party can then check with the appropriate key if the identity is correct or not, since only a communication partner that owns the correct key is able to produce the correct answer to a random challenge. Mutual authentication requires a three-pass protocol to authenticate both communication partners. There are other methods to provide authentication, like passwords or zero knowledge schemes, but in the context of WSNs, challenge-response protocols are the most meaningful.

Non-Repudiation prevents to deny commitments or services that were previously given. For example, if a system supports non-repudiation of origin, a party which sent a certain message cannot deny afterwards that this message has been sent. Non-repudiation of delivery is supported when it is possible to prove that a certain party received a message. Non-repudiation basically ensures that one party can prove that the other party committed to something. This can be important for some applications; all protocols need a third party that is trusted by the others so that the service can be achieved.

All described protocols use different cryptographic algorithms or primitives as underlying building blocks. The cryptographic primitives are typically separated into three different categories:

Unkeyed Primitives: This category is best represented by arbitrary-length hash functions. They produce a fixed-length message digest out of an input message of arbitrary length. Hash functions are important to provide message integrity. One important property of cryptographic hash functions is that it is computationally infeasible to find any two messages, so that their hashing yields the same hash value. Random number generators are another type of unkeyed primitives. Typically, random numbers are required for use of cryptographic primitives in different modes of operations (e.g. unpredictable initialization vectors for block cipher modes) or in cryptographic protocols (e.g. random numbers as challenges for challenge-response authentication).

Symmetric Primitives or symmetric-key ciphers consist of an encryption and a decryption transformation using the same key value as input, or more accurately, a key input for encryption and a key input for decryption that is basically equal, or "easy" to derive. Block ciphers and stream ciphers are the best known representatives. While block ciphers operate on fixed-length input blocks, the input of stream ciphers is bit oriented. Message Authentication Codes (MACs), that are basically arbitrary length hash function with additional key input, are also considered as symmetric primitives. The major advantage of symmetric-key primitives is their limited computational complexity. The algorithms can be performed in reasonable time on small microcontrollers or small dedicated hardware modules can be designed to increase performance or decrease energy consumption. Since security of encryption schemes using symmetric-key encryption is established as long as the key is a shared secret between the communication partners, such schemes are often referred to as secret-key cryptosystems. Delivery of the keys over insecure communication channels to all communication partners and management of all active keys in a system is the major concern for symmetric-key cryptography. Often asymmetric cryptography is used to establish session keys for symmetric encryption.

Asymmetric Primitives are a combination of an encryption function and a decryption function that use different key inputs. The two key values are typically generated together, and often one of the two keys is published (public key) while the other is kept secret (private key). Unlike in symmetric encryption schemes, it is computationally infeasible to deduce the private key from the public key. Asymmetric primitives allow secure communication although one of the two key values was published. Therefore, they are often referred to as public-key cryptosystems. This characteristic facilitates key establishment and management over insecure channels compared to systems using only symmetric cryptographic methods. The major disadvantage of asymmetric cryptography is the high computational effort that is necessary to compute the algorithms. On small microcontrollers (e.g. 8-bit word size) some of the algorithms can lead to an unacceptably long computation time or require too much memory. Furthermore, the energy consumption due to the higher computation effort is often not negligible. The size of the key to achieve the same level of security is furthermore bigger than for symmetric crypto algorithms. This increased key length often results in higher communication effort (and therefore higher energy consumption for transmitting data) to perform cryptographic protocols with asymmetric primitives.

This chapter will focus on application and implementation issues of symmetric cryptographic primitives in the context of WSNs. Due to their similar application and implementation aspects we also consider unkeyed primitives like cryptographic hash functions.

Due to the heterogeneous characteristic and broad application area of WSNs, a general view of application of symmetric cryptographic primitives is hard to discuss. We have chosen an approach to illustrate their application on different nodes in a wireless sensor network for a specific application. In all our examples we refer to applications of a WSN in the area of telematics. The network consists of a high number of small, low-cost sensor nodes, so-called motes, that are placed along roads, e.g. in combination with traffic signals. Those motes are designed for long lifetime, use a small 8-bit or 16-bit microcontroller to observe road conditions and traffic status. They communicate via an RF-link (e.g. ZigBee) with all other parties in their communication range. We consider a scenario where these motes are powered by a battery. A very high number of motes is considered to participate in the network, therefore the cost for a single mote needs to be as low as possible. The lifetime is a significant cost factor, since changing the batteries of motes placed in remote areas is expensive.

The second type of devices in the WSN we have chosen are mobile high-end nodes with powerful batteries and improved communication facilities (ZigBee [79], WLAN [29], and Bluetooth [32]). These PDA-like nodes may or may not have sensors, they are placed in cars as part of the on-board automotive network or are mobile devices operated by the road maintenance staff. PDA-like nodes are considered to have significant processing power (typically 32-bit embedded processors) for mobile application. Their batteries are recharged on a regular basis; low energy consumption is therefore an issue, but less restrictive than for motes. They communicate with motes via ZigBee and with the other devices using the best available communication link.

The third type of devices encompasses gateways. Those gateways are high-performance computer systems with wireless communication interfaces, but also with connection to a backbone network and databases. Gateways could be placed for example in road service stations or tollgates, at entrances to airports, parking decks, and similar positions. They collect the information from nearby motes or communicate with the mobile PDA-like nodes of passing cars. Their number is very limited compared to sensor nodes or PDA-like nodes in the system and their cost is therefore of less importance. Computational power and memory can be treated as nearly unrestricted resources.

The data generated and provided by the network is critical in many senses. Privacy aspects require that data about individuals is not available to unauthorized parties. Wrong data provided by e.g. road condition sensors can be perilous for drivers relying on the sensor data. It is therefore obvious that all devices in the network that allow such an application need to be protected against attacks.

We suggest to rely on established cryptographic standards for protection of the communication within a wireless network as illustrated above. The choice of the standards for different levels of the network topology may be different, nevertheless it should be assured that devices of lower levels are still able to communicate with devices of upper levels in the network topology using the same encryption primitives. Due to the higher computational resources of devices on higher network levels, the choice of different primitives or algorithms is naturally broader; nevertheless, we want to point out that the protection level should be the same in the overall network.

An attacker will always target the weakest point in the system. Therefore, it normally does not make sense to decide for lower protection levels in mote-to-mote communication but protecting the gateway communication with stronger algorithms. Although the "value" of the motes' messages might be much lower than that of gateway messages, application of low-security primitives for mote communication could open the way for a multitude of attacks. As a general rule, we suggest to use standardized or established protocols eventually with reduced functionality instead of re-inventing the wheel with the high chance to generate security bottlenecks due to poor security evaluation of proprietary algorithms or protocols. For mote-to-mote and mote-to-PDA-like device communication we suggest to use TinySec with AES as encryption primitive, for communication between PDA-like nodes and PDA-like devices with gateways we suggest to rely on TLS with a limited number of cipher suites. For protection of data on the gateway backbone, standardized security mechanisms like IPSec/TLS are suggested.

After this introduction, Section 1 describes the modes of operation of cryptographic primitives. This is especially important for application of cryptographic primitives in WSNs since selection of proper primitives and application in different modes can save critical resources without reducing the overall security. In Section 2, the reader will find a brief description of important symmetric-key and unkeyed algorithms. Since a large number of algorithms is available, the description is by no means complete. We tried to focus on symmetric encryption algorithms and hash algorithms that are especially interesting for application in WSN. A more detailed description of the Advanced Encryption Standard (AES) is also given, because Section 3 will then discuss implementation issues of AES in software, hardware and a hybrid approach using instruction set extensions. All implementation options are directly mapped to the requirements of specific parts of the above illustrated telematic WSN. Section 4 gives a short overview of currently available sensor node products and their support of cryptographic features.

1. Modes of Operation

Cryptographic primitives can be used in a multitude of ways in order to provide different security assurances. A method which defines a specific usage of one or more cryptographic primitives is denoted *mode of operation*. For constrained devices whose limitations might only allow the support of a single primitive, the careful selection of the mode of operation may be the sole chance for providing the desired security services.

Block ciphers are very popular building blocks and there exists a large number of modes of operation for them. The most common ones deal with providing either confidentiality or data integrity. Some modes of operation offer both of these assurances. Another class uses the block cipher as a building block for a hash function.

1.1. Confidentiality Modes

A block cipher primitive just defines the encryption and decryption of a fixed-size data block. Confidentiality modes of operation deal with encryption and decryption of messages of different size. In some modes the message can be of arbitrary size, while in others it needs to conform to some prerequisites (e.g. multiple of the block size), which may require message padding. The following five modes of operation have been recommended by the U.S. *National Institute of Standards and Technology (NIST)* [53].

The most basic mode of operation is Electronic Codebook (ECB) mode. Every block of plaintext is encrypted independently under the same cipher key. A given key therefore defines a "codebook" which always maps a specific plaintext block to a specific ciphertext block. This constant mapping can lead to undesirable properties, as patterns in the plaintext may be still visible in the resulting ciphertext. Therefore, ECB should be only used in rare circumstances, where its properties have been taken into account.

In the Cipher-Block Chaining (CBC) mode the previous ciphertext block is bitwise exclusive-ored to the following plaintext block. The result is encrypted with the block cipher, yielding the next ciphertext block. The first plaintext block is exclusive-ored with the so-called initialization vector (IV). The IV needs not to be kept secret, but it needs to be unpredictable. In CBC encryption, the plaintext blocks need to be encrypted in sequence. On the other hand, if the complete ciphertext is available, CBC decryption can be performed in parallel on all ciphertext blocks.

The Cipher Feedback (CFB) mode allows encryption of plaintext blocks of a smaller size than the blocks for the underlying block cipher. These smaller blocks are normally denoted segments. A shift register of the block size of the block cipher is used as input to the cipher. The register is initialized with an unpredictable IV. A part of the cipher output with segment size is exclusive-ored to the current plaintext segment to produce a segment of ciphertext. This ciphertext segment is also used to update the contents of the shift register by shifting it in from the right, which results in the next block cipher input. CFB mode can be used to encrypt and transmit small data chunks (e.g. bytes), without the need to wait for or pad to complete cipher blocks. Another interesting property is that the encryption function of the block cipher is used for both CFB encryption and decryption.

In the Output Feedback (OFB) mode, the IV is repeatedly encrypted with the block cipher, i.e. the cipher output is used as input for the next cipher encryption. Similarly to the CFB mode, the cipher output blocks are exclusive-ored to the plaintext blocks to produce the ciphertext blocks. OFB mode also just requires the use of the block cipher encryption function. Its main advantage over CFB mode is that errors in the ciphertext only lead to errors in the respective plaintext bits and do not affect other parts of the recovered plaintext.

Similarly to CFB and OFB, the Counter (CTR) mode of operation only uses the block cipher encryption for both CTR encryption and decryption. Unique input blocks (normally denoted counter blocks) are encrypted with the block cipher and the resulting output blocks are exclusive-ored to the plaintext, yielding the ciphertext. The counter blocks can be encrypted in parallel and independent of the ciphertext. Therefore, CTR mode allows to parallelize encryption and decryption and to precompute cipher output blocks (i.e. most of the cryptographic work) prior to the availability of the plaintext.

1.2. Authentication Modes

Authentication modes of operation can be used to achieve data integrity and authentication. Generally, the input for such a mode is a message and the output is a *Message Authentication Code (MAC)*. A verifier can then check whether the MAC has been generated by an entity which knows the correct key. The size of the MAC has to be chosen in such a way, that the chance of an attacker to guess a valid MAC is limited to an acceptable small degree. Another potential issue which is not addressed by the authenti-

cation mode is that an attacker can record authentication messages and replay them at a later time. If protection against such *replay attacks* is a concern, it must be provided by additional measures, e.g. inclusion of timestamps or sequence numbers in messages.

Cipher Block Chaining MAC (CBC-MAC) is a very basic method for generating a MAC. It simply consists of CBC mode encryption with a zero IV, where the last resulting ciphertext block is used as MAC. However, CBC-MAC has security deficiencies, e.g. it is not secure for variable-sized messages [45]. There have been various extensions to solve the problems of CBC-MAC.

The Cipher-based MAC (CMAC) mode is one of the extensions of CBC-MAC which has been recommended by NIST [56]. It derives two subkeys (K1 and K2) from the key and divides the input message into blocks (conforming to the cipher's block size), padding the last block if necessary. Depending on whether padding was necessary, K1 or K2 are exclusive-ored to the last block. Then CBC-MAC encryption is performed on the message blocks and the last block (or a subset of it) are returned as MAC.

1.3. Authentication and Confidentiality Modes

With the help of the aforementioned modes of operation, the provision of confidentiality and data integrity/authentication requires the use of two separate modes. As a general rule, separate keys should be used if two (or more) modes of operation are employed to process the same message. This is because a common presumption of all these modes is the sole use of the key within the specified functions. Additional use of the same key in another mode may enable various forms of attacks.

Another class of modes of operation can provide both authentication and confidentiality under the use of a single key. While an *authenticated encryption* (AE) scheme [5,38] allow to protect authentication and confidentiality of a message, an *authenticated encryption with associated data* (AEAD) scheme [61] aims to provide authenticity of additional data (associated data) as well. A good usage example for the latter is network packet protection, where the payload needs to be encrypted and authenticated, while it is sufficient to authenticate the packet header.

An additional distinction of these modes is between *one-pass* and *two-pass* schemes. While the first can provide confidentiality and authentication with a single pass over the message, the latter takes two passes (not necessarily employing the same cryptographic primitive). One-pass modes like Integrity Aware Parallelizable Mode (IAPM) [37], Offset Codebook (OCB) mode [62], and eXtended Cipher Block Chaining Encryption (XCBC) mode [22] are potentially faster than two-pass modes, but the one-pass approach is encumbered by patents. Therefore the research effort has turned towards fast two-pass modes, whose application is not hindered by intellectual property claims. The most important of these two-pass modes are described in the following.

There have been a number of newly proposed modes of operations which address the AEAD problem. NIST has recommended the Counter with CBC-MAC (CCM) mode in [54]. CCM mode is intended for use when the complete message is available for processing, i.e. it is not suited for *on-line* processing. It basically generates a MAC using the CBC-MAC mode (zero IV), and encrypts the message and the MAC in CTR mode. The associated data is just used for MAC generation but is not CTR encrypted.

The EAX mode of operation [6] has been proposed as a more flexible alternative to the CCM mode. It uses CTR mode for encryption and One-Key CBC-MAC (OMAC)

mode (which is basically equivalent to the NIST-recommended CMAC mode) for authentication. One of the advantages of EAX over CCM is that it allows to process data on-line.

The Carter-Wegman Counter (CWC) mode specified in [41] uses a variant of CTR mode for encryption. Authentication is done over the resulting ciphertext and the associated data with a universal hash function (Carter-Wegman [75]), which is based on 127-bit integer arithmetic. The mode allows for parallelization, can handle data *on-line* and is aimed at high-speed implementation in hardware and software. However, the dependence on long integer arithmetic makes CWC mode less attractive for processors of small word size.

NIST is currently working on a recommendation for the Galois/Counter mode (GCM) [57]. GCM is very similar to CWC mode—its main difference is the use of another universal hash function. While CWC mode relies on long integer arithmetic, GCM employs a universal hash function over a binary finite (Galois) field. This construction allows faster implementation in hardware as well as in software under the use of lookup tables. However, when software implementation cannot support large lookup tables, the performance degrades significantly [21].

Recently, there have been interesting proposals for new two-pass modes of operations like Counter Cipher Feedback (CCFB) mode [42] and SLC mode [2]. Both of these modes are claimed by their designers to be better suited for constrained embedded devices than other two-pass modes. However, a good estimate of the actual benefits cannot be given, as there is currently no empirical data available from practical implementations of these modes on constrained devices.

Most of the proposed block cipher modes of operation have been submitted to NIST for recommendation. A good overview of these modes can be found on NIST's Modes of Operation website [55].

1.4. Hashing Modes

Block ciphers can also be used to build hash functions. Following the popular generic Merkle-Damgård construction of hash functions [17,46,47], a compression function is used repeatedly to combine blocks of the input message with the previous output of the compression function (or an IV in the case of the first message block). Different modes of operation have been proposed to construct a compression function, e.g. Matyas-Meyer-Oseas, Davies-Meyer, and Miyaguchi-Preneel [45]. In the confidentiality and authentication modes of operation, the block cipher key was fixed for multiple invocations. The hashing modes on the other hand vary both the input block and the cipher key for each invocation of the block cipher. This means that precomputation of the key schedule of the block cipher cannot be used to increase the performance and that a block cipher with a computationally inexpensive key expansion is better suited to be employed in a hashing mode of operation.

An example for a modern block cipher-based hash function is Whirlpool [4], which employs the Miyaguchi-Preneel mode of operation. An extensive treatment of block cipher hashing modes can be found in [59].

2. Description of Important Algorithms

This section describes the most commonly used symmetric primitives in cryptography. This selection is a limited excerpt of available block ciphers, hash functions, stream ciphers, and authentication codes. We will present a short historical view of the algorithms and their use in standards and applications.

2.1. Block Ciphers

Block ciphers are symmetric-key primitives which operate on so-called blocks with a fixed number of bits. A transformation is defined which modifies the input data block using the secret key and outputs the ciphertext. Decryption is the inverse operation which allows to recover the plaintext from the ciphertext and the secret key.

The Data Encryption Standard (DES) algorithm, which was standardized in the year 1976 by NIST [51] has been the most important block cipher for several decades. Due to its limited key size of 56 bit it was first extended by Triple-DES and is now replaced by the Advanced Encryption Standard (AES) which has larger key sizes (128, 192, and 256 bits). The Rijndael algorithm was selected as the winner of an open selection process initiated by NIST and has been standardized as the AES algorithm in the year 2001 [52]. During this selection process the four algorithms MARS, RC6, Serpent, and Twofish have also been selected as finalists with no discovered severe weaknesses. Hence, these algorithms are also worth consideration in special environments like embedded systems and WSNs. RC6 has for example been shown to have very favorable properties on embedded processors with limited memory resources [23]. In comparison to Rijndael, which has the advantage of its efficient implementation on various platforms, Serpent has a higher security margin and is therefore much slower. Other interesting block ciphers for application in embedded systems are the Tiny Encryption Algorithm (TEA) and its extension (XTEA), Skipjack, and IDEA. TEA is based on very simple and fast operations which require very little code size. Skipjack is used for authentication in TinySec applications whereas IDEA is used in PGP.

Some of the above mentioned primitives are very well suited for constrained environments like WSNs. Especially the algorithms AES, RC6, and TEA are suitable for motes which do not have many computing resources available. Nevertheless, standardized algorithms like the AES algorithm should always be preferred because the security of the algorithm has been scrutinized by a large number of cryptanalytic experts. There exist several bad examples of using proprietary algorithms which were shown to be insecure. Especially in WSNs with long-time deployment and without the possibility of reconfiguration and maintenance, standardized algorithms should be preferred.

2.2. Hash Functions

Hash functions belong to the category of unkeyed cryptographic primitives and are mainly used for providing data integrity. The input of these primitives is a potentially large message which is condensed to a relatively short value. This so-called hash value can be seen as a fingerprint of the message. One important property of a hash function is the collision resistance which means that it is computationally infeasible to find any two messages that produce the same hash value. This allows to detect data manipulation due

to technical reasons or due to an attacker. A hash can be used for a message authentication code (MAC) which allows to simultaneously verify data integrity and authenticity of a message. Due to the demanding design of hash functions regarding computing power and memory resources it might be favorable to minimize the use of hash functions and consider the use of block ciphers in an appropriate mode of operation.

For a long time, the design of cryptographic hash functions has received little research effort. Most of the MD4-family hash functions were believed to be secure. So far, the best known hash functions are the MD4-family hash functions SHA-1, SHA-256, MD4, MD5, and RIPEMD. All designs base more or less on the (today insecure) MD4 algorithm which was invented by Ron Rivest in 1990. Whereas MD5 is used only in a few applications, SHA-1 is widely employed in many security protocols and applications like SSH, S/MIME, SSL, TLS, and IPSec or the keyed-hash message authentication code (HMAC-SHA1). Since the publication of an attack against a reduced version of the SHA-1 algorithm [74], the crypto community continues to improve the attacks (e.g. [11]) and endeavors to design new, more secure hash algorithms. So far NIST recommends to use the more secure SHA-2 variants (SHA-256, SHA-384, SHA-512) or to switch to alternative designs like Whirlpool, which is standardized in ISO/IEC 10118-3 [35], or Tiger. Recently, NIST has announced that a selection process similar to that of the AES algorithm is planned for a new hash standard. There are already some new upcoming hash function designs like Grindahl, VSH, LASH, RadioGatun, and Maelstrom available.

Implementing most of the above mentioned hash functions in WSNs is costly in terms of memory resources and computing power. Hence, it could be prohibitive to use them in motes. PDA-like nodes and gateways will likely have sufficient performance to calculate hashes because hash functions like SHA-1 are optimized for implementation on 32-bit microprocessors.

2.3. Stream Ciphers

The name stream cipher comes from that fact that in this symmetric cipher the plaintext bits are encrypted one at a time mostly by applying an exclusive-or operation with a key stream that depends on the current internal state. The main difference to block ciphers is that a stream cipher does not operate on large blocks of fixed length, but on individual bits. However, depending on the mode of operation, a block cipher can also be operated as a stream cipher. Typically, stream ciphers require less hardware resources and have a higher data throughput. However, stream ciphers have a bad reputation because of security problems when they are used incorrectly and because the most famous stream cipher RC4 has been shown to have some weaknesses. Although RC4 is widely applied in security protocols like SSL and WEP, it is recommended not to be used anymore in new applications. Other stream ciphers are currently used in GSM technology, e.g. A5/1 and A5/2 (both of which are broken).

The European Network of Excellence ECRYPT has started a contest called eSTREAM [18] in order to identify new stream ciphers for future standardization. In addition to a high security level, the ciphers should be suited for efficient implementation in software and/or hardware. So far there have been several interesting stream cipher designs published. The most promising approaches targeting hardware implementation are called Trivium, Grain, Mickey-128, and Phelix. The designs that aim for efficient soft-

ware implementation and which are in the final evaluation phase are Dragon, HC-256, LEX, Phelix, Py, Salsa20, and SOSEMANUK.

Using stream ciphers in WSNs could be an interesting alternative to block ciphers. Especially on motes where little data is sent at a certain time the ability to encrypt only a few bits could lead to a significant performance gain.

2.4. Key Generation

In a well-designed cryptographic system the security should solely depend on the strength of the symmetric key (Kerckhoffs' principle). If the key is disclosed or a weak key is chosen during key setup, an attacker can decrypt every secret message or forge MACs. Hence, the length of the keys and the key generation process—although it is omitted in many investigations—is an important security aspect. Good cryptographic keys are normally randomly generated using a true *random number generator (RNG)* followed by a statistical process. Commonly, the source of randomness is a physical effect like various kinds of noise. Such RNGs can be implemented on microchips either using full-custom design or a standard-cell approach. A further possibility of key generation is to use a *pseudo-random number generator (PRNG)*. This is an arithmetic algorithm that generates a sequence of values that have similar properties as true random numbers. Typically, PRNGs require a so-called seed that initializes the internal state of the generator. Generation of keys for a large number of devices which should have a relation to each other it is also possible to derive secrets from a master key. Herein, a secret master key is used to generate a new subkey by applying a key derivation function which uses as input the master secret and an identifier of the device. There also exist some "exotic" methods like *physical unclonable functions (PUF)* for generating random numbers. The principle is that even two microchips with the same layout are not identical and that it is possible to extract key material by exploiting this uniqueness.

The secret keys for nodes of a WSN can either be stored on the node during the personalization phase before operation or via a public-key protocol at runtime. Motes that do not support public-key cryptography need a pre-shared key that is stored on the device in the initialization phase of the network. This has to be done in a protected environment where it is not possible for an attacker to discover the keys. The key distribution problem, that is inherent to symmetric-key systems, and the secure storage of the keys has to be addressed by the gateways or the PDA-like nodes that support key-distribution mechanisms.

2.5. Authentication Codes

Authentication codes are similar to digital signatures and MACs in their goal of providing message integrity and authentication. However, they differ significantly in their fundamental concepts. While digital signatures and MACs can provide computational security, authentication codes are conceived to provide unconditional security. More specifically, authentication codes can give upper bounds for the probability that an attacker can construct or forge a valid message. However, these bounds only hold for maximally one authenticated message with a specific key, requiring a change of key after each message to retain unconditional security. Nevertheless, there are practical approaches to limit the amount of key material without impairing security, e.g. the Toeplitz approach. Univer-

sal hash functions can be used to construct authentication codes. Practical mechanisms include *message authentication code based on universal hashing* (UMAC) [36] and the AES-based Poly1305-AES [7].

2.6. Description of AES

The symmetric block cipher Rijndael was invented by Joan Daemen and Vincent Rijmen and became the Advanced Encryption Standard (AES) in the year 2001. The National Institute of Standards and Technology (NIST) selected Rijndael as the winning algorithm after a contest of several years for the new Federal Information Processing Standard FIPS-197 [52]. Due to its flexibility, the AES algorithm can be implemented in software as well as in hardware very efficiently. Various target platforms like small 8-bit microcontrollers, 32-bit processors, and dedicated hardware implementations are suitable. The free availability of the algorithm paved the way for deployment of AES in numerous standards like wireless LAN according to the IEEE802.11i standard [31], IPSec, and TLS.

The strength of the AES lies in its simplicity which was one of its main design principles [16]. In addition to efficient implementation on various platforms under different sets of constraints, this simplicity is realized by means of utilizing the symmetry properties at different levels and the choice of a small set of basic operations. The first level of symmetry is realized by applying the same round function to the fixed block size of 128 bits several times. The round function of the AES algorithm is outlined in Figure 1.

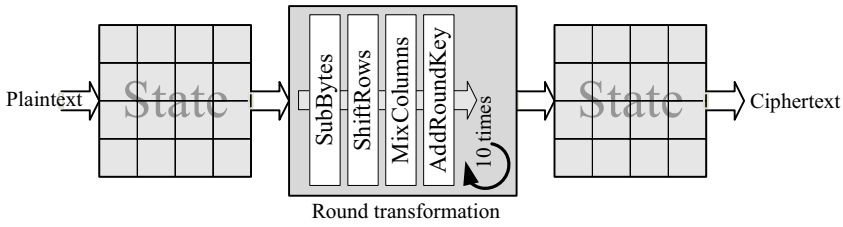


Figure 1. AES algorithm.

All operations of the AES algorithm transform a block of data which is referred to as the *State*. Internally, the State is organized as a 4×4 matrix of bytes. Within the standard, the three different key lengths 128, 192, and 256 bits are defined. Nowadays, mainly 128-bit keys need to be employed. Only for long-term security or top-secret documents 192-bit keys and 256-bit keys are used. The number of round transformations for encrypting and decrypting one input block are 10, 12, or 14 depending on the selected key size. The round transformation modifies the 128-bit State from its initial value (the input plaintext) to get the output ciphertext after the last round. The non-linear, linear, and key-dependent transformations in each round are all defined as operations over various finite fields ($\mathbb{F}_2$, $\mathbb{F}_{2^8}$, $\mathbb{F}_{2^8}[t]/(g(t))$). The operations are called SubBytes, ShiftRows, MixColumns, and AddRoundKey for encryption and InvSubBytes, InvShiftRows, InvMixColumns, and AddRoundKey for decryption. All operations scramble the bytes of the State either individually, row-wise, or column-wise. Before the first round of encryption, an initial AddRoundKey is performed while in the last round the MixColumns operation is omitted. Encryption calculates the ciphertext from the plaintext and the de-

cryption function of AES computes the plaintext from a ciphertext. The decryption operations are executed in reverse order and all round keys are also used in reverse order. In the following we will describe the AES operations in more detail. We only focus on the encryption operations because the decryption operations work very similarly.

The only non-linear operation of the AES is the SubBytes transformation. It substitutes all 16 bytes of the State individually by using the same table lookup. The contents of the table can be calculated by an inversion in the binary extension field $\mathbb{F}_{2^8}$ followed by an affine transformation. This byte substitution is often referred to as S-box operation. The possibilities to implement the S-box are manifold. In software, the 256 different values are mostly stored either in program or data memory. For hardware implementations there is a much larger design space. A detailed analysis for implementing the S-box starting from low chip area and low-power applications up to high-throughput applications can be found in [68].

The ShiftRows function is a simple operation which rotates each row of the State using a specific byte offset. The offset is given by the row index (starting at 0), which means that the first row is not rotated at all and the last row is rotated by three bytes. The implementation in software is reduced to an appropriate addressing of the bytes in the subsequent operation. In hardware, it depends on the implemented architecture whether ShiftRows migrates to a simple wiring or an appropriate addressing of the data is required.

MixColumns operates on columns of the State separately. Each column is interpreted as a polynomial of degree three with coefficients from the field $\mathbb{F}_{2^8}$. MixColumns is defined as a multiplication of this polynomial with a constant polynomial $(03 \cdot x^3 + 01 \cdot x^2 + 01 \cdot x + 02)$ modulo the irreducible polynomial $x^4 + 1$. The operation can also be expressed as a matrix multiplication of a constant 4×4 matrix with the input column. The matrix multiplication for MixColumns can be seen in Equation 1, where a_0 to a_3 are the $\mathbb{F}_{2^8}$ coefficients of the input column and b_0 to b_3 are the $\mathbb{F}_{2^8}$ coefficients of the output column.

$$\begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} \tag{1}$$

The three operations SubBytes, ShiftRows, and MixColumns constitute the substitution-permutation network of the AES algorithm. SubBytes is the substitution part to increase confusion whereas ShiftRows and MixColumns build the permutation part for increasing diffusion. The AddRoundKey function adds a round key to the State. It is an exclusive-or operation over all 128 bits where in each round another key is used. The first 128-bit round key is equal to the cipher key. All other keys are computed from the previous round key using the key expansion operation. This operation uses the S-box functionality and some constants referred to as Rcon. There are also various possibilities to implement the key expansion in software and in hardware. In systems with sufficient memory resources it is common to calculate and store the whole key schedule in advance. The other possibility is the on-the-fly key expansion where the current round key is calculated on demand during the encryption or decryption operation. This variant allows saving memory resources but possibly reduces the calculation speed.

The AES algorithm was designed in a way that it is flexible concerning the platform on which it is implemented. In a heterogeneous WSN with many different types of nodes the AES algorithm is a very good choice to secure information. Beside its good security properties it has the advantage that it can be implemented on all classes of devices very efficiently. AES can be implemented on low-end motes with 8-bit microcontrollers or in dedicated hardware for low chip area and low power consumption. It was even shown that it can be implemented on passively powered devices like RFID tags [19]. Faster and energy-efficient implementations of AES are suitable for PDA-like nodes where a higher data throughput is necessary. Even for large gateways which have enormous data rates AES can be implemented either on high-end microprocessors or as a coprocessor, e.g. on an acceleration card.

3. Implementation of AES

The AES algorithm allows efficient implementation on processors of different word size and dedicated hardware for high performance or low power. Another interesting implementation aspect is the hardware/software co-design approach in the form of instruction set extensions to general-purpose processors. In regard to implementation approaches and properties, the AES algorithm is probably amongst the best-studied cryptographic algorithms. The AES Lounge [28] provides a good overview of the most important publications regarding different implementation options.

3.1. Software

AES has been designed to allow efficient implementation on 8-bit and 32-bit platforms. The natural granularity of operations—mainly determined by the S-box operations—is set on bytes. Moreover, all round transformations can be done with simple shifts/rotations, table lookups, and Boolean operations, which eases implementation on low-cost 8-bit microcontrollers. On 32-bit platforms, larger lookup tables can be employed to increase performance. The book by the authors of the AES algorithm [16] introduces the most important design approaches for software implementations on 8-bit and 32-bit processors.

An important design issue is precomputation of the key schedule vs. on-the-fly key expansion. If there is only a small number of keys in use (e.g. on a WSN mote), a precomputed key schedule can increase the performance. On devices which have to handle a large number of different keys simultaneously (e.g. a gateway in a WSN with group-key or pair-wise key distribution), storage of all key schedules might become prohibitive, so that an on-the-fly key expansion could be preferred.

In the following we will give an overview of the most important implementation aspects for AES on 8-bit, 16-bit and 32-bit processors. We include performance figures of optimized implementations on representative embedded processors for each class, which can be found in WSN nodes. The cited figures all refer to AES with a 128-bit key (AES-128), as this will be the most commonly used algorithm variant. However, AES with 192-bit or 256-bit keys differ basically only in the number of executed rounds, so that fairly accurate performance estimates for these variants can be derived from the given figures by simple scaling ($\times 1.2$ for AES-192 and $\times 1.4$ for AES-256). In the normal case,

an optimized implementation of AES-128 should be able to deliver the performance and security required by most WSN applications.

3.1.1. 8-bit Microcontrollers

Low-cost 8-bit microcontrollers are at the heart of many WSN motes, e.g. Crossbow Mica2 and MicaZ [14]. The same microcontroller architectures are contained in many modern smartcards, which have been used as target platforms for a number of publications dealing with AES implementation efficiency.

The SubBytes transformation can be efficiently done with a 256-byte S-box table. It is indexed with a State byte and yields the substituted byte value conforming to the AES S-box. This table is normally stored as a constant in program memory. The S-box table is not only required for SubBytes in AES encryption but also for the key expansion. If AES decryption is needed, an additional 256-byte inverse S-box table is required for the InvSubBytes transformation. The arithmetic definition of the AES S-box allows the dynamic generation of these tables at runtime. In this case the tables can be stored in RAM and need not be included in the program memory. However, the AES program code is extended by code for generating the tables. Depending on the specific microcontroller, the size of the table-generating code is around 200 bytes. When only a single S-box table is required, dynamic generation normally has only a small advantage over static storage in program memory. If both S-box and inverse S-box are needed, dynamic generation can help to reduce program size. Of course the choice of static vs. dynamic tables might depend on other parameters, e.g. the access speed to the different types of memory or whether there is time to perform the dynamic generation of the tables.

ShiftRows is can normally done in combination with SubBytes through the proper addressing and storage of the substituted State bytes. The conventional approach is to employ two 16-byte memory regions, where one holds the State prior to SubBytes and ShiftRows and the other is used to store the State after these two transformations. The latter memory region is subsequently used as input for MixColumnns. It is also possible to perform the complete AES on a single 16-byte memory region, by doing SubBytes "in-place" (i.e. each byte of the State is overwritten with its substituted counterpart) and performing the byte-wise rotate of ShiftRows on the State explicitly. Another approach is to do SubBytes "in-place" and cater for ShiftRows implicitly in the MixColumns transformation by appropriate addressing of the State memory. This strategy exploits the fact that after four subsequent ShiftRows on the State, all State bytes are again at their original positions. Therefore, there are only four differently shifted version of the State possible. Depending on the index of the current round, MixColumns then needs to use one of four different addressing schemes to cater for ShiftRows.

The basic operations for implementing MixColumns are addition and doubling of elements of $\mathbb{F}_{2^8}$. Addition consists of a simple bitwise exclusive-or, while doubling can be achieved with a sequence of a few simple steps. The doubling operation (normally denoted *xtime*) can be done with a left shift, followed by conditional addition of the lower eight coefficients of the irreducible polynomial (1B), whenever the shifted out bit was one. Care must be taken that xtime always takes constant time, as otherwise timing attacks [40] would become possible. Multiplication with a constant can be calculated by a series of additions and doublings. In MixColumns each byte of a State column is multiplied with several constants (01, 02, and 03). These multiplications can be combined, so that four xtime operations are sufficient for each column [16].

For the InvMixColumns transformation in decryption, the constants are larger (09, 0B, 0D, and 0E), requiring eight xtime operations per State column. The constant polynomial for this transformation can be expressed as a product of the polynomial for MixColumns and another polynomial: $0B \cdot x^3 + 0D \cdot x^2 + 09 \cdot x + 0E = (04 \cdot x^2 + 05) \cdot (03 \cdot x^3 + 01 \cdot x^2 + 01 \cdot x + 02)$. Consequently, the constant matrix for InvMixColumns can be split up into two matrices, where one is identical to the one of MixColumns, and the other is of a very simple structure. The matrix for InvMixColumns and its decomposition is shown in Equation 2.

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} = \begin{bmatrix} 05 & 00 & 04 & 00 \\ 00 & 05 & 00 & 04 \\ 04 & 00 & 05 & 00 \\ 00 & 04 & 00 & 05 \end{bmatrix} \times \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \tag{2}$$

Following Equation 2, InvMixColumns can be split up into an application of MixColumns and another operation which also requires four xtime operations [16]. Note that the second operation can be performed before or after MixColumns. Although the number of xtime operations is equal for this approach, an AES implementation requiring both encryption and decryption can be reduced in size by using MixColumns for both.

Code size can be reduced by packing xtime into a separate subroutine, whereas performance can be increased by inlining it into the MixColumns/InvMixColumns transformation. Another memory-performance tradeoff can be done by performing xtime with a lookup into a 256-byte table, which holds the double of each element of $\mathbb{F}_{2^8}$. It is also possible to have additional tables which contain the results of multiplication with other constants.

AddRoundKey can simply be implemented with exclusive-or operations. The key expansion can reuse the S-box table, requiring just a few additional constants (Rcon). Most of the key expansion can also be done with exclusive-or operations.

Table 1 lists reported performance figures for AES-128 encryption and decryption on 8-bit platforms. For quick reference, we have also added the date of publication. Most of the implementations include on-the-fly key expansion and can only perform encryption. Requirements for RAM typically range around a few dozen bytes, while code size is normally a few KB. Note that the cycle count for 8051 implementations refers to instruction cycles. Depending on the actual microcontroller implementation, one instruction cycle requires a specific number of clock cycles, e.g. the original Intel 8051 microcontroller requires 12 clock cycles.

3.1.2. 16-bit Microcontrollers

In comparison to the permeation of 8-bit and 32-bit microcontroller and microprocessors in the whole embedded systems field, 16-bit architectures occupy only a niche segment. Nevertheless, there are some WSN motes which feature 16-bit microcontrollers, e.g. the Tmote Sky from Moteiv [49]. AES has not been designed towards 16-bit architectures and except for [33] there are virtually no performance figures available regarding implementation on such platforms.

For AES, a 16-bit data path offers little advantage over an 8-bit data path. SubBytes, ShiftRows, and MixColumns need to work on 8-bit values in a similar way as on 8-bit

Table 1. Reported AES/Rijndael performance on 8-bit platforms

Platform	Year	Encr. / Decr. cycles	Details
68HC08 [15]	1998	8,390 / N/A	On-the-fly key expansion
68HC05 [39]	1999	14,945 / N/A	On-the-fly key expansion
Z80 [64]	2000	25,494 / N/A	Excluding key expansion (10,318 cycles)
8051 [15]	1998	3,168 / N/A	On-the-fly key expansion
8051 [1]	2001	7,542 / N/A	Not optimized
8051 [33]	2006	3,905 / 5,876[a]	On-the-fly key expansion
AVR [63]	2003	7,498 / 11,548	On-the-fly key expansion
AVR [33]	2006	4,009 / 6,073	On-the-fly key expansion

[a]These performance figures include overhead for a flexible programming interface (requiring about 650 cycles), which can be omitted for a high-speed implementation.

architectures. In fact, there might be some overhead due to extraction of 8-bit values from or storage of two 8-bit values into the 16-bit registers. The only clear advantage of the 16-bit data path is for the exclusive-or operations in MixColumns, AddRoundKey, and the key expansion.

The implementation for a TI MSP430 reported on [33] requires 5,432 cycles for AES-128 encryption and 8,802 cycles for AES-128 decryption. It uses an on-the-fly key expansion and has a codesize of about 2,5 KB.

3.1.3. 32-bit Microprocessors

Some high-end motes like the Crossbow Imote2 [14] feature 32-bit microprocessors. Embedded 32-bit processors will also be predominant in PDA-class devices, whereas gateways will have at least 32-bit desktop or server class processors.

AES can be realized on 32-bit platforms with the help of one or more 1 KB-sized lookup tables, which are commonly denoted as T-boxes [16]. Each T-box consists of 256 32-bit entries. A State byte at the beginning of the round is used as index into a T-box, and the result is the contribution of this byte to the respective State column after MixColumns. A complete AES round (excluding round key loading or expansion) can therefore be performed with 16 lookups (using four different T-boxes), 12 exclusive-or operations for combining the four contributions for each State column and 4 exclusive-or operations for AddRoundKey. The last round can be calculated conventionally or with four separate tables. Moreover, as the four T-boxes of a set are only rotated versions of each other, memory can be preserved by the use of only a single T-box at the expense of three additional rotate operations per State column. On a platform where rotates are cheap (e.g. ARM architectures), this memory preservation measure might even be achievable with virtually no overhead.

For AES decryption, different T-boxes and a transformed key schedule (most round keys transformed with InvMixColumns) are required. As InvMixColumns is rather costly to compute, the T-box-approach is only efficient in combination with a precomputed key schedule.

Like the S-box table, the T-box tables can be stored statically in program memory or calculated dynamically in RAM. For some settings, the T-box approach might not be suited, e.g. when memory is limited, memory performance is low or cache pollution

is undesirable. A conventional implementation with S-box tables might be preferred in such situations.

On 32-bit platforms, the State is normally stored in four 32-bit registers, where each State column is held in a register (*column-wise* approach). In a conventional S-box implementation, SubBytes and ShiftRows are still performed on State bytes, which requires extraction from the respective 32-bit register and merging of the four substituted bytes into a new State column. An advantage of 32-bit platforms is that MixColumns can be performed on complete columns instead of individual bytes. When a hardware integer multiplier is available, xtime can be done efficiently on four bytes in parallel. The exclusive-ors of AddRoundKey and the key expansion also benefit from the 32-bit data path.

Bertoni et al. have proposed a method for speeding up InvMixColumns for conventional S-box implementations by processing the AES State in a *row-wise* fashion [8]. Their solution requires a slightly more complex key expansion which is no significant disadvantage if the key schedule is precomputed. Compared to the column-wise approach, the solution of Bertoni et al. is about as fast for encryption but significantly faster for decryption.

Some reported performance figures for embedded processors with ARM or SPARC architectures are given in Table 2. If T-box tables are employed, their total size is given in the last column of the table. All other implementations just use the conventional approach with S-boxes. Code size ranges usually ranges between 1 KB and 2 KB without T-boxes. Extensive use of T-boxes can increase the code size to up to 20 KB [70]. However, more than 4 KB of T-boxes normally bring not much advantage on these embedded processors. If the tables get too large, the increase in cache misses can even lead to a decrease in performance [71].

Table 2. Reported AES/Rijndael performance on 32-bit platforms

Platform	Year	Encr. / Decr. cycles	Details
ARM [25]	1999	2,889 / N/A	On-the-fly key expansion
ARM [25]	1999	1,467 / N/A	On-the-fly, 1 KB T-box
ARM7TDMI [8]	2003	1,675 / 2,074	Precomputed key schedule
ARM9TDMI [8]	2003	1,384 / 1,764	Precomputed key schedule
ARM7TDMI [8]	2003	2,074 / 2,378	On-the-fly key expansion
ARM9TDMI [8]	2003	1,755 / 1,976	On-the-fly key expansion
SPARC V8 [70]	2006	1,637 / 1,955	Precomputed key schedule
SPARC V8 [70]	2006	2,239 / 2,434	On-the-fly key expansion
SPARC V8 [70]	2006	1,097 / 1,262	Precomputed, 4 KB T-box
SPARC V8 [70]	2006	1,497 / 5,939	On-the-fly, 4 KB T-box

3.1.4. A Note on Side-Channel Attacks

Side-channel attacks (SCAs) are an important category of implementation attacks, which can threaten the security of cryptographic devices. They are based on the observation that secret information (e.g. cryptographic keys) can leak through physical values like execution timing, power consumption or electromagnetic emanations while the device performs cryptographic operations, e.g. AES. There have been some publications on AES software implementations including countermeasures for SCAs, e.g. [1,26,72]. For a comprehensive monograph on power analysis attacks on smartcards refer to [44].

3.2. Hardware

The implementation of the AES algorithm in hardware requires an exhaustive design-space exploration regarding chip area, data throughput, power consumption, and several other design parameters. The selection of the target technology is one of the first decisions that has to be taken. In the area of wireless sensor networks we can either work on FPGA platforms for gateways or on ASIC implementations for motes. Hence, we try to keep the analysis of possible architectures more general. As the AES algorithm is very flexibly designed, many different architectures are possible. The hardware architecture for a mote, which requires very low die size, is totally different from a PDA-like node which has a limited energy budget available. In gateways, it could be necessary to have high data throughput while the required hardware resources are of minor importance. In this section we will describe three different architectures. The first will be a small-area implementation using an 8-bit architecture with a limited power budget as it would be available from harvesting energy from a solar panel in a mote. The second approach describes a faster module which has the best energy efficiency implemented as a 32-bit architecture. Such an AES implementation could be applied in a PDA-like node with a battery power supply. The last approach will describe a 128-bit architecture for gateways which require high performance.

The word size of the architecture mainly defines the die size of the circuit. Whereas an 8-bit implementation has reasonably only a single S-box to compute the SubBytes transformation, a 32-bit architecture requires four S-boxes. Because the S-box is—apart from the memory for the State and the key—the largest part of an AES hardware module, an 128-bit architecture with 16 S-boxes requires significantly more hardware resources. This section will describe the possible trade-offs concerning chip area and data throughput and will summarize the results of the given implementations.

3.2.1. Lightweight Implementation

In this section we will present the low-power AES implementation designed by one of the authors and presented in [19]. The term "lightweight" in the context of AES hardware implementation means that low die size and low power consumption are the major design goals. All design options were evaluated regarding their impact on the silicon size and on the power efficiency. The evaluation is based on synthesis results and circuit-level simulations. In the following, we present the architecture of the AES module according to [19] and discuss details of the circuit that contribute to its efficiency.

The described AES implementation has a fixed key size of 128 bits. This reduces the number of rounds to ten and the required memory for the State plus the round key to 256 bits. Low-power requirements restrict a 128-bit width of the data path. Even 32-bit implementations would be rather costly in terms of power consumption. Hence, an 8-bit architecture of AES is the best choice because all operations consume significantly less power than 32-bit operations. The architecture of the AES module can be seen in Figure 2.

The main parts of the AES are the controller, the RAM, the data path, and the IO module. The IO module has a microcontroller interface that allows to use the AES module as a coprocessor. The controller accepts commands from the IO module and provides control signals for RAM and the data path to sequence AES operations. The controller is realized as a hard-wired finite-state machine. This allows optimization of the efficiency

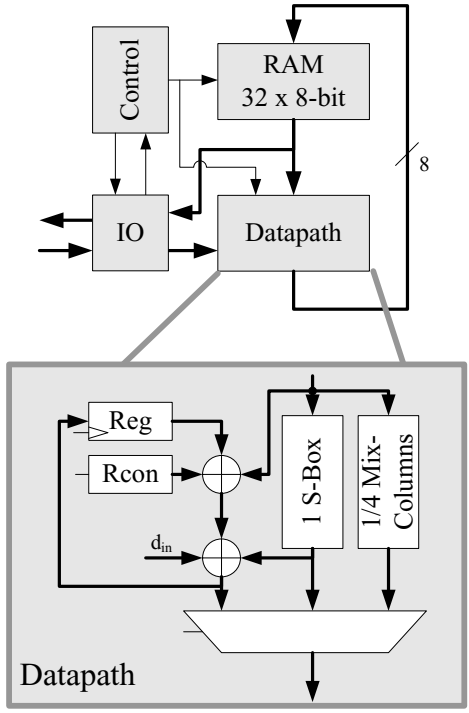


Figure 2. Architecture of 8-bit AES module.

concerning low power consumption and low die size. It mainly consists of a 4-bit round counter and address registers for addressing rows and columns of the RAM. These counters are implemented as shift registers using one-hot encoding. One-hot encoding ensures that changes of the state cause only two signal transitions. Moreover, one-hot encoding reduces undesired glitching activity of control signals.

The finite-state machine sequences the ten rounds consisting of the operations AddRoundKey, ShiftRows, SubBytes, MixColumns, or their inverse operations. Additionally, all round keys are generated just-in-time for every round of the AES. This on-the-fly round key generation helps to reduce the necessary storage capacity of the RAM block to 256 bits. The first 128 bits store the actual State and the second 128 bits hold the current round key. As no spare memory is present for storing intermediate values, the controller has to assure that no State byte nor key byte is overwritten if it is needed again during calculation.

The RAM is single ported to ease silicon implementation. It is realized as a flip-flop based memory. The extensive use of clock gating lowers the power consumption. Additionally, this standard-cell based approach facilitates the physical realization compared to using a dedicated RAM macro block.

The data path of the AES module contains combinational logic to calculate the AES transformations SubBytes, MixColumns, AddRoundKey, and their inverse operations (see Figure 2). The ShiftRows/InvShiftRows transformation is implemented by appropriate addressing of the RAM. It is executed when results of the S-box operation are written back.

The remaining components of the data path are the submodule Rcon, some exclusive-or gates, and an 8-bit register to store intermediate results during key scheduling. Rcon is a circuit which provides constants needed for the key schedule. The exclusive-or gates are needed for round key generation and are reused for adding the round key to the State during the AddRoundKey transformation. Additionally, the data input and key input are handled by the data path.

The encryption or decryption of 16-byte blocks works as follows. The 16 bytes of input data are successively written to the RAM through the 8-bit microcontroller interface followed by the 16 bytes of the key. The initial AddRoundKey operation is performed during the loading of the key. For decryption, the last round key must be loaded because all round keys are calculated in reverse order. Issuing the start command to the control input starts encryption or decryption. The ten AES rounds with the functions SubBytes, ShiftRows, MixColumns for encryption and the functions InvSubBytes, InvShiftRows, InvMixColumns for decryption are performed according to the algorithm specification. While computing AddRoundKey, which is equal for encryption and decryption, the subsequent round key is derived from its predecessor using the S-box, Rcon, and the exclusive-or functionality of the data path. Encryption can be done within 1,032 clock cycles including the IO operation. Decryption requires 1,165 clock cycles (some overhead in incurred due to addressing overhead of the concrete architecture).

A significant advantage of the 8-bit architecture of the design is to reduce the number of S-boxes from at least four of a 32-bit implementation to one instance. This reduces the required silicon resources. The single S-box is used for the SubBytes and the InvSubBytes operation as well as for computation of the key schedule. The S-box is the biggest part of the AES data path. There are several options for implementing an AES S-box. The most obvious option is a 512×8-bit ROM to implement the 8-bit table lookup for encryption and decryption. Unfortunately, ROMs do not have good properties regarding low-power design. A particularly suitable option is to calculate the substitution values using combinational logic as presented in [77]. One feature of this S-box is that it can be pipelined by inserting register stages. Our S-box implementation uses one pipeline stage which shortens the critical path of the S-box and lowers glitching activity. Furthermore, this pipeline register is used as intermediate storage during the ShiftRows operation. During the substitution of one byte, the next byte is read from the memory. The substituted byte is written to the current read address. By choosing the read addresses properly, the SubBytes and the ShiftRows operation are combined and ShiftRows degrades to mere addressing.

Another innovative solution is the calculation of the MixColumns and InvMixColumns operations. We achieved to build a submodule which calculates one fourth of the MixColumns and InvMixColumns operation in one cycle. Instead of using four multipliers as stated in [76] we use one modified multiplier.

3.2.2. Compact Implementation

An example of a compact AES hardware module is given in [43] (standard data unit). It uses a highly regular architecture of 16 data cells and four hardware S-boxes. While the S-boxes are used for SubBytes, InvSubBytes and on-the-fly key expansion, the data cells (and their flexible interconnect) are able to perform all other round transformations. Therefore this architecture has a hybrid 32-bit/128-bit data path: 32-bit for SubBytes/InvSubBytes to preserve area, and 128-bit for all other transformation to increase

performance. Moreover, the regular design facilitates the extension of the data path, as shown for the high-performance data unit in [43], which has 16 S-boxes.

3.2.3. High-Speed Implementations

The applications of high-speed AES hardware are various. Data rates in the Gigabit range are necessary in network protocols like IPsec or TLS. While in wireless computing the data rates increase fast, traditional wired networks and optical networks use data rates up to 20 Gbps (OC-256). The possible implementations of the AES for high-speed requirements are manifold. We will investigate two different high-speed architectures of the AES algorithm. The first implementation can be used for non-feedback and feedback modes of operations. It is an iterative approach for encryption and decryption targeted for standard cell technology. The second implementation is a fully unrolled architecture for maximum data throughput for non-feedback modes of operations. The target technology is an FPGA. Both architectures are designed for 128-bit key size and are described in the following.

The maximum data throughput can be achieved when the data path width of the AES architecture is 128 bits. The iterative approach allows the computation of one AES round per clock cycle. This brings the maximum hardware utilization while the required hardware resources are relatively small compared to an unrolled architecture. Besides the architecture for the data unit, the key unit implements the key scheduling. Here also one round key is generated per clock cycle. The architecture of the data unit can be seen in Figure 3 and is described in detail in the following.

The data unit of the AES module contains combinational logic to calculate the AES transformations SubBytes, ShiftRows, MixColumns, AddRoundKey, and their inverse operations InvSubBytes, InvShiftRows, InvMixColumns. Figure 3 shows the interconnection of the different parts in the circuit which allows the calculation of one round in a single step.

The SubBytes transformation operates independently on each byte of the State using a substitution table (S-box). Sixteen instances of the S-box circuit are used in this architecture. Each one can be realized using a composite field implementation e.g. like the one according to Wolkerstorfer el al. [77]. As the sequence for AddRoundKey and MixColumns are exchanged for encryption and decryption, the exclusive-or operation with the key for decryption follows immediately after the InvShiftRows operation while for encryption it is applied as a last step in the round transformation. A multiplexer circuit is used which allows to select the output of the MixColumns circuit, skip this MixColumns circuit, or use the data input port. The MixColumns and the InvMixColumns operations are implemented using four finite-field constant multipliers.

The key unit performs the key expansion algorithm. Starting with the cipher key as input it delivers one round key per clock cycle. For decryption either the last round key has to be supplied or the last key is calculated during a precomputation phase of ten cycles. In each step of the key schedule the first 32-bit word is rotated and the S-box lookup has to be applied. Therefore, four instances of the S-box are required. A constant called Rcon is also added to the output of the S-box circuit with an exclusive-or gate. This result is combined with the last 32-bit word with an exclusive-or. The other three new 32-bit words are calculated from the old values through an exclusive-or operation with the other inputs according to the algorithm. The different order of the inputs for

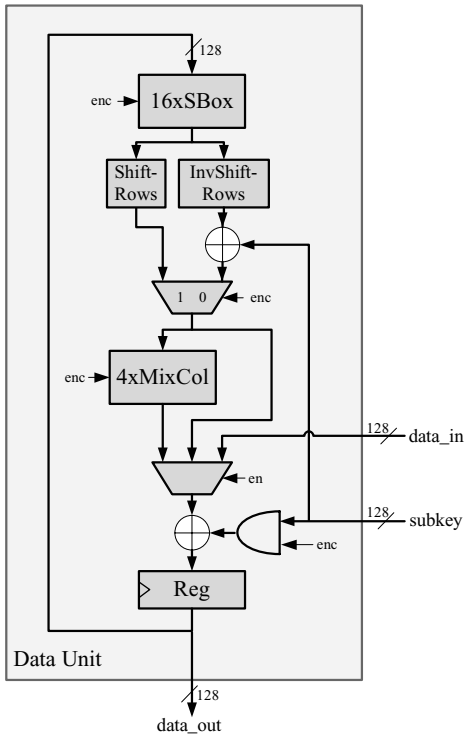


Figure 3. Data unit of 128-bit AES architecture.

the exclusive-or gates of encryption and decryption is realized using multiplexers. One 128-bit register stores the current round key.

For gateways where many data streams have to be encrypted simultaneously high data rates are necessary. Because of the small number of such high-speed modules, FPGAs are commonly used as target technology. The highest data throughput for AES encryption can be reached when all ten rounds are unrolled and pipelining registers are introduced between each round. Therefore, the fully unrolled architecture which can be seen in Figure 4 is proposed. A property of this unrolled architecture is that only non-feedback modes of operation like CTR mode or the encryption of independent data streams can be done at full speed.

The architecture presented in Figure 4 consists of an initial round which is an exclusive-or of the original key with the data input followed by nine instances of the round transformation. The last round instance does not include the MixColumns circuit because the operation is not required in the last round. For an FPGA target technology, the 16 S-box circuits can be realized as a table lookup in dedicated Block RAMs. The ShiftRows operation is again a fixed rewiring from the outputs of the S-box to the input of the following circuit. The MixColumns architecture is the same as explained above and is usually realized in FPGA LUTs. The ten different round keys have to be generated in advance and can be stored on the FPGA in Block RAMs. Table 3 cites typical implementation figures for AES hardware modules that use standard-cell technology with different data path widths.

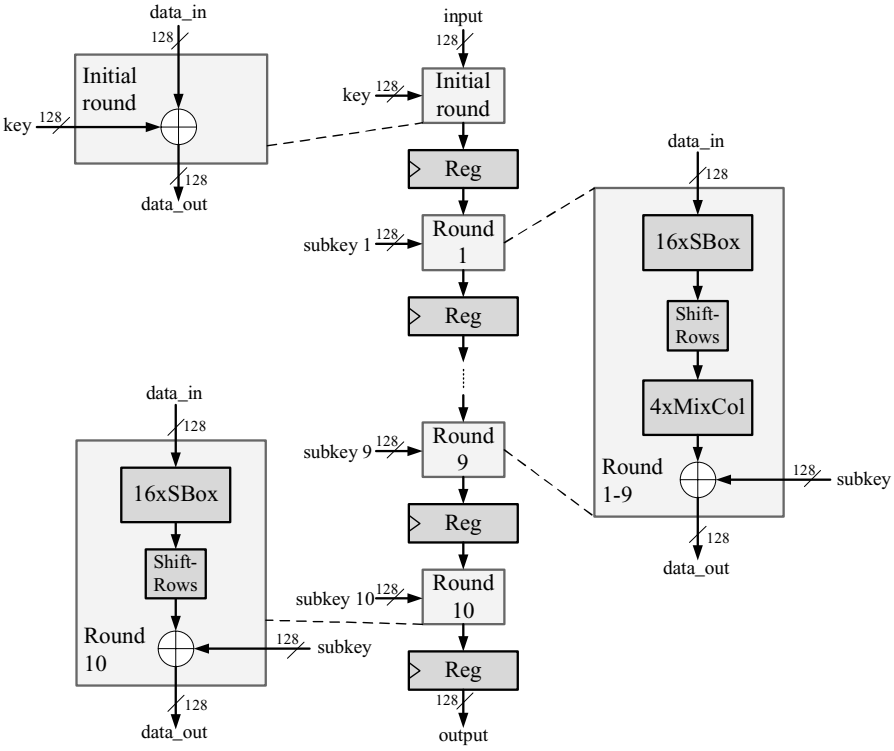


Figure 4. Fully unrolled AES architecture.

Table 3. Result of AES hardware implementations.

Architecture	Technology	Chip area	Current consumption (@ 100 kHz)	Clock cycles
8-bit [19]	0.35 μm	3,400	8.18 μA @ 3.3 V	1,032
32-bit [43]	0.35 μm	8,930	30 μA @ 3.3 V	64
128-bit [27]	0.18 μm	37,038	60 μA @ 1.8 V	11

3.3. Instruction Set Extensions

While software implementations of cryptographic algorithms normally offer the highest degree of flexibility, their performance or energy efficiency may not be sufficient for some of the embedded systems, which are part of a WSN solution. The traditional way to overcome these limitations is the integration of a cryptographic coprocessor which can take over most of the cryptographic workload. Cryptographic hardware is typically much faster and more energy efficient than software running on an embedded processor, and it might—depending on the application—also help to reduce the memory requirements of a cryptographic algorithm.

Instruction set extensions (ISEs) are a hybrid approach, where custom cryptographic instructions are added to the main processor itself. This solution has been highly successful in the application domains of multimedia (e.g. Intel's MMX and SSE technology) and digital signal processing (leading to the development of DSPs), so it comes at no surprise that instruction set extensions have also been proposed for cryptography. It

has been shown that custom cryptographic instructions can lead to a considerable reduction of processing time and—in consequence—energy consumption. Moreover, memory requirements can also be reduced significantly.

The ISE solution is in line with the modern approach to System-on-Chip design, which increasingly involves configurable processors (e.g. Tensilica Xtensa [67], MIPS CorExtend [48], ARC ARChitect [3], CoWare Processor Designer [13]). In the last years, support for AES on programmable processors has been investigated for both *application-specific processors (ASIPs)* as well as *general-purpose processors (GPPs)*. In the following we will give an overview of the work performed in this field.

Burke et al. [10] reported on the development of custom instructions for several of the AES candidates. The authors proposed a 16-bit modular multiplication, bit-permutation support, several rotate instructions, and an instruction to facilitate address generation for memory table lookups. The follow-up work of Wu et al. [78] lead to the development of the cryptographic coprocessor CryptoManiac. This is a *Very Long Instruction Word (VLIW)* machine, which can execute up to four instructions per cycle. Instructions with a short latency (e.g. like logical or arithmetic instructions) can be dynamically combined and executed in a single cycle. This feature requires instructions to have up to three source operands.

Another cryptographic processor with a VLIW architecture and support for AES is the Cryptonite [58]. AES is supported with a set of special instructions which are able to perform byte-permutation, rotation and exclusive-or operations. Most of AES is done with parallel table lookups using dedicated memories.

Fiskiran and Lee [20] investigated how hardware lookup tables can be included in a processor as a measure to accelerate different symmetric ciphers (including AES). The authors propose the inclusion of on-chip scratchpad memory in order to support parallel table lookup. They examined data path widths of 32, 64 and 128 bit with 4, 8 and 16 tables, respectively, where each table consists of 256 entries of 32-bit each (i.e. is 1 KB in size).

The PLX general-purpose RISC architecture has been extended by Irwin and Page in order to support AES [34]. Additionally, the authors examined the use of multimedia extensions of a PLX processor with a data path width of 128 bits for implementing AES with a minimal number of accesses to memory. However, these concepts do not map very well to processors with a smaller data path.

Ravi et al. [60] have investigated the automatic generation of instruction set extensions for cryptographic algorithms (including AES) using the 32-bit Xtensa processor from Tensilica. Nadehara et al. [50] proposed a custom instruction for AES which calculates most of the round transformations for a single State byte. This approach performs the round (T-box) lookup of fast AES software implementations on 32-bit platforms into a dedicated functional unit. Several instructions for AES have been proposed by Bertoni et al. [9], and performance figures for an Intel StrongARM processor have been given by the authors.

Tillich et al. [69] have investigated the application of instruction set extensions for public-key cryptography to speed up AES implementations. Another work has focused on reducing memory requirements of AES software using a single low-cost custom instruction [71]. More AES custom instructions for 32-bit processors have been proposed in [70]. This work described different instructions which are fit to be implemented independently or in combination in order to allow trade-offs between performance and sili-

con area. In the following, we will describe the extensions proposed in [70] to give an idea of the whole concept of using ISEs for AES and cryptography at large.

There are two types of instructions with a focus either set on low cost (resulting in a moderate increase of performance) or on high performance (which require more silicon area). The high-performance variant utilizes the full 32-bit data path for all round transformations. Figure 5 depicts the structure of a functional unit, which could be included in the execute stage of a processor's pipeline to implement the functionality for the AES extensions. The $\mathbb{F}_{2^8}$ constant multiplier is used for MixColumns and the four parallel S-box units for SubBytes (and can also be used in the key expansion). The custom wiring of the bytes of the instruction operands allows to perform ShiftRows implicitly with the other transformations at no additional cost. AddRoundKey can be efficiently performed in the original ALU of the processor. The hardware cost of the high-performance AES extensions is only about 3 kGates [70]. The functional units can be based on optimized hardware implementation of AES transformations for the S-box [12] and MixColumns [76].

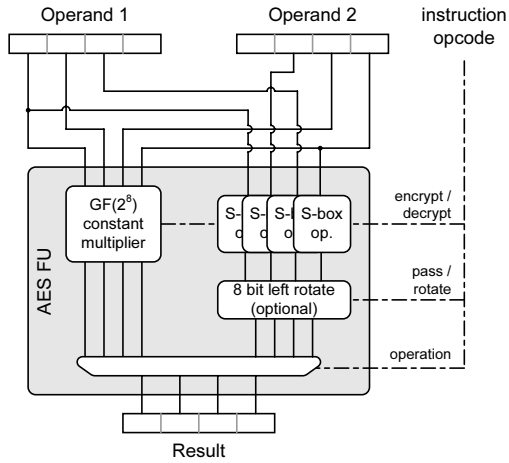


Figure 5. High-performance instructions set extensions for AES ([70]).

The low-cost extensions proposed in [70] only produce 8-bit results per cycle, but can be implemented with a smaller functional unit. The total overhead in silicon area is only a few hundred gates.

The performance for AES-128 encryption and decryption which can be achieved with ISE on single-issue embedded processors is shown in Table 4. The code size of such implementations is typically below 1 KB. The implementation in [71] makes partial use of instruction set extensions for arithmetic over binary extension fields originally conceived for Elliptic Curve Cryptography (ECC); thus demonstrating the potential flexibility of the ISE approach. The opportunities for code size reduction are shown in [70], where an AES-128 encryption with on-the-fly key expansion is reported with a mere code size of 260 bytes.

3.3.1. Application of ISE for WSN

Up to now, instruction set extensions have only been proposed for processors of a word size of 32 or more bits. WSN devices like high-end sensor nodes (PDA-like nodes) and

Table 4. Reported AES/Rijndael performance on 32-bit platforms with instruction set extensions.

Platform	Year	Encr. / Decr. cycles	Details
RISC-like [50]	2004	$\sim$ 360 / 360	Precomputed, estimated cycle count
ARM [9]	2006	311 / N/A	Precomputed key schedule
ARM [9]	2006	497 / N/A	On-the-fly key expansion
SPARC V8 [71]	2005	612 / 881	On-the-fly key expansion, ECC
SPARC V8 [70]	2006	196 / 196	Precomputed key schedule
SPARC V8 [70]	2006	226 / 262	On-the-fly key expansion

gateways could make use of a cryptographically enhanced general-purpose processor. On the one hand, the increased energy efficiency of a cryptographic processor will lengthen the lifespan of battery-powered devices. On the other hand, secure communication can be extended to a much higher number of WSN motes if the gateways are able to handle cryptographic workloads more efficiently.

Due to the flexibility of cryptographic instruction set extension, the inclusion of a set of custom instructions for AES opens up a range of options for efficient implementation in regard to e.g. key agility or mode of operation. A WSN node equipped with such an enhanced processor could offer a good mix of performance, flexibility and resource efficiency and thus be of tremendous use for many different application scenarios.

4. WSN Products with Support for Cryptography

Coming to the end of this chapter we give a short overview of available products that support cryptographic operations, referring to the requirements taken from our example application of the telematic WSN.

For the gateway nodes the choice is not too hard. Every standard PC-like terminal with a state-of-the-art CPU and memory is capable of performing the necessary cryptographic operations with reasonable performance. It is not expected that for gateway communication with PDA-like nodes, special cryptographic equipment is necessary due to performance considerations. For workstations in the backbone that communicate with a high number of other gateways, accelerators may be necessary. A variety of standard products is available for this purpose, like the IBM 4758 PCI Cryptographic Coprocessor that accelerates crypto operations and provides memory for secure key storage.

For PDA-like devices that need to compute cryptographic operations only with a relatively low throughput, software implementations of crypto primitives would be sufficient. Cryptographic libraries are available on the market for this purpose e.g. the Java Cryptography Extension IAIK-JCE from SIC [65]. Devices which require high throughput (e.g. data aggregating nodes communicating with many motes) might need a customized processor (e.g. Tensilica Xtensa [67]) featuring a coprocessor or instruction set extensions.

Due to the restricted resources for the motes, the choice for the correct product in such an application is the most difficult of the three different device classes. Unfortunately the availability of sensor nodes with security features on the market is currently not yet satisfying. On the one hand there are sensor nodes for military applications available which have high-security measures embedded. Information about the applied protection is scarce and the price for such nodes does not fit our suggested telematic application

(nor many typical civil applications). On the other hand there are low-cost motes available that support IEEE 802.15.4 and ZigBee. IEEE 802.15.4 is a low data rate wireless communication standard [30]. The goals when defining this standard were low power consumption, robustness and low cost. It covers the PHY and the MAC layer. The ZigBee specification [79] is defined on top of IEEE 802.15.4. The IEEE 802.15.4 standard uses AES-128 in CCM mode for authentication and confidentiality.

The first generation ZigBee motes were mainly based on 8-bit microcontrollers, where the AES functionality to fulfill the standard mainly is embedded into the radio module (e.g. Chipcon CC2420 which uses a hardware module to compute AES-128, or the radio module MC13213 from Freescale that embeds another 8-bit microprocessor which performs AES-128 in its firmware). An example for these motes is Crossbow's MicaZ mote [14], using an 8-bit controller with 128 KB of program memory and 512 KB flash memory for sensor data. The TriBee mote from Tritech [73] is a similar product using also an 8-bit controller with 60 KB flash memory. Another type of those low-cost products is the Tmote Sky from moteiv [49]. It features a 16-bit RISC controller running at 8 MHz with 10 KB RAM, and 1 MB of measurement memory. AES hardware support is again embedded into the radio module. Expanding the security functionality of those motes in SW is not possible with reasonable performance, since the access to the HW-crypto module is only possible for the radio module, but not for the main controller. The mote product familiy JN5139 from Jennic resolves this problem. It features a 32-bit RISC controller running at 16 MHz, where the radio module is coupled to the controller in a way so that access to the AES module is possible also for programs running on the controller core.

Newer version of ZigBee motes, like the Imote2 from Crossbow use similar radio modules but more powerful processors, like an Intel 32-bit controller that includes a DSP. The Imote2 platform provides 32 MB of flash memory, 32 MB of SRAM and is running at 13 MHz - 414 MHz. The available computational power allows symmetric encryption in software, while for asymmetric operations the multiplier structure of the embedded DSP can be used to improve performance. Sun provides a different platform for IEEE 802.15.4, that allows simple programming of applications in Java. Their mote nodes, called Sun Spots [66], provide a powerful 32-bit processor running at 180 MHz with 512 KB RAM and 4 MB of flash memory. Sun has also authored Sizzle, which is a tiny secure webserver for embedded devices supporting SSL with ECC. Sizzle has been shown to be suited for execution on the Mica2 mote [24].

Although some products allow cryptographic functionality for a reasonable price already, we think that further development and improvement is necessary. Improving the energy consumption and tamper resistance, dedicated HW modules for symmetric (and possibly asymmetric) cryptographic operations will be necessary also for motes with more powerful processors. Additionally, we see a big chance in the application of dedicated instruction set extensions due to the improvements of energy efficiency that can be achieved for cryptographic operations.

5. Conclusions

In this book chapter, the importance of symmetric cryptographic primitives for providing security in wireless sensor networks has been highlighted. The security assurances cryp-

tography can provide have been described and an overview of symmetric primitives has been given. A comprehensive analysis of different modes of operation demonstrated the versatility of symmetric block ciphers in regard to such diverse security provisions like confidentiality and authentication. The most commonly used representatives of block ciphers, hash functions, stream ciphers, key generation algorithms and authentication codes were surveyed and the Advanced Encryption Standard (AES) algorithm described in more detail.

The main part of the book chapter provided an extensive survey of the implementation options of the AES and showed its high scalability in terms of required resources and performance. In more detail, software implementations on 8-bit, 16-bit and 32-bit processors, which can be frequently found ins WSN nodes, have been covered and detailed performance figures were given. Furthermore, AES hardware implementations with different design goals have been analyzed. This included an 8-bit design with low power consumption and went up to a high throughput implementation suitable for the backend of a WSN. A speciality was the described hardware/software co-design approach with cryptographic instruction set extensions. Here, the advantages of the flexible software solution could be combined with the speed-up of a hardware module. An overview of state-of-the-art cryptographic support in today's WSN products rounded off the chapter.

References

[1] M.-L. Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Çetin Kaya Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 2001.

[2] A. Alkassar, E. Andreeva, and H. Lipmaa. SLC: Efficient Authenticated Encryption for Short Packets. In J. Dittmann, editor, *Sicherheit 2006: Sicherheit - Schutz und Zuverlssigkeit, Beitrge der 3. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft fr Informatik e.v. (GI)*, volume 77 of *Lecture Notes in Informatics*, pages 270–278. Springer, February 2006.

[3] ARC International. The ARC International Website. `http://www.arc.com/`.

[4] P. S. Barreto and V. Rijmen. The WHIRLPOOL Hashing Function. Available online at `http://paginas.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html`, May 2003.

[5] M. Bellare and C. Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In T. Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, December 2000.

[6] M. Bellare, P. Rogaway, and D. Wagner. The EAX Mode of Operation. In B. K. Roy and W. Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 389–407. Springer, February 2004.

[7] D. J. Bernstein. The Poly1305-AES Message-Authentication Code. In H. Gilbert and H. Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*, pages 32–49. Springer, February 2005.

[8] G. Bertoni, L. Breveglieri, P. Fragneto, M. Macchetti, and S. Marchesin. Efficient Software Implementation of AES on 32-Bit Platforms. In B. S. K. Jr., Çetin Kaya Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 159–171. Springer, 2003.

[9] G. Bertoni, L. Breveglieri, F. Roberto, and F. Regazzoni. Speeding Up AES By Extending a 32-Bit Processor Instruction Set. In *Proceedings of the IEEE 17th International Conference on Application-*

specific Systems, Architectures and Processors (ASAP'06), pages 275–282. IEEE Computer Society, September 2006.

[10] J. Burke, J. McDonald, and T. Austin. Architectural Support for Fast Symmetric-Key Cryptography. In *ASPLOS-IX Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge, MA, USA, November 12-15, 2000*, pages 178–189, New York, NY, USA, 2000. ACM Press.

[11] C. D. Canniére and C. Rechberger. Finding SHA-1 Characteristics: General Results and Applications. In X. Lai and K. Chen, editors, *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, volume 4284 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006.

[12] D. Canright. A Very Compact S-Box for AES. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 441–455. Springer, 2005.

[13] CoWare Inc. The CoWare Website. `http://www.coware.com/`.

[14] Crossbow Technology Inc. The Crossbow Technology Website. `http://www.xbow.com/`.

[15] J. Daemen and V. Rijmen. AES proposal: Rijndael. First AES Conference, August 1998.

[16] J. Daemen and V. Rijmen. *The Design of Rijndael*. Information Security and Cryptography. Springer, 2002. ISBN 3-540-42580-2.

[17] I. Damgård. A Design Principle for Hash Functions. In G. Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, August 1989.

[18] ECRYPT. eSTREAM - The ECRYPT Stream Cipher Project Website. `http://www.ecrypt.eu.org/stream/`.

[19] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. AES Implementation on a Grain of Sand. *IEE Proceedings on Information Security*, 152(1):13–20, October 2005.

[20] A. M. Fiskiran and R. B. Lee. On-Chip Lookup Tables for Fast Symmetric-Key Encryption. In *Proceedings of the IEEE 16th International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, pages 356–363. IEEE, IEEE Press, July 2005.

[21] B. Gladman. AES and Combined Encryption/Authentication Modes. Available online at `http://fp.gladman.plus.com/AES/index.htm`, March 2007.

[22] V. D. Gligor and P. Donescu. Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes. In M. Matsui, editor, *Fast Software Encryption, 8th International Workshop, FSE 2001 Yokohama, Japan, April 2-4, 2001, Revised Papers*, volume 2355 of *Lecture Notes in Computer Science*, pages 92–108. Springer, April 2001.

[23] J. Großschädl, S. Tillich, C. Rechberger, M. Hofmann, and M. Medwed. Energy Evaluation of Software Implementations of Block Ciphers under Memory Constraints. In *10th Design Automation and Test in Europe Conference (DATE07), Nice, France, April 16-20, 2007, Proceedings*, pages 1110–1115. IEEE Computer Society Press, April 2007.

[24] V. Gupta, M. Millard, S. Fung, Y. Z. N. Gura, H. Eberle, and S. C. Shantz. Sizzle: A Standards-based end-to-end Security Architecture for the Embedded Internet. In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications, 2005. PerCom 2005*, pages 247–256. IEEE Computer Society, March 2005.

[25] G. Hachez, F. Koeune, and J.-J. Quisquater. cAESar results: Implementation of Four AES Candidates on Two Smart Cards. In *Second Advanced Encryption Standard Candidate Conference*, pages 95–108, Hotel Quirinale, Rome, Italy, March 1999.

[26] C. Herbst, E. Oswald, and S. Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. In J. Zhou, M. Yung, and F. Bao, editors, *Applied Cryptography and Network Security, Second International Conference, ACNS 2006*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 2006.

[27] A. Hodjat, D. Hwang, B.-C. Lai, K. Tiri, and I. Verbauwhede. A 3.84 Gbits/s AES crypto coprocessor with modes of operation in a 0.18-um CMOS Technology. In J. Lach, G. Qu, and Y. I. Ismail, editors, *15th ACM Great Lakes Symposium on VLSI 2005, Chicago, Illinois, USA, April 17-19, 2005, Proceedings*, pages 60–63. ACM, ACM Press, April 2005. Available online at `http://portal.acm.org/ft_gateway.cfm?id=1057677\&type=pdf\&coll=GUIDE\`

&dl=GUIDE\&CFID=50585284\&CFTOKEN=38947284.

[28] IAIK—Institute for Applied Information Processing, Graz University of Technology. AES Lounge. http://www.iaik.tugraz.at/research/krypto/AES.

[29] IEEE. IEEE Standard 802.11g-2003: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band. Available online at http://standards.ieee.org/getieee802/, June 2003.

[30] IEEE. IEEE Standard 802.15.4-2003: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). Available online at http://standards.ieee.org/getieee802/, May 2003.

[31] IEEE. IEEE Standard 802.11i-2004: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 6: Medium Access Control (MAC) Security Enhancements. Available online at http://standards.ieee.org/getieee802/, July 2004.

[32] IEEE. IEEE Standard 802.15.1-2005: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs). Available online at http://standards.ieee.org/getieee802/, June 2005.

[33] Institute for Applied Information Processing and Communication, Graz University of Technology. VLSI Products–Software Modules. http://www.iaik.tugraz.at/research/vlsi/02_products/index.php, May 2006.

[34] J. Irwin and D. Page. Using Media Processors for Low-Memory AES Implementation. In E. Deprettere, S. Bhattacharyya, J. Cavallaro, A. Darte, and L. Thiele, editors, *14th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 144–154. IEEE Computer Society Press, 2003.

[35] ISO/IEC. Information technology—Security techniques—Hash-functions—Part 3: Dedicated hash-functions. Available from http://www.iso.org/ (with costs), 2004.

[36] H. K. T. K. John Black, Shai Halevi and P. Rogaway. UMAC: Fast and Secure Message Authentication. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 216–233. Springer, 1999.

[37] C. S. Jutla. Encryption Modes with Almost Free Message Integrity. In B. Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceedings*, volume 2045 of *Lecture Notes in Computer Science*, pages 529–544. Springer, May 2001.

[38] J. Katz and M. Yung. Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation. In B. Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 284–299. Springer, April 2001.

[39] G. Keating. Performance Analysis of AES candidates on the 6805 CPU core. In *Proceedings of The Second AES Candidate Conference*, pages 109–114, Hotel Quirinale, Rome, Italy, March 1999.

[40] P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, number 1109 in Lecture Notes in Computer Science, pages 104–113. Springer, 1996.

[41] T. Kohno, J. Viega, and D. Whiting. CWC: A high-performance conventional authenticated encryption mode. In B. K. Roy and W. Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 408–426. Springer, February 2004.

[42] S. Lucks. Two-Pass Authenticated Encryption Faster Than Generic Composition. In H. Gilbert and H. Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*, pages 284–298. Springer, February 2005.

[43] S. Mangard, M. Aigner, and S. Dominikus. A Highly Regular and Scalable AES Hardware Architecture. *IEEE Transactions on Computers*, 52(4):483–491, April 2003.

[44] S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks – Revealing the Secrets of Smart Cards*. ADIS Book Series. Springer, 2007. ISBN 0-387-30857-1.

[45] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. Series on Discrete Mathematics and its Applications. CRC Press, 1997. ISBN 0-8493-8523-7, Available online at

http://www.cacr.math.uwaterloo.ca/hac/.

[46] R. C. Merkle. *Secrecy, authentication, and public key systems*. Ph.d. thesis, Stanford University, 1979. Available online at http://www.merkle.com/papers/Thesis1979.pdf.

[47] R. C. Merkle. A Certified Digital Signature. In G. Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer, August 1989.

[48] MIPS Technologies, Inc. The MIPS Technologies Website. http://www.mips.com/.

[49] Moteiv. The Moteiv Wireless Sensor Networks Website. http://www.moteiv.com/.

[50] K. Nadehara, M. Ikekawa, and I. Kuroda. Extended Instructions for the AES Cryptography and their Efficient Implementation. In *IEEE Workshop on Signal Processing Systems (SIPS'04)*, pages 152–157, Austin, Texas, USA, October 2004. IEEE Press.

[51] National Institute of Standards and Technology (NIST). FIPS-46-3: Data Encryption Standard, October 1999. Available online at http://www.itl.nist.gov/fipspubs/.

[52] National Institute of Standards and Technology (NIST). FIPS-197: Advanced Encryption Standard, November 2001. Available online at http://www.itl.nist.gov/fipspubs/.

[53] National Institute of Standards and Technology (NIST). Special Publication 800-38A 2001 ED, Recommendation for Block Cipher Modes of Operation - Methods and Techniques, December 2001. Available online at http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf.

[54] National Institute of Standards and Technology (NIST). Special Publication 800-38C 2004, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, May 2004. Available online at http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C.pdf.

[55] National Institute of Standards and Technology (NIST). Proposed Modes of Operation. Available online at http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/, June 2005.

[56] National Institute of Standards and Technology (NIST). Special Publication 800-38B 2005, Cipher Modes of Operation: The CMAC Mode for Authentication, May 2005. Available online at http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf.

[57] National Institute of Standards and Technology (NIST). Draft Special Publication 800-38D 2006, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) for Confidentiality and Authentication, April 2006. Available online at http://csrc.nist.gov/publications/drafts/Draft-NIST_SP800-38D_Public_Comment.pdf.

[58] D. Oliva, R. Buchty, and N. Heintze. AES and the Cryptonite Crypto Processor. In *CASES '03: Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems*, pages 198–209, New York, NY, USA, 2003. ACM Press.

[59] B. Preneel, R. Govaerts, and J. Vandewalle. Hash Functions Based on Block Ciphers: A Synthetic Approach. In D. R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer, 1993.

[60] S. Ravi, A. Raghunathan, N. Potlapally, and M. Sankaradass. System design methodologies for a wireless security processing platform. In *DAC '02: Proceedings of the 39th Conference on Design Automation*, pages 777–782, New York, NY, USA, 2002. ACM Press.

[61] P. Rogaway. Authenticated-encryption with associated-data. In V. Atluri, editor, *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washingtion, DC, USA, November 18-22, 2002*, pages 98–107. ACM, November 2002.

[62] P. Rogaway, M. Bellare, and J. Black. OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. *ACM Transactions on Information and System Security*, 6(3):365–403, August 2003.

[63] C. Röpke. Praktikum B:Embedded Smartcard Microcontrollers. http://www.christianroepke.de/studium_praktikumB.html, 2003.

[64] F. Sano, M. Koike, S. ichi Kawamura, and M. Shiba. Performance Evaluation of AES Finalists on the High-End Smart Card. In *Proceedings of the Third Advanced Encryption Standard Conference*, pages 82–93, Hilton New York and Towers in New York, New York, USA, April 2000. Available online at http://csrc.nist.gov/CryptoToolkit/aes/round2/conf3/papers/AES3Proceedings.pdf.

[65] SIC. The Secure Information and Communication Technologies Website. http://jce.iaik.tugraz.

at/.

[66] Sun Microsystems, Inc. The SunSportWorld Website. http://www.sunspotworld.com/.

[67] Tensilica Inc. The Tensilica Website. http://www.tensilica.com/.

[68] S. Tillich, M. Feldhofer, and J. Großschädl. Area, Delay, and Power Characteristics of Standard-Cell Implementations of the AES S-Box. In S. Vassiliadis, S. Wong, and T. Hämäläinen, editors, *6th International Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS 2006, Samos, Greece, July 17-20, 2006, Proceedings*, volume 4017 of *Lecture Notes in Computer Science*, pages 457–466. Springer, July 2006.

[69] S. Tillich and J. Großschädl. Accelerating AES Using Instruction Set Extensions for Elliptic Curve Cryptography. In M. Gavrilova, Y. Mun, D. Taniar, O. Gervasi, K. Tan, and V. Kumar, editors, *Computational Science and Its Applications - ICCSA 2005*, volume 3481 of *Lecture Notes in Computer Science*, pages 665–675. Springer, 2005.

[70] S. Tillich and J. Großschädl. Instruction Set Extensions for Efficient AES Implementation on 32-bit Processors. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 270–284. Springer, 2006.

[71] S. Tillich, J. Großschädl, and A. Szekely. An Instruction Set Extension for Fast and Memory-Efficient AES Implementation. In J. Dittmann, S. Katzenbeisser, and A. Uhl, editors, *Communications and Multimedia Security — 9th IFIP TC-6 TC-11 International Conference, CMS 2005, Salzburg, Austria, September 2005, Proceedings*, volume 3677 of *Lecture Notes in Computer Science*, pages 11–21. Springer, 2005.

[72] S. Tillich, C. Herbst, and S. Mangard. Protecting AES Software Implementations on 32-bit Processors against Power Analysis. In *Proceedings of the 5th International Conference on Applied Cryptography and Network Security (ACNS'07)*, Lecture Notes In Computer Science, 2007. To be published.

[73] Tritech Technology AB. The Tritech Website. http://www.tritech.se/.

[74] X. Wang, Y. L. Yin, and H. Yu. Finding Collisions in the Full SHA-1. In V. Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005.

[75] M. N. Wegman and L. Carter. New Hash Functions and Their Use in Authentication and Set Equality. *Journal of Computer and System Sciences*, 22(3):265–279, June 1981.

[76] J. Wolkerstorfer. An ASIC Implementation of the AES-MixColumn operation. In P. Rössler and A. Döderlein, editors, *Austrochip 2001*, pages 129–132, 2001. ISBN 3-9501517-0-2.

[77] J. Wolkerstorfer, E. Oswald, and M. Lamberger. An ASIC implementation of the AES SBoxes. In B. Preneel, editor, *Topics in Cryptology - CT-RSA 2002, The Cryptographers' Track at the RSA Conference 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings*, volume 2271 of *Lecture Notes in Computer Science*, pages 67–78. Springer, 2002.

[78] L. Wu, C. Weaver, and T. Austin. CryptoManiac: A Fast Flexible Architecture for Secure Communication. In *ISCA '01: Proceedings of the 28th annual international symposium on Computer architecture*, pages 110–119, New York, NY, USA, 2001. ACM Press.

[79] ZigBee Alliance. The ZigBee Alliance Website. http://www.zigbee.org/.

Wireless Sensor Network Security
J. Lopez and J. Zhou (Eds.)
IOS Press, 2008

Public-key Primitives

Lejla BATINA [a,1], Stefaan SEYS [a], Bart PRENEEL [a], and Ingrid VERBAUWHEDE [a]
[a] *SCD/COSIC, Dept. Electrical Engineering (ESAT), K. U. Leuven, Belgium*

Abstract. Wireless sensor networks consist of tiny senor nodes with limited computing and communicating capabilities and, more importantly, with limited energy resources. In this chapter we evaluate the power consumption of Public-key algorithms and investigate whether these algorithms can be used within the power constrained sensor nodes. We evaluate conventional digital signature schemes and encryption schemes, one-time signature schemes and Public-key authentication schemes.

Keywords. Public-key cryptography, efficient implementation, power consumption

1. Introduction and Motivation

The suitability of Public-key (PK) algorithms for sensor networks is an open research problem as limitations in costs, area and power are quite severe. There exists a common concept of Public-key Cryptography (PKC) being too slow and too expensive for low-cost pervasive applications and most of the protocol proposals deal only with symmetric-key cryptography.

However, the security in wireless sensor networks is becoming more and more relevant as a large number of nodes is exposed in sometimes hostile environments. The key protection is therefore of utmost importance and for that reason there is a clear advantage of using PKC.

Besides, various cryptographic services are required for these applications such as encryption, broadcast authentication, key exchange *etc.* Usual solutions to use symmetric-key algorithms such as AES and MACs are not just complicating issues such as key protection and management but can be at the same time even more expensive with respect to power and energy consumed. In addition, the use of PKC reduces power due to less protocol overhead [1].

To the best of our knowledge very few papers discuss the possibility for PKC in these applications although the benefits of PKC are evident especially for key distribution between the nodes and various authentication protocols. For example, the authentication of the base station is easily performed assuming the Public key of the base station can be stored in each node [2]. One of the reasons is that software-only approach for PKC is very expensive. There has a been a several studies showing that pure software implementations are to slow on some platforms to fulfil the security requirements [4].

[1]Corresponding Author: Lejla Batina, SCD/COSIC, Dept. Electrical Engineering (ESAT), K. U. Leuven, Belgium; E-mail: lejla.batina@esat.kuleuven.be.

However, already a small hardware module to accelerate computationally intensive finite field operations results in a huge improvement in performance of a factor of almost 2 orders of magnitude [5]. Nevertheless, one of the recent works shows that PKC for sensor nets might be possible even as a pure software solution [3].

In this chapter we describe Public-key Cryptography based solutions for security services such as encryption, key-distribution and authentication as required for wireless sensor networks. We discuss suitable protocols and the costs they imply. We restrict ourselves to existing protocols, although we believe that a substantial improvement can be obtained by revisiting and eventually redesigning the existing solutions. So, the selection of algorithms we address in this work is mainly based on previous proposals. We also argue that a custom hardware assisted approach to implement PKC in order to obtain stronger cryptography as well as to minimize the power might be the right one for lightweight applications as proposed by several publications so far [14,1,7]. We discuss and elaborate our design choices on all levels of the protocols hierarchy. Our results show feasible solution for Public-key Cryptography for these applications.

As an example we show that the Elliptic Curve Cryptography (ECC) algorithms that minimize the memory requirements and that require the fewest field operations seem to be a suitable for sensor nets applications. Furthermore, as we focus on hardware assisted approach for ECC implementations we describe a very compact arithmetic and control unit to support ECC protocols [14].

The chapter is organized as follows. In sections 2 and 3 we describe Public-key algorithms that can be used for encryption and digital signatures respectively. Next to traditional signature schemes, section 3 also includes a selection of efficient one-time signature schemes based on a general one-way function (OWF) f. Next we describe a number of authentication schemes in section 4. Finally, in section 5 we describe the performance of the different systems we described. This includes both a performance evaluation in software and in hardware.

2. Public-key Encryption Schemes

2.1. RSA Encryption

The best known Public-key encryption scheme is RSA. It was invented by Rivest, Shamir and Adleman in 1978 [49]. The textbook version of the RSA encryption scheme is depicted in Alg. 1.

The security of RSA against a chosen-plaintext attack relies on the difficulty of computing the e-th root of a ciphertext c modulo a composite integer n. This is known as the *RSA problem*. The difficulty of the RSA problem depends on the difficulty of the *integer factorization problem*, i.e., given an odd composite integer with at least two distinct prime factors it is hard to provide one of these prime factors. Clearly, an algorithm that solves the integer factorization problem will solve the RSA problem since this is exactly what happens in the RSA key setup process. However, the converse is still an open problem: can the integer factorization problem be hard if the RSA problem is not hard? For a rigorous security analysis of the RSA scheme and further references we refer to [39].

The naive description of the RSA encryption scheme in Alg. 1 should not be used in practice. Bellare and Rogaway [31] have proposed a provably secure way of encrypting messages using RSA or Rabin (see below), known as the OEAP scheme.

Algorithm 1 The RSA Public-key encryption system

Key setup

1. Generate two large distinct random primes p and q such that $|p| \approx |q|$;
2. compute $n = pq$ and $\phi(n) = (p-1)(q-1)$;
3. generate a random integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$;
4. use the extended Euclidean algorithm to compute the unique integer d such that $1 < d < \phi$ and $ed = 1 \pmod{\phi(n)}$;
5. publish (n, e) as the public key, keep (p, q, d) or (n, d) as the private key.

Encryption

Given a public key (n, e), the ciphertext c of message $m \in \mathbb{Z}_n^*$ is

$$c = \mathsf{E}_{(n,e)}(m) = m^e \pmod{n}\,.$$

Decryption

To decrypt the ciphertext c using the secret key (n, d) one computes

$$m = \mathsf{D}_{(n,d)}(c) = c^d \pmod{n}\,.$$

It is clear from Alg. 1 that the efficienty of the RSA algorithm depend on the selection of the parameters p, q and e. The paramters p and q have to be sufficiently large in order for the algorithm to be secure (i.e., $|p| \approx |q| \geq 512$ bit). The minimum version for the public exponent e is theoretically 3, but for encryption this is not deemed secure enough and usually $e = 2^{16} + 1$ is used. The size of the resulting private exponent d is much larger ($|d| \approx |n|$). From this we can already conclude that the RSA encryption (public operation) is much more efficient than RSA decryption (private operation).

2.2. Encryption based on EC

The security of many asymmetric cryptographic primitives (e.g., the DSA) relies on the difficulty of computing a discrete logarithm in a finite cyclic group. In elliptic curve cryptography, this group is provided by an elliptic curve $\mathcal{E}$ defined over $\mathbb{F}_q$ with $q = p^m$ and p a prime number, and a definition of a method to add two points on the curve. The *elliptic curve discrete logarithm problem* can be defined as follows: given the points $P \in \mathcal{E}$ and $Q = \alpha P$ (with α an integer smaller than the order P), find the discrete logarithm α of Q.

Two well known elliptic curve based encryption schemes are the ECIES [51] (also known as the Elliptic Curve Augmented Encryption Scheme or simply the Elliptic Curve Encryption Scheme) and PSEC [51]. The security of ECIES is based on the difficulty of the *Computational Diffie-Hellman problem for elliptic curves*: Given the points $P \in \mathcal{E}$, $Q = \alpha P$ and $R = \beta P$ (with α, β integers smaller than the order P), compute $\alpha\beta P$. The ECIES is closely related to the Diffie-Hellman Integrated Encryption Scheme (DHIES) construction in [28]. PSEC is a family of Diffie-Hellman based encryption schemes that are all provably secure in the random oracle model. The security of each member of the

Table 1. Elliptic curve, symmetric primitives, RSA and discrete log in $\mathbb{F}_q^*$ key length comparison.

Symmetric primitive key lengths	Elliptic curve key lengths	discrete log ($\mathbb{F}_q^*$) and RSA key lengths
80	160	1024
112	224 ($\times 1.4$)	2048 ($\times 2$)
128	256 ($\times 1.6$)	3072 ($\times 3$)
192	384 ($\times 2.4$)	7680 ($\times 7.5$)
256	512 ($\times 3.2$)	15360 ($\times 15$)

family is based on a different variant of the Diffie-Hellman problem. Note that these encryption schemes have been standardized by different standardization bodies, and unfortunately the versions are not always compatible.

The advantage of elliptic curves is that they can provide the same level of security as RSA or the DSA with substantially smaller key sizes. Note that the signature sizes for both, the DSA and the ECDSA are exactly 2 times the field size. Table 1 lists elliptic curve key lengths and rough estimates of key sizes of symmetric primitives, RSA and discrete log based cryptosystems (both over $\mathbb{F}_q^*$ and elliptic curves) that provide the same level of security. These estimates were obtained from [50], and they are roughly the same as those proposed in a very detailed paper by Lenstra and Verheul [38]. Table 1 also shows how fast the key sizes grow (compared to the security of an 80-bit symmetric key). This means that in the future, the key size advantage of elliptic curve based cryptosystems will only improve.

2.3. NTRU encrypt

NTRU [19] is one of the most efficient Public-key cryptosystems known and it provides encryption as well as signatures. The corresponding protocols are NTRUEncrypt and NTRUSign. The security of NTRU-based schemes depends on the difficulty of certain lattice problems. The authors of the NTRU cryptosystem recommend to use a lattice of dimension ≥ 500, based on their own experiments [20].

The security of NTRU primitives has been an active research area for the past decade and NTRUSign has been successfully attacked as well as its improvements. On the other hand NTRUEncrypt appeared to be a secure scheme against all known attacks till very recent publication of Gama and Nguyen [26]. The authors introduced new chosen-ciphertext attacks on NTRUEncrypt that work in the presence of decryption failures. NTRU cryptosystems are now evaluated by IEEE standardization group.

The efficiency i.e. the speed and compactness are due to the fact that both protocols rely on inexpensive operations on polynomials with small coefficients. Here we focus on the encryption scheme that seems to have better chances to be adopted for applications such as smart cards, RFID tags and sensor nets due to its suitability for constrained environments as well as stronger security than NTRUSign and the successors of it.

NTRU schemes are based on arithmetic in a polynomial ring $\mathbb{Z}(x)/((x^N - 1), q)$ with a parameters (N, p, q) with certain properties [19]. The key operation is multiplication in the ring (here denoted as *) that can be explained as the convolution product of two vectors a and b written as: $a(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{N-1}x^{N-1}$ and $b(x) = b_0 + b_1x + b_2x^2 + \ldots + b_{N-1}x^{N-1}$. Then the product $c(x) = a(x) * b(x) \bmod q, p$ has coefficients c_k that are computed as the sum of partial products a_ib_j where $i + j \equiv$

Algorithm 2 NTRU Public-key encryption system

Key setup

1. Choose a random polynomial $F(x)$ from the ring R. $F(x)$ should have small coefficients i.e. either from the set $\{0, 1\}$ (when $p = 2$) or from $\{-1, 0, 1\}$ (when $p = 3$ or $p = x + 2$);
2. Let $f(x) = 1 + pF(x)$. (This construction for $f(x)$ is not necessary but it is recommended in order to decrease the decryption failure rate.);
3. Choose a random polynomial $g(x) \in R$ in the same way as $F(x)$ is chosen;
4. Compute the inverse of $f(x)$, so $f^{-1} \bmod q$;
5. Compute $h(x) = g(x) * f^{-1}(x) \bmod q$;
6. Publish $h(x)$ as the public key and keep $f(x)$ as the private key.

Encryption

1. Encode the plaintext message into a polynomial $m(x)$ with coefficients from $\{0, 1\}$ or $\{-1, 0, 1\}$;
2. Choose a random polynomial $\phi(x) \in R$ as above;
3. Compute the ciphertext as polynomial $c(x) = p\phi(x) * h(x) + m(x)(\mathrm{mod} q)$.

Decryption

To decrypt the ciphertext $c(x)$ using the secret key $f(x)$ one computes the message polynomial $m'(x) = c(x) * f(x) \bmod p$ and maps the coefficient of m' to plaintext bits.

$k \bmod N$. The moduli for reduction are p and q for decryption and encryption/key setup respectively. For a detailed description as well as mathematical background we refer to [19]. Algorithm 2 shows the details of the encryption algorithm.

Considering performance numbers Gaubatz et al. [2] showed that Rabin's scheme is not a feasible solution while NTRUEncrypt can be implemented in not more than 3000 gates with an average power consumption of less than 20 μW at a clock frequency of 500 kHz. These figures are obtained for the parameters that are guaranteeing 57 bits of security and they are acceptable for sensor nets applications.

3. Digital Signature Schemes

3.1. ECDSA

A well known elliptic curve based digital signature scheme is the ECDSA [36]. The ECDSA is the elliptic curve analogue of the DSA. It was added to the DSS in 2000 (FIPS PUB 186-2). The algorithm is shown in Alg. 3.

The Digital Signature Algorithm (DSA) is well known as the service that uniquely binds a message and a sender. This problem is based on the Discrete logarithm problem (DLP) i.e. on the difficulty of computing logarithms in a large finite field. Another protocol which provides a digital signature, is the ECDSA protocol, which is performed as follows [21]:

ECDSA protocol:

The ECDSA is specified by an elliptic curve *E* defined over $\mathbb{F}_q$ and a publicly known

Algorithm 3 The ECDSA signature scheme

Setup of system parameters

Select an elliptic curve $\mathcal{E}$ defined over $\mathbb{F}_q$ (with $q = p^m$, where p is a prime number), and a publicly known point $G \in \mathcal{E}$ of large prime order n.

Key setup

1. Select a random integer $d \in_{\mathbb{R}} [1, n[$,
2. Compute $Q = dG$,
3. Publish Q as the public key, and keep d as the private key.

Signature generation

1. Select a random integer $k \in_{\mathbb{R}} [1, n[$;
2. Compute $kG = (x_1, y_1)$;
3. Compute $r = x_1 \pmod n$, if $r = 0$ the go back to step 1;
4. Compute $s = k^{-1}(H(m)+dr) \pmod n$, where H is a hash algorithm that maps $\{0,1\}^*$ to $[1, n[$; if $s = 0$ the go back to step 1;
5. the signature of message m is (r, s).

Signature verification

Given a public key Q, the system parameters and a message-signature pair $(m, (r, s))$, the verifier can verify the signature with the following procedure:

1. Verify that r and s are integers in the interval $[1, n[$;
2. Compute $w = s^{-1} \pmod n$;
3. Compute $u_1 = H(m)w \pmod n$ and $u_2 = rw \pmod n$;
4. Compute $u_1G + u_2Q = (x_0, y_0)$ and $v = x_0 \pmod n$;
5. $\mathsf{Verify}_Q(m, (r, s)) = \mathsf{true}$ if $v = r$.

point $G \in E$ of prime order n. A private key of Alice is a scalar x and the corresponding public key is $Q = xG \in E$. The ECDSA requires a hash function which is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length, called hash-values as defined in Chapter 1.

3.2. RSA, Rabin and DSA Signatures

3.2.1. RSA Signature Scheme

The key setup of the RSA signature scheme is the same as the key setup of the RSA encryption scheme. The signature of message $m \in \mathbb{Z}_n^*$ is $s = m^d \pmod n$. Given a public key (n, e) and a message-signature pair (m, s), a signature is valid if $m = s^e \pmod n$.

It is easy to see that (for the textbook version) the RSA signing procedure is the same as the RSA decryption procedure, and signature verification is the same as encryption. Without additional measures, it is straightforward to forge a signature (i.e., generate a valid signature without knowledge of the private key on a message that has not been signed by the owner of the private key). Suppose an adversary has obtained

two valid message-signature pairs (m_1, s_1) and (m_2, s_2). By multiplying the signatures she gets a valid signature on the product of the two messages: $s_1 \times s_2 \pmod n = m_1^d \times m_2^d \pmod n = (m_1 \times m_2)^d \pmod n$. Note that the adversary has no option (next to multiplying different message pairs) to manipulate the message on which he obtains a signature. This and other methods of forgery are known as existential forgeries.

A usual method of detecting existential forgeries is to add recognizable redundancy to the message to be signed, which permits a verifier the correct "format" of the signed message. The most common method for adding recognizable information to a message is to apply a cryptographic hash function before applying the signature algorithm. Usually, this hash computation (and other measures such as padding) are an integral part of a digital signature scheme. Bellare and Rogaway [32] have presented a provably secure way of creating signatures with RSA and Rabin (see below), known as the PSS. The proof of security for PSS relies on the random oracle model, in which hash functions are modeled as being truly random functions. In contrast, the method for creating digital signatures with RSA that is described in PKCS #1 [45] has not been proven secure, even if the underlying RSA primitive is secure. PSS-R is a message recovery variant of PSS with the same provable security.

3.2.2. DSA Signature Scheme

The DSA is part of the DSS which was first announced in 1991 by NIST and published in 1994 (FIPS PUB 186). The security of the DSA is based on the difficulty of computing a discrete logarithm in the finite cyclic group $\mathbb{F}_q^*$ with q prime. This is called the *discrete logarithm problem*: Given the integers x and $y = x^\alpha \in \mathbb{F}_q^*$ (with x a generating element of the group, and $\alpha < q - 1$), find the discrete logarithm α of y. As the best algorithms known for the integer factorization and discrete log problems have the same expected running times [40], the required key sizes for RSA and the DSA are the same. The DSA algorithm is the same as the ECDSA algorithm (Alg. 3), but uses the finite cyclic group $\mathbb{F}_q^*$ instead of the group provided by an elliptic curve.

3.3. One-time signature schemes

3.3.1. Introduction

One-time signatures have been known since the late 1970s. They were introduced by Diffie and Hellman [34], Lamport [37] and Rabin [47]; but they are usually known in the form presented by Merkle [42,43]. These schemes are based on one-way function, rather than on trapdoor functions that are used in traditional schemes such as RSA and the DSA.

In its basic form, the Lamport-Diffie scheme can be used to sign a single bit of data. The secret key consists of two random values x_0 and x_1, while the public key is obtained by applying the OWF f to the secret values, resulting in the pair $\{f(x_0), f(x_1)\}$. The signature for bit b is x_b. The security of this scheme relies on the one-wayness of the function f, i.e., given the public key, it is impossible to compute the private key (and thus forge a signature) without breaking the one-way property of f. It is also clear that a public/private key pair can only be used once since the signature is equal to part of the private key. To sign longer messages, several instances of this scheme are used. In order to sign an s-bit message one requires $2s$ public key values and $2s$ private key values. A signature consists of s values.

Algorithm 4 Lamport-Diffie signature scheme with Merkle improvement (LDM)

Setup of system parameters

Select a OWF f mapping $\{0,1\}^l$ to $\{0,1\}^l$. Let s be the fixed length of the messages to be signed ($s = |m|$).

Key setup

1. Generate $t = s + \lceil \log_2(s) \rceil$ random values $x_1, x_2, \ldots, x_t$ with $|x_i| = l$,
2. let $sk = \{x_1, x_2, \ldots, x_t\}$,
3. compute $pk = \{f(x_1), f(x_2), \ldots f(x_t)\}$,
4. publish pk as the one-time public key, keep sk as the one-time private key.

Signature generation

Let b_i be the i-th bit of $\langle m, w \rangle$ with w the Hamming weight of the message m. The signature of message m is

$$\sigma = \mathsf{Sign}_{sk}(m) = \{\text{all } x_i \text{ for which } b_i = 1\}\,.$$

Signature verification

Given a public key $pk = \{v_1, v_2, \ldots, v_t\}$ and a message-signature pair (m, σ), the verifier can verify the signature with the following procedure (b_i is the i-th bit of $\langle m, w \rangle$):

$$\mathsf{Verify}_{pk}(m, \sigma) = \mathsf{true} \text{ if } f(\sigma_\alpha) = v_i \text{ for all } i \text{ where } b_i = 1\,.$$

(σ_α indicates the corresponding value in the signature, e.g., $\alpha = 3$ for the 3rd '1' bit in $\langle m, w \rangle$.)

3.3.2. Lamport-Diffie scheme with Merkle improvement

Merkle [41] proposed an improvement that allows to reduce the key sizes by a factor of two and the signature size by almost a factor of two. Instead of generating two private key values for every bit in the message, Merkle suggests to only generate one value. The public key values are still obtained by applying the function f to the private key values. Now, for every bit in the message that is '1', include the corresponding private key value in the signature; for every bit that is '0', omit the corresponding private key value. On average, this results in a signature that contains $s/2$ private key values. Obviously, the verifier can always claim not to have received a particular private key value, and therefore pretend that some of the '1' bits in the message that was signed were actually '0' bits. This can be remedied by adding the Hamming weight of the message before signing it. This count requires $\log_2(s)$ bits. We will refer to this scheme as the Lamport-Diffie one-time signature scheme with Merkle improvement (LDM) (Alg. 4).

3.3.3. Lamport-Diffie scheme with Winternitz improvement

In [43] Merkle proposes a different variant of the Lamport-Diffie scheme, attributed by Merkle to Winternitz. This scheme reduces the size of the signature at the cost of additional computations. Instead of applying the OWF f once to the private key to obtain the public key, the function f is applied iteratively a fixed number of times. With ev-

ery resulting public/private key value pair it is possible to sign multiple bits. Briefly the scheme works as follows: To sign 4 bits with a single public/private key value pair, we apply the function f 15 times ($= 2^4 - 1$), thus the public key becomes $v = f^{15}(x)$. To sign the message 1001 (9 in decimal), the signer makes $\sigma = f^{15-9}(x)$ public. Anyone can check that $f^9(\sigma) = f^9(f^{15-9}(x)) = v$, thus confirming that $f^{15-9}(x)$ was made public. No one besides the signer could have generated this value. Again extra redundancy has to be added to the signature in order to prevent people from changing the signature on 1001 into a signature on, for example 1000 (8 in decimal), by computing $f(f^{15-9}(x)) = f^{15-8}(x)$.

The complete scheme is described in Alg. 5. Note that a different mechanism is used to add the necessary redundancy to the signature. This solution reduces the signature size even further at the cost of additional computations. The redundancy in this scheme is encoded in the signature value σ_0 and requires one additional public/private key value pair $\{x_0, v_0\}$. Again we assume that the message m is hashed before it is fed to the signing algorithm.

3.3.4. The HORS one-time signature scheme

Reyzin and Reyzin propose a very efficient one-time signature scheme based on subset selection [48]. Their scheme builds on a construction proposed by Bos and Chaum in [33] that allows to sign $r \geq 1$ messages with a single public/private key pair.

In short, the signature scheme proposed by Bos and Chaum works as follows: the public/private key pair is generated as in the basic Lamport-Diffie scheme, i.e., the public key is obtained by applying a OWF f to each of the t values of the private key. The signing algorithm uses a bijective function S that, on input m ($0 \leq m < \binom{t}{k}$), outputs the m-th k-element subset of the set $T = \{1, 2, \ldots, t\}$. Let this subset be $\{i_1, i_2, \ldots, i_k\}$. The signature for message m is $\{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$. Because each message results in a different k-element subset (due to the bijective property of S), in order to forge a signature after obtaining a single message-signature pair, the forger will have to invert the OWF f at least once (i.e., for all elements in the forged signature that are not part of the obtained signature). This makes it possible to reduce the security of this scheme to the one-wayness of the function f.

Reyzin and Reyzin propose to replace the subset selection function S by a cryptographic hash function H. The hash value $h = H(m)$ is split into k parts of equal length. Every part is interpreted as an integer and the collection of all these integers is the subset that will be used to select the private key values to be included in the signature. Reyzin and Reyzin have called this scheme HORS for "Hash to Obtain Random Subset"; this algorithm is described in Alg. 6.

If we assume that the hash function H behaves like a random oracle [30], and that the adversary obtained signatures on r random messages using the same private key. The probability that an adversary is able to forge a signature on a new message (without inverting the OWF f) is at most $(rk/t)^k$. This is the probability that after rk elements of sk have been made public, k elements are chosen (at random) that are a subset of them.

Security level of HORS The security level of the HORS signature scheme in combination with the hash function H, is defined as $\Sigma = -\log_2(P)$. Here, P is the probability of breaking the $(r + 1, k)$ subset resilience of the hash function H, assuming that H behaves like a random oracle. This probability P is at most $(rk/t)^k$.

Algorithm 5 Lamport-Diffie one-time signature scheme with Winternitz improvement (LDW)

Setup of system parameters

Select a OWF f mapping $\{0,1\}^l$ to $\{0,1\}^l$. Let s be the fixed length of the messages to be signed ($s = |m|$). Select the system parameter g such that $g|s$.

Key setup

1. Generate $s/g + 1$ random values $x_0, x_1, \ldots, x_{s/g}$ with $|x_i| = l$,
2. let $sk = \{x_0, x_1, \ldots, x_{s/g}\}$,
3. compute $pk = \{f^{(2^g-1)s/g}(x_0), f^{2^g-1}(x_1), \ldots\ f^{2^g-1}(x_{s/g})\}$,
4. publish pk as the one-time public key, keep sk as the one-time private key.

Signature generation

1. Split the message m into s/g parts, let these parts be $m_1, m_2, \ldots, m_{s/g}$,
2. interpret each m_j as an integer I_j,

The signature of message m is

$$\sigma = \mathsf{Sign}_{sk}(m) = \{\sigma_0, \ldots, \sigma_{s/g}\}$$

$$\text{with } \begin{cases} \sigma_i = F^{2^g-1-I_i}(x_i) \text{ for } 1 \geq i \geq s/g \\ \sigma_0 = F^{\delta}(x_0) \text{ with } \delta = \sum_{i>0} I_i\,. \end{cases}$$

Signature verification

Given a public key $pk = \{v_0, v_1, \ldots, v_{s/g}\}$ and a message-signature pair (m, σ), the verifier can verify the signature with the following procedure:

1. Split the message m into s/g parts, let these parts be $m_1, m_2, \ldots, m_{s/g}$,
2. interpret each m_j as an integer I_j,
3. verify the validity of the signature using

$$\mathsf{Verify}_{pk}(m, \sigma) = \mathsf{true} \text{ if } \begin{cases} v_i = F^{I_i}(\sigma_i) \text{ for } 1 \geq i \geq s/g \\ v_0 = F^{2^g-1-\delta}(\sigma_0) \text{with } \delta = \sum_{i>0} I_i \end{cases}$$

The parameters k, t and s cannot be chosen independently, but have to satisfy $s = k\log_2(t)$. Using this, the probability can be rewritten as $2^{-\Sigma}$ with

$$\Sigma = k\big(s/k - \log_2(k) - \log_2(r)\big)\,. \tag{1}$$

As an example, for $s = 160$, $k = 16$ and $r = 1$ (and $t = 1024$), the security level is 96; for $r = 4$ the security level drops to 64. Using Eq. 1 we can compute the number r of signatures we can generate per public key, while maintaining a security level $\bar{\Sigma}$:

Algorithm 6 The HORS one-time signature scheme

Setup of system parameters

Select a cryptographic hash function H with output length $|H| = s$ and a OWF f mapping $\{0,1\}^l$ to $\{0,1\}^l$. Select the system parameters k and t such that $k \log_2(t) = s$.

Key setup

1. Generate t random values $x_1, x_2, \ldots, x_t$ with $|x_i| = l$,
2. let $sk = \{x_1, x_2, \ldots, x_t\}$,
3. compute $pk = \{f(x_1), f(x_2), \ldots, f(x_t)\}$,
4. publish pk as the one-time public key, keep sk as the one-time private key.

Signature generation

1. Let $h = H(m)$,
2. split h into k substrings $h_1, h_2, \ldots, h_k$ of length $|h_i| = \log_2(t)$,
3. interpret each h_j as an integer I_j.

The signature of message m is

$$\sigma = \mathsf{Sign}_{sk}(m) = \{x_{I_1}, x_{I_2}, \ldots, x_{I_k}\}\,.$$

Signature verification

Given a public key pk and a message-signature pair (m, σ), the verifier can verify the signature with the following procedure:

1. Let $h = H(m)$, $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$, and $pk = \{v_1, v_2, \ldots, v_t\}$,
2. split h into k substrings $h_1, h_2, \ldots, h_k$ of length $|h_i| = \log_2(t)$,
3. interpret each h_j as an integer I_j,
4. verify the validity of the signature using

$$\mathsf{Verify}_{pk}(m, \sigma) = \mathsf{true} \text{ if } f(\sigma_j) = v_{I_j} \text{for } 1 \leq j \leq k\,.$$

$$r = (1/k)2^{(s-\bar{\Sigma})/k}\,. \tag{2}$$

Finally, from Eq. (2) we can compute how many public keys we need to sign S messages:

$$\#pk = S/r = Sk2^{(\bar{\Sigma}-s)/k}\,. \tag{3}$$

As a practical example we will use two different parameter sets with roughly the same security level: HORS-20 with $(s, k, t, r) = (160, 20, 256, 1)$, providing $\Sigma = 73.5$, and HORS-18 with $(s, k, t, r) = (162, 18, 512, 2)$, providing $\Sigma = 68.9$.

3.4. One-time Public-key authentication

Two obvious disadvantages of one-time signature schemes are the size of the public key and the fact that it can only be used a limited number of times. All possible verifiers need *authenticated* copies of these public keys, i.e., they need evidence that a particular public key is related to a particular user . For example, when using LDM with $s = 160$ and $l = 80$, the total size of 1000 public keys is about 1.63 MBytes. One obvious mechanism to provide authenticated copies of this public key set is to transfer the complete set to every verifier over some authenticated channel (e.g., using a traditional digital signature such as ECDSA). The disadvantage is that every verifier has to store 1.63 Mbytes of data for every potential signer. Fortunately, there are more efficient solutions.

3.4.1. Merkle trees

Merkle proposed the use of binary trees to reduce the authentication of a large number of public keys to the authentication of a single value, i.e., the root of the tree [43].

Due to space limitations, we will limit the description of Merkle trees to a few key concepts, for more details we refer to [43].

A Merkle tree is a complete binary tree with an l-bit value associated with each node such that each interior node value is a OWF of the node values of its children. The N values that need to be authenticated are placed at the leaves of the tree. Usually these values need to be kept secret. In that case, the hash values of these secret values are placed at the leaves of the tree. The hash values $leaf_i = H(\mathsf{value})$ are called the leaves and the secret values are usually called the leaf pre-images. Every parent's node value is calculated as a OWF f of the concatenation of the two child values. Because of this construction the root of the tree can be used to authenticate the leaves. A particular leaf can be authenticated with respect to the root value and the *authentication path* of the leaf.

Authentication paths. Let sib_i be the value of the sibling of the node on height i on the path from the leaf to the root. A leaf has height 0, the OWF of two leaves has height 1, etc., and the root has height H if the tree has 2^H leaves. The authentication path is then the set $\{sib_i \mid 0 \leq i \leq H\}$.

The goal of *Merkle tree traversal* is the sequential output of the leaf values and their authentication paths. In [35], Jakobsson et al. present an algorithm which allows a time-space trade-off. When storage is minimized, the algorithm requires about $2\log_2(N)/\log_2(\log_2(N))$ invocations of f, and a maximum storage space of $1.5\ \log_2^2(N)/\log_2(\log_2(N))$ outputs of f. The Merkle tree traversal algorithm by Jakobsson starts with the calculation of the tree root. During this root calculation, the initial internal state of the algorithm is also calculated and stored in memory. This initialization requires $N - 1$ invocations of f.

3.4.2. One-way chains

In the three one-time signature schemes we have described, the public key is computed by applying the OWF f one or more times to the private key, which in turn is nothing more than a set of random values. Another way of authenticating these public keys is using one-way chains. Perrig suggests in [44] the use of these one-way chains to authenticate the public keys that are used in the BiBa (Bins and Balls) one-time signature scheme, but this idea applies equally to other signature schemes. This authentica-

tion mechanism is especially useful in the setting of a single verifier, or a set of "synchronized" verifiers (i.e., verifiers who all receive the same message-signature pairs at the same time). A typical example of the latter is *broadcast authentication* in wireless networks. All nodes within range of a sending node A have an authenticated copy of the root of the one-way chain used by A. Node A signs every message it broadcasts with a one-time signature scheme; this signature is verified by all receiving nodes at the same time. For the LDM and the LDW a new public key becomes active after every signature, for the HORS scheme this happens after r signatures. Note that the private key that was used for signature i becomes the public key for signature $i + 1$.

The private key of the three one-time signature schemes we have described is always a set of random values. When using one-time chains for public key authentication, the signer first generates the root private key (i.e., a single set of random values). He then applies the OWF f as many times as required (e.g., for LDM and HORS, the public key is obtained by applying f a single time to the private key; this means that we obtain n public/private key pairs by applying the function OWF n successive times).

The last set of values we obtain is the *first* public key that will be used. This public key is transferred in an authenticated way to all the potential verifiers. The second to last set of values we obtain is the first private key that will be used. Once this private key has been used, it becomes the public key to be used for the next message, etc., until the initial private key is used. At that time, a fresh chain has to be generated and the first public key of this chain has to be transferred to the verifiers.

Note that longer chains are only disadvantageous for the signer, and not for the verifiers. The computational effort for signature generation grows for longer chains since the signer always has to start from the start of the chains. The signer could improve this by storing multiple intermediate private keys in memory. This technique provides a means to exchange storage requirements for computation time. For verifiers the length of the chains has no influence on the performance.

4. Authenticated Key Establishment

4.1. Diffie-Hellman

As already known, key management and exchange do not scale well in the case of large number of users as the case is for sensor networks. Also the protection of keys is easily solved by use of PKC. The cost for key exchange in the case of EC operations is one point multiplication (on each side).

We stress here that all mentioned protocols include other cryptographic primitives as well, such as hash functions, MACs *etc.* but for the total cost of a protocol in the case of ECC point multiplication operation is the most substantial. Therefore, for the results we take only number of point multiplications into account.

The **Diffie-Hellman key agreement** protocol gives the first practical solution to the key exchange problem for two parties. The basic version of this protocol is as follows [22]:

Algorithm 7 Diffie-Hellman key agreement

One time setup

1. A prime p such that $(p-1)/2 = p'$, where p' is also a prime and a generator α of $\mathbb{Z}_p^*$, $2 \leq \alpha \leq p-2$ are selected and published;

Protocol messages

$A \rightarrow B : \alpha^x \bmod p \quad (1)$
$B \rightarrow A : \alpha^y \bmod p \quad (2)$

Protocol actions

1. A chooses a random secret x, $1 \leq x \leq p-2$ and sends B message (1).
2. B chooses a random secret y, $1 \leq y \leq p-2$ and sends A message (2).
3. B receives α^x and computes the shared key as $K = (\alpha^x)^y \bmod p$.
4. A receives α^y and computes the shared key as $K = (\alpha^y)^x \bmod p$.

4.2. Protocols for Authentication

Sensor node identification requires authentication as a cryptographic service. This property can be achieved by symmetric as well as asymmetric primitives. Previously known work considered mainly symmetric-key algorithms e.g. AES [25].

Here we discuss two authentication protocols. They are both written for the case of ECC because we look into performance of ECC in more detail in the last section. The anti-counterfeiting problem can also be rephrased as an authentication problem. This was explored in [18] for RFIDs.

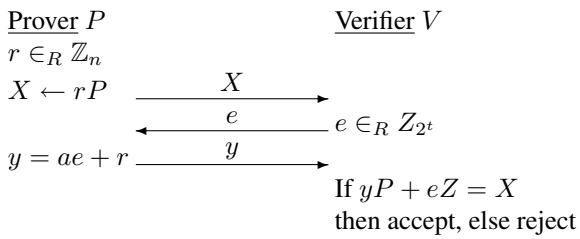
Other protocols found in the literature include Beth's identification protocol [23] and the XDL-IBI scheme in [24]. Beth's protocol only requires one point multiplication but it remains an open problem to prove its security against active adversaries. The XDL-IBI scheme also requires only one point multiplication but is only secure against passive adversaries and concurrent attacks (under a modified assumption). Thus, it seems that by analyzing both Schnorr's and Okamoto's we cover the efficiency of all *available* ID protocols based on the hardness of the DL problem.

4.2.1. Schnorr Identification Protocol based on ECDLP

This protocol is only secure against passive attacks but it is very efficient as it requires just one ECC point multiplication (for a node).

Here we specify the Schnorr identification protocol [22] based on ECDLP. In this case a node proves its identity to a station in a 3-pass protocol.

1. **Common Input:** The set of system parameters in this case consists of: (q, a, b, P, n, h). Here, q specifies the finite field, a, b, define an elliptic curve, P is a point on the curve of order n and h is the cofactor. In this case of tag authentication, most of these parameters are assumed to be fixed.
2. **Prover-Tag Input:** The prover's secret a such that $Z = -aP$.
3. **Protocol:** The protocol involves exchange of the following messages:



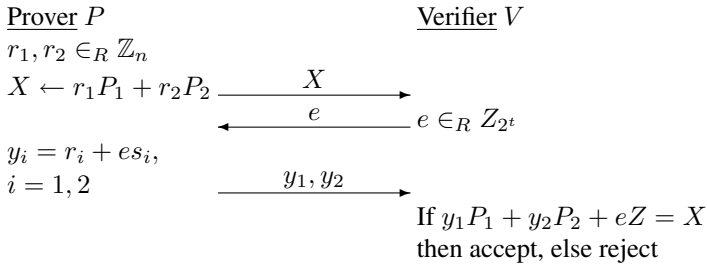
More precisely, the steps of the protocol are:

- *Commitment by a Prover-Tag:* The tag picks $r \in_R \{0, \ldots, n-1\}$, and sends $X = rP$ to the reader.
- *Challenge from a Verifier-Reader:* The reader picks a number $e \in [1, 2^t]$ and sends it to the tag.
- *Response from a Tag:* The tag computes $y = ae + r$ and sends it to the reader.
- The verifier checks that $yP + eZ$ equals X. Check: $yP + eZ = (ae + r)P + eZ = aeP + rP + (-eaP) = rP = X$

4.2.2. The Okamoto Identification Protocol based on ECDLP

Another option for secure identification is Okamoto's identification protocol. We are considering Okamoto's identification protocol as it provides security against active adversaries and it is based on the hardness of the DL problem.

1. **Common Input:** Common input is the set of system parameters consisting of (q, a, b, P_1, P_2, n, h) as before.
2. **Prover-Tag Input:** The prover's secret (s_1, s_2) such that $Z = -s_1P_1 - s_2P_2$.
3. **Protocol:** The protocol involves the exchange of the following messages:



More precisely, the steps of the protocol are:

- *Commitment by a Prover-Tag:* The tag picks $r_i \in_R \{0, \ldots, n-1\}$, $i = 1, 2$ and sends $X = r_1P_1 + r_2P_2$ to the reader.
- *Challenge from a Verifier-Reader:* The reader picks a number $e \in [1, 2^t]$ and sends it to the tag.
- *Response from a Tag:* The tag computes $y_i = r_i + es_i$, $i = 1, 2$ and sends those values to the reader.
- The verifier checks that $y_1P_1 + y_2P_2 + eZ$ equals X.
  Check: $y_1P_1 + y_2P_2 + eZ = (r_1 + es_1)P_1 + (r_2 + es_2)P + e(-s_1P_1 - s_2P_2)Z = r_1P_1 + r_2P_2 = X$

Algorithm 8 Simultaneous point multiplication

Require: $k = (k_{t-1}, \ldots, k_0)_2$, $l = (l_{t-1}, \ldots, l_0)_2$, P, Q points on the curve
Ensure: $R = kP + lQ$
1: Compute $P + Q$
2: $R \leftarrow \infty$
3: **for** *i* from $t - 1$ downto 0 **do**
4: $R \leftarrow 2R$
5: $R \leftarrow R + (k_i P + l_i Q)$
6: **end for**
7: Return(R)

In [18], the *feasibility* of the ECC version of Schnorr's identification protocol in an RFID system was investigated and area and latency estimates were provided.

As can be seen from Okamoto's scheme, the required computation for a node is of a form $kP + lQ$ *i.e.* so-called multiple-point multiplication. For the purpose of speeding-up this computation one uses Shamir's trick [9]. The scalars k and l are stored in a 2-row matrix in which each row contains binary representation of one of the scalars. All values of the form $iP + jQ$, $0 \leq i, j < 2^w$ are precalculated and stored where w is given width of the window. The algorithm to perform this so-called simultaneous point multiplication is computing at each of $\lceil \frac{t}{w} \rceil$ steps w doublings and 1 addition from the list of the precalculated values of the form $iP+jQ$. As a width of the window w is a variable that allows some trade-off, we chose the smallest window *i.e.* $w = 1$. In this way, the memory requirements are minimized as only 3 points have to be stored: $P, Q, P + Q$. The exact computation is given in Algorithm 8 [9]. The expected running time of the algorithm for $w = 1$ is $\frac{3}{4}t$ point additions and $(t - 1)$ point doublings.

5. Performance Evaluation

5.1. Performance of RSA, DSA and ECDSA

Table 2 shows power consumption measurements on a 32-bit Intel StrongARM microprocessors by Potlapally *et al.* [46]. Next to the performance of the AES, the performance of SHA-1, the DSA, and the public key algorithms RSA, DSA and ECDSA is presented. These measurements clearly show the performance gap between symmetric and asymmetric techniques.

5.2. Efficiency of one-time signature schemes

All the one-time signature and Public-key authentication schemes evaluated in this chapter are based on a general OWF f. In order to be able to provide algebraic expressions for the cost (= power requirement) of these schemes we assume that:

- the input size of f is a multiple of l bits; we will refer to a group of l bits as one "block";
- the output size of f is 1 block;
- the cost of f for an input size of t blocks is t BF (for Block Function), i.e., the cost of f grows linearly with respect to the input size.

Table 2. Power consumptions of SHA-1, AES, RSA, DSA and ECDSA on a 32-bit Intel StrongARM SA1100 @ 206MHz [46].

Operation	|key| |hash|	Power consumption	
SHA-1	160	0.76 μJ/Byte	
AES key scheduling	128	7.83 μJ	
AES encryption	128	1.21 μJ/Byte	
		Verification	Signing
RSA	1024	15.97 mJ	546.5 mJ
DSA	1024	338.02 mJ	313.6 mJ
ECDSA-$\mathbb{F}_{2^{131}}$	163	196.2 mJ	134.2 mJ

Efficient instances of the OWF f can be built from fast block ciphers or cryptographic hash functions. We assume that f maps $n \times 80$ bits to 80 bits, i.e., $l = 80$. Since collision resistance is not required from f we believe that this parameter is sufficient.

We further assume that a cryptographic hash function H is applied to all messages before they are fed to the LDM or LDW signature scheme. The output size of this hash is $|H| = s = 160$ bits. We assume that this is the same hash function that is used in the HORS scheme. As this hash function has to be applied for all three schemes, the cost of this operation is not taken into account for the efficiency evaluation.

In order to compare the efficiency of the one-time signature schemes with elliptic curve based signature schemes, we use the measurements from [46] that are presented in Table 2 on p. 17. Assuming one invocation of the function f requires one invocation of the AES block encryption algorithm, then the cost of an ECDSA verification (signature) is equal to the cost of 10^4 ($7 \cdot 10^3$) invocations of f.

Another important cost factor (certainly for one-time signature schemes) is the *communication cost*. A rigorous performance analysis of the popular Mica2 and Mica2dot motes is presented in [29]. The authors show that the effective throughput available to applications on a Mica2 mote is only 4.6 $\mathrm{kbits/s}$ (a fraction of the nominal bandwidth of 19.2 $\mathrm{kbits/s}$). In order to achieve this, the radio module of the mote requires 48 mW in receive mode and 54 mW in transmit mode. Thus, the mote uses 10.4 $\mu\mathrm{J/bit}$ in receive mode and 11.7 $\mu\mathrm{J/bit}$ in transmit mode. This results in the following assumptions we will use for the numeric evaluation:

- The size of the output of the hash function H is 160 bits ($|H| = s = 160$),
- 1 block = 80 bits,
- 1 BF = 16×1.21 μJ $= 19.36$ μJ,
- the transmission cost of 1 block = 936 μJ,
- the receiving cost of 1 block = 832 μJ.

5.2.1. Efficiency of the LDM

Looking at Alg. 4, we see that the key setup requires $s + \lceil \log_2(s) \rceil$ BF. The public and private key size is $s + \lceil \log_2(s) \rceil$ blocks. Signature generation is "free". Assuming a uniform distribution of the possible messages in the message space, on average the message and padded redundancy $\langle m, w \rangle$ will contain 50% zeros and 50% ones. This means that the average signature size is $\frac{1}{2}(s + \lceil \log_2(s) \rceil)$ blocks and that verification requires $\frac{1}{2}(s + \lceil \log_2(s) \rceil)$ BF on average.

Note that the secret key $sk = \{x_0, \ldots, x_{m/t}\}$ can be generated with a good pseudo-random generator using a single seed $\underline{sk}$. The entropy of the output of the pseudo-random generator is at most $|\underline{sk}|$, therefore we propose to use a seed with size $2|x_i|$, i.e., 2 blocks. This means that storing the secret key only requires a fraction of the total size of the private key. Obviously this is not true for the public key.

As a practical example, the total cost of signing a message, transmitting this message to the verifier and verifying the message is 148 mJ for the communications and 1.6 mJ for the computations, totalling about 150 mJ.

5.2.2. *Efficiency of the LDW*

Looking at Alg. 5, we see that the key setup requires $2(s/g)(2^g - 1)$ BF. The public and private key size is $s/g + 1$ blocks. The costs of signature generation and verification are both $(s/g)(2^g - 1)$ BF, independent of the message. The signature size is $s/g + 1$ blocks.

Notice that the computational cost grows exponentially with the group size g, while the communication cost only drops linearly with g. This indicates that performance gain, if any, will only be possible for small values of g. Table 3 shows the total cost of a signature for different values of g. This cost includes signature generation, one signature verification and one transmission from sender to receiver. The minimum cost occurs for $g = 4$, i.e., signing 4 bits with a single public/private key value pair. Using the LDW with $g = 4$ offers a 37% performance gain compared to the LDM.

Table 3. Cost of the LDW for different group sizes g (mJ).

group size g	1	2	3	**4**	5	6
communications	285	143	96	**72**	58	49
computations	6.20	9.29	14.46	**23.23**	38.41	65.05
total	291	153	111	**96**	97	114

5.2.3. *Efficiency of the HORS scheme*

We assume that the system-wide parameter k is fixed, and is not included in the public/secret key. The cost of HORS can be summarized as follows:

- The *key setup* requires t evaluations of f, resulting in a total computational cost of t BF. The public and private key size is t blocks.
- *Signing* requires no additional operations besides applying the hash function to the message. The signature size is k blocks.
- *Verifying* requires a maximum of k BF if the signature is valid (if the signature is invalid the verification process can be stopped earlier and the cost will be less).

For HORS-20 the total cost for signing a message, transmitting the signature to the verifier and verifying the message is 452.6 mJ for the communications and 387 μJ for the computations, totalling about 453 mJ. For HORS-18, this becomes 905.2 mJ for the communications and 348 μJ for the computations, totalling about 906 mJ. Note that the total cost is dominated by the transmission cost which is 3 orders of magnitude larger than the computation cost.

5.3. Efficiency of one-time signature schemes with Public-key authentication

5.3.1. Efficiency of Merkle tree authentication

In order to evaluate the cost of Merkle trees we first describe the different processes involved.

- The algorithm precalc generates the leaf pre-images and requires $precost$ BF.
- The algorithm leafcalc generates the leaves and requires $leafcost$ BF. Note that this includes generation of the leaf pre-images.
- *Root generation* is the process of computing the root node of the tree. This root node will serve as the public key of the signature scheme.
- *Authentication path generation* or *Merkle tree traversal* is the task of generating the authentication paths for successive leaves.

Next to these steps we also consider the signing and verifying processes:

- *Signing* a message consists of (1) regenerating the private key, (2) computing the signature, and (3) generating the authentication path.
- *Verifying* a signature consists of (1) verifying the authenticity of the received public key (i.e., a leaf pre-image), and (2) verifying the signature.

Authentication path generation and verification cost The fractal Merkle tree traversal algorithm presented in [35] by Jakobsson *et al.* allows a time-space trade-off. Briefly explained, the algorithm splits the original tree into subtrees of height $h \leq H$. These subtrees are constructed in such a way that the root of one tree is at the same time a leaf of a tree above it, i.e., they are stacked on top of each other. Assuming $h|H$, let $L = H/h$ be the number of subtree *levels*. Exactly one such subtree for each level is kept in memory and all these stacked subtrees together contain the authentication path for the leaf that is being authenticated at the moment. For each output of an authentication path a second new subtree for each level is being constructed. The construction is programmed in such a way that the new subtree will be finished just in time to be used to create the authentication path; at this moment the old subtree is discarded and the construction of a fresh subtree is initiated. Fractal tree traversal requires a maximum of $2(L-1)$ evaluations of f per round, and a maximum space of $2L2^{h+1} + HL/2$ memory units (each unit being the size of the output of f, i.e., 1 block). Note that according to our definition of f, one evaluation here requires 2 BF. Taking into account the cost of generating the leaves (*leafcost* BF per leaf), the computational cost of fractal tree traversal becomes $(L-1)(2 + leafcost)$ BF per round. The required space is minimized using $h = \log_2(H) = \log_2(\log_2(N))$. The authentication path generation cost then becomes

$$\frac{\log_2(N)}{\log_2(\log_2(N))}(2 + leafcost) \text{ BF per path.} \tag{4}$$

The root of the tree is computed at the initialisation phase of the fractal Merkle tree traversal algorithm. This root generation process requires the computation of the N leaves and all nodes in the tree. By cleverly scheduling the order in which these nodes are computed, the memory requirements can be limited to $\log_2(N) + 1$ blocks. The total root generation cost is

$$N(2 + \mathit{leafcost}) - 2 \text{ BF}. \tag{5}$$

Upon receipt of a leaf pre-image and the corresponding authentication path, the recipients have to compute nodes of the tree until they arrive at the root. The total cost of this verification process is

$$2\log_2(N) + (\mathit{leafcost} - \mathit{precost}) \text{ BF per path}. \tag{6}$$

5.3.2. Energy cost factors when using Merkle tree authentication

We consider the following energy consumption cost factors:

1. *Key setup cost*. The signer generates the necessary private keys and computes the root of the Merkle tree. The cost of this process is given by Eq. (5). A copy of the resulting root value is transferred to the verifiers over an authenticated channel.
2. *Cost of signing a message.* The signer has to regenerate the private/public key pair, compute the signature, and generate the authentication path for this particular public key.
3. *Cost of verifying a signature.* The verifier has to check the validity of the signature and of the public key (i.e., verifying the authentication path).
4. *Signature size.* A signature consists of the public key that was used, the signature itself, and the authentication path to authenticate the public key.

Due to space limitations, we refer to [27] for a detailed analysis and algebraic expressions of these four cost factors for every one-time signature scheme we mentioned. We refer to Sect. 5.5 for a numeric evaluation and comparison of the different energy costs of the different schemes.

5.4. Efficiency of one-way chain authentication

One-way chains may seem more efficient than Merkle trees, as no authentication paths need to be computed, exchanged and verified. On the other hand, the use of one-way chains requires rather strict synchronization between the signer and the verifiers, limiting the applications of this authentication mechanism.

The different processes involved are:

- *One-way chain generation* (key setup) is the process of generating all the hash chains. The output of this process is the root secret key sk_N and the first public key pk_1 to be used by the verifiers.
- *Active private key regeneration* is the process of computing the currently active private key.
- *Signing a message* consists of (1) regenerating the active private key and (2) generating the signature.
- *Verifying a signature* consists of verifying the received signature values by walking down the chains until known values are reached (see Sect. 3.4.2).

5.4.1. One-way chain generation and verification

A private key consists of $t+\gamma$ random values. For the LDM and HORS scheme $\gamma = 0$ and for the LDW $\gamma = 1$. These values are generated using a single seed value $\underline{sk}$, requiring

$t + \gamma$ BF. From this root private key, the signer computes the chains. Assuming we wish to sign S messages with a single set of one-way chains and assuming that a single private key can be used r times, the required length N of the chains is S/r. The total cost of the *one-way chain generation* process is

$$(t + \gamma) + N(t\alpha + \gamma\beta) \text{ BF}. \tag{7}$$

When using one-way chains, the cost of regenerating the first private key is much higher than the cost of regenerating the last private key. This is because the signer always starts from the root private key and works his way down the chains until he reaches the active private key. Therefore, we compute the *average* cost of the private key regeneration process. In total, N private keys can be verified with a single set of one-way chains. The cost of computing these N private keys consists of computing the root private key and walking down the chains until the active key is reached. The total cost of this process is

$$N(t + \gamma) + \frac{N(N-1)}{2}(t\alpha + \gamma\beta) \text{ BF}.$$

Assuming the signer keeps the active private key in memory until the r signatures are generated, the average cost of private key regeneration *per signature* is

$$\frac{t + \gamma}{r} + \frac{N-1}{2r}(t\alpha + \gamma\beta) \text{ BF}. \tag{8}$$

The cost of signing a message is the cost of generating the active private key and using this key to generate the signature. This last cost depends on the signature scheme that is used.

Assume that the verifiers obtained an authenticated copy of the first public key pk_1. Upon reception of a signature, the verifier has to walk down the chains until he reaches a known authenticated value. Once the verifier has checked the authenticity of the received signature, he only keeps the most recently used values for each chain (see Sect. 3.4.2). This way he will have to compute every value in each chain exactly once, except for the root private key and possibly some other values close to this root private key. The total signature verification cost is at most $N(t\alpha + \gamma\beta)$ BF or

$$(t\alpha + \gamma\beta)/r \text{ BF} \tag{9}$$

per signature.

5.4.2. Energy cost factors when using one-way chains

We consider the following energy consumption cost factors:

1. *Key setup cost*. The signer generates the root private key and the first public key. A copy of this public key is transferred to the verifiers over an authenticated channel.
2. *Cost of signing a message.* The signer regenerates the active private key and computes the signature.

3. *Cost of verifying a signature.* The verifier verifies the signature against the public key values he has in memory. Once the the signature has been verified, he updates the public key values.
4. *Signature size.* Only the signature itself has be transferred (public keys nor authentications paths have to be transmitted).

Again we refer to [27] for a detailed analysis and algebraic expressions, and to Sect. 5.5 for a numeric evaluation and comparison of the different energy costs of the different schemes.

5.5. Comparison

In order to compare the efficiency of the different one-time signature schemes, including the cost of communications, we will use assumptions presented in Sect. 5.2 which are based on the measurements shown in Table 2. This table also allows us to compare the one-time signature schemes with ECDSA. We have evaluated the energy cost *per signature* when we sign S messages with a single set of one-way chains or Merkle tree. The results in Fig. 1 up to Fig. 5 do not include the cost of bootstrapping the system. We assume that the verifiers have already obtained an authenticated copy of the public key material that they need. Figure 6 shows an example scenario with 10 verifiers, and includes the cost of bootstrapping the system.

Figure 1 shows how the energy cost per *signature generation* varies with the number of signatures. For Merkle based schemes, this cost includes the root generation process (Eq. (5)) next to the cost of generating a signature and the authentication path. The cost of the root generation process is divided over all S signatures. Because of the efficient authentication path generation algorithm, the cost per signature only grows very slowly with the number of signatures ($\sim \log_2(S)/\log_2(\log_2(S))$). For schemes based on one-way chains, the signing cost includes the overhead of computing the first public key (i.e., the last values of the chains, Eq. (7)) next to the cost of generating the signature. In contrast to Merkle based schemes, the cost per signature grows linearly with respect to S. This is because of the inefficient private key generation process (Eq. (8)). The ECDSA does not suffer from any overhead and therefore the energy cost per signature is constant. The first column in Table 4 shows the signing cost of the different schemes after the distances between the cost curves have stabilized (at $S = 250$). Note that the one-time cost curves continue rising, thus at some point the ECDSA will be the most efficient option. Obviously this operation point should never be used. For one-way chain based schemes the relative distance never stabilizes.

Figure 2 shows the cost per *signature verification* as a function of the number of signatures S. The verification cost for Merkle based schemes grows with $\log_2(S)$ because of the leaf verification process (Eq. (6)). However, this cost is negligible compared to the constant cost of verifying a signature, resulting in a near-constant verification cost. One-way chain based schemes have a constant verification cost. The ECDSA verification cost is also constant. The second column in Table 4 shows the verification cost of the different schemes at $S = 250$.

Comparing Fig 1 with Fig. 2 we see that the verification cost is lower than the signature generation cost for all one-time signature schemes. For one-way chain base schemes this difference grows rapidly ($\sim S$), while for Merkle based schemes the difference remains near-constant.

With respect to *communications or signature size* (Fig. 3) we see that ECDSA is most efficient as the signature size is only 320 bits and there is no overhead. The fact that we need to include the public key and the authentication path in a signature, makes Merkle based schemes less efficient than one-way chain based schemes. The HORS schemes have small signature sizes but relatively large public key sizes. Therefore, they are more efficient than both the LDM and the LDW when using one-way chains, but less efficient than the LDM when using Merkle trees. The third column in Table 4 shows the communications cost in bits/signature of the different schemes at $S = 250$.

Figures 4 and 5 show the total cost, computations and communications, for the signer and the verifier. For the signer, i.e., the transmitter, the communication cost is $11.7\ \mathrm{mJ/kbit}$, while for the verifier it is $10.4\ \mathrm{mJ/kbit}$. As the communication cost is near constant, we obtain shifted versions (with a different shift for every scheme) of Fig 1 and Fig. 2 respectively. For the signer, we see that the communications cost dominates for the Merkle based schemes. For the one-way chain based schemes the computational cost rapidly grows and starts dominating. For the verifier, the communications cost always dominates. For the ECDSA, the computational cost is much larger than the cost of communications.

Figure 6 shows the total cost per signature for an example scenario. In this scenario a single signer transmits signatures to 10 verifiers using a single broadcast message. The one-time schemes are bootstrapped using an ECDSA signature on the root of the Merkle tree or on the first public key of the one-way chains. The cost of this bootstrapping process is divided over the S signatures. Because of the initialization and bootstrapping costs, the cost of the different schemes first decreases with the number of signatures. Depending on the scheme, this decrease levels out at around 50–75 signatures. Beyond this threshold the cost per signature keeps increasing. The rate of increase depends on the *signing* cost of the particular scheme. The most efficient solution for this particular setting is HORS-18 using one-way chains with about 75 signatures per one-way chain. The last column in Table 4 shows the cost of the different schemes for $S = 75$ signatures per Merkle tree or one-way chain. Using multiple instances of HORS-18 (including the bootstrapping cost) will yield a five-fold increase in efficiency compared to using plain ECDSA. Figure 5 shows that ECDSA is less efficient than HORS-18 for the verifier. This means that the efficiency difference between ECDSA and HORS-18 will grow further in the case of more verifiers.

Summarizing we see that (1) one-way chains are most efficient for signature verification, (2) ECDSA and Merkle trees are most efficient for signature generation, and (3) ECDSA is the best candidate with respect to communications costs. If we take the average costs of all one-time signature schemes in Table 4, we obtain $92.8\ \mathrm{mJ/signature}$ for signature generation, $10.4\ \mathrm{mJ/signature}$ for verification and $8.9\ \mathrm{mJ/signature}$ for communications.[2] This shows that signature generation is about ten times more demanding than verification or communications.

5.6. An example for a hardware assisted approach for PKC: low-cost and low-power ECC processor

In this section we also roughly estimate PKC performance for the case of ECC processor that performs the point multiplication. This operation is the foundation of all required

[2] At a rounded $10\ \mathrm{mJ/kbit}$.

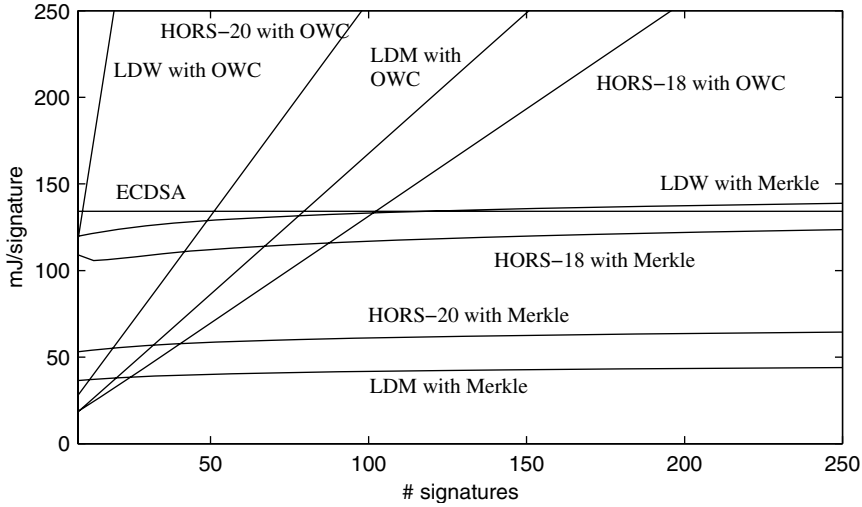


Figure 1. Signing efficiency of one-time signature schemes.

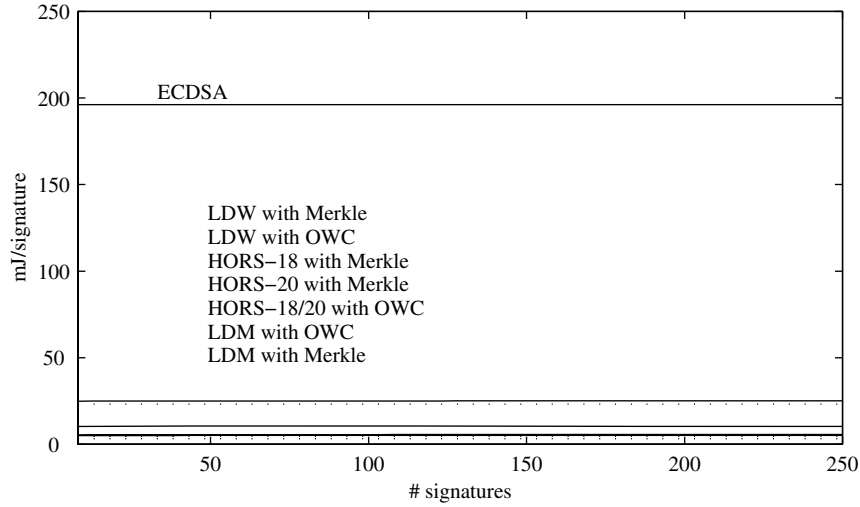


Figure 2. Verification efficiency of one-time signature schemes.

protocols as shown above. We discuss design criteria that should lead to a feasible ECC solution for sensor network applications. We do not give details about architectures as hardware implementations are discussed in another chapter.

The main operation in any elliptic curve-based primitive is the scalar multiplication. The hierarchical structure for operations required for implementations of ECC is as follows. Point multiplication is at the top level. At the next (lower) level are the point group operations (addition and doubling). The lowest level consists of finite field operations such as addition, subtraction, multiplication and inversion required to perform the group operations. The levels of ECC hierarchy usually map to typical architectures for low-

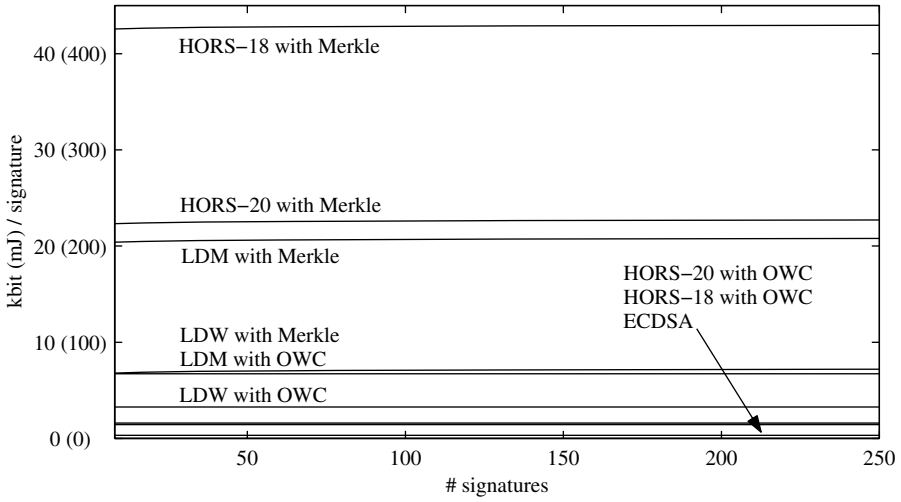


Figure 3. Communication efficiency of one-time signature schemes. The energy cost (mJ/signature) is a rough approximation at 10 mJ/kbit.

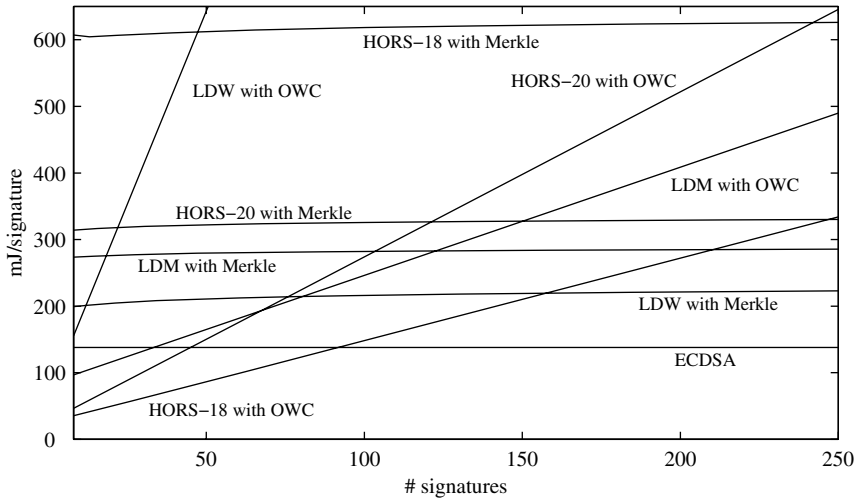


Figure 4. Energy consumption of signer (communications and computations).

cost applications in the following way: point multiplication is the building block for all ECC-based protocols and it is realized as a control unit, memory (RAM and ROM) and an arithmetic unit. In ROM the ECC parameters and the constants can be stored. On the other hand, RAM contains all input/output and intermediate variables [18]. Mainly all proposals for hardware implementations of ECC for light weight cryptography deal with ECC over binary fields.

We address each of those levels separately and for on all levels we discuss some

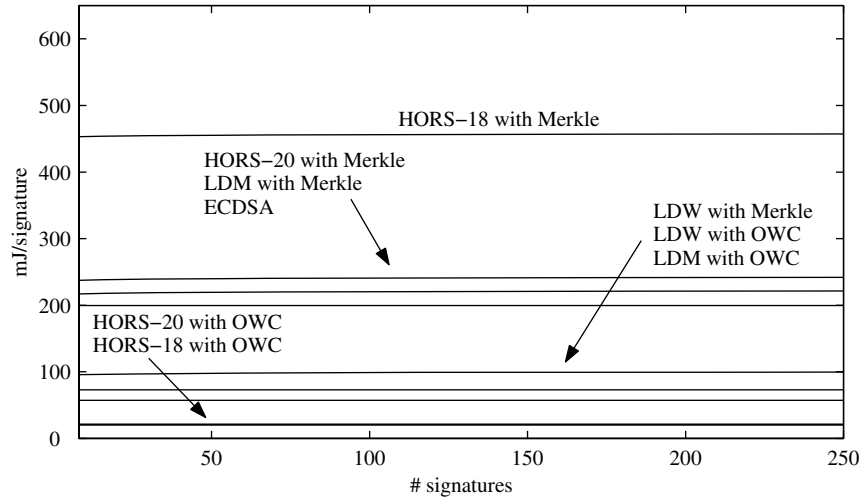


Figure 5. Energy consumption of verifier (communications and computations).

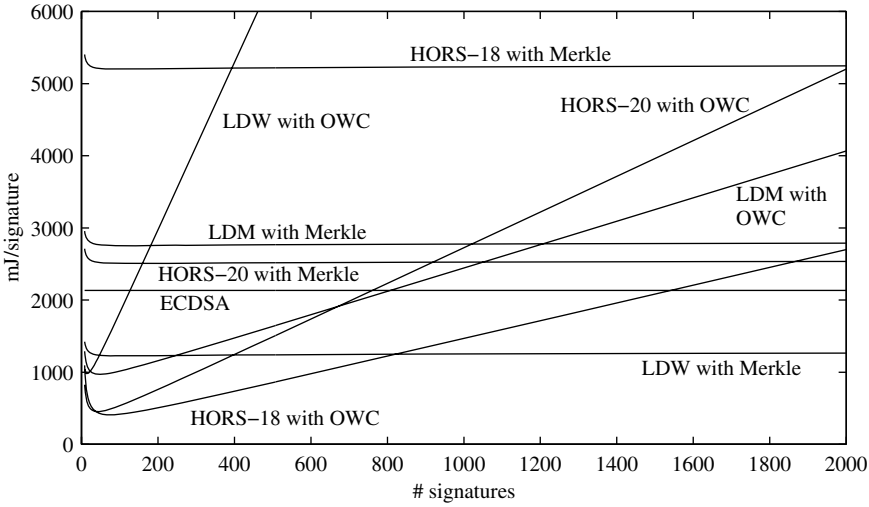


Figure 6. Overall energy consumption of one-time signature schemes. This setting assumes one signer and ten verifiers.

algorithmic and architectural choices leading to a feasible solution. One important issue is flexibility. Namely, ECC allows a great choice of fields, curves and other parameters and algorithms. However, it is necessary to limit the flexibility in order to minimize the area and maximize the performance. In this way for binary fields one choice for the field and irreducible polynomial results in the smallest possible arithmetic unit for the given security level. This is, of course dictated by the field size. Another examples leading to optimization in the area and performance are use of special curves (e.g. Koblitz curves), specific algorithms for point multiplication (such as windowing methods), special pro-

Table 4. Efficiency of one-time signature schemes.

# signatures	Signing 250 [mJ]	Verification 250 [mJ]	Comms 250 [bit]	Total 75 [mJ]
LDM with Merkle	**42.3**	5.19	20,797	2,509.2
LDW with Merkle	138.9	25.09	7,197	1,226.7
HORS-20 with Merkle	64.5	5.65	22,717	2,756.6
HORS-18 with Merkle	123.7	10.53	42,957	5,202.0
LDM with OWC	↑	**3.25**	6,720	986.1
LDW with OWC	↑	23.23	3,280	1,540.6
HORS-20 with OWC	↑	4.96	1,600	488.1
HORS-18 with OWC	↑	5.61	1,440	**408.3**
ECDSA	134.2	196.20	**320**	2,133.2

The entries are cost per signature.

Algorithm 9 Algorithm for point multiplication

Require: an integer $k > 0$ and a point $P(x, y)$
Ensure: $Q = kP$
$k \leftarrow k_{l-1}, ..., k_1, k_0$
$X_1 \leftarrow x,\ Z_1 \leftarrow 1, X_2 \leftarrow x^4 + b, Z_2 \leftarrow x^2.$
for i from $l - 2$ downto 0 **do**
If $k_i = 1$ then
$x(P_1) \leftarrow x(P_1 + P_2), x(P_2) \leftarrow x(2P_2)$
Else
$x(P_2) \leftarrow x(P_2 + P_1), x(P_1) \leftarrow x(2P_1)$
end for
Return $x(P_1)$

jective coordinates, irreducible polynomial (e.g. trinomial or pentanomial). Here we discuss some special choices such as Montgomery ladder for scalar multiplication, projective coordinates of Lopez and Dahab etc. We also deal here only with the field $\mathbb{F}_{2^{163}}$, which is recommended by many standards and it provides the same level of security as RSA of 1024 bits (or higher [15,16]). We stress again, that fixing arithmetic as well as parameters results in a compact PKC solution for pervasive security.

5.6.1. Minimizing memory requirements

For the point multiplication a good choice is the method of Montgomery (Algorithm 9) [11] that maintains the relationship $P_2 - P_1$ as invariant. It uses a representation where computations are performed on the x-coordinate only in affine coordinates (or on the X and Z coordinates in projective representation). That fact allows one to save registers which is one of the main criteria for obtaining a compact solution.

The formulas for point addition/doubling that are suitable for Algorithm 9 are given in the work of Lopez and Dahab [12]. The original formulas in [12] require 2 intermediate variables but it is possible to eliminate one more intermediate register [14]. The formulae for point operations in that case are shown in Algorithm 10. The performance remains intact as one variable can be eliminated by simply reordering the field operations.

Algorithm 10 EC point operations that minimize the number of registers

Require: X_i, Z_i, for $i = 1, 2$, $x_4 = x(P_2 - P_1)$
Ensure: $X(P_1 + P_2) = X_2$, $Z(P_1 + P_2) = Z_2$
1: $X_2 \leftarrow X_2 \cdot Z_1$
2: $Z_2 \leftarrow X_1 \cdot Z_2$
3: $T \leftarrow X_2 \cdot Z_2$
4: $Z_2 \leftarrow Z_2 + X_2$
5: $Z_2 \leftarrow {Z_2}^2$
6: $X_2 \leftarrow x_4 \cdot Z_2$
7: $X_2 \leftarrow X_2 + T$

Require: $c \in \mathbb{F}_{2^n}$, $b = c^2$, X_1, Z_1
Ensure: $X(2P_1) = X_1$, $Z(2P_1) = Z_1$,
1: $T \leftarrow c$
2: $X_1 \leftarrow X_1^2$
3: $Z_1 \leftarrow Z_1^2$
4: $T \leftarrow T \cdot Z_1$
5: $Z_1 \leftarrow X_1 \cdot Z_1$
6: $T \leftarrow T^2$
7: $X_1 \leftarrow X_1^2$
8: $X_1 \leftarrow X_1 + T$

Algorithm 10 requires only one intermediate variable T, which results in 5 registers in total. Namely, the required registers are for the storage of the following variables: X_1, X_2, Z_1, Z_2 and T. Also, the algorithm shows the operations and registers required if the key-bit $k_i = 0$. Another case is completely symmetric and it can be performed accordingly. More precisely, if the addition operation is viewed as a function $f(X_2, Z_2, X_1, Z_1) = (X_2, Z_2)$ for $k_i = 0$ due to the symmetry for the case $k_i = 1$ we get $f(X_1, Z_1, X_2, Z_2) = (X_1, Z_1)$ and the correct result is always stored in the first two input variables. This is possible due to the property of scalar multiplication based on Algorithm 9. When Algorithm 9 deploys Algorithm 10 we get the following number of operations in $\mathbb{F}_{2^n}$:

$$\begin{aligned} \#registers &= 5 \\ \#multiplications &= 11\lfloor log_2 k \rfloor + 2 \\ \#additions &= 6\lfloor log_2 k \rfloor + 1 \end{aligned}$$

5.6.2. Reduction of control logic

The point multiplication and point addition/doubling can be implemented as Finite State Machines (FSMs). The optimization in the number of register requires frequent checking of the value of a key-bit k_i. Namely, due to necessary savings in memory the same registers can be used for both point operations and since $k_i \in \{0, 1\}$ in the both cases one has to choose which variable to write/read in available memory locations. This fact enlarges the size of the control logic block but memory requirements are minimized as discussed above.

5.6.3. Simplifying arithmetic unit

From the formulae for point operations as given in Algorithm 10 it is evident that one needs to implement only multiplications and additions. Squaring can be considered as a special case of multiplication in order to minimize the area and inversion is avoided by use of projective coordinates. We assume that one inversion that is necessary for conversion of projective to affine coordinates can be computed at the base station's side. Note also that, if necessary, the one inversion that is required can be calculated by use of multiplications i.e. by means of Fermat's theorem [10].

Here we consider polynomial bases, where the elements of $\mathbb{F}_{2^n}$ are polynomials of degree at most $n-1$ over $\mathbb{F}_2$, and arithmetic is carried out modulo an irreducible polynomial $f(x)$ of degree n over $\mathbb{F}_2$. In this case the basis elements have the form $1, \omega, \omega^2, \ldots, \omega^{n-1}$ where ω is a root in $\mathbb{F}_{2^n}$ of the irreducible polynomial $f(x)$ of degree n over $\mathbb{F}_2$. According to this representation an element of $\mathbb{F}_{2^n}$ is a polynomial of length n and can be written as: $a(x) = \sum_{i=0}^{n-1} a_i x^i = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1 x + a_0$, where $a_i \in \mathbb{F}_2$.

So-called classical modular multiplication is typically based on the following equation:

$$\begin{aligned} a(x) \cdot b(x) &= (a_{n-1}x^{n-1} + \cdots + a_1 x + a_0) \cdot b(x) \bmod f(x) \\ &= (\cdots (a_{n-1}b(x)x + a_{n-2}b(x))x + \cdots \\ &\quad + a_1 b(x))x + a_0 b(x) \bmod f(x) \end{aligned} \tag{10}$$

which illustrates the Horner scheme for multiplication. This scheme is the basis of the Most Significant Bit-First (MSB) multiplier [13].

The modular addition operation in binary fields is simply the XOR operation which can be performed in one clock cycle. By reusing the logic for multiplication, it is possible to implement modular addition by adding only some more control logic to the existing multiplier [17].

More details about the architecture following the principles above are given in [14].

5.6.4. Performance Estimates and Discussion

In this section we list some numbers for the latency and power consumption for ECC algorithms with the choices for arithmetic as described above [14]. The architecture we refer to has a possibility to deploy the multiplier with various digit-sizes d [17]. For the point multiplication we used Algorithm 1 and for point operations Algorithm 2. The numbers for power assume the operating frequency of 500 kHz for the field $\mathbb{F}_{2^{163}}$. With this frequency the power stays below 30 μW which is assumed to be acceptable for sensor networks applications.

The results for the total number of cycles of one point multiplication for fields $\mathbb{F}_{2^{131}}$ and $\mathbb{F}_{2^{163}}$ are given in Table 5. To calculate the time for one point multiplication we need

Table 5. The number of cycles required for one point multiplication for ECC [14].

Field size	*d*=1	*d*=2	*d*=3	*d*=4
131	191 750	98 800	68 770	53 040
139	215 694	112 470	76 038	59 340
151	254 250	109 200	74 880	57 720
163	295 974	151 632	103 518	80 352

an operating frequency. However, the frequency that can be used is strictly influenced by the total power. We assumed an operating frequency of 500 kHz as suggested in [2] in order to estimate the actual timing. We get 106 ms for the best case of ECC over $\mathbb{F}_{2^{131}}$ ($d = 4$) and 160.70 ms for the best case of ECC over $\mathbb{F}_{2^{163}}$ ($d = 4$).

The current generation of sensor networks is powered by batteries, so ultra-low power circuitry is a must for these applications. For low-power consumption it is neces-

sary to minimize the switching or dynamic power, the circuit size (for the current CMOS technology) and the operating frequency. This can be achieved by architectural decisions, which we discussed above. However, besides switching power, energy efficiency is even more crucial as sensor nodes are still battery operated. More precisely, the metric that is typically used and that should be minimized accordingly is energy per processed bit E. This value can be calculated as:

$$E = \frac{P}{throughput} \left[\frac{J}{bit}\right] .$$

The results for the total power consumption and for energy per bit are given in Figure 7 and Figure 8 respectively for the field $\mathbb{F}_{2^{163}}$. We can observe that both contributions to the total power *i.e.* static and dynamic are close to each other. We also notice that the energy per bit improves as d increases. This can be explained due to a decrease in number of cycles, which is more dominant than an increase in power consumption for higher d.

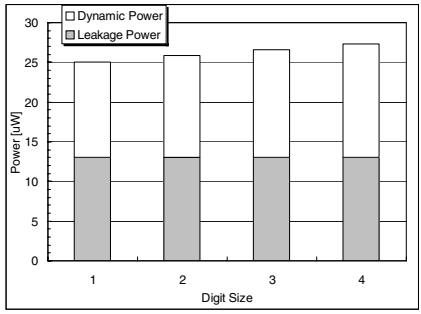


Figure 7. The power consumed for various digit sizes for ECC over $\mathbb{F}_{2^{163}}$.

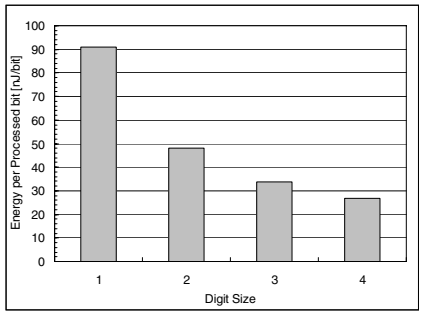


Figure 8. Results for energy per encrypted bit for all digit sizes.

The point multiplication is the most consuming operation in ECC-based protocols so we do not need to take other operations into account. Since each protocol consists of a certain number of point multiplication (usually one or two) we can estimate costs per ECC-protocol. As an example we estimate the energy per message for ECDSA (see Table 6). In this way we can also roughly estimate the energy costs per protocol.

Table 6. Energy per message for ECDSA for the field $\mathbb{F}_{2^{163}}$.

Protocol	d=1	d=2	d=3	d=4
ECDSA-sign	14.822 μJ	7.837 μJ	5.509 μJ	4.393 μJ
ECDSA-verify	26.643 μJ	15.674 μJ	11.018 μJ	8.785 μJ

Published results for area, power and latency for PKC are compared in Table 7. The performances given for ECC are for one point multiplication.

Table 7. Comparison with other related work.

Ref.	PKC	Area [$gates$]	Techn. [μm]	f [kHz]	Perf. [ms]	P [μW]
[7]	ECC over $\mathbb{F}_{2^{131}}$	11 969.93	0.35	13 560	18	-
[1]	ECC over $\mathbb{F}_{p_{100}}$	18 720	0.13	500	410.45	< 400
[2]	NTRU	2 850	0.13	500	58.45	< 21
[6]	ECC over $\mathbb{F}_{2^{191}}$, $\mathbb{F}_{p_{192}}$	23 000	0.35	68 500	9.89	n.a.
[14]	$\mathbb{F}_{2^{131}}$	8104	0.13	500	106	< 30

Acknowledgements

The work described in this document has been partly financially supported by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

It was also supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), by FWO projects EMA G.0475.05 and BBC G.0300.07, by the IBBT-QoE project of the IBBT and by the K.U.Leuven-BOF (OT/06/40).

The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

[1] G. Gaubatz, J.-P. Kaps, E. Öztürk and B. Sunar, "State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks", 2nd IEEE International Workshop on Pervasive Computing and Communication Security (PerSec 2005).

[2] G. Gaubatz, J.-P. Kaps and B. Sunar, "Public Key Cryptography in Sensor Networks - Revisited", 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004).

[3] L. Uhsadel, A. Poschmann, and C. Paar, "Enabling Full-Size Public-Key Algorithms on 8-bit Sensor Nodes", In 4th European Workshop on Security and Privacy in Ad hoc and Sensor Networks, Lecture Notes in Computer Science 4572, S. Capkun, C. Meadows, and F. Stajano (eds.), Springer-Verlag, pp. 73-86, 2007, Springer-Verlag.

[4] L. Batina, D and Hwang, A. Hodjat, B. Preneel and I. Verbauwhede, "Hardware/Software Co-design for Hyperelliptic Curve Cryptography (HECC) on the 8051 μP", Proceedings of 7th International Workshop on Cryptographic Hardware and Embedded Systems CHES 2005, eds. J. R. Rao and B. Sunar, LNCS 3659 pp. 106-118, Springer-Verlag.

[5] A. Hodjat, L. Batina, D. Hwang and I. Verbauwhede, "HW/SW Co-design of a Hyperelliptic Curve Cryptosystem using a μCode Instruction Set Coprocessor", Elsevier Science Integration the VLSI Journal, vol. 1, nr. 40, pp.45-51, 2006.

[6] J. Wolkerstorfer, "Scaling ECC Hardware to a Minimum", In ECRYPT workshop - Cryptographic Advances in Secure Hardware - CRASH 2005, Leuven, Belgium, Invited talk.
[7] S. Kumar and C. Paar, "Are standards compliant Elliptic Curve Cryptosystems feasible on RFID?", Proceedings of Workshop on RFID Security, 2006, Graz, Austria.
[8] I. Blake, G. Seroussi and N.P. Smart, "Elliptic Curves in Cryptography", Cambridge University Press, 1999, London Mathematical Society Lecture Note Series.
[9] D. Hankerson, A. Menezes and S. Vanstone, "Guide to Elliptic Curve Cryptography", Springer-Verlag, 2004
[10] N. Koblitz, "A Course in Number Theory and Cryptography", Springer-Verlag, 1994
[11] P. Montgomery, "Speeding the Pollard and Elliptic Curve Methods of Factorization", Mathematics of Computation, 1987, Vol. 48, nr. 177, pp. 243-264.
[12] J. López and R. Dahab, "Fast Multiplication on Elliptic Curves over GF(2^m)", Proceedings of 1st International Workshop on Cryptographic Hardware and Embedded Systems CHES 1999, eds. Ç.K. Koç and C. Paar, LNCS 1717 pp. 316-327. Springer-Verlag.
[13] T. Beth and D. Gollmann, "Algorithm Engineering for Public Key Algorithm", IEEE Journal on Selected Areas in Communications, Vol. 7, nr. 4, pages 458-465, 1989.
[14] L. Batina, N. Mentens, K. Sakiyama, B. Preneel and I. Verbauwhede, "Low-cost Elliptic Curve Cryptography for wireless sensor networks", Proceedings of Third European Workshop on Security and Privacy in Ad hoc and Sensor Networks, eds. L. Buttyan, V. Gligor and D. Westhoff, LNCS 4357 pp. 6-17, Springer-Verlag, 2006.
[15] A. Lenstra and E. Verheul, "Selecting cryptographic key sizes", Proceedings of Third International Workshop on Practice and Theory in Public Key Cryptography (PKC 2000), eds. H. Imai and Y. Zheng, LNCS 1752, pp. 446-465, Springer-Verlag.
[16] ECRYPT, "ECRYPT Yearly Report on Algorithms and Keysizes (2006)".
[17] K. Sakiyama, L. Batina, N. Mentens, B. Preneel and I. Verbauwhede, "Small-footprint ALU for public-key processors for pervasive security", Proceedings of Workshop on RFID Security 2006, Graz, Austria.
[18] P. Tuyls and L. Batina, "RFID-tags for Anti-Counterfeiting", Topics in Cryptology - CT-RSA 2006, ed. D. Pointcheval, LNCS 3860, pp. 115-131, Springer Verlag.
[19] J. Hoffstein, J. Pipher and J.H. Silverman, "NTRU: a ring based public key cryptosystem", In Proceedings of ANTS III, 1998, LNCS 1423, pp. 267Ű288. Springer-Verlag.
[20] ECRYPT, "Hardness of the Main Computational Problems Used in Cryptography", Deliverable 1.2, 2007.
[21] D. Johnson and A. Menezes, The Elliptic Curve Digital Signature Algorithm (ECDSA), Department of Combinatorics & Optimization, 2000, `http://www.cacr.math.uwaterloo.ca`.
[22] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1997.
[23] T. Beth, *Efficient Zero-Knowledge Identification Scheme for Smart Cards*, Advances in Cryptology — EUROCRYPT'88, ed. C. G. Günther, 1988, pp. 77-84.
[24] M. Bellare, C. Namprempre and G. Neven, Security proofs for identity-based identification and signature schemes, Advances in Cryptology — Eurocrypt 2004, eds. C. Cachin and J. Camenisch, LNCS 3027, 2004, pp. 268-286, Springer-Verlag.
[25] M. Feldhofer, S. Dominikus and J. Wolkerstorfer, Strong Authentication for RFID Systems using the AES Algorithm, Proceedings of 6th International Workshop on Cryptographic Hardware in Embedded Systems (CHES), eds. M. Joye and J. -J. Quisquater, LNCS 3156, pp. 357-370, 2004, Springer-Verlag.
[26] P. Nguyen and N. Gama, New Chosen-Ciphertext Attacks on NTRU. In Proceedings of PKC 2007, eds. T. Okamoto and X. Wang, LNCS 4450, 2007, pp. ?, Springer-Verlag.
[27] S. Seys Cryptographic algorithms and protocols for security and privacy in wireless ad hoc networks. PhD thesis, *Katholieke Universiteit Leuven*, B. Preneel (promotor), 172+37 pages, 2006.
[28] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In *Topics in Cryptology – RSA Conference Cryptographers' Track (RSA-CT 2001)*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer-Verlag, 2001.
[29] G. Anastasi, A. Falchi, A. Passarella, M. Conti, and E. Gregori. Performance measurements of motes sensor networks. In *Proceedings of the 7th International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2004)*, pages 174–181. ACM Press, 2004.
[30] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1th ACM Conference on Computer and Communications Security (CCS 1993)*, pages 62–73. ACM Press, 1993.

[31] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology - EUROCRYPT 1994*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1995.

[32] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In *Advances in Cryptology - EUROCRYPT 1996*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–414. Springer-Verlag, 1996.

[33] J. N. E. Bos and D. Chaum. Provably unforgeable signatures. In *Advances in Cryptology - CRYPTO 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 1–14. Springer-Verlag, 1993.

[34] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[35] M. Jakobsson, T. Leighton, S. Micali, and M. Szydlo. Fractal Merkle tree representation and traversal. In *Topics in Cryptology – RSA Conference Cryptographers' Track (RSA-CT 2003)*, volume 2612 of *Lecture Notes in Computer Science*, pages 314–326. Springer-Verlag, 2003.

[36] D. Johnson and A. Menezes. The elliptic curve digital signature algorithm (ECDSA). Technical report CORR 99-34, Departement of Combinatorics & Optimizations, University of Waterloo, Canada, 1999. Updated: 2000/02/24.

[37] L. Lamport, "Constructing digital signatures from a one-way function," Technical Report CSL-98, SRI International, 1979.

[38] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001.

[39] W. Mao. *Modern Cryptography – Theory & Practice*. Prentice Hall PTR, 2004.

[40] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[41] R. C. Merkle. *Secrecy, Authentication and Public Key Systems*. UMI Research Press, 1982.

[42] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology - CRYPTO 1987*, vol. 293 of *Lecture Notes in Computer Science*, pp. 369–378, Springer-Verlag, 1987.

[43] R. C. Merkle. A certified digital signature. In *Advances in Cryptology - CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer-Verlag, 1990.

[44] Adrian Perrig. The BiBa one-time signature and broadcast authentication protocol. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS 2001)*. ACM Press, 2001.

[45] PKCS #1 version 2.1: RSA cryptography standard. Public-Key Cryptography Standard 1, RSA Laboratories, 2002.

[46] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha. Analyzing the energy consumption of security protocols. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design (ISLPED 2003)*, pages 30–35, 2003.

[47] M. O. Rabin, "Digitalized signatures," *Foundations of Secure Computation*, pp. 155–168, 1978.

[48] L. Reyzin and N. Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In *Proceedings of the 7th Australian Conference on Information Security and Privacy*, volume 2384 of *Lecture Notes in Computer Science*, pages 144–153. Springer-Verlag, 2002.

[49] Ronald L. Rivest, Adi Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[50] SEC 2: Recommended elliptic curve domain parameters. Version 1.0, Standards for Efficient Cryptography Group, 2000.

[51] V. Shoup. A proposal for an ISO standard for public key encryption. Version 2.1, IBM Zurich research lab, 2001.

Wireless Sensor Network Security
J. Lopez and J. Zhou (Eds.)
IOS Press, 2008

Key Management in Wireless Sensor Networks

Dr. Seyit Ahmet ÇAMTEPE [a,1], Assoc. Prof. Bülent YENER [b]
[a] *DAI-Labor - Technische Universität Berlin, Sekretariat TEL 14, Ernst-Reuter-Platz 7, 10587 Berlin, Germany.*
ahmet.camtepe@dai-labor.de
[b] *Rensselaer Polytechnic Institute, Computer Science Department, 110 8th Street, Troy, NY 12180-3590, USA.*
yener@cs.rpi.edu

Abstract. This chapter presents a comparative survey of recent key management (key distribution, discovery, establishment and update) solutions for wireless sensor networks. We consider both distributed and hierarchical sensor network architectures where unicast, multicast and broadcast types of communication take place. Probabilistic, deterministic and hybrid key management solutions are presented, and we determine a set of metrics to quantify their security properties and resource usage such as processing, storage and communication overheads. We provide a taxonomy of solutions, and identify trade-offs in these schemes to conclude that there is no one-size-fits-all solution.

Keywords. Wireless sensor networks, key pre-distribution, shared-key discovery, key establishment, key update, dedicated pair-wise key, reusable pair-wise key, group-wise key, network-wise key, key-pool, key-chain, symmetric balanced incomplete block design, generalized quadrangle, key matrix, master key, polynomial key share.

1. Introduction

Sensors are inexpensive, low-power devices with limited resources. They are small in size, and have wireless communication capabilities within short distances. A sensor node typically contains a power unit, a sensing unit, a processing unit, a storage unit and a wireless transmitter / receiver. A wireless sensor network (WSN) is composed of large number of sensor nodes which have limited power, computation, storage and communication capabilities. Sensor nodes are deployed in controlled (e.g., home, office, warehouse and forest) or in uncontrolled environments (e.g., hostile or disaster areas and toxic regions). If the environment is known and controlled, then the deployment might be achieved manually to establish an infrastructure. However, manual deployments become infeasible or even

[1] This work has been done during the Ph.D. study of Seyit Ahmet Çamtepe at Computer Science Department of Rensselaer Polytechnic Institute under the supervision of Assoc. Prof. Bülent Yener.

impossible as the number of nodes increases. If the environment is uncontrolled, or the WSN is very large, the deployment has to be performed by randomly scattering sensor nodes to the target area. It may be possible to provide denser sensor deployments at certain spots, but exact positions of the sensor nodes can not be controlled. Thus, network topology can not be known precisely a priori to the deployment. Although the topology information can be obtained by using mobile sensor nodes and self-deployment protocols as proposed in [1] and [2], these solutions are not feasible for a large scale WSN.

Key management problem in a wireless sensor networks (WSN) can be decomposed into four phases. The first is the *key distribution* or *pre-distribution* phase where secret keys are distributed to sensor nodes for use with the security mechanisms (i.e., confidentiality, authentication and integrity). In a large scale WSN, it may be infeasible, or even impossible in uncontrolled environments, to visit large number of sensor nodes and change their security configuration. Thus, keys and keying materials may be pre-distributed to sensor nodes in a central location a priori to the deployment. This phase is also named as *key setup*. The second is the *shared-key discovery* phase, which starts after the sensor network deployment, each sensor node discovers its *neighbors* and a common key with each of them. The third is the *key establishment* phase where each pair of neighboring nodes, which do not have common keys, establish one or more keys. Key establishment between two nodes can be achieved by using pre-distributed keying materials and by exchanging messages directly over their insecure wireless link or over one or more secure paths on which each link is secured with a secret key. Sensor nodes have a limited life time, and they are subject to variety of attacks including node capture. New sensor nodes may be deployed and security materials on existing ones may need to be updated. Thus, the fourth is the *key update* phase where the secret keys which are used to secure the links between neighboring nodes are updated. We classify and evaluate key management solutions by considering following properties:

1. *Underlying network architecture.* In *distributed WSN*, there is no resource rich member, and sensor nodes have equivalent capabilities. In *hierarchical WSN*, there are one or more resource rich central stations, and there is a hierarchy among the sensor nodes based on their capabilities.
2. *Communication style.* A secure unicast communication between a pair of neighboring nodes requires a pair-wise key shared between them. A *reusable pair-wise key* is used to secure the unicast communication between more than one pairs of neighboring nodes. Disadvantage is that more than one links are compromised when a reusable pair-wise key is compromised. For improved security, a *dedicated pair-wise key* may be assigned to each pair of neighboring nodes. A secure multicast communication within a group of sensor nodes requires a *group-wise key*, and a secure broadcast communication within a WSN requires a *network-wise key*.
3. *Key pre-distribution method.* Keys and keying materials are distributed to sensor nodes based on a probabilistic, deterministic or hybrid algorithm.
4. *Key discovery and establishment method.* A set of solutions pre-distribute a list of keys, called a key-chain, to each sensor node, and a pair or a group of sensor nodes can secure their communication if they have a key

in common. Other solutions pre-distribute keying materials (i.e., one-way hash functions, pseudo-random number generators, partial key matrices and polynomial shares). A pair or a group of sensor nodes can use these materials to securely generate a common key.

Terms and Notations

Table 1 lists abbreviations in use. Following terms are used throughout this chapter:

- *Credentials* are keys and keying materials,
- *Group-wise key* is used to secure multicast communication among a group of sensor nodes over single or multi-hop wireless links,
- *Key-chain* is a list of keys or keying materials which are stored on a sensor node,
- *Key graph* is an undirected graph which has a node for each sensor node, and there is an edge in between two nodes if the corresponding sensor nodes share a key to secure their communication,
- *Keying materials* are any kind of public or private information and cryptographic algorithms which are used to generate keys,
- *Key-pool* is the list of all keys or keying materials which are used in the WSN,
- *Key reinforcement* is establishing a fresh session key between two sensor nodes by using existing link or path-key,
- *Link-key* is used to secure communication over a direct wireless link,
- *Network-wise key* is used to secure broadcast messages,
- *Pair-wise key* is used to secure unicast communication between a pair of sensor nodes over single or multi-hop wireless links,
- *Path-key* is used to secure communication over multi-hop wireless links, through one or more sensor nodes,
- *Physical graph* is an undirected graph which has an edge in between two nodes if the corresponding sensor nodes are within each others radio range,
- *Secure graph* is an undirected graph which has an edge in between two nodes if the corresponding sensor nodes have an edge in between in both key and physical graphs.

Organization

The organization is as follows: in Section 2 network models, security vulnerabilities and requirements are discussed, in Sections 3 and 4 key management solutions for distributed and hierarchical WSN are presented respectively, and in Section 5 a taxonomy and a comparison of key management solutions in WSN are presented.

Abbreviations			
KDC	Key Distribution Center	N	WSN size
WSN	Wireless Sensor Network	KP	Key-Pool
HWSN	Hierarchical WSN	KC	Key-Chain
DWSN	Distributed WSN	K	Key
MAC	Message Authentication Code	BS	Base Station
PRF	Pseudo Random Function	S	Sensor node
ENC	Encryption	RN	Random Nonce
DAG	Directed Acyclic Graph	P	Polynomial

Table 1. Abbreviations. Functions *MAC* and *ENC* accept a key and a message as the parameter. Function *PRF* accepts a seed to generate a random number or a key in which case part of the seed must be kept secret.

2. Network Models and Security Issues

2.1. Network Models

WSN communication usually occurs in ad hoc manner, and shows similarities to wireless ad hoc networks. Likewise, WSNs are dynamic in the sense that radio range and network connectivity changes by time; sensor nodes die and new nodes may be added to the network. However, WSNs are more constrained, denser, and may suffer from (or take advantage of) redundant information. In general, WSNs are organized in hierarchical or distributed structures as shown in Figure 1.

In a Hierarchical WSN (HWSN), there is a hierarchy among the nodes based on their capabilities: base stations, cluster heads and sensor nodes. Base stations are many orders of magnitude more powerful than sensor nodes and cluster heads. A base station is typically a gateway to another network, a powerful data processing-storage center, or an access point for human interface. Base stations collect sensor readings, perform costly operations on behalf of sensor nodes and manage the network. In some key management solutions, base stations are assumed to be trusted and are used as key distribution centers. Sensor nodes are deployed around one or more hop neighborhood of the base stations. They form a dense network where a cluster of sensors located within a specific area may provide similar or close readings. Nodes with better resources, cluster heads, may be used to collect and merge local traffic, and to send it to base stations. Transmission power of a base station is usually sufficient to reach all sensor nodes, but sensor nodes depend on ad hoc communication to reach base stations. Thus, data flow in such networks may be: (i) pair-wise (unicast) among pairs of sensor nodes and from sensor nodes to base stations, (ii) group-wise (multicast) within a cluster of sensor nodes, and (iii) network-wise (broadcast) from base stations to sensor nodes.

In a Distributed WSN (DWSN), there is no fixed infrastructure, and network topology is unknown a priori to deployment. Sensor nodes are randomly scattered over a target area. Once they are deployed, each sensor node scans its radio coverage area to locate its neighbors. Data flow in DWSN is similar to data flow in HWSN with a difference that network-wise (broadcast) messages can be sent by every sensor nodes.

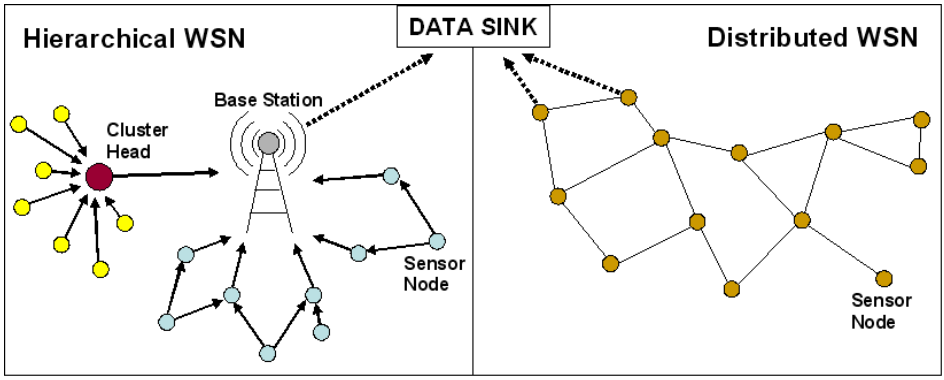


Figure 1. Network Models: Hierarchical and Distributed Wireless Sensor Networks.

2.2. Security Issues

Vulnerabilities

Adversaries want to manipulate (eavesdrop, modify, insert, delete and jam) application data. Wireless nature of communication, lack of infrastructure and uncontrolled environments improve capabilities of adversaries in a WSN. Stationary adversaries equipped with powerful computers and communication devices may access whole WSN from a remote location. They can gain mobility by using powerful laptops, batteries and antennas, and move around or within the WSN. They can plant their own sensor nodes, base stations or cluster heads; replace, compromise or physically damage existing ones. Wireless communication helps adversaries to perform variety of passive, active and stealth type of attacks [3]. In passive mode, adversaries silently listen to radio channels to capture data and security credentials (i.e., keys or cryptographic tools to derive them). In active attacks, adversaries may actively intercept key management traffic, capture, read or modify the contents of sensor nodes. They can use wireless devices with various capabilities to play man-in-the-middle or to hijack a session. They can insert, modify, replay, delete or jam the traffic [4]. In stealth attacks, an adversary manipulate a set of nodes, and the routing information trough them, to perform various attacks such as disconnecting network and hijacking traffic.

Base stations are usually trust centers and store information such as security credentials, sensor readings and routing tables. Thus, compromise of one or more of them can render the entire network useless. Similarly, cluster heads are the places where the sensor readings are aggregated. They are also accepted as the trusted components and sensor nodes rely on routing information from them.

Content of data flowing in a WSN can be classified into five categories: (i) sensor readings, (ii) mobile code, (iii) key management, (iv) routing information and (v) location information. Adversaries may improve their capabilities by accessing mobile codes, routing and location information. An adversary can insert a malicious mobile code which might spread to whole WSN, and use the location information to locate critical nodes to attack.

Requirements

Security requirements in WSNs are similar to those of ad-hoc networks [5], [6] due to similarities between MANET and WSN:

- *Availability.* It is ensuring that the service offered by whole WSN, by any part of it, or by a single sensor node is available whenever required.
- *Authentication.* It is ensuring that sensor nodes, cluster heads and base stations are authenticated before granting a limited resource or revealing information. Authentication ensures a receiver that data, mobile code or control data such as route updates, location information and key management messages originate from the correct source.
- *Authorization.* It ensures that only authorized nodes are involved in a specific activity (e.g., only a base station can broadcast a route update message).
- *Integrity and freshness.* It is ensuring that a message or an entity under consideration is not altered in transit and recent.
- *Confidentiality.* It provides privacy for wireless communication channels so that eavesdropping is prevented. Application data, mobile codes, control messages such as route updates, location information and key management traffic should be kept confidential.
- *Non-repudiation.* It prevents malicious nodes to hide their activities.

In addition to these, WSNs have two more requirements: (1)*Survivability* which means a WSN should provide a minimum level of service in the presence of power losses, failures or attacks, and (2) *Degradation of security services* which is the ability to change security level as resource availability changes.

Challenges

Wireless sensor networks inherit security problems from wireless networks and introduce far more challenges towards design of efficient key management solutions. The challenges described below should be kept in mind while evaluating the key management solutions presented in this chapter.

- *Wireless nature of communication.* Communication media is the air where everybody has access to. An adversary can perform variety of active and passive attacks on the key management traffic due to broadcast nature of the communication.
- *Resource limitation on sensor nodes.* Storage, processing, communication and battery life limitations on a sensor node prevent use of costly key management solutions. (i.e., Mica2Dot has a 7.3MHz Atmel ATMEGA128L low-power micro-controller which runs TinyOS, 128KB of read-only program memory, 4KB of RAM, a 433MHz Chipcon CC1000 radio which provides a 19.2 Kbps data rate with an approximate indoor range of 100 meters). Energy is the biggest concern because sensor nodes operate on batteries and it may not be possible to visit large number of nodes to replace their batteries. The biggest energy consuming operation for a sensor node is the communication. Thus, key management solutions with large communication overhead are not feasible.

- *Very large and dense WSN.* Most of the proposed sensor applications require hundreds to thousands of nodes densely deployed on a target application area.
- *Unknown and dynamic network topology.* There is no fixed infrastructure in a distributed WSN. Although there are resource rich members such as base stations in a hierarchical WSN, still large amount of sensor nodes are randomly scattered over a target area. Thus, post-deployment topology is unknown a priori to the deployment. Moreover, due to energy constraints and environmental or adversarial causes, sensor nodes may die and the new ones may be added. Hence, topology is dynamic and changes by time.
- *High risk of physical attacks.* Unattended sensors might be operating under adversarial conditions in uncontrolled hostile environments. Nodes can be physically damaged, captured and compromised. New malicious nodes can be planted. Thus, any critical information stored on a sensor node is subject to compromise.

Security of a WSN depends on existence of efficient key management solutions. Manual solutions which require visit to each sensor node are definitely inefficient. Use of a single network-wide shared-key is not a good idea either. Although the single key solution looks resource friendly, compromise of the key can jeopardize the security of the whole WSN. There are ongoing works [7,8,9] to customize the public key cryptography and the elliptic key cryptography for low-power devices, but such approaches are still considered as costly due to high communication, storage and processing requirements. Thus, sensor nodes have to adapt their environments, and establish a secure WSN by: (1) using pre-distributed keys or keying materials, (2) exchanging information with their immediate neighbors, or (3) exchanging information with computationally robust nodes.

Evaluation Metrics

In this chapter, a comparative survey and taxonomy of key management solutions are provided. It may not be always possible to give strict quantitative comparisons due to distinct assumptions made by these solutions; however, following metrics can be used to evaluate and compare the key management solutions.

- *Scalability.* Ability of a key management solution to handle an increase in the WSN size. More scalable solutions require less additional resources (i.e., storage, processing and communication) as the WSN size increases. A scalable key management solution must be flexible against substantial increase in the size of the network even after deployment.
- *Key Connectivity (probability of key-share).* Probability that a pair or a group of sensor nodes can generate or find a common secret key to secure their communication. A key management solution should provide sufficient key connectivity for a WSN to perform its intended functionality.
- *Resilience.* Resistance of the WSN against node capture. Keys which are stored on a sensor node or exchanged over radio links should not reveal any information about the security of any other links. Resilience is inversely related to the fraction of communication which is compromised when an

Problem	Approach	Mechanism	Papers
Dedicated Pair-wise	Probabilistic	Pre-distribution	[10]
	Deterministic	Pre-distribution	[11], [12], [13], [14], [15]
		Key Generation	[16], [17], [18], [11], [19], [12], [20], [21]
	Hybrid	Key Generation	[22], [23], [24]
Reusable Pair-wise	Probabilistic	Pre-distribution	[25], [10], [26], [27], [28], [29], [30], [31]
	Deterministic	Pre-distribution	[32], [33], [34]
	Hybrid	Pre-distribution	[32]
Group-wise	Deterministic	Key Generation	[17], [34]

Table 2. Classification of papers on dedicated pair-wise, reusable pair-wise and group-wise key management solutions in Distributed WSN.

adversary captures or replicates a sensor node. Higher resilience means lower number of compromised links.

- *Storage complexity.* Amount of memory units required to store security credentials.
- *Processing complexity.* Amount of processing cycles required by each sensor node to generate or find a common secret key.
- *Communication complexity.* Amount and size of messages exchanged between a pair or a group of sensor nodes to generate or find a common secret key.

In general, scalability, key connectivity, resilience and resource usage are conflicting requirements; therefore, trade-offs among these requirements must be carefully observed.

3. Key Management in Distributed Wireless Sensor Networks

In a distributed WSN, sensor nodes use dedicated pair-wise, reusable pair-wise and group-wise keys to secure their communication, or use keying materials to generate these keys. A part of key management solutions, called key pre-distribution schemes, assign a list of keys, called a key-chain, to each sensor node a priori to the deployment. Others, called key generation schemes, assign keying materials (i.e., PRF, HASH, key matrix, polynomial share and master key) to each node by using which a pair or a group of nodes can generate keys to secure their communication. Solutions to distribute keys and keying materials can be classified as: (i) probabilistic, (ii) deterministic, and (iii) hybrid. In probabilistic solutions, keys and keying materials are randomly selected from a pool. In deterministic solutions, deterministic processes are used: (1) to design the pool, and (2) to decide which keys and keying materials to assign to each sensor node so that the key connectivity is increased. Finally, hybrid solutions use probabilistic approaches along with deterministic algorithms to improve the scalability and key resilience. Table 2 classifies the papers which provide pair-wise and group-wise key management solutions in DWSN.

3.1. Dedicated Pair-wise Key Management

The trivial solution is to use a dedicated pair-wise key for each link in the WSN. Node S_i $(1 \leq i \leq N)$ stores a dedicated pair-wise key for each one of $N-1$ other sensor nodes in the WSN. Thus, each sensor S_i stores a key-chain $KC_i = \{K_{i,j} | i \neq j \text{ and } 1 \leq j \leq N\}$ of size $|KC_i| = N-1$ out of $N(N-1)/2$ keys. However, not all $N-1$ keys are required to be stored in nodes' key-chain to have a connected key graph. Although such an exhaustive solution creates unnecessary storage burden on a sensor node, this solution has perfect key resilience because compromise of a sensor node does not reveal any information about any other links in the WSN.

Probabilistic Key Pre-distribution

Random pair-wise key scheme [10] reduces the storage overhead by relaxing key connectivity, but it still provides perfect key resilience. It is based on Erdos and Renyi's work. Each sensor node stores a random set of Np dedicated pair-wise keys to achieve probability p that two nodes share a key. At key setup phase, each node ID is matched with Np other randomly selected node IDs with probability p. A dedicated pair-wise key is generated for each ID pairs, and is stored in both nodes' key-chain along with the ID of other party. Each sensor uses $2Np$ units of memory to store a key-chain with Np keys and Np key IDs. During shared-key discovery phase, each node broadcasts its ID so that neighboring nodes can tell if they share a pair-wise key. If a pair of neighboring nodes do not share a key, they can establish one through a secure path during key establishment phase.

Deterministic Key Pre-distribution

Matrix key distribution scheme [15] generates an $m \times m$ key matrix for a WSN of size $N = m^2$. During key pre-distribution phase, each sensor node is assigned a position (i, j), and receives all the keys in i^{th} column and all the keys in j^{th} row of the key matrix as the key-chain (total of $2m$ keys). During shared-key discovery phase, any pair of sensor nodes S_A and S_B exchange their position information (i, j) and (u, v) respectively to locate common dedicated pair-wise keys $K_{i,v}$ and $K_{u,j}$. Solution requires each sensor node to store $2\sqrt{N}$ keys and a position information. It has a low key resilience because an adversary needs to capture $\sqrt{N}$ nodes to recover whole key matrix.

Closest (location-based) pair-wise keys pre-distribution scheme [11] is an alternative to *Random pair-wise key scheme* [10]. It distributes keys and keying materials to sensor nodes based on the deployment knowledge to improve the key connectivity. Sensor nodes are deployed in a two dimensional area, and each sensor has an expected location. The idea is to have each sensor to share pair-wise keys with its c closest neighbors. In key setup phase, for each sensor node S_A, a unique master key K_A and c closest neighbors $S_{B_1}, \ldots, S_{B_c}$ are selected. For each pair (S_A, S_{B_i}), a pair-wise key $K_{A,B_i} = PRF(K_{B_i} | ID_A)$ is generated. Node S_A stores all pair-wise keys, whereas node S_{B_i} only stores the key K_{B_i} and the PRF. Thus, each sensor uses $2c+1$ units of memory to store $c+1$ keys and c key IDs. Solution preserves a good key connectivity if deployment error is low. A sensor

either searches for a pair-wise key or generates it with a PRF function. Similar to *Random pair-wise key scheme* [10], this solution has perfect resilience.

ID based one-way function scheme (IOS) [12] distributes dedicated pair-wise keys according to connected r-regular graphs which have edge decomposition into star-like subgraphs. Each sensor node S_A receives a secret key K_A. Moreover, if the node S_A is in the star-like graph centered around a node S_B, then S_A also receives $Hash(K_B|ID_A)$. In this case, during shared-key discovery phase, S_A sends its ID to S_B and S_B can generate the key $Hash(K_B|ID_A)$. In this solution, any pair of neighboring nodes either share a key or can generate one through two-hop secure paths in key establishment phase. In an r-regular graph, each sensor node can be the center of one and leaf of $r/2$ star-like subgraphs. Thus, each sensor uses $r+1$ units of memory to store $r/2+1$ keys and $r/2$ key IDs. Solution has perfect key resilience. *Multiple IOS* [12] is proposed to improve scalability of *ID based one-way function scheme (IOS)*. Every node in the graph corresponds to ℓ nodes $S_A = S_{A_1}, \ldots, S_{A_\ell}$. Thus, sensor nodes S_{A_i} store a common master key K_A and a secret $Hash(K_B|ID_{A_i})$. Every node S_{B_j} in the class of node S_B, can use common master key K_B to generate the secret $Hash(K_B|ID_{A_i})$ for node S_{A_i}. *Multiple IOS* decreases memory usage by a factor of ℓ. It relaxes the key resilience, because compromise of a master key means compromise of the links of ℓ sensor nodes.

Expander graph based key distribution scheme [13] is a dedicated pair-wise key pre-distribution scheme based on ramanujan expander graphs which are best known explicit expanders. An expander is a regular multi-graph in which any subset of vertices has large number of neighbors (i.e. expands to a larger set). Solution takes advantages of the expansion property to provide better key connectivity than *random pair-wise key scheme* [10]. Basically, a ramanujan expander $X^{s,t}$, which is an $(s+1)$-regular graph with $N = t+1$ nodes, is generated. Two sensor nodes are assigned a unique pair-wise key if the corresponding vertices have an edge in $X^{s,t}$. It is possible to generate any key-chain size $s+1$ for any network size $t+1$ required that s and t are primes congruent to 1 (mod 4).

Peer intermediaries for key establishment (PIKE) [14] is a dedicated pair-wise key pre-distribution scheme where one or more sensor nodes are used as trusted intermediaries. In key setup phase, each node is associated with a point (x, y) in a $\sqrt{N} \times \sqrt{N}$ two dimensional grid for a WSN of size N. A node at point (x, y) shares a dedicated pair-wise key for each node at point (i, y) and (x, j) for $0 \leq i, j \leq \sqrt{N} - 1$. If a pair of nodes S_A and S_B do not share a key, solution guarantees that there are exactly two nodes with which they both share a key. Geographically closest one of these two nodes is used as a trusted intermediary to establish a pair-wise key in between S_A and S_B. Each sensor uses $4(\sqrt{N} - 1)$ units of memory to store $2(\sqrt{N} - 1)$ pair-wise keys and $2(\sqrt{N} - 1)$ key IDs.

Deterministic Key Generation

Basic idea is that each sensor node is pre-distributed a small amount of private and public information as keying materials by using which any pair of sensor nodes can generate a dedicated pair-wise key. A trivial solution, *broadcast session key negotiation protocol (BROSK)* [18], is based on a single master key K_m which is pre-distributed to all sensor nodes. During key establishment phase, a pair of

sensor nodes (S_i, S_j) exchanges random nonce values (RN_i, RN_j). They use master key K_m to generate the dedicated pair-wise key $K_{i,j} = PRF(K_m|RN_i|RN_j)$. Each sensor stores the same master key, and it is possible to derive all pair-wise keys once the master key is compromised; therefore the scheme has very low key resilience.

Lightweight key management system [19] proposes a solution with slightly better resilience where more than one master keys are employed. It assumes a WSN where groups of sensor nodes are deployed in successive generations of size θ. Each sensor node stores an authentication key bk_1 and a key generation key bk_2. During key establishment phase, if two sensor nodes S_A and S_B are from the same generation, they authenticate each other by using the authentication key bk_1, exchange random nonce values RN_A and RN_B, and generate the dedicated pair-wise key $K_{A,B} = PRF(bk_2|RN_A|RN_B)$. A sensor node S_A, of an old generation i, stores a random nonce RN_A and a secret $S_{A,j}$ for each new generation j. Secret $S_{A,j}$ is used to authenticate sensor nodes from new generation j: a node S_B of new generation j can generate the secret $S_{A,j} = PRF(gk_j|RN_A)$ where secret gk_j is only known to nodes of generation j. Once authenticated, both parties use $S_{A,j}$ as the key generation key to generate the dedicated pair-wise key $K_{A,B}$. If there are g of such generations, each sensor needs at most $2g+1$ units of memory initially to store $g+2$ keys and $g-1$ random nonce. Resilience of the scheme is still low because an adversary only needs to compromise the secrets bk_1, bk_2 and gk_j of generation j to compromise all the links of nodes in generation j. Furthermore, an adversary may log the messages flowing in the network to process them when the required credentials are compromised completely.

Blom's scheme [16] uses a public $(\lambda+1) \times N$ matrix G and a private $(\lambda+1) \times (\lambda+1)$ symmetric matrix D which are both generated over GF(q). All possible link keys in a network of size N are represented as an $N \times N$ symmetric key matrix $K = (D \times G)^T \times G$. Solution is λ-secure, meaning that keys are secure if no more than λ nodes are compromised. In key setup phase, sensor node S_i stores $column_i$ of size $\lambda+1$ from matrix G as public information, and row_i of size $\lambda+1$ from matrix $(D.G)^T$ as private information. During key establishment phase, a pair of sensor nodes (S_i, S_j), first exchange their public information then generate dedicated pair-wise key $K_{ij} = K_{ji}$ as described in Figure 2. The scheme requires costly multiplication of two vectors of size $\lambda+1$ where the elements are as large as the corresponding cryptographic key size. Each sensor node broadcasts one message, and receives one message from each node within its radio range where messages carry a vector of size $\lambda+1$.

Scalability of *Blom's scheme* is improved in *multiple space Blom's scheme (MBS)* [12]. This scheme assumes that underlying physical network graph is a complete bipartite graph, and divides nodes into two sets U and V to form bipartite key graph in which case not every pair of nodes share a key and private matrix D is not necessarily symmetric. In key setup phase, S_u and S_v store public information $column_u$ and $column_v$ from public matrix G respectively. Private information $column_u^T \times D$ is assigned to each node $S_u \in U$, and $D \times column_v$ is assigned to each node $S_v \in V$. During shared-key discovery phase, each pair of neighboring nodes S_u and S_v exchange their public information, and They gen-

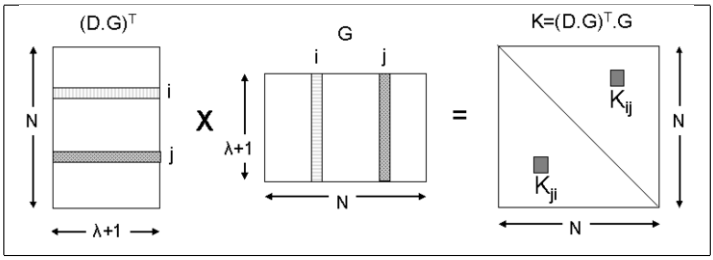


Figure 2. Blom's scheme. Sensor node S_i stores $column_i$ from matrix G as public information, and row_i from matrix $(D.G)^T$ as private information. Nodes S_i and S_j exchange their public column vectors and generate $K_{ij} = row_i \times column_j$ and $K_{ji} = row_j \times column_i$ respectively where $K_{ij} = K_{ji}$

erate the dedicated pair-wise key $K_{u,v} = K_{v,u} = column_u^T \times D \times column_v$ in key establishment phase.

Larger networks are supported in *deterministic multiple space Blom's scheme (DMBS)* [12] where ℓ copies of strongly regular graph R (regular of degree r) are used. Each vertex of R can be considered as a class of ℓ nodes (i.e., $S_u = S_{u_1}, \ldots, S_{u_\ell}$). An arbitrary direction is assigned to each edge in R, and each edge e has a random private matrix D_e which is not necessarily symmetric. In key setup phase, each sensor node S_{u_i} receives its public column information $column_u$ from public matrix G. For a directed edge $(S_{u_i}, S_{v_j}) \in R$, source node S_{u_i} receives private information $column_u^T \times D_{uv}$, and destination node S_{v_j} receives private information $D_{uv} \times column_v$. Thus, each node stores r vectors of size $(\lambda + 1)$. During shared-key discovery phase, neighboring nodes S_{u_i} and S_{v_j} exchange their public information. They generate the dedicated pair-wise key $K_{u_i,v_j} = K_{v_j,u_i} = column_u^T \times D_{uv} \times column_v$ in key establishment phase. *DMBS* increases scalability with the cost of decreased key resilience because capture of one sensor node compromises credentials of $\ell - 1$ others.

Polynomial based key pre-distribution scheme [17] distributes a polynomial share (a partially evaluated polynomial) to each sensor node by using which every pair of nodes can generate a dedicated pair-wise key. Symmetric polynomial $P(x, y)$ (i.e., $P(x, y) = P(y, x)$) of degree λ is used. $\lambda + 1$ coefficients of the polynomial come from $GF(q)$ for sufficiently large prime q. In key setup phase, sensor node S_i receives its polynomial share $f_i(y) = P(i, y)$. During key establishment phase, S_i (a.k.a. S_j) can generate the dedicated pair-wise key $K_{i,j} = P(i, j)$ for node S_j (a.k.a. S_i) by evaluating its polynomial share $f_i(y)$ (a.k.a. $f_j(y)$) at point j (a.k.a. i). Every pair of sensor nodes can establish a key. The solution is λ-secure meaning that coalition of less than $\lambda + 1$ sensor nodes knows nothing about keys of others.

Location-based pair-wise keys scheme using bivariate polynomials [11] considers deployment knowledge during key setup phase. Deployment area is divided into R rows and C columns, total of $R \times C$ cells. The scheme is based on *polynomial based key pre-distribution scheme* [17]. For each cell at c^{th} column and r^{th} row, a unique polynomial $f_{c,r}(x, y)$ is generated. Each sensor node stores polynomial share of its home cell and four immediate neighboring cells, total of five

polynomials. During key establishment phase, two sensor nodes simply exchange their cell coordinates to agree on a polynomial share.

Deterministic pair-wise key distribution scheme [20] uses polynomial pool idea along with symmetric design technique as in *combinatorial design based pair-wise key pre-distribution scheme* [32] to distribute a chain of polynomial-sets to each sensor node so that any pair of sensor nodes can find a common polynomial-set to generate a dedicated pair-wise key. Each polynomial-set contains polynomial shares due to *polynomial based key pre-distribution scheme* [17] where the number of polynomial shares is determined by the size of the dedicated pair-wise key in bits. *Hypercube multivariate scheme (HMS)* [21] uses an n-dimensional hypercube in the multidimensional space where a multivariate polynomial is assigned to each point on the hypercube. Each sensor node is assigned to a unique coordinate on the hypercube and receives n polynomial shares. Any pair of sensor nodes which are at a Hamming distance of one can establish a dedicated pair-wise key.

Hybrid Key Generation

In hybrid solutions, keys and keying materials are distributed to sensor nodes based on both probabilistic and deterministic techniques. Probabilistic part generally helps to improve the scalability and key resilience while deterministic part improves the key connectivity.

Multiple space key pre-distribution scheme [22] improves the resilience of *Blom's scheme* [16]. It uses a public matrix G and a set of ω private symmetric matrices D. These matrices form ω spaces (D_i, G) for $i = 1, \ldots, \omega$ (i.e., i^{th} space uses private symmetric matrix D_i and public matrix G). In key setup phase, for each sensor node, a set of τ spaces are randomly selected among these ω spaces. Thus, each sensor node stores τ private row information and a public column information. In shared-key discovery phase, each pair of nodes agree on a common space for which they have to exchange an extra message which includes τ space IDs. Once agreed upon a space, during key establishment phase, a dedicated pair-wise key is generated as in *Blom's scheme* [16]. It is possible that a pair of nodes do not share a common space, in that case they have to establish a pair-wise key through one or more secure paths connecting them.

Polynomial pool-based key pre-distribution scheme [23] considers the fact that not all pairs of sensor nodes have to establish a key. It combines *polynomial based key pre-distribution scheme* [17] with the pool idea in [25,10] to improve the resilience and scalability. In key setup phase, a pool F of λ-degree symmetric polynomials (i.e., $P(x, y) = P(y, x)$) over finite field GF(q) is generated. For each sensor node S_i, a subset $F_i \subseteq F$ of polynomials is picked and polynomial shares $f_i(y) = P(x = i, y)$ for each $P(x, y) \in F_i$ are generated. Each sensor node also stores the list of sensor ID's with which it shares a polynomial. During key establishment phase, if a pair of neighboring nodes S_i and S_j have a polynomial share in common, they can generate a dedicated pair-wise key $K_{i,j} = f_i(j) = f_j(i)$.

In *grid-based key pre-distribution scheme* [23], for a network size of N, $m \times m$ grid (for $m = \lceil \sqrt{N} \rceil$) with a set of $2 \times m$ column and row symmetric polynomials $\{f_i^r(x, y), f_j^c(x, y)\}$ $(i, j = 0, \ldots, m - 1)$ are generated. Each row i of the grid is associated with a polynomial $f_i^r(x, y)$ and each column j is associated with a

polynomial $f_j^c(x, y)$. Each sensor is randomly assigned to a coordinate (i, j) on the grid, and receives polynomials $\{f_i^r(x = j, y), f_j^c(x = i, y), \}$ along with $ID = (i, j)$. In shared-key discovery phase, each pair of neighboring nodes exchange their IDs. In key establishment phase, if their row addresses overlap, (i.e., (i,j) and (i,j')), then the dedicated pair-wise key is generated as $f_i^r(j, j')$, or if their column addresses overlap (i.e., (i,j) and (i',j)), then the dedicated pair-wise key is generated as $f_j^c(i, i')$.

Grid-group deployment scheme [24] divides deployment area into cells over which groups of sensor nodes are uniformly distributed. *Polynomial pool-based key pre-distribution* [23] and *multiple space key pre-distribution* [22] schemes are used to establish dedicated pair-wise keys within each cell. Moreover, each sensor node selects exactly one sensor node from each neighboring cell, and shares a dedicated pair-wise key with it for the inter-cell communication.

3.2. Reusable Pair-wise Key Management

The trivial solution is to deploy a single master key to all sensors. It has very low resilience since an adversary can capture a node and compromise the master key. The solutions in this category mostly use probabilistic, deterministic or hybrid approaches to pre-distribute a key-chain to each sensor node. The keys are named as reusable pair-wise keys because it is possible for a key to appear in more than two key-chains meaning that a key may be used to secure more than one links in the WSN.

Probabilistic Key Pre-distribution

The original solution is *basic probabilistic key pre-distribution scheme* [25] by Eschenauer *et al.* It relies on probabilistic key sharing among the nodes of a random graph. In key setup phase, a large key-pool of KP reusable pair-wise keys and their identities are generated. For each sensor node, k keys are randomly drawn from the key-pool KP without replacement. These k keys and their identities form the key-chain for the sensor node. Thus, probability of key share among two sensor nodes becomes $p = \frac{((KP-k)!)^2}{((KP-2k)!KP!)}$. In shared-key discovery phase, two neighboring sensor nodes exchange and compare the list of identities of keys in their key-chains. It is possible to protect the key identities by using a method similar to *Merkle Puzzle* [35], but this method substantially increases processing and communication overhead. Common keys are used to secure the link. If a pair of neighboring nodes do not share a key, they can establish one through one or more secure paths connecting them during key establishment phase. Unlike schemes that uses dedicated pair-wise keys, it may be possible in this solution that same key is used to secure more than one links. Probability that a link is compromised when a sensor node is captured is k/KP. Resilience of the solution can be improved by using larger key pools at a cost of decreased probability of key share.

Cluster key grouping scheme [29] builds upon basic probabilistic scheme and divides key-chains into C clusters where each cluster has a *start key ID*. Remaining key IDs within a cluster are implicitly known from its *start key ID*. Thus, only start key IDs for clusters are exchanged during shared-key discovery phase

which means that messages carry key ID list of size c instead of k. *Pair-wise key establishment protocol* [27] further reduces communication overhead of shared key discovery phase. Each sensor node gets a unique ID. Key IDs for keys in the key-chain of a sensor node S_A are generated by $PRF(ID_A)$. Thus, sensor nodes only exchange their IDs during shared-key discovery phase. *Transmission range adjustment scheme* [30] proposes sensor nodes to increase their transmission ranges during shared-key discovery phase. Nodes return to their original optimal transmission range once common keys are discovered. Motivation of this scheme is to decrease communication burden in key establishment phase.

Q-composite random key pre-distribution scheme [10] is an extension to basic probabilistic scheme where a pair of neighboring sensor nodes should have more than q common keys to secure their link. The key $K_{A,B}$ between sensor nodes S_A and S_B is set as the hash of all common keys $K_{A,B} = Hash(K_1||K_2||K_3||\ldots||K_q)$. This scheme improves the resilience because probability that a link is compromised, when a sensor node is captured, decreases from k/KP to $\binom{k}{q}/\binom{KP}{q}$. But, this improvement comes at a cost of decreased key sharing probability.

Key pre-distribution by using deployment knowledge scheme [28] is based on the idea that distant sensor nodes do not need to have common keys in their key-chains. Sensor nodes are divided into $t \times n$ groups $G_{i,j}$, and each group is deployed at a resident point (x_i, y_j) for $1 \leq i \leq t$ and $1 \leq j \leq n$ where the points are arranged as a two dimensional grid. The resident point of a node $m \in G_{i,j}$ follows the probability distribution function $f_m^{i,j}(x, y|m \in G_{i,j}) = f(x - x_i, y - y_j)$ where $f(x, y)$ is a two dimensional Gaussian distribution. In *key setup* phase, a key-pool KP is divided into $t \times n$ subsets $KP_{i,j}$ where $|KP_{i,j}| = \omega_{i,j}$. $KP_{i,j}$ is used as the key-pool for the nodes in group $G_{i,j}$ located at coordinate (x_i, y_j). Given $\omega_{i,j}$ and overlapping factors α and β, a key-pool KP is divided into subsets $KP_{i,j}$ where (i) two horizontally and vertically neighboring groups have $\alpha \times \omega_{i,j}$ keys in common, (ii) two diagonally neighboring groups have $\beta \times \omega_{i,j}$ keys in common, and (iii) non-neighboring groups do not share a key. Finally, *basic probabilistic key pre-distribution scheme* is applied within each group $G_{i,j}$ with the key-pool $KP_{i,j}$. Problem in this scheme is the difficulty to decide on parameters $\omega_{i,j}$, α and β to provide a good key connectivity.

Random key pre-distribution scheme using probability density function [31] is another solution which uses deployment knowledge. The solution assumes that sensor nodes are deployed through air; thus, their resident points may vary widely due to the air resistance. This unevenness in deployment is modeled by using *two dimensional normal distribution*. Application area is considered as a two dimensional grid of size $\sqrt{|KP|} \times \sqrt{|KP|}$. $|KP|$ keys are logically mapped to $|KP|$ locations on the grid. Each sensor node receives a set of keys according to its expected residence locations on the grid which are estimated by using the node deployment Probability Density Function (PDF). For a sensor node S, a resident point (x, y) is selected according to the PDF. A point (w, z) within the circle of radius r centered at (x, y) is selected. Node S receives the key assigned to the grid location which includes the point (w, z). Another resident point (x', y') is selected and the process is repeated until node S receives K distinct keys.

Deterministic Key Pre-distribution

Key sharing probability can be increased by *designing* the key-chains. *Combinatorial design based key pre-distribution scheme* [32] is based on the block design techniques, namely *symmetric design* and *generalized quadrangles*, from combinatorial design theory. *Symmetric design* (or *symmetric BIBD*) with parameters $(n^2+n+1, n+1, 1)$ separates a set of objects into blocks where every pair of blocks have exactly one object in common. Each object is mapped to a reusable pair-wise key and each block to a key-chain. Thus, *symmetric design based key pre-distribution* scheme generates $N = n^2+n+1$ key-chains, and uses a key-pool of n^2+n+1 reusable pair-wise keys. In key setup phase, n^2+n+1 key-chains are generated where a key-chain contains $n+1$ keys and every pair of key-chains have exactly one key in common. Every key appears in exactly $n+1$ key-chains meaning that the same key might be used to secure more than one links. For example, $(7, 3, 1)$ design uses a key-pool of 7 reusable pair-wise keys and generates 7 key-chains where each key-chain has 3 keys and every pair of key-chains have one key in common. Key-chains for this design are $\{K_1, K_2, K_3\}$, $\{K_1, K_4, K_5\}$, $\{K_1, K_6, K_7\}$, $\{K_2, K_4, K_6\}$, $\{K_2, K_5, K_7\}$, $\{K_3, K_4, K_7\}$ and $\{K_3, K_5, K_6\}$. During shared-key discovery phase, every pair of nodes can find exactly one key in common. Thus, probability of key sharing among a pair of sensor nodes is 1. Probability that a link is compromised when a sensor node is captured is approximately $1/n$. Disadvantage of this solution is that, parameter n has to be a prime power; therefore, not all network sizes can be supported for a fixed key-chain size.

More scalable solutions can be provided by using *generalized quadrangles (GQ)* design with the property that not all pairs of neighboring nodes need to share a key directly. In *generalized quadrangles based key pre-distribution* scheme [32], a pair of key-chains may not have a key in common, but GQ guarantees that there are other key-chains which share exactly one key with both. Proposed GQ designs, $GQ(n, n)$, $GQ(n, n^2)$ and $GQ(n^2, n^3)$, support network sizes of orders $O(n^3)$, $O(n^5)$ and $O(n^4)$ in key-chain size respectively [32]. Although GQ is more scalable than symmetric design, parameter n still needs to be a prime power. Very similar approaches based on combinatorial design theory are proposed in [33].

Hybrid Key Pre-distribution

Hybrid symmetric and *hybrid generalized quadrangle (GQ)* designs [32] use probabilistic approaches along with the deterministic designs to support arbitrary network sizes. An *hybrid design* first generates core symmetric or GQ design then randomly selects remaining key-chains from subsets of the complementary symmetric or GQ design blocks as summarized in Figure 3. Given the key-pool KP and $M < N$ key-chains KC_i generated due to the symmetric or the GQ design, complementary design has M key sets $\overline{KC_i} = KP \setminus KC_i$. Remaining $N - M$ key-chains are randomly selected among the $|KC|$-subsets of the complementary design key sets $\overline{KC_i}$. *Hybrid design* improves the scalability and the resilience, but sacrifices the key sharing probability of the core symmetric or GQ design.

Key Update Mechanisms

There are several key update proposals to refresh pair-wise keys for improved key resilience. After the sensor deployment, each pair of neighboring sensor nodes

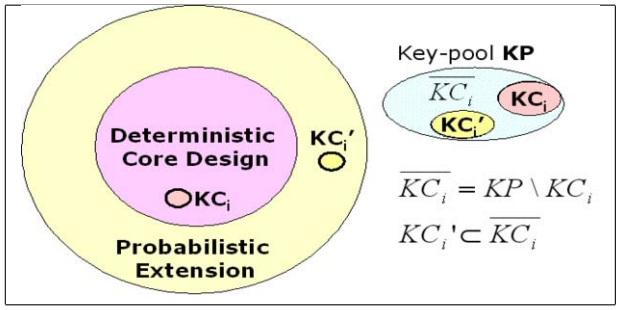


Figure 3. Hybrid design with a symmetric (or GQ) core of size M and a probabilistic extension of size $N-M$. Key-chains KC_i' of probabilistic extension are randomly selected among k-subsets of $\overline{KC_i} = KP \backslash KC_i$ which are complements of core symmetric (or GQ) design key-chains KC_i.

establish one or more dedicated or reusable pair-wise keys during shared-key discovery and key establishment phases. A reusable pair-wise key might be known to more than one pairs of sensor nodes, and it is subject to compromise if one of these nodes is captured. If a pair of neighboring nodes establish a dedicated pair-wise key, any malicious nodes involved in key establishment phase may leak the key or information that can be used to reproduce it. It is also possible that information about the pair-wise keys are compromised during the WSN operations. Thus, keys should be updated when: (i) key establishment phase is completed and a pair-wise key is established between each pair of neighboring sensor nodes, (ii) life time for the keys expires, (ii) the number of sensor nodes added or deleted goes beyond a threshold value, (iii) nodes are compromised, and (iv) malicious nodes are detected.

In *probabilistic key pre-distribution scheme* [25], special sensor nodes with large communication range, called controller nodes, can revoke the compromised or expired key-chains. Each sensor node S_i shares a dedicated pair-wise key K_i^{cj} with j^{th} controller node. When a key-chain needs to be revoked, the controller node broadcasts a list of key IDs signed with the signature key K_e. Each sensor node S_i receives K_e encrypted with K_i^{cj} from the controller node. On receiving the signature key, each sensor node can authenticate the list and revoke matching keys in its key-chain. If revoked keys are used in securing wireless links, then shared-key discovery and key establishment phases are restarted for those links.

In *multi-path key reinforcement scheme* [10] between sensor nodes S_A and S_B, node S_A generates j random key updates rk_i and sends them through j disjoint secure paths. S_B can generate a new key $K_{A,B}^r = K_{A,B} \oplus rk_1 \oplus \ldots \oplus rk_j$ upon receiving all key updates. This approach requires nodes S_A and S_B to send and receive j more messages each of which carries a key update. Moreover, each node on the j disjoint path has to send and receive an extra message. *Recursive key establishment protocol (RKEP)* [36] is an extension to *multi-path key reinforcement scheme*. It provides a recursive key establishment protocol which does not require a pair of nodes to find j node-disjoint paths explicitly.

Pair-wise key establishment protocol [27] uses *threshold secret sharing* for key update. S_A generates a secret key $K_{A,B}^r$, $j-1$ random shares $sk_1, \ldots, sk_{j-1}$, and

$sk_j = K^r_{A,B} \oplus sk_1 \oplus \ldots \oplus sk_{j-1}$. S_A sends the shares through j disjoint secure paths. S_B can recover $K^r_{A,B}$ upon receiving all the shares.

In *Co-operative pair-wise key establishment protocol* [26], S_A first chooses a set $C = \{c_1, c_2, \ldots, c_m\}$ of co-operative nodes for the key update between nodes S_A and S_B. A co-operative node provides a hash $HMAC(K_{c_1,B}, ID_A)$. Updated key is then $K^r_{A,B} = K_{A,B} \oplus \left(\bigoplus_{c \in C} HMAC(K_{c,B}, ID_A)\right)$ where $K_{A,B}$ and $K_{c,B}$ are the pair-wise keys which are established during shared-key discovery or key establishment phase. Node S_A shares set C with node S_B; therefore, S_B can generate the same key. This approach requires nodes S_A and S_B to send and receive c more messages. Moreover, each cooperative nodes has to send and receive two extra messages. In addition to increased communication cost, each cooperative node has to execute $HMAC$ function twice for S_A and S_B. The key update solutions in general increase processing and communication overhead, but provide good resilience in the sense that a compromised sensor node does not directly affect security of any other links in the WSN. But, it may be possible for an adversary to recover the key updates from the recorded messages when initial pair-wise keys are compromised.

3.3. Group-wise Key Management

Straightforward approach is to use existing pair-wise keys to establish group-wise keys. For example, *Lightweight key management system* [19] proposes to distribute group-wise keys through the links which are secured with pair-wise keys. Yet another approach is to pre-distribute polynomial shares to sensor nodes by using which group members can generate a common group key. *Polynomial based key pre-distribution scheme* [17] proposes two models. The first model is a non-interactive model where users compute a common key without any interaction. A random symmetric polynomial $P(x_1, \ldots, x_t)$ in t variables of degree λ is selected initially where the coefficients come from $GF(q)$ for prime q which is large enough to accommodate the key length of the underlying cryptosystem. Each user S_i receives share $P_i(x_2, \ldots, x_t) = P(i, x_2, \ldots, x_t)$. Users $S_{j_1}, \ldots, S_{j_t}$ can generate the group-wise key $K_{j_1,\ldots,j_t}$ by evaluating their polynomial shares, i.e. each user S_{j_i} can evaluate $P_{j_i}(j_1, \ldots, j_{i-1}, j_{i+1}, \ldots, j_t)$ and obtain the group-wise key $K_{j_1,\ldots,j_t}$ independently. In the second, interactive model, polynomial $P(x, y)$ of degree $(\lambda + t - 2)$ is selected initially. Each user S_i receives the share $P_i(y) = P(i, y)$. Users $S_{j_1}, \ldots, S_{j_t}$ can calculate the conference key $K_{j_1,\ldots,j_t}$ as follows: (i) S_{j_t} selects a random key K, (ii) S_{j_t} calculates $K_{j_t,j_\ell} = P_{j_t}(j_\ell) = P(j_t, j_\ell)$ for each $\ell = 1, \ldots, t-1$, (iii) S_{j_t} sends $\chi_\ell = K_{j_t,j_\ell} \oplus K$ to each S_{j_ℓ} for $(\ell = 1, \ldots, t-1)$, and (iv) each S_{j_ℓ} generates $K_{j_\ell,j_t} = P_{j_\ell}(j_t)$, and derives the secret $K = \chi_\ell \oplus K_{j_\ell,j_t}$. Sensor node S_{j_t} performs $t-1$ polynomial evaluations, and sends $t-1$ messages which carry a single χ value to establish a group-wise key.

Hashed random preloaded subsets (HARPS) scheme [34] pre-distributes a key-chain, a one-way hash function $HASH$ and a public-key-generation function $F()$ to each sensor node so that a group of sensor nodes can generate a pair-wise or a group-wise key by simply exchanging their node IDs. During key pre-distribution phase, P root keys $M_1, \ldots, M_P$ are generated. Each root key can be used along with the HASH to generate L derived keys $K^j_i = HASH^j(M_i)$ (for $1 \leq i \leq P$

Problem	Approach	Mechanism	Papers
Pair-wise	Deterministic	Key Generation	[37], [38,39], [40], [41], [42]
Group-wise	Deterministic	Key Generation	[43], [44], [45], [46], [41], [47], [48], [49]
Network-wise	Deterministic	Key Generation	[50], [51], [52], [53], [54,55], [38,39] [56], [41]

Table 3. Classification of solutions on pair-wise, group-wise and network-wise key management solutions in Hierarchical WSN.

and $1 \leq j \leq L$). Thus, key-pool has $P \times L$ keys. Each sensor node S_A receives a key-chain of $\{K_{A_1}^{a_1}, \ldots, K_{A_k}^{a_k}\}$ where $\{(A_1, a_1), \ldots, (A_k, a_k)\} = F(S_A)$. After the deployment, a group of sensor nodes exchange their node IDs. Each sensor can use $F()$ and the received node IDs to generate key IDs stored in the key-chains of others. Once one or more keys with the same subscript (generated from the same root key M_i) are located (i.e., nodes S_A, S_B and S_C find out that keys $K_{A_i}^{a_i}$, $K_{B_j}^{b_j}$ and $K_{C_l}^{c_l}$ have the same subscripts $A_i = B_j = C_l$), each node can independently use function HASH to generate the key with highest superscript (a derived key) (i.e., let $a_i > b_j$ $a_i > c_l$, node S_B can obtain $K_{A_i}^{a_i} = K_{B_j}^{a_i} = HASH^{a_i - b_j}(K_{B_j}^{b_j})$ and node S_C can obtain $K_{A_i}^{a_i} = K_{C_l}^{a_i} = HASH^{a_i - c_l}(K_{C_l}^{c_l})$). If $P = k$ then any pair or group of nodes can generate a pair-wise or group-wise key.

4. Key Management in Hierarchical Wireless Sensor Networks

A Hierarchical WSN (HWSN) includes one or more computationally robust base stations. Sensor nodes are deployed in one or two-hop neighborhood around base stations or resource rich sensor nodes (called cluster heads) as illustrated in Figure 1. Base stations are usually assumed to be trusted and used as the key distribution centers. In a HWSN, pair-wise, group-wise and network-wise keys are required to secure unicast, multicast and broadcast types of communications among sensor nodes, cluster heads and base stations. Table 3 classifies the papers which provide solutions to pair-wise, group-wise and network-wise key management problems in HWSN. Based on this classification, the solutions are described in Sections 4.1, 4.2 and 4.3.

4.1. Pair-wise Key Management

In a Hierarchical WSN, base station to sensor node, or sensor node to base station unicast communications are secured by using dedicated pair-wise keys. A straightforward approach is to pre-distribute a dedicated pair-wise key to each sensor node so that each base station shares a dedicated pair-wise key with each sensor node deployed within its close vicinity. Very similar solutions are proposed in *perimeter protection scenario* [37], *base station authentication protocols* [38,39], and *localized encryption and authentication protocol (LEAP)* [41]. A base station can intermediate establishment of a dedicated pair-wise key between any pair of surrounding sensor nodes since it shares a dedicated pair-wise key with each of them. A very similar approach is used in *ESA* [40].

Localized encryption and authentication protocol (LEAP) [41] proposes to use a network-wise master key to establish a dedicated pair-wise key between each pair of neighboring sensor nodes. Each sensor node S_u is deployed with a network-wide master key K_I, a one-way hash function $HASH()$ and a unique ID. Node S_u generates the secret key $K_u = HASH_{K_I}(ID_u)$. In *shared-key discovery* phase between neighboring nodes S_u and S_v, node S_u broadcasts (ID_u, RN_u) and node S_v responds with $(ID_v, MAC_{K_v}(RN_u|ID_v))$. During key establishment phase, node S_u generates the secret key $K_v = HASH_{K_I}(ID_v)$, and both nodes S_u and S_v generate the dedicated pair-wise key $K_{u,v} = HASH_{K_v}(ID_u)$. A multi-hop dedicated pair-wise key may be required to secure a communication between a sensor node S_u and its cluster head S_c which are connected through other sensor nodes. In LEAP, node S_u generates a secret $K_{u,c}$ and divides it into shares $K_{u,c} = sk_1 \oplus sk_2 \oplus \ldots sk_m$. Node S_u then finds m intermediate nodes S_{v_i} $(1 \leq i \leq m)$ and sends each share through a separate intermediate node. Basically, node S_u sends the message $ENC_{K_{u,v_i}}(sk_i), HASH_{sk_i}(0)$ to node S_{v_i}, and S_{v_i} sends the message $ENC_{K_{v_i,c}}(sk_i), HASH_{sk_i}(0)$ to the cluster head S_c. Solution has high communication overhead because S_u sends m messages through m intermediate sensor nodes to increase the key resilience. However, security of the system depends on the master key K_I which can be compromised when a sensor node is captured. It is possible to compromise all pair-wise keys generated by LEAP algorithm once the master key K_I is compromised.

Low-energy key management protocol [42] assumes a HWSN with a trusted base station, cluster heads and sensor nodes deployed around cluster heads. In key pre-distribution phase, each sensor node S_i is assigned to a cluster head CH_i and shares two dedicated pair-wise keys with the base station and CH_i. Each cluster head shares pair-wise keys with the base station and other cluster heads. After the deployment, if a sensor node S_i is deployed around a cluster head CH_j with which it does not share a key, CH_j contacts CH_i to receive the pair-wise key for sensor node S_i. If the deployment error is too high meaning that all sensor nodes deployed into wrong clusters, then communication overhead of the scheme is very high. Compromise of a cluster head compromises all the communication with the cluster.

4.2. Group-wise Key Management

A set of solutions propose to use costly asymmetric cryptography based key management solution. Burmester-Desmedt [43] and IKA2 [44] use a Diffie-Hellman based group key transport protocol. These two algorithms are improved by ID-STAR [46]. ID-STAR uses *Identity based cryptography* [57,58] where sensor nodes' public keys can be derived from their identities.

In a HWSN where each base station shares a dedicated pair-wise key with each sensor node deployed within its close vicinity, the base station can intermediate group-wise key establishment. *Localized encryption and authentication protocol (LEAP)* [41] proposes a group-wise key generation scheme which follows LEAP pair-wise key establishment phase. Assume that sensor node S_u wants to establish a group-wise key with all its neighbors $S_{v_1}, S_{v_2}, \ldots, S_{v_m}$. Node S_u generates a unique group-wise key K_u^g and sends $ENC_{K_{u,v_i}}(K_u^g)$ to each of its neighbors S_{v_i}.

Security of the scheme depends on security of the pair-wise keys which in turn has very low resilience.

Hierarchical key generation and distribution protocol [47] proposes a multi-level security solution because sensor data varies in importance level. Each sensor node is assigned to an access class. Access classes are ordered in a hierarchy where root is the highest access class and leaves are the lowest. Protocol assumes a HWSN with a base station and clusters of sensor nodes where each cluster is assigned to an access class. A random number D_0 is assigned to the root access class, one-way hash of it $D_0^i = HASH(2^i D_0)$ is assigned to its i^{th} children. This process recursively continues and assigns a number to each access classes. Each node representing an access class in the hierarchy receives one number which is called as its residual set. Note that, given the residual set of a higher class, it is possible to generate the residual set of a lower class. A hierarchical key corresponding to an access class is its residual set. After the deployment, base station generates the access class hierarchy and the hierarchical keys. Base station uses PKI to distribute hierarchical key of the highest access class clusters. An higher access class cluster generates and distributes a hierarchical key for a lower access class cluster. Resilience of the solution is low since sensor nodes in root access class have more powerful keys than the others, i.e. they can generate the keys of the lower access class.

Group key distribution via local collaboration [45] is based on the *key sharing scheme* due to Shamir [59]. Proposed scheme assumes that each sensor node can identify the compromised sensors within its vicinity and report them to the base station. Once the base station knows the list of compromised nodes, a group-wise key can be established by using local collaboration among the sensor nodes. During key setup phase, for each sensor node S_i, two polynomials $h(x)$ and $l(x)$ in F_q are randomly selected. Sensor node S_i receives $h(i)$ and $l(i)$. Group-wise key establishment is initiated by the base station. Base station sends a broadcast message which includes a set R of compromised node IDs and a polynomial $w(x) = g(x)f(x) + h(x)$ where $g(x) = (x - r_1)(x - r_2)\dots(x - r_w)$ for $r_1, r_2, \dots, r_w \in R$. On receiving the broadcast message, each sensor node S_i obtains its personal key $f(i)$ as $f(i) = \frac{w(i)-h(i)}{g(i)}$. A compromised node S_r can receive the broadcast message but it can not generate $f(r)$ since $g(r) = 0$. Finally, each sensor receives concealed personal key $s(j) = f(j) + l(j)$ from its t neighbors. Thus, each sensor can apply *key sharing scheme* due to Shamir [59] to obtain the group-wise key $s(0)$. Group-wise key can be updated by the base station when new nodes are added or nodes are compromised. Proposed scheme provides good resilience but has high storage, communication and processing overhead.

Logical key distribution architecture [48] is proposed as a solution to the *multicast key management problem* which is also used in [49] as a group-wise key management scheme for HWSN. In this approach, keys are organized in a hierarchical tree structure. Each sensor node corresponds to a key on a leaf. In key pre-distribution phase, each node receives the key on the corresponding leaf and all the keys on the path from the leaf to the root. Figure 4 provides a sample key hierarchy for eight nodes. The key at the root can be used as the group-wise key. Advantage of this architecture is its efficiency for node insertion and deletion. When a node is deleted, all the keys stored in its key-chain should be

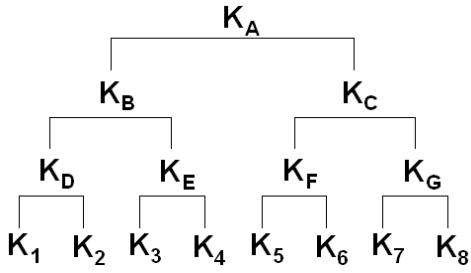


Figure 4. Logical key distribution architecture for eight sensor nodes. Each sensor node is assigned to a leaf and stores all keys on the path from the leaf to the root.

updated in the key tree. Assume that node S_8 is compromised in Figure 4. The keys $\{K_A, K_C, K_G\}$ must be updated and K_8 must be deleted. Each updated key is encrypted with its children and sent to the nodes on corresponding leaves (e.g., K_G is encrypted with K_7 and is sent to node S_7, K_C is encrypted with K_F and is sent to nodes S_5 and S_6). Solution requires each node to store $\log N + 1$ keys for a network size N. Deletion of a node requires $2 \log N$ messages broadcasted by the cluster head.

4.3. Network-wise Key Management

Network-wise keys are used to secure base station to sensor node broadcast traffic in HWSN. A straightforward but insecure approach is to pre-distribute a single network-wise key to all sensor nodes. *Multi-tiered security solution* [51] proposes to protect data items to a degree consistent with their value. In key setup phase, each sensor node receives a list of m master keys, a PRF and a seed. The PRF is used with the seed to obtain an index within the list of master keys. Selected master key is named as *active master key*. *RC6* is used as encryption algorithm. Three security levels are defined. In level one, a strong encryption algorithm and active master key is used to secure *mobile codes*. In level two, sensor nodes are divided into cells. A common location security key is generated within each cell, and it is used to secure *location information*. Finally in level three, MD5 hash of the active master key is used to secure *application data*. Problem is that public credentials (i.e., master key list, PRF and seed) are subject to compromise.

Timed Efficient Stream Loss-tolerant Authentication (TESLA) [50] is a multicast stream authentication protocol. *TESLA* uses *delayed key disclosure mechanism* where the key used to authenticate i^{th} message is disclosed along with $(i+1)^{th}$ message. *SPINS* [52] uses $\mu - TESLA$ which is an adoption of TESLA for HWSNs and provides authentication for data broadcasts. $\mu - TESLA$ requires base station and sensor nodes be loosely time synchronized. Basically, base station (BS) randomly selects the last key K_n of a chain, and applies one-way public hash function *HASH* to generate the rest of the chain K_0, K_1, ..., K_{n-1} as $K_i = HASH(K_{i+1})$. Given K_i, every sensor node can generate the sequence K_0, K_1, ..., K_{i-1}. However, given K_i, no one can generate K_{i+1}. At i^{th} time slot, BS sends authenticated message $MAC_{K_i}(Message)$. Sensor nodes store the message

until BS discloses the verification key in $(i+d)^{th}$ time slot where d is the delay disclosure. Sensor nodes can verify disclosed verification key K_i by using the previous key K_{i-1} as $K_{i-1} = HASH(K_i)$. In $\mu - TESLA$, nodes are required to store a message until the authentication key is disclosed. This operation may create storage problems, and encourages DoS types of attacks. An adversary may jam key disclosure messages to saturate storages of sensor nodes. $\mu - TESLA$ requires sensor nodes to bootstrap from the BS; that is, they receive the first key of the chain, called *key-chain commitment*, secured with pair-wise keys. Bootstrapping procedure requires unicast communication, causes high volume of packets flowing in WSN, and creates scalability problems.

$\mu - TESLA$ is used in [38,39] to authenticate message broadcasts from BS, in [53] to authenticate route update broadcasts, and in *LEAP* [41] to update predeployed network-wise keys in the case of a node compromise. *TESLA Certificate* [56] uses a base station as certificate authority (CA). In this scheme, CA generates certificate $Cert(ID_A, t_{i+d}, ..., MAC_{K_i}(...))$ for sensor node S_A at time t_i. It discloses the key K_i at time t_{i+d} when the certificate expires.

Bootstrapping of the key-chain commitments in $\mu - TESLA$ causes high volume of packets flowing in WSN, and creates scalability problems. $\mu - TESLA$ *extensions* [54,55] proposes five approaches based on two or more level key-chains to address scalability issues. In *predetermined key-chain commitment*, commitment is pre-distributed to sensors before the deployment. In this solution, key-chain must cover lifetime of sensor nodes to prevent bootstrapping requirements. This can be achieved by using either long chains or large time intervals. A new coming node has to generate whole key-chain from the beginning to authenticate recently disclosed key. Thus, long key-chain means excessive processing for sensor nodes which are deployed at a later time. Large time interval means increased number of messages to store because sensor nodes have to store incoming messages until the authentication key is disclosed. *Two-level key chains scheme* addresses these problems as illustrated in Figure 5. There is a high-level key-chain with long enough time interval to cover the life time of sensor nodes, and multiple low-level key-chains with short enough intervals. High-level key-chain is used to distribute and authenticate randomly generated commitments of low-level key-chains. In this scheme, sensor nodes are initialized with the commitment of high-level chain, time intervals of high-level and low-level key chains and one-way functions of high and low-level chains. However, low-level keys are not chained together. Thus, loss of a low-level key disclosure can only be recovered with a key which is disclosed later within the same interval. Moreover, loss of a low-level key commitment may also mean loss of entire interval. An adversary may take advantage of this, and may jam disclosure of low-level key commitments. *Fault tolerant two-level key-chains scheme* is proposed to address these issues. In this scheme, the commitments of low-level key chains are not randomly generated, but obtained from high-level keys by using another one-way hash function. Low-level key commitments are periodically broadcasted; however, an adversary may still recover the commitment period, and can jam disclosure of low-level key commitments. *Fault tolerant two-level key-chains with random commitments scheme* uses a random process to broadcast the low-level commitments. Finally, *multi-*

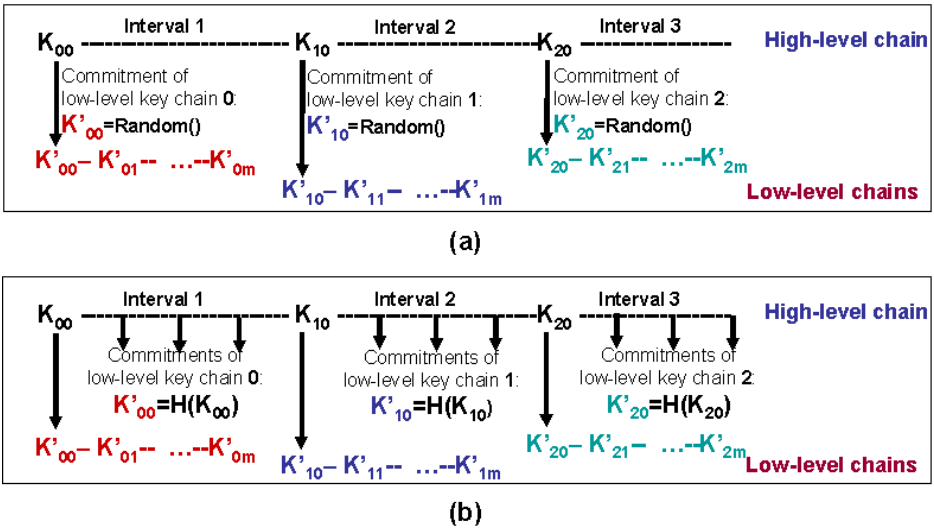


Figure 5. $\mu - TESLA$ extensions: (a) two-level key chains scheme and (b) fault tolerant two-level key chains scheme. Downward arrows show broadcast of the low-level key commitments for each interval: (a) commitments are broadcasted at the beginning of the interval, (b) commitments are broadcasted periodically throughout the interval.

level chains scheme is proposed to provide smaller time intervals and shorter key chains.

5. Summary and Discussions

Figure 6 provides taxonomy of papers on key distribution problems in DWSN and HWSN. In this figure, graphs are DAGs (directed acyclic graphs) where nodes represent papers. Directed edges show predecessor-successor relations among the papers. There is an edge from a paper to another one if latter provides improvement for the solutions proposed by former. Nodes (papers) are ordered over a horizontal time axis according to their publication dates. Vertical axis groups papers under three problems: (i) pair-wise, (ii) group-wise, and (iii) network-wise key management problems. Each problem is represented with a specific node, named as *origin node*, which has only outgoing edges. The style of an edge (dotted, dashed, solid) in between two nodes represents the problem in which an improvement is provided. A paper may provide more than one solutions to more than one problems; therefore, corresponding node may be reachable from more than one *origin nodes*, and there may be more than one edges with different styles in between two nodes.

Detailed evaluation for the solutions is given in Table 4. Solutions are grouped, as in Sections 3 and 4, based on the keying problem and style. Details about the solutions are provided for six metrics: (S)-scalability, (K)-key connectivity, (R)-resilience, (M)-storage complexity, (P)-processing complexity, and (C)-communication complexity. Scalability "S" is ability to support larger networks.

Larger networks can be supported if there is enough storage for the required security credentials which is related to storage complexity of the solution. Scalability of the similar (same keying problem and keying style) solutions are compared with each other. Basically, each solution is assigned a scalability rank where higher rank means higher scalability. There can be more than one solution sharing the same rank which means that corresponding solutions have roughly the same scalability. Resilience "R" of each solution is given as either one of the following ways: (i) probability that a link is compromised when an adversary captures a node, (ii) number of nodes whose security credentials are compromised when an adversary captures a node, or (iii) number of sensor nodes required to be captured to compromise whole WSN. Third one is represented as *n-secure* meaning that it is enough to capture *n+1* nodes to compromise whole WSN. As first two values increase, network becomes less secure; therefore, resilience decreases. Key connectivity "K" considers probability that two (or more) sensor nodes store the same key or keying material to be able to establish pair-wise, group-wise or network-wise keys. Efficiency of the solutions is measured with their storage, processing and communication overheads. Storage complexity "M" is amount of memory units required to store security credentials. We consider key, key ID, node ID, node locations, etc. as one memory unit. Processing complexity "P" is number of unit functions executed. Unit functions can be: (i) *Search* for one or more key in a key-chain, (ii) functions such as *PRF*, *Hash*, *MAC*, *XOR* and *ENC*, (ii) *VecMul(size)* which multiplies two vectors of given sizes, and (iii) *PolyEval* which evaluates a polynomial at a given point. Communication is the most energy consuming operation performed by a sensor node. Communication complexity "C" is measured as number and size of packets sent and received by a sensor node.

Based on the results shown in Tables 4, we conclude that there are significant tradeoffs and, there is no one-size-fits-all solution for key distribution problems in WSNs.

References

[1] G. Wang, G. Cao, , T. L. Porta, Movement-assisted sensor deployment, in: IEEE INFOCOM, 2004, pp. 2469–2479.

[2] Y. Zou, K. Chakrabarty, Sensor deployment and target localization based on virtual forces, in: IEEE INFOCOM, 2003, pp. 1293–1303.

[3] M. Jakobsson, S. Wetzel, B. Yener, Stealth attacks on ad-hoc wireless networks, in: IEEE Vehicular Technology Conf., 2003, pp. 2103–2111.

[4] C. Karlof, D. Wagner, Secure routing in wireless sensor networks: Attacks and countermeasures, in: IEEE Int. Workshop on Sensor Netw. Protocols and Appl., 2003, pp. 113–127.

[5] L. Zhou, Z. Haas, Securing ad hoc networks, IEEE Netw. Magazine 13 (6) (1999) 24–30.

[6] F. Stajano, R. Anderson, The resurrecting duckling: security issues for ad-hoc wireless networks, in: Int. Workshop on Security Protocols, 1999, pp. 172–194.

[7] D. J. Malan, M. Welsh, M. D. Smith, A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography, in: IEEE Int. Conf. Sensor and Ad Hoc Commun. and Netw., 2004, pp. 71–80.

[8] G. Gaubatz, J. P. Kaps, B. Sunar, Public key cryptography in sensor networks, in: European Workshop on Security in Ad-Hoc and Sensor Netw., 2004, pp. 2–18.

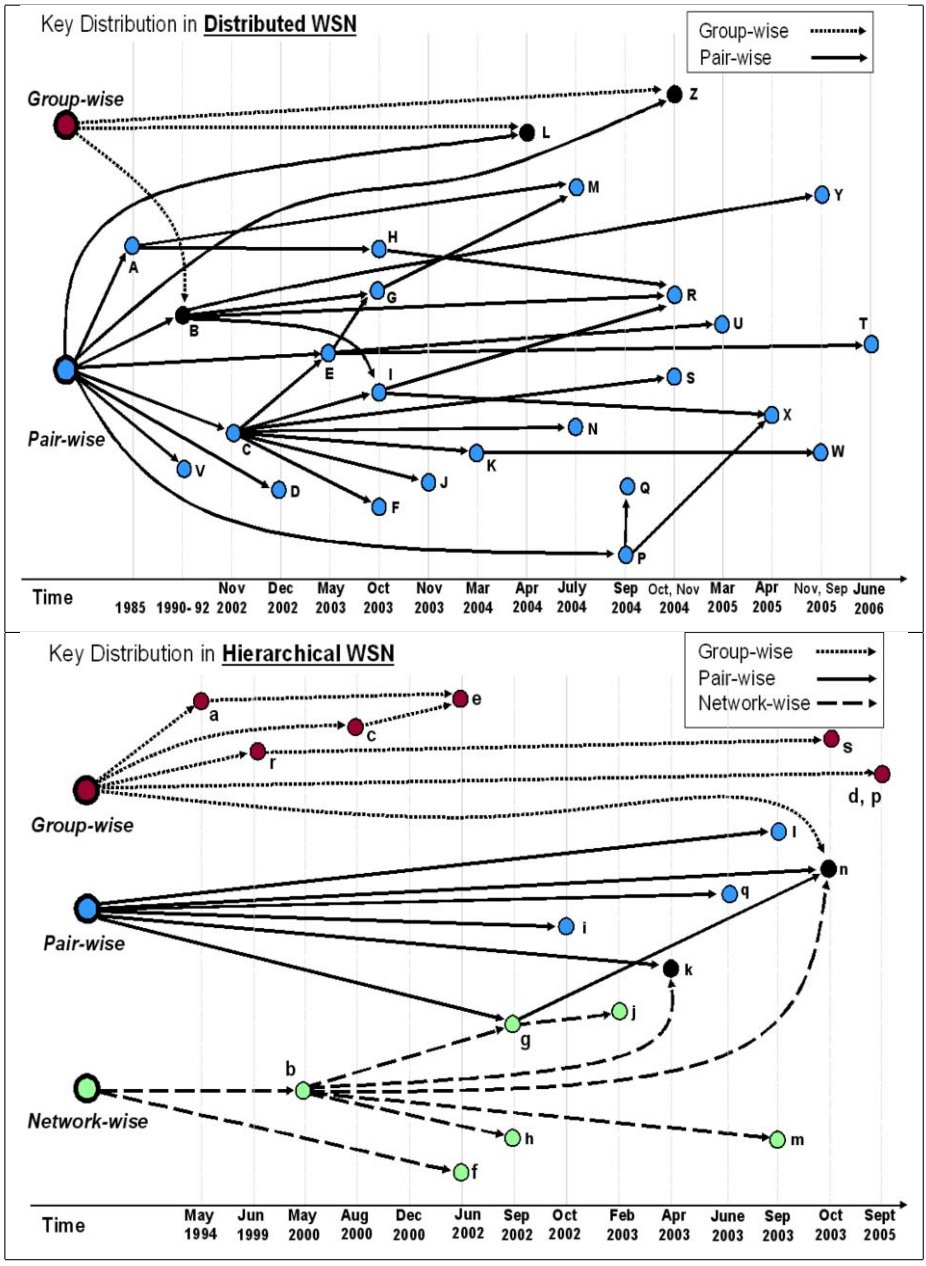


Figure 6. Taxonomy of the papers on key distribution problems in DWSN and HWSN. Graphs are DAGs (directed acyclic graphs) where nodes represent papers, and edges represent predecessor/successor relations (improvements) among solutions provided by the papers. There are three nodes which have only outgoing edges, and which represent the pair-wise, group-wise and network-wise key distribution problems. Style of an edge represents the problem on which destination node (paper) provides improvements. The nodes are: A[16], B[17], C[25], D[18], E[10], F[26], G[11], H[22], I[23], J[27], K[28], L[19], M[12], N[29], P[32], Q[33], R[24], S[30], T[13], U[14], V[15], W[31], X[20], Y[21], Z[34], a[43], b[50], c[44], d[45], e[46], f[51], g[52], h[53], i[37], j[54,55], k[38,39], l[40], m[56], n[41], p[47], q[42], r[48], s[49].

[9] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, J. Zhang, Fast authenticated key establishment protocols for self-organizing sensor networks, in: ACM Int. Workshop on Wireless Sensor Netw. and Appl., 2003, pp. 141–150.
[10] H. Chan, A. Perrig, D. Song, Random key predistribution schemes for sensor networks, in: IEEE Symp. Security and Privacy, 2003, p. 197.
[11] D. Liu, P. Ning, Location-based pairwise key establishment for static sensor networks, in: ACM Workshop on Security of Ad Hoc and Sensor Netw., 2003, pp. 72–82.
[12] J. Lee, D. R. Stinson, Deterministic key pre-distribution schemes for distributed sensor networks, in: ACM Symp. Applied Computing, 2005, pp. 294–307.
[13] S. A. Camtepe, B. Yener, M. Yung, Expander graph based key distribution mechanisms in wireless sensor networks, in: IEEE Int. Conf. on Commun., 2006, pp. 2262–2267.
[14] H. Chan, A. Perrig, Pike: Peer intermediaries for key establishment, in: IEEE INFOCOM, 2005, pp. 524–535.
[15] L. Gong, D. J. Wheeler, A matrix key distribution scheme, Journal of Cryptology 2 (1) (1990) 51–59.
[16] R. Blom, An optimal class of symmetric key generation systems, in: EUROCRYPT, 1984, pp. 335–338.
[17] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung, Perfectly-secure key distribution for dynamic conferences, in: Advances in Cryptology, 1992, pp. 471–486.
[18] B. Lai, S. Kim, I. Verbauwhede, Scalable session key construction protocol for wireless sensor networks, in: IEEE Workshop on Large Scale Real-Time and Embedded Systems, 2002.
[19] B. Dutertre, S. Cheung, J. Levy, Lightweight key management in wireless sensor networks by leveraging initial trust, Tech. Rep. SRI-SDL-04-02, System Design Laboratory (2004).
[20] D. S. Sanchez, H. Baldus, A deterministic pairwise key pre-distribution scheme for mobile sensor networks, in: Security and Privacy for Emerging Areas in Commun. Netw., 2005, pp. 277–288.
[21] F. Delgosha, F. Fekri, Key pre-distribution in wireless sensor networks using multivariate polynomials, in: Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2005, pp. 118– 129.
[22] W. Du, J. Deng, Y. S. Han, P. K. Varshney, A pairwise key pre-distribution scheme for wireless sensor networks, in: ACM Conf. Computer and Commun. Security, 2003, pp. 42–51.
[23] D. Liu, P. Ning, Establishing pairwise keys in distributed sensor networks, in: ACM Conf. Computer and Commun. Security, 2003, pp. 231–240.
[24] D. Huang, M. Mehta, D. Medhi, L. Harn, Location-aware key management scheme for wireless sensor networks, in: ACM Workshop on Security of Ad Hoc and Sensor Netw., 2004, pp. 29–42.
[25] L. Eschenauer, V. D. Gligor, A key-management scheme for distributed sensor networks, in: ACM Conf. Computer and Commun. Security, 2002, pp. 41–47.
[26] R. D. Pietro, L. V. Mancini, A. Mei, Random key assignment secure wireless sensor networks, in: ACM Workshop on Security of Ad Hoc and Sensor Netw., 2003, pp. 62–71.
[27] S. Zhu, S. Xu, S. Setia, S. Jajodia, Establishing pairwise keys for secure communication in ad hoc networks: a probabilistic approach, in: IEEE Int. Conf. Netw. Protocols, 2003, p. 326.
[28] W. Du, J. Deng, Y. S. Han, S. Chen, P. K. Varshney, A key management scheme for wireless sensor networks using deployment knowledge, in: IEEE INFOCOM, 2004, p. 597.
[29] D. Hwang, B. Lai, I. Verbauwhede, Energy-memory-security tradeoffs in distributed sensor networks, in: ADHOC-NOW, LNCS, Vol. 3158, 2004, p. 7081.
[30] J. Hwang, Y. Kim, Revisiting random key pre-distribution for sensor networks, in: ACM Workshop on Security of Ad Hoc and Sensor Netw., 2004, pp. 43–52.
[31] T. Ito, H. Ohta, N. Matsuda, T. Yoneda, A key pre-distribution scheme for secure sensor networks using probability density function of node deployment, in: ACM Workshop on Security of Ad Hoc and Sensor Netw., 2005, pp. 69–75.
[32] S. A. Camtepe, B. Yener, Combinatorial design of key distribution mechanisms for wireless sensor networks, in: ESORICS, 2004, pp. 293–308.

[33] J. Lee, D. R. Stinson, A combinatorial approach to key pre-distributed sensor networks, in: IEEE Wireless Commun. and Netw. Conf., 2005, pp. 1200–1205.
[34] M. Ramkumar, N. Memon, An efficient random key pre-distribution scheme, in: IEEE Global Telecommunications Conference, 2004, pp. 2218– 2223.
[35] R. Merkle, Secure communication over insecure channels, Commun. of the ACM 21 (4) (1978) 294–299.
[36] A. Wacker, M. Knoll, T. Heiber, K. Rothermel, A new approach for establishing pairwise keys for securing wireless sensor networks, in: Proceedings of the 3rd international conference on Embedded networked sensor systems, 2005, pp. 27–38.
[37] J. Undercoffer, S. Avancha, A. Joshi, J. Pinkston, Security for sensor networks, in: CADIP Research Symp., 2002.
[38] J. Deng, R. Han, S. Mishra, Enhancing base station security in wireless sensor networks, Tech. Rep. CU-CS-951-03, Department of Computer Science, University of Colorado (2003).
[39] J. Deng, R. Han, S. Mishra, A performance evaluation of intrusion-tolerant routing in wireless sensor networks, in: IEEE IPSN, 2003, pp. 349–364.
[40] Y. W. Law, R. Corin, S. Etalle, P. H. Hartel, A formally verified decentralized key management for wireless sensor networks, in: Int. Conf. Personal Wireless Commun., 2003, pp. 27–39.
[41] S. Zhu, S. Setia, S. Jajodia, Leap: Efficient security mechanisms for large-scale distributed sensor networks, in: ACM Conf. Computer and Commun. Security, 2003, pp. 62–72.
[42] G. Jolly, M. C. Kuscu, P. Kokate, M. Younis, A low-energy key management protocol for wireless sensor networks, in: Eighth IEEE International Symposium on Computers and Communication, 2003, pp. 335– 340.
[43] M. Burmester, Y. Desmedt, A secure and efficient conference key distribution system, in: EUROCRYPT, 1994, p. 275.
[44] M. Steiner, G. Tsudik, M. Waidner, Key agreement in dynamic peer groups, IEEE Trans. Parallel and Distributed Systems 11 (8) (2000) 769–780.
[45] A. Chadha, Y. Liu, S. K. Das, Group key distribution via local collaboration in wireless sensor networks, in: IEEE Int. Conf. Sensor and Ad Hoc Commun. and Netw., 2005, pp. 46–54.
[46] D. W. Carman, B. J. Matt, G. H. Cirincione, Energy-efficient and low-latency key management for sensor networks, in: Army Science Conf., 2002.
[47] M. Shehab, E. Bertino, A. Ghafoor, Efficient hierarchical key generation and key diffusion for distributed sensor networks, in: IEEE Int. Conf. Sensor and Ad Hoc Commun. and Netw., 2005, pp. 76–84.
[48] D. Wallner, E. Harder, R. Agee, Key management for multicast: Issues and architectures, RFC 2627 (Informational) (jun 1999).
URL `http://www.ietf.org/rfc/rfc2627.txt`
[49] R. D. Pietro, L. Mancini, Y. W. Law, S. Etalle, P. Havinga, Lkhw: A directed diffusion-based secure multicast scheme for wireless sensor networks, in: 32nd Int. Conf. on Parallel Processing Workshops (ICPP), 2003, pp. 397–406.
[50] A. Perrig, R. Canetti, J. D. Tygar, D. X. Song, Efficient authentication and signing of multicast streams over lossy channels, in: IEEE Symp. Security and Privacy, 2000, pp. 56–73.
[51] S. Slijepcevic, M. Potkonjak, V. Tsiatsis, S. Zimbeck, M. B. Srivastava, On communication security in wireless ad-hoc sensor network, in: IEEE WETICE, 2002, pp. 139–144.
[52] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J. D. Tygar, Spins: Security protocols for sensor networks, Wireless Networks 8 (5) (2002) 521 – 534.
[53] J. Staddon, D. Balfanz, G. Durfee, Efficient tracing of failed nodes in sensor networks, in: ACM Int. Workshop on Wireless Sensor Netw. and Appl., 2002, pp. 122–130.
[54] D. Liu, P. Ning, Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks, in: Network and Distributed System Security Symp., 2003, pp. 263–276.
[55] D. Liu, P. Ning, Multi-level u-tesla: A broadcast authentication system for distributed sensor networks, Tech. Rep. TR-2003-08, Department of Computer Science, North Carolina

State University (2003).
[56] M. Bohge, W. Trappe, An authentication framework for hierarchical ad hoc sensor networks, in: ACM WiSe, 2003, pp. 79 – 87.
[57] A. Shamir, Identity-based cryptosystems and signature schemes, in: Advances in Cryptology, 1984, pp. 47–53.
[58] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, in: Advances in Cryptology, 2001, pp. 213–229.
[59] A. Shamir, How to share a secret, Commun. of the ACM 22 (11) (1979) 612–613.

Table 4.: Evaluation of the solutions. Solutions are grouped, as in Sections 3 and 4, based on the keying problem and solution style. Citation of the paper which provides the corresponding solution is listed in *Ref* column. Details of the solutions are provided for six metrics: (S)-scalability, (K)-key connectivity, (R)-resilience, (M)-storage overhead, (P)-processing overhead, and (C)-communication overhead. Numerical values in scalability column are the ranks of the solutions within each section where higher ranks mean higher scalability. Resilience column can take three different classes of values : (i) a number or an equation which represents probability that a link is compromised when an adversary captures a node, (ii) a number or an equation with keyword *nodes* which represents number of sensor nodes whose security credentials are compromised when an adversary captures a node, and (iii) a number or an equation with keyword *secure* which represents number of sensor nodes required to compromise security of whole WSN. Processing overhead is provided in terms of unit functions such as Srch (Search), Hash, MAC, PRF, HMAC, VecMul(size), PolyEval(count), etc. Communication overhead includes number and size of messages sent and received where $a \times b,\ c \times d$ means a number of messages of size b units are sent and c number of messages of size d units are received. Parameters used for each solution are described in detail in Sections 3 and 4. Summary of parameters are: (**d**) degree of a node, (**p**) probability that two nodes are connected due to Erdos and Renyi's work, (**c**) number of cooperative nodes, (**ac**) number of access classes, (**r**) regularity of a connected key distribution graph, (ℓ) number of nodes in a node class, (θ) number of nodes in a generation, (**g**) number of generations, (**j**) number of paths, (ω) number of spaces, (τ) number of spaces assigned to a node, (**k**) key-chain size $|KP|$, (**m**) number of keys in master key list of a node, (**u**) number of commitment disclosure, (**v**) number of high level commitment disclosure, (**w**) number of low level commitment disclosure, and (**L**) number of derived keys.

Solution	Ref	(S)	(K)	(R)	(M)	(P)	(C)
Dedicated pair-wise key management solutions in DWSN (Section 3.1)							
All pair-wise	-	1	1	0	2(N-1)	Srch	1x1,dx1
Random pair-wise	[10]	2	Np/(N-1)	0	2Np	Srch	1x1,dx1
Matrix key	[15]	1	1	$\sqrt{N}$-secure	$2\sqrt{N}$	Srch	1x2,dx2
Closest pair-wise	[11]	3	c/(N-1)	0	2c+1	Srch or 1xPRF	1x1,dx1
IOS	[12]	3	r/(N-1)	0	r+1	Srch or 1xHash	1x1,dx1
Multiple IOS	[12]	4	rℓ/(N-1)	ℓ nodes	r/ℓ+1	Srch or 1xHash	1x1,dx1
Expander graph	[13]	2	r/(N-1)	0	r+1	Srch	1x1,dx1

Continued on Next Page...

Table 4 – Continued

PIKE	[14]	2	$\frac{2(\sqrt{N}-1)}{(N-1)}$	0	2($\sqrt{N}$-1)	Srch	1x1,dx1
BROSK	[18]	1	1	1	1	1xPRF	1x1,dx1
Lightweight key management	[19]	1	1	θ nodes	1+2g	Srch or 1xPRF	1x2,dx2
Blom's scheme	[16]	2	1	λ-secure	2(λ+1)	VecMul(λ+1)	1x(λ+1),dx(λ+1)
MBS	[12]	3	r/(N-1)	λ-secure	2(λ+1)	VecMul(λ+1)	1x(λ+1),dx(λ+1)
DMBS	[12]	4	rℓ/(N-1)	ℓ nodes	(r/ℓ+1)(λ+1)	VecMul(λ+1)	1x(λ+1),dx(λ+1)
Polynomial based	[17]	3	1	λ-secure	λ+1	PolyEval(1)	1x1,dx1
Location-based pair-wise	[11]	1	1	λ-secure	5(λ+1)	PolyEval(1)	1x2,dx2
Deterministic poly. pool	[20]	2	1	λ-secure	k(λ+1)	PolyEval(1)	1xk,dxk
Multiple space	[22]	1	$\frac{((\omega-\tau)!)^2}{((\omega-2\tau)!\omega!)}$	λ-secure	2τ(λ+1)	VecMul(λ+1)	1xτ+1x(λ+1), dxτ+dx(λ+1)
Polynomial pool	[23]	2	$\frac{2(\sqrt{N}-1)}{(N-1)}$	λ-secure	2(λ+1)	PolyEval(1)	1x2,dx2
Grid-group deployment	[24]	2	1	λ-secure	2(λ+1)	PolyEval(1)	1x2,dx2
Reusable pair-wise key management solutions in DWSN (Section 3.2)							
Basic probabilistic	[25]	2	$\frac{((KP-k)!)^2}{((KP-2k)!KP!)}$	k/KP	2k	Srch	1xk,dxk
Cluster key grouping	[29]	2	$\frac{((KP-k)!)^2}{((KP-2k)!KP!)}$	k/KP	2k	Srch	1xC,dxC
Pair-wise key establishment	[27]	3	$\frac{((KP-k)!)^2}{((KP-2k)!KP!)}$	k/KP	k	Srch+1xPRF	1x1,dx1
Q-composite random	[10]	1	see [10]	$\binom{k}{q}/\binom{KP}{q}$	2k	Srch	1xk,dxk
Using deployment knowledge	[28]	2	see [28]	k/KP	2k	Srch	1xk,dxk
Using sensor pdf	[31]	2	$\frac{((KP-k)!)^2}{((KP-2k)!KP!)}$	k/KP	2k	Srch	1xk,dxk
Combinatorial - Symmetric	[32]	1	1	1/n	2(n+1)	Srch	1xn,dxn
Combinatorial - GQ(n, n^2)	[32]	2	$1/n^2$	$1/n^3$	2(n+1)	Srch	1xn,dxn
Combinatorial - Hybrid	[32]	3	see P	$1/n^3$	2(n+1)	Srch	1xn,dxn
Key update mechanisms							

Continued on Next Page...

Table 4 – Continued

Multi-path key reinforcement	[10]	2	$\frac{((KP-k)!)^2}{((KP-2k)!KP!)}$	0	2k	j XOR+Srch	1xk+jx1,dxk+jx1
Pair-wise with threshold	[27]	2	$\frac{((KP-k)!)^2}{((KP-2k)!KP!)}$	0	2k	j XOR+Srch	1xk+jx1,dxk+jx1
Co-operative pair-wise	[26]	2	$\frac{((KP-k)!)^2}{((KP-2k)!KP!)}$	0	2k	c XOR+Srch	1xk+cx1,dxk+cx1
Group-wise key management solutions in DWSN (Section 3.3)							
Polynomial - non-interactive	[17]	1	1	λ-secure	λ+1	PolyEval(1)	1x1,(t-1)x1
Polynomial - interactive	[17]	1	1	λ-secure	λ+1	PolyEval(t-1) + 1xXOR	tx1,(t-1)x1
HARPS	[34]	1	1	L-secure	2k	Srch and Hash	1x1,dx1
Pair-wise key management solutions in HWSN (Section 4.1)							
LEAP pair-wise	[41]	1	1	1	2	1xMAC	2x2,2x1
Low-energy	[42]	1	1	0	2	1xENC	2x4,2x4
Group-wise key management solutions in HWSN (Section 4.2)							
LEAP group-wise	[41]	1	1	1	1	mxENC	0,mx1
Hierarchical key	[47]	1	1	1	1	acxHash	0,mx1
Local collaboration	[45]	1	1	0	4	PolyEval(2)+ [59]	dx1,dx1+1x2t
Network-wise key management solutions in HWSN (Section 4.3)							
Multi-tiered	[51]	1	1	1	m	1xPRF+1XHash	0,0
micro-TESLA	[52]	1	1	0	high	1xMAC+1XHash	0,ux1
TESLA Certificate	[56]	1	1	0	high	2xMAC	0,3ux1
μ-TESLA extensions	[54] [55]	2	1	0	low	1xMAC+1XHash	0,vwx1

Wireless Sensor Network Security
J. Lopez and J. Zhou (Eds.)
IOS Press, 2008

WSN Link-layer Security Frameworks

Ioannis KRONTIRIS [1], Tassos DIMITRIOU, Hamed SOROUSH and Mastooreh SALAJEGHEH
Athens Information Technology, Greece

Abstract. In this chapter we elaborate on the need for security frameworks at the link-layer and describe what services they provide to the upper layers. We review the proposed frameworks in the bibliography and discuss about their pros and cons. Then we present in more detail the design and implementation of one of them, the L^3Sec framework, in order to show what issues arise in such a process and how they can be solved. Some of these features include providing acceptable resistance against node capture attacks and replay attacks, as well as the run-time composition of security services in a completely transparent way. The framework is able to satisfy its requirements based on an an efficient scalable post-distribution key management scheme, which we also present.

Keywords. Frameworks, Security Services, Link-layer security

Introduction

The different directions of ongoing research in WSNs are based on security challenges that address several classes of attacks and how sensor networks can defend against them. Another, more general approach to address security in WSNs can be on a per-layer basis, instead of a per-attack basis. Under this perspective, security protocols can be designed to provide security in a particular layer and cooperate with security protocols in other layers to compose a complete defence for the nodes and the network.

This layer based classification of security protocols can help towards a more clear understanding of WSN security and better protocol design. However, in practical sensor networks, such as those using the TinyOS platform, there does not exist a clear formal way for demarcating between the various layers. Nevertheless, one could break the network stack in TinyOS into four major layers: the physical layer, the link/MAC layer (in the remainder of this chapter we will refer to this simply as the link layer), the routing layer, and the application layer.

On this basis we could range security protocols in a corresponding layered taxonomy, as shown in Figure 1 [1]. Protocols in higher layers use security services provided by lower layers or depend on their reliable functionality. There are also other security issues like intrusion detection that cannot be classified in this layered approach as they are more general problems that spam different layers. Currently, research on security solutions for WSNs has focused mainly in the following three categories:

[1]Corresponding Author: Ioannis Krontiris, Athens Information Technology, 19.5 Km Markopoulo Avenue, 19002 Peania, Greece; E-mail: ikro@ait.edu.gr.

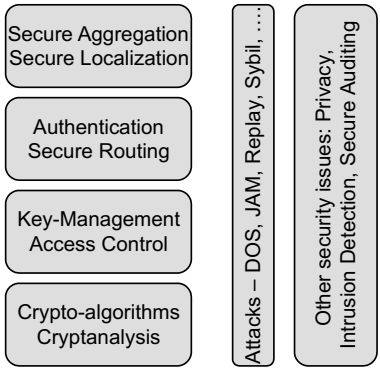


Figure 1. Sensor networks security map based on a layered approach.

1. *Key management:* A lot of work has been done [2] in establishing cryptographic keys between nodes to enable encryption and authentication. These protocols can be classified as link layer protocols.
2. *Authentication and Secure Routing:* Several protocols [3] have been proposed to protect information from being revealed to an unauthorized party and guarantee its integral delivery to the base station. These protocols clearly belong to the routing layer.
3. *Secure services:* Certain progress has been made in providing specialized secure services, like secure localization [4], secure aggregation [5] and secure time synchronization [6]. These services are closer to the application layer.

Historically speaking, research in WSN security first focused on key management, as it is the most basic requirement for providing further security services. Another area that attracted a lot of research work was secure routing, but the solutions given either targeted a specific routing protocol or confronted only a narrow class of attacks. Also a lot of effort has been made to provide specialized secure services, but as network and application protocols continued to flourish, security researchers realized that more general solutions were needed to address the diversity of these protocols. As a result, attention gradually turned to lower layers of the protocol stack, notably the link layer.

What makes link-layer security important is that end-to-end security mechanisms are not possible in sensor networks, so more transparent mechanisms provided by the link layer are needed. Protocols used in conventional networks for end-to-end security, such as SSH [7], SSL[8], or IPSec[9], even though they are feasible in constrained embedded devices [10], they are considered inappropriate since they don't allow in-network processing and data aggregation which play an important role in energy-efficient data retrieval. These operations require the intermediate nodes to access and possibly modify the contents of packets, which would not be possible if an end-to-end security scheme was used.

In sensor networks it is also important to allow intermediate nodes to check message integrity and authenticity, or else the network would be prone to several denial of service attacks. Using an end-to-end security mechanism, packets would have to be routed all the way to the base station before these checks could be performed, since the intermediate nodes do not have the keys to verify their authenticity and integrity. On the other

hand, using a transparent security mechanism at the link layer, malicious packets can be identified and rejected at the first hop.

However, since the usual traffic pattern in WSN is many-to-one, pre-loading one-to-one keys between two sensors and refreshing the keys are practically impossible tasks. Public key cryptography is also considered to be computationally expensive for WSN and therefore, light-weight, yet reasonably secure key management schemes are crucial in order to bring about acceptable security services in WSN. In addition to this, any WSN security protocol has to be flexible and scalable enough to easily allow nodes to join or leave the network.

In the rest of this chapter we first emphasize on the main issues involved in designing link-layer security protocols. In Section 2, we review some of the proposed protocols in the bibliography by describing in some detail their design and implementation. Then, in Section 3 we describe a flexible and scalable post-distribution key management module which provides basic cryptographic services and explain how this module can be easily merged with other components and used to secure different operations at different layers. Finally, we describe a security framework based on this module and present several of its implementation details under TinyOS [11], a popular component-based event-driven operating system for WSN.

1. Providing Security at the Link Layer

In this section we describe what features a link layer security protocol provides to other protocols in higher layers. Different protocols that we are going to review in Section 3 use different approaches to provide these features, but here we describe the common directions that they all follow and what issues exist for them to solve.

1.1. Data Confidentiality

Data confidentiality is one of the most basic security primitives and it is used in almost every security protocol. The standard approach for providing confidentiality is to encrypt the data with a secret key that only intended receivers possess. However, even though there are studies [12,13,14,15] indicating that protocols using optimized software implementations of public-key cryptography may be viable on small wireless devices, currently most of the security protocols use symmetric key encryption methods for their power consumption efficiency.

Symmetric encryption schemes to be used in sensor networks can be chosen from stream ciphers (RC4), block ciphers (DES, AES, RC5, and Skipjack) or hashing techniques (SHA-1, MD5). A comparison [16] of encryption overhead amongst schemes from the above groups implemented in embedded architectures showed that RC4 outperforms RC5 on encryption and that hashing techniques require almost an order of a magnitude higher overhead. However, block ciphers are the most widely used schemes in sensor networks because they offer code size optimization, i.e. they can be used both for encryption and authentication. Sensor nodes need to implement a block cipher in any case, in order to provide message authentication, so using it for encryption also, conserves code space. This is why, as we will see, all the protocols reviewed in the remaining sections use a block cipher.

1.2. Data Authentication

Data authentication allows the receiver to verify that the data was actually sent by the claimed sender and not injected in the network by an adversary. It is most often achieved using a message authentication code, or MAC. When the sender and the receiver share a secret key, the sender can compute a MAC of the data to be sent and embed it in the packet. If a packet with a correct MAC arrives, the receiver knows that it must have been sent by the sender and has not been modified in transit.

The most common MAC scheme is CBC-MAC which uses some underlying block cipher to encrypt the base data and then takes the last encrypted block as the MAC value. CBC-MAC is secure for fixed-length messages, given that the underlying block cipher is also secure [17]. However, by itself, it is not secure for variable-length messages. In this case, the messages must be padded to a multiple of the cipher block size. One way to handle this is through the method known as ciphertext stealing, which for the case of sensor networks means that the nodes will need to spend energy for transmitting extra bits. To avoid this problem, some protocols in sensor networks use other block cipher modes, like OCB and CTR, in which the size of the ciphertext is exactly the size of the plaintext and not a multiple of the block size.

1.3. Semantic Security

Semantic Security ensures that an eavesdropper can gain no information about the plaintext, even after observing multiple encryptions of the same plaintext. One common method of achieving this, using a block cipher, is to use a random value or a counter as an Initial Vector (IV) in the encryption function, so that sending the same message will never result in the same ciphertext. However, these IVs would have to be transmitted with the packet, which would consume bandwidth and increase power consumption.

One method used by some protocols is the use of a shared counter between the sender and the receiver which is used as the IV for the block cipher. If both sides increment this counter for each message, the counter does not need to be sent with the message. However, this creates synchronization problems between the nodes, which leads to the need of spending additional energy for re-synchronization.

Another issue here is that when using a counter in the IV it will eventually reset itself and the same value will have to be used again, which means the IV will be the same unless the key has been changed in the mean time. If the counter is to be transmitted with the packet, the counter cannot be a lot of bytes which means it will wrap around sooner. On the other hand, if the shared counters approach is used then the counters are not transmitted and thus they can be longer offering better security.

1.4. Run-time Composition of Security Services

Some link-layer security protocols provide the possibility to choose which of the security features described above will be actually offered. Since each one comes with an extra cost, it might not always be desirable to apply all the options in all cases. So, for example one may want to offer authentication only, but no encryption or replay attack protection. That depends of the risk analysis and the desirable security level for the sensor network. A security framework should implement all of them but it can also offer a mechanism to the user for including or excluding security features.

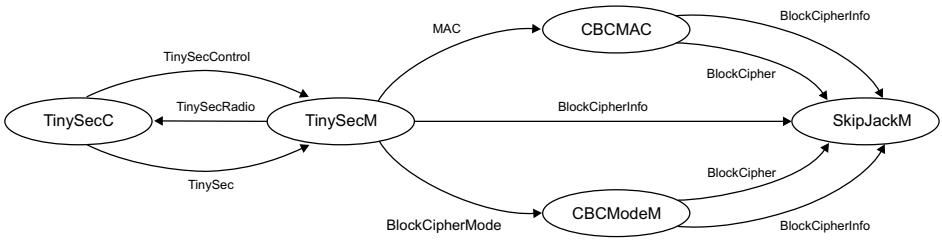


Figure 2. Relationship between components in TinySec.

2. Existing Link-Layer Security Protocols

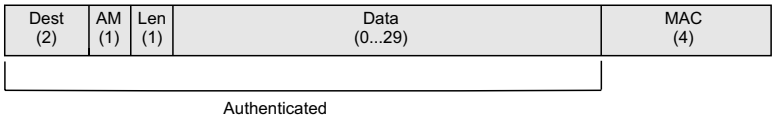
In this section we review some of the proposed link-layer security frameworks for sensor networks. To facilitate their evaluation we first introduce some requirements based on the discussion in Section 1. Then, during the review of each protocol we will elaborate on whether and how it satisfies these requirements.

- *Flexibility*: Various security services should be supported but not imposed to the application level communications. This means that the run-time composition of security services (see Section 1.4) should be provided.
- *Scalability*: Adding or deleting nodes should have minimum overhead in terms of energy consumption and memory usage as well as having no effect on the functionality of the security scheme.
- *Transparency*: The provision of security services should be transparent to other components or services.
- *Lightweightness*: The constrained resources of sensor nodes especially limited memory and computational power should be taken into account.
- *Node Capture Resistance*: The effects of node capture attacks should be constrained as much as possible. If a node is compromised it should not allow the attacker to disclose the communication in the whole network.
- *Simplicity*: The integration of this scheme with other services or components should have a minimal overhead.

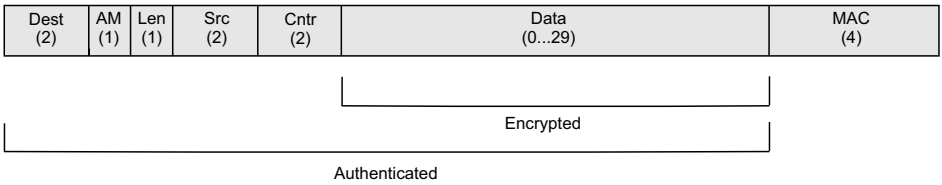
2.1. TinySec

TinySec [18] is a link-layer security architecture for wireless sensor networks that is part of the official TinyOS release. It generates secure packets by encrypting data packets using a group key shared among sensor nodes and calculating a MAC for the whole packet including the header. It provides two modes of operation for communication namely, authenticated encryption and authentication only. Authentication only is the default mode of operation, where the payload in the TinyOS packet is not encrypted; each packet is simply enhanced with a MAC. In the authenticated encryption mode the payload is encrypted before the MAC is computed on the packet. The key distribution mechanism was left out and must be implemented as a separate part of the software. The TinySec architecture is shown in Figure 2.

(a) TinyOS packet format



(b) TinySec-Auth packet format



(c) TinySec-AE packet format

Figure 3. TinySec packet format in the TinySec-Auth and TinySec-AE modes.

2.1.1. Encryption and Authentication

TinySec uses a block cipher algorithm for its encryption scheme that is also used for the message authentication code (MAC) resulting in greater code efficiency. In particular the authors chose the Skipjack block cipher in cipher block chaining (CBC) mode for encrypting TinyOS packets. However, instead of a random IV, they used a counter, which is pre-encrypted. They also used the cipher stealing technique to ensure the ciphertext is the same length as the underlying plaintext. For the authentication of packets, TinySec uses the same block cipher encryption in CBC-MAC mode to generate a 4 byte Message Authentication Code (MAC) for each message. To provide additional security, it XORs the encryption of the message length with the first plaintext block.

2.1.2. TinySec packet format

Figure 3 shows the packet formats for the two modes of TinySec, authenticated encryption (TinySec-AE) and authentication only (TinySec-Auth). As observed, the header fields do not get encrypted, to allow motes quickly determine whether they should reject the packet. The destination address and the AM type are used by the motes for this purpose.

To detect transmission errors, TinyOS motes compute a 16-bit cycle redundancy check (CRC) over the packet. At the receiver the CRC is re-computed and verified with the CRC field in the packet. If they are equal, the receiver accepts the packet and rejects it otherwise. However, CRCs provide no security against malicious modifications or forgery of packets. TinySec replaces the CRC and the GroupID fields with a 3-byte MAC. The MAC protects the payload as well as the header fields. So, since the MAC can detect any changes in the packet, it can also detect transmission errors, therefore CRC is no longer needed.

In the TinySec-AE mode, the data field (payload) is encrypted by the block cipher in CBC mode. Then the MAC is computed over the encrypted data and the packet header. In order to reduce overhead, TinySec uses an 8 byte IV, which is composed of all the header fields in Figure 3(c). In this case the overhead is only 4 bytes, i.e. the source address and a 16-bit counter. This raises an issue on the security level due to the repetition of the IV value. Since the counter is 16 bits, a node can send 2^{16} packets before IV reuse occurs. However, when this happens, only the length (in blocks) of the longest shared prefix of the two repeated messages will be revealed, since CBC mode is used.

2.1.3. TinySec limitations

TinySec by default relies on a single key manually programmed into the sensor nodes before deployment. This network-wide shared key provides only a baseline level of security. It cannot protect against node capture attacks. If an adversary compromises a single node or learns the secret key, she can gain access on the information anywhere in the network, as well as inject her own packets. This is probably the weakest point in TinySec, as node capture has been proved to be a fairly easy process [19]. As we will see in later sections, more recent link-layer security protocols have used stronger keying mechanisms to deal with node capture attacks.

Another limitation of TinySec is that messages of less than 8 bytes are not addressed efficiently. This is because TinySec uses a k-byte block cipher to encrypt the message. For longer messages CBC mode is chosen that encrypts the message block by block. But it is not so unusual for a message (i.e. the payload of the TinyOS packet) to be less than 8 bytes, in which case TinySec will cause a ciphertext expansion, because ciphertext stealing requires at least one block of ciphertext. This kind of ciphertext expansion would cause extra communication power cost when sending data with variable length.

2.2. SenSec

SenSec is a link layer security platform similar to TinySec but with a slightly different packet type and a more resilient keying mechanism. The provision of security services is transparent so that the applications running on the sensors are not aware of the operations on encryption and authentication taking place at the link layer.

While TinySec offers the option between authentication-only and authentication with encryption (AE), SenSec has only one default mode: authentication with encryption. SenSec uses an 8 byte initial vector (IV) and a block cipher in cipher block chaining (CBC) mode to encrypt the data field of the packet, just like TinySec in AE mode. Their difference, however, comes from the IV format. The fixed portions of both IVs are the destination address, the AM type and the length fields. These fields take 4 bytes totally. TinySec fills the other portions of its IV with a 2 byte source address and a 2 byte counter. While this is a reasonable solution so that IV is not repeated often, it requires the nodes to maintain a counter that increases both storage and computation cost. In order to avoid the cost of using a counter mode, SenSec employs a random number mode to generate a three byte random number and fills the IV field with that number along with the one byte group ID.

Instead of generating random numbers with a new algorithm, SenSec employs the existing block cipher module that uses as the main security primitive in its architecture. For simplicity, the first random number is generated by encrypting the first packet header

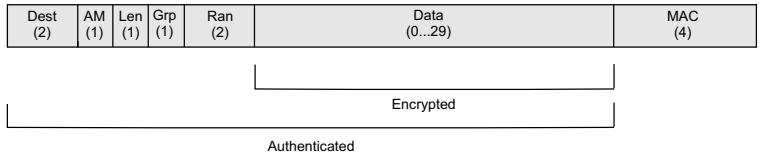


Figure 4. SenSec packet format.

with 3 byte random number field set to 0. Then, the three least significant bytes of the ciphertext are used to fill the random number field in the packet. For the next packets, the 3 byte random number field is determined from the 3 least significant bytes of the MAC, which is computed and stored during the previous packet formatting.

There have been slight enhancements in the cryptographic primitives used in SenSec to improve the security of the platform and further reduce the energy consumption. While 80-bit key Skipjack block cipher is used as the encryption primitive in TinySec, SenSec uses a variant of Skipjack called Skipjack-X which is more resilient against exhaustive key search attacks. It has been shown that SkipjackX is as secure as Skipjack with respect to the various attacks such as the differential cryptanalysis and linear cryptanalysis. Furthermore, it proves stronger against brute force attacks which are the most practical attacks against many block ciphers with small key sizes.

The computation cost of the encryption and MAC is also reduced in SenSec. While TinySec requires two block cipher operations for each message block in a two-pass authentication-encryption scheme, SenSec uses a one-pass authentication-encryption scheme called XCBC. This scheme is secure as long as the total amount of packets being encrypted and authenticated with the same key is much less than 2^{32}. Also, similar to TinySec, SenSec uses 32 bit MACs in order to reduce the packet size.

The default XCBC mode in SenSec, carries out the encryption and authentication for every packet. The reason that SenSec designers have chosen XCBC as their default mode is that it causes the encryption to be done without additional cost, while the encryption in TinySec needs separate CBC-mode operations.

2.2.1. SenSec Packet Format

SenSec's packet format is built upon the current TinyOS packet format and improves slightly upon the TinySec packet format. Figure 4 illustrates the packet format used in SenSec. Compared with TinySec, the unchanged fields in SenSec are the destination field, the active message type and the length field. Unlike TinySec, SenSec keeps the Group ID field, which is also contained in the original TinyOS packet format.

2.2.2. SenSec's Keying Mechanism

In a wireless sensor network, different packets are exchanged between different entities (e.g. sensor to base station, sensor to gateway, etc.). These packets might have different security requirements. For example sensor readings forwarded to the base station need confidentiality and authentication, while routing packets need authentication only. A single keying mechanism such as TinySec's cannot satisfy all these requirements and therefore, designers of SenSec use a many-keying mechanism to protect the whole network from various attacks on one hand, and on the other hand to support effective sensor functions like in-network processing. SenSec employs three level of keys, namely global key

(GK), cluster key (CK) and sensor key (SK) to map the sensor deployment. All of the keys are generated and pre-loaded before deployment.

SenSec's keying mechanism provides some partial resilience against node capture attacks compared to TinySec: if a sensor node is compromised, an adversary can only disclose the group communication of that node through the cluster key, CK. She can also broadcast messages to the network using the global key, GK. But still its node capture resilience is better than that provided by TinySec, which uses only one network-wide shared key.

2.3. SNEP

SNEP (Sensor Network Encryption Protocol) is a building block of SPINS [20] that provides data confidentiality, two-party data authentication, integrity, freshness, semantic security and replay protection. SNEP uses symmetric cryptography with one cryptographic function (RC5) for all of the encryption, decryption, MAC, pseudo random number generation and hash function operations. In order to prevent any potential interaction between the cryptographic primitives that might introduce a weakness, SNEP derives independent keys for its encryption and MAC operations.

SNEP employs the CBC-MAC scheme to construct message authentication codes. For the underlying block cipher it uses RC5. SNEP also offers semantic security by randomization. In order to avoid the extra overhead of sending the randomized data with the packet, it introduces a shared counter between the sender and the receiver which is used as an initialization vector (IV) for the block cipher in counter mode (CTR). Since the counter state is kept at each end point it is not required to be transmitted over the radio channel. The counter value is long enough that it never repeats within the lifetime of the node. This counter value in the MAC also prevents replaying old messages as any messages with the old counter values would be discarded by the device. However, since the counter value is not being sent with the packet, there might be synchronization problems caused by dropped packets. So a re-synchronization protocol may be needed to overcome this problem.

2.4. MiniSec

MiniSec has two different operating modes for unicast and broadcast communication between sensor nodes, called MiniSex-U and MiniSec-B respectively. Both schemes employ the OCB-encryption scheme for both encryption and authentication. They also provide semantic security by the use of a counter as a nonce. In the case of the unicast mode two synchronized counters are kept at the sender and at the receiver, while in the broadcast mode the authors propose the use of a Bloom-filter[2] based mechanism that precludes per-sender state.

2.4.1. MiniSec-U

In unicast mode, MiniSec requires each pair of nodes in the network to share two keys: K_{AB} and K_{BA} for $A \rightarrow B$ and $B \rightarrow A$ communication, respectively. A 32-bit counter

[2]A Bloom filter is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set. False positives are possible, but false negatives are not.

that is increased for each new message is assigned to each key to guarantee semantic security. Counter C_{AB} is used for key K_{AB} and counter C_{BA} for key K_{BA}. Only the last x bits of the counter value are included in each packet to save the energy of transmitting more bits. Both sender and receiver keep track of the counters which have to be synchronized on both sides. The receiver can accept only messages with a counter value greater than this in the previous messages. However the counters can be desynchronized and a counter resynchronization protocol is needed.

Unless it is known before deployment which pairs of nodes are going to use unicast communication, each node in the network should maintain a counter for each possible sender (i.e. its neighbors), resulting in high memory overhead and making counter resynchronization very expensive. These problems also dictate the use of a different mechanism for the broadcast case.

2.4.2. MiniSec-B

Two mechanisms are used in MiniSec-B to provide semantic security and replay protection. The first one requires time synchronization among the nodes and divides time in epochs $E_1, E_2, E_3, \ldots$. The number of the current epoch is used as the nonce for OCB-encryption. When a node receives a packet, it attempts decryption twice; one with the current epoch number and one with the immediately previous epoch number. The epoch length is defined in a way that compensates for time synchronization errors and network latency.

The second mechanism defends against replay attacks within the current epoch. Each sender nodes keeps a counter which is incremented for each new message. At the end of each epoch the counter is reset, which means it can be shorter than the counter in MiniSec-U (the authors found that it was sufficient to use an 8-bit counter). The receiver keeps two alternating Bloom filters, one for the current epoch and one for the previous epoch. Each time it receives a packet it queries the corresponding Bloom filter and if the query returns true, the packet is considered to be a replay. The problem, however, is that the Bloom filters may cause false positives, causing a legitimate packet to be rejected as a replayed packet.

2.4.3. MiniSec packet format

Figure 5 shows the packet formats for MiniSec-U and MiniSec-B compared to the TinyOS packet format for the CC2420 radio (compliant with IEEE 802.15.4). Like in TinySec, the Group ID has been removed from the header, since access control is achieved through the use of different cryptographic keys. The 2-byte CRC is replaced by a 4-byte MIC (Message Integrity Code). The difference between MiniSec-U and MiniSec-B is that for the unicast mode, only $x = 3$ bits of the counter are sent in the packet header, while for the broadcast mode the whole counter has to be sent.

2.5. SecureSense

SecureSense [21] provides dynamic security service composition using the TinySec infrastructure. It introduces a new 1-byte field in its packet format called SCID as an indicator of the services provided by the message. The values of the bits in this field determine the combination of services provided, like confidentiality, integrity, semantic security and replay protection.

(a) TinyOS packet format for CC2420 radio

(b) MiniSec-U packet format

(c) MiniSec-B packet format

Figure 5. MiniSec packet format in the unicast and broadcast modes.

SCID has replaced the TinyOS active message type (AM) field in order not to increase the packet length and consequently, the packet transmission time. However, the removal of active message type field introduces several major problems for upper layer services as it directly affects the *Active Message Model* of TinyOS. According to this model, each packet on the network specifies a handler ID that will be invoked on recipient nodes. When a message is received, the receive event associated with this ID is signalled. This mechanism allows different network protocols to operate concurrently without conflict. Removing the AM ID, therefore, significantly affects implementation of such protocols under TinyOS and introduces new complexities.

2.6. ZigBee

Security services provided for ZigBee include methods for key establishment, key transport, frame protection, and device management[22]. To secure messages transmitted over a single hop, ZigBee uses MAC layer security, provided by the IEEE 802.15.4 standard, while for multi-hop messages it relies on higher layer security (i.e. the network layer).

The ZigBee specifications provide different means to achieve the following security requirements:

- *Authentication*: Network level authentication is achieved by using a common network key. Authentication of messages exchanged between two nodes is achieved by using unique link keys shared by these nodes. This prevents insider and outsider attacks but requires high memory usage.
- *Encryption*: ZigBee uses 128-bit AES encryption. Encryption protection is possible at network level or link level. As some applications may not need any encryption, encryption can be turned off without impacting freshness, integrity, or authentication.

- *Freshness*: ZigBee nodes maintain incoming and outgoing freshness counters to maintain data freshness. These counters are reset every time a new key is created. Devices that communicate once per second will not overflow their freshness counters for 136 years.
- *Message Integrity*: ZigBee specifications provide options of providing 0, 32, 64 or 128 bit data integrity for the transmitted messages. The default is 64 bit integrity.

Encryption at the MAC layer is done using AES in Counter (CTR) mode and integrity is done using AES in Cipher Block Chaining (CBC- MAC) mode [16]. A combination of encryption and integrity is done using a mixture of CTR and CBC- MAC modes called the CCM mode. Encryption at the network layer is also done using AES. However, in this case the security suites are all based on the CCM* mode of operation. The CCM* mode of operation is a minor modification of the CCM mode used by the MAC layer. It includes all of the capabilities of CCM and additionally offers encryption-only and integrity-only capabilities. These extra capabilities simplify the network layer security by eliminating the need for CTR and CBC-MAC modes.

3. Building Transparent Security Services

In the previous section we reviewed some of the most known link-layered security frameworks for sensor networks. In the remaining sections we move to describing in more details the issues involved in designing such a framework. In particular, we are going to emphasize in the design and implementation of the following two main mechanisms necessary to achieve the various security requirements:

- Key management techniques that look into the different ways to establish and distribute cryptographic keys among the different nodes in the sensor network, and
- Mechanisms used to encrypt the important data (to provide data confidentiality) and to calculate the MAC (to provide data authenticity and data integrity) using the established cryptographic keys.

So, first we begin by describing a key management scheme that will provide the necessary keys for our security framework (called L^3Sec, for Lightweight Link Layer Security, first presented in [23]), and then in Section 3.2 we complete it by showing how it provides secure services to the higher levels.

3.1. Key Management Module

In the bibliography there are three major approaches for key management in WSN:

- *Deterministic pre-assignment*. Examples of this approach are SPINS[20] and LEAP[24] in which unique symmetric keys shared by the nodes with the base station are assigned before the network is deployed. Using this approach, cryptographically strong keys can be generated, however, this involves a significant pre-deployment overhead and is not scalable.
- *Random pre-distribution*. Schemes like those in [25], [26] and [27] and PIKE[28] refer to probabilistically establishing pair-wise keys between neighboring nodes

in the network. Usually in this approach a random subset of keys from a key pool is pre-assigned to every node; two nodes establish a pair-wise key based on the subset of the shared keys between them. This framework is quite flexible; the choice of protocol parameters determines the trade-off between scalability and resiliency to node capture. However, most of the key pre-distribution schemes rely on sensor nodes to broadcast a large number of pre-loaded key IDs to find pair-wise keys between neighboring nodes, thus leading to a huge communication overhead. In addition, to guarantee network connectivity, each node has to store several hundreds keys or key spaces, which may greatly decrease the memory availability.

- *Deterministic post-deployment derivation*. In this approach, nodes use some globally shared secret and pseudo-random number generators to derive the keys at runtime. LEAP[24] use this approach in order to establish pair-wise and group keys. A node erases the global secret after the completion of the initial key establishment phase to provide resilience against possible node compromises. However, most of the techniques based on this approach make it infeasible for even non compromised nodes to generate new keys at a future time making these protocols inefficient for dynamic sensor network topologies.

The key management scheme that we describe in this section follows the post-deployment approach with support for newly added nodes. It addresses flexibility and scalability issues and is resistant to node capture attacks. This module forms the core of the L^3Sec framework described in the next section, but it can also be used as a stand alone component or it can be easily integrated into other security protocols providing the related services to them.

A comparison of the above-mentioned key management schemes as well as the one presented here is given in Table 1. PW stands for pair-wise keys, G stands for a global key common among all nodes, NB is the node-base key common between each one of the nodes and the base station, and BC is the broadcast key of each node common between the node and its neighbors.

Table 1. Comparison of key establishment protocols in WSN

Protocol	SPINS	LEAP	PIKE	L^3Sec
Preloading Overhead	yes	yes	yes	no
Capture Resistance	no	partially	partially	yes
Scalability	no	no	yes	yes
Preloaded keys	NB	NB,G	PW	G
Key Types	PW,NB	PW,NB,G	PW	PW,NB,BC

3.1.1. The Key Establishment Protocol

Before the sensor nodes start establishing the keys, they need to run a *neighbor discovery* phase. This is achieved in two steps by a pair of handshake messages. In the first step, node i broadcasts a specific type of message containing its ID so that every other node in i's communication range (like j for example) can receive it. We refer to this message as a *ping message*. Every node receiving the ping message answers back to the sender (i) with a *pong message* containing its ID (steps 1 and 2 in Table 2). Node i can then

add j to its own neighbor list. After a sufficient amount of time (see Table 3 and more explanations in Section 3.1.2), i will discover all of its neighbors and this phase will be finished.

Next, we need to provide cryptographic keys in order to secure both one-to-one and one-to-many communication in a wireless sensor network. For this purpose the protocol establishes three different kinds of keys in each sensor node:

1. *Pair-wise (PW) key* that is established between two neighbors to protect their one-to-one communications.
2. *Broadcast (BC) key* that is established in order to secure the broadcast messages sent by a node to its neighbors.
3. *Node-Base (NB) key* that is established in order to secure the communication between a node and the base station (note that this communication is not necessarily direct). A message encrypted by this key, can only be decrypted by the base station.

Each node i computes its own node-base key and its pair-wise keys with its neighbors as well as their broadcast keys as follows:

$$NB_i = F(i||baseStationAddress||K)$$

$$PW_{i,j} = F(min(i,j)||max(i,j)||K)$$

$$BC_i = F(i||K)$$

where "||" is the concatenation operator and F is a secure pseudo-random function usually implemented by a hash function such as SHA-1 or MD5. K is a global master key that is distributed to all nodes before deployment of the network. As we will explain later, K will eventually be deleted from the memory of the nodes in order to make the scheme more secure against node capture attacks.

Table 2. Steps of Key Establishment Protocol

Step	Message
1	$i \rightarrow j : \{i\}$
2	$j \rightarrow i : \{j\}$
3	$i \rightarrow j : \{i, PW_{ij}, N_A\}_{NB_j}$
4	$i \rightarrow j : \{i, BC_i, N_B\}_{NB_j}$
5	$j \rightarrow i : \{j, N_A, N_B\}_{PW_{ij}}$
6	i Deletes master key K and node-base key of j

When these calculations are over, node i has a complete table of all the needed keys. However, it needs to announce its pair-wise and broadcast keys to its neighbors in order to communicate with them. In other words, a neighbor node j has to update its key table with the keys corresponding to node i. Thus, node i has to send a message M containing these keys to node j. Obviously, M should not be sent in plain. Therefore, node i should calculate an appropriate key in order to send an encrypted version of M to node j. A proper key, as we will see, is the node-base key of node j which can be derived by i as follows:

$$NB_j = F(j||baseStationAddress||K)$$

Having this key, node i can encrypt and send to j the key it shares with it ($PW_{i,j}$) as well as its own broadcast key (BC_i). The related messages are the following (Steps 3 and 4 in Table 2):

$$i \rightarrow j : \{i, PW_{i,j}, N_A\}_{NBj}$$

$$i \rightarrow j : \{i, BC_i, N_B\}_{NB_j}$$

where N_A and N_B are two nonces to guarantee the freshness of these messages[3].

After sending these two messages, node i will delete the node-base key of node j from its memory. Therefore the only non-base station node that can decrypt these messages is node j (note that we assume the base station is secure). Node i will also delete the master key K from its memory.

Upon receiving the keys, node j will answer back to node i by sending a message containing the nonces N_A and N_B. This message is encrypted with the pair-wise key of i and j (Step 5 in Table 2). At this point, key establishment is complete.

Notice how this message exchange enforces the *scalability* aspect of the protocol: related keys can be established when a new node is added to a previously deployed network. Any new node that joins the network (such as i) can initiate the key establishment phase by broadcasting a *ping message*. Following that, related keys are calculated by the new node. Then the broadcast key of this added node, as well as its pair-wise keys with each of its neighbors are sent to related neighbors, encrypted with their node-base keys.

Using the node-base keys for this purpose is quite an appropriate choice in order to make the protocol scalable and secure. This is because the already available network nodes have already deleted the master key K from their memory and consequently cannot use it to either calculate the keys or decrypt any message encrypted with it. It is not a good idea to use the broadcast key of previously joined neighbor nodes (like j) since other neighbors of j have that key available and can decrypt messages encrypted with it; a fact that results in providing a looser security scheme.

The deletion of master key K and the temporarily calculated node-base key of j (NB_j) by i makes the protocol resilient to node capture attacks. It reduces the negative consequences of capturing a node as the attacker will gain access only to that neighborhood and *not the entire network*. Since the needed time for key establishment is negligible, we can assume that the adversary does not have enough time to find the master key K before it is deleted from the memory of the nodes (see also LEAP[24] for a similar assumption).

Moreover, newly joined nodes must come with the master key K in order to calculate the cryptographic keys. Therefore, an adversary cannot introduce new nodes to the network, since she doesn't know K. In addition to that, it is important to note that if one of the above mentioned messages in key establishment protocol is not delivered,

[3]The reason that message *M* is broken into two consecutive messages is only a practical nuance. The overall size of M – combination of the two mentioned messages – which would be 32 bytes (node ID is 2 bytes and the pair-wise key, the broadcast key and the nonce are 10 bytes each) is larger than the maximum allowed message size in TinyOS which is 29 bytes. Hence, we were forced to break M into two different messages. However, if keys are 8 bytes long then these two messages can be merged to one.

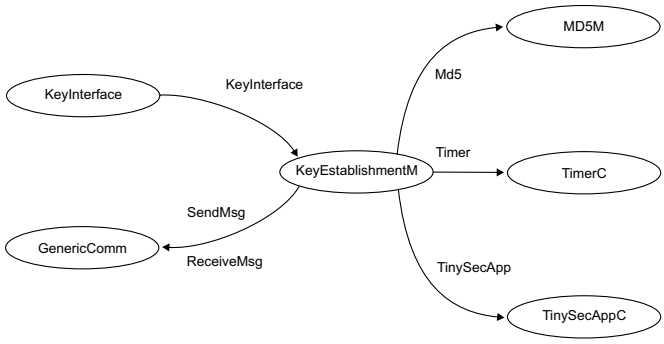


Figure 6. Key establishment module architecture.

the receiving node will not get stuck. If node i does not receive the last message of the protocol (Step 5 in Table 2), it will not add any entry for node j in its key table.

3.1.2. Implementation and Performance

We implemented the key management module described above in TinyOS in order to study its time and memory overhead and energy consumption. Figure 6 depicts the components of the key establishment module and how they are wired together.

As a stand alone library, this module implements an interface *KeyInterface* which contains three commands and three events. Command $init(k)$ initializes the key establishment module. When all of the keys are established among current nodes, the user will be notified by an *initDone*() event. Other commands provide security services such as encryption, decryption and MAC computation. The higher level application can use these services to design and implement its own security scheme. The complete solution discussed in Section 3.2 provides even more transparent security services embedded in implemented interfaces, *Send*() and *Receive*(). The MD5 module is used as pseudo-random function that is needed to establish the different type of keys.

The memory overhead of the key management module for each node can be calculated as follows:

$$Overhead\ M = [(|BC| + |PW|) * d] + |NB|,$$

where $|BC|$, $|PW|$ and $|NB|$ are the size of broadcast key, pair-wise key and node-base key respectively and d stands for the maximum number of neighbors each node may have[4]. The default size of all types of keys in the key establishment module is 10 bytes, which provides strong security (2^{80} bit key space) for sensor network applications. As a result, in a very dense network where $d = 50$, we will have $M \approx 1KB$. Although this value of d is far more than enough to keep the network connected, this memory overhead is well within the memory capabilities of motes (MICA2 motes have 4KB of RAM).

During the key establishment phase, prior to deletion of the master key, an adversary has a chance to find it and use it to derive all the other keys. However, this time is so

[4]In our current implementation of neighbor discovery phase, a node willing to discover its neighbors, broadcasts a ping message and waits for t milliseconds to receive pong messages from the potential neighbors. Yet it discards pong messages if they arrive after t milliseconds or if the number of discovered neighbors is already d. Values of d and t are decided during deployment time and play important roles in network connectivity.

Table 3. Required time and energy before the global key deletion

Phase	Neighbor discovery	Key computation	Key Sending
Time	1000ms	10ms	10ms
Energy	1592640nJ	157nJ	38049000nJ

small that the probability that the attacker will capture a mote and retrieve the key is considerably small. Table 3 shows the related duration (calculated with Tossim) that it takes to delete the master key from memory of a newly added mote during its initialization phase[5].

Table 3 also presents the estimated amount of energy consumption for each phase of the key establishment for the same network ($d = 50$). This estimation was calculated by multiplying the total amount of communications by an average communications cost of 18 $\mu J/bit$ (see PIKE[28] for a similar assumption). As a result, the estimated energy consumption of the key management scheme presented here is approximately $0.4J$ (note that as presented in Table 3 the energy needed for key computation is quite smaller than the needed energy for communication) comparing to PIKE-2D that is more than $8J$ or PIKE-3D[28] which is around $6J$. This high energy efficiency of the L^3Sec's key establishment scheme comes with a comparable cost in terms of memory overhead; it uses about 1000 bytes of memory to establish and manage the keys while PIKE-2D and PIKE-3D need around 600 bytes and 500 bytes respectively.

In the L^3Sec's key establishment scheme the effect of having a node captured is reduced to its neighborhood, i.e. the captured node's pairwise keys with its neighbors, its broadcast key and its node-base key are the only keys that can be retrieved by the adversary. This is a small fraction of the established keys and communication still remains secure in the rest of the network.

3.2. A Link-layer Security Framework

Having built a key management scheme to use as the base, we now present the L^3Sec link-layer security framework, first proposed in [23] and compare it with other similar protocols that we reviewed in Section 2. The main goal of this framework is to be transparent and easy to use. More specifically, it features the following properties:

- The process of key establishment as well as related computations regarding the provision of security services such as confidentiality and authentication is completely hidden from the protocols in the upper layers.
- It is flexible so that the developers can adapt it to the security needs of their applications. This is an important requirement, especially in resource constrained systems such as sensor networks. As different messages being exchanged in the network require different security services, a security platform has to be flexible enough to address all the security needs of different types of communications while not imposing extra overhead due to redundancies.

Additional to these features, the security framework satisfies the properties that we described in Section 1, namely data confidentiality, data authentication, semantic security

[5]The higher the value of t, the higher the time prior to the deletion of master key. The current value of $t = 1000ms$ as appears in Table 3 is quite appropriate for a very dense network where $d = 50$ and all of the nodes are supposed to be able to discover all of their potential neighbors.

and freshness. Next we describe what approaches have been taken for each of these properties.

3.2.1. Data Confidentiality

In order to protect the messages being exchanged among the nodes from eavesdropping by unauthorized parties, appropriate encryption mechanisms are provided. The default cipher that is used in the implementation of this framework for this purpose is *SkipJack*, however, the platform is not bound to use any specific cipher and related settings can be changed easily by the platform user (the other currently available cipher in the implementation is RC5).

3.2.2. Data Authentication

Proper message authentication codes (MAC) are used to allow nodes detect any modifications in received messages. The MAC generation is performed by applying a pseudo-random function (implemented by a hash function) to the concatenation of the message and the related established key. The default hash function that is used in the implementation of L^3Sec is MD5. However, related settings can be easily changed to replace it with any other hash function, such as SHA-1.

Since the already established keys (e.g. pair-wise keys established among the neighbors) are used to generate the message authentication codes, network nodes are able to verify the authenticity of the received messages. Using this service, unauthorized nodes will not be able to send legitimate messages into the network. Thus, an access control service is also provided using generated MACs.

3.2.3. Freshness and Semantic Security

Common defences to protect a network from message replay attacks is to either timestamp the messages using some network time synchronization protocol or include a monotonically increasing counter in related messages in order to be able to detect replayed old messages and reject them.

Providing time synchronization in WSN is usually quite complicated. Moreover, doing it in a secure manner is even more demanding and has to rely on a security platform like the one we describe here. As a result, in the implementation of L^3Sec the increasing counter approach is used to guarantee the freshness of the messages. However, the memory cost of having every recipient maintaining a table of the last received counter value of all of the other nodes is quite high. This is the reason why TinySec, as pointed out by the authors of its paper, does not provide acceptable security against replay attacks.

As a matter of fact, a complete provision of such a service in link layer is not practically feasible. This is because information about the network's topology and communication patterns seem to be mandatory to provide protection against replay attacks, yet this information is not available in the link layer. However, having neighbor information available at the end of the neighbor discovery phase, L^3Sec lets each sensor maintain only the last counter value for its *neighbors* and not for all nodes. Thus, as a message is being routed to a specific destination, the counter value is updated on each hop.

The amount of memory needed to keep the neighbors' counter value is quite small (for example, if each counter requires 4 bytes, the total amount of memory needed to keep the counter values in a *very dense* network where each node has 20 neighbors will be 80 bytes).

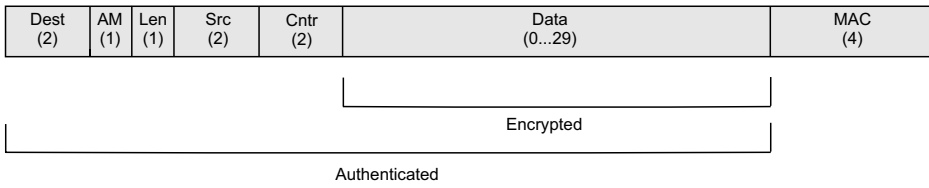


Figure 7. L^3Sec packet format for full security service.

3.2.4. Run-time Composition of Security Services

An approach similar to the one used in SenSec is not appropriate to be used in a general security framework for WSN, since it provides the highest possible degree of security for all of the exchanged messages. A more flexible scheme is provided in L^3Sec by using the most significant *three* bits of the data length field in the packet format as an indicator of the security service(s) to be used. These three bits are never used by TinyOS, as the maximum data length in TinyOS is chosen to be 29 bytes (see also TinySec for a similar approach). Consequently, the provision of this feature comes with no overhead.

Being motivated by TinySec, L^3Sec provides run-time composition of security services *without* removing the AM ID or adding extra fields to support integration of services. Each one of the higher three bits of data field of the packet stands for a security service (from higher order bit to the lower: Replay Attack Protection, Access Control and Integrity, Confidentiality) and setting *any* bit means that the related service is provided for that packet. Thus the desired services for different packets can be composed at runtime.

3.2.5. Protection against Node Capture Attacks

L^3Sec is the first link-layer security framework providing acceptable resistance against node capture attacks. This feature minimizes the effects of compromising a node to the neighborhood of that node, keeping the rest of the network in a secure state. No assumption about tamper resistance is made. While tamper resistance might be an effective solution for node capture attacks, it is considered noticeably expensive for the sensor nodes, which are intended to have low cost.

3.3. Packet Format

We conclude the description of L^3Sec by showing how the TinyOS packets should be modified to support the properties that we mentioned above. Figure 7 shows the fields included in the packet format in the full security mode.

The *Source* field of the packet is used to find the appropriate established pair-wise or broadcast key needed for the security services. Note that TinySec does not use the *Source* field when it is set to authentication only (TinySec-Auth) mode. This is because it assumes that if the attached MAC of a received message is valid then it comes from an authorized source (note that in TinySec the MAC is derived using a specific *global* key shared among all valid nodes, a bad security practice as we explained in Section 1). However, this assumption is not necessary in L^3Sec, which uses established keys in order to resist against node capture attacks. As a result, we must include the *Source* field in the packet format.

Table 4. Operational modes and related settings.

Mode	SetBits	Omitted Fields	Omitted Operations
"RAC"	111	-	-
"RA"	110	-	Encryption
"RC"	101	MAC	MAC
"R"	100	MAC	MAC & Encryption
"AC"	011	-	Counter Saving
"A"	010	Counter	Counter Saving& Encryption
"C"	001	MAC	Counter Saving & MAC
"-"	000	All Security Fields	All Security Operations

In other related service modes, such as replay attack protection mode, the packet format contains a counter (*Counter*). Together with the *Source* field, this 2 bytes long counter can be used to avoid *IV* reuse in *CBC* encryption mode. In L^3Sec, similar to TinySec, the *IV* includes the destination address, the active message (AM) type, the data length, the source address and the counter. The $Source||Counter$ format guarantees that each node can send 2^{16} messages with the same AM type and the same destination, but with different IV values. As mentioned, another application of the counter value is its role in providing resistance against replay attacks.

Finally, the most significant *three* bits of the data length field indicate the different major security modes that are provided. These include

1. Authentication, Access Control and Integrity (A). In this mode the *Counter* field is not required, but obviously the *MAC* field is needed.
2. Confidentiality (C). In this mode the *Source* and *Counter* fields are used in the packet format, however receiver nodes do not save the related counter values.
3. Replay Attack Protection (R). *Source* and *Counter* fields are also necessary in this mode, but the counter value of each neighbor is kept.

As we mentioned earlier, these modes can be combined in any variation setting the corresponding bits. Table 4 shows in more detail the different modes, provided services and the corresponding bit values.

Conclusion

Reviewing the proposed link-layer security frameworks for sensor networks we saw that all of them succeed in providing data confidentiality, data authentication and semantic security following some common approaches. However, what is missing is the provision of some features that would make them more secure and practical to use. These features include the resilience to node capture attack and scalability, both of which require that the framework is tied to an appropriate key management protocol. Another important feature is flexibility, which allows different types of security services for different types of communications among nodes. We presented a security framework, namely L^3Sec, which incorporates these features while at the same time it remains energy efficient and easy to use. Looking into this framework we described how the underlying key management scheme should be used and how the TinyOS packet structure should be changed for such a scheme to provide a complete range of security services and satisfy the aforementioned requirements.

References

[1] T. Li, "Security map of sensor network," Infocomm Security Department, Institute for Infocomm Research, Tech. Rep., 2005.

[2] S. Camtepe and B. Yener, "Key distribution mechanisms for wireless sensor networks: a survey," Rensselaer Polytechnic Institute, Troy, New York, Technical Report 05-07, March 2005.

[3] E. Shi and A. Perrig, "Designing secure sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 38–43, December 2004.

[4] L. Lazos and R. Poovendran, "SeRLoc: Robust localization for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 1, no. 1, pp. 73–100, 2005.

[5] T. Dimitriou and I. Krontiris, *Security in Sensor Networks*. CRC Press, 2006, ch. Secure In-network Processing in Sensor Networks, pp. 275–290.

[6] S. Ganeriwal, S. Capkun, C.-C. Han, and M. Srivastava, "Secure time synchronization service for sensor networks," in *Proceedings of the 4th ACM workshop on Wireless security (WiSe '05)*, 2005, pp. 97–106.

[7] T. Ylonen, "SSH - secure login connections over the internet," vol. Proceedings of the 6th Security Symposium) (USENIX Association: Berkeley, CA), p. 37, 1996.

[8] "OpenSSL," Tech. Rep., 2001, available from http://www.openssl.org.

[9] S. Kent and R. Atkinson, "Security architecture for the internet protocol," Internet Eng. Task Force RFC 2401, November 1998.

[10] V. Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, H. Eberle, and S. C. Shantz, "Sizzle: A standards-based end-to-end security architecture for the embedded internet," in *PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, 2005, pp. 247–256.

[11] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *Proceedings of ASPLOS-IX*, 2000, pp. 93–104.

[12] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, 2005, pp. 324–328.

[13] E.-O. Blaß and M. Zitterbart, "Towards acceptable public-key encryption in sensor networks." in *ACM 2nd International Workshop on Ubiquitous Computing*, May 2005, pp. 88–93.

[14] K. Piotrowski, P. Langendoerfer, and S. Peter, "How public key cryptography influences wireless sensor node lifetime," in *SASN '06: Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, 2006, pp. 169–176.

[15] L. Batina, N. Mentens, K. Sakiyama, B. Preneel, and I. Verbauwhede, "Low-cost elliptic curve cryptography for wireless sensor networks," in *ESAS 2006: Proceeding of the Third European Workshop on Security and Privacy in Ad hoc and Sensor Networks*, Hamburg, Germany, September 2006, pp. 6–17.

[16] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiu, "Analyzing and modeling encryption overhead for sensor network nodes," in *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, 2003, pp. 151–159.

[17] M. Bellare, J. Kilian, and P. Rogaway, "The security of the cipher block chaining message authentication code," *Journal of Computer and System Sciences*, vol. 61, no. 3, pp. 362–399, 2000.

[18] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A link layer security architecture for wireless sensor networks," in *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004, pp. 162–175.

[19] C. Hartung, J. Balasalle, and R. Han, "Node compromise in sensor networks: The need for secure systems," Department of Computer Science, University of Colorado, Technical Report CU-CS-990-05, January 2005.

[20] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[21] Q. Xue and A. Ganz, "Runtime security composition for sensor networks (SecureSense)," in *IEEE Vehicular Technology Conference (VTC Fall 2003)*, October 2003.

[22] Z. Alliance, "Zigbee specification," Document 053474r13, December 2006.

[23] H. Soroush, M. Salajegheh, and T. Dimitriou, "Providing transparent security services to sensor networks," in *ICC'07: Proceedings of the IEEE International Conference on Communications*, Glasgow, Scotland, June 2007.

[24] S. Zhu, S. Setia, and S. Jajodia, "Leap: efficient security mechanisms for large-scale distributed sensor networks," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, 2003, pp. 62–72.

[25] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, 2002, pp. 41–47.

[26] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, 2003, p. 197.

[27] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, 2003, pp. 52–61.

[28] H. Chan and A. Perrig, "PIKE: Peer intermediaries for key establishment in sensor networks," in *Proceeding of the 24th Conference of the IEEE Communications Society (Infocom)*, 2005.

Wireless Sensor Network Security
J. Lopez and J. Zhou (Eds.)
IOS Press, 2008

Secure Routing in Wireless Sensor Networks

Gergely Ács and Levente Buttyán
Laboratory of Cryptography and Systems Security (CrySyS)
Department of Telecommunications
Budapest University of Technology and Economics, Hungary
{acs, buttyan}@crysys.hu

Abstract. In this chapter, we study how sensor network routing protocols can be secured. First, we describe the adversary model, the objectives of attacks against routing, as well as the different attack methods that may be used in wireless sensor networks. All these are illustrated by example attacks on well-known sensor network routing protocols. Then, we describe various countermeasures that can be used in sensor networks to secure the routing protocols. These include link layer security measures, secure neighbor discovery techniques, authenticated broadcast algorithms, and multi-path routing techniques. Finally, we illustrate the application of some of these countermeasures by presenting and explaining the operation of some secured sensor network routing protocols.

1. Introduction

Routing is a pivotal element of network communications. It is concerned with ensuring the delivery of messages from a source to some destinations. This involves two functions: (1) the discovery of routes from the source to the destinations, and (2) the forwarding of the messages via the discovered routes. Both functions need to be secured in order to ensure the proper operation of the network.

In traditional networks, the routing functions are performed by special nodes, called *routers*, which are often physically protected. In contrast to this, in wireless sensor networks, the sensor nodes themselves perform the routing functions. However, due to the facts that sensor networks often operate in unattended environments and the sensor nodes are usually not tamper resistant, the sensor nodes can be easily compromised. Thus, the nodes cannot be assumed to execute the routing protocol faithfully, which makes the provision of security for routing a challenging task.

Besides ensuring the delivery of messages, routing protocols for sensor networks have additional objectives. In particular, some protocols are concerned with real-time requirements and aim at minimizing the message delivery time, while others try to maximize the lifetime of the network by minimizing and balancing the energy consumption of the nodes. These objectives can be satisfied by employing different routing techniques. For instance, energy consumption can be decreased by using hierarchical routing approaches or simply taking into account some energy-aware metrics during next-hop decisions. Moreover, these objectives can also be conflicting. For instance, always using the

shortest path towards the base station to minimize network delay can also yield decreased network lifetime, because frequently using the same path towards the base station can quickly deplete the battery of the nodes comprising that path [33].

The different objectives and application environments of sensor networks resulted in a wide spectrum of sensor network routing protocols (see e.g., [2] for a overview). These protocols can be classified in many different ways. In this chapter, we will use the following simple classification that suits our purposes:

- *Topology-based routing protocols*: These protocols typically build a routing topology during the route discovery process that is used later for data forwarding towards the base station. Topology-based protocols can be
  - hierarchical (e.g., LEACH [19], TEEN [37], APTEEN [38]);
  - distance vector based (e.g., TinyOS beaconing [20]);
  - link-state protocols (e.g., INSENS [12]); or
  - data-centric (e.g., Directed Diffusion [23]).

  In hierarchical protocols, the nodes form clusters, they elect a cluster leader, and forward data packets to the cluster leader, which then passes further the packets directly to the base station or to other, higher level cluster leaders.
  Distance vector protocols select the next hop towards the base station based on some distance-like routing metric. In TinyOS beaconing, for instance, a beacon message originating from the base station is flooded in the network, and each node chooses the node from which it first received the beacon as the next hop towards the base station. Thus, the time needed for the beacon to reach a node is used as the metric.
  Link-state protocols for wireless sensor networks are often centralized, where the nodes send their link-state information to the base station, and based on these link-state information, the base station reconstructs the topology of the entire network and computes the routing tables for every node. The routing tables are then distributed to the nodes. The main drawback of this approach is that it does not scale well, and therefore, it cannot be applied in large networks.
  Finally, in the case of data-centric routing protocols, the next hop towards the base station is selected based on the content of the data packets. The advantage of these protocols is that the nodes do not need globally unique addresses, as routing decisions are not based on addressing information.
- *Location-based routing protocols*: These protocols (e.g., GPSR [26], GOAFR [30]) are also called position-based or geographic routing protocols. Here each node forwards a packet based on the location of the destination, which is carried by the packet, and the locations of the forwarding node's neighbors. These protocols are often considered stateless, because the nodes do not need to store any additional routing information besides the locations of their neighbors. As a consequence, location-based routing protocols are mainly concerned with the message forwarding function of routing, and the discovery function is reduced to neighbor discovery instead of route discovery.
- *Hybrid protocols*: Hybrid protocols use both geographic and topological information to forward data packets (i.e., sensor nodes maintain some additional routing information besides the locations of their neighbors). These protocols are typi-

cally designed to incorporate energy-awareness in the simple forwarding process of geographic routing approaches (e.g., GEAR [64], Energy Aware Routing [52]).

Wireless sensor networks are often related to ad hoc networks, but there are important differences between these two types of networks with respect to routing. First of all, sensor nodes are more resource constrained devices than the nodes of an ad hoc network, which can be powerful laptop class computers and PDAs. Therefore, energy efficient design is even more important for sensor networks than for ad hoc networks. Second, in sensor networks, there is a special entity, the base station, whereas ad hoc networks are usually homogeneous, where all nodes have the same role. Third, sensor networks have more specialized communication patterns. In particular, communications can be many-to-one communication (a.k.a., reverse-multicast or convergecast) between the sensors and the base station and one-to-many communication (multicast or broadcast) between the base station and the sensor nodes. In contrast to this, ad hoc networks typically support routing between any pair of nodes. Due to these differences, secure routing protocols proposed for ad hoc networks cannot be directly applied to wireless sensor networks.

In the rest of this chapter, we study how sensor network routing protocols can be secured. In the following section, we describe the adversary model, the objectives of attacks against routing, as well as the different attack methods that may be used in wireless sensor networks. All these are illustrated by example attacks on well-known sensor network routing protocols, such as TinyOS beaconing, Directed Diffusion, and GPSR. In Section 3, we describe various countermeasures that can be used in sensor networks to secure the routing protocols. These include link layer security measures, secure neighbor discovery techniques, authenticated broadcast algorithms, and multi-path routing techniques. Finally, in Section 4, we illustrate the application of some of these countermeasures by presenting and explaining the operation of some secured sensor network routing protocols. In particular, we describe the operation of an authenticated variant of TinyOS beaconing, a secure link-state protocol called INSENS, a family of secure position based routing protocols called SIGF, and a secured variant of the Directed Diffusion protocol. Note that these protocols belong to different classes according to the classification presented above; hence, we provide examples of securing different protocol classes.

2. Attacks on Sensor Network Routing

2.1. Adversary model

The adversary can mount her attacks from sensor-class devices and more powerful laptop-class devices. It is quite reasonable to assume that both sensor-class and laptop-class devices can be easily acquired by the adversary, or alternatively, she can capture honest sensor-class devices directly in the network field. Of course, capturing is viable only if sensor nodes are not tamper resistant devices and the adversary can gain unsupervised access to them. The sensor-class devices and laptop-class devices that are under the control of the adversary are further called adversarial nodes. Sensor-class devices have identical capabilities to an ordinary sensor node (i.e., their energy supply as well as their computational power is typically heavily constrained). On the contrary, laptop-class devices are more powerful; besides having unconstrained energy supply and computa-

tional capability, they also have powerful transmitters with much greater power range than sensor nodes have. Additionally, laptop-class devices may also have more sensitive receivers, though such equipment bears much higher costs than transmitters.

All attacks performed by the adversary consist of simple message manipulations like message injection, deletion, modification, re-ordering, and simply relaying messages without following the routing protocol rules faithfully. Before investigating more sophisticated attacks that employ the previously listed message manipulations, we describe how the adversary can perform such message manipulations.

Injection of messages in a radio channel is trivial. Message deletion can also be easily done by simply not forwarding a message according to the protocol rules, or by performing jamming [61]. Message modifications and re-ordering can be performed in a straightforward way if the adversary acts as a relay node between the sender and the receiver (i.e., the sender and the receiver cannot reach each other directly). However, if the receiver and the sender can communicate directly, then the adversary must use sophisticated jamming techniques that prevents the receiver from receiving messages, while at the same time allows the adversary to receive those messages. Once a message is deleted in this way, the adversary can modify it and send the modified message to the receiver. Specific scenarios for this case are detailed in [1].

Subsequently, we distinguish two types of adversaries. An *outsider adversary* is assumed to be able to manipulate messages sent by honest nodes, however, she cannot control legitimate sensor nodes. On the contrary, an *insider adversary* has all the power as the outsider adversary, and additionally, she is able to control some legitimate sensor nodes in the routing process (this may also mean the compromise of the cryptographic keys of those sensor nodes).

2.2. *Objectives of attacks*

Generally speaking, the adversary primarily intends to thwart the objectives of routing protocols. More specifically, she wants to degrade the performance of routing, or ultimately, she may attempt to completely disrupt the routing service and cause network malfunctioning. Degrading the performance of routing can mean degrading the packet delivery ratio, shortening the network lifetime, and/or increasing the network delay. In addition to these, the objective of attacking the routing protocol can be to increase the hostile control over the traffic. Note that some of these adversarial goals are highly correlated (e.g., if the adversary can successfully divert the traffic through adversarial nodes, then she can easily degrade the packet delivery ratio or increase the network delay by dropping packets and delaying packet forwarding).

2.3. *Attack methods*

In the previous subsections, we described the capabilities of the adversary in sensor networks and the general objectives of the attacks launched by this adversary. Now, we give an overview of the specific attack methods that can be used by the adversary in order to achieve her objectives.

The simplest attack methods are composed of the basic message manipulations techniques. This includes *dropping*, *modification*, *delaying*, *injection* and *re-ordering* of routing control messages. In order to further refine these methods, we separate the route dis-

covery and the data forwarding phase of routing protocols. In both the route discovery and data forwarding processes, the adversary can inject extra packets in order to consume valuable network resources at honest sensor nodes (leading to denial-of-service). In the route discovery phase, injecting a forged control packet can result in corrupt routing states at honest nodes that may ultimately yield increased traffic control as well as shortened network lifetime and increased network delay. Dropping control packets have trivial effects: in this way, the adversary can separate some of the nodes from the base station that can only reach the base station through an adversarial node. The adversary can also degrade the packet delivery ratio and the network delay by dropping packets in the data forwarding process. By modifying control packets, the adversary can cause honest nodes to store corrupt routing states, which may have similar effects to injecting forged control packets. Furthermore, the adversary can also modify data packets, which may lead to re-transmissions, and hence increased energy consumption. Re-ordering and delaying of control packets can influence the next-hop selection mechanism. We note that while topology-based and link-state routing protocols are usually vulnerable to control packet manipulation, geographic and hybrid routing protocols seem to be more resistant against these attacks.

Besides these basic packet manipulation attacks, the adversary may also be capable to mount attacks at higher level. These are *tunneling*, *rushing*, *selective forwarding*, and *replay* attacks.

In the tunneling attack, the adversary controls some corrupted nodes in the network, and tunnels routing control messages between these controlled nodes in the payload part of normal data packets using the multi-hop forwarding mechanism of the network. In this way, the adversary can make some routes appear shorter than they really are, and thus, these routes may be preferred by the other nodes.

Rushing is a protocol-dependant attack. In particular, the adversary can launch this attack only against routing protocols that employ a duplicate suppression technique to control flooding. When using duplicate suppression, a node only considers the first copy of a given control packet and drops any further copies. For instance, a node A running tinyOS beaconing (as it will be described in Subsection 2.4) sets the neighboring node from which it received the first copy of the beacon as the next hop towards the base station. Any further beacons are simply discarded by A. The adversary employing rushing can exploit this duplicate suppression technique to divert the traffic: The adversary forges a beacon and broadcasts that to node A. As a result, A will set the identifier found in this forged beacon as the next-hop towards the base station. Later, when A receives the real beacon, it discards it due to duplicate suppression. In this way, the adversary can divert traffic to itself and increase hostile traffic control. This can be the first step of further severe attacks.

Selective forwarding refers to the capability of dropping data packets in the data forwarding process in a selective manner. Generally, this attack can be a second step after a successful tunneling or rushing attack. If the adversary jointly uses selective forwarding with any route diversion techniques, then she can easily setup a *blackhole* (where all packets are dropped) or a *grayhole* (where only specific packets are dropped).

Replay attacks can occur during topology construction as well as during the data forwarding process. The adversary can replay obsolete routing control packets that no longer reflect the current network topology, which may yield inefficient routing paths.

In the data forwarding process, replaying obsolete data packets causes incorrect reports about the monitored environment.

Finally, there are other attacks that are mainly related to neighbor discovery, such as the *wormhole attack*, the *Sybil attack*, and the *node replication attack*. We consider these attacks against routing, since many sensor network routing protocols integrate neighbor discovery as part of a cross-layer design.

A wormhole is an out-of-band connection, controlled by the adversary, between two physical locations in the network. The adversary installs radio transceivers at both ends of the wormhole, and it transfers packets (possibly selectively) received from the network at one end of the wormhole to the other end via the out-of-band connection, and re-injects the packets there into the network.

The effect of a wormhole on neighbor discovery is that some nodes that would not be neighbors otherwise may establish a neighbor relationship. This has a direct effect on route discovery mechanisms that operate on the connectivity graph, since they may identify routes that use virtual links created by the adversary. Thus, a well placed wormhole gives considerable power to the adversary, who can monitor the network traffic flowing through the wormhole, or mount a denial-of-service attack by permanently or selectively dropping data packets sent via the wormhole.

Some routing protocols do not rely on explicit neighbor discovery mechanisms, but the nodes discover their neighbors implicitly via processing the overheard routing control messages. Many of these protocols are equally vulnerable to the wormhole attack. For instance, an adversary can use a wormhole to mount a rushing attack against routing protocols based on flooding a route request and controlling the flood with duplicate suppression.

The wormhole attack has similar effects on routing protocols than the tunneling attack, but it is based on slightly different assumptions about the adversary. In particular, in the tunnelling attack, the adversary controls some corrupted nodes in the network, and tunnels routing messages between these controlled nodes in the payload part of normal data packets using the multi-hop forwarding mechanism of the network. Therefore, by definition, in order to mount a tunneling attack, the adversary needs to have corrupted nodes in the network, which use (possibly compromised) identifiers. In contrast to this, the adversary can mount a wormhole attack without corrupting any nodes or compromising any node identifiers, because the wormhole uses only low level repeaters transparent to higher layer protocols.

In a Sybil attack [13], a single adversarial node illegitimately uses *multiple* identities during the routing process. This can have devastating effects on multipath routing protocols [25], because a node may believe that it routes packets via node disjoint paths, while in reality these paths may all go through the adversarial node implementing the Sybil attack. Sybil attacks employed together with tunneling or wormhole attacks can be even more powerful, as the tunnels and the wormholes can be used by the adversarial nodes to share their invented identities.

The node replication attack is the dual of the Sybil attack, where the adversary uses the *same* identity for multiple devices, and thus a single adversarial node may be virtually represented in multiple locations in the network. Replication attacks can also severely influence the operation of most routing protocols; in the worst case, the adversary can copy the identity of the base station and use it in different locations of the network. If

the adversary manages to impersonate the base station, then she may be able to attract all traffic to her; this is often referred to as the *sinkhole* attack [25].

All these basic attack methods listed so far can serve as building blocks for further more complex attacks such as the HELLO flood attack against TinyOS beaconing, the creation of routing loops, black- and grayhole attacks, or route diversion attacks [25]. Some of these will be exemplified in the following subsection.

2.4. Specific examples

In this subsection, we illustrate the previously described attack methods on different types of sensor network routing protocols: we show how the adversary can subvert TinyOS beaconing, Directed Diffusion and GPSR. Many other routing protocols, even some "secure" ones, as we will see in Section 4, are also vulnerable to these simple attacks.

2.4.1. TinyOS beaconing

Originally, the authors of TinyOS proposed a very simple routing protocol in [20], called TinyOS beaconing. In this protocol, each node is addressed by a globally unique identifier, and the base station periodically initiates a route discovery by flooding the network with a beacon message. Upon the reception of the first beacon within a single beaconing interval, each sensor node stores the identifier of the immediate sender of the beacon as its parent (a.k.a., next-hop towards the base station), and then re-broadcasts the beacon after replacing the sender identifier with its own identifier. As for each node only one parent is stored, the resulted routing topology is a tree. In the data forwarding process, every sensor node receiving a data packet forwards that towards the base station by sending the packet to its parent. This beaconing mechanism is a straightforward method to build a simple routing topology, where each node sets a neighbor as its parent if this neighbor lies on the fastest path to the base station. The protocol assumes symmetric links in the network and does not consider any energy metric to optimize network lifetime.

In the following, we show how the adversary described in Subsection 2.1 can create a routing loop in a sensor network using TinyOS beaconing. Let us consider Figure 1. First, the base station B floods the network with a beacon containing its identifier. Before re-broadcasting the beacon, node E replaces the sender identifier with its own identifier to indicate to its neighbors that they can reach the base station through E. Receiving this beacon, the adversarial node A does not replace the sender identifier with A according to the protocol rules, but it replaces it with D. Hence, C sets D as its parent node, and rebroadcasts the packet with its own identifier causing node D to set C as its parent node. As a result, node C will forward all data packets to D and D will forward all data packets to C without ever reaching the base station and consuming valuable resources.

2.4.2. Directed Diffusion

Directed Diffusion [23] is another mainstream topology-based routing protocol. The base station initially floods the network with an *interest*, which contains attribute-value pairs describing the requested data. Upon the reception of an interest, each sensor node sets a gradient pointing to the immediate sender node. A gradient defines the requested data at each sensor node in conjunction with the next-hop towards the base station through

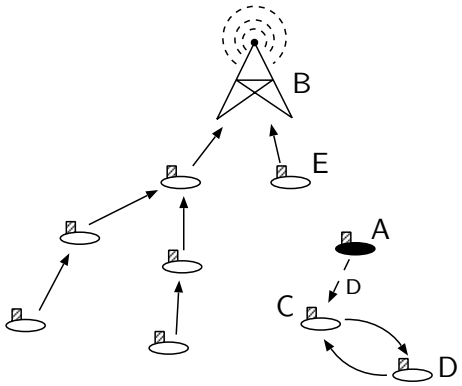


Figure 1. Creating a routing loop in TinyOS beaconing. The only adversarial node is denoted by A. For each node, the solid line points to the parent node. Initially, the base station B floods the network with a beacon. Receiving this beacon from node E, the adversarial node A rebroadcasts it in the name of node D that is denoted by a dashed arrow. Upon the reception of this falsified beacon, node C will believe that the sender of this beacon is node D. Thus, C sets D as its parent node.

which a message containing the requested data should be forwarded. Moreover, each gradient is weighted proportionally to the amount of data that is allowed to traverse the gradient. If a node receives the same interest from different neighbors, then the node can set multiple gradients, which correspond to the same interest, pointing to different neighbors. The neighbors are differentiated by locally unique identifiers.

The data is forwarded to the base station by the intermediate nodes along their gradients. If there are more gradients at a node for the same interest, then the node forwards one copy of the message along each gradient. After a while, the base station selects the route with the best quality and increases the weight of the gradients along the route (positive reinforcement), whereas it decreases the weights on the others (negative reinforcement).

Intermediate nodes may aggregate the received data, and forward this aggregated data along the corresponding gradients at rate that is proportional to the weight of the gradient. The base station periodically re-sends the interests along the used routes in order to keep the gradients of intermediate nodes alive. In this way, the base station keeps the empirically best routes and eliminates the routes that have worse quality. Optionally, all nodes can cache data in order to achieve shorter response time and increase robustness. A more comprehensive description can be found in [23].

The adversary can easily mount black- and grayhole attacks against Directed Diffusion. Blackhole attack means that the adversary allures all traffic from a particular area along an adversarial node, and then drops all received packets. Grayhole attack is more sophisticated selective forwarding; the adversary first allures the traffic and then selectively drops some data packets. Let us consider the network topology depicted in Figure 2. The adversarial node can simply allure the traffic by broadcasting a forged interest in the name of B. Thus, all nodes receiving this forged interest will send data packets to node A. A more clever adversary can exploit the reinforcement strategy of Directed Diffusion; the adversarial node A reinforces some paths without forging any interests (i.e., receiving the original interest from node B, A rebroadcasts that containing

increased data-rate values). Consequently, all nodes receiving this modified interest will forward data packets towards the base station along A at higher data rates, which then can drop, modify, or forward packets at her own wish. Moreover, this false reinforcement also causes honest nodes' batteries to deplete faster.

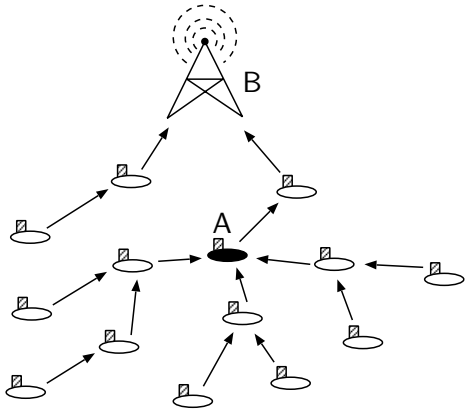


Figure 2. Black- and grayhole attack against Directed Diffusion. The only adversarial node is denoted by A. For each node, the solid lines denote the gradients set towards the base station. Node A can allure the traffic by disseminating faked interests in the name of B and/or by manipulating the reinforcement strategy of Directed Diffusion. As a result, all honest nodes that receive the falsified interests originating from A will select the path that traverse node A. This is illustrated in the figure by that all nodes in a considerably part of the network set gradients towards node A. After alluring the traffic, node A can mount selective forwarding.

2.4.3. GPSR

GPSR (Greedy Perimeter Stateless Routing) [26] is a geographic routing protocol proposed for wireless ad hoc and sensor networks that can be used to route data packets between any pair of nodes (i.e., it supports node-to-node communication). GPSR assumes that every sensor node is aware of its own location and the locations of its neighbors.

Initially, nodes construct a planar subgraph based on the network topology in a distributive manner. This distributive planarization algorithm can be GG (Gabriel Graph) [17], RNG (Relative Neighborhood Graph) [54], CLDP (Crossing Link Detection Protocol) [27] or LCLR (Lazy Cross Link Removal) [28]. We do not detail the planarization process further, more interested readers are referred to the corresponding literature. This planar graph will be used to circumvent voids in data forwarding.

Upon the reception of a data packet that carries the location of the destination node D, each node I checks whether it has a neighbor that is closer to node D than itself. If it has, the message is forwarded to that node. Otherwise, I switches to face routing mode, which means that it determines the neighboring face in the planar subgraph that is intersected by the imaginary line connecting I and the destination, denoted by d. After putting its own location into the packet, I uses the right-hand rule to select the next-hop on the perimeter of that face. Each node on the perimeter of the face uses the same rule to select the next-hop for the packet until one of the following events occurs:

- A node is reached that is either D or it is closer to D than I. In the latter case, the node that is closer to D than I performs the same steps that I did and the process repeats.
- An edge is reached, denoted by e, which is intersected by line d. In that case, GPSR switches to the neighboring face that is intersected by d (i.e., the neighboring face contains e).
- If the packet completely traverses the perimeter of the face (i.e., e or I is traversed again) without reaching a node being closer to D, then the packet is marked as undeliverable.

We show how the adversary can divert the traffic and create detours between a source and a destination causing increased energy consumption, and thus decreased network lifetime. In Figure 3, a source node S is assumed to send a packet to the base station B. As node E is closer to B than any other neighbors of S, S forwards the packet to E. Similarly, E forwards the packet to the first adversarial node A. In order to divert the traffic, A alters the destination location in the packet to the location of the second adversarial node A′. Hence, following the GPSR rules the packet will be forwarded to A′ along nodes H, K. Afterwards, A′ recovers the original destination location to B's location, and the packet will be delivered along node Q. Therefore, the packet reaches node B along nodes E, A, H, K, A′, Q instead of nodes E, A, C that would be a much shorter and less energy consuming route.

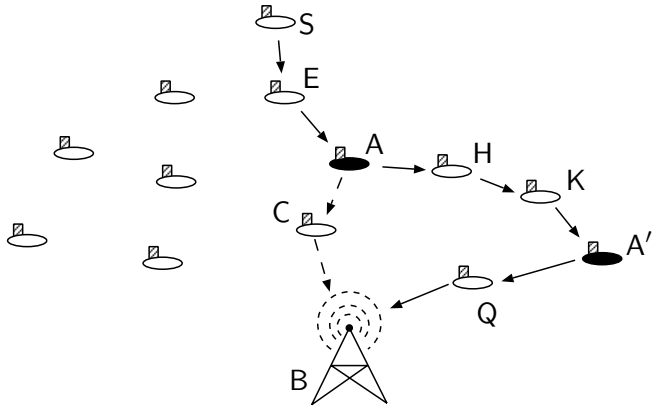


Figure 3. Route diversion attack against GPSR. The adversarial nodes are denoted by A and A′. Using GPSR, the source node S sends a packet towards B. Receiving this packet, A alters the destination of the packet from B to A′. When A′ receives the packet from K, it recovers the original destination to B. Thus, the packet reaches B along nodes E, A, H, K, A′, Q that is denoted by solid arrows. In contrast to this, if the adversary does not divert the packet, it will traverse nodes E, A, C, denoted by dashed arrows, which is a much shorter route.

3. Countermeasures

In this section, we review some countermeasures that can be used to defend against the attacks described in Subsection 2.3. Some of these countermeasures will serve as building blocks for secure sensor network routing protocols, which will be described

in the next section. In Subsection 3.1, we address link layer security focusing on how one can ensure message authenticity, confidentiality and freshness at the link layer. In Subsection 3.2, we consider the security of neighbor discovery; we describe the basic defenses against the Sybil, the node replication and the wormhole attacks. The techniques of broadcast authentication are discussed in Subsection 3.3. Finally, we are concerned with robust data delivery in Subsection 3.4.

3.1. Link layer security (prevention of outsider attacks)

To protect against malicious manipulations of routing messages exchanged between sensor nodes as well as between base station and sensor nodes, one can employ traditional cryptographic primitives, whereby one can provide authenticity in conjunction with integrity, as well as confidentiality for messages. However, providing these security services in an end-to-end manner is not desirable in sensor networks, because the authenticity of routing control messages need to be verified and their content need to be accessed by intermediate nodes. Therefore, we need to provide these services in a hop-by-hop manner at the link layer.

Providing link layer security is quite straightforward, however, one must keep in mind the resource constraints of the sensor nodes. In particular, sensor nodes are less capable to perform expensive cryptographic computations and communicate long signatures and MACs. For this reason, symmetric key cryptographic mechanisms are often preferable in sensor networks.

An example of a widely used software package that provides symmetric key cryptographic mechanisms and that can be used to implement security services at the link layer is TinySec [24], which runs on top of TinyOS [20].

Finally, we note that the countermeasures implemented at the link layer do not defend against insider attacks, where the adversary controlling some intermediate nodes can manipulate messages in an undetectable way.

3.2. Secure neighbor discovery

3.2.1. Preventing the Sybil attack

A Sybil node, which uses multiple identities, can obtain these identities by either fabricating new identities or stealing existing ones from honest nodes (e.g., by node capturing). Current defense mechanisms can be grouped into two major categories; *decentralized* and *centralized* techniques. The former includes resource testing, cryptographic defenses, and secure position verification, while the latter incorporates registration and anomaly detection.

Resource testing relies on the observation [13] that each node (device) is generally limited in some resources (e.g., computation, storage, or communication). Even an adversarial node that uses multiple fake identities cannot multiply its resources, therefore, with appropriate mechanisms one can discover that those fake identities do not belong to different nodes. For instance, one feasible way to do this is the radio resource testing [44], which assumes that every node has only a single antenna that is generally incapable of using multiple channels simultaneously. Applying this scheme, a verifier node assigns different channels to all its neighbors. Then, the verifier selects randomly a channel and broadcasts a message on that. If the neighbor assigned to this channel is legitimate, it

should answer or at least acknowledges the message. However, if the channel is assigned to a Sybil node, the corresponding adversarial node will not respond with a certain probability, because it has only one antenna on which he can listen on a single channel. Repeating this test, the probability that the misdeed is not detected can be made arbitrarily small [44], even if the verifier can test only a subset of its neighbors at one time (i.e., the number of channels that can be used is less than the number of the neighbors).

One advantage of all resource testing schemes is that they can be used against both fabricated and stolen identities. By contrast, all cryptographic defenses can only prevent the illegal fabrication of identities. However, if the adversary retrieves somehow the cryptographic material (e.g., secret keys) needed to authenticate an existing identity (i.e., she steals the identity), then this identity can be used by a Sybil node.

Some cryptographic defenses are based on the *random key predistribution* scheme proposed in [16]. In this approach, each node is randomly assigned a subset of symmetric keys from a common key pool. If the size of this subset is large enough, any two nodes will have at least one common shared key with high probability based on the birthday paradox. The adversary who aims at fabricating a *new* identity can easily do that by capturing multiple nodes and use every combination of the compromised keys. In [44], the authors propose a defense by binding keys to identities. Here, we only outline the main idea: All keys are indexed in the common key pool with m keys, and the identity ID is assigned to k keys by calculating the key index with $F^{PRF}_{h(ID)}(i)$ $(1 \leq i \leq k)$, where h is an one-way hash function and F^{PRF} is a Pseudo-Random Function (e.g., F^{PRF} can be a CBC-MAC construction [46]). Therefore, the best that an adversary can do is to perform an exhaustive search on the set of fabricated identifiers (i.e., calculating $F^{PRF}_{h(ID')}(i)$ to find a valid ID') until she finds a candidate such that all keys assigned to the candidate are known by the adversary. This scheme provides a reasonable level of security until the number of captured nodes reaches a certain threshold. However, above this threshold, the adversary can easily fabricate new identities despite using this scheme. Single space pairwise key distribution [6,4] approaches are inherently resistant against identity fabrication until no more than λ nodes are compromised due to the λ-secure property. If more than λ nodes are compromised, the adversary will be able to fabricate arbitrary number of identities. Single space approaches and the basic pool approach are combined in the multi-space pairwise key distribution approaches [14,34]. Here, k out of m key spaces are randomly assigned to each node, and the adversary must compromise more than λ instances of each space to compromise that only one key space. In [44], the authors showed that multi-space pairwise approaches provide stronger defense against identity fabrication than both basic key pool and single space approaches (i.e., by multi-space scheme, the adversary needs to compromise more nodes to successfully fabricate a new identity than in the basic key pool or single space schemes).

Further cryptographic defenses can be provided by using some *identity certification method* like digital certificates. However, these schemes are based on digital signatures that are considered to be expensive in terms of computation and communication. A more viable identity certificate method was proposed in [65] that uses Merkle-trees [41] and one-way hash chains instead of expensive public key cryptography to prove identities.

There are also some additional decentralized defenses like *secure position verification* [50] and *code attestation* (a.k.a, remote code verification) [45,53]. The former has shown significant progress recently, but the latter one still incurs substantial overhead.

In case of the centralized approaches, such as registration and anomaly detection, a trusted entity (e.g., the base station) is required to have a global view of the network. In case of the *registration* approach, this central entity can preclude Sybil attacks by accurately tracking the operation of the network (e.g., a node can verify its neighbors by querying the central entity for the list of nodes registered in the network). Alternatively, this central entity can also detect *anomaly* related to Sybil attacks (e.g., nodes which are reported to have too many neighbors are good candidates for Sybil nodes). All these centralized schemes may provide a reasonable solution against Sybil attacks in many simple scenarios (e.g., in small-sized networks).

Summarizing all countermeasures against Sybil attacks, we can conclude that none of them provide a perfect defense. The radio resource verification may be circumvented by using special radio hardware and it can also incur substantial energy consumption. All centralized techniques has limited applicability to sensor networks, since the central element becomes a single point of failure and these solutions are not scalable in general. Moreover, cryptographic solutions cannot be used against stolen identities and the success of secure position verification highly depends on the accuracy of the localization method.

3.2.2. Detecting node replication attacks

Node replication occurs when a single identity is used by multiple nodes simultaneously in the network. For instance, the adversary may try to deploy additional adversarial nodes that use the same identity or an identity that is already in use by a honest node in the network. Note that Sybil attacks and node replication attacks can be jointly mounted at the same time; a single adversarial node can use multiple identities from which at least one is already in use simultaneously by further adversarial nodes or honest nodes.

There exist centralized and decentralized detection algorithms for node replication attacks. By centralized detection, each node sends its whole neighborlist to the base station which then can filter out replicated nodes and revoke them by flooding the network with authenticated revocation lists [16]. However, similar to the centralized methods in the case of Sybil attacks, this method also has many drawbacks such as scalability problems and degraded robustness. Decentralized detection protocols are much promising due to the distributive nature of sensor networks. Here, we briefly present two decentralized approaches with illustration purposes: randomized multicast [9] and line selected multicast [9].

Both randomized multicast and line selected multicast operates similarly; the location claim of all nodes are disseminated in the network to some witness nodes which record all received location claims. If a witness node receives duplicate location claims with the same originator identity but with different locations, then the replication is detected and the witness floods the whole network with an authenticated revocation message. Here, we assume that each node is aware of its location and also knows the size of the communication range.

Using randomized multicast detection, all nodes in the network broadcast their location claims. The location claim is authenticated by each node using public key cryptography or another more efficient broadcast authentication scheme described in Subsection 3.3. The location claim of node A is received by the neighbors of A. Then, each neighbor of A verifies the authenticity and the plausibility of the location claimed by A (e.g., the plausibility can be checked by calculating the distance between the claimed location and

the location of the neighbor, and if this distance is less than the radius of the communication range, then the location claim is plausible, otherwise, it is considered to be implausible). If the location claim is not authentic or it is implausible, the neighbor discards the location claim. Otherwise, with probability p, each neighbor selects r random locations in the network and for each of these locations, the neighbor uses geographic routing to forward the location claim of A to the node that is closest to the selected location (e.g., by using the routing algorithm in GHT [48]). If the average number of a node's neighbors is denoted by k, then $p \cdot r \cdot k$ is a good approximation for the number of nodes that receive A's location claim. If $r \approx \sqrt{n}$ (i.e., $p \cdot r \cdot k$ is in the order of $O(\sqrt{n})$), where n denotes the number of all nodes in the network, then based on the birthday paradox a single replication can be detected with high probability (≈ 0.5) [9] (i.e., there exists at least one witness node that receives the location claims from both the original and the replicated node with high probability). Moreover, as the number of the replication of the same node increases so does the probability of the detection (assuming that each node has at least one legitimate neighbor). On average, each node needs to store $p \cdot r \cdot k$ location claims, thus the storage cost is in the order of $O(\sqrt{n})$, and the communication cost is in the order of $O(n\sqrt{n} \cdot p \cdot r \cdot k) = O(n^2)$ [9]. However, the storage cost can be further reduced by using some time synchronization enhancements detailed in [9].

In order to reduce the communication cost of the randomized multicast detection, another scheme called line selected multicast was proposed in [9] that was inspired by Rumor Routing [7]. The operation of line selected multicast is illustrated in Figure 4. The idea is that along the path between a neighbor of A and a witness node each intermediate node verifies whether it has already received a different location claim with the identity of A. If the intermediate node has multiple location claims with A's identity, then it floods the network with an authenticated revocation message. Otherwise, the node buffers the location claim (if there is no any location claim with identity A in the buffer), and forwards the location claim towards the witness node. In this way, there will be line segments throughout the network that radiate out in random directions from node A. Now, if there is a replicated node with identity A in the network, the probability that the replication is detected equals to the probability that at least two line segments radiating out from A and its replica, respectively, intersect. It can be shown by using Monte-Carlo simulations that the probability that two random line segments originating from A and its replica intersect is greater than 0.56. If there are five line segments per node this probability can further increase even up to 0.95 depending on the network topology. If each line segments has a length in the order of $O(\sqrt{n})$, the communication cost of this scheme is $O(n\sqrt{n})$ for the whole network. Regarding the storage cost, each node stores $O(\sqrt{n})$ location claims. Moreover, the storage cost can be further reduced by using some time synchronization enhancements detailed in [9].

Both schemes rely on the assumption that node A has at least one honest neighbor that faithfully forwards the location claim of A. However, this assumption does not hold for every practical scenario (e.g., the adversary may compromise all neighbors of A). To circumvent this problem, the authors in [9] proposed a slightly modified scheme where not the immediate neighbors of A but the closest honest nodes to A will act as pseudo-neighbors of A (i.e., they explicitly query A for its location claim, and in case they do not receive any response, they deny to forward any traffic originating from A). More interested readers are referred to [9].

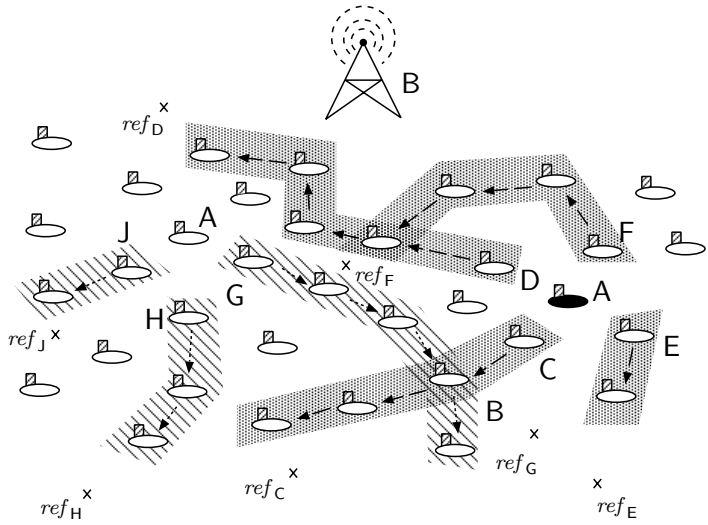


Figure 4. Line selected multicast to defend against node replication attack. The single replica of node A is denoted by a black filled node A. Let us assume that the replica passes the plausibility check of its neighbors. Thus, some of these neighbors B, D, E, F choose random geographic locations denoted by ref_B, ref_D, ref_E, ref_F, respectively, and route the location claim of the replica to the nodes (witnesses) closest to these selected locations (routes are denoted by dashed arrows). The location claim is stored at each intermediate node. The neighbors of honest node A, which are J, G, H, perform the same steps (these routes are denoted by dotted arrows). In this example, the replication is detected at node B which receives the location claims of both honest node A and its replica.

We note that randomized multicast detection as well as line selected multicast detection rely on some existing routing infrastructure like some variant of GPSR. Thus, the effectiveness of the schemes depends on two factors: (i) how fast the misdeed is detected by the sensor nodes, (ii) how fast the revocation list is distributed *before* routing any data packets. Finally, we note that these schemes are not applicable to those protocols that use locally unique identities, because in that case, two nodes being far away from each other can legally use the same identity.

3.2.3. Detecting wormholes

Broadly speaking, the different wormhole detection mechanisms fall into two classes: the centralized mechanisms and the decentralized ones. In case of the centralized approach, data collected from the local neighborhood of every node are sent to the base station. The base station uses the received data to construct a model of the entire network, and tries to detect inconsistencies in this model that are potential indicators of wormholes. In the decentralized approach, each node constructs a model of its own neighborhood using locally collected data, and performs wormhole detection locally. The advantage of the decentralized wormhole detection mechanisms is that they are more energy efficient and scale better than the centralized mechanisms, therefore, they can be used in a wider range of applications. Their disadvantage is that they usually rely on some strong system assumptions (e.g., accurate time synchronization between the nodes) or on the availability of some special hardware in the nodes (e.g., a GPS receiver or a directional antenna).

Two examples of the centralized approach are the wormhole detection mechanisms proposed in [10] and [58]. In the scheme described in [10], the nodes report only the list of their believed neighbors to the base station, and the model constructed by the base station consists of the connectivity graph of the network. A crucial observation is that a wormhole *always* increases the number of edges in the connectivity graph, as it introduces new neighbor relationships. This increase in the number of edges changes the properties of the connectivity graph in a detectable way with respect to some expectations that are based on some basic assumptions about the system (e.g., the distribution of node positions, the communication range of the nodes, etc). The main idea is to detect the changes in the connectivity graph using statistical hypothesis testing methods. In particular, wormholes are detected in [10] by identifying distortions in the distribution of the number of neighbors and in the distribution of the length of the shortest paths between all pairs of nodes using the χ^2-test.

In [58], the nodes also estimate the distances from their believed neighbors, and send their neighbor list with the estimated distances to the base station. In this case, the model constructed by the base station is a virtual layout of the network. The crucial observation here is that a wormhole contracts the virtual layout in certain regions, since it makes some nodes appearing neighbors while in reality these nodes are far away from each other. The main idea is then to detect these contractions by visualizing the virtual layout.

Examples of the decentralized wormhole detection approach can be found in [22]. Most of these decentralized approaches are based on the straightforward idea of estimating the real physical distance between the nodes that are believed to be neighbors. If the estimated distance is larger than the nodes' communication range, then the nodes are likely connected through a wormhole.

Two mechanisms for distance estimation with the specific purpose of wormhole detection are proposed in [22]; they are called geographical and temporal *packet leashes*, respectively. The main idea of both mechanisms is to add some extra information to the packets that restricts their maximum allowed transmission distance.

A geographical leash is based on location information, and it allows the receiver of the packet to determine an upper bound on its distance to the sender. It is assumed that each node is aware of its own location, which may be determined using GPS or some other positioning mechanism. It is further assumed that the nodes maintain loosely synchronized clocks.

A temporal leash is based on timing information, and it ensures that the packet has an upper bound on its lifetime. Indirectly, however, this also ensures an upper bound on the distance between the sender and the receiver, since the packet cannot travel faster than the speed of light. Temporal leashes require that the nodes have tightly synchronized clocks, such that the maximum difference between any two nodes' clocks is in the order of a few microseconds or even hundreds of nanoseconds.

Both geographical and temporal leashes require that the packets carrying the leashes are authenticated and their integrity is protected, since otherwise an adversary can modify a leash and jeopardize the distance estimation. Origin authentication and integrity protection can be based on digital signatures or on symmetric key MACs. The advantage of digital signatures is that they provide broadcast authentication, and therefore, they can be used efficiently for protecting neighbor discovery beacons and route discovery messages that are usually broadcast messages. The disadvantage of digital signatures is that they are several orders of magnitude slower than symmetric key MAC computations, and

speed is critical, especially in the case of temporal leashes. In order to overcome this problem, in [22], the authors propose to use μTESLA with Instant Key-disclosure (TIK) to authenticate temporal leashes in packets.

Both types of leashes can be used for wormhole detection, because they allow the receiver of the packet to detect if the sender is further away than the nodes' communication range. More precisely, the receiver can determine only an upper bound on its distance to the sender. However, if this upper bound is greater than the nodes' communication range, then the receiver should not accept the packet. In this way, packets that arrive through a wormhole are always rejected.

Packet leashes can be added to the neighbor discovery beacons or to the packets of the neighbor discovery protocol when the nodes use such mechanisms explicitly for setting up their neighbor relationships. In that case, the application of packet leashes prevents the establishment of fake neighbor relationships. When no explicit neighbor discovery mechanism is used in the system, packet leashes can still be added to the packets of the routing protocol, in order to prevent the undesirable effects of wormholes on routing.

Another approach that is also based on distance estimation between the nodes, but does not require any clock synchronization or localization mechanisms, is proposed in [55]. This approach is based on the concept of *distance-bounding*, which was first introduced by Brands and Chaum in [8]. Distance bounding is based on the facts that electromagnetic waves propagate nearly with the speed of light and with current technology it is easy to measure local timings with nanosecond precision. Brands and Chaum's technique essentially consists of a series of rapid bit exchanges between the two nodes. Each bit sent by the first node is considered to be a challenge for which the other node is required to send a one bit response immediately. By locally measuring the time between sending out the challenges and receiving the responses, the first node can estimate its distance to the other node, assuming that the messages travel with the speed of light and the processing delay at the other node is zero.

Note that the estimated distance is only an upper bound on the real distance between the nodes, because the second node may be closer, but it may delay the responses in order to appear to be farther. Even if the nodes are trusted for not delaying their responses, an active adversary can delay the messages between the parties, and hence, the estimated distance will still be just an upper bound on the real distance. However, in case of a wormhole attack, the adversary's goal is not to make the two nodes believe that they are far away from each other, but on the contrary, the adversary wants that the two nodes believe that they are within each other's range, while in reality they are not. But in order to achieve that the estimated distance is smaller than the nodes' real distance, the adversary should arrange that the messages travel faster than the speed of light, which is impossible. Thus, distance-bounding can be used for wormhole detection.

In [55], a slightly modified version of the above described distance-bounding technique is proposed, which is called Mutual Authenticated Distance-bounding, or shortly MAD. As its name suggests, the MAD protocol allows both nodes to measure the distance between them simultaneously. In addition, it uses symmetric key cryptographic primitives for authentication purposes. In order for this to work, it is assumed that each pair of nodes share a symmetric key, which is established before running MAD between them. Extensions of this idea can be found in [56,57].

Another wormhole detection mechanism that uses location information is described in [32]. However, the advantage of this mechanism compared to geographical packet leashes is that it requires only a few specialized nodes to be aware of their locations. These specialized nodes are called *guards*, and their role is to help other nodes to set up their neighbor relationships in a secure way.

In [32], it is assumed that the guards have a larger transmission range than that of regular sensor nodes. Let us denote the former by R. Two nodes consider each other neighbors only if they hear each other, and in addition, they hear more than a threshold number of common guards. The messages originating from the guards contain the location information of their sources. The nodes can use this location information to detect wormhole attacks based on the following two principles:

1. Since any guard heard by a node must lie within a range of radius R around the node, a node cannot hear two guards that are $2R$ apart from each other.
2. Normally, a node should not receive the same message twice from the same guard.

It is shown in [32] that if there is a wormhole in the system, then at least one of the above principles is violated with probability close to one. This means that the wormhole is detected in a reliable way.

In [21], the authors propose to use directional information of messages to mitigate wormhole attacks. It is assumed that each node in the network is equipped with a directional antenna. Every such antenna has n, non-overlapping zones, and each zone has a spanning angle of $2\pi/n$; hence, the zones collectively cover the entire area around a node. When a node is idle, it listens to the carrier in omni-directional mode. When it receives a message, it determines the zone in which the received signal strength is maximal, and uses that zone to communicate with the sender. An important assumption is that the orientation of the zones is always established with respect to the Earth's median, and therefore, all nodes use the same orientation no matter of their physical locations and their own orientations. This can be achieved in modern antennas with the help of a magnetic needle that always remains collinear to the Earth's magnetic field.

The main idea is that when two nodes are within each other's communication range, they must hear each other's transmission from opposite directions. However, if the nodes communicate through a wormhole, then this condition is not always satisfied. Unfortunately, this simple principle is not sufficient to reliably detect wormholes as the wormhole can be placed in such a way that two nodes hear each other form opposite directions even though they are communicating through the wormhole. For this reason, the authors of [21] introduce the notion of *verifier* nodes that can help to detect and avoid this situation.

There are specific conditions that a node should satisfy to qualify as a verifier, and it is possible that two nodes are within each other's communication range, but there are no potential verifier nodes that they can use. In that case, the nodes cannot set up a neighbor relationship and we lose a potential link. Unfortunately, losing links is not desirable, because it reduces the robustness of the network in case of link failures, and increases the average length of the routes in the network. Another disadvantage of this approach is that it requires directional antennas in the nodes, which may be expensive.

3.3. Authenticated broadcast (prevention of malicious flooding)

As many routing protocols rely on flooding or broadcasting routing information, authentication of broadcast data sent by the base station (or rarely by sensor nodes) is a fundamental issue. As we have already seen, the adversary can easily inject extra packets and modify existing ones during routing procedures. Thus, the receiver of a packet should be ensured that the packet is indeed originated from the claimed sender (*source authentication*) and the packet is not altered during the transit (*data authentication*).

The first naive solution would be to employ pairwise shared symmetric keys (i.e., the sender shares a pairwise key with each receiver, and it computes a MAC for each receiver using the corresponding pairwise key). However, this solution is impractical in case the number of receivers is large because the sender may have to attach a huge number of MACs in a single message that is not tolerable due to the high communication cost.

Another simple idea might be to use a network-wide symmetric key. However, in that case, the adversary compromising a node could retrieve the global key, and thus, the adversary would be able to forge messages from the sender. However, if some secure group re-keying mechanism is also provided to refresh network-wide keys in each node either in a periodic or an on-demand manner like in LEAP [66], this can also be an appropriate alternative to perform broadcast authentication. A special case of broadcast authentication is when the receivers are limited to the immediate neighbors of the sender. In that case, a local broadcast key [66] shared with all the one-hop neighbors of the sender can be used to authenticate local broadcast messages.

In the following, we review some asymmetric techniques that are employed to authenticate broadcast messages in sensor networks; digital signatures, one-way hash chains, μTesla, and one-time signatures.

3.3.1. Digital signatures

The most common way to authenticate broadcast messages is to apply some digital signature schemes like RSA [49] or ECDSA [60]. Employing RSA is reasonably secure with 1024 and 2048 bit keys but the signature size is quite long compared to the typical message size. Moreover, RSA requires extensive computational effort on behalf of the sender (signer). Elliptic Curve Cryptography (ECC) [43,29] can provide another signature scheme called ECDSA (Elliptic Curve Digital Signature Algorithm) with the same security level as RSA but with significantly lower signature size. For instance, ECC with 160 bits public key offers the same security as the RSA with 1024 bits key, where the ECC signature size is about 41 bytes compared to the 128 bytes long RSA signature. ECDSA requires less computational effort from the sender (signer) but puts more burden to the receiver (verifier).

Recent empirical studies showed that ECDSA signature generation/verification and RSA signature verification can be effectively implemented on various sensor nodes.The clear advantage of using RSA signature scheme is the fast (and less energy consuming, see [47]) verification compared to ECDSA. As the signature verification is mostly performed by the constrained sensor nodes, the energy consumption of this verification process is a critical issue. As long as the RSA signature is generated on the base station, RSA signature computations incur less overhead with respect to sensor nodes but bears higher communication cost due its larger signature size than ECDSA. Moreover, receiving and sending an RSA signature consumes about 3 percent of energy needed to verify

an ECDSA-160 signature on average [47]. As a conclusion, there cannot be made an ultimate decision between RSA and ECDSA regarding broadcast authentication. ECDSA may be more favorable in those applications where the signature is generated by sensor nodes, and/or the communication costs of RSA signatures exceed the verification costs of ECDSA signatures.

Although recent advances in public key cryptography (PKC) of sensor networks are very promising, PKC still falls behind the standard symmetric cryptography approaches in terms of computational performance; the signature verification and generation are still much slower than MAC verification and generation, respectively. Hence, researchers have also proposed other alternatives for broadcast authentication based on symmetric cryptography that we will discuss below.

3.3.2. One-way hash chains

Using one-way hash chains [31] is the most straightforward technique to provide source authentication. By employing this scheme, the sender generates a collection of values $(c_0, \ldots, c_n)$ such that each value c_i is a one-way function of the next value c_{i+1}. c_n is also referred to as the seed of the chain. For instance, the sender can derive a seed denoted by c_n from a secret key K_{pr} by applying a Pseudo-Random Function F^{PRF} to a constant value using K_{pr} (i.e., $F^{PRF}_{\mathsf{K}_{pr}}(0) = c_n$). Afterwards, the sender generates a hash-chain with length n by iteratively applying a public one-way hash function h to c_n n times, where the ith element of the chain equals to $c_i = h^{(n-i)}(c_n)$, for $0 \leq i < n$. Then, c_0, also called as the commitment of the chain, is stored at each receiver node. When the sender sends broadcast messages, it places c_1 into the first broadcast message, and c_2 into the second broadcast messages, and so on. A node receiving a broadcast message containing hash value w can check whether there exists j $(1 \leq j \leq n)$ such that $c_0 = h^{(j)}(w)$, as the receiver has c_0. If there exists such j, the receiver node is assured that w is generated by the sender, otherwise, the adversary should invert h that is computationally infeasible due to the one-way property of h.

This authentication technique is favorable in sensor networks due to its generally low resource demand; the base station can generate a hash chain, and the sensor nodes can check the authenticity of the base station if they are provided by the seed beforehand. On the other hand, the receiver has to perform $n - j$ computations to verify c_j if c_i is given, which can be expensive for a constrained sensor node if $j-i$ is large. This motivated more efficient hash chain constructions that were proposed in [63,51,11]. Moreover, using hash-chains generally does not guarantee data integrity, unless all messages to be sent are known by the sender a priori (i.e., the message contents are involved in the chain construction), which only holds for a few applications.

3.3.3. μTesla

μTesla is an efficient broadcast authentication protocol [46] that uses symmetric cryptography (MACs) and hash chains to provide authenticity for the sender. μTesla uses *time* to provide asymmetry for the sender which requires that all receivers are loosely synchronized.

Initially, the sender splits up the time into time intervals where each time interval has a constant duration denoted by T_{int}, and creates a one-way hash chain, as described in the previous subsection, where each element of the hash chain c_i corresponds to a

secret key K_i. We recall that the commitment of the hash chain (i.e., c_0) is stored at every receiver node. Subsequently, the sender assigns K_i to interval I_i and defines a disclosure lag d for keys, which is typically in the order of few time intervals. We assume that along with the seed of the hash chain the key disclosure schedule including d and T_{int} is also stored at every receiver node.

After the initialization, the sender can authenticate broadcast messages in the ith interval in the following way. The sender computes a MAC on the message content using the key K_i, and appends the MAC to the message. Additionally, the sender may attach a key to the message that can be disclosed (i.e., in interval I_i key K_{i-d} can be disclosed). Of course, the key disclosure lag d highly depends on the maximum synchronization error denoted by δ. In particular, it must be assured that after a key K_i is disclosed by the sender, none of the receivers accept any messages authenticated by key K_i. Hence, each receiver that receives a broadcast message must verify the following security condition: $\lfloor \frac{T_{current}+\delta-T_0}{T_{int}} \rfloor < I_i + d$, where $T_{current}$ is the local time at the verifier node and T_0 is the start time of the first interval. In other words, each receiver checks whether the key used to generate the MAC carried by the message is still secret assuming that the clock of the receiver is loosely synchronized with the sender. If the MAC key is still secret, then the receiver buffers the message. In addition, each receiver also checks whether the disclosed key is correct: the receiver has to iteratively apply the hash function, which is used to generate the key chain, to the disclosed key until the most recent (already received) *correct* key is resulted. Afterwards, the receiver can check the correctness of MACs in the buffered messages that were sent during interval I_i. Note that if intermediate keys are lost, they can be regenerated using later keys based on the property of one-way hash chains. Thus, if some disclosed keys are lost due to malicious packet dropping or jamming, a receiver can recover the key from keys disclosed later and check whether earlier messages are authentic.

μTesla is inappropriate for real-time applications, when messages should be authenticated immediately. For instance, alarms should be typically disseminated and authenticated with minimal delays. Moreover, if such alarms occur infrequently, a receiver may need to compute a long part of the hash chain to verify the authenticity of the key (in the case of infrequent messages, the sender releases keys that can be far away from each other in the key chain). This can be exploited by the adversary to mount a DoS attack: the adversary sends incorrect keys to the receiver which performs key verification on the incorrect key and becomes overloaded (i.e., the receiver may need to perform thousands of hash computations which takes several seconds before it notices that the key is incorrect).

To circumvent the problem of authentication delay, the authors in [39] proposed a modified version of μTesla called RPT (Regular-Predictable Tesla). This protocol is an alternative of the μTesla for scenarios where the messages are sent at regular and predictable times. For instance, if we consider a monitoring application where the base station distributes the sensing tasks once per day but the sensors must continuously report measurements, we generally cannot postpone the execution of current sensing tasks for a whole day. RPT solves this problem by performing message broadcast jointly with key disclosure only after the distribution of the corresponding MAC (i.e., when the MAC is received by all nodes the sender can broadcast the message along with the corresponding secret keys). Of course, the delay between the MAC distribution and the key disclosure

must be large enough to ensure that all receivers will receive the secret key by the time the sender releases the key. Keys are authenticated by hash chains similar to μTesla.

Previously, we assumed that the clocks of the sender and receiver nodes are synchronized off-line and the key-disclosure schedule along with the seed of the key-chain is pre-programmed into each node. However, μTesla was proposed to be used by symmetric key cryptography with a master key shared between the sender and each receiver. In this way, new receivers can also be boostrapped; the receiver first sends a request to the sender, which then replies a message containing the current time for synchronization purposes, a key of the key chain used in a passed interval I_i, the start time of I_i, and the interval duration T_{int} along with the disclosure lag d. However, all these solutions does not scale to large networks with thousands of receiver nodes and may require an existing routing infrastructure. For this reason, several multi-level μTesla schemes were proposed in [35] where the initial parameters of the μTesla scheme can be effectively distributed in large sensor networks. Additionally, multi-level μTesla schemes can be used over a longer time period than basic μTesla scheme. Furthermore, basic μTesla has also been extended to multiple senders in [36], where the revocation of compromised senders is also treated. More interested readers are referred to [36,35].

3.3.4. One-time signatures

A one-time signature is considerably faster to generate and verify than previously described digital signatures, but a private key can be used to sign only a single message. If a private key is used to sign multiple messages the probability that the adversary can successfully forge a valid signature can also significantly increases. Due to its lower computation demand, one-time signatures would be more suitable for sensor networks than RSA or ECDSA signature schemes. However, one-time signatures are still quite large compared to the message size. For instance, signing 16 bytes results in a 280 bytes long signature using the Merkle-Winternitz signature scheme [42] with 8 byte long hash chain values that is still too long in sensor networks. One-time signatures are only beneficial if we use short messages, or more precisely, messages with lower entropy, typically containing only a small number of bits (e.g., a 8 bytes long message has a 32 bytes long signature using the aforementioned scheme and this can be further reduced to 24 bytes if we require a bit more verification effort).

The problem still remains: how can we authenticate more than one messages without degrading security? In particular, the sender should provide a unique public key per message for the receivers. One solution is to pre-deploy n public keys at bootstrap and these keys can be used to sign the first n messages. Afterwards, the sender employing a μTesla scheme can distribute further public keys, which number is further denoted by k, at regular time in a periodic manner. In that case, the sender is limited to sign k messages between the μTesla key disclosure times. This solution entails a new problem: the greater is k the more memory resource the receiver needs. LEA (Low-Entropy Authentication) proposed in [39] resolves this resource problem by chaining all public keys, where the verification of one signature will automatically authenticate the public key of the next signature. More interested readers are referred to [39].

In summary, one-time signatures combined with some variant of the μTesla scheme can be used to authenticate broadcast messages in an effective way, but this solution is only practical for small sized messages due to the increased signature size.

3.4. Multi-path routing (to achieve robustness)

Multi-path routing, which encompasses delivering of data packets on multiple paths towards the destination, is a common technique to achieve robustness and load-balancing in every communication network. The multiple paths between the source and the destination can be partially or completely disjoint and they are maintained at the expense of increased energy consumption and traffic generation. Apart from load-balancing and robustness against node failures, multi-path routing also inherently provides some defense against selective forwarding attacks and malicious control packet dropping; in order to prevent a packet to reach the base station, the adversary must control a node on each used path to drop the packet. However, the more the number of (disjoint) paths is the less likely that the adversary can drop packets on all paths. We note that there are alternative methods to defend against malicious packet dropping in ad-hoc networks. However, they offer a limited usage in sensor networks, as they either induce heavy communication overload for sensor nodes by using promiscuous mode [40], or they could only be used with source routing protocols that are less attractive in sensor networks due to the increased length of the packet header [3]. Below, we review some multipath techniques used in sensor networks. They are further divided into three groups in order to ease their short exploration:

- The source makes multiple copies of a packet, and routes these copies on different paths in order to increase robustness [23,18]. These paths can be calculated in advance and maintained proactively by sending data packets at a low rate *only* on these paths [18]. Alternatively, if the sources have data to send, they flood the *whole* network with data packets at a low rate, and the destination select the best quality paths according to some network metric [23]. In [18], two further localized methods were proposed to build disjoint multipaths and braided (partly disjoint) multipaths.
- The source routes the single copy of each packet on different paths per packet, where the paths are selected in a probabilistic or deterministic fashion in order to aid load-balancing, and thus prolong network-lifetime. In this category, centralized and decentralized approaches can be further distinguished. A centralized and probabilistic method was proposed in [52], where the probability that the packet is forwarded to a particular next-hop depends on the residual energy of the next-hop or the energy consumed by passing the packet to that next-hop. This calculation is done by each node independently from each other. In contrast to this, a centralized and deterministic approach was proposed in [33], where the paths are calculated by the base station based on the energy consumption of the *entire* path considering the residual energy of *all* nodes on that path.
- The source splits the original data packet into subpackets, adds some redundancy to each subpacket, and then sends each subpacket on one of the n available paths. As it was studied in [15], if some forward error correcting code is applied that corrects k $(k < n)$ errors, then the method is a kind of trade-off between amount of traffic and reliability: even if some of the subpackets were lost, the original message can still be reconstructed due to the added redundancy to each subpacket (i.e., only k subpackets are needed at the destination to reconstruct the original message). In other words, even if the adversary manages to drop $n-k$ subpackets, the packet can still be reconstructed at the base station.

4. Secured Sensor Network Routing Protocols

In this section, we describe the operation of some secure routing protocols in more details. This description differs from the current literature in the sense that it puts more effort to precisely describing the exact format of routing messages that may ease both their formal and informal security analysis. We present two secure topology based routing protocols in Subsections 4.1 and 4.4, a secure geographic routing protocol in Subsection 4.3, and a secure link-state routing protocol in Subsection 4.2. Every protocol description is followed by a short discussion. We emphasize that we are concerned with the security of the route discovery process (i.e., the construction of the routing topology in case of topology-based and link-state routing protocols) and not the security of the data forwarding process in this section.

4.1. Authenticated TinyOS beaconing

As we described in Subsection 2.4.1, TinyOS beaconing is a straightforward method to construct routing topology in sensor networks. Thus, it may serve as a good starting point for a simple secure topology-based routing protocol.

In [46], μTesla is employed to guarantee authenticity of the messages of the base station. In particular, the base station B uses a μTesla key K_i in the ith beaconing interval to provide authenticity for the beacon, where K_i is derived from the corresponding element of the hash chain (see Subsection 3.3.3). The uniform length of every beaconing interval is further denoted by T_{int}[1]. At the beginning of the first beaconing interval, the base station broadcasts the following beacon:

$$\mathsf{B} \rightarrow * \ : \ (\mathsf{BEACON}, Id_{\mathsf{B}}, \mathsf{MAC}_{K_1})$$

where BEACON is a constant message type identifier, Id_{B} is the identifier of the base station, and MAC_{K_1} is a MAC generated on Id_{B} and BEACON using key K_1. A sensor node S receiving this beacon first checks whether $\lfloor \frac{T_{current}+\delta-T_0}{T_{int}} \rfloor < 2$ (i.e., S verifies if the beacon is received in the first beaconing interval), where $T_{current}$ is the local time when the beacon is received, δ is the maximum synchronization error between the base station and the sensor nodes, and T_0 is the start time of the first beaconing interval. If this does not hold, the beacon is silently dropped. Otherwise, the pair $(Id_{\mathsf{B}}, \mathsf{MAC}_{K_1})$ is pushed into a FIFO queue, and S re-broadcasts the beacon after replacing the sender identifier Id_{B} with its own identifier Id_{S}:

$$\mathsf{S} \rightarrow * \ : \ (\mathsf{BEACON}, Id_{\mathsf{S}}, \mathsf{MAC}_{K_1})$$

Every sensor node receiving the beacon performs the same verification process that S has done before.

In the second beaconing interval, B discloses K_1 by including that in the new beacon message. More generally, in the jth beaconing interval $(2 \leq j)$:

[1] It is assumed that all sensor nodes are aware of the key disclosure policy that is in use.

$$\mathsf{B} \rightarrow * : (\mathsf{BEACON}, Id_{\mathsf{B}}, K_{j-1}, \mathsf{MAC}_{K_j})$$

Additionally, each node receiving a beacon during the jth interval selects the first authenticated beacon, which is received during the $(j-1)$th interval, in the FIFO queue, and sets the sender of that beacon as its parent. Finally, similar to the first beaconing interval, S re-broadcasts the current beacon after replacing the sender identifier Id_{B} to Id_{S_i}:

$$\mathsf{S}_i \rightarrow * : (\mathsf{BEACON}, Id_{\mathsf{S}}, K_{j-1}, \mathsf{MAC}_{K_j})$$

4.1.1. Discussion

μTesla is a lightweight and efficient way to provide the source authentication of the base station that prevents the adversary to initiate an unauthorized network-wide flooding. Such unauthorized flooding can easily overload the network and deplete the nodes' batteries. As μTesla is vulnerable to some DoS attacks [39], the authenticated beaconing inherits all these weaknesses.

A more serious problem with this beaconing protocol is the lack of hop-by-hop authentication. Particularly, the adversary can subvert the protocol by simply altering the identifier of the immediate sender in the beacon. For instance, routing loops can also be created similar to the attack against (unauthenticated) TinyOS beaconing described in Subsection 2.4. Moreover, the adversary can broadcast a beacon message with a large enough transmission power (e.g., by using a laptop-class device) by which she convinces the covered nodes that the sender identifier in the beacon is their neighbor. Therefore, data packets may be routed into oblivion using this faked parent identifier. This attack is also referred to as HELLO flood attack in [25].

4.2. INSENS

INSENS [12] is an intrusion-tolerant link-state routing protocol proposed for wireless sensor networks. The protocol consists of three operational phases. In Phase 1, each node determines its neighborlist by overhearing the route request flooded by the base station. When a node has learnt its neighborhood it sends its own neighborlist to the base station which is responsible for computing the routing table for all sensor nodes in the network in Phase 2. Finally, in Phase 3, the base station distributes the computed forwarding tables of all nodes in a secure way. In the following, we describe each operational phase in more details.

INSENS assumes that every node has a single symmetric key that is shared with the base station. Since the base station uses a one-way hash chain to prevent malicious flooding, every node is additionally assumed to store the last element of the chain created by the base station. Moreover, bidirectional communication channels are assumed between each pair of nodes.

4.2.1. Calculation of neighborlist

The base station initiates the routing topology construction by flooding the network with a request message, which has the following format:

$$\mathsf{B} \rightarrow * \ : \ (\mathsf{REQ}, \mathsf{hash}, [id_{\mathsf{B}}])$$

where REQ is a constant message type identifier, hash is the next undisclosed element of the hash chain, and id_{B} is the identifier of the base station. The hash chain mechanism is intended to provide authenticity of the base station, and thus, some defense against DoS attacks.

A node S_1 receiving the request from B checks whether hash is correct (i.e., the commitment of the chain stored at each node can be calculated by iteratively applying the public hash function to value hash). If the message is not originated from the base station, S_1 discards the request. Otherwise, S_1 checks whether the request is fresh (i.e., the most recent element of the chain received by S_1 so far can be calculated from hash by iteratively applying the hash function to hash). If it is not fresh, S_1 adds B to its neighborlist (in case B is not in the neighborlist), and drops the request. If the request is fresh, S_1 stores hash and id_B in its local storage, adds id_{B} to its neighborlist, appends id_{S_1} to the message, and re-broadcasts the modified request:

$$\mathsf{S}_1 \rightarrow * \ : \ (\mathsf{REQ}, \mathsf{hash}, [id_{\mathsf{B}}, id_{\mathsf{S}_1}], \mathsf{MAC}^{\mathsf{REQ}}_{\mathsf{S}_1})$$

where $\mathsf{MAC}^{\mathsf{REQ}}_{\mathsf{S}_1}$ is a message authentication code computed on the elements $[id_{\mathsf{B}}, id_{\mathsf{S}_1}]$, REQ, and hash using the symmetric key shared with the base station.

Every subsequent node S_i receiving request

$$(\mathsf{REQ}, \mathsf{hash}, [id_{\mathsf{B}}, id_{\mathsf{S}_1}, \ldots, id_{\mathsf{S}_{i-1}}], \mathsf{MAC}^{\mathsf{REQ}}_{\mathsf{S}_{i-1}})$$

performs the same verification steps as S_1, but before re-broadcasting the modified request, S_i stores $\mathsf{MAC}^{\mathsf{REQ}}_{\mathsf{S}_{i-1}}$ in conjunction with $id_{S_{i-1}}$ locally and replaces $\mathsf{MAC}^{\mathsf{REQ}}_{\mathsf{S}_{i-1}}$ in the request with $\mathsf{MAC}^{\mathsf{REQ}}_{\mathsf{S}_i}$, which is the MAC generated by S_i on list $[id_{\mathsf{B}}, \ldots, id_{\mathsf{S}_{i-1}}, id_{\mathsf{S}_i}]$, REQ, and hash. Finally, S_i re-broadcasts the following request:

$$\mathsf{S}_i \rightarrow * \ : \ (\mathsf{REQ}, \mathsf{hash}, [id_{\mathsf{B}}, \ldots, id_{\mathsf{S}_{i-1}}, id_{\mathsf{S}_i}], \mathsf{MAC}^{\mathsf{REQ}}_{\mathsf{S}_i})$$

Since duplicate requests are not re-broadcast by sensor nodes and hash chains do not ensure message integrity, the adversary can easily mount a rushing attack by broadcasting a modified request message. However, this attack is confined to the local subtree of the nodes below the adversarial node.

A node can construct its own neighborlist based on these request messages: each node overhears the request messages broadcast by its neighbors, and thus it can identify all neighboring nodes.

4.2.2. Forwarding neighborlist towards the base station

In this phase, each sensor node forwards its computed neighborlist to the base station.

After receiving a request message, a node S_x waits for a specified time before constructing its own neighborlist. If S_x does not receive further request messages during this interval, it closes its neighbor discovery and constructs the neighborlist. Then, S_x sends the following message to S_{x-1} from which it received the first valid request (we recall that S_x has stored $id_{\mathsf{S}_{x-1}}$ and $\mathsf{MAC}^{\mathsf{REQ}}_{\mathsf{S}_{x-1}}$ in Phase 1):

$$\mathsf{S}_x \rightarrow \mathsf{S}_{x-1} \ : \ (\mathsf{NLIST}, \mathsf{hash}, \mathsf{MAC}^{\mathsf{REQ}}_{\mathsf{S}_{x-1}}, \mathsf{Enc}_{\mathsf{S}_x}(path_{\mathsf{S}_x}, neighborlist_{\mathsf{S}_x}), \mathsf{MAC}^{\mathsf{NLIST}}_{\mathsf{S}_x})$$

where the elements of the message are as follows:

- NLIST is a constant message type identifier,
- hash is the hash value of the corresponding request message,
- $\mathsf{MAC}^{\mathsf{REQ}}_{\mathsf{S}_x}$ is the MAC, called parent MAC[2], of S_{x-1} sent in the corresponding request,
- $\mathsf{Enc}_{\mathsf{S}_x}(path_{\mathsf{S}_x}, neighborlist_{\mathsf{S}_x})$ is the neighborhood information and the path information of S_x encrypted by the symmetric key shared with the base station. $neighborlist_{\mathsf{S}_x}$ contains the identifiers of each neighboring node *and* their corresponding $\mathsf{MAC}^{\mathsf{REQ}}$s received in Phase 1. $path_{\mathsf{S}_x}$ is $[id_{\mathsf{S}_x}, \ldots, id_{\mathsf{S}_1}, id_{\mathsf{B}}, \mathsf{MAC}^{\mathsf{REQ}}_{\mathsf{S}_x}]$, which is the reverse of the path received in the corresponding request message including the $\mathsf{MAC}^{\mathsf{REQ}}$ of node S_x,
- $\mathsf{MAC}^{\mathsf{NLIST}}_{\mathsf{S}_x}$ is the MAC computed by node S_x on NLIST, hash, $path_{\mathsf{S}_x}$, and $neighborlist_{\mathsf{S}_x}$.

A node receiving the reply message first checks if $\mathsf{MAC}^{\mathsf{REQ}}_{\mathsf{S}_{x-1}}$ in the reply message equals to its own MAC that has been broadcast with the request containing hash. If it is true, then the node is the parent of S_x, and replaces the parent MAC in the message with its own parent MAC that is stored in Phase 1. Finally, S_{x-1} sends the reply to its parent, which is node S_{x-2}:

$$\mathsf{S}_{x-1} \rightarrow \mathsf{S}_{x-2} \ : \ (\mathsf{NLIST}, \mathsf{hash}, \mathsf{MAC}^{\mathsf{REQ}}_{\mathsf{S}_{x-2}}, \mathsf{Enc}_{\mathsf{S}_x}(path_{\mathsf{S}_x}, neighborlist_{\mathsf{S}_x}), \mathsf{MAC}^{\mathsf{NLIST}}_{\mathsf{S}_x})$$

Every node receiving this message performs the same verifications as S_{x-1} did. After successful verifications, each intermediate node rebroadcasts the reply, which propagates back to the base station in this way. Upon the reception of a reply message

$$(\mathsf{NLIST}, \mathsf{hash}, \mathsf{Enc}_{\mathsf{S}_x}(path_{\mathsf{S}_x}, neighborlist_{\mathsf{S}_x}), \mathsf{MAC}^{\mathsf{NLIST}}_{\mathsf{S}_x})$$

the base station checks whether $\mathsf{MAC}^{\mathsf{NLIST}}_{\mathsf{S}_x}$ is correct, after decrypting $\mathsf{Enc}_{\mathsf{S}_x}(neighborlist_{\mathsf{S}_x})$. If the reply has not been tampered with during the transit, the base station checks the correctness of $neighborlist_{\mathsf{S}_x}$ by verifying the MACs, and the consistency of the neighborlist with the topology computed so far. Note that the $\mathsf{MAC}^{\mathsf{REQ}}$s in $neighborlist_{\mathsf{S}_x}$ can be checked only when the NLIST messages of the corresponding nodes (i.e., the neighbors of S_x) in $neighborlist_{\mathsf{S}_x}$ are also received. If all these verifications succeed, the base station computes the forwarding table for each node using a global centralized algorithm like the one detailed in [12]. When the routing tables are constructed, the base station distributes the forwarding table of each node.

4.2.3. Distributing forwarding tables

The forwarding tables are propagated to respective nodes in a breadth-first manner; first, the immediate neighbors of the base station receive their forwarding tables directly from

[2]In this context, parent node is the next-hop that forwards neighborhood information, instead of measured data, towards the base station.

the base station. Afterwards, these one-hop neighbors forward the forwarding tables of the two-hop neighbors of the base station based on their forwarding tables, and so on.

In particular, the base station first sends the forwarding table of S_1:

$$\mathsf{B} \rightarrow \mathsf{S}_1 \ : \ (\mathsf{FTABLE}, \mathsf{id}_{\mathsf{S}_1}, \mathsf{hash}, \mathsf{Enc}_{\mathsf{S}_1}(ftable_{\mathsf{S}_1}), \mathsf{MAC}_{\mathsf{S}_1}^{\mathsf{FTABLE}})$$

where FTABLE is a constant message type identifier, $\mathsf{Enc}_{\mathsf{S}_1}(ftable_{\mathsf{S}_1})$ is the encrypted form of the forwarding table of S_1, and $\mathsf{MAC}_{\mathsf{S}_1}^{\mathsf{FTABLE}}$ is the MAC generated by B on the complete message. Upon the reception of this message, S_1 checks if $\mathsf{MAC}_{\mathsf{S}_1}^{\mathsf{FTABLE}}$ is correct. If it is true, S_1 sets its forwarding rules according to $ftable_{\mathsf{S}_1}$. Otherwise, S_1 drops the message. When S_1 receives a message from B containing the forwarding table of S_2, S_1 forwards this message to S_2 which must be a neighbor of S_1 according to $ftable_{\mathsf{S}_1}$. In this way, all nodes can receive and set their own forwarding tables coming from the base station.

4.2.4. Discussion

The link-state routing approach has some advantages. Firstly, the base station based on its global view can construct optimal routing topologies for data forwarding. Secondly, intermediate nodes forwarding the link-state information do not need to access the packet in order to derive some routing state or update the routing information (e.g., hop counts) carried by the packet. This implicitly eliminates many potential attacks (e.g., hop-count manipulations). Thirdly, many centralized countermeasures described in Section 3 (like centralized wormhole detection schemes, registration techniques against replication attacks, anomaly detection against Sybil attacks, etc.) can be coupled with the central construction of a routing topology.

On the other hand, the applicability of INSENS is strongly constrained by its centralized nature. In general, centralized algorithms are not desirable in sensor networks for at least two reasons; the base station is the single point of failure in the network (e.g., it can be exposed to DoS attacks even for an outsider adversary) and the nodes in the close vicinity of the base station can be overloaded due to the increased traffic. Moreover, controlling such nodes the adversary can easily cause incomplete (but correct) routing topologies by simply dropping some NLIST, REQ and FTABLE messages. In addition, the length of these messages can be too large, and thus they may cause substantial energy consumption (even more, the message size is constrained to 36 bytes in TinyOS [20]). On the other hand, the incured overhead of the topology construction can be minimized if it is invoked infrequently (i.e., the network does not change frequently). Moreover, if these changes are not significant, they can be securely sent to the base station in an incremental form. Although INSENS is originally designed to support only many-to-one communication between the nodes and the base station, the link-state mechanism presented above can be easily employed to implement various routing topologies supporting any communication patterns.

Finally, we note that the encryption of neighborlists used in INSENS ensures not only confidentiality for neighborhood relations, but it also prevents the adversary to impersonate any honest nodes that are not covered by the transmission range of any adversarial nodes. For instance, if the neighborhood information was not encrypted, an adversarial node could easily retrieve the identifiers and corresponding MACs from an NLIST message passing the adversarial node. Using these retrieved MACs and identifiers, the

adversary could broadcast fabricated REQ messages causing the base station to consider false neighborhood relations for routing topology construction.

4.3. SIGF

SIGF (Secure Implicit Geographic Forwarding) [59] is a configurable secure geographic routing protocol family for wireless sensor networks. SIGF consists of three protocols that are built on each other. SIGF-0 is the first building block towards secure routing; it selects the next-hop non-deterministically and dynamically. SIGF-1 extends SIGF-0 by maintaining and storing reputation information about the neighbors locally. Finally, based on SIGF-1, SIGF-2 provides cryptographic defenses against malicious message manipulations and eavesdropping including replay attacks. Therefore, SIGF-2 requires the most storage, communication and computational effort but it gives the highest security warranty compared to the others. These protocols are a trade-off between the security provided and the state maintained and stored by sensor nodes. In this way, SIGF gives some freedom to a network designer to choose the one among the three protocols that fits most to the current network application considering the possible adversarial threats. Finally, apart from many-to-one and one-to-many communication patterns, the SIGF protocols also support one-to-one communication between each pair of nodes.

4.3.1. SIGF-0

SIGF-0 [59] is based on IGF (Implicit Geographic Forwarding) [5], which is a geographic routing protocol integrated with the MAC (Medium Access Control) layer in order to increase packet delivery ratio and decrease end-to-end delay and overhead. IGF, and thus all protocols in SIGF, determines the next-hop by using the RTS/CTS handshake of the MAC layer.

When the sender S is allowed to send a message in the MAC layer, it broadcasts an Open RTS (ORTS) message containing the location of the sender and the destination:

$$\mathsf{S} \rightarrow * \ : \ (Id_{\mathsf{S}}, \ell_{\mathsf{S}}, Id_{\mathsf{D}}, \ell_{\mathsf{D}}, \mathsf{Area})$$

where Id_{S} and Id_{D} are the identifiers of S and D, resp., ℓ_{S} is the location of S, ℓ_{D} is the location of D, and Area denotes the forwarding area. Those neighbors are eligible to forward the message which are inside Area. In IGF, Area is fixed to a 60° sextant centered on the imaginary line connecting the sender and the destination, but in SIGF protocols this area is adjustable (see below). These neighbors are called candidate nodes. Upon receiving the ORTS message, each candidate node sets a timer inversely proportional to a weighted sum of their distance from the sender, their remaining energy levels, and their perpendicular distances to the line connecting the sender and the destination. When the timer of a candidate node R fires, R responds with a CTS packet to S:

$$\mathsf{R} \rightarrow \mathsf{S} \ : \ (Id_{\mathsf{R}}, \ell_{\mathsf{R}})$$

where Id_{R} and ℓ_{R} are the identifier and the location of the candidate node R, respectively. In IGF, the first node from which the sender receives the CTS packet will be the next-hop, and because other candidates are within the sextant they also hear the first CTS packet

that causes their timers to be cancelled. Then, S forwards the data packet to the selected candidate node:

$$\mathsf{S} \rightarrow \mathsf{R} : (Id_{\mathsf{S}}, Id_{\mathsf{R}}, \ell_{\mathsf{R}}, \mathsf{Data})$$

where Data is the measured data to be sent to the base station. Network voids are treated by shifting the forwarding sextant to the side and rebroadcasting a new ORTS message.

SIGF-0 introduces probabilistic defenses to alleviate the following major problems of IGF:

1. *One responder selection*: Selecting only one responder renders IGF vulnerable to several attacks (e.g., an adversarial node can respond before all honest nodes not following the protocol rules and causing a special CTS rushing attack). SIGF-0 offers two additional adjustments apart from the one responder scheme; it can wait until a fixed multiple number of CTS responses are received, or if not enough responses are received, the sender can dynamically increase the collection window of CTS responses.
2. *First responder selection*: The next-hop relay is always the first node that responds to the ORTS message in IGF. This results in similar problems to the one responder selection scheme (i.e., the adversary can mount CTS rushing attack). By contrast, SIGF-0 offers priority selection, random selection, and multiple selection of the next-hop relay with keeping the option of the first responder selection which gives the best performance when no adversary is presented. Priority selection means that a sender node selects the next-hop relay which makes the greatest progress towards the final destination. Random selection equals to selecting the next-hop relay randomly among all candidate nodes (hence, as the set of the candidate nodes increase the less likely that an adversarial node will be selected), while multiple selection is simply forwarding the message to multiple candidate nodes (i.e., employing multipath routing) to achieve some robustness.
3. *Size of the forwarding area*: If node density is low and the Area is set to 60° sextant it can occur that the adversarial nodes fill the candidate set causing honest nodes to be excluded from the forwarding process. Thus, the network designer using SIGF-0 can optionally increase the size of the forwarding area.
4. *Location manipulation*: An adversarial node can not only mount rushing attack by responding immediately to the ORTS message, but it can also manipulate the next-hop decision by claiming false location coordinates (e.g., the adversary appears to be closer to the destination than any other honest nodes). Thus, the location information can be simply omitted in SIGF-0.

Although some messages may need to be buffered in SIGF-0 until the end of the CTS collection window, they can be immediately discarded after sending the message. Thus, SIGF-0 is still considered to be a stateless protocol. As one can see, the level of security is adjustable at the expense of degraded routing efficiency in SIGF-0.

4.3.2. SIGF-1

Building on SIGF-0, SIGF-1 [59] uses a simple reputation scheme against selective forwarding attacks (and thus, against black- and grayhole attacks as well). Each node stores some limited information about itself and its neighbors that renders SIGF-1 a stateful

routing protocol. On the other hand, SIGF-1 delivers more packets on average in the presence of adversarial nodes.

Each node A maintains a local variable I_{A} that is incremented each time when a message is sent. In addition, each node also has a small buffer that is used to store recently relayed messages. For every observed neighbor C, the following numbers are stored by A: C_{sent} is the number of total messages sent to C by A, $C_{forward}$ is the total number of messages forwarded by C on behalf of A, $C_{location}$ is the last claimed location of C, and C_{delay} is the average delay between sending a message to C and overhearing C relaying the same message. Therefore, a message is stored by A until the corresponding node is overheard sending that message, when C_{delay} and $C_{forward}$ is updated accordingly. If the buffer is full, the oldest message is dropped and the associated C_{delay} is fixed to a maximum value D.

Then, node A derives the following values that are further used to calculate the reputation of node C;

- $C_{success} = \frac{C_{forward}}{C_{sent}}$ is the forwarding success ratio
- $C_{fairness} = \frac{I_{\mathsf{A}} - C_{sent}}{I_{\mathsf{A}}}$ is the forwarding fairness ratio. It measures the distribution of the next-hop relay selections among the candidate nodes.
- $C_{consistency}$ that reflects the variance of node C's claimed location coordinates. Assuming that honest nodes move moderately, $C_{consistency}$ is decreased when C's location changes (i.e., if the node constantly moves or claims false locations, it is penalized). In turn, if the claimed location remains constant for a while, some redemption is given by increasing $C_{consistency}$.
- $C_{performance} = \frac{D - C_{delay}}{D}$ denotes the forwarding performance of C regarding the maximum delay D. When C consistently delivers packets with short delays, this value is increased. However, if it shows high delay due to packet dropping or deliberately delayed message forwarding, C is penalized by decreasing $C_{performance}$.

All listed values are initialized to 0.5, and vary between 0 and 1.

Finally, the reputation value R_{C} maintained by A with respect to C is calculated by

$$R_{\mathsf{C}} = \alpha C_{success} + \beta C_{fairness} + \gamma C_{consistency} + \delta C_{performance}$$

where α, β, γ, and δ are tunable weighting factors. Every term in the expression and the reputation value itself must be between 0 and 1. The reputation value is not shared with any nodes, it is stored locally. If node C responds to an ORTS message of A, and R_{C} falls below a certain threshold, C is removed from the set of candidate nodes. If there is no node above the reputation threshold, the node with the highest reputation value is chosen. This threshold with the other predefined values like D, α, β, γ, and δ are system parameters that can be set according to the network designer's choice. In this way, SIGF-1 offers another trade-off between security and routing efficiency.

4.3.3. SIGF-2

The last and most secure member of SIGF family is SIGF-2 [59] that uses symmetric key cryptography among one-hop neighbors to guarantee message authentication and confidentiality with freshness. Each node is required to store a symmetric key that is distributed by some key pre-distribution scheme. For instance, LEAP [66] may be a good

candidate as it provides a pairwise shared key between each pair of neighbors as well as a local broadcast key used to authenticate broadcast messages sent to immediate neighbors.

In more details, SIGF-2 extends SIGF-1 by adding the following security features:

- *Message authentication*: The network designer can choose between computing MAC (Message Authenticion Code) (i) on every message, (ii) only on data messages, and (iii) not using MACs at all.
- *Message confidentiality*: The payload of messages can be (i) completely encrypted in order to prevent traffic analysis and illegal data access, or (ii) not encrypting the payload at all.
- *Freshness*: Message sequencing can be used to prevent the adversary from reordering of message sequences and replaying obsolete packets by attaching a monotonically increasing integer to each packet per neighbor.

These countermeasures also prevent the outsider adversary to manipulate routing messages. The effect of insider attacks can be mitigated by using the defense mechanisms of SIGF-1.

4.3.4. Discussion

SIGF is the first family of secure routing protocols in sensor networks that tries to provide a configurable security framework for sensor network routing. SIGF mitigates selective forwarding as well as black and grayhole attacks by using the described reputation scheme in SIGF-1 and by employing the optional multiple or random next-hop selection in SIGF-0. SIGF defends against Sybil attacks by using the pairwise symmetric keys in SIGF-2. Furthermore, the DoS attacks caused by replaying old ORTS and/or CTS messages are prevented by employing message sequencing and message authentication in SIGF-2 with the random next-hop selection in SIGF-0. The authors also emphasized in [59] that their primary intention was to design a fully configurable and efficient instead of *completely* secure family of routing protocols. This means that members of SIGF are not the "best" but "good enough" secure routing protocols.

The main problem with SIGF might be the lack of source authentication (i.e., the authentication of the originator of messages). SIGF only provides hop-by-hop authentication that is not sufficient to prevent the adversary from diverting the traffic in order to shorten the network lifetime or increase the network delay. In particular, a similar attack to the one described in Subsection 2.4 against GPSR also works here: If a corrupted node receives a packet it can alter the destination location carried by the packet that may cause it to go along a detour. Note that the reputation mechanism used by SIGF-1 cannot detect this malicious behaviour. On the other hand, current mechanisms (e.g., broadcast authentication of sender nodes) to provide source authentication may yield significant storage, communication or computational cost.

4.4. Secured directed diffusion

Secure Diffusion [62] is a secure variant of the Directed Diffusion communication paradigm [23]. Secure Diffusion uses μTesla to guarantee authenticity for interest propagation. It also employs a novel security primitive, called location-binding key [62], to enable the sensor nodes to authenticate the sensed data. Additionally, each node shares pairwise keys with each of its one-hop neighbors to ensure hop-by-hop authenticity. The

protocol naturally assumes that every node is aware of its own location coordinates. Furthermore, the authors did not consider such powerful adversary like the one described in Subsection 2.1. Namely, the adversary is assumed not to have more powerful radio equipment than an ordinary sensor node. Finally, similar to the LEAP assumption [66] it is also assumed that each node has sufficiently long time in the bootstrap phase to perform some key agreement protocol to derive necessary pairwise shared keys and location-binding keys. The authors in [62] also proposed another solution for deriving pairwise shared and location-binding keys, in case this assumption cannot be made (i.e., the adversary can compromise nodes right after their deployment), which uses the assistance of the base station[3] to agree on pairwise symmetric keys. More interested readers are referred to [62].

Initially, the network field is divided into a geographical grid, and multiple keys are bound to each cell on this grid. Each node establishes two types of keys:

- *Location-binding keys*: For each node, this key is bound to the cells that are covered by the sensing range of the node (a cell is considered to be covered by a sensor node, if at least one of the four corners of the cell is covered by the node's sensing range). This key is employed for the authentication of the sensed data. The purpose is that nodes who sense the same cell should ideally derive different keys for that single cell in order to increase security (i.e., nodes sensing the same phenomena can provide sufficient authenticity independently from each other). However, in practice, this would incur communication overhead. Thus, the authors propose a probabilistic scheme for deriving location-binding keys. Initially, the network field is divided into a pre-defined grid, which is defined by the cell size c and an arbitrary reference point (x_0, y_0). Thus, a cell i within the grid can be described by its center coordinates (x_i, y_i), where $x_i = x_0 + i \cdot c$ and $x_i = x_0 + i \cdot c$ $(i, j = 0, \pm 1, \pm 2, \ldots)$. Each node derives a key by hashing the coordinates of the cell with a random number r ranging from 1 to L:

  $$K_{x_i, y_i, r} = h_{K_M}(x_i, y_i, r)$$

  where h is a secure one-way hash function, and K_M is the pre-deployed master key that is shared with all nodes in the boostrapping phase similar to LEAP [66]. As each node selects r at random independently from each other, nodes sensing the same cell will use different keys with high probability depending on the value of L.

  The rationale behind the usage of location-binding keys is that the base-station usually disregards the identity of the node who originates the data, it rather wants to know *where* the data originates from. A node generating a valid MAC using a location-binding key can prove that it is indeed in the vicinity of the sensed phenomena. Compared to traditional keys bound to the identity, this technique may be more beneficial due to its less incured management cost. Moreover, location-binding keys can be computed independently by individual nodes.
- *Neighbor key*: Each node agrees on a common pairwise key shared with its neighbors. The key is bound to the location of both nodes, and it is used to ensure

[3]We consider the single base station scenario for the sake of simplicity.

hop-by-hop authentication for local messages. For instance, if node A and node B hears each other, they can derive a pairwise key in the following way:

$$K_{\mathsf{A,B}} = h_{K_M}(Id_{\mathsf{A}}, \ell_{\mathsf{A}}, Id_{\mathsf{B}}, \ell_{\mathsf{B}})$$

where Id_{A}, Id_{B} and $\ell_{\mathsf{A}}, \ell_{\mathsf{B}}$ are the local identifiers and location coordinates (i.e., ℓ_A as well as ℓ_B consists of two coordinates) of node A and B, respectively.

4.4.1. Secure neighbor discovery

Secure neighbor and cell discovery can be performed by using the pre-deployed master key in the bootstrapping phase. For instance, node A can broadcast a HELLO message to indicate its presence:

$$\mathsf{A} \rightarrow * \ : \ (\mathsf{HELLO}, Id_{\mathsf{A}}, \ell_{\mathsf{A}}, \mathsf{MAC}^{\mathsf{HELLO}}_{K_M})$$

where HELLO is a constant message type identifier, and $\mathsf{MAC}^{\mathsf{HELLO}}_{K_M}$ is the MAC computed on HELLO, Id_{A}, ℓ_{A} using the master key K_M.

If a node G is deployed after bootstrapping, it can also compute the pairwise key shared with its neighbors. The details can be found in [62].

When the bootstrapping phase terminates, the master key is erased from the memory of all nodes except the base station.

4.4.2. Secure interest propagation

The base station disseminates the queries (interests) using μTesla described in Subsection 3.3.3. The interest contains the description of the queried data, and the data-rate at which the data should be sent towards the base station. A gradient, which is set by each node receiving this interest, points to the next-hop for which the data described in the interest should be forwarded at the specified rate. Since the next-hop information is not protected by μTesla, the pairwise keys are used to authenticate the immediate sender of the interest. In particular, if node A receives an interest denoted by $interest$, it forwards that to its neighbor B:

$$\mathsf{A} \rightarrow \mathsf{B} \ : \ (\mathsf{INTEREST}, Id_{\mathsf{A}}, interest, \mathsf{MAC}^{\mathsf{INTEREST}}_{K_{\mathsf{A,B}}})$$

where $\mathsf{MAC}^{\mathsf{INTEREST}}_{K_{\mathsf{A,B}}}$ is the MAC computed on $interest$, Id_{A} and INTEREST with the pairwise key $K_{\mathsf{A,B}}$ shared between A and B. Receiving this interest, B verifies whether the carried $\mathsf{MAC}^{\mathsf{INTEREST}}_{K_{\mathsf{A,B}}}$ is correct. In addition, B also checks the authenticity of $interest$, which is protected by μTesla. If one of these verifications is incorrect, B drops the interest. Otherwise, if B cannot respond to the query, B forwards it as A did previously. The reason for using pairwise shared key is that an interest can serve as a reinforcement for some specific neighbors instead of all of them (i.e., a node selects a particular neighbor according to some network metric, see later in Subsection 4.4.4).

4.4.3. Secure data reporting

When a sensor has some data to report, it uses the corresponding location-binding key to authenticate the data. As a specific example, let us assume that node A wishes to send $data$, which is sensed in the cell (x_j, y_j), to the base station. A sends the following message to node B for which A maintains a gradient matching to $data$ (e.g., $data$ may be composed of the coordinates, the timestamp, the type and the intensity of the sensed data):

$$\mathsf{A} \rightarrow \mathsf{B} \ : \ (\mathsf{DATA}, Id_{\mathsf{A}}, data, \mathsf{MAC}^{\mathsf{DATA}}_{K_{x_j,y_j,r}}, \mathsf{MAC}^{\mathsf{NEIGHBOR}}_{K_{\mathsf{A,B}}})$$

where DATA is a constant message type identifier, $K_{x_j,y_j,r}$ is the location-binding key of A with respect to cell (x_j, y_j), and $\mathsf{MAC}^{\mathsf{DATA}}_{K_{x_j,y_j,r}}$ is the MAC generated on $data$ using key $K_{x_j,y_j,r}$. Finally, $\mathsf{MAC}^{\mathsf{NEIGHBOR}}_{K_{\mathsf{A,B}}}$ is the MAC generated on the same elements including $\mathsf{MAC}^{\mathsf{DATA}}_{K_{x_j,y_j,r}}$ and Id_{A} with the pairwise key $K_{\mathsf{A,B}}$. We note that A sends a similar message to all its neighbors for which it maintains a gradient matching to $data$.

Upon the reception of this message, B checks whether $\mathsf{MAC}^{\mathsf{NEIGHBOR}}_{K_{\mathsf{A,B}}}$ is correct. If it is not, B silently drops the packet. Otherwise, it stores the message in its local cache, and begins to flow the data towards the base station according to the gradients that it have. For instance, forwarding the message to C, B replaces $\mathsf{MAC}^{\mathsf{NEIGHBOR}}_{K_{\mathsf{A,B}}}$ with $\mathsf{MAC}^{\mathsf{NEIGHBOR}}_{K_{\mathsf{B,C}}}$, the sender identifier to Id_{B}, and sends the resulted message to C:

$$\mathsf{B} \rightarrow \mathsf{C} \ : \ (\mathsf{DATA}, Id_{\mathsf{B}}, data, \mathsf{MAC}^{\mathsf{DATA}}_{K_{x_j,y_j,r}}, \mathsf{MAC}^{\mathsf{NEIGHBOR}}_{K_{\mathsf{B,C}}})$$

If B receives multiple copies of the *same* event, it can merge all copies into a single report by concatenating all MACs, which are different with high probability. Thus, this new report will contain the data that is identical in all copies, and the ensemble of the MACs generated by the nodes that sensed the event. The rationale behind this is that these multiple MACs generated on the same event bear more credibility than a single report with a single-source MAC, and this new report is trivially shorter than the sum of the original ones.

The above forwarding process repeats until the message reaches the base station which then verifies the authenticity of $data$ by checking $\mathsf{MAC}^{\mathsf{DATA}}_{K_{x_j,y_j,r}}$. Particularly, we recall that the base station has key K_M and the received $data$ containing (x_j, y_j). Since r is generated at random ranging from 1 to L, maximum L keys can be bound to a single cell. Thus, upon the verification of a $\mathsf{MAC}^{\mathsf{DATA}}_{K_{x_j,y_j,r}}$, the base station can regenerate all L different MACs, and can compare these to $\mathsf{MAC}^{\mathsf{DATA}}_{K_{x_j,y_j,r}}$.

4.4.4. Secure reinforcement

As one can observe, the sensed data can flow towards the base station on multiple path. Based on Directed Diffusion, Secure Diffusion also applies some reinforcement strategy to select the empirically best path. Another aim of secure reinforcement is to select such paths which contain no adversarial nodes.

Before sending any reinforcement interests, the base station first checks from which neighbors it received correct data messages (i.e., messages with correct MACs). Moreover, it can also happen that the base station receives different data from the same cell

which may indicate that there is a corrupted node in the network. Therefore, each neighbor of the base station can be classified as either safe or unsafe node according to the correctness of the received messages. A neighbor is safe if it has not delivered incorrect messages at all. If a neighbor has sent at least one incorrect message it is categorized as an unsafe node. Another problem is that a corrupted node may not send consistently correct messages but may alternate between sending correct and incorrect messages which makes its elimination more difficult.

Hence, if the base station has any safe neighbors, it chooses the best one among them according to some network metric (e.g., smallest delay) similar to Directed Diffusion. Then, this neighbor reinforces its own best neighbor from which it received the specified data and so forth. Thus, an adversarial node, which deliberately sends incorrect messages, cannot be reinforced in this way.

If the base station has no safe neighbor, it selects the best among the unsafe nodes. Then, it sends a recently received correct message to this node. Each neighbor can select its own neighbors from which it received this correct message based on the maintained local cache. Then, it can make a list from these neighbors ordered by some network metric, where the first node in the list is the empirically best one. However, it may still happen that this path contains some corrupted nodes. Thus, each selectively reinforced node employs a round-robin strategy to maintain this list described previously. In particular, when it receives a new selective reinforcement message it moves the first node to the end of the list, then it reinforces the new first node. This procedure also ensures that the consecutively reinforced paths are disjoint. Therefore, the paths that contain corrupted nodes are less frequently reinforced than those ones which contain only honest nodes, since corrupted nodes are assumed to deliver incorrect messages in the future. More details can be found in [62].

4.4.5. Discussion

Secure Diffusion is the first routing protocol that uses location-binding symmetric keys instead of traditional symmetric keys bound to the identity of sensor nodes. Location-binding keys fit perfectly to most applications used in large-scale sensor networks, where the thousands of identity-based keys would incur significant management cost at the base station.

The adversary aiming to inject fabricated data must have the corresponding location key in order to authenticate the false data. Without knowing the master key, this is only possible if she captures some honest nodes that cover the particular cell. Even if the adversary is assumed to do that, she must capture as many nodes as possible in the close vicinity of the cell. Otherwise, the honest nodes around the cell can report correct data about the same event that causes the base station to detect the misdeed. Another option might be for the adversary to capture nodes or deploy adversarial nodes (not necessarily around the cell) that control all traffic originating from the particular cell .

However, the source authentication of the sensed data used in Secure Diffusion prevents intermediate nodes to perform data aggregation. The problem is that the base station should always verify the MACs generated on the original data by the source nodes, however, the original data is no more accessible after some aggregations. Thus, in order to correctly verify the source MACs, the intermediate node should leave the original data in the packet that would make in-network aggregation useless.

Since μTesla is employed to authenticate interests, the adversary cannot misuse the reinforcement strategy to divert the traffic. Moreover, the secured reinforcement strategy detailed in Subsection 4.4.4 ensures that sooner or later misbehaving adversarial nodes will be eliminated from all routes between the source nodes and the base station.

As the authors discussed in [62] Secure Diffusion does not update or revoke compromised location-binding keys. When deploying new nodes to an area where compromised nodes have already installed by the adversary, the new nodes may derive such keys that are already possessed by the compromised nodes. The problem is that every node trusts the master key, which is though not updated.

5. Conclusions and Outlook

The security of routing protocols is a fundamental issue in sensor networks. Due to the limitations of the tiny sensor devices, the attacks mounted against the routing service can have devastating effects in wireless sensor networks. Moreover, some of the standard techniques that have been used to defend against these typical routing attacks so far are no longer desirable in sensor networks because of their high resource demands. Hence, designing secure routing protocols in wireless sensor networks is a challenging task.

In this chapter, we gave an up-to-date picture of the security problems of routing protocols in sensor networks. In particular, in the first part of the chapter, we described the sensor routing specific adversary model with its primary objectives, and we also listed some possible attack methods used by this adversary to achieve her objectives. We demonstrated the severity of these attacks on real protocols: we showed how the adversary can subvert TinyOS beaconing, GPSR and Directed Diffusion along with simple attack scenarios. Then, we presented some countermeasures against the described attacks; we focused on link-layer security, secure neighbor discovery, broadcast authentication, and robust data delivery. In the second part of the chapter, we illustrated how these efficient defense mechanisms can be used by the secure sensor network routing protocols. We detailed the operation of two secure topology-based routing protocols (Authenticated TinyOS beaconing, Secure Diffusion), a secure link-state routing protocol (INSENS) and a family of secure geographic routing protocols (SIGF). We also provided a short discussion about the advantages and disadvantages of each secure routing protocol.

Although there is an on-going research effort on designing new secure routing protocols for wireless sensor networks, the number of *secure* sensor network routing protocols is still limited. Finally, we note that the security analysis of these protocols has been done by only informal reasoning so far, however, precise and mathematically rigorous analysis would also be needed.

Acknowledgements

The work described in this chapter is based on results of IST FP6 STREP UbiSec&Sens (`http://www.ist-ubisecsens.org`). UbiSec&Sens receives research funding from the European Community's Sixth Framework Programme. Apart from this, the European Commission has no responsibility for the content of this work. The information in this document is provided as is and no guarantee or warranty is given that the infor-

mation is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. The presented work has been further supported by the HSN Lab.

References

[1] G. Ács, L. Buttyán, and I. Vajda. Modelling adversaries and security objectives for routing protocols in wireless sensor networks. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2006.

[2] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11:6–28, 2004.

[3] K. Balakrishnan, J. Deng, and P. K. Varshney. Twoack: preventing selfishness in mobile ad hoc networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2137–2142, 2005.

[4] R. Blom. Non-public key distribution. In *Advances in Cryptology (Crypto)*, pages 231–236, 1982.

[5] B. Blum, T. He, S. Son, and J. Stankovic. IGF: A state-free robust communication protocol for wireless sensor networks. Technical Report CS-2003-11, University of Virginia, Charlottesville, 2003.

[6] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In *Advances in Cryptology (Crypto)*, pages 471–486, 1992.

[7] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[8] S. Brands and D. Chaum. Distance-bounding protocols. In *Proceedings of the Workshop on Theory and Application of Cryptographic Techniques*, 1993.

[9] P. Bryan, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2005.

[10] L. Buttyán, L. Dóra, and I. Vajda. Statistical wormhole detection in sensor networks. In *Proceedings of the European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS)*, 2005.

[11] D. Coppersmith and M. Jakobsson. Almost optimal hash sequence traversal. In *Proceedings of Financial Cryptography (FC)*, 2002.

[12] J. Deng, R. Han, and S. Mishra. INSENS: Intrusion-tolerant routing in wireless sensor sensor networks. Technical Report CU-CS-939-02, Department of Computer Science, University of Colorado, 2002.

[13] J. R. Douceur. The sybil attack. In *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[14] W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, pages 42–51, 2003.

[15] S. Dulman, T. Nieberg, J. Wu, and P. Havinga. Trade-off between traffic overhead and reliability in multipath routing for wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2003.

[16] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, pages 41–47, 2002.

[17] K. Gabriel and R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[18] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly resilient, energy efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review (MC2R)*, 8(2), 2001.

[19] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS)*, 2000.

[20] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2000.

[21] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *Proceedings of the IEEE Symposium on Network and Distributed Systems Security (NDSS)*, 2004.

[22] Y.-C. Hu, A. Perrig, and D. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *Proceedings of the IEEE Conference on Computer Communications (Infocom)*, 2003.

[23] C. Intanagonwiwata, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 55–67, 2000.

[24] C. Karlof, N. Sastry, and D. Wagner. TinySec: A link layer security architecture for wireless sensor networks. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[25] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks*, 1, 2003.

[26] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless sensor networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[27] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic routing made practical. In *Proceedings of the ACM Symposium on Networked Systems Design and Implementation (NSDI)*, pages 217–230, 2005.

[28] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. Lazy cross-link removal for geographic routing. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[29] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[30] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 267–278, 2003.

[31] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, 1981.

[32] L. Lazos, R. Poovendran, C. Meadows, P. Syverson, and L. Chang. Preventing wormhole attacks on wireless ad hoc networks: a graph theoretic approach. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2005.

[33] Q. Li, J. Aslam, and D. Rus. Hierarchical power-aware routing in sensor networks. In *Proceedings of the DIMACS Workshop on Pervasive Networking*, 2001.

[34] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, pages 52–61, 2003.

[35] D. Liu and P. Ning. Multi-level μTESLA: Broadcast authentication for distributed sensor networks. *ACM Transactions in Embedded Computing Systems (TECS)*, 3(4):800–836, 2004.

[36] D. Liu, P. Ning, S. Zhu, and S. Jajodia. Practical broadcast authentication in sensor networks. In *Proceedings of the International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous)*, pages 118–129, 2005.

[37] A. Manjeshwar and D. P. Agarwal. TEEN: a routing protocol for enhanced efficiency in wireless sensor networks. In *Proceedings of the International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, 2001.

[38] A. Manjeshwar and D. P. Agarwal. APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. In *Proceedings of the Parallel and Distributed Processing Symposium (IPDPS)*, pages 195–202, 2002.

[39] L. Mark, A. Perrig, and B. Whillock. Seven cardinal properties of sensor network broadcast authentication. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2006.

[40] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[41] R. C. Merkle. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1980.

[42] R. C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology (Crypto)*, 1988.

[43] V. Miller. Uses of elliptic curves in cryptography. In *Advances in Cryptology (CRYPTO)*, pages 417–462, 1985.

[44] J. Newsome, R. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004.

[45] Next-Generation Secure Computing Base (NGSCB). `http://www.microsoft.com/resources/ngscb/default.mspx`, 2003.

[46] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. *Wireless Networks Journal (WINE)*, 8, 2002.

[47] K. Piotrowski, P. Langendoerfer, and S. Peter. How public key cryptography influences wireless sensor node lifetime. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2006.

[48] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: a geographic hash table for data-centric storage. In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[49] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *CACM 21,2*, 1978.

[50] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, 2003.

[51] Y. Sella. On the computation-storage trade-offs of hash chain traversal. In *Proceedings of Financial Cryptography (FC)*, 2003.

[52] R. C. Shah and J. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2002.

[53] Trusted Computing Group (TCG). `https://www.trustedcomputinggroup.org/`, 2003.

[54] G. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.

[55] S. Čapkun, L. Buttyán, and J.-P. Hubaux. SECTOR: Secure tracking of node encounters in multi-hop wireless networks. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2003.

[56] S. Čapkun and J.-P. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *Proceedings of the IEEE Conference on Computer Communications (Infocom)*, 2005.

[57] S. Čapkun and J.-P. Hubaux. Secure positioning in wireless networks. *IEEE Journal on Selected Areas in Communications*, February 2006.

[58] W. Wang and B. Bhargava. Visualization of wormholes in sensor networks. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, 2004.

[59] A. D. Wood, L. Fang, J. A. Stankovic, and T. He. SIGF: A family of configurable, secure routing protocols for wireless sensor networks. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2006.

[60] ANSI X9.63. The elliptic curve digital signature algorithm (ECDSA), 1999.

[61] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2005.

[62] H. Yang, S. Wong, S. Lu, and L. Zhang. Secure diffusion for wireless sensor networks. In *IEEE International Conference on Broadband Communications, Networks, and Systems (Broadnets)*, 2006.

[63] H. Yih-Chun, M. Jakobsson, and A. Perrig. Efficient constructions for one-way hash chains. In *Proceedings of the Conference of Applied Cryptography and Network Security (ACNS)*, 2005.

[64] Y. Yu, D. Estrin, and R. Govindan. Geographical and energy-aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical Report TR-01-0023, University of California, Los Angeles, Computer Science Department, 2001.

[65] Q. Zhang, P. Wang, D. S. Reeves, and P. Ning. Defending sybil attacks in sensor networks. In *Proceedings of the International Workshop on Security in Distributed Computing Systems (SDCS)*, pages 185–191, 2005.

[66] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, 2003.

Wireless Sensor Network Security
J. Lopez and J. Zhou (Eds.)
IOS Press, 2008

Secure Data Aggregation in Wireless Sensor Networks

Sanjeev SETIA [a], Sankardas ROY [b] and Sushil JAJODIA [b]
[a] *Computer Science Department, George Mason University, Fairfax, VA, USA*
[b] *Center for Secure Information Systems, George Mason University*

Abstract. In many sensor applications, the data collected from individual nodes is aggregated at a base station or host computer. To reduce energy consumption, many systems also perform in-network aggregation of sensor data at intermediate nodes enroute to the base station. Most existing aggregation algorithms and systems do not include any provisions for security, and consequently these systems are vulnerable to a wide variety of attacks. In particular, compromised nodes can be used to inject false data that leads to incorrect aggregates being computed at the base station. We discuss the security vulnerabilities of data aggregation systems, and present a survey of robust and secure aggregation protocols that are resilient to false data injection attacks.

Keywords. Sensor networks, aggregation, security

1. Introduction

Sensor networks are increasingly deployed for applications such as wildlife habitat monitoring, forest fire prevention, and military surveillance [19,21,25]. In these applications, the data collected by sensor nodes from their physical environment needs to be assembled at a host computer or data sink for further analysis. Typically, an aggregate (or summarized) value is computed at the data sink by applying the corresponding aggregate function, e.g., `MAX, COUNT, AVERAGE` or `MEDIAN` to the collected data.

In large sensor networks, computing aggregates *in-network*, i.e., combining partial results at intermediate nodes during message routing, significantly reduces the amount of communication and hence the energy consumed. An approach used by several data acquisition systems [16,31] for sensor networks is to construct a spanning tree rooted at the data sink, and then perform in-network aggregation along the tree. Partial results propagate level-by-level up the tree, with each node awaiting messages from all its children before sending a new partial result to its parent. Researchers have designed several energy-efficient algorithms [16,28] for computing aggregates using the tree-based approach.

Tree-based aggregation approaches, however, are not robust to communication losses which result from node and transmission failures and are relatively common in sensor networks [16,34,35]. Because each communication failure loses an entire subtree of readings, a large fraction of sensor readings are potentially unaccounted for at the data sink, leading to a significant error in the aggregate computed. To address this problem,

researchers have proposed novel algorithms that work in conjunction with multi-path routing for computing aggregates in lossy networks [6,18,20]. In particular, a robust and scalable aggregation framework called *Synopsis Diffusion* has been proposed for computing aggregates such as `COUNT`, `SUM`, `UNIFORM SAMPLE` and `MOST FREQUENT ITEMS` [18,20].

Unfortunately, none of the above algorithms or systems include any provisions for security; as a result, they are vulnerable to many attacks that can be launched by unauthorized or compromised nodes. To prevent *unauthorized nodes* from eavesdropping on or participating in communications between legitimate nodes, we can augment the aggregation and data collection systems with any one of several recently proposed authentication and encryption protocols, e.g., [23,37]. However, securing aggregation systems against attacks launched by *compromised nodes* is a much more challenging problem since standard authentication mechanisms cannot prevent *insider* attacks launched by a compromised node.

Compromised nodes can be used to launch a wide variety of insider attacks that disrupt the operation of the sensor application and network [24,22]. In this chapter, however, we focus on an important class of insider attacks in which the adversary uses compromised nodes to inject malicious data into the network within the framework of a data aggregation system. In particular, we discuss the vulnerabilities of existing data aggregation approaches to such attacks, and present a survey of secure aggregation techniques that are designed to be resilient to such attacks. We also discuss the closely related problem of false data injection in sensor networks in general, and describe an approach that can be used to prevent this attack.

2. Security Challenges

Data acquisition systems for sensor networks can be classified into two broad categories on the basis of the data collection methodology employed for the application:

Query-based systems In query-based systems, the base station (the data sink) broadcasts a query to the network and the nodes respond with the relevant information. Messages from individual nodes are potentially aggregated enroute to the base station. Finally, the base station computes one or more aggregate values based on the messages it has received.

In some applications, queries may be persistent in nature resulting in a continuous stream of data being relayed to the data sink from the nodes in the network. For such applications, the query broadcast by the base station specifies a period (referred to as an epoch); nodes in the network send their readings to the base station after each epoch.

Event-based systems In event-based applications, such as perimeter surveillance and biological hazard detection, nodes send a message to the base station station only when the target event occurs in the area of interest. If multiple reports being relayed correspond to the same event, they can be combined by an intermediate node on the route to the base station.

Data acquisition systems can also be categorized based on how sensor data is aggregated. In single-aggregator approaches, aggregation is performed only at the data sink.

In contrast, hierarchical aggregation approaches make use of in-network aggregation. Hierarchical aggregation schemes can be further classified into tree-based schemes and ring-based schemes on the basis of the topology into which nodes are organized.

As discussed in the introduction, most existing data management and acquisition systems for sensor networks are vulnerable to security attacks launched by malicious parties. Sensor nodes are often deployed in unattended environments, so they are vulnerable to physical tampering. Since current sensor nodes lack hardware support for tamper-resistance, it is relatively easy for an adversary to compromise a node without being detected. The adversary can obtain confidential information (e.g., cryptographic keys) from the compromised sensor and reprogram it with malicious code. Moreover, the attacker can replicate the compromised node and deploy the replicas at various strategic locations in the network [22].

A compromised node can be used to launch a variety of security attacks. These attacks include jamming at the physical or link layer as well as other resource consumption attacks at higher layers of the network software. Compromised nodes can also be used to disrupt routing protocols and topology maintenance protocols that are critical to the operation of the network [14,11]. In this chapter, however, we focus on attacks that target the data acquisition protocol used by the application. Specifically, we discuss attacks in which the compromised nodes send malicious data in response to a query (or send false event reports in event-based systems). By using a few compromised nodes to render suspect the data collected at the sink, an adversary can effectively compromise the integrity and trustworthiness of the entire sensor network.

In event-based systems, compromised nodes can be used to send false event reports to the base station with goal of raising false alarms and depleting the energy resources of the nodes in the network. We refer to this attack as the *false data injection attack*. In Section 3, we discuss an approach for filtering the false data injected by malicious nodes.

Similarly, in query-based systems, compromised nodes can be used to inject false data into the network with the goal of introducing a large error in the aggregate value computed at the data sink. The aggregate computed by the sink is erroneous in the sense that it differs from the true value that would have been computed if there were no false data values included in the computation. Unlike event-based systems, however, in aggregation systems the effectiveness of a false data injection attack depends on both the aggregate being computed, e.g., `MAX` or `MEDIAN`, and whether sensor data is aggregated enroute to the data sink or only at the data sink, and thus the techniques used for preventing this attack differ from the techniques used in event-based systems.

In an aggregation system, a compromised node M can corrupt the aggregate value computed at the sink in four ways. First, M can simply drop aggregation messages that it is supposed to relay towards the sink. This has the effect of omitting a large fraction of sensor readings being aggregated. Second, M can alter a message that it is relaying to the data sink. Third, M can falsify its own sensor reading with the goal of influencing the aggregate value. Fourth, in systems that use in-network aggregation, M can falsify the sub-aggregate which it is supposed to compute based on the messages received from its child nodes.

The first attack in which a compromised node intentionally drops aggregation messages can substantially deviate the final estimate of the aggregate if tree-based aggregation algorithms are used. The deviation will be large if the compromised node is located near the root of the aggregation hierarchy because a large fraction of sensor readings will

be omitted from being aggregated. Countermeasures against this attack include the use of multi-path routing and ring-based topologies [20] as well as the use of probabilistic techniques in the formation of aggregation hierarchies [30].

To prevent the second attack in which a compromised node alters a message being relayed to the sink, it is necessary for each message to include a message authentication code (MAC) generated using a key shared exclusively between the originating node and the sink. This MAC enables the sink to check the integrity of a message, and filter out messages that have been altered. Hence, the effect of altering a message is no different from dropping it, and countermeasures such as multi-path routing are needed to mitigate the effect of this attack.

We refer to the third attack in which a sensor intentionally falsifies its own reading as *the falsified local value attack*. This attack is similar to the behavior of nodes with faulty sensors, and also to the false data injection attack in event-based systems. Potential countermeasures to this attack include approaches used for fault tolerance such as majority voting and reputation-based frameworks [12,15]. In Section 4, we discuss how aggregation schemes can be designed to be resilient to this attack.

The three attacks discussed above apply to both single-aggregator and hierarchical aggregation systems, whereas the fourth attack applies only to hierarchical aggregation systems. This attack in which a node falsifies the aggregate value it is relaying to its parent(s) in the hierarchy is much more difficult to address. We refer to this attack as *the falsified sub-aggregate attack*. In Section 5, we discuss two approaches that have been proposed for resilient hierarchical aggregation that include countermeasures for this attack.

3. Preventing False Data Injection

As discussed in Section 2, in a false data injection attack, an adversary injects false data into the network with the goal of deceiving the base station or depleting the energy resources of the relaying nodes. Several researchers [32,33,36] have proposed solutions to prevent false data injection attacks in sensor networks. Below we briefly discuss the Interleaved Hop-by-Hop Authentication scheme proposed by Zhu *et al* [36] for preventing false data injection attacks.

Interleaved Hop-by-hop Authentication Protocol

We illustrate the false data injection problem and the proposed solution using the scenario depicted in Figure 1 where several sensors been deployed in a cluster for monitoring an area of interest. Events reported by these sensors are relayed to the data collection sink via a multi-hop path formed by several sensor nodes. For ease of exposition, in the description below we will assume that a path between the cluster and the data sink has been established and is stable.

Zhu *et al*'s authentication scheme has two objectives. First, the data sink should be able to verify the authenticity of a report. Second, the nodes on the path from the area of interest to the data sink should be able to filter out false data packets. The scheme is able to meet these objectives as long as the number of compromised nodes is below a threshold number, t.

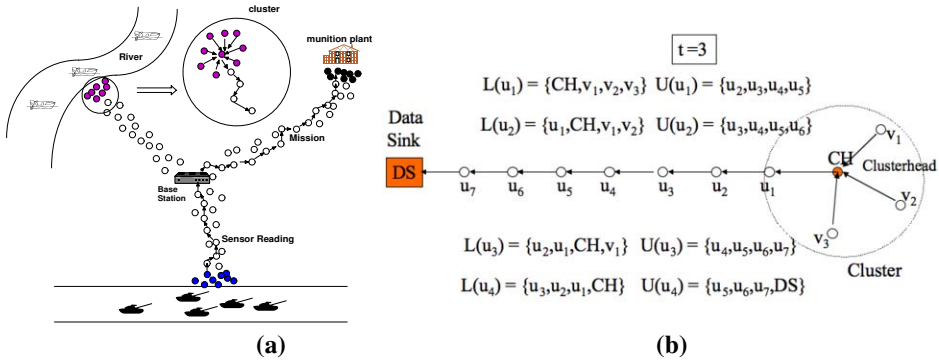


Figure 1. **(a)** A scenario in which sensor nodes have been deployed to monitor three areas of interest – the road, the munitions plant, and the river. **(b)** A logical view of a sensor network in which nodes v_1, v_2, v_3 and CH have been deployed in a cluster to monitor an area of interest. The upper and lower association sets of nodes u_1, u_2, u_3, and u_4 are shown for $t = 3$, where t is the number of compromised nodes that can be tolerated.

Filtering out false data packets within the network (instead of at the sink) has two benefits. First, it results in energy savings since false data packets are immediately dropped instead of being relayed multiple hops to the sink. More importantly, when a false data packet is detected, it reveals the presence of a compromised node. Even though it may be not be possible to identify the compromised node, by detecting the false data close to its origin, it is possible to narrow down the nodes that might be compromised to a small set at a specific location in the network. This information can then be used while taking steps to recover from or mitigate the potential disruptions due to the compromise.

Report Verification To verify the validity of a report, the scheme requires that each report be endorsed by at least $t + 1$ different sensor nodes. To achieve this goal, the sensors monitoring the area of interest are organized into a cluster. If several nodes detect an event of interest, they report the event to the clusterhead, which in turn generates a report for the sink if at least $t + 1$ nodes agree on the event. Note, however, that it is possible that the clusterhead is compromised. Thus, the data sink cannot trust a report simply because it was generated by the clusterhead. An authentication mechanism is needed that allows the data sink to verify that the report was endorsed by $t + 1$ nodes.

Let the event being reported by the nodes in the cluster be E. A straightforward approach to the problem is for each node endorsing the event to compute a Message Authentication Code (MAC) on the event E using its individual key (i.e. key shared exclusively with the data sink) and for the clusterhead to attach $t + 1$ such MACs to the report it sends to the data sink. However, this approach would be very expensive in terms of communication overhead and energy consumption. The solution to this issue is to compress the $t + 1$ individual MACs into a single MAC using the bitwise XOR operation[3]. The report created by the clusterhead then contains the event E, a list of ids of the endorsing nodes, and the compressed MAC. The data sink can thus validate the report by verifying the authenticity of the compressed MAC. (Bellare *et al* have shown this compression scheme is secure [3].)

Enroute Filtering Although this approach will allow the data sink to validate a report, as discussed earlier, it is desirable that a false report be detected as early as possible on the path from the clusterhead to the data sink. To achieve this goal, Zhu *et al* propose to use *interleaved hop-by-hop authentication*.

The basic idea underlying this scheme is that every node en-route to the data sink accepts a report received from an upstream node only if it has been verifiably endorsed by at least $t + 1$ nodes. The scheme requires every node on the path from the cluster generating the report to the data sink to have established security associations (i.e., established pairwise shared keys) with $t + 1$ nodes that are immediately upstream from it. A report is accepted by the node if it has been endorsed by these associated nodes. We refer to the $t + 1$ upstream nodes associated with a node as its lower association set. A node in turn is also in the lower association set of $t + 1$ downstream nodes on the path to the sink. We refer to this set of nodes as the node's upper association set. For nodes that are less than t hops away from the clusterhead, the nodes in its lower association set are designated by the clusterhead from among the nodes in the cluster that reported the event E. (See Figure 1(b)).

To understand how interleaved hop-by-hop authentication works, let us consider a message received by u_4 from u_3 containing a report of an event E. Let $XMAC_{u_4}(u_3, u_2, u_1, CH)$ refer to the compressed MAC created by XORing together the four MACs computed over E using the pairwise keys shared by u_4 with u_3, u_2, u_1 and CH respectively. Then the message received by u_4 should have the following $t+1 = 4$ compressed MACs attached to it – $XMAC_{u_4}(u_3, u_2, u_1, CH)$, $XMAC_{u_5}(u_3, u_2, u_1)$, $XMAC_{u_6}(u_3, u_2)$, and $XMAC_{u_7}(u_3)$. Note that $XMAC_{u_7}(u_3)$ is simply the MAC computed over E using the pairwise key shared between u_7 and u_3, and that $XMAC_{u_4}(u_3, u_2, u_1, CH)$ can be verified by u_4 whereas the remaining three XMACs are destined for nodes u_5, u_6, and u_7 respectively. On receiving a message, if u_4 is able to verify this XMAC, it computes four MACs over E using the pairwise keys it shares with nodes u_5, u_6, u_7, and DS respectively, i.e., the nodes in its upper association set $U(u_4)$. It then XORs these MACs with the XMACs destined for the nodes in its upper association set, i.e., it computes $XMAC_{u_5}(u_4, u_3, u_2, u_1)$, $XMAC_{u_6}(u_4, u_3, u_2)$, $XMAC_{u_7}(u_4, u_3)$, and $XMAC_{DS}(u_4)$. It then attaches these XMACs to the event report E and forwards the message to the node u_5, which will authenticate the message in the same way as u_4 before forwarding it to u_6.

In this manner, at each hop the relaying node can verify if a report is endorsed by $t + 1$ other nodes. As long as the number of compromised nodes does not exceed t, the scheme can detect and filter out false data packets injected into the network within one hop. The size of a message depends upon the number of XMACs attached to a message which is equal to $t+2$, where one of the XMACs is the compressed MAC used by the sink to verify the authenticity of a report and the remaining $t + 1$ XMACs correspond to the XMACs created by the nodes in the receiving node's lower association set as discussed above. Each node has to compute $2(t + 1)$ MACs to verify and authenticate a message. Given that the energy consumed in computing a MAC is roughly equivalent to that used in transmitting one byte [36], this is a beneficial trade.

4. Robust and Secure Single-Aggregator Systems

We now present a survey of secure aggregation techniques proposed for sensor networks. In this section, we discuss approaches for single-aggregator systems in which only the data sink performs the aggregation and there is no in-network aggregation.

In these systems, malicious nodes can attempt to corrupt the aggregate computed at the sink via a falsified local value attack. This attack is similar to the false data injection attack discussed in the previous section, and at first glance it might appear that techniques such as interleaved authentication could also be used to prevent this attack. In event-based systems, however, the nodes monitoring a geographical region simply need to agree that an event of interest has occurred, and thus techniques such as majority voting can be used to filter out malicious reports. In applications that use data aggregation, it is much more difficult and expensive to verify the validity of a sensor reading reported by a node using majority voting. Instead the techniques used to compute the aggregate at the sink need to be designed to be resilient to malicious data.

4.1. Resilient Aggregation Functions

Wagner [29] addressed the following question: in a system with a single aggregator node, which aggregation functions can be securely and meaningfully computed in the presence of a few compromised nodes? Wagner observed that aggregation functions can be viewed as statistical estimators. Based on this observation, he introduced a theoretical framework for modeling the security of data aggregation.

Given a sequence of observations $x_1, \ldots, x_n$ from a known parameterized distribution $p(X \mid \theta)$, where θ is a hidden parameter, the goal of statistical estimation theory is to estimate θ as accurately as possible. Let $X_1, \cdots, X_n$ denote n random variables that are distributed according to a known parameterized distribution $p(X \mid \theta)$, where the hidden parameter θ's distribution is not specified. A parameterized distribution $p(X \mid \theta)$ is a family of distributions, one for each possible value of θ. For instance, $N(\theta, 1)$, the Gaussian distribution with mean θ and variance 1, is a distribution with parameter θ.

An estimator is an algorithm $f : \mathbf{R}^n \rightarrow \mathbf{R}$, where $f(x_1, \ldots, x_n)$ is intended as an estimate of some real-valued function of θ. Let the random variable Θ be equal to $f(X_1, \ldots, X_n)$. Then,

$$\text{root mean square (r.m.s.) error (at } \theta) : \text{rms}(f) = E[(\Theta - \theta)^2 | \theta]^{\frac{1}{2}}$$

Note that $\text{rms}(f)$ is a function of θ, the underlying parameter. Also, an unbiased estimator is one for which $E[\Theta | \theta] = \theta$ for all θ.

To define a resilient aggregation function, Wagner first defines a k-node attack. A k-node attack A changes k of the input values $X_1, \ldots, X_n$ of an aggregation function. In particular, the attack A is specified by a function $\tau_A : \mathbf{R^n} \rightarrow \mathbf{R^n}$ with the property that the vectors x and $\tau_A(x)$ never differ at more than k positions. We can define the r.m.s. error associated with A by

$$\text{rms}^*(f, A) = E[(\Theta^* - \theta)^2 | \theta]^{\frac{1}{2}} \text{ where } \Theta^* = f(\tau_A(X_1, \ldots, X_n)).$$

In the context of resilient aggregation, Θ^* is a random variable that represents the aggregate calculated at the base station in the presence of the k-node attack A,

and $\mathrm{rms}^*(f, A)$ is a measure of the inaccuracy of the aggregate after the attack. If $\mathrm{rms}^*(f, A) \gg \mathrm{rms}(f)$, then the attack has succeeded in noticeably affecting the estimate of the aggregate. If $\mathrm{rms}^*(f, A) \approx \mathrm{rms}(f)$, the attack has had little or no effect. Wagner defined

$$\mathrm{rms}^*(f, k) = \max \{\mathrm{rms}^*(f, A) : A \text{ is a } k \text{ node attack}\},$$

so that $\mathrm{rms}^*(f, k)$ denotes the r.m.s. error of the most powerful k-node attack possible.

Note that $\mathrm{rms}^*(f, 0) = \mathrm{rms}(f)$. Roughly speaking, we can think of an aggregation function f as being resilient if $\mathrm{rms}^*(f, k)$ grows slowly as a function of k. More formally, an aggregation function f is (k, α)-resilient (with respect to a parameterized distribution $p(X \mid \theta)$) if $\mathrm{rms}^*(f, k) \leq \alpha \cdot \mathrm{rms}(f)$ where α is a real number. The intuition is that for small values of α, a (k, α)-resilient function is a function that can be computed meaningfully and securely in the presence of up to k compromised or malicious nodes.

Wagner also introduces the concept of the breakdown point from robust statistics in the context of resilient aggregation. The breakdown point is defined as

$$\epsilon = \sup\{\frac{k}{n} : \mathrm{rms}^*(f, k) < \infty\}$$

The breakdown point indicates the fraction ϵ of nodes that can be captured before security breaks down. If an ϵ fraction of nodes are compromised, then the r.m.s. error becomes unbounded, and the adversary attains complete control over the aggregation operation. For instance, we can easily verify that Sum has breakdown point $\epsilon = 0$, because any single compromised node can skew the Sum by an arbitrary amount. Consequently, the breakdown point is one measure of the security of an aggregation function against the falsified local value attack. Wagner shows that the breakdown point for aggregates such as `MAXIMUM, MINIMUM` and `AVERAGE` is also 0.

In contrast, `COUNT` can be shown to be a resilient aggregate. Assuming that the inputs X_i follow a Bernoulli distribution with parameter θ (i.e. $P[X_i = 1 \mid \theta] = \theta$ and $P[X_i = 0 \mid \theta] = 1 - \theta$), it can be shown that `COUNT` is (k, α)-resilient for $\alpha = 1 + k.(n\theta(1-\theta))^{-\frac{1}{2}}$. Further, the value of α grows slowly with k when n is sufficiently large.

The aggregate `MEDIAN` is a robust replacement for Average. Since the median represents the sensor reading at the midpoint of the sorted list of all the readings, a 1-node attack can only deviate `MEDIAN` by one place in the sorted list. In general, after a k-node attack, `MEDIAN` will be deviated by at most k places, and when $n > 2k$, `MEDIAN` will be associated with an uncompromised node. Assuming that sensor readings follow a Normal distribution, it can be proved that `MEDIAN` is (k, α)-resilient where $\alpha = (1 + 0.101k^2)^{\frac{1}{2}}$ for large n. Also, the breakdown point of `MEDIAN` is $\frac{1}{2}$.

4.1.1. Filtering Outliers

Wagner also recommended the use of outlier filtering techniques such as truncation and trimming for achieving resilient aggregation. Outliers can be detected and filtered out in a hop-by-hop fashion or at the base station after the the sensor readings reach the base station.

Truncation: One way to make an aggregation function robust to outliers is to place upper and lower bounds on the acceptable range of a sensor reading. If we know that valid sensor readings are usually in the interval $[l, u]$, then we can truncate every input to be within this range. Given any base aggregation function f, we can construct a truncated version by applying f to the truncated data values. The truncated aggregate is an improvement over the conventional aggregate, but it is not an entirely satisfactory solution. A wide interval between the upper and lower bounds gives the attacker a great deal of power, while a narrow interval reduces the utility of the aggregation function.

Trimming: Trimming is a better outlier filtering technique where we ignore the highest ρ and lowest ρ fraction of the sensor readings, and compute the aggregate using the remaining readings, where ρ is a security parameter. Intuitively, we might expect this technique to be fairly robust to outliers, as long as no more than ρ fraction of the sensors are compromised or faulty.

RANBAR: Buttyan *et al* [4] proposed a technique called RANBAR for filtering outliers. RANBAR is based on the use of RANSAC [8], a well-known algorithm used to estimate the parameters of a mathematical model from a set of observed data which contains outliers. Buttyan shows that compared to other resilient aggregation functions such as the trimmed average and the median, RANBAR results in smaller distortion, especially when a large fraction of nodes are compromised.

4.2. SIA: Handling Malicious Aggregators

In the work described above, it is assumed that the aggregator is not compromised. Przydatek *et al* [26] relaxed this assumption in the design of their secure aggregation framework named Secure Information Aggregation (SIA). SIA considers a sensor network where a large number of sensors are deployed in an area distant from a home server (i.e., a user) and a base station is used as an intermediary between the home server and the sensor nodes. After sensor nodes send their readings to the base station, the base station performs the aggregation task and forwards the result to the home server. The base station is the only aggregator node in the network.

The goal of SIA is to enable the user to verify that the answer given by the aggregator in response to a query is a good approximation of the true value even when the aggregator and some fraction of the sensor nodes are corrupted. In particular, SIA includes efficient protocols for secure computation of the `MEDIAN`, `AVERAGE`, `COUNT`, `MINIMUM` and `MAXIMUM` aggregates.

SIA is designed to prevent stealthy attacks, where the attacker's goal is to make the user accept false aggregation results without being detected. In particular, SIA aims to guarantee that if the user accepts an aggregation result reported by the aggregator, then the reported result is close to the true aggregation value with high probability; otherwise, if the reported value is significantly different from the true value due to the misbehavior of the compromised aggregator, the user will detect the attack and reject the reported aggregate with high probability.

The result $\hat{y}$ reported by the aggregator is said to an ϵ-approximation of the true aggregate y if $(1-\epsilon)y \leq \hat{y} \leq (1+\epsilon)y$. In addition to the approximation error ϵ, which describes the quality of a reported result, SIA also uses a parameter δ which upper bounds the probability of not detecting a cheating aggregator (i.e., an aggregator reporting a

result not within ϵ bounds). Formally, a protocol is said to an (ϵ, δ)-approximation, if the protocol finds an ϵ-approximation with probability at least $1 - \delta$, and runs in time polynomial in $1/\epsilon$ and $1/(1 - \delta)$.

To achieve its goal, SIA employs an *aggregate-commit-prove* approach. In this approach, the aggregator not only performs the aggregation, but also proves that it has computed the aggregate correctly.

In the first step of this approach, the aggregator collects sensor data from nodes and computes the aggregation result. In the second step, the aggregator constructs a commitment based on Merkle hash-trees corresponding to the sensor data collected in the first step. In this construction, all the collected data is placed at the leaves of the tree, and the aggregator then computes a binary hash tree starting from the leaf nodes; each internal node in the hash tree is computed as the hash value of the concatenation of the two child nodes. The root of the tree is called the commitment of the collected data. Because the hash function in use is collision resistant, once the aggregator commits to the collected values, it cannot change any of the collected values. In the third step, the aggregator sends the the home server both the aggregation result and its associated commitment. The home server and the aggregator engage in an interactive protocol in which the aggregator proves to the home server that the reported results are correct. The interactive protocol used depends upon the aggregate being computed.

To illustrate the aggregate-commit-prove approach, we now discuss how the `MEDIAN` aggregate is securely computed in SIA. First we describe a naive approach for securely computing `MEDIAN`, before discussing the more efficient approach used in SIA.

A straightforward approach for estimating the median using sub-linear communication complexity is random sampling; the user collects a random sample of l measured values via the base station and considers the median of the sample as an approximation of the median of all the measurements. Bar-Yossef *et al* [2] show that $l = \Omega(1/\epsilon^2)$ samples are necessary for an ϵ-approximation of the median.

To reduce the communication overhead, SIA uses an approach based on interactive-proofs. First, the base station sorts the measurements received from the sensors and commits the measurements using a hash-tree construction. Then, the base station engages with the home server in an interactive proof session in which the home server verifies the correctness of the alleged median a_{med} by conducting two tests.

The first test verifies that the committed sequence is indeed sorted. This test can be performed using Sort-Check-II spot checker from [7] with subsequent uniform sampling of pairs of neighboring elements, which requires $O(\log n/\epsilon)$ samples in total. In the second test, the home server checks that a_{med} is sufficiently close to the median of the committed sequence. Here the home server picks $1/\epsilon$ elements from random positions in the committed sequence, and checks that elements picked from the left half of the sequence are smaller than a_{med}, and the elements from the right half are larger than a_{med}.

Przydatek *et al* proved that by requesting only $O(\log n/\epsilon)$ samples from the base station, the home server can check whether the reported value is an ϵ-approximation of the median, and at the same time guarantee a constant probability of detecting a cheating aggregator.

5. Attack-resilient Hierarchical Aggregation

Compared to the single-aggregator scenario, it is far more challenging to design a secure hierarchical aggregation protocol. As discussed in Section 2, the use of in-network aggregation makes it possible for an adversary to launch a falsified sub-aggregate attack. Recently, several researchers have proposed schemes for attack-resilent hierarchical data aggregation [30,27,5]. In this section, we describe two approaches for securing hierarchical aggregation protocols. The first scheme assumes that sensor nodes are organized in a tree-based hierarchy such as that used in the TAG aggregation framework [16], whereas the second scheme assumes the use of a ring-based hierarchy such that used in the Synopsis Diffusion framework [20].

5.1. SDAP

The Secure Hop-by-hop Data Aggregation Protocol (SDAP) [30] is based on the principles of divide-and-conquer and commit-and-attest. We first present a high level overview of the protocol.

SDAP uses a novel probabilistic grouping technique to dynamically partition the nodes in a tree topology into multiple logical groups (subtrees) of similar sizes. An aggregate is computed within each logical group using a standard hop-by-hop aggregation protocol. The leader of each logical group transmits the group's aggregate to the base station, along with a commitment that is generated based on the contributions of each node in the group. After the base station has collected all the group aggregates, it uses an outlier detection algorithm to identify groups whose contribution is potentially suspect. Finally, each group under suspicion participates in an attestation process to prove the correctness of its group aggregate. The base station discards any suspicious aggregates from groups that fail the attestation procedure. The final aggregate is calculated over all the group aggregates that have either passed the outlier detection algorithm or the attestation procedure.

SDAP assumes that the base station cannot be compromised and that a secure broadcast authentication mechanism (e.g., μTESLA [23]) is available. It also assumes that every sensor node has an individual secret key shared with the base station. Further, there is a unique pairwise key shared between each pair of neighboring nodes.

We now discuss the four main phases of SDAP – query dissemination, probabilistic grouping and data aggregation, suspicious group detection, and group attestation – in more detail.

Query Dissemination: In the query dissemination phase, the base station broadcasts the aggregation query message throughout the network. The aggregation tree is also constructed in this phase, if it does not already exist. μTESLA is used for authenticating the broadcast message.

Probabilistic Grouping and Data Aggregation: In this phase, SDAP randomly groups all the nodes into multiple logical groups and performs aggregation within each group. Probabilistic grouping is achieved via the selection of the leader node for each group. Because grouping is a dynamic process, a node will not know in advance whether it will become a group leader or which group it will belong to.

Group leaders are selected dynamically based on the number of nodes in its sub-tree that have not yet been grouped (referred to as the node's *count value*) and a grouping seed

S_g. While the grouping seed is included in the broadcast query, each node calculates its count value based on the count values received from its children during the aggregation process (as discussed below).

Two functions are used in group leader selection. One is a cryptographically secure pseudo-random function H that uniformly maps the inputs (the node id and S_g) into the range of $[0, 1)$; the other is a grouping function F_g that takes the local node's count value as the input and outputs a real number between $[0, 1]$. More specifically, each node, say x, decides if it is a leader node by checking whether

$$H(S_g|x) < F_g(c)$$

where c is the count value of node x. If this inequality is true, node x becomes a leader. Once a node becomes the leader, all the nodes in its subtree that have not been grouped yet become members of its group. The function $F_g()$ is constructed such that $F_g(c)$ increases with the count value c. This ensures that nodes with more descendants have a higher probability of becoming group leaders.

In SDAP, a node's aggregation message contains its node id, its count value, and its aggregate. In addition, each message includes a flag that indicates whether the aggregate needs to be aggregated further by the nodes enroute to the base station. The pairwise key shared between each pair of parent and child is used to encrypt the aggregate. Further, each message includes a message authentication code (MAC) that is computed using the nodes individual key (shared with the base station).

Data aggregation starts from the leaf nodes. Since a leaf node u does not do any aggregation, it just sends a packet containing its id, aggregation flag set to '0', its data, and its count value as '1' to the parent, say v. The above information is encrypted using the pairwise key K_{uv}. The message also includes a MAC generated using node u's individual key K_u that authenticates the information in the message.

When a non-leaf node v receives an aggregate from a child node, it first checks the aggregation flag. If the flag is '0', it performs further aggregation; otherwise, the node directly forwards the packet to its parent node. A non-leaf node aggregates its own reading with all the aggregates received from child nodes with aggregation flag '0'. A new count is calculated as the sum of the count values in the received aggregates with flag '0' plus one. The node v checks if it is a group leader using its own id and the new count as the inputs to the functions $F_g()$ and H. The node v then encrypts the new count value and aggregation data using the pairwise key shared with its own parent.

Node v's message also includes a MAC which is calculated over not only the above fields but also the XOR of the MACs received from its children. In this way, the MAC computed by a node represents the authentication information of all the nodes contributing to its aggregate.

Now suppose that a node x has processed the aggregates from its child nodes and it finds out that it is a group leader. Like any other node, it also computes a new aggregate and appends a corresponding MAC calculated using its individual key shared with the base station. Unlike other (non-leader) nodes, it encrypts the new aggregate with its individual key and sets the flag to '1' in its aggregation packet, so that data from this group will not be aggregated any more enroute to the base station. Note that the MAC computed by a group leader represents an aggregation commitment, which is used in the attestation phase of the protocol.

Determining Suspicious Groups: A compromised node can be expected to forge its sub-aggregate so that it will have a non-trivial influence on the final result; otherwise the attack would not gain much. Thus, the aggregate of a group which contains a compromised node can be treated as an outlier compared to the aggregates of other groups. SDAP uses Grubbs' test [10], also known as the maximum normalized residual test, to detect outliers. The original Grubb's test cannot be directly applied for this purpose because it can handle only univariate data where a group's aggregate has two dimensions, the count value and the aggregate being computed. Yang *et al* modified Grubb's test so that it can detect outliers in bivariate data. Groups whose aggregate values are outliers are considered suspicious, and their aggregates are verified using the attestation process described below.

Group Attestation: The BS broadcasts an attestation message including the id of the leader for the group to be attested, a random number S_a, and S_g. S_a is used as the seed for the attestation and it will determine a unique and verifiable attestation path whereas S_g is included for identifying the query.

During the attestation process, a physical attestation path between the group leader and a leaf node in its logical subtree is dynamically formed. After the group leader node receives the attestation request from the base station, it selects the next hop on the attestation path using a probabilistic procedure in which the probability of a child node to be selected on the path is proportional to its count value reported in the aggregation phase. A selected child runs the same process to select one of its own children to be on the attestation path. Recursively, an attestation path between the leader and a leaf node in the logical group subtree is formed.

Subsequently, each node on the path sends back to the base station its count value and its own reading. If the node is not a leaf node, its parent also asks its sibling nodes to send back their count values, sum values, and their MACs.

After the base station decrypts the received data, it first verifies whether the responding nodes are really the nodes on the attestation path based on S_a, the nodes' ids, and the counts. It verifies whether the count value of every node is the sum of its children's counts plus one. If this check succeeds, it aggregates the data and reconstructs the aggregation result of the group to examine whether nodes on the path have forged the aggregation results in the aggregation phase.

The commit-and-attest technique used by SDAP ensures that once a group has committed its aggregation result, every node involved in the attestation phase has to report its original aggregate. Otherwise, the base station will detect the attack by finding the inconsistency between the committed aggregate and the reconstructed aggregate. Due to the probabilistic grouping scheme, an attacker cannot selectively compromise nodes – for example, making no more than one compromised node appear in the same group. Also, because the aggregates from all the groups are encrypted, a compromised node cannot know if its own aggregate will become an outlier after Grubbs' test is run. Further, a node cannot know whether it will be selected on the attestation path because the attestation path is also dynamically selected in a probabilistic fashion.

5.2. Attack-resilient Synopsis Diffusion

Tree-based aggregation approaches are vulnerable to communication losses which result from node and transmission failures and are relatively common in sensor networks [16,

34,35]. Because each communication failure loses an entire subtree of readings, a large fraction of sensor readings are potentially not incorporated in the final aggregate at the querying node. Although protocols such as SDAP and the resilient aggregation scheme recently proposed by Chan *et al* [5] make the tree-based aggregation approach resilient to malicious data, they remain vulnerable to communication loss.

To address this problem, researchers have proposed the use of multi-path routing techniques for forwarding sub-aggregates [16]. For aggregates such as `MIN` and `MAX` which are duplicate-insensitive, this approach provides a loss-tolerant solution. For duplicate-sensitive aggregates such as `COUNT` and `SUM`, however, multi-path routing leads to double-counting of sensor readings, resulting in an incorrect aggregate being computed. Researchers [6,18,20] have presented novel algorithms that solve the double-counting problem associated with multi-path approaches. A robust and scalable aggregation framework called *Synopsis Diffusion* has been proposed for computing aggregates such as `COUNT, SUM, UNIFORM SAMPLE` and `MOST FREQUENT ITEMS`.

Synopsis Diffusion, however, does not include any provisions for security, and a compromised node can launch several attacks against this framework which can potentially cause the querier to accept an incorrect result. Roy *et al* [27] propose attack-resilient aggregation protocols within the synopsis diffusion framework to compute aggregates such as `COUNT` and `SUM`. These secure protocols are developed by augmenting the original synopsis diffusion algorithms with authentication techniques.

Before discussing Roy *et al*'s protocol, we provide an overview of synopsis diffusion, and discuss how a compromised node could launch a falsified sub-aggregate attack against the protocol.

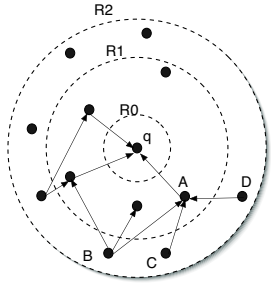


Figure 2. Synopsis Diffusion over a rings topology

Synopsis Diffusion There are two primary elements of the synopsis diffusion approach – the use of a ring-based topology instead of a tree-based topology, and the use of duplicate-insensitive algorithms for computing aggregates based on Flajolet and Martin's algorithm for counting distinct elements in a multi-set [9]. Figure 2 illustrates how this approach uses a rings topology for aggregation. In the query distribution phase, nodes form a set of rings around the querying node q based on their distance in hops from q. During the subsequent query aggregation period, starting in the outermost ring each node generates a local synopsis $s = SG(v)$ where v is the sensor reading relevant to the query, and broadcasts it. (Here $SG()$ is a order- and duplicate-insensitive aggregate function.) A node in ring R_i will receive broadcasts from all the nodes in its range in ring R_{i+1}.

It will then combine its own local synopsis with the synopses received from its children using a synopsis fusion function $SF()$, and then broadcast the updated synopsis. Thus, the fused synopses propagate level-by-level until they reach the querying node, who first combines the received synopses with its local synopsis using $SF()$ and then uses the synopsis evaluation function $SE()$ to translate the final synopsis to the answer to the query.

The functions $SG()$, $SF()$, and $SE()$ depend upon the target aggregation function. We now describe synopsis diffusion algorithms for the COUNT aggregate. In the COUNT algorithm, each node generates a local synopsis which is a bit vector ls of length $k > \log n$, where n is an upper bound on the nodes in the network. To generate its local synopsis, each node executes the function $CT(X, k)$ where X is the node's identifier and k is the length of ls in bits. $CT()$ can be interpreted as a coin-tossing experiment (with the random binary hash function $h()$ simulating a fair coin-toss), which returns the number of coin tosses until the first heads occurs or $k + 1$ if k tosses have occurred with no heads occurring. In the local synopsis ls of node X, a single bit i is set to 1, where i is the output of $CT(X, k)$. Thus ls is a bitmap of the form $0^{i-1}1\cdots$ with probability 2^{-i}.

Algorithm 1 $CT(X, k)$

```
i=1;
while i < k + 1 AND h(X, i) = 0 do
   i = i + 1;
end while
return i;
```

For COUNT, the synopsis fusion function $SF()$ is simply the bitwise Boolean OR of the synopses being combined. Each node fuses its local synopsis ls with the synopses it receives from its children by computing the bit-wise OR of all the synopses. Let S denote the final synopsis computed by the querying node by combining all the synopses received from its children and its local synopsis. We observe that S will be a bitmap of length k of the form $1^{r-1}0\cdots$. The querying node can estimate COUNT from S via the synopsis evaluation function $SE()$: if r is the lowest-order bit in S that is 0, the count of nodes in the network is $2^{r-1}/0.7735$. Intuitively, the number of sensor nodes is proportional to 2^{r-1} since no node has set the rth bit while computing $CT(X, k)$.

Attacks Just as in the case of a tree-based aggregation protocol, a compromised node can cause the synopsis diffusion algorithms to output an incorrect aggregate by manipulating a sub-aggregate or its own local sensor reading. We now discuss this attack in the context of the COUNT aggregate.

Since the base station estimates the aggregate based on the lowest-order bit r that is 0 in the final synopsis, a compromised node would need to falsify its own fused synopsis such that it would affect the value of r. It can accomplish this quite easily by simply inserting ones in one or more bits in positions j, where $r \leq j \leq k$, in its own fused synopsis which it broadcasts to its parents. Let r' be the lowest-order bit that is 0 in the corrupted synopsis, whereas r is the lowest-order bit that is 0 in the correct synopsis. Then the base station's estimate of the aggregate will be larger than the correct estimate by a factor of $2^{r'-r}$. We also observe that even a single node can launch this attack with

a high rate of success because the use of multi-path routing makes it highly likely that the falsified synopsis will be propagated to the base station.

On the other hand, it is very hard to launch an attack which results in the aggregate estimated at the sink being lower than the true estimate. This is because setting a bit in the falsified synopsis to 0 has no effect if there is another node X that contributes a 1 to the same position in the fused synopsis. To make this attack a success the attacker has to compromise all the possible paths from node X to the sink so that X's 1 cannot reach the sink, which is hard to achieve.

In the rest of this chapter, we restrict our discussion to the previous attack where the goal of the attacker is only to increase the estimate. We note that Garofalakis *et al* [13] have proposed defenses against attacks aimed at decreasing the aggregate estimated at the sink. However, their work does not focus on networks with resource-constrained nodes. Their approach assumes the use of digital signatures and does not consider the use of multi-path routing by the aggregation system.

ARSD Roy et al [27] propose attack-resilient aggregation protocols within the synopsis diffusion framework to compute aggregates such as `COUNT` and `SUM`. These protocols enable the base station to detect the attack and to filter out the false data injected by the attacker so that the estimate of the aggregate is not affected.

In the Attack-Resilient Synopsis Diffusion (ARSD) protocol, a subset of the nodes responding to a query include along with their synopses a message authentication code (MAC) that can be used by the base station to verify the validity of their contribution to the aggregate function. The key observations behind the design of ARSD are that:

- In order to derive the correct estimate of the aggregate (`COUNT` or `SUM`) from the final synopsis (say S) computed at the base station, it is only necessary to figure out the correct lowest order bit (say r) in S that is 0.
- The number of nodes contributing a 1 to bit j decreases as we move from the lowest order bit ($j = 1$) to higher order bits of the synopsis. For example, in the case of `COUNT`, on average, half the nodes in the network will contribute to the leftmost bit of the synopsis, one-fourth of the nodes contribute to the second bit of the synopsis, and so on.

Thus, it is expected that only a small number of nodes will contribute to the bits around the expected value of r; the expected number of nodes that contribute to bits i, where $r < i \leq k$ in the synopsis (k is the length of the synopsis) is very small. In fact, it can be showed that expected number of nodes contributing to the bits to the right of bit r is less than $1/\phi \approx 1.29$. In this approach, nodes contributing to bit i, $i \geq r - 2$ include along with its response to an aggregation query a MAC computed using a pairwise key shared exclusively with the base station. These MACs enable the base station to verify the computed aggregate and to filter out the contributions of the falsified sub-aggregates injected by the compromised nodes.

To design the attack-resilient `COUNT` protocol, two issues need to be addressed. First, since a node cannot locally determine the position of bit r in the final synopsis, the base station needs to specify a global criterion which determines if a node needs to include a MAC along with its synopsis. Second, this criterion should be designed so that the number of such nodes who include a MAC is minimized.

Roy *et al* [27] present two approaches to address these issues. In the basic approach, it is assumed that the base station has an estimate of the lower bound and the upper

bound of Count which are denoted by l_c and u_c respectively. Based upon these bounds, the base station knows that bit r will lie between a and b, which are the bit positions in the synopsis S corresponding to l_c and u_c respectively, i.e., $a = \lceil \log_2(\phi l_c) \rceil$ and $b = \lfloor \log_2(\phi u_c) \rfloor$. Thus, there is no need for the base station to verify the bits to the left of a; only nodes contributing to bits in the range a to b need to prove to the base station that their contribution to the synopsis S is valid.

The collection of bits in the range a to b in synopsis S is referred to as the *synopsis-edge*. It is easy to see that the length of the synopsis-edge is $(\lfloor log_2(\frac{u_c}{l_c}) \rfloor + 1)$ bits. If we denote the number of nodes contributing to the synopsis-edge by η, then, by the properties of the COUNT synopsis, $\eta \leq (\frac{u_c}{2^a} + \ldots + \frac{u_c}{2^b}) \approx \frac{1}{\phi} \cdot (\frac{2u_c}{l_c} - 1)$.

Thus, in the basic protocol, each node checks whether it contributes to the synopsis-edge, and if so, it generates a MAC and forwards the MAC along with its fused synopsis. Specifically, if node X contributes to bit i in the synopsis-edge, it generates a MAC, $M = MAC(K_X, m)$ over the message m whose format is $[X|i]$. After the base station receives the MAC, it checks its validity by executing the synopsis generation function $SG()$ with the inputs for node X. If the verification fails, the contribution of X is discarded. The bits in the synopsis-edge for which the base station has not received a valid MAC are reset to 0. The bits at positions to the left of the synopsis-edge are set to 1. Finally, the base station computes the COUNT aggregate using the synopsis evaluation function $SE()$.

Although a compromised node can falsely set some bits in its local fused synopsis, the base station will be able to discard them by verifying the corresponding MACs. This implies that the attacker cannot falsely increase the COUNT. On the other hand, the attacker may attempt to decrease the estimated COUNT by dropping a genuine MAC (or by corrupting a genuine MAC) sent by a contributing node, but the genuine MAC is likely to reach base station via an alternate path. If base station receives at least one valid MAC for each 1 bit in the synopsis-edge, then base station obtains the true estimate of COUNT.

The basic protocol outlined is not scalable if the ratio of the upper bound (u_c) of COUNT to the lower bound (l_c) is high, because in the worst case a node has to forward $\frac{1}{\phi} \cdot (\frac{2u_c}{l_c} - 1)$ MACs. To address this problem, Roy *et al* propose another protocol which results in lower communication overhead at the expense of greater latency.

In the extended protocol, the synopsis is computed by a sliding window approach where the aggregation phase is divided into multiple epochs. Starting at the rightmost bit k, the extended protocol proceed from right to left within the synopsis S using a bit window of w bits. In each epoch, only the nodes that contribute a 1 to the bits in S corresponding to the current position of the window, send MACs to the base station that can be used to verify their contribution to S. In other words, in the first epoch, only nodes that contribute a 1 to bits k to $k - w + 1$ respond to the query. In epoch two, nodes that contribute to bits between $k - w$ and $k - 2w + 1$ respond, and so on.

The algorithm terminates when the base station has determined that the remaining bits of S to the left of the current window are likely to be 1 with high probability. Once the base station determines that the algorithm can terminate it broadcasts a STOP message to the network to announce the end of the iterative aggregation procedure.

Roy *et al* show via analysis and simulation that the iterative procedure outlined above has low communication overhead in comparison to the basic protocol. They also show that this algorithm results in the same estimate as the original synopsis diffusion

approach when there are no compromised nodes, while providing additional resilience in the presence of compromised nodes.

6. Conclusions

In this chapter, we have discussed the security vulnerabilities of data aggregation protocols for sensor networks. We also presented a survey of secure and resilient aggregation protocols for both single-aggregator and hierarchical systems.

A number of challenges remain in the area of secure aggregation for sensor networks. Secure tree-based aggregation protocols [30,5] remain vulnerable to message losses either due to node failure or compromised nodes. The performance and security tradeoffs between resilient tree-based approaches and multi-path approaches such as Attack Resilient Synopsis Diffusion [27] have yet to be explored. The research community is yet to design a secure aggregation protocol for computing holistic aggregates such as Order-statistics and Most Frequent Items. Finally, many data acquisition systems use persistent queries in which nodes periodically send readings to the sink resulting in streams or flows of sensor data [17,1]. These systems make extensive use of data aggregation. Issues in securing such sensor data streaming applications remain to be investigated.

References

[1] Daniel J. Abadi, Wolfgang Lindner, Samuel Madden, and Jörg Schuler. An integration framework for sensor networks and data stream management systems. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *VLDB*, pages 1361–1364. Morgan Kaufmann, 2004.

[2] Ziv Bar-Yossef, S. Ravi Kumar, and D. Sivakumar. Sampling algorithms: Lower bounds and applications. *Proc. of 33rd STOC*, pages 266–275, 2001.

[3] M. Bellare, R. Guerin, and P. Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. In *Proc. of the 15th Annual International Cryptology Conference on Advances in Cryptology - CRYPTO'95*, pages 15–28, 1995.

[4] L. Buttyan, P. Schaffer, and I. Vajda. RANBAR:RANSAC-based resilient aggregation in sensor networks. In *Proc. of ACM Workshop on Security of Sensor and Adhoc Networks (SASN)*, 2006.

[5] Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation in sensor networks. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2006.

[6] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Proc. of IEEE Int'l Conf. on Data Engineering (ICDE)*, 2004.

[7] Funda Ergun, Sampath Kannan, S. Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spotcheckers. *JCSS*, 60:717–751, 2000.

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[9] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[10] Grubbs Frank. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):121, 1969.

[11] A. Gabrieli, L. Mancini, S. Setia, and S. Jajodia. Securing topology maintenance protocols for sensor networks: Attacks & countermeasures. In *SecureComm 2005. First International Conference on Security and Privacy for Emerging Areas in Communications Networks, 2005.* IEEE, 2005.

[12] S. Ganeriwal and M. B. Srivastava. Reputation-based framework for highly integrity sensor networks. In *Proc. of ACM Workshop on Security of Sensor and Adhoc Networks (SASN)*, Washington, DC, 2004.

[13] M. Garofalakis, J. M. Hellerstein, and P. Maniatis. Proof sketches: Verifiable in-network aggregation. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE '07)*.

[14] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks*, 1(2-3):293–315, 2003.

[15] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli. Fault tolerance techniques in wireless ad-hoc sensor networks. In *Sensors 2002, Proceedings of IEEE*, pages 1491 – 1496.

[16] S. Madden, M. J. Franklin, J.M. Hellerstein, and W. Hong. TAG: A tiny aggregation service for ad hoc sensor networks. In *Proc. of 5th USENIX Symposium on Operating Systems Design and Implementation*, 2002.

[17] Samuel Madden and Michael J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *Proceedings of Intl. Conf. on Data Engineering*, pages 555–566. IEEE Computer Society, 2002.

[18] A. Manjhi, S. Nath, and P. Gibbons. Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In *Proc. of ACM International Conference on Management of Data (SIGMOD)*, 2005.

[19] James Reserve Microclimate and Video Remote Sensing. http://www.cens.ucla.edu.

[20] S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proc. of the 2nd international conference on Embedded networked sensor systems (SenSys)*, 2004.

[21] Habitat Monitoring on Great Duck Island. http://www.greatduckisland.net/.

[22] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed detection of node replication attacks in sensor networks. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2005.

[23] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *Seventh Annual International Conference on Mobile Computing and Networks (MobiCOM)*, 2001.

[24] Adrian Perrig, John A. Stankovic, and David Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, 2004.

[25] The Firebug Project. http://firebug.sourceforge.net.

[26] B. Przydatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *Proc. of the 1st international conference on Embedded networked sensor systems (SenSys)*, 2003.

[27] Sankardas Roy, Sanjeev Setia, and Sushil Jajodia. Attack-resilient hierarchical data aggregation in sensor networks. In *Proc. of ACM Workshop on Security of Sensor and Adhoc Networks (SASN)*, 2006.

[28] A. Silberstein, R. Braynard, C. Ellis, K. Munagala, and J. Yang. A sampling-based approach to optimizing top-k queries in sensor networks. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE '06)*.

[29] D. Wagner. Resilient aggregation in sensor networks. In *Proc. of ACM Workshop on Security of Sensor and Adhoc Networks (SASN)*, 2004.

[30] Y. Yang, X. Wang, S. Zhu, and G. Cao. SDAP: A secure hop-by-hop data aggregation protocol for sensor networks. In *Proc. of ACM MOBIHOC*, 2006.

[31] Y. Yao and J. E. Gehrke. The cougar approach to in-network query processing in sensor networks. *ACM SIGMOD Record*, 31(2):9–18, September 2002.

[32] Fan Ye, Haiyun Luo, Songwu Lu, and Lixia Zhang. Statistical en-route filtering of injected false data in sensor networks. In *Proc. of IEEE Infocom*, 2004.

[33] W. Zhang and G. Cao. Group rekeying for filtering false data in sensor networks: A predistribution and local collaboration-based approach. In *Proc. of IEEE Infocom*, 2005.

[34] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proc. of the 1st international conference on Embedded networked sensor systems (SenSys)*, 2003.

[35] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring sensor networks. In *Proc. of the 2nd IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[36] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved hop-by-hop authentication scheme for filtering injected false data in sensor networks. In *Proc. of IEEE Symposium on Security and Privacy*, 2004.

[37] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. LEAP+: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks*, 2(4):500–528, 2006.

Wireless Sensor Network Security
J. Lopez and J. Zhou (Eds.)
IOS Press, 2008

Privacy Protection Mechanisms for Sensor Networks

Efthimia AIVALOGLOU [a], Stefanos GRITZALIS [a] and Sokratis KATSIKAS [b]
[a] *University of the Aegean*
Karlovassi, Samos, GR-83200, Greece
[b] *University of Piraeus*
150 Androutsou St., Piraeus, GR-18532, Greece
e-mail: {eaiv,sgritz}@aegean.gr, ska@unipi.gr

Abstract. While sensor networks provide the opportunity for sophisticated, context-aware services, privacy concerns can seriously affect user acceptance and become a barrier to their long-term successful deployment. This chapter discusses privacy issues in sensor networks, by identifying the requirements for privacy preserving deployments, analysing the challenges faced when designing them, and discussing the main solutions that have been proposed. Privacy can be addressed in different levels of the network stack and at different points of the information flow. The privacy requirements and the mechanisms that address them are categorised into privacy sensitive information gathering schemes, controlled information disclosure approaches, and mechanisms for the protection of the communications context. The separate discussion of the approaches highlights the diverse privacy aspects that they are focused on, and shows how the approaches can be viewed as complementary to fulfill the complete spectrum of sensor networks' privacy needs.

Keywords. privacy, anonymity, sensor networks

Introduction

In the forthcoming era of ubiquitous computing, sensor networks are expected to have a significant impact by providing timely and accurate data to new classes of context aware monitoring applications. Because of the increased data collection capabilities that large scale sensor network deployments can offer, they are not only envisioned to be globally deployed for commercial, scientific or military purposes, but also to be integrated into our daily lives, providing context rich information for highly sophisticated services. The deployment of pervasive sensing environments that contain numerous almost invisible sensors that constantly monitor their surroundings, collect, process and communicate a variety of information, inevitably causes concerns related to their potential of abuse and the risks they impose to the privacy of individuals. The potential risks are aggravated by the fact that different types of sensor networks may be deployed for different purposes, others being trusted, for example subscription based sensor networks offering health services, others being partially trusted, like these deployed for customer assistance in shopping malls, and others untrusted, like surveillance networks the users might be completely unaware of. Even sensor networks initially deployed for legitimate purposes

may be abused or violated. Historically, it is believed that as surveillance technology has become cheaper and more effective, it has been increasingly used for privacy abuses [1].

As for any ubiquitous computing technology, privacy concerns need to be addressed in order to gain user acceptance in the long term. In order to preserve privacy, one primarily has to ensure that sensed information is confined to the sensor network and is accessible only to authorised parties, with the individuals being empowered to control how their personal information is obtained, processed, distributed, and used by any other entity. The notion of privacy is related more to controlling disclosure of personal information in exchange of some perceived benefit, than to ensuring complete secrecy or anonymity. The fact that the perceived benefit and the purpose of data collection are the main determinants of personal data disclosure decisions has been verified by experimental studies on the value that individuals assign to location information [2].

However, the infrastructureless nature of sensor networks, the computational capability limitations of sensor nodes, and the fact that in-network data aggregation operations are performed, are only some of the challenges that the designers of privacy preserving sensor networks face. Moreover, an issue that must be taken into consideration during the design of realistic security mechanisms is that security services do not provide core network functionality, and, being supportive services, they impose reasonable computational overhead. Existing privacy solutions, like these suggested for internet communications, are not considered feasible for the resource-constrained sensors. Moreover, since they mostly focus on protecting the contextual aspects of the communications, they can not fulfill the complete spectrum of the sensor networks' privacy requirements, that also includes issues such as controlled information gathering and aggregation, and privacy sensitive disclosure of information of varying levels of detail.

This paper is structured as follows: In Section 1 the privacy requirements of sensor networks are identified and discussed. Section 2 outlines the mechanisms that have been proposed to meet these requirements for traditional networks. Section 3 discusses why these mechanisms are inapplicable in sensor networks, by outlining the main constraints that apply and the challenges confronted. The solutions that have been proposed for sensor networks and correspond to each of the identified requirements are then categorised and discussed in Sections 4, 5, and 6. The paper concludes with some remarks in Section 7.

1. Privacy Requirements

Preserving the privacy of the individuals that act within an information system generally entails keeping their personal information confidential and accessible only to authorised parties. The narrow perception of privacy as secrecy and access control, however, can not cover the wide range of issues that fall under the notion. Especially in the case of sensor networks, privacy can be addressed at multiple levels of the network stack and at different points of the sensed information flow, depending on issues such as which entities can be perceived as trusted by the users, and what the architectural and technological limitations of each deployment are.

The privacy requirements of sensor networks are separately discussed in two dimensions: the *application and service privacy requirements* as they are identified from the user's perspective, and the requirements *as they apply to the design* of privacy preserving

sensor networks. Although specific design level privacy requirements depend both on the application space of a sensor network, which determines the sensitivity of the collected and communicated information, and the users' preferences, roles and level of trust to the service provider, strong privacy will be achieved by designing a sensor network so that the requirements discussed in Section 1.1 are satisfied. The requirements in the application and service dimension, which are mainly related to anonymity and location privacy, are discussed separately in Section 1.2. The discussion of the solutions that have been proposed for sensor networks in the subsequent sections corresponds to the categorisation of requirements in the design dimension since, once these are addressed, the application and service level privacy requirements are fulfilled as a result.

1.1. Requirements for the Design of Privacy Preserving Sensor Networks

For sensor network deployments to be privacy preserving, one has to ensure that the sensed data is protected from disclosure both to adversaries and to illegitimate service providers. The fist two requirements below address the issue of *how* the data should be communicated in the network so that it is protected from eavesdroppers, while the next two address *what* data should be collected and released to data requestors so that the privacy of the individuals is preserved.

Confidentiality of the sensed data As for all privacy preserving systems, data confidentiality must be ensured through message encryption. Encryption is the typical defence against eavesdropping, and the only method for preserving the confidentiality of the content of exchanged messages. For sensor networks this requirement becomes even more crucial, because an outsider can induce information by correlating the results reported from multiple sensors surrounding an individual. Moreover, because of the in-network data aggregation operations, data of different granularity and sensitivity with respect to the user's privacy is being communicated and needs to be protected [1].

Protection of the communications context Ensuring the confidentiality of the messages' content does not always suffice, since an adversary might induce sensitive information by observing the communications' contextual data, especially since they can be correlated with prior information about the people and the physical locations that are being monitored by a set of sensors. For example, the disclosure of both spatial and temporal data through traffic analysis, may allow tracking the relative or actual -through correlation with prior knowledge- location of the mobile sensor nodes that might be carried by users, which would constitute a serious privacy breach. Independently of what encryption scheme is being used, the cipher texts should not allow induction of any information related to their context. The information that can be considered sensitive is the network identity of the communicating parties, the frequency of the communications, the traffic patterns, the size of the messages and the location and time at which the sensor's measurements are being sent. Especially for the identities of the communicating parties, multiple levels of protection can be identified: sender, recipient, or mutual anonymity per-

[1] Protecting data confidentiality is a standard network security issue, that also has to be addressed to provide privacy. Encryption schemes for sensor networks are the focus of Chapters 3 (Symmetric Primitives) and 4 (Public-key Primitives), and will not be discussed further in this chapter.

tains to protecting the identity of the source, the destination, or both nodes, while sender and recipient unlinkability protects the relation of the nodes from inference by third parties. Techniques like timestamping, padding, using serial numbers, or frequent key redistribution can be used so that the communicated cipher texts do not reveal information through their similarity or size. However, protecting the traffic patterns within the network and the identities of the nodes is not trivial, as it requires interference with the routing protocol.

Privacy Sensitive Information Disclosure In a setting with multiple data collectors, with varying levels of trust associated to them by each user, that offer a variety of services in exchange of information, users need to be empowered to reject, accept or negotiate the release of private data. Empowering sensor network users to control the level of information privacy according to the context, their role and communication partner mainly entails two actions: First, mechanisms should be provided to allow users to define their privacy preferences and to inform them what privacy policies are being announced by data requestors. Second, having provided the users with access to information about the service provider, the service offered and the purpose of data collection, mechanisms that enable the enforcement of their preferences would allow them to control the disclosure of their personal data, its level of detail, and the pseudonym utilised. The issues that thus mainly need to be addressed are related to data access control, data granularity control, and protection from inference through information correlation. The issue of context awareness for the release decisions is also crucial. It has been argued, for example, that the system needs to support special exceptions for emergencies in crisis situations, where safety outweighs privacy needs [3]. Moreover, the enforcement of privacy preferences should be made transparently and with minimal user interaction demands for the system to be non-intrusive. The contolled information disclosure requirement can be addressed in the middleware and application layer, since it is related to the content of the messages, as opposed to the previous requirement about the context of the communications. For this requirement to be satisfied, one has to be assume that users trust the entities and devices up to the level where the privacy preferences can be enforced.

Privacy Sensitive Information Gathering Sensor network deployments can not be considered privacy sensitive unless two conditions are fulfilled: First, mechanisms should be provided for the notification of individuals within the sensing areas. Second, data collection should be restricted to the minimum required level for the services to be provided. User's notice and choice mechanisms address the first issue by providing awareness of data collection and requiring user consent. A first step towards addressing the second issue is restricting the network's ability to gather data at a detail level that could compromise privacy [4], for example through depersonalising the results reported by sensors or through applying discrete information flows instead of continuous, when continuous data is not required by the applications. This requirement is more proactive than the previous one, in the sense that it protects user privacy at the point of information capture, before any data release decisions are made. For this requirement to be satisfied, it has to be assumed that users trust only the sensor network that collects the data.

1.2. Application and Service Level Privacy Requirements

While many applications that sensor networks can support shall require the identification of users in order to provide some services, others will only require anonymised or pseudonymised data. Examples of the first case are medical applications gathering data from body sensor networks that monitor the vital functions of patients in hospitals, while examples of the second case are applications for traffic monitoring that gather data from environmental and vehicular sensor networks in order to extract traffic statistics. Providing *support for anonymous data* is an essential requirement in the application and service dimension, that can be addressed in multiple levels of the design requirements categorisation: The context of the communications needs to be protected in order to avoid the disclosure of the network identities of the user-carried devices, since this would allow adversaries monitoring network traffic and having some prior knowledge to correlate the network identities with the actual user identities. Privacy sensitive information disclosure mechanisms need to be provided for the data to be pseudonymised or anonymised before being released to applications that do not require user identification. Finally, privacy sensitive information gathering schemes could be applied to depersonalise the results reported by sensors during the data aggregation process.

Independently of how it is addressed, the anonymity requirement is set to allow individuals to use the network services, while protecting their identity both from adversaries and from data requestors. Anonymity can be defined [5] as the state of not being identifiable within a set of subjects, called the anonymity set, which is the set of all possible subjects who might cause an action. The size of the anonymity set can thus be used as a naive degree of anonymity, i.e., the degree of how indistinguishable the users remain within their anonymity set. It is however been argued that "'Anonymity is the stronger, the larger the respective anonymity set is and the more evenly distributed the sending or receiving, respectively, of the subjects within that set is"' [6]. The degree of anonymity depends not only on the size of the anonymity set, but also on the distribution of probabilities that an observer can assign to the entities within the set for causing particular actions or for assuming particular roles. For example, an adversary observing anonymised network traffic could assign probabilities to each sender as being the originator of the message, based on the information that the system leaks, through traffic analysis, timing or message length attacks [7]. In order to accurately quantify anonymity, metrics based on entropy have been proposed [7,8], that take into account the probabilities that observers can assign to different members of the anonymity set.

For legitimate network services that require user identification, the identity of the user can be secretly shared between him and the service provider through the use of pseudonyms. The scope of pseudonyms may differ according to the context of their use, providing different strength of long-term linkability [5]. A static person pseudonym would be the weakest, while the establishment of transaction pseudonyms can be used to achieve strong unlinkability. A role-relationship pseudonym, different for each service provider and user role is an intermediate case that would protect against identity information correlation on the service provider's side. However, even if data is anonymised or pseudonyms are being used, each user's anonymity depends on his anonymity set: if it shrinks, his identity can be disclosed. A solution would be to use, together with the anonymity mechanisms, dummy traffic or background noise [9].

Another requirement in the application and service dimension is the one of *location privacy*. Location is considered as sensitive information both because the unauthorised

access to information related to the actual or relative locations of users constitutes a serious privacy breach, and because accurate location information enables the correlation of network with actual user identities. Location information can be captured either by the located object itself by using some positioning technology like GPS, or by some external entity like a wireless network operator, using techniques such as signal triangulation. In the case of smart vehicles, for example, location information can be captured using the roadside infrastructure through distance bounding and multilateration [10]. Location information can also be captured through environmental sensor networks that monitor the presence of individuals in particular locations. Independently of the localisation technology utilised, location information is reported by the entity that captured it, called location information provider, to some provider of location based services. These services can either be user-initiated, like tourist information services that report the areas of interest around the user, or continuous location tracking services, where the users' location is frequently reported, as in the case of traffic monitoring applications. Location privacy concerns are more related to the case of location tracking services, where location information discloses the user's movements.

Similarly to the anonymity requirements, location privacy can be addressed at multiple levels: The context of the communications can be protected in order to avoid location tracking from adversaries that monitor the network traffic and can identify the messages transmitted by user-carried devices. Privacy sensitive information disclosure mechanisms can ensure that the external service providers receive data of an accuracy level that is in accordance with the user's preferences. Finally, privacy sensitive information gathering schemes can be used both to notify the users within sensing areas, and to ensure that the level of detail of the captured location information is the minimum required.

2. Privacy Protection Mechanisms for Traditional Networks

There exist significant differences between the privacy requirements identified for ubiquitous computing and sensor networking, and the privacy requirements for the case of internet, mail and telephony technologies applied to traditional wired networks . The most important privacy risks in online transactions are these related to the context of the communications, and more specifically to the anonymity of the clients and to the unlinkability of requests. The mechanisms that have been proposed for enhancing web privacy [11] are focused on protecting the contextual aspects of communications, and on enabling informed and controlled online interactions through the enforcement of privacy policies. Although most approaches proposed for traditional networks are considered inapplicable in the case of sensor networks for the reasons that are later explained in Section 3, many privacy schemes and mechanisms for sensor networks have borrowed aspects or are designed as lightweight versions of these.

The World-Wide-Web Consortium's *Platform for Privacy Preferences Project (P3P)* provides a framework for enabling users to apply preferences over privacy policies of web sites and online services, that may describe why and what data is being collected, by and for which entity. P3P allows the encoding of privacy policies into machine-readable XML, and enables automated processing and decision making based on personal privacy preferences expressed using a machine-readable preference language such as APPEL. It can be considered as a protocol for exchanging structured data: it includes specification

of both syntax and semantics for describing privacy policies and preferences, and of the mechanisms for users agents to get access to the announced privacy policies. It is however more of a privacy enabler than a privacy enforcer, since it does not enforce privacy or anonymity through technology, but relies on the trustworthiness of organisations to comply with the privacy policies they announce. It thus acts as complementary to other web anonymity tools and protocols.

Approaches that address privacy and anonymity issues for the contextual aspects of communications either interpose an intermediary between the sender and the receiver to provide sender anonymity, or use a network of intermediate nodes, which can protect both the server and the receiver identities from eavesdroppers. An example of the first category is the *Anonymizer* web anonymity tool, where a web proxy or a group of proxies act as intermediaries, receive the web requests from users and filter the client identification information from the request headers before forwarding to web servers. The identities of the clients are not revealed to the web servers, but only to the Anonymizer service, which is the only component that needs to be trusted. Although Anonymizer is effective in hiding the user identity from the web server, this does not apply for eavesdroppers that may monitor the traffic between the user machine and the Anonymizer server.

The first approach in the second category introduced the concept of *MIXes* [12] to address the anonymity and untraceability issues for email communications in the presence of eavesdroppers performing traffic analysis. The basic idea of this approach is that each message is sent over a series of independent servers, called mixes. Each mix collects a number of messages from multiple sources, performs cryptographic transformations on them to change their outlook, and forwards them in a random order to the next destination, so that eavesdroppers cannot link incoming messages to outgoing messages. *ISDN-MIXes* [9] were proposed as an extension of the basic mix approach, where the use of mixes is combined with dummy traffic and with broadcasting of the encrypted incoming messages.

The *onion routing scheme* [13] is also a mix-based approach, where during an initialisation phase the sender determines the message route path to the destination through a series of onion routers. Encrypted routing information is repeatedly added to the payload, so that the message finally transmitted is an onion consisting of several layers of encryption that are stripped off by the onion routers as it traverses the path. Because of the onion structure, each router in the path can only determine the previous and next hop, so the identities of the communicating parties are protected. Onion routers also act as mixes, to make the path untraceable not only by intermediate routers but also by eavesdroppers.

An alternative approach to protecting anonymity is the *Crowds* protocol [14], which is based on the concept that any member of a crowd can remain anonymous if his actions are indistinguishable from the actions of other members of the crowd. Like the other approaches, Crowds uses a set of intermediaries, called jondos, logically positioned between the sender and the receiver. The routing path is not determined by the sender, as in Onion Routing, but randomly chosen at each hop that the message traverses. Moreover, the message remains the same along all hops of the path, so that no jondo can distinguish whether the previous hop in the path was the sender or a forwarding node. Once a path is established, it is used for all communications between the source and destination, with each jondo communicating with its predecessor or successor on the path. The *Hordes* protocol [15] similarly uses multiple jondos to anonymously route a message toward the receiver. Instead of using the reverse path of the request, Hordes routes the replies back

to the sender through multicasting. Except from ensuring sender anonymity, this allows the use of the shortest reverse path, thus reducing the communication latency.

An evaluation of these anonymisation approaches with respect to their architecture, operational principles and vulnerabilities is presented in [11]. Overall, it is considered that the better the level of anonymity protection, the greater the overhead latencies. Crowds is considered less expensive than Onion Routing as it does not require complex cryptographic operations, such as multi-layer encryption. Hordes incurs lower overheads despite the fact that it uses the same mechanisms as Crowds, since Hordes' members do not perform any complex tasks during the backward routing procedure. The anonymity properties of these approaches in the presence of compromised nodes have been studied in [16]. It was found that the probability that the true identity of a sender is discovered might not always decrease as the length of communication path increases, and that the complexity of the path topology does not significantly affect this probability.

3. Challenges in Protecting Privacy in Sensor Networks

While several privacy mechanisms, tools and applications have been proposed to enhance privacy and anonymity in traditional networks, these solutions can not be directly applied to the case of sensor networks. Their inherent properties at the node, network and data level pose challenges that are unique in the networks security area. The following issues are usually encountered when designing architectures, schemes and mechanisms that aim to protect, among others, the privacy of the network's users [17]:

Sensor node capabilities Sensor nodes are designed to be small and inexpensive and are thus constrained regarding their energy, memory, computation and communication capabilities. Due to their severe energy limitations, the number of transmissions required for each message to reach its destination has to be minimised. Routing schemes for sensor networks are thus designed to optimise the multihop path selection, as opposed to the mix-based approaches that randomise the path from the source to the destination to achieve untraceability. Moreover, the capabilities of the nodes pose limitations on the range of cryptographic primitives they can support. Traditional public key cryptography is considered unrealistic for sensor networks [18]. Techniques like onion routing, which relies on extensive asymmetric encryption operations for the construction of the onion using layers of encryption that intermediate nodes can strip off using their private key, would impose high computational overhead to all nodes in each path. The use of dummy traffic or background noise, that has been proposed as an extension to the MIX approach in order to achieve indistinguishability [9], would exhaust the energy supplies of the sensor nodes. Overall, the privacy mechanisms that require either extensive encryption operations or additional network traffic because of the message routing strategies they include, incur overheads that are prohibitive for use in resource constrained sensor nodes.

Communication issues The wireless nature of sensor network communications and the standardised communication technologies that are employed makes it even more challenging to satisfy most of the set requirements, as the wireless medium exposes information about the network traffic. Sensor networks are more vulnera-

ble to eavesdropping and traffic analysis attacks, since an adversary does not need to gain physical access to the networking infrastructure. Moreover, in the general case, sensor networks are infrastructureless and dynamic. The lack of central servers and static base stations, the dynamically changing network topology, the possibility of addition or deletion of sensor nodes through all stages of their life cycle, all combined with the scale of the deployments (hundreds or thousands of sensor nodes), set strict requirements on the privacy schemes that can be used: they need to be distributed, flexible, scalable, and cooperative. Privacy schemes that require some designated proxy to protect the anonymity of the source nodes, like Anonymizer, can not be directly applied. Privacy policies negotiation schemes like P3P require the existence of network nodes that make privacy reference files and privacy policies available, but for sensor networks the availability of such nodes can not be guaranteed. The lack of centralised host relationships and the transmission range limitations also affects the routing mechanisms of the data packets, that will typically follow multi hop routes before arriving at their final destination. The trustworthiness of the intermediate nodes of each path can not be guaranteed a priori.

Data handling In order to minimise communication overheads, large streams of data are converted to aggregated information within the sensor network by data aggregator nodes. The fact that in-network processing is performed sets additional security requirements for node-to-aggregator node communication. In traditional networks, where each node communicates with some base station, most data and identity confidentiality requirements can be satisfied by end to end encryption and the use of pseudonyms. In sensor networks, however, for data aggregation purposes, intermediate node authentication will be required. Moreover, possibly untrusted aggregator nodes gain access to large streams of data that may be sensitive. The increased complexity in the flow of information makes it challenging to enforce privacy sensitive data gathering mechanisms. Issues related to how user pseudonyms can be handled in the presence of aggregator nodes, and which raw or aggregated data can be anonymised at each point within the network, also need to be resolved.

Although most privacy protection schemes are considered unsuitable for sensor networks because of these characteristics and constraints, they are not entirely rejected. Similar architectures and mechanisms have been proposed for sensor networks that introduce modifications and adjustments to the basic schemes in order to be more lightweight and less dependent on fixed infrastructures. For example, network identity privacy approaches borrow aspects from schemes like Crowds, while being designed to be more targeted for ad hoc and sensor networks. Privacy policy enforcement schemes use modified versions of P3P policies, while adjusting the privacy negotiation protocols to fit the decentralised nature of sensor networks. As discussed in the next sections, many basic aspects for protecting privacy are borrowed from traditional networks and adjusted to fit the case of sensor networks.

4. Protection of the Communication Context

The messages communicated in a sensor network, independently of their content or of the encryption scheme that is being used, should not allow induction of information re-

lated to their context by adversaries that can overhear them. Techniques like message timestamping, padding, using serial numbers, or frequent key redistribution can be used so that the communicated cipher texts do not reveal information through their similarity or size. Examples of other contextual information surrounding sensor nodes that could be derived are the location or the time of measurements. In many scenarios, this information is considered sensitive and needs to be protected. In remote monitoring applications where sensing devices are used to track assets of significant value, it would be necessary to avoid the disclosure of their locations to malicious entities. In military applications where wearable sensors are being used in order to securely monitor the status of soldiers, it would be crucial to avoid the disclosure of their movements to the enemy that might monitor the wireless communications.

However, protecting the traffic patterns within the network, which include the network identities and the relative locations of the sender of data packets, is not trivial, as it requires interference with the routing protocols. The solutions that are presented in this section aim to obscure the communication traffic patterns from adversaries through routing strategies that either protect the source location or the network identities from communication traffic observers.

4.1. Source Location Privacy

In the general case of large scale sensor networks, messages will follow multihop routes from the source node to the destination. The shared wireless medium makes it feasible for traffic observers to identify the origins of radio transmissions within their range through the use of localisation techniques. If the network traffic for a given time period can be correlated to distinguishable sources, a mobile observer can perform hop by hop traceback to the source node of each communication. The worst case scenario for preserving source location privacy is when there is traffic only from a single source in a network, like in the Panda-Hunter problem presented in Section 4.1.1. The routing protocols that are analysed in Section 4.1.3 aim to suppress the adversary's ability to successfully traceback the location of the message source node.

4.1.1. The Panda-Hunter Problem

The Panda-Hunter problem [19] pertains to how to enable monitoring of pandas through detection sensor nodes that have been spread over a sensing field, without exposing the location of the pandas to hunters. For simplicity, it is assumed that there is a single sink node for collecting the data and a single panda in the sensing field, that is detected by a single stationary source node each time. When the presence of a panda is detected, the corresponding source node P will start reporting data periodically to the sink node S through multihop routes. It is assumed that the messages are encrypted.

The adversary in this scenario is the hunter H, who tries to capture the panda by locating the node that reports the panda's presence by sending messages to the sink. The hunter starts at the location of the sink and is constantly in a receiving mode. Devices like antenna and spectrum analyzers allow the hunter to observe the messages and identify their immediate senders. By moving to the immediate sender node each time he receives a periodic message and waiting until the next message is received, the hunter makes consequent movements towards the originator node, which he will eventually traceback and capture the panda.

For each experiment in the Panda-Hunter problem, the initial location of the panda is random and the panda is not mobile. If the hunter reaches a specified hop distance from the panda within a threshold amount of time, the panda is considered captured. The primary goal for any message routing strategy that achieves source location privacy is therefore delivering the messages to the sink, while obscuring the location of the source node for a safety period, calculated as the number of periodic messages sent by the source before the hunter localises it. The likelihood of the panda being captured within the safety period highly depends on the traceability of the information leaked by the message routing strategy of the sensor network.

4.1.2. The Effects of the Routing Strategy

In order to examine the effects of the message routing strategy on the source location privacy protection, two popular classes of routing protocols for sensor networks are considered [19,20]: the class of flooding protocols, and the class of single path routing protocols.

In the simplest case of flooding, each message is broadcasted from the source node to its neighbours, who in turn rebroadcast until it eventually reaches the sink node having followed a set of different paths. Provided that the hunter's initial position is the sink node, the first message that will reach him has most likely followed the shortest path from the source to the sink. The hunter will be able to traceback the shortest path by moving each time to the last forwarding node and waiting for the next first message that arrives. Simple flooding thus offers the least possible privacy protection, since the safety period is equal to the shortest path length.

The privacy protection performance of the single path routing protocols is similar. Irrespectively of how the path from the source to the sink is selected by each protocol, since only the nodes that are on the selected path participate in message forwarding, the hunter can traceback the single message that is observed when located in each node in the path. The safety period is thus equal to the selected path length.

A strategy that can offer increased privacy protection performance is flooding while using fake message sources. In order for these to be indistinguishable from the real source, the fake messages that are produced should also be encrypted and of the same length as the real messages. For this technique to be effective in misleading the hunter, it was found [20] that the fake sources should be persistent throughout the experiment, they should generate messages as frequently as the real source, and they should have a distance from the sink similar to that of the real source. Even by using fake sources to produce fake message paths, however, a persistent hunter will eventually select to follow the message path to the real source. At the same time, this strategy is too costly in terms of communication overheads and of energy consumption.

4.1.3. The Phantom Routing Strategy

The poor privacy protection performance of the routing protocols discussed in the previous section is attributed to the fact that they allow the use of stable message paths that lead the hunter from the sink to the source node. The phantom routing strategy [19] that was introduced for sensor networks aims to provide source location privacy through directing the periodic messages from the source node towards different paths in the network. This prohibits the hunter from receiving a stable stream of messages that would

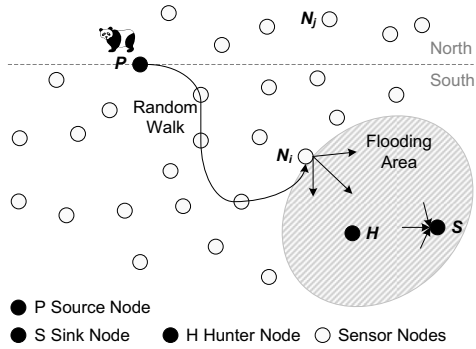


Figure 1. The phantom flooding strategy for source location privacy in the panda hunter problem

enable backtracing the source. Instead of that, by the received messages the hunter is led towards phantom sources.

The phantom routing strategy, depicted in Figure 1, consists of two subsequent phases for every transmitted message: a pure or directed random walk for a given number of hops that directs the message to a phantom source N_i away from the original source P, and a message flooding phase that delivers the message to the sink S. An alternative that has been proposed in [20] for the second phase is the use of single path routing instead of flooding. As long as the random walk of the first phase leads to a different phantom source for every message, both approaches are equally effective in increasing the safety period. If the hunter H detects a message forwarded by node N_i and moves to that node to get closer to the source, the next message is unlikely to follow the same random path, thus making the hunter's previous move worthless. A further advantage that the random walk offers is that the safety period improves as the network size increases, as the paths followed by subsequent messages, and consequently the phantom sources, become more diverse.

The diversity of the paths is not, however, the only issue that the random walk implementation needs to ensure. The main purpose of this phase is to send each message to a phantom source that is far from the original source. In a pure random walk, if the network is uniformly deployed, the message will likely loop around the source node for the required number of hops. A solution to this problem is to use directed walk for the first phase by dividing the neighbouring nodes of the source into two sets, for example north and south as in Figure 1, and having the source node randomly pick one partition for forwarding. All nodes in the path, starting from the source node, will choose to forward the message to a random neighbour from the partition initially selected. The partitioning of the network nodes can either be sector-based [19], depending on their relative locations, or hop-based [20], depending on their distance from the sink in hop counts.

The self-adjusting directed random walk approach [21] was proposed in order to overcome a serious problem that was observed in the two previous approaches: Their performance would drop if the source was located in certain areas of the network where there were not enough nodes of the initially selected partition to perform the given number of hops in a truly random way. In Figure 1, where the nodes were divided according to their vertical location in relation to the source, node N_j would be used as a phantom source for half of the messages, i.e., every time the north partition was selected. In the

self-adjusting directed random walk approach, the nodes are divided into four partitions, two for each dimension, so that each node is included in one set for each dimension. The initially selected partition can then be changed by the intermediate nodes if it is found that the message can not be further forwarded to the given set for each dimension.

The greedy random walk approach [22] is a two way random walk, performed both from the source and the sink. It was inspired by the observation that if the hunter had gained a good coverage of the network by distributing a number of observation points around the sink, the source location could be approximated because the flooding phase would reveal too much information. In order to avoid this, instead of using flooding to deliver the message to the sink, the sink sets up a random walk which serves as the receptor of the messages. Each message is randomly forwarded from a source until it reaches the receptor, and is then forwarded to the sink through the pre-established path.

Phantom routing strategies, independently of the random walk selection techniques that are used, offer increased safety period compared to traditional sensor network routing protocols. Source location privacy increases the largest the network is, the more hops the random walk contains and the more mobile the source can be. Moreover, compared to basic flooding, the energy consumption, which mainly depends on the number of the transmitted messages, is not increased. The message latency, however, could be significantly increased depending on the length of the random walk.

4.2. Network Identity Privacy

In the panda hunter problem, the panda can remain hidden within the network only as long as the source node remains out of the network area that the hunter monitors. In the case of an adversary with multiple observation points that has gained enough coverage of the network, hop by hop trace back would not be required, since all messages, including the one from the source node, would be overheard. Even in scenarios where messages are not transmitted by one single node, like in the panda hunter problem, but multiple nodes are transmitting messages at the same time, adversaries would be able to distinguish the ones that originate from a given source. Message sources are distinguishable, since the routing information that is embedded in the packet headers includes permanent identifiers, like network addresses, of the communicating nodes.

Serious privacy breaches can be caused because of unprotected network identities, especially in the case of wearable sensor nodes, where the network identities correspond to user-carried devices. By observing the network traffic, adversaries can trace the motion patterns of the mobile nodes over the periods of time during which they overhear communications. In a military application where wearable sensors are being used in order to monitor the status of soldiers, their number, relative location and movement could be disclosed to an enemy that has deployed a surveillance network and analyses the headers of the overheard packets.

The aim of network identity privacy approaches is to enable message transmission without disclosing the exact permanent identities of the communicating nodes. While onion routing and mix-based techniques can be used to meet this requirement in traditional networks, they are inapplicable for sensor networks for the reasons described in Section 3. The approaches that can be applied to sensor networks meet this requirement by using either node or route pseudonyms instead of permanent identities for routing.

4.2.1. Node Pseudonymity Approach

The permanent identities of the communicating nodes are protected by the node pseudonymity approach through the use of mutually verifiable temporary pseudonyms. The pseudonyms are used as common identifiers that replace the identification information in the header of the exchanged packets, so that they appear as unlinkable random identities for anyone except the sender and the receiver. In order to achieve unlinkability between communications, pseudonyms should be frequently updated. For the pseudonym update process to be resilient against pseudonym correlation attacks, mechanisms that utilise the concept of silent periods have been recently proposed [23]. Silent periods are defined as transition periods between the use of new and old pseudonyms that introduce ambiguity into the spatial and temporal relation between the node's disappearing and emerging locations and times. These mechanisms, however, require the existence of pseudonymisation authorities to handle the synchronisation.

A distributed approach to node pseudonymity was proposed in [24]. The toolbox for the use, generation and update of unlinkable pseudonyms P_i for node i consists of symmetric keys K_{ij}, random nonces R_n and hash functions H. A three-way handshake designed as modification of the ID packet from the original Bluetooth specification can be followed for the pseudonymised communication between nodes A and B that share a symmetric cryptographic key:

$$A \rightarrow B : (R_1 | H(P_B | R_1 | K_{AB}))$$
$$B \leftarrow A : (R_2 | H(P_A | R_1 | R_2 | K_{AB}))$$
$$A \rightarrow B : (R_3 | H(P_B | P_A | R_1 | R_2 | R_3 | K_{AB}))$$

Node A that initiates the communication chooses a random nonce R_1, computes the hash that is used to protect B's pseudonym and sends the message to node B. The next two steps are needed in order for both nodes to verify that they both know their pseudonyms and their shared key. The three random nonces are used in order to ensure the freshness of the messages. The protection that the handshake provides to the pseudonyms is based on the randomness of the nonces, the shared key and the use of the hash function.

Another approach that was proposed for wireless personal area networks [25] uses the symmetric key to produce a chain of pseudonyms for each A and B. The initial mutually verifiable pseudonym is computed by using as input to a pseudo-random function the shared key K_{AB} and a random publicly known value. After each communication, the pseudonym is updated by using the old pseudonym as input to the pseudo-random function. Because of these updates, the header of each message between the two nodes will contain a different identifier, that can be linked to the previous ones only by the nodes that know the secret key.

Although the node pseudonymity approaches succeed in protecting the network identities when two nodes communicate, an issue that is insufficiently addressed is how these solutions could scale to multihop transmissions. For single path routing protocols, for example, either the intermediate nodes would need to possess some information on the pseudonym of the destination, or the source node would need to know the pseudonyms of all intermediate hops. For large scale sensor networks, synchronisation issues for the update of pseudonyms would also need to be addressed.

4.2.2. Route Pseudonymity Approach

The alternative approach for the protection of the permanent identities of communicating nodes against adversaries performing traffic analysis is the route pseudonymity approach. This approach does not provide any identification for the sender and receiver in the packet headers. Instead, it pseudonymises the routes that the messages follow during their multihop transmissions. Data forwarding is performed through unlinkable pseudonyms that are assigned to each hop of the message route.

The problem of developing untraceable routes through the use of route pseudonyms is addressed by the anonymous on-demand routing protocol [26] that was proposed for mobile ad hoc networks. The protocol uses an on-demand route discovery process to establish route pseudonyms through randomly naming each transmission hop and recording the mapping between subsequent hops in the forwarding table of each node. At the end of the process, each hop in the route is associated with a random route pseudonym $P_1 \dots P_n$. The protocol is based on the concept of broadcast with embedded trapdoor information tr, that is known only to the receivers of data packets that can open the trapdoor and provide proof pr for it, so that data is anonymously delivered only to them. The route discovery process by a communication source N_s for a receiver N_r is initiated by assembling a request packet and broadcasting it. The request packet contains a unique sequence number $secNum$, the trapdoor for the receiver and a cryptographic onion. The onion is formed by encrypting a tag that denotes the source using a random symmetric key K_s. Each intermediate forwarding node $N_1 \dots N_n$ that receives the request packet, embeds a random nonce $R_1 \dots R_n$ to the cryptographic onion and encrypts it with it own random symmetric key $K_1 \dots K_n$:

$$
\begin{aligned}
N_s &\rightarrow N_1 : \langle Request, secNum, tr, K_A(src) \rangle \\
&\vdots \\
N_n &\rightarrow N_r : \langle Request, secNum, tr, K_n(R_n, K_{n-1}(R_{n-1}, \dots, K_A(src))) \rangle
\end{aligned}
$$

When the request packet is received by N_r, the embedded cryptographic onion is a valid structure to establish an anonymous route towards the source. The receiver opens the trapdoor to get the proof pr that it embeds in the response packet along with a randomly selected route pseudonym P_r and the received onion. The response packet is bounced back to the source. Every intermediate node N_i that can peel off one layer of the onion using its symmetric key K_i, i.e. has participated in the route path of the request packet, selects a random route pseudonym P_i, stores the association $P_i \rightleftharpoons P_{i+1}$ in its forwarding table, replaces P_{i+1} with P_i in the response and rebroadcasts it:

$$
\begin{aligned}
N_r &\rightarrow N_n : \langle Response, P_r, pr, K_n(R_n, K_{n-1}(R_{n-1}, \dots, K_A(src))) \rangle \\
&\vdots \\
N_1 &\rightarrow N_s : \langle Response, P_1, pr, K_A(src) \rangle
\end{aligned}
$$

Once the source node receives the response and verifies pr, it stores the outgoing route pseudonym P_1 in its forwarding table, which can be later used in the headers of the

packets for anonymous data forwarding. When a message is transmitted by the source, the node in the set of the local receiving nodes that has the stored association for P_1 replaces it with the matched outgoing pseudonym and broadcasts the changed packet. The process is repeated until the packet reaches the destination.

Since node identities are not used for message routing, and the unlinkable route pseudonyms have the scope of a single hop for each sender and receiver, adversaries overhearing communications around any node participating in data forwarding can neither identify the communicating parties nor traceback the messages. Furthermore, the described protocol is intrusion tolerant. In the case of a compromised node, only the route pseudonym correspondences in its own forwarding table would be revealed, thus the effects would be localised. The route pseudonymity approach is better fitted for multihop communications than the node pseudonymity approach, because of the pseudonym synchronisation issues of the latter. Regarding the processing and communication overheads of the route pseudonymity protocol, it does require the use of expensive message flooding techniques, cryptographic operations and trapdoor functions, but only during the route discovery process.

5. Privacy Sensitive Information Disclosure

As opposed to the *context* protection mechanisms discussed in the previous section, the privacy sensitive information disclosure schemes aim to protect the *content* of the messages from disclosure to illegitimate entities. Guaranteeing strong privacy through complete or minimal disclosure of information is not desirable in all situations. Some services that users may perceive as useful shall require the disclosure of sensitive information. The solutions that are discussed in this section aim to control the privacy tradeoffs according to the service information requirements and the restrictions applied by user privacy preferences. The mechanisms that are discussed aim to enable users to control if any of their personal data should be disclosed, and at what level of detail, according to the context surrounding data requests. The main issues that are addressed are related to data access control, data granularity control, and protection from inference through information correlation.

5.1. Privacy Policies and Preferences for Access Control

The control of sensitive information flow from deployments that collect the data to applications that request the data in order to provide services is typically addressed through the use of privacy policies and preferences. The more complex the data collection and distribution environment is, the more challenging the specification and enforcement of privacy preferences becomes. The case of location based services is a typical example: location information can be collected by multiple deployments with varying levels of trust that use different technologies, like networks that identify and locate user-carried devices or environmental sensor networks that monitor the presence of individuals, and distributed by some middleware service to multiple, possibly untrusted, user-centred or location-centred applications.

In one of the earliest approaches to privacy preferences enforcement in environments with different administrative entities, trusted user agent components that reside on user

controlled devices were proposed to act as intermediaries, collecting and controlling access to personal information through predetermined access policies [27]. A distributed location query service responds to location-centred requests of external third parties. Location brokers residing in the middleware layer are used to interact with the user agents in their regions. However, although this approach enables access control in a decentralised manner, it requires the user agents to collect all information required for access control decisions.

The middleware service that allows applications to query the locations of users independently of the underlying technologies being used is provided by a centralised location server in [28]. This service is designed to control the distribution of location information by using machine readable privacy policies, defined as an extension of P3P policies, and without the need for repeated user intervention, in order to minimize intrusiveness. It uses user-defined privacy preferences to determine the acceptance or rejection of information release requests, while requiring user approvals only if the request can not be handled by the established preferences. The preferences that govern access permissions can include factors like the organisation requesting the data, its information handling policies, the type of service that is offered, time, location and context limitations, and constraints on the types of requests that can be accepted, i.e., specific user location requests, enumeration requests for specific locations, or asynchronous requests for information on specific events, like when users enter or leave specific areas. An issue that was studied in this approach is how user privacy preferences will be specified. It is proposed that default preferences are provided by service providers and other trusted organizations, and simple tools like wizards are used to help users create appropriate configurations.

When the users subscribe to the location server to make their location information available for external applications, they register their privacy preferences, which take the form of system components, called validators. When an application requests location information, it includes with the query a statement of its privacy policy, in order for the validators to evaluate the acceptability of the privacy policy, to decide if the requested information will be released, and whether any special constraints, like for the acceptable data accuracy, should be imposed. Multiple validators may exist for a single user, at least one corresponding to each of his identifiers, and the data release decision may be taken cooperatively. Potential validator components include user confirmation components and external validation services, for example services providing information on the requestor's reputation. Anonymity support is provided through enabling the validators to determine for each request which user identifier will be returned - the long-term identifier, a short-term identifier associated with the user, or a new randomly generated identifier.

The centralised nature of the access control decisions by the middleware location service of this solution, however, raises scalability concerns. In the approach proposed in [29], it is considered that access control through location privacy policies enforcement should be performed in a distributed way. It should be flexible enough to support multiple sources that provide location information, that may be within different administrative domains or belong to different organisations, with different levels of trust. Privacy preferences in this approach similarly determine who can access location information, what level of accuracy is allowed to be disclosed, and for which locations and time intervals. One significant difference from the previous approach is that privacy policies can be specified not only by individuals, but also by central authorities, like companies that

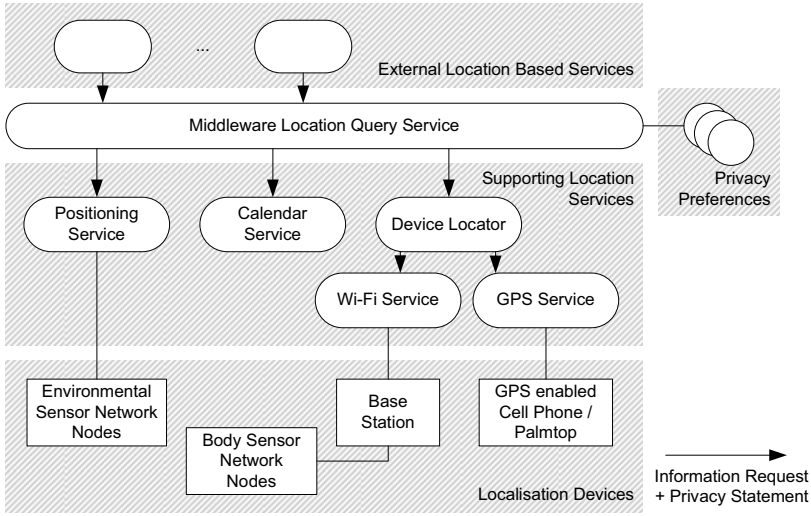


Figure 2. Example setting of distributed privacy policies enforcement for location based services

do not want the location of their employees to be disclosed to outsiders. Moreover, the scheme allows access rights to be forwardable.

The sources that provide location services are hierarchically organised as in the example of Figure 2, and each of them either uses a particular technology for collecting location information, like the positioning service, or processes the information received from other location services, like the device locator. In order for the hierarchically organised location services to cooperate in sharing information and propagating it to the upper layers for the requestor to receive it, the notion of service trust is introduced. Within the privacy preferences of each entity, the trusted location services are defined. Each trusted service implements access control by checking if the privacy preferences allow disclosing the requested information to the entity that issued or forwarded the request. This way, access control is performed in all steps of data collection or forwarding by the distributed location services. Privacy policies are encoded as digital certificates and the SPKI/SDSI scheme, originally proposed for decentralized trust management that supports specification and evaluation of security policies, is used for the implementation of the certificates.

Policy based access control is based on an alternative data model in the Confab framework [3], proposed in order to provide software architecture support for privacy-sensitive decentralized ubiquitous computing applications. According to its data model, all contextual entities, like users, services or locations, are assigned infospaces, which are network-addressable logical storage units that keep static and dynamic, intristic and extristic sensed data about these entities. Infospaces representing contextual information about users are kept in user-owned devices. Each piece of information in infospaces can be associated with privacy tags to describe how it should be handled when external entities request it. Infospaces contain operators for enforcing the privacy preferences of the user and the restrictions set by the privacy tags, and operators for user notification and interaction and for the evaluation of service descriptions that the applications publish

when requesting data in order to describe the type and granularity of personal information required for each provided service level.

5.2. Information Granularity Control

Privacy policies can be used not only to control if personal data will be disclosed within a specified context, but also to control the acceptable level of detail and accuracy of the disclosed data. In this sense, information granularity control approaches do not focus on controlling data release, but instead enable controlling the granularity of the data so that its disclosure is in accordance to the user's privacy preferences for the given context. Granularity control applies especially to location related requests, enabling the adjustment of location data accuracy. These approaches assume that accurate and detailed data is being collected by a trusted network and is released upon request to service providers. Because the service providers are not necessarily trusted by the users, the aim is to prevent the release of detailed data to them, by restricting data accuracy to the minimum level that is necessary for the services to be offered.

Except from location information, in the scheme proposed in [30], the observations of data subjects that can be protected through granularity control can include their identities, their speed during the observation, and the time that the observation was made. Information granularity control based on location privacy preferences is performed by the location providers, i.e., the trusted entities that collect the user's accurate location information and have some interest (e.g. for compliance with regulations or service agreements) in letting subscribers control the release of their location data. The service providers in this setting are the untrusted entities that require location data of some level of detail from the location providers to provide location based services. The granularity of private data is reduced prior to its release to the service providers according to the privacy preferences, the role of the requestors and the purpose of information usage. The responsibility for privacy policies enforcement in this scheme is thus split between location providers that are responsible for reducing the accuracy of the data, and the service providers that are responsible for restricting the usage of the data to the stated purpose. The accuracy of data related to the location, time, speed and identity of the user is modelled using lattice structures. Partial identification of users can be done by defining sets corresponding to sex, occupation, nationality, employer etc. The scheme enables adjustment of spatial, temporal and identity accuracy according to policies and data use purposes.

Although reducing information accuracy according to predefined policies can protect the exact location of users, this approach does not suffice in protecting their anonymity. In the approach proposed in [31], spatial and temporal information granularity is reduced according to the user's anonymity set size, in order for the disclosed information to be sufficiently altered to prevent reidentification. Similarly to the previous scheme, external service providers receive information through trusted location brokers. The location brokers act as a mixrouters by randomly reordering incoming messages to make them unlinkable with the outgoing messages, remove any identifiers such as network addresses, and reduce information granularity before releasing the information in order to make the subject k-anonymous, i.e., not identifiable within a set of k subjects comprising its anonymity set. With respect to location information, k-anonymity is achieved if the location information released is indistinguishable from the location information of at least $k - 1$ other subjects.

The quadtree-based algorithm that has been proposed for reducing information granularity meets an anonymity constraint in any location, regardless of population density, by decreasing the resolution of the revealed spatial and temporal data. It takes as inputs the accurate position of the subject, the coordinates of the area covered by the location broker, the positions of all other subjects reporting their location through it, and the minimum acceptable anonymity set size k_{min}. For reducing spatial accuracy, it works by subdividing the area around the given position until the number of subjects within the area falls below k_{min}, and returns the last acceptable quadrant, which represents the smallest area where k_{min} anonymity is preserved. For spatial resolution to be improved to a smaller area, anonymity can alternatively be preserved by adjustments in the resolution of the temporal dimension, by delaying information release until k_{min} subjects have reported from within the given area. Although this would allow for more precise anonymous information, it is unsuitable for time critical services, because of the delay.

In the database field, the work presented in [32] has a similar goal: to disclose person-specific information by adhering to k-anonymity while ensuring minimal distortion, so that the released information remains practically useful but the identity of the individuals who are the subjects of the data cannot be determined. K-anonymity is achieved through generalisation and suppression mechanisms. The difference of information granularity control that is discussed in this section, however, is that the data to be protected is dynamic and collected in real time.

5.3. Protection against Location and Identity Inference

The solutions that have been discussed in the previous sections to provide user identity privacy and location privacy in location sensing environments involved the enforcement of privacy policies and granularity control by trusted middleware services. However, for services that require precise location information, the use of privacy policies alone can not protect from identity and location inference by adversaries or illegitimate service providers that may be given access to location records. In order to protect identity and location privacy while allowing users to take advantage of location based services, mechanisms that protect from identity or location inference are proposed as complementary to the privacy policies enforcement schemes, for scenarios where granularity control can not be applied.

The problem of precise location information in location tracking environments that enables user identity inference can be tackled though pseudonymisation and the use of mix zones [33]. This approach assumes that the user trusts both the deployment that collects location information and the middleware service, but distrusts the location service providers. The middleware service acts as the access policy enforcement point and as an anonymisation proxy, responsible for the generation and frequent update of pseudonyms, so that users can not be identified by their presence in the reported locations. The purpose of the pseudonyms in this approach is the temporary identification of users so that they are provided with a return address for the services offered. However, even if the pseudonyms are frequently updated, their unlinkability depends on the size of the user's anonymity set: if it shrinks, his pseudonyms can be correlated. The concept of mix zones is used for pseudonyms to be updated in an unlinkable way. A mix zone for a group of users is the largest connected spatial region where no user has registered any location service callback. An example mix zone setting is presented in Figure 3. Mix zones can

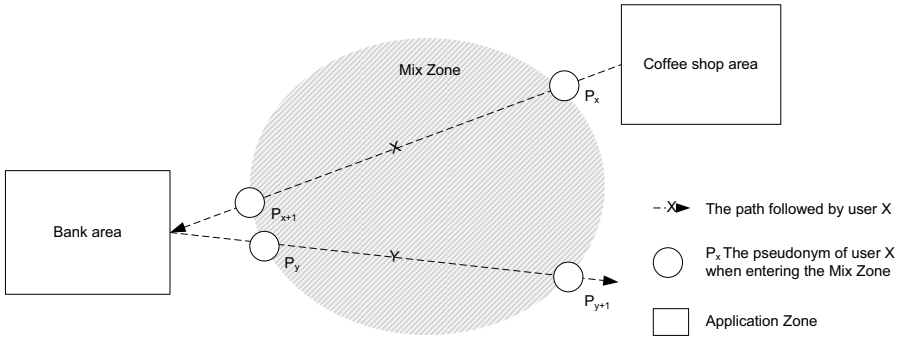


Figure 3. Sample mix zone arrangement with two application zones and two users

either be defined by the middleware service a priori or be calculated as the spatial areas that are not application zones, i.e., where no location service requests location information. Pseudonyms are updated when users are within mix zones, and therefore location services do not receive location information, so that the pseudonyms of the users coming into the mix zone can not be linked with the updated ones.

The anonymity set of any user in a mix zone is equal to the number of incoming people during his stay in it. The size of the anonymity set can therefore be used as a measure for the unlinkability of the pseudonyms and the resulting location privacy. It is, however, an optimistic measurement, since a powerful adversary could make intuitive observations and assign different probabilities to pseudonym correspondences. An example observation is the consistency that would probably exist in the direction people are moving towards when entering and leaving the mix zone. In Figure 3, for example, it is more likely that pseudonym P_x is linked to P_{x+1} than to P_{y+1}, since this would mean that the user changed his direction while being in the mix zone. The entropy metric is therefore considered as more accurate in measuring location privacy when using the concept of mix zones.

A different problem is the one of location inference of users that define in their privacy preferences some sensitive areas where their presence should not be revealed. This is an issue that can not be addressed by privacy policies enforcement alone. A solution that would protect against location inference in environments where periodic location reports are accumulated and sent to service providers should not only suppress location updates when the user is located within sensitive areas, but also suppress prior movement path data that would indicate or enable inference of the current location [34]. This approach also assumes that the user trusts both the location positioning network and the middleware service, and that the adversary is any entity that seeks to infer which sensitive areas the user visits through accessing the location records that are transmitted to external service providers. The middleware service uses location sensitivity maps that identify sensitive locations according to the user's settings, acts as a policy enforcement point and executes the disclosure control mechanisms to determine whether location updates can be sent to the requesting service providers.

An adversary should not be able to probabilistically infer user presence in sensitive areas from prior or future location updates. The mechanism that controls disclosure of location data should maximise position uncertainty when the user is in a sensitive area,

while minimising location information distortion when the user is outside sensitive areas. The k-area algorithm is used to suppress location updates in a region around a sensitive area, so that the area can not be distinguished from at least $k-1$ other sensitive areas that the user might have visited. The location sensitivity map is partitioned in zones, each including k sensitive areas. All location updates from each zone the user is moving within are stored and released to external applications only when the user crosses a zone boundary and has not visited any sensitive areas in the zone. This way, location accuracy is preserved when the user is moving outside sensitive areas, while making the sensitive areas the user visited indistinguishable.

As in the approach against identity inference, however, the size k of the set of indistinguishable sensitive areas is an optimistic measure of the location privacy protection that this approach offers. An adversary might have prior knowledge on the user's relationship to sensitive locations, which enables the assignment of different probabilities for the sensitive areas in each zone that the user is known to enter. Moreover, the algorithm discloses other information, such as the frequency and duration of visits to zones. At the same time, the delay that this approach introduces for the disclosure of information makes it inapplicable for time critical location based services.

6. Privacy Sensitive Information Gathering

The approaches that were presented in the previous section for safeguarding user privacy were based either on the use of privacy policies or on the reduction of data granularity before disclosing it to service providers. The first case assumes that the service provider is trusted to adhere to privacy policies, while in the second case, trust is assumed for the intermediary that is responsible for adjusting data accuracy before disclosing. However, even the deployments of legitimate intermediaries and service providers, may be passively or actively attacked, violated or misused. This section discusses controlled data gathering and user's notice and choice schemes that are more proactive, in the sense that they aim to protect user privacy at the point of information capture, before any data release decisions need to be made.

6.1. Restricted Data Gathering

The privacy sensitive data gathering approach aims to protect the privacy of users through restricting the sensor network's data collection capabilities to the minimum level that is required for the services to be provided. The privacy mechanisms of this approach are applied during data collection to prevent privacy-sensitive data from being accumulated in the sensor network, before intermediaries or service providers gain access to it. The only part of the deployment that needs to be trusted is the sensor network, since, if sensitive data is not collected at all, trust does not need to be assumed for how it is subsequently handled.

A type of applications for which sensitive data gathering can be applied are these that require aggregated information for the population density in certain areas without needing to track the movements of specific individuals, and thus without needing to identify them. Applications for road traffic monitoring or transportation schedule monitoring are examples of location-centric deployments. The approach presented in [35] used the

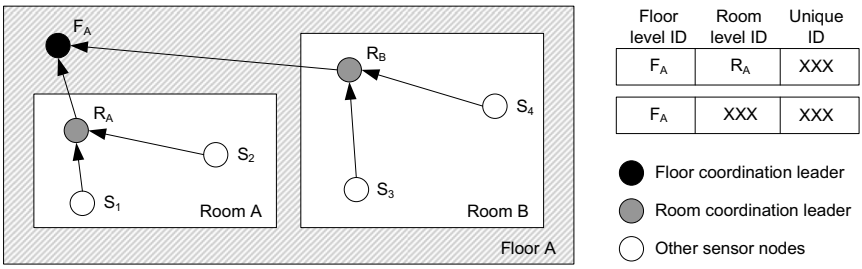


Figure 4. Privacy preserving data aggregation through blurred node identifiers in a building sensor network

example of a location sensor network for an in-building occupant movement tracking system like the one in Figure 4. The network is composed of a number of environmental sensor nodes capable of determining the number of individuals in the area monitored, some base stations and a location server that collects the data and makes it available to applications. In order to defend against traffic analysis attacks and to avoid the panda-hunter problem of Section 4.1.1, at the cost of increasing computation and communication overhead, messages are encrypted and data traffic is regularised by requiring all nodes to send at least one packet per data gathering interval even if they have no activity to report. Both the sensor nodes and the monitored areas are hierarchically organised, so that population statistics can be extracted for different levels of spatial resolution. Within this sensing environment, although no information about the identities of individuals is gathered, an adversary that has prior knowledge about the individuals and the spaces that they frequently use could link identities to the information reported by the location server and could track their movements within the area.

To counteract this threat, the data gathering capabilities of the sensor nodes are leveraged through applying distributed, in-network anonymity mechanisms that dynamically change data accuracy in order to preserve k-anonymity of the subjects within the sensor network before the data reaches the location server. In order to preserve k-anonymity while retaining the usefulness of the data, these mechanisms force the minimum necessary reduction in data accuracy in various steps of a hierarchical data aggregation process. The hierarchy reflects the spatial hierarchy of the sensed area, and multiple nodes in all levels of the hierarchy aggregate data while reducing its accuracy, so that no single node has a complete view of the data. Each node can be identified either uniquely or by the identifiers of the nodes that are above it in the hierarchy. The data accuracy is reduced by a rounding function in two ways: the spatial accuracy which is reflected by the identification level that the node which provides the information uses, and the accuracy of the number of subjects reported in the node's area. Given an anonymity level k_{min} for each node i, if the number of subjects k is above k_{min}, the unique node identifier S_i is used and the accuracy of k is reduced. Otherwise, the node identifier is blurred by using only the identifier of a higher level, as in the cases in Figure 4. Eventually, the information that reaches the location server is moderately accurate and only for the levels of the area hierarchy where the total number of subjects in their sub-regions exceeds k_{min}.

The hitchhiking approach [36] is targeted to the case of privacy sensitive user presence identification through carried devices instead of the environmental sensors alone. Like the previous approach, it was inspired by the observation that protecting user

anonymity through reducing the accuracy of location data can make location information useless for applications that are location-centric. It requires each user to approve reporting from each location he visits. The identifiers that are used for the messages to the location server are location identifiers, computed by the client devices based on the physical properties, e.g. GPS coordinates, of the location. Since the location server knows these physical properties, it can infer what location is being reported on without being able to infer the exact identity of the device that sent a report. The data gathering capabilities of the network are restrained only in the user identification dimension, by enforcing the location reports that are sent to the location servers to include only the total number of the users detected around the reporting device, without identifying them. Since location data is anonymised, the degree of location privacy in this scheme depends on the density of the population in each location, as this comprises each user's anonymity set.

6.2. User Notice and Choice through User Agents

The provision of user notice and choice functionality requires the addition of extra components in the general case of sensor networks architecture: the privacy assistants, being the user gateways to the surrounding sensor network applications, acting as intermediaries, and applying user-defined policies on information requests. User agent components are introduced especially in the case of sensor networks composed of user-related nodes, like body or vehicular sensor networks, that during their lifecycle may share information with services that are not fully trusted. As proposed in one of the earliest approaches in user controlled information disclosure in ubiquitous computing environments [27], the user agent components that reside on user controlled devices can collect and control access to personal information. Any request from external services for such information must be routed through the user agents that enforce predetermined privacy policies. By providing context awareness capability to the user agents, they can act as policy coordinators, enforcing context-sensitive and customisable access control.

In order for user agents to enforce privacy policies according to the context, their role and the communication partner, mechanisms should be included to provide awareness about whether data collection is being performed and what service privacy policies are being announced. These user notice mechanisms can either be provided by service providers or by third parties. A privacy awareness system that allows for user control in ubiquitous computing environments, assuming that the service provider is cooperating and is trustworthy, is pawS [37]. It includes mechanisms for the network service to announce its privacy policies and data handling practices, and for the users to apply their privacy preferences on accepting, declining or customising a service.

The pawS scheme includes a user agent component and a privacy beacon component, responsible for announcing the data collection requirements and privacy policies of the services offered. The scheme differentiates between two types of data collection that require different mechanisms for communicating the privacy policies to user agents: implicit announcement, when the user initiates the service discovery process and actively requests the service privacy policy, and active policy announcement for services working continuously in the background, in which case the user agent receives the policy from a beacon upon entering the data collection area. The announced privacy policy can include various levels of service customisations, together with the type of data required for each case, so that the user agent can determine the accepted service level according to the

predefined privacy preferences. The scheme also provides mechanisms for access and recourse of personal information through privacy proxies and privacy-aware databases. It empowers the user agent to keep track of data collections around the user, and enable or disable optional services based on the privacy preferences. It is, however, set as a prerequisite that the services are optional and configured to suit the users' decisions related to their privacy, and that the service providers are willing to cooperate. The scheme should thus be viewed more as a privacy enabler than as a privacy protector.

An alternative privacy awareness scheme that has been proposed for pervasive sensor networks [38] enables the user to conclude whether he is inside some sensing areas, without disclosing his exact position within the area. It is based on the protocol of secure two-party point-inclusion problem to test the privacy state of the user. If the user agent device is at a point p and a data collection area covers a polygon P, the secure two-party point-inclusion protocol is used to determine if p is inside P without revealing to each other any information about their exact position. This scheme also achieves to protect information about the boundaries of the data collection area, which may be necessary in some commercial or military networks. For the execution of the protocol, two parties are required: a user agent device that can compute its current position through some external location service, and a sensor network node that either belongs to the service provider or to a trusted third party. The sensor network node can also serve as the privacy policy announcement point if it is concluded that the user is within the sensing area.

In order to provide software architecture support for privacy-sensitive and context-aware ubiquitous computing applications, Confab [3] was proposed as a toolkit that facilitates the development of client-centered architectures, where personal information is sensed, stored, and processed on user-owned devices as much as possible. It both defines mechanisms for end users to control sensitive information disclosure, and abstractions and customizable privacy mechanisms for developers of privacy-aware applications. In order to give to end users flexibility over the privacy tradeoffs they are willing to make, Confab enables applications to publish service descriptions that include various service level options. It facilitates the use of three basic interaction patterns, namely optimistic, pessimistic and mixed-initiative, where data disclosure decisions are made interactively by the users, and offers mechanisms for user control over the access, flow, and retention of personal information.

7. Conclusions

From the schemes that have been proposed, it becomes apparent that there exist solutions to meet most of the set requirements. Privacy issues are addressed at multiple levels in the network stack and at different points of the information flow. Some schemes interfere with the routing protocol, by requiring modifications of the message headers or introducing routing path selection strategies, others interfere with the information flow through introducing intermediaries, while others interfere with the data through or enforcing adjustments in its granularity. Some schemes aim to protect against adversaries overhearing the communications, while others aim to protect against illegitimate access of information from service providers. Different assumptions are made about the entities that are considered trusted, with some schemes trusting only the sensor network deployment and other schemes assuming that there exist trusted intermediaries to enforce pri-

vacy mechanisms. Some schemes address user privacy concerns related to the information captured and transmitted by his carried devices, while others address concerns with respect to environmental sensor networks that capture information about people in their proximity.

However, some issues can not be disregarded; First, most of the schemes presented, especially for protecting the context of the communications, influence the system design at the networking protocol level, which complicates their actual integration to the deployments. Moreover, issues related to anonymising or pseudonymising data depend on the application domain, the in-network data processing schemes and the privacy sensitivity of each user. Thus, it may be infeasible to design a generic and high level privacy architecture, that could both be independent of the underlying networking protocols and guarantee some level of privacy independently of the context of the deployment.

Another issue is related to the level of trust users need to have to the deployments in order to take full advantage of the services that can be offered. None of the schemes discussed can protect against malicious service providers, that do not adhere to their announced privacy policies. The definition by the users of strict privacy policies that would guarantee that personal information is not disclosed, would also not allow them to use legitimate services that require that information. It would thus be necessary to build some level of trust to legitimate deployments, which can not be accomplished using solely technical means. Trusted privacy certification authorities, the appropriate legal deterrence and societal norms are expected to help toward this direction in the future.

References

[1] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Commuications ACM*, vol. 47, no. 6, pp. 53–57, 2004.

[2] D. Cvrcek, M. Kumpost, V. Matyas, and G. Danezis, "A study on the value of location privacy," in *Proceedings of the 5th ACM workshop on Privacy in electronic society (WPES'06)*. 2006, pp. 109–118, ACM Press.

[3] J. I. Hong and J. A. Landay, "An architecture for privacy-sensitive ubiquitous computing," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services (MobiSys'04)*. 2004, pp. 177–189, ACM Press.

[4] H. Chan and A. Perrig, "Security and privacy in sensor networks," *IEEE Computer*, vol. 36, no. 10, pp. 103–105, 2003.

[5] A. Pfitzmann and M. Hansen, "Anonymity, unlinkability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology," February 2006, Version v0.27.

[6] A. Pfitzmann and M. Kohntopp, "Anonymity, unobservability, and pseudonymity - a proposal for terminology," in *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability*, 2000, LNCS 2009.

[7] C. Díaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," in *Proceedings of Privacy Enhancing Technologies Workshop (PET'02)*. 2002, Springer-Verlag, LNCS 2482.

[8] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Privacy Enhancing Technologies (PET'02)*. 2002, Springer-Verlag, LNCS 2482.

[9] A. Pfitzmann, B. Pfitzmann, and M. Waidner, "Isdn-mixes – untraceable communication with very small bandwidth overhead," in *Proceedings of the 7th IFIP International Conference on Information Security (IFIP/Sec'91)*. 1991, pp. 245–258, Elsevier.

[10] J. Hubaux, S. Capkun, and J. Luo, "The security and privacy of smart vehicles," *IEEE Security and Privacy*, vol. 2, no. 3, pp. 49–55, 2004.

[11] S. Gritzalis, "Enhancing web privacy and anonymity in the digital era," *Information Management & Computer Security*, vol. 12, no. 3, pp. 255–287, 2004.

[12] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms.," *Communications ACM*, vol. 24, no. 2, pp. 84–88, 1981.
[13] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous connections and onion routing," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 482–494, 1998.
[14] M. Reiter and A. Rubin, "Crowds: anonymity for web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, 1998.
[15] C. Shields and B.N. Levine, "A protocol for anonymous communication over the internet," in *Proceedings of the 7th ACM Conference on Computer and Communications Security*. 2000, pp. 33–42, ACM Press.
[16] Y. Guan, X. Fu, R. Bettati, and W. Zhao, "A quantitative analysis of anonymous communications," *IEEE Transactions on Reliability*, vol. 53, no. 1, pp. 103–116, 2004.
[17] E. Aivaloglou, S. Gritzalis, and C. Skianis, "Requirements and challenges in the design of privacy-aware sensor networks," in *Proceedings of the 49th IEEE GLOBECOM Conference - World Class Solutions: Networking the Globe Technical Symposium*, 2006.
[18] B. Arazi, I. Elhanany, O. Arazi, and H. Qi, "Revisiting public-key cryptography for wireless sensor networks," *IEEE Computer*, vol. 38, no. 11, pp. 103–105, 2005.
[19] C. Ozturk, Y. Zhang, and W. Trappe, "Source-location privacy in energy-constrained sensor network routing," in *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN'04)*. 2004, pp. 88–93, ACM Press.
[20] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk, "Enhancing source-location privacy in sensor network routing," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*. 2005, pp. 599–608, IEEE Computer Society.
[21] L. Zhang, "A self-adjusting directed random walk approach for enhancing source-location privacy in sensor network routing," in *Proceeding of the 2006 international conference on Communications and mobile computing (IWCMC'06)*. 2006, pp. 33–38, ACM Press.
[22] Y. Xi, L. Schwiebert, and W. Shi, "Preserving source location privacy in monitoring-based wireless sensor networks," in *Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS'06)*, 2006.
[23] L. Huang, K. Matsuura, H. Yamane, and K. Sezaki, "Enhancing wireless location privacy using silent period," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'05)*, 2005, vol. 2, pp. 1187–1192.
[24] F. Wong and F. Stajano, "Location privacy in bluetooth," in *Proceedings of the 2nd European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS'05)*. 2005, LNCS 3813, pp. 176–188, Springer-Verlag.
[25] D. Singelée and B. Preneel, "Location privacy in wireless personal area networks," in *Proceedings of the 5th ACM workshop on Wireless security (WiSe'06)*. 2006, pp. 11–18, ACM Press.
[26] J. Kong and X. Hong, "Anodr: anonymous on demand routing with untraceable routes for mobile ad-hoc networks," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing (MobiHoc'03)*. 2003, pp. 291–302, ACM Press.
[27] M. Spreitzer and M. Theimer, "Providing location information in a ubiquitous computing environment," in *Proceedings of the fourteenth ACM symposium on Operating systems principles (SOSP'93)*. 1993, pp. 270–283, ACM Press.
[28] G. Myles, A. Friday, and N. Davies, "Preserving privacy in environments with location-based applications," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 56–64, 2003.
[29] U. Hengartner and P. Steenkiste, "Access control to people location information," *ACM Trans. Information Systems Security*, vol. 8, no. 4, pp. 424–456, 2005.
[30] E. Snekkenes, "Concepts for personal location privacy policies," in *Proceedings of the 3rd ACM conference on Electronic Commerce (EC'01)*. 2001, pp. 48–57, ACM Press.
[31] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proceedings of the 1st international conference on Mobile systems, applications and services (MobiSys'03)*. 2003, pp. 31–42, ACM Press.
[32] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 571–588, 2002.
[33] A. R. Beresford and F. Stajano, "Location privacy in pervasive computing," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 46–55, 2003.

[34] M. Gruteser and X. Liu, "Protecting privacy in continuous location-tracking applications," *IEEE Security & Privacy*, vol. 2, no. 2, pp. 28–34, 2004.

[35] M. Gruteser, G. Schelle, A. Jain, R. Han, and D. Grunwald, "Privacy-aware location sensor networks.," in *HotOS*, 2003, pp. 163–168.

[36] K. P. Tang, P. Keyani, J. Fogarty, and J. I. Hong, "Putting people in their place: an anonymous and privacy-sensitive approach to collecting sensed data in location-based applications," in *Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI'06)*. 2006, pp. 93–102, ACM Press.

[37] M. Langheinrich, "A privacy awareness system for ubiquitous computing environments.," in *Ubicomp*, 2002, pp. 237–245.

[38] Y. Sang and H. Shen, "A scheme for testing privacy state in pervasive sensor networks.," in *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05)*. 2005, pp. 644–648, IEEE Computer Society.

Wireless Sensor Network Security
J. Lopez and J. Zhou (Eds.)
IOS Press, 2008

Intrusion Detection Techniques in Sensor Networks

Aikaterini Mitrokotsa
Department of Informatics, University of Piraeus
80 Karaoli & Dimitriou Str. Piraeus, 18534, Greece
mitrokat@unipi.gr

A. Karygiannis
NIST
100 Bureau Drive, MS 8930, Gaithersburg,, MD, 20899, USA
karygiannis@nist.gov

1. Introduction

Research has been conducted in wired network Intrusion Detection Systems (IDS) for over 25 years. Although there is ongoing research in wired IDS techniques, it is considered a mature technology. Wireless area networks and personal area networks have been the focus of recent research, as they represent new risks and security challenges. Mobile Ad Hoc Networks (MANETs) have further challenged researchers to develop IDS techniques in an even more challenging environment. The promise of wireless sensor network technology to provide cost-effective monitoring of critical applications ranging from industrial control to border monitoring necessitates new research in the area of wireless sensor network IDS. The unattended nature and the inherent computational and communication limitations of sensor networks make them vulnerable to a broad range of attacks. Given the relative infancy of this new technology, the limited documented cases of actual sensor network attacks, the lack of publicly available network traces of sensor network attacks, most experience in this area is limited to simulations or laboratory emulations, with few approaches having been vetted in the field. This chapter outlines the unique challenges of wireless sensor network IDS and provides a survey of solutions proposed in the research literature.

2. Wireless Sensor Network IDS Challenges

Intrusion detection in wireless sensor networks presents a number of new and significant challenges not faced by wired, IEEE 802.11-based wireless, or even mobile ad hoc networks (MANETs). Although, previous research in IDS in these networks can serve as stepping stones for developing IDS techniques in wireless sensor networks, many of the techniques are not applicable due to the nature of the resource-constrained environment in which they will be deployed. In a typical wired network an adversary can launch an attack to compromise the network security perimeter from any other

interconnected computer in the world. The adversary may use various techniques to conceal their attempts as well as their virtual and physical identity. A typical IEEE 802.11 wireless network in infrastructure mode presents new security challenges to the network administrator because communication between the access point and the clients is broadcast, and unlike the wired network, an eavesdropper does not need to have access to the wired network infrastructure to capture this traffic. On the other hand, the eavesdropper must be physically present and within the wireless transmission range to eavesdrop. This limits the number of potential attackers from anyone in the world with Internet access, to anyone sufficiently motivated that is within the physical transmission range and is willing to assume some risk of being physically identified. The goal of the attacker of the wired network is to compromise the network, to gain unauthorized access to information stored on network devices, and possibly to use the network resources to launch other attacks. The goal of the attacker in the IEEE 802.11 wireless networks is similar to those of the attacker of a wired network or may simply be the theft of service in the form of Internet connectivity. The goal of the attacker in a sensor network may be to either disrupt the operation of the sensor network, to provide false information to the sensor network application by providing incorrect data, or to gain unauthorized access to the sensor network data which is inevitably stored in the base station and forwarded to other computing devices. The base station is an attractive target for attack because it contains almost all the sensor data. Fortunately, the base station is typically more powerful than the sensor nodes themselves and can make use of existing mature security countermeasures that are not yet available to the sensor nodes. Although the universe of potential attackers of a particular sensor network is less than that of potential attackers of a computer on the Internet, an adversary of a sensor network can be assumed to be more motivated and less opportunistic. An attacker on the Internet might use attack scripts to attack any vulnerable network, while a sensor network is more likely to be the focus of a goal-oriented attack. The attacker of the sensor network has risen above the threshold of the remote attack using “kiddie scripts” by taking greater physical risks.

Many scenarios have been proposed in the research literature citing the ease of deployment of sensor networks, many metrology experts, however, would argue that the placement of sensor nodes is highly dependant on the application and that test measurements are essential whenever possible to ensure the quality and usefulness of the data collected. If the sensors can be physically placed by an organization, it stands to reason that the distribution of the keying material can also be restricted to an authorized user that has physical access to the sensor nodes during the deployment stage. The ability to restrict the distribution of keying material only to authorized users can serve as an advantage to those responsible for the security of the sensor network. Physical access to the sensors to distribute keying material makes the sensors less vulnerable to networks attacks, while the unattended deployment of the sensors makes them more vulnerable to physical attacks.

Like all wireless technologies, sensor network communication is susceptible to jamming. Although jamming is difficult to prevent, it is more easily detected and located than a Denial of Service (DoS) attack on the wired Internet. DoS attacks on the Internet may be launched by botnets and carried out by compromised machines running zombie processes in the background unbeknownst to the owner of the machine, thus the risk for physical identification and apprehension of the attacker is reduced.

Jamming a sensor network requires the adversary to place the jamming equipment within the transmission range of the sensor network, thereby exposing the adversary to a greater risk of physical identification and apprehension.

Wireless sensor networks IDS's face different challenges and have more constrained resources with which to counter these threats. The selection of which IDS technique to use depends on a number of factors. These factors include:

- **Network Topology.** Although many sensor networks are described as self-organizing, what this really means is that the sensors will discover the route to the base station on their own. The physical placement of the sensors themselves must ensure a minimum level of connectivity and in many cases sufficient redundancy. The placement of the sensors requires careful study and the selection of the sensor location is not as arbitrary as some of the literature may suggest. If the sensor network topology provides for redundancy of measurements, then it also provides the opportunity for one sensor to validate the measurements reported by another sensor. Just as the base station provides a rich target for an adversary, so do the sensor nodes that are closest to the base station as more data collected from the edge of the network passes through these sensors. We note that a mesh topology is more resilient to individual node failure or compromise. If the sensor network topology is tree-like, individual nodes are likely to be vertex-cuts of the sensor network and their failure or compromise could lead to a disconnected network.

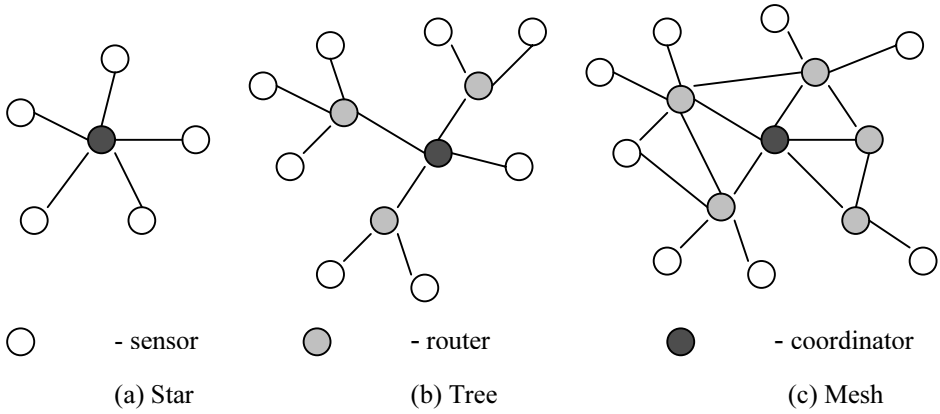


Figure 1. Sensor Network Architectures.

- **Mobile vs. Stationary.** In a sensor network both the sensors and base station may be mobile, both the sensors and base station may be fixed or stationary, or the network may be some combination of the two. In the case of mobile sensors, the sensors may be attached to shipping containers to monitor the condition of the cargo while in transit and report the measurements to different base stations along their journey. These sensors may be queried by an authorized base station. In this case, there may be limited physical controls to restrict access to the sensor and the sensor may be unattended for long periods of time. Base stations may be mobile,

for example, in the case of a utility company a base station or sensor reader may query fixed sensors as the utility company vehicle travels through a densely populated area. In the case of the stationary sensor, monitoring water consumption for example, physical controls may be used to protect the sensor hardware. The base station may be physically secured in the utility company vehicle. In this case the main threat and financial concern would be someone tampering with the integrity of the sensor readings.

- **Open vs. Closed Sensor Networks.** A closed network is one in which participation is limited to nodes under the same administrative domain, while an open network allows any node, sometimes with prior authorization, to join on an ad hoc basis. A closed network allows for more administrative control over each individual node, while an open network must support standards-based protocols and interoperability in order to allow nodes without any prior security associations to join the network. If membership in the closed network requires the possession of a shared key or identity certificates, then the sensor nodes must have cryptographic support and require online security services such as certificate authorities in order to dynamically join the network. These higher level security services may also be targets of attack in order to disrupt the sensor network operation.

- **Physically Accessible or Inaccessible.** An attack on the sensor network communication protocols or simply eavesdropping on sensor network communication requires physical proximity to the sensor network. Although eavesdropping may be confined to sensor communication operating ranges, jamming is a much less selective technique that depends on the power of the jamming source. If an adversary has physical access to the sensors, then one could assume that the adversary can place their own sensor, depending on the application, in the same location. If the sensors are complex or cost-prohibitive to replicate then naturally the adversary would want to target the base station or the sensor communication. Clearly, each sensor application needs to be analyzed in order to quantify the risks and to select the appropriate countermeasures. Biochemical sensors, for example, are typically more complex, costly, and larger than temperature, pressure, and accelerometer sensors. The more costly the sensor the less likely the availability of redundant measurements or the overlapping of monitored areas.

- **Application Domain.** Typical network-based and host-based IDS may make use of log and audit files to detect intrusions. Although these techniques may not always be available to sensor networks, sensor networks can also take advantage of application-level IDS for each particular domain. For example, a faulty sensor that is reporting physical measurements that are inconsistent with other physical phenomena can be detected by analyzing the data at the application level. These applications would be similar to system diagnosis techniques that analyze the collected data to detect and diagnose potentially faulty sensors. The problem of differentiating between a faulty sensor and a malicious sensor is very difficult. A sensor may be experiencing a physical failure and report erroneous data, an intruder may be tampering with the sensor hardware, or the data may be modified on the sensor or while in transit.

- **Critical or Non-Critical Application.** The necessity for the collection of real-time data for critical applications will also have an impact on the types of countermeasures that can be used. An industrial application, for example, would require real-time detection and response to a potentially dangerous situation. A long-term environmental monitoring application may not require an immediate response. The environmental monitoring application may employ sensors that are unattended for long periods of time, while the industrial application may have the sensors confined within the physical perimeter of a factory and thus face a different threat environment. Sensor networks may often be deployed in critical applications such as biochemical agent monitoring in urban areas or transportation systems. In the case of biochemical agent sensor networks, video sensors are often used to help system operators respond to critical events. In the case of a biochemical sensor, the system operators are faced with the decision of sending a team of experts to the location in special suits to protect first responders against hazardous materials and as a result raise the risk of public panic. In a fire monitoring sensor network, visual corroboration can be used to eliminate costly responses to false-positives. Sensor networks can be comprised of nodes of varying degrees of complexity, network connectivity, and cost depending on the application. Moreover, depending on the criticality of the application, sensor networks are likely to be deployed in conjunction with other technologies and human processes to ensure the robustness of the monitoring and control processes.

- **Hazardous or Non-hazardous Environment**. If the sensor network is deployed in a non-hostile environment we must also consider the physical threats faced by the sensor network. These threats can include an adversary tampering with the exposed sensor circuitry, physical attacks to extract the keying material from the sensor node, and placing new or cloned sensors within the network to provide false data. If sensors are deployed in hazardous environments, then it would stand to reason that the sensors themselves are less vulnerable to physical attacks, but they are more likely to be the target of either denial of service or eavesdropping. These environments also make the base station, from which the data is ultimately to be collected, a more attractive target. If sensors are deployed in non-hazardous environments and no physical countermeasures are available to prevent physical tampering with the devices, then the sensors themselves and the sensor network application must be able to detect tampering. Tamper-resistant and tamper-evident technologies can be employed to help detect and diagnose physical attacks against the sensor network.

- **Routing Algorithms.** Support for more sophisticated routing algorithms provides the opportunity to use network traffic analysis to detect malicious activity. Sensor network protocols, however, have been designed with simplicity and power conservation as their main goals. The most common sensor routing technologies are ZigBee, TinyOS, and IEEE 802.15.4. TinyOS is an event based operating environment designed for sensor networks. TinyOS is designed to support two of the most basic protocols used in sensor networks: dissemination and collection. Dissemination reliably delivers small data items to every node in a network, while collection builds a routing tree rooted at a sink node. Together, these two technologies enable a wide range of data collection applications. The Zigbee standards define the network, security and application software layers for wireless

sensor networks. The lowest level of the communications stack defined by Zigbee is the networking level. Zigbee is not a MAC protocol, an operating system, or a hardware platform. The Zigbee standards are not open source and are only available to Zigbee Alliance members. The IEEE 802.15.4 standard defines a MAC and PHY layer for wireless sensor networks (http://www.ieee802.org/15/pub/TG4.html). The IEEE 80.15.4 standard has 16 channels in the 2.4GHz ISM band, 10 channels in the 915MHz I and one channel in the 868MHz band. The IEEE 802.15.5 standard has CSMA-CA channel access, supports data rates of 250 kbps, 40 kbps, and 20 kbps, provides automatic network establishment by the coordinator, and incorporates power management to ensure low power consumption. The Zigbee standards assume an underlying 802.15.4 layer.

- **Cryptographic Support.** Support for cryptographic protocols allows sensor networks to address access control, message integrity, and message confidentiality requirements. Public key cryptography is prohibitively expensive for sensor networks in terms of computation and energy consumption. It must be used sparingly or not at all. Developing security mechanisms using efficient symmetric key cryptography is more promising, but packet overhead is still a significant problem. Symmetric key encryption and authentication mechanisms for conventional networks typically require at least 16 bytes of overhead per packet. This is almost half the current packet length used in sensor networks. TinySec is a link layer encryption mechanism for devices running TinyOS. The core of TinySec is a block cipher and keying mechanism that is coupled with the Berkeley TinyOS radio stack. TinySec uses a single, symmetric key that is shared among a collection of sensor network nodes. Before transmitting a packet, each node first encrypts the data and applies a Message Authentication Code (MAC), a cryptographically strong hash to protect the data integrity. The receiver verifies that the packet was not modified in transit using the MAC and then deciphers the message. TinySec supports three main security requirements: access control, integrity, and confidentiality. TinyECC is an Elliptic Curve Cryptography (ECC) implementation for TinyOS. TinyECC supports all elliptic curve operations, including point addition, point doubling and scalar point multiplication. In addition to the basic elliptic curve operations, TinyECC supports ECDSA operations (signature generation and verification). TinyECC has been tested on both MICAz, TelosB and Imote2. TinKeyMan provides an implementation for pairwise key establishment in wireless sensor networks using the polynomial pool-based key predistribution scheme.

- **Sensor Network as Part of a Larger Network.** The vast amount of information collected by sensor networks would prove far more valuable if it could be shared with authorized users connected to other wired or wireless networks. In open networks, for example, a mobile base station could traverse a city gathering sensor data from various data providers. Each sensor network deployed by each organization will likely be under a different administrative domain, support different security protocols, and employ different access control policies. For example, a first responder could run an application on a wireless handheld device that would collect data from different sensor networks deployed by the Centers for Disease Control and Prevention (CDC), the National Oceanic and Atmospheric Administration (NOAA), and the New York Metropolitan Transportation

Authority (MTA). The handheld device can be part of a MANET, and the application can help the first responder best react to an emergency and help commuters evacuate a certain area, for example, by taking traffic patterns and weather conditions into consideration. Clearly, this example can be extended to support countless applications that need to access sensor data over heterogeneous wired and wireless networks. To enable these types of applications, the consumer of the data must be able to locate the producer of the data, and then subsequently gain authorized access to this data by providing the appropriate credentials. The producers of the data must be able to advertise the services they offer and be able to authenticate the consumers of this data. Some of the sensor data collected by the sensor base station may be very useful, but not confidential, such as the temperature or wind speed in a particular location; other sensor data may be confidential and require certain privileges to access. Thus, the sensor base station must have security policies in place to provide access control to its data. If the sensor network base station can serve as a gateway to a wider network, then the base station must employ security mechanisms that are deployed in conventional wired and wireless networks.

- **Sensor Network Support Infrastructure.** The data collected by the sensors can be targeted while on the sensors themselves, during intra-sensor communication, during sensor-to-base station communication, while on the base station, while the base station is transferring the sensor data back to a data repository over wireless or wired links. A number of other security services may be used to support the sensor network, such as service directories and certificate authorities; all of which could be the target of an attack to disrupt the sensor network operation. Countermeasures for security risks outside the sensor network are well documented and should be used by the base station assuming the base station is acting as a gateway to a broader network.

3. Wireless Sensor Networks Attacks

In order to better understand the attacks an IDS must be able to prevent, counter, detect, and respond to, this section provides a brief overview of sensor network attacks. We note that an attacker may be equipped with either malicious nodes or more sophisticated computing machinery like a laptop or signal generator and signal processing equipment, may be an inside attacker or an outside attacker, or may be a passive or an active attacker. Most trust models assume that the base station is trustworthy as long as it is available. Given the great value of the base station one can argue that it is more likely to be attacked than a sensor especially since it is also more likely to have network connectivity through a wired or wireless gateway.

Sensor networks are susceptible to attacks starting from the physical layer and going all the way up the stack to the application layer. Roosta et. al. [2] provide the following classification of sensor network attacks:

- *Physical Tampering.* Physical attacks can include probing techniques on the sensor circuitry or side channel attacks.

- *Software Attacks*. Software attacks attempt to modify code and exploit software implementation vulnerabilities.

- *Physical Layer Attacks*. Physical layer attacks take advantage of the wireless broadcast medium to simply jam the radio frequency so that no useful communication can occur.

- *Link Layer Attacks*. Link layer attacks can cause excessive collisions in the packet transmission, exhaust the battery's capacity as the result of unnecessary retransmissions, or not adhering to the Carrier Sense Multiple Access (CSMA) protocol and monopolizing the wireless channel.

- *Network Layer and Routing Layer Attacks*. Network and routing layer attacks include: black holes attacks, wormhole attacks, spoofed, altered, and replayed packets, selective forwarding, sinkhole attacks, and acknowledgement spoofing.

- *Transport Layer Attacks*. Transport layer attacks include flooding attacks and desynchronization attacks. The flooding attack aims to exhaust the target's resources by sending an excessive amount of communication requests, while the desynchronization attack alters the sequence numbers of the packets in order to disrupt the communication protocol.

- *Traffic Analysis Attacks*. Traffic analysis attacks allow an adversary to deduce information about the network topology and the location of the base station by monitoring traffic transmission patterns. Once the topology of the network is known, the attacker can selectively target nodes to attack.

- *Key Management Protocol Attacks*. Key management protocol attacks observe the nodes during the discovery process of the shared keys and try to break the cryptographic techniques using more powerful computing devices.

- *Sybil Attack*. Sybil attacks occur when a sensor node or base station assumes multiple identities by changing its MAC and IP address or other identifying information.

4. Wireless Sensor Network Intrusion Detection

Conventional intrusion detection techniques are divided into two main categories: misuse detection and anomaly detection. Misuse detection, or signature-based detection, uses *a priori* knowledge of intrusions and tries to detect attacks by comparing the observed behavior with known attack patterns (signatures). Although misuse detection systems are very accurate in revealing known attacks without many false alarms, their basic disadvantage is that attack mechanisms are continuously evolving, which leads to the need for an up-to-date knowledge base. Thus, misuse detection systems are unable to detect attacks not included in the knowledge base. Misuse detection techniques in resource-constrained environments require additional

research in the efficient storage and updating of attack signatures in order to be a viable solution in sensor networks.

Anomaly detection systems use established normal profiles and attempt to track deviations from normal behavior in order to detect anomalies or possible intrusions. If the normal behavior is accurately characterized then anomaly detection has the advantage of being able to discover previously unknown attacks. Anomaly detection systems, however, are not extensively used in commercial systems due their high false alarm rate. Sensor networks are typically private networks and there may be no incentive to share attack signatures. Unlike public networks like the Internet, there is no precedent for sharing of sensor attack signatures and little is known of any actual sensor network attacks. Most attacks are postulated and the defenses against them are confined to simulations. Organizations deploying sensor networks may be reluctant to share the details of sensor network intrusions for the same privacy and security reasons that most organizations are reluctant to disclose details about wired network intrusions. Furthermore, anomalies are not easily distinguishable from localized, incomplete and possibly outdated information or simple sensor failures and malfunctions.

4.1 Intrusion Detection in Sensor Networks

Intrusion detection techniques developed for wired networks cannot easily be deployed in sensor networks due to the differences between the two types of networks. First of all, sensor networks do not rely on any fixed infrastructure. Thus, compared to wired networks where traffic can be monitored in gateways, routers and switches, sensor networks and wireless ad hoc networks in general, lack traffic management points where real-time traffic monitoring can be performed. In addition, audit data collection is limited by the radio range of the devices. Furthermore, differentiating between malicious network activity and spurious, but typical problems associated with an ad hoc networking environments is a challenging task. In an ad hoc network, malicious nodes may enter and leave the immediate radio transmission range at random intervals or may collude with other malicious nodes to disrupt network activity and avoid detection. Malicious nodes may behave maliciously only intermittently, further complicating their detection. The loss or capture of unattended sensors and personal computing devices may allow for a malicious node to obtain legitimate credentials and launch more serious attacks. A node that sends out false routing information could be a compromised node, or merely a node that has a temporarily stale routing table due to volatile physical conditions. Dynamic topologies make it difficult to obtain a global view of the network and any approximation can become quickly outdated. Traffic monitoring in wired networks is usually performed at switches, routers and gateways, but an ad hoc network does not have these types of network elements where the IDS can collect audit data for the entire network. A wired network under a single administrative domain allows for discovery, repair, response, and forensics of suspicious nodes. A MANET is most likely not under a single administrative domain, making it difficult to perform any kind of centralized management or control. Network traffic can be monitored on a wired network segment, but ad hoc nodes or sensors can only monitor network traffic within their observable radio transmission range [31].

Although, many intrusion detection algorithms have recently been proposed for wireless ad hoc networks, neither intrusion prevention nor intrusion detection solutions for wireless ad hoc networks can be directly applied to wireless sensor networks. We outline some of the important differences between wireless ad hoc networks and sensor networks that seriously affect security requirements.

In a sensor network, every node has an asymmetric broadcast communication pattern. Each node sends data and receives control packets from the base station, which is usually managed by a human. An important advantage of this forwarding structure is its immunity against many elaborate routing attacks [3]. Furthermore, the computing and power resources are even more constrained in sensor nodes than in ad hoc nodes. Thus, resource depletion attacks may be launched more easily in sensor networks. Sensor nodes in most applications are stationary. Thus, the routing overhead is decreased. Moreover, sensor networks are application-oriented, with specific characteristics depending on the target application. Thus, both hardware modules and communication/configuration protocols are highly specialized, making it difficult to define "usual" or "expected" behavior. Additionally, since sensor nodes are subject to more severe resource constraints than ad hoc nodes they are more prone to failure and disappearance from the network [4]. Also sensor nodes may not have global identification (ID) due to the large amount of overhead, the large number of sensors and the lack of Domain Name System (DNS) for sensor nodes unless they support a network stack or have an Internet Protocol (IP) address.

These differences have a direct impact on the way that intrusion detection can be performed in sensor networks. Having an active full-powered agent inside every node [4] is can be very resource-intensive. The memory constraints of sensor nodes makes, for example, make the creation and possible recovery of a detection logfile extremely difficult.

4.1.1 Requirements for Intrusion Detection in Sensor Networks

The design of an IDS for sensor networks should consider the following requirements and constraints [5]:

- An IDS for sensor networks should be based on a distributed architecture applied not only for the data collection, but also for the execution of the intrusion detection algorithm and the alarm correlation. However, we should keep in mind that any collaboration or and trust-building techniques should keep the demands on resources at a minimum..

- Since sensor networks face resource constraints, the IDS system should conserve as much power as possible [6]. Furthermore, considering the fact that the majority of the power consumption comes from the communication interface and not the computation itself, the IDS techniques should not have an inordinate amount of communication overhead.

- Furthermore, considering the resource constraints of sensor devices, the locations where packets are sniffed and analyzed should be minimized while balancing any negative impact on the efficiency of the IDS.

- The lack of centralized points (apart from the base station) in sensor networks that could be used for global collection of audit data, render the localized collection of audit data a necessity.

- Since sensor nodes are highly vulnerable, no node can be assumed to be secure and cooperative algorithms should not consider any node as fully trusted.

- An IDS must be able to safeguard itself against malicious attackers. The possible compromise of a monitoring node should not have a negative impact on the normal operation of legitimate nodes.

- In order to minimize the impact of a possible intrusion in critical applications, it is important that an IDS for sensor networks to function in real time.

5. Sensor Network IDS Architectures

There are two basic sensor network architectures, *flat* and *hierarchical,* that specify how sensors are grouped and how sensor information is routed through the network. In flat architectures all sensor nodes have almost the same communication capabilities and resource constraints and the information is routed sensor by sensor. In hierarchical architectures, sensors are grouped into clusters. One of the member nodes is the "cluster head" and is responsible for management and routing tasks. A challenging research issue is the placement of the IDS modules in sensor networks in order to achieve efficient and effective intrusion detection. A number of placement strategies have been proposed. The following paragraphs describe the most important of those found in the research literature [7], as well as their advantages and disadvantages.

Promiscuous monitoring: A simple strategy would be to place IDS modules in every sensor node as illustrated in Figure 2 and to have each node operate in a promiscuous mode (always listening on the wireless interface). In this way, any malicious packet can be easily detected. However, because of the high overhead associated with this strategy, each participating node's ability to forward network traffic is severely reduced. Furthermore this IDS module placement strategy may lead to network traffic collisions and power consumption.

- **A node monitors only the packets that pass through it.** According to this placement strategy the IDS modules are also placed on every sensor node as illustrated in Figure 2, but only the packets that pass through each sensor node are used for the analysis. Thus, the IDS modules are placed on every sensor along the path from a source to a destination. This approach implies that each packet is analyzed multiple times leading to a waste of computational resources.

- **IDS modules on the base station.** Another possible strategy would be to place the IDS modules on the base station as illustrated in Figure 3. The base station can analyze all the traffic in the sensor network regardless of topology or routing changes and each packet is not processed multiple times. Nevertheless, although a

packet is not processed many times, this placement strategy might overwhelm the base station leading to a large number of packets not being analyzed.

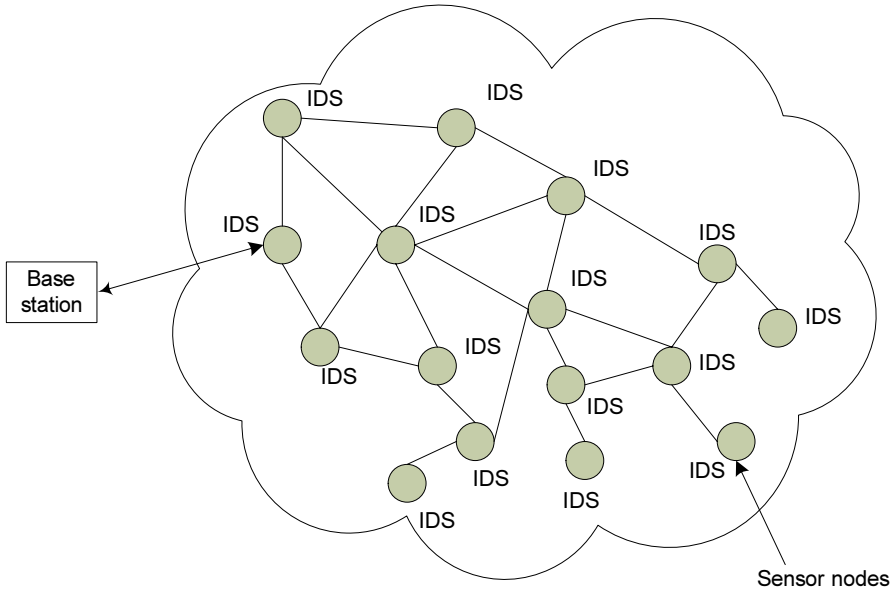


Figure 2. Distributed IDS architecture.

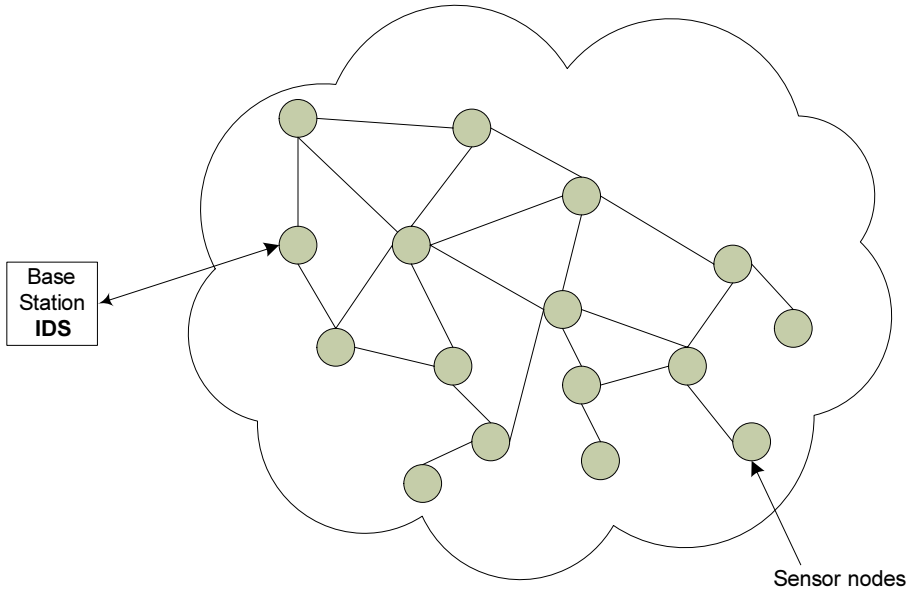


Figure 3. IDS module on the base station.

- **IDS modules on every neighbor of the base station.** In order to reduce the computational load on the base station, IDS modules can be placed on every neighbor of the base station. However, this architecture, cannot combat resource exhaustion attacks since flooding packets will only be dropped when they reach their destination.

- **IDS modules in "cluster heads".** An efficient solution for the placement of IDS modules would position the IDS monitors in such a way that all the packets would be inspected only once, in order to address the resource constraints of the sensor networks. Thus, the IDS modules could be placed in selected sensor nodes (Figure 4) that would be able to cover all the paths from every source node to the base station. In order to achieve this, the sensor network may be divided into clusters with each cluster having a cluster head. This placement strategy implies that every member node of a cluster should forward its data packets to the cluster head which correspondingly forwards them to the base station. However, this approach may lead to a high overhead since the member nodes do not select the shortest path, but instead have to forward their packets through the cluster head. This disadvantage may be limited if the hops between each member node and the cluster head are minimized.

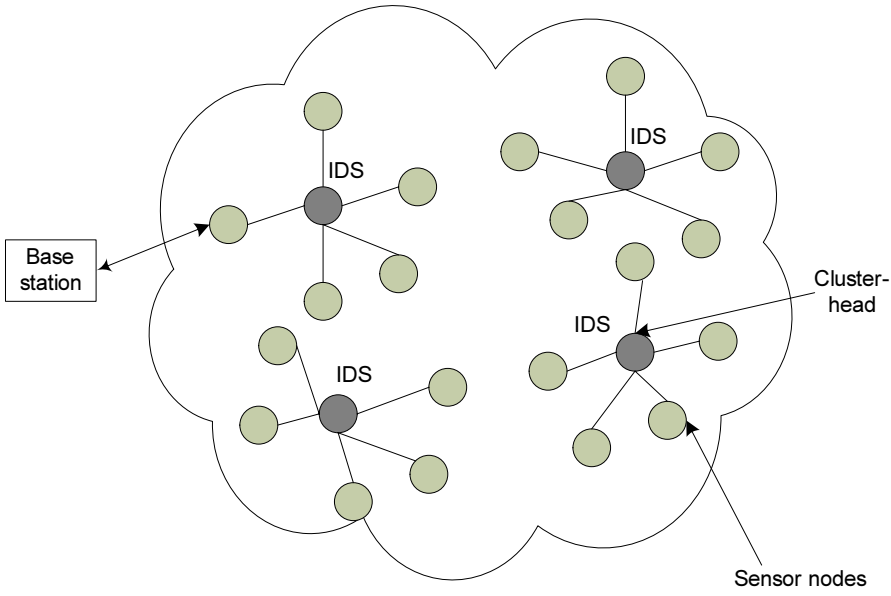


Figure 4. Cluster-based IDS architecture.

The placement of IDS modules is an active research field and many resource optimization approaches have been proposed. Anjum et. al. [7] use a minimum cut set to choose the minimum number of cluster heads. The IDS modules are placed on the nodes that belong to the cut set, but in order to minimize the communication overhead since now the packets do not take the shortest path to the base station, the concept of

the minimum weighted dominating set is used. A distributed implementation to determine the minimum cut set is provided. Anjum et. al. have shown that the proposed algorithms perform well when compared to the random placement of IDS modules on sensor nodes. They also discuss the importance of cooperation amongst the defenders of sensor networks. Furthermore, the proposed approach is evaluated in the case of multiple colluding intruders. We note here that signature-based intrusion detection is assumed, thus; this approach does not address the problem of unknown attacks.

Techateerawat et. al. [8], investigate the accuracy of detecting attacks in sensor networks versus energy efficiency. They investigate new approaches to intrusion detection based on the layout and the selection of monitoring nodes. An analysis is presented based on the response of intrusion detection nodes, the number of required alert messages and the intrusion detection ability. The analysis includes variations in the size of clusters. They are based on a decision mechanism derived from Siraj et al. [9] which combines misuse and anomaly detection. The authors attempt to minimize the number of nodes where the IDS module will be deployed and investigate the following three strategies: core defense, boundary defense, and the distributed defense. According to the core defense the selected nodes are around a central point. In the boundary defense the selected nodes are along a boundary at the perimeter of the cluster. While in the distributed defense the nodes are selected according to a voting algorithm [10]. Their simulation results demonstrate that small clusters may present efficient performance with all defense strategies without significant differences in energy consumption. On the other hand, when the cluster size is large, distributed defense is very energy-intensive, while boundary and core defenses are more economical in energy use, but are vulnerable to insider attacks (from within the cluster).

Roman et. al. [4] propose a general architecture for applying IDS in static sensor networks and propose a *spontaneous watchdog* technique for optimally monitoring neighbor nodes. According to the proposed architecture, IDS agents are located on every node. Every node has an internal database that is used for storing security information collected by the node IDS agents. The IDS agents are divided into two parts, local agents and global agents. *Local agents* are responsible for monitoring the local activities and the packets sent and received by the sensor. Thus, they attempt to discover any attack that may affect the normal behavior of a sensor node by analyzing the local information. This analysis is carried out only when the sensor is active, thus the imposed overhead is low. *Global agents* are responsible for watching the communications of their immediate neighbors and may behave as watchdogs [11]. If all the global agents are activated and listen to their neighbors at the same time, this would be a highly costly operation, thus only a small subset of nodes watch the network communications at any given time.

The way global agents are activated depends on the routing architecture that a sensor network adopts. In hierarchical architectures, global agents are activated in every cluster, since the combination of all clusters covers the entire network. In flat architectures the selection of activated global agents is difficult since it is not possible to know what agents cover the network. Since clustering techniques add complexity and increased overhead for the creation and maintenance of the clusters, an alternative distributed solution, called spontaneous watchdogs approach is proposed. The

spontaneous watchdog technique relies on the broadcast nature of sensor communications and takes advantage of high density sensor deployments. The main goal is to activate only one global agent per packet circulating in the network and is performed by algorithm that checks the destination of every packet.

6. Sensor Network IDS Approaches

IDS approaches proposed for safeguarding sensor networks can be classified into four distinct categories:

- IDS using routing protocols,
- IDS based on neighbor monitoring,
- IDS based on innovative techniques, and
- IDS based on fault tolerance.

6.1. IDS using Routing Protocols

Loo et. al. [12] and Bhuse and Gupta [13] describe two intrusion detection techniques for routing attacks in sensor networks. However, both proposed approaches are based on the assumption that routing protocols for ad hoc networks can also be applied to sensor networks [5]. The AODV (Ad hoc On-Demand Distance Vector) routing protocol is used by Loo et. al. [12], while DSDV and DSR (Dynamic Source Routing) protocols are used by Bhuse and Gupta [13]. Most commercially available sensors, however, do not support ad hoc routing algorithms such as these.

Loo et. al. [12] propose an intrusion detection scheme that uses a clustering algorithm and classifies anything that deviates from normal routing traffic patterns as abnormal traffic patterns. The proposed approach is based on a *fixed-width clustering* algorithm, which is highly effective for anomaly detection in Internet Protocol (IP) networks. It is based on anomaly detection thus; it is able to detect unknown attacks. The intrusion detection approach is based on a set of traffic features that could be used for the detection of a wide range of attacks. The proposed approach is based on distributed Local Intrusion detectors that rely solely on information extracted form each node's routing tables. Thus, no communication between sensor nodes is required, an important advantage considering the scarce power resources of sensor networks. The limitation of this approach, however, is that ad hoc routing algorithms use routing tables, but many sensor nodes simply communicate with a parent or child node without being aware of which route packets may traverse to reach their final destination. The proposed approach is evaluated for three routing attacks: the *Periodic Route Error attack, Active Sinkhole attack, Passive Sinkhole attack.* Also this scheme proved to be very efficient for the detection of sinkhole attack, an attack with severe impact on sensor networks.

Bhuse et. al. [13] propose lightweight methods in order to perform intrusion detection in sensor networks. The proposed methods use existing system information such as neighbor lists, routing tables, sleep and wake up schedules etc., and attempt to detect the malicious behavior at multiple layers. Thus, if an attacker manages to escape detection at one layer, there are many opportunities to detect the malicious activity in

the other layers. The detection methods at each layer are independent of each other and may be used in combination depending on the needs and the available resources.

6.2. IDS based on Neighbor Monitoring

Da Silva et. al. [6], Onat and Miri [3], Krontiris et. al. [5] and Hsin et. al [14] propose intrusion detection approaches that present some similarities to each other. In all these approaches some sensor nodes monitor their neighbors in order to detect possible intrusions. According to the proposed methods, the monitoring nodes select data from messages transmitted in their radio range and select related information including message fields. This data is used as input to the corresponding IDS. This is similar to the proposed watchdog techniques described in the MANET IDS research literature [11].

More specifically, Da Silva et. al. [6] propose a decentralized intrusion detection approach. According to this approach, the monitoring nodes watch their neighbors. If a sensor node changes, delays, replicates or simply keeps and does not forward the received messages as required, then the monitoring node records it as a failure and an indication of a possible intrusion. Thus, a monitoring node creates a history for each of its neighbors. If the number of failures detected by the monitoring node is greater than an expected value then an attack level is raised. In order to avoid a possible high false positive rate, a learning stage is introduced, during which a monitoring node does not consider any abnormal event until the average number of failures has settled. The evaluation of the proposed approach has been performed for three types of occasional network failures and eight types of intruder attacks. The occasional network failures are data alteration, message loss and message collision. The intruder attacks are message delay, repetition and wormhole, jamming, data alteration, message negligence, blackhole and selective forwarding.

The proposed intrusion detection approach by Onat et. al. [3] is based on the average packet arrival rate and the average received power level as representative features for each neighbor. Each node uses only the last n packets received from each neighbor and these packets are used for the calculation of statistics for each neighbor node. However, in both these proposed approaches [6], [3] the monitoring nodes do not collaborate and the buffer size plays a substantial role in the efficiency of the detection and the low false alarm rates [5].

Krontiris et. al. [5] propose a lightweight distributed intrusion detection approach for sensor networks based on specification-based detection,. According to the proposed approach, each sensor node hosts an IDS agent. The IDS agent performs network monitoring, decision making and response. During the network monitoring task, every sensor node monitors each immediate neighbor and collects audit data. During the decision-making task every node based on local audit data determines the existence of possible intrusions and forwards their conclusions to each immediate neighbor in order to make the final collective decision. The local detection engine applies the defined specifications about what is normal behavior and monitors audit data according to these constraints. The cooperation between neighboring nodes is performed by applying the majority vote rule in order to determine the existence of an attack. Furthermore, when an attack is detected the local response module is activated. Depending on the severity

of the attack the response might be direct or indirect. The direct response excludes the suspect node from the routing paths and forces regeneration of cryptographic keys for the rest of the neighbors. The indirect response notifies the base station about the suspected behavior of the possible intruder and reduces the reputation of the link to that node so that gradually it will be characterized as unreliable. The proposed approach is evaluated against the blackhole attack and selective forwarding attacks. This approach is a bit antithetical to many sensor designs that try to keep the wireless interface off unless the sensor node must report data. We note that the wireless interface consumes much more power than the computing processor.

Hsin et. al. [14] propose a low overhead network health monitoring algorithm. This approach is relevant in the context of an IDS since the IDS must always be able to differentiate between malicious behavior and normal operational failures. Making this distinction in sensor networks is even more important because of the physical exposure of the sensors to environmental elements and physical attacks. They propose a distributed monitoring mechanism for wireless sensor networks which focuses on localized decision making and the minimization of false alarms by testing whether a group of sensor nodes are dead or alive. The proposed schema can be seen as a passive monitoring mechanism which notifies the control center only if it is highly confident that something is wrong. The basic principle of the proposed approach is based on neighbor monitoring. Each sensor sends its update messages only to its neighbors and every sensor monitors its neighbors. The monitoring is controlled by a timer associated with each neighbor. Thus, if a sensor has no communication with a neighbor for a pre-defined period of time, then the neighbor is assumed to be dead. By mutual monitoring between neighbors, the monitoring performed throughout the network and the control center has to monitor only a small subset of nodes. Furthermore, local decision making is performed since each node makes some decisions before communicating with the control center. Each node increases the confidence of its decisions by consulting each neighbor. The total amount of traffic sent to the control center is minimized by adopting a simple query-rejection or query confirmation procedure and minimal neighbor coordination. We should note here that the neighbor monitoring works only if every node is reachable from the control center.

C.C Su et. al. [15] propose an authentication-based intrusion detection and response approach for wireless sensor networks. Using an intrusion prevention mechanism they authenticate exchange control messages between sensor nodes. According to their intrusion detection approach different monitoring mechanisms are needed to monitor cluster heads and member nodes relative to their importance. Network members perform the monitoring of cluster-heads in turns. Thus, the monitoring time is reduced and member nodes conserve energy. The monitoring of member nodes is performed by cluster heads that have the authority to detect any misbehaving node and isolate the malicious node by broadcasting an encrypted alarm message. Energy conservation is achieved, since cluster-heads are used for monitoring instead of using all member nodes to monitor each other. Furthermore, the intrusion response of the proposed approach is activated only when the detection of alarm messages comes from at least X monitoring nodes that detect the misbehavior of the same cluster head. Thus, the proposed intrusion detection approach has the advantage of being tolerant of compromised nodes within a threshold in a cluster. However, a cluster head could isolate too many member nodes and this way to reduce its own

security. In order to avoid such an event, the member nodes that monitor the cluster head should determine whether or not the cluster head should be changed depending on the number of monitoring nodes. Nevertheless, the proposed intrusion prevention mechanism does not allow new nodes to join the network and assumes that sensor nodes do not move.

6.3. IDS based on Innovative Techniques

Agah et. al. ([16], [17]) suggest that game theory can be applied to intrusion detection in sensor networks. They propose an approach for the prevention of DoS attacks in sensor networks based on a game theory approach. The prevention approach is formulated as a repeated game between an intrusion detector and the nodes of a sensor network. They propose a protocol based on game theory which is able to recognize the nodes that agree to forward packets, but fail to do so. The approach categorizes different nodes based upon their dynamically changing behavior and enforces cooperation among nodes. Any non-cooperative behavior is punished. The intrusion detector situated at the base station monitors the collaboration of other nodes and builds up a history that represents their reputation. Sensor nodes that contribute to common network operation increase their reputation. The reputation is used as a metric of trustworthiness and is used to statistically predict the future behavior of sensor nodes. The advantage of the proposed approach is that through the history created by the base station for each sensor node, and the negative reputation the base station assigns to any malicious behavior, it is possible to create routing paths consisting of less malicious nodes for more secure transmissions. Thus the malicious nodes are isolated. The main disadvantage of the proposed approach is when the number of malicious nodes in the sensor network increases, the success rate of the IDS decreases. This can be explained if we consider the fact that the IDS attempts to lower false positive and false negative rates and as a result the detection rate is decreased since it misses more malicious nodes. This technique may not be suitable for certain environmental monitoring applications, but may be considered in more critical applications in which an intrusion is likely and cannot be tolerated.

Doumit et. al. [18] propose a light-weight intrusion detection technique for wireless sensor nodes based on naturally occurring events and the analysis of fluctuations in sensor readings. Based on the Self-Organized Criticality (SOC) of the deployment region they acquire some knowledge, on which they deploy hidden Markov models (HMM). According to the proposed approach, the sensor network adapts to characterize the norm of the sensor network dynamics in its natural surroundings so that any unusual activities can be detected. Furthermore, the proposed approach is scalable to the addition of new sensor nodes. In order to avoid the transfer of a great amount of data to new sensor nodes and consequently consume scarce time and resources.

6.4. IDS for Specific Intrusions and Operations

Some other approaches propose intrusion detection techniques for specific intrusions and operations ([19], [20], [21], [22], [23]). Ngai et. al. [20] focus on Sinkhole attacks and propose a light-weight intrusion detection algorithm. The

algorithm consists of two main steps. The first step is to create a list of the suspected malicious nodes by checking the data consistency. The identification of the intruder through the analysis of the network flow of information is performed during the second step. The algorithm focuses on the many-to-one communication mode and the solution explores the asymmetric property between the sensor nodes and the base station while effectively using the relatively high computation and communication power of the base station. They also consider the scenario in which a collection of cooperative malicious nodes attempt to hide the identity of the real intruder. They examine the suspicious nodes and attempt to identify the real attacker using majority voting.

Du et. al. [21] propose a general scheme for detecting localization anomalies caused by adversaries. They approach the problem as an anomaly intrusion detection problem, and they propose a number of ways to detect normalization anomalies. The proposed scheme exploits the topology deployment knowledge that is available in many network sensor applications and the group membership of each node's neighbors. It uses this knowledge in order to determine if the estimated location agrees with its observations. If there is an inconsistency above a certain threshold then it is assumed to be an indicator of malicious behavior. The inconsistency is formulated as an anomaly and the problem is studied as an anomaly detection problem. The simulation results demonstrate that the proposed framework is able to detect localization anomalies effectively even when a significant portion of a node's neighbors is compromised. Furthermore, the authors claim that the proposed scheme prevents the cause of an undetected large localization error since the more severe an attack is, the higher the detection time is and the lower the false positive rate is.

Wood et. al. [22] propose an approach for detecting and mapping jammed regions. They describe a mapping protocol whose goal is to surround a jammer. The jamming detection module monitors the radio and the medium access control layers. The protocol applies heuristics to determine if a node is jammed. After deciding that a node may possibly be jammed, is sends a message to its neighbors and notifies the application layer. The application layer may apply power management techniques in order to outlast the jamming. The mapping is initiated by the neighbors of the jammed node that have received jamming notifications. Each receiver of the notification message forms a group by adding nearby jammed nodes as jammed members. Even if the proposed approach is not able to prevent jamming, mapping the jammed area contributes substantially in the mitigation of its impact. Performance evaluation of the proposed approach demonstrates that regions can be mapped in 1-5 seconds when the network is at least moderately connected. Furthermore, the jamming detection and the mapping protocol may be viewed as a cheaper strategy since they use existing data and facilities in the typical sensor communication stack. We note that jamming attacks are often not lengthy in time and are launched at critical times to disable the network or conceal an adversary's activities, or misdirect an intrusion detection system. The authors assume partial jamming of the sensor network and that not all the jamming modules are themselves jammed.

Ye et. al. [23] propose a Statistical n-route filtering (SEF) of injected false data in sensor networks. In large-scale sensor networks individual sensors may be compromised. Compromised nodes might be exploited in order to inject erroneous sensing reports. If the compromised node is not detected then the erroneous reports

may be forwarded to the base station causing not only false alarms, but also depletion of power resources. SEF can be used in order to detect and drop the erroneous reports.

7. Summary

Wireless sensor networks present a number of security challenges not faced by traditional wired or wireless networks. As a result, more research and practical experience is needed to develop effective wireless sensor network IDS. The design of an IDS for wireless sensor networks must take into consideration a number of factors. These factors include: the sensor communication protocol, the network topology, physical accessibility, application criticality, node redundancy, node mobility, computational resources, network membership requirements, base station network connectivity, and cryptographic support. This chapter maps the sensor network IDS landscape and provides a snapshot of the present state of research in this field. This chapter classifies wireless sensor network IDS in four distinct categories: IDS using routing protocols, IDS based on neighbor monitoring, IDS based on innovative techniques and IDS based on fault tolerance. A survey of research in this area reveals that most proposed solutions and results are theoretical or based on simulations, and that real-world experience in preventing, detecting, or responding to sensor network attacks has yet to be published. Since wireless sensor networks are most likely to be initially deployed as private networks, publicly available network traces, attack tools, and attack forensic information for researchers to study will be very limited. This makes it difficult to objectively evaluate and compare the performance of various sensor network IDS technologies. Publishing datasets of actual sensor network traffic and simulated or real intrusions would help researchers advance the state of the art and work from a common baseline in order to compare the effectiveness of their approaches.

References

[1] J. Hill, M. Horton, R. Kling and L. Krishnamurthy, The platforms enabling wireless sensor networks, *Communications of the ACM, Special Issue Wireless Sensor Networks* ,47, Issue 6, (June 2004), 41-46.

[2] T. Roosta, S. Shieh, and S. Sastry, A Taxonomy of Security Attacks in Sensor Networks and Countermeasures, *In Proceedings of the 1st IEEE International Conference on System Integration and Reliability Improvements*, Hanoi, Vietnam, 13-15 December 2006.

[3] I.Onat and A. Miri, An intrusion detection system for wireless sensor networks, In Proceeding of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, vol. 3, Montreal, Canada, August 2005, 253–259.

[4] R. Roman, J. Zhou, J. Lopez, Applying Intrusion Detection Systems to Wireless Sensor Networks, *In Proceedings of 3rd IEEE Consumer Communications and Networking Conference, 2006*, (CCNC 2006), 1, 8-10 Jan 2006, Las Vegas, Nevada, USA, 640-644.

[5] I. Krontiris, T. Dimitriou, F. C. Freiling, Towards Intrusion Detection in Wireless Sensor Networks, t*o appear in Proceedings of the 13th European Wireless Conference*, 1-4 April 2007.

[6] A. P. R. da Silva, M.H.T. Martins, B. P.S. Rocha, A. A.F. Loureiro, L. B. Ruiz, H. C. Wong, Decentralized Intrusion Detection in Wireless Sensor Networks, *In Proceedings of the 1st ACM International Workshop on Quality of service & security in wireless and mobile networks, (2005)*, Montreal, Quebec, Canada, 16 – 23.

[7] F. Anjum, D. Subhadrabandhu, S. Sarkar and R. Shetty, On Optimal Placement of Intrusion Detection Modules in Sensor Networks, *In Proceedings of the 1st International Conference on Broadband Networks (BROADNETS'04)*, 25-29 October 2004, San Jose, California, USA, 690-699.

[8] T. Techateerawat, A. Jennings, Energy Efficiency of Intrusion Detection Systems in Wireless Sensor Networks, *In Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2006 Workshops)(WI-IATW'06)*, Dec. 2006, Hong Kong, 227-230.

[9] A. Siraj, S. Bridges and R. Vaughn. "Fuzzy Cognitive Maps for Decision Support in an Intelligent Intrusion Detection System", IFSA World Congress and 20th NAFIPS International Conference 2001, 4, (2001), 2165-2170,.

[10] O. Kachirski and R. Guha, Intrusion Detection Using Mobile Agents in Wireless Ad Hoc Networks, *In Proceeding of the IEEE Workshop on Knowledge Media Networking*, (2002), 153-158.

[11] S. Marti, T. Giuli, K. Lai, and M. Baker, Mitigating Routing Misbehavior in Mobile Ad Hoc Networks, *In Proceedings of the 6th ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'00)*, August 2000.

[12] C. E. Loo, M. Y. Ng, C. Leckie, and M. Palaniswami, Intrusion detection for sensor networks, *International Journal of Distributed Sensor Networks*, 2005.

[13] V. Bhuse, A. Gupta, Anomaly intrusion detection in wireless sensor networks Source, *Journal of High Speed Networks*, 15, Issue 1 (January 2006), 33 – 51.

[14] C.F. Hsin, M. Liu, A Distributed Monitoring Mechanism for Wireless Sensor Networks, *International Conference on Mobile Computing and Networking, In Proceedings of the 3rd ACM workshop on Wireless Security 2006*, Atlanta, GA, USA, 57 – 66.

[15] C.C. Su, K.M. Chang, Y.H. Kuo, The New Intrusion Prevention and Detection Approaches for Clustering-based Sensor Networks, *In Proceedings of IEEE Wireless Communications and Networking Conference 2005 (WCNC 2005)*, 13-17 March, 2005, New Orleans, LA USA, 4, 1927-1932.

[16] A. Agah, S. K. Das, K. Basu, and M. Asadi, Intrusion detection in sensor networks: A non-cooperative game approach, In Proceedings - Third IEEE International Symposium on Network Computing and Applications, NCA 2004, Aug 30-Sep 1 2004, 343–346.

[17] A. Agah, S. K. Das, Preventing DoS Attacks in Wireless Sensor Networks: A Repeated Game Theory Approach, *International Journal of Network Security*, Sept. 2007, 5, No. 2, 145-153.

[18] S. S. Doumit and D. P. Agrawal, Self-organized criticality and stochastic learning based intrusion detection system for wireless sensor networks, in Proceedings of IEEE Military Communications Conference 2003 (MILCOM 2003), Oct 13-16 2003, 1 609–614

[19] O. Dousse, C. Tavoularis, P. Thiran, Delay of Intrusion Detection in Wireless Sensor Networks, *In Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, (MobiHoc'06), May 22–25, 2006, Florence, Italy, 155-165.

[20] E. C. H. Ngai, J. Liu, M.R. Lyu, On the Intruder Detection for Sinkole Attack in Wireless Sensor Networks, *In Proceedings of IEEE International Conference on Communications 2006 (ICC 2006)*, 8, June 2006, Instabul, Turkey, 3383-3389.

[21] W. Du, L. Fang and P. Ning, LAD: Localization Anomaly Detection for Wireless Sensor Networks, *In Proceedings of the 19th International Parallel and Distributed Processing Symposium (IPDPS'05)*, April 4-8, 2005, Denver, Colorado, USA.

[22] A. D. Wood, J. A. Standovic, and S. H. Son, JAM: A Jammed-Area Mapping Service for Sensor Networks, *In Proceedings of the 24th IEEE Real-Time Systems Symposium (RTSS)*, Dec 2003, 286-297.

[23] F. Ye, H. Luo, S. Lu, and L. Zhang, Statistical En-Route Filtering of Injected False Data in Sensor Networks, *In Proceedings of the 23rd IEEE INFOCOM*, Mar 2004, 2446-2457.

Wireless Sensor Network Security
J. Lopez and J. Zhou (Eds.)
IOS Press, 2008

Remote Attestation – Identification

Alec Yasinsac
Florida State University

Abstract. Many Wireless Sensor Networks are composed of nodes that are virtually identical, i.e. they are clones of one another. The classic "identity" definition cannot properly capture this notion. Even if each node contains an artificial serial number, there is no unique property...no DNA to distinguish devices. Still, all sensor networks must capture some identity notions; code integrity being one such notion. This chapter investigates sensor network node identity relative to the functional properties reflected in executing code found on small computing sensors. We show how wireless sensor networks can establish an important identity property of ensuring code integrity through the process of remote attestation.

Keywords: Authentication, Tamperproof Software, Obfuscation, Hardware Protection

1. Introduction.

One of things that we take for granted in our day to day lives is our ability to refer to our colleagues, friends, and relatives by their names with assumed confidence that we have correctly identified them. Of course, knowing who someone is does not mean that we know everything about them, but for the most part, knowing their name is good enough. We recognize their faces and voice, we understand their role in business, neighborhood, or family. We understand enough about their potential impact on our lives without going to the effort and expense of conducting a detailed background analysis to capture all properties of people's identity.

1.1. How Much Identity is Enough?

We seek a similar "identity balance" in sensor networks. Specifically we ask: "How much do we need to know about sensor network nodes before we accept their reported data as valid?" We consider the entire scale from the weakest filter of accepting any data that meets time, functionality, and format ranges to the strong member identification requirement of remote attestation.

At the weak end of the spectrum, it is difficult to think of communication that does not rely on some type of identification. Closed environments, where communicating members are authenticated out of band, provide one example. Board room decision systems, where each member anonymously [within the board membership] enters comments illustrate such an environment. Essentially, all communications that can reach the message distribution center are considered to be authentic. Similarly, sensors created for an environment where no other communicating entities are possible, or

where interference by non-member parties is improbable or low impact by function, may not require any identity assurance.

Conversely, most communication requires authentication. The ability to establish one's identity is a fundamentally difficult problem and much has been written about it, capped by Gollman's seminal work on entity authentication [1]. There, Gollman summarizes identity in a communications environment with the foundational statement: "A claim about the identity of an entity thus becomes a claim about the identity of the originator of a message." His primary point is that it is common in communications systems that most messages do not originate from neighbors. Rather, messages are routinely cooperatively relayed from network node to node, whether targeted for a specific destination node or to effect a message broadcast. In those circumstances, it may be important to know who originated, rather than who delivered, a received message.

1.2. The Problem of Identifying Wireless Peers.

The identity problem is exacerbated in wireless networks, where research has yet to establish even effective peer identification technology. The invisible node attack [2] and the less threatening wormhole attack [3] leverage the fact that it is very difficult or impossible for a receiving node to determine if a received message was originated by a neighbor, or was merely relayed without change by that neighbor.

Several proposed solutions satisfy partial peer identity requirements. For example, Beth and Desmedt [4] provide the scientific foundation to detect relayed messages by applying timing constraints. Some implementations have emerged, but these are cumbersome and some question whether clock precision and non-propagation latency limitations render this method useless in practice. Others have introduced collaborative, location-based identity methods [5] that rely on special purpose equipment and demand significant communication overhead. Finally, some attempt to utilize physical layer signaling properties for identification [6]. At best, present attempts to identify peers provide non-conclusive peer identity evidence. Table 1 summarizes wireless node identification approaches.

Cryptography	**Geography-Based**		**Physical Layer**	**Introduction**
Signatures	Message Timing	Location	Signal Properties	Hash Chains
Group keys	Collaboration	Distance based	Frequency Hopping	Zero Knowledge

Table 1. Wireless Node Identity Approaches

Fortunately, in many communications environments, peer identification is not important. If the only application requirements are to authenticate the message originator and to verify message integrity, then the identity of the entity that delivered the message may not matter.

1.3. Providing Strong Identification Confidence

When applications demand strong identity guarantees, developers rely upon cryptography to establish the necessary trust. Gollman [1] notes that in instances that demand strong identity assurances, it is natural to equate 'identity' with 'cryptographic key holder'. While properly engineered and applied cryptographic practices can provide strong identity guarantees, their improper use can lead to false confidence. Careless key handling yields cryptography useless and even under a watchful eye they can be lost, stolen, or manipulated.

Additionally, authentication protocols (also known as "security protocols" and "cryptographic protocols") used to generate and distribute cryptographic keys, in general are notoriously prone to subtle errors. In fact, a research discipline emerged from early security protocol standards, albeit de facto standards, introduced by Needham and Schroeder [7] and first exploited by Denning and Sacco [8] some three years later. Thirty years of security protocol research has shown dramatic improvements, but absolute, or even measurable, confidence through cryptographic protocols remains illusive.

A constraint in security protocol analysis is that application demands vary so greatly. For example, the idea of "identity" reflects many varied properties, even when limited to a narrowly defined sensor network paradigm. In some environments, it is essential to be able to connect a data item to a specific sensor, e.g. by serial number. Other environments can suffice by only connecting a data item to any member of group [9, 10] or to a prior interaction [11].

The Sybil Attack [12] reflects another identity paradigm, where one entity announces itself under two distinct identities. Yet other situations demand only connection between a series of data submissions. Wulf et al. proposed an authentication mechanism that connects a user to a previous identity announcement [11]. There are many other identity nuances that complicate protocol verification. In the next section, we examine identity properties specific to wireless sensor networks.

2. Identity Requirements in Wireless Sensor Networks

While wireless sensor networks share many important properties, they can also diverge significantly in other ways. That extends to identity notions and requirements. The following sections illustrate some commonly recognized identity concepts that are important to wireless sensor networks.

2.1. The Fundamentals: Identity and Integrity

In communication systems, for a message to have "integrity" means that the message is unmodified from its originally constructed form, that is, it is authentic, pristine in some sense. Inherent in this notion is that there is an entity that holds control or jurisdiction over each message's content [and intent], thus "integrity" is inherently identity related. The message originator naturally controls message content, thus integrity assessment requires originator identification.

An additional notion inherent in message integrity is that the originator's message may not be modified while in transit. Fortunately, cryptographic methods can provide both message-originator binding and modification detection in a single, efficient

operation. A common approach to linking a message to its originator is by using a Message Authentication Code (MAC), as in [13]. MAC computation is efficient for reasonably short keys and provides integrity and authenticity.

There are two identity attacks that form the foundation for most identity attacks in wireless networks: (1) The Sybil Attack [14] and the Invisible Node Attack [15]. The Sybil Attack occurs when one node participates in a single or, several different conversations using more than one identity. Depending on the application environment, the Sybil Attack can be devastating if not detected or prevented outright. Unfortunately, MAC protocols alone cannot prevent the Sybil Attack.

The Invisible Node attack is also a difficult attack to overcome. It is accomplished by an outsider or insider that simply relays messages without accomplishing actions or inserting information dictated by the governing communications protocol.

Each of these attacks illustrate important, but often overlooked identity properties. The Sybil attack leverages society's comfort with representing a single identity with different tags and that we focus on being confident that a particular tag uniquely selects the proper individual. This vital identity property remains intact during a Sybil attack; whether you know a Sybil node as Alice or Bob, calling either reference selects the correct entity. The Sybil threat is an orthogonal identity concept, essentially that an entity can partition their actions across multiple identities.

2.2. Identity Through Distinctive Device Characteristics

While much recent research focuses on cryptographic authentication techniques and key exchange paradigms, Kranakis et al. [6] introduce a physical layer approach to wireless peer identification. They leverage the natural entropy inherent in signaling devices and show that they can correlate all message traffic by its signal properties or its "signal signature". Essentially, once they detect and catalog a signal from a new peer, they can correlate future messages to the initial transmission, allowing them to establish a message history for each peer and to prevent Sybil attacks under the scheme's assumptions.

Similarly, Kohno et al. introduced Remote Physical Device Fingerprinting [16] in 2005. They showed how a passive observer can establish a unique identity for any node that regularly communicates through a common configuration of TCP connections. Theirs is a powerful technique that can identify nodes across the globe as easily as they can identify machines in the next room. There approach is hardware and operating system independent, thus is applicable to virtually any environment that utilizes TCP connectivity, and can apply to ICMP and some other communication protocols.

The RPDF fingerprint is more accurately termed a "TCP Clock-print". Like any other mechanical or electrical mechanism, clocks are imperfect devices. In this case, the clock's "consistent imprecision" provides a distinctive, identifying property for the clock's host. Based on captured TCP session timestamps, for example in a TCPdump or other TCP session trace, RPDF calculations identify the clock skew for the transmitting device. This clock-skew identity can correlate transmissions to the original transmitter.

Each of these device identity approaches provide an interesting capability that leverages natural properties based in physical science, and they are reasonably reliable in non-malicious environments. However, if an adversary is aware that these mechanisms are in place, there are many ways to modify signal properties to counteract

the natural entropy that allows identification. Similarly, RPDF can be thwarted by altering or omitting TCP timestamps in outgoing messages.

2.3. Routing in Wireless Sensor Networks

The ad hoc nature of wireless sensor networks implies an infrastructure-free environment. Thus, the sensors must organize themselves into a functional network that allows the necessary message passing. The canonical communications paradigm in wireless sensor networks is through message relay, often referred to as routing.

The purpose of routing algorithms is to maximize messaging efficiency by minimizing the number of transmissions necessary to guarantee message delivery. Routing canonically occurs in two phases[1]: (1) Route discovery and (2) Message delivery. Efficiency-aggressive routing algorithms only generate path information when it is needed, thus are dubbed "On Demand" routing protocols. On Demand protocols achieve high efficiency at the expense of some message latency.

At the other end of the spectrum, "Table-Driven Routing" algorithms conduct constant routing updates to ensure the each node has adequate routing information whenever they need to transmit a message.

While all sensor networks are self organizing, some are mobile while others do not move once they are in place. In the static model, static sensor networks need only establish the communication infrastructure once, thus are well-suited to the table-driven approach. Routing paths are created as part of the initialization process and stored for subsequent use in table-driven routing protocols.

Conversely, in the mobile model, the sensors face a different communications challenge. At any time the communications structure can change due to normal movements. Depending on the network density and upon mobility factors, it may not be possible to store a useful route at any node. This situation demands On Demand routing protocols where routes are discovered and used when they are requested.

There are many excellent hybrid routing protocols that optimize various other communication properties. For example, in static wireless sensor networks, nodes may only be interested in establishing one route, say to a local coordinating node. In this case, generating complete routing tables at the outset is unnecessary and wasteful. A hybrid protocol utilizes available information before requesting route discovery. When a discovery request is received, the table is checked before retransmitting the request and if the route is completed locally, no further route request in necessary.

All of the above generally refers to environments where all nodes cooperate and identity other than for the destination node, is unnecessary. While this is the case in many envisioned wireless sensor environments, if the routing protocol must mitigate threats posed by malicious outsiders or insiders, the problem becomes much more sticky. Where security threats apply, corresponding communications protocols must know, and leverage, identity of each node in every active path.

We identified two important identity threats to wireless networks in Section 2.2 above: The Sybil Attack and the Invisible Node Attack. It is easy to see how these attacks can damage wireless sensor network operation by manipulating the routing process. For example, most route discovery protocols gather the desired path by passing a route request on a forward traversal where each node in the path adds their

[1] We recognize that some messaging paradigms deliver the message in one pass. We contend that such message passing protocols do not constitute "routing".

identity to the path. The discovered path's description is returned to the requesting node by transmitting a route reply across the reverse of the discovered path. An "Invisible Node" can trick the requester into accepting an invalid route by simply not attaching their identifying information on the forward route and then retransmitting the route reply as though they were a member of the path.

Similarly, every node in a network where a successful Sybil attack is in effect is subject to false path entries. As a simple example, if node B connects nodes A and D, and if node B can assume identity as both B and C, then both A and B may believe that there are two valid paths between them when in fact there is only one path.

In the simple attack versions given here, these attacks may not appear to be of concern. Unfortunately, in large networks where communication capabilities are pivotal, bandwidth is a scarce resource, and where the stakes are high, these attacks can cause immeasurable damage. Strong identity mechanisms could resolve both of the Sybil and Invisible Node attacks.

2.4. Key Management in Sensor Networks

Generally, sensor network requirements are well-suited to symmetric, group key cryptographic methods. If individual node identity is unnecessary, a pre-assigned key can allow nodes to efficiently encrypt messages to ensure their privacy among group members, authenticity as coming from a group member, and their integrity.

However, even simple additional requirements can invalidate this approach. If nodes are subject to capture in a way that an adversary can learn the group key of if the group key could be compromised in some other way, then each privacy, authenticity, and integrity are compromised.

There is significant research on mechanisms to refresh or redistribute secret group keys [9]. There is also some work on finding efficient mechanisms that may apply to sensor networks. Much of this work focuses on establishing peer-wise secret keys [17, 18]. though there are also efforts to establish efficient group key redistribution methods [19].

The central dividing factor in most group key establishment protocols is whether or not nodes are powerful enough to efficiently compute modular arithmetic with large modulus and large exponents and if they are, does the impact on their power of these computations limit their application?

There are three primary approaches for secure group key distribution: Station-to-station Diffie-Hellman [20] hybrid computation, hash chains, and probabilistic key sharing. In the station-to-station method, nodes self-organize by establishing a coordinator node and an ordering that reflects a valid communication path in the network. The coordinator triggers a station to station computation that results in a participatory group key, where every member can ensure that the resulting key is random and that their key share is incorporated. In one simple variation, the coordinator encrypts the resulting group key with each member's public key and distributes it along the existing path.

Of course this method requires at least two large modulo arithmetic computations per node and may demand the order of n such computations by the coordinator each time a key is distributed. For the near future, many wireless sensor networks are not expected to have sufficient computational or battery capabilities to implement key distribution systems based on Diffie-Hellman.

Cryptographic hashes are less computationally intensive than public key operations, and they demand less power, thus are more well-suited to many wireless sensor network environments. One general approach to hash-based group key computations relies on the one-way properties of the cryptographic hash and a hierarchical node organization.

In addition to reduced resource demands, hash chain-based group key mechanisms may also offer significant opportunity for off-line computation, allowing sensor nodes to be preloaded with hash values and further preserving battery power at the tradeoff of additional onboard memory.

More recently, Gligor, et al. [17] recently introduced a novel key distribution approach wireless sensor networks based on probabilistic key sharing. Their scheme is based on pre-computed random number shares stored on each host. These shares are selectively redundant in a way that ensures that members are highly likely to own pair-wise shares that will enable full network connectivity. This is a major advance in that it enables node-to-node security services including peer-wise privacy.

3. Mobile Code Protection

Protecting programs from illegitimate use is a classic problem in computer science [21, 22, 23]. It turns out to be a very difficult problem to protect program execution, manipulation, and copying in an environment that is controlled by a sophisticated adversary, as may be the case in wireless sensor networks. Normally simple security functions, like protecting the privacy of an encryption key, are difficult (or impossible) when the execution environment is hostile. The ability of the adversary to make mass copies of the code to attack in parallel combined with our benchmark of guaranteeing security in the worst case paints a bleak picture.

3.1. Obfuscation

The goal of obfuscation is to disguise programs so that a user can execute them, but cannot determine their meaning. The foundational reasoning is that if an adversary does not know what the program is supposed to do, it does no good to copy it, nor can the program be changed in any meaningful way, either on the executing host, or in the code that may migrate as part of a mobile agent or other code-passing, distributed system. So, while obfuscation does not completely prevent malicious manipulation, it does limit the adversary to the narrow malice that they can cause with blind disruption, a dramatically reduced impact compared to intentioned attacks, since blind disruption is difficult to disguise.

Program obfuscation literature is deep and wide, pivoted by the seminal work by Sanders and Tschudin [24]. Others introduce practical processes based on code interleaving and term rewriting as well as more theoretical notions [25, 26, 27, 28]. Today, program, algorithm, and system obfuscation techniques are even embedded with shrink-wrapped, modern development environments and compilers. The challenge with delivered techniques is that there is no foundation for confidence in their protection or even in understanding what it is that they are intended to protect. Obfuscation is a difficult concept to grasp and attempts to model the concept are presently in a state of research flux caused by the demise of the model that had become the de facto standard.

One way to formalize the notion of program obfuscation is the Virtual Black Box (VBB) paradigm. In VBB, P', an obfuscated program for P (a program or circuit that has the same functionality and polynomially similar performance), is a VBB obfuscation of P if an adversary can prove anything using an oracle for P that it can prove if given access to the code of P'. This formalization seems to capture the intuition of program obfuscation, i.e. that having possession of the code renders nothing about the meaning of the program. Figure 1 captures the intuitive relationship between a program, its obfuscation and two probabilistic, polynomial time Turing machines that emulate them.

In their seminal paper in 2001, Barak, et al. show that it is impossible to create a general program obfuscator under the Virtual Black Box paradigm [29]. Their primary result is to contrive a family of circuits (or programs) that cannot be obfuscated in the sense that access to the circuit will always yield more information than can be gathered from oracle access to the protected circuit. Since the Barak result, some narrow classes of programs have been shown to be obfuscatable while others extended the Barak result to a wider program array. Nonetheless, obfuscation research and implementation continues.

Recently, McDonald et al., proposed an obfuscation approach based on computing Small Atomic Functions [30]. Their mechanism (hereafter referred to as SAF) provides perfect obfuscation for programs with small-sized input. This approach is appealing since it fits well with most wireless sensor network applications that are designed for computationally constrained devices. In this case, the tradeoff is not computational, as the fundamental operation is table lookup, rather the challenge is storage capacity for the large tables demanded by the SAF method.

There are two interesting foundations for obfuscating atomic functions. First, SAF demands and then leverages fixed and equal input and output sizes. For example, if we are interested in thirty two bit functions, then all functions must have exactly thirty two bits of input and thirty two bits of output. Notice that this is also a property of many encryption algorithms that are often the target of obfuscation techniques. The power here is that in this paradigm it is possible, and semantically clear, to compose any two functions or to compose any function with itself.

Secondly, the fixed input-output size scenario allows the analyst to compute the likelihood that an arbitrary function is the composition of any two other functions. So, while an adversary could determine the function that an obfuscating node is computing through black box analysis, the algorithm does not leak any information about the composition (or compositions) that resulted in the final function. Since no information is leaked to the adversary, perfect obfuscation is achieved.

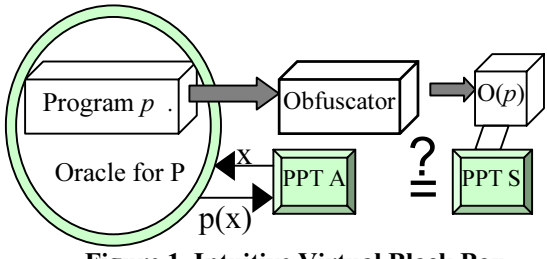


Figure 1. Intuitive Virtual Black Box

3.2. Proof Carrying Code

A novel, yet highly intuitive mobile code protection approach is to have the developer include a proof with the delivered code that definitively establishes (or ensures, or proves) the security properties relative to the implementation environment security policies. This is the goal of Proof Carrying Code (PCC) [31].

The theory behind PCC is that security information can be partitioned into (1) Security requirements in the execution environment and (2) Security capability in the delivered applications. PCC provides a framework for the environment manager to formally express their security requirements so that application developers can produce proofs that their code meets the environmental baseline. The local manager can exercise the proof to ensure that ensure that the code execution will be safe.

PCC is a static analysis method used by a host that is considering whether or not it should execute [usually] remote code. The approach is based on specification languages and formal semantics and the proofs are founded in the inherent code properties.

4. Attestation

Informally and intuitively, we think of attestation as evidence. In our case, we seek evidence about identity, but like most environments, identity evidence can come in many forms. More specifically, we are concerned about evidence that we can gather about remotely located sensor network nodes. We are truly interested in remote attestation.

While the term "attestation" has its foundation in evidence, it has recently taken on a more specific connotation in computing jargon. Here, attestation refers to one's ability to assess whether or not a communicating node is executing an authentic and unblemished version of the expected software. Such assurances challenge the limits of possibility and attestation via direct interaction with a target computer is a classic problem in computer science.

The challenge is that if a malicious intruder has modified or replaced the executing code, it may also have injected malicious software, or malware, whose purpose is to trick inquirers, preventing them from detecting the modifications. For example, consider hashing as a mechanism to allow a developer to ensure code integrity. In this environment, the developer has a replica of the distributed code and delivers the hash function itself as a component of the application software installation. To ensure code integrity, the developer may:

1. Pose a random query to the local replica code
2. Stop the computation at a randomly selected point
3. Compute the hash of the executing module in the resulting intermediate state and
4. Ask for the same computation and the resulting hash value from the host that is supposedly executing the same code.

For cooperating hosts, this is a reasonable approach. Unfortunately, in a compromised host, a malicious intruder with sufficient access privileges can subjugate the intended application. Once in control of the processor, the intruder can conduct a classic man-in-the-middle attack by using a confined, but otherwise pristine, version of

the original code as an oracle. This confined software may be in a separate device, or may be a within a virtual machine within the compromised host. When an integrity query is received, the installed malware feeds the query into the pristine code and returns the correct response to the verifier. Figure 2 illustrates the malware Man-In-The-Middle interaction.

This challenging problem has lead many researchers to seek a hardware-based solution, as we discuss in the first subsection below.

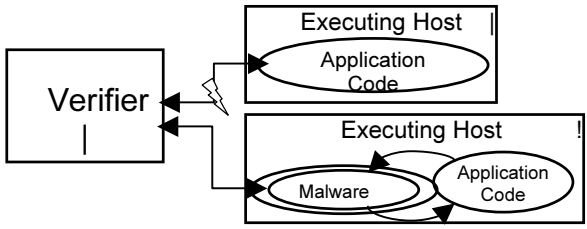


Figure 2. Malware MitM

4.1. Trusted WSN Computing Foundations

Since the Orange Book [32] introduced the notion of a Trusted Computing Base (TCB), there has been significant interest in developing a hardware platform that is malware resistant. This trusted platform would contain a hardware-embedded set of algorithms that cannot be replaced by a malicious intruder no matter what level of privilege they attain, thus is not subject to software tampering. These algorithms must be well-understood and also embedded with a private key that can authenticate trusted platform actions. Moreover, this platform must be designed to be hardware tamper-resistant to ensure that an adversary could not defeat the platform's security properties by replacing some hardware component.

While the Orange Book may have triggered the search for secure hardware, industry has driven the more recent efforts. The Trusted Computing Group web page[2] details the state of the art in their approach to achieving some level of trusted hardware in general purpose computing devices. The Trusted Computing Group is a consortium of technology industry companies with common interests in trusted platforms.

Maybe the most visible company involved in establishing a trusted computing platform is the Microsoft Corporation. Introduced as Palladium and now identified as the Next Generation Secure Computing Base, their effort addresses the software impact and technology that can leverage trusted hardware.

A computing change of focus is at the heart of the trusted computing movement. Functionality, inclusiveness, and connectivity has been the foundation of computer development methodology since the mid-1900s and they certainly drove the technology that now forms the Internet. Design decisions reflected this mindset, consistently favoring operational capabilities over attribution, heterogeneity over confinement, speed over authentication, etc. This is not surprising, as designers have traditionally been driven to meet the customer's immediate functional requirements rather than designing products for foundational or future protection.

[2] https://www.trustedcomputinggroup.org/home

While many security professionals strongly support the trusted computing effort, the Trusted Computing Group and the Next Generation Secure Computing Base effort have been the subject of complaints of privacy invasion and unfair competition by others concerned about the control shift that hardware security features hands to software vendors. At the center of these complaints is that the trusted computing platform can aid software vendors in protecting their intellectual property by allowing them to more easily and completely enforce copyrights and to prevent or complicate software piracy to implement digital rights protection.

Trusted computing technology itself is not particularly sophisticated or complex. Rather, it depends on a single supplemental chip with special security functionality and restrictive access policies. The security capabilities themselves leverage well-known vulnerability and security targeted areas of memory confinement, protected storage that can host strong cryptographic keys and corresponding encryption algorithms and by establishing strongly private input and output channels.

Another key provision of trusted computing that has attracted much of this criticism is the approach it takes to remote attestation. We discuss remote attestation in the next subsection, so for now, suffice it to say that antagonists claim that the trusted computing group approach inappropriately favors vendors by leveraging the ability to identify code changes into an ability to prevent users from modifying legitimately purchased products. The debate, and the trusted computing movement, continue.

Each of the four targeted trusted computing security capabilities directly addresses important threats in modern computing. For example, buffer overflows are overwhelmingly the dominant computer vulnerability today. In buffer overflow attacks the intruder leverages a memory overrun to inject malicious code, which can be detected and in some cases prevented by remote attestation. Samples of vulnerabilities addressed by trusted computing is given in Table 2.

Protected User Interface	**Memory Confinement**	**Protected Persistent Storage**	**Remote Attestation**
Screen Captures	Process-to-Process Leaks	Key protection	Malware
Keyboard Loggers	Covert Channels	Strong Encryption	Software Piracy

Table 2. Sample Trusted Computation Security Capabilities

At face value, trusted computing does not seem well-suited to wireless sensor networks. Fundamentally, wireless sensor network nodes are small devices with limited power capabilities that may not support supplemental hardware components. Additionally, the extra computing components add per-node cost, which may prevent wide acceptance where most applications demand hundreds or thousands of nodes.

Still, some researchers have suggested using trusted computing in wireless sensor networks. In their Internet manuscript [33], Ganeriwal, et al. suggest a tiered architecture consisting of a few trusted nodes that perform trusted computation for the vast untrusted sensor network itself.

In their approach, the Trusted Platform Modules (TPM) are tamper resistant, contain protected memory and a high quality random number generator, and are programmed with a unique RSA key pair. The key exchange protocol is a typical trusted key server variation triggered by a sensing node's request for a key and

containing a nonce encrypted. Key generation is initiated by a request from a non-TPM equipped node who forwards a message containing a nonce encrypted with a newly generated ephemeral key and the ephemeral key encrypted under the TPM node's public key, which is a canonical approach to establish a session key.

This approach is interesting in that it leverages hierarchal design to optimize computing capability and to preserve battery power for the multitude of nodes. In fact, if the ratio of TMP nodes to sensing nodes is sufficiently small, it could mitigate the wireless sensor network trusted computing cost limitation even if the cost for substantially stronger battery power or more powerful processors substantially elevates the cost of the trusted nodes.

4.2. Remote Software Attestation

Detecting malware and ensuring code integrity is difficult when the verifier has physical access to the host in question. The task is even greater when the code must to be verified resides remotely.

Remote software attestation generally leverages detailed, orthogonal knowledge about the code and the environment. Specifically, the verifier must mirror the target's execution environment and must be able to extract precise timing measures, both for code execution and for message transmission. The two canonical, orthogonal architectural mechanisms are: (1) A correct selected memory hash computation and (2) A measure of the time that it takes to perform memory hash computation.

A foundational idea behind remote software attestation is that to spoof the verifier, a malicious host must be forced to execute computations that would take sufficiently long that the verifier could detect the delay. Thus, the remote attestation process is an interactive system where the verifier poses some question or query to the responder and the responder provides an answer that the verifier has pre-computed. If the responder provides an incorrect response, or does not provide the correct response within the prescribed time threshold, the verifier identifies the responder as a compromised node.

Shaneck, et al. propose a Remote Software-based Attestation scheme for wireless sensor networks [34]. They leverage program obfuscation techniques such as self-modifying functions, ruse code, and indirection to protect their verification procedure from spoofing. Still their scheme reduces to a challenge-response interaction based on random numbers, hashing, and measuring the time that it takes the responder to generate the answer, similarly to other attestation approaches.

Another remote software attestation approach is called SWATT, short for "A Software-based ATTestation for Embedded Devices" [35]. SWATT follows the software attestation model given above by implementing an interactive verification process based on computing memory hashes and enforcing response timing limits. Since the challenges request random memory samplings, the adversary cannot guess, or pre-compute, responses and the timing threshold prevents adversaries from modifying the system on the fly to enable them to compute the correct response to defeat the verification. Thus, it is not breakthrough technology from a theoretical standpoint.

SWATT does offers several important optimizations, some that are particularly critical to wireless sensor networks. For example, the author's argue (though they do not prove) that their verification code performance is optimal, at least in the sense that its performance cannot be further optimized. Of course this optimization is critical for the foundational purpose, i.e. if an adversary can further optimize the verification process, they could subvert the timing threshold that is pivotal to the verification

process. Performance optimization is also important to computationally and power constrained devices, such as sensor nodes, particularly if attestation frequently recurs.

The SWATT authors offer a myriad of potential remote software attestation applications inncluding: Virus checkers, voting systems, smart cards, personal digital assistants, cell phones, and network printers. This is clearly a short list as most computer applications would benefit from integrity protected execution.

Still, remote software attestation faces many challenges, foremost among them is attestation's temporal limitation. At best, remote software attestation can only ensure that the verified code was pure at time, say T. It does not ensure the code's status just prior to verification (at time $T - \theta$), nor can it predict the code state shortly after the verification (at $T + \theta$). While there is value in ensuring that no malware is present at a given point in time, secure computing demands perpetual attestation.

Finally, the authors of SWATT point out a possible attack that illustrates the difficulty in predicting threats and limitations to remote software attestation. To their considerable credit, they acknowledge a possible attack on their verification if large portions of memory are blank. In this case, the attacker can optimize the checksum process and gain significant time that may enable alternate response generation, though the authors suggest randomizing empty memory to mitigate this threat. Though this vulnerability technically simple, its threat is subtle, and it illustrates that inherently subtle threats exist. We point out that the authors do not prove that no other such subtle threats exist.

4.3. Guaranteed Code Execution Integrity

The natural next step in the evolution toward fully trustable computing is to develop mechanisms that can ensure that a given code segment executes (or executed) precisely as it was intended. That is, we would like to be able to ensure that once a code segment begins execution, it cannot be interrupted nor can its execution sequence and manifestation be altered in any other way.

The clear logical progression is to follow remote attestation with guaranteed atomic execution for the target application. Such mechanisms would justify strong behavioral expectations about code executing in those environments and corresponding confidence in its output..

Indisputable Code Execution (ICE) was first proposed in [36]. As an incremental extension of Pioneer [37], ICE performs a series of checksums on itself, thus coining the term *self-checksumming code*. These checksums are designed to ensure that any code alteration will be detected in the verification process. However, even if self-checksumming can ensure that there is no change to the executing code, it cannot guarantee execution integrity on its own, since interrupt processing can fundamentally alter the application's manifestation.

To protect against interrupts and other process-altering attacks, ICE sets up environmental state to protect the execution sequence and includes important state information in the checksum presented to the verifier. The program counter, the operating system interrupt-disabled bit, and the data pointer to the requested memory areas are three examples of state retention that are embedded in the checksum.

Integrity protected execution is a critical step toward trustable computing platforms and ICE is an important contribution in this area. While the ICE developers note that ICE cannot prevent the Man in the Middle attack discussed above and illustrated in Figure 2, it is still an important step toward verifiably trustable computing.

5. Leveraging Remote Attestation

There are many scenarios where it is important to have confidence in software integrity. In many wireless sensor networks, remote attestation can capture the only necessary identity information to ensure accurate information collection, which is the goal in most wireless sensor networks. In this section, we illustrate remote attestation by addressing two issues that are important to wireless sensor networks: (1) Secure data aggregation and (2) Secure code update.

5.1. Data Aggregation

In virtually all sensor applications, sensor network nodes gather data and report back to a central collection site. In situations where the reporting path is long, nodes may aggregate data to conserve power and bandwidth. Of course aggregation must preserve the essential data properties for each aggregated message, thus many aggregation methods are application specific. Nonetheless, there is significant work on general aggregation mechanisms for sensor networks, e.g. [38, 39, 40, 41].

Identity requirements vary for preserving integrity in data aggregation based on application properties. For example, Tilak, et al. [42] discuss four data delivery models for sensor networks as continuous, event-driven, observer-initiated and hybrid observation models. In the continuous model, sensors produce a continuous data stream, while the event-driven model triggers transmission when an event of interest takes place. For example, a sensor placed at a traffic light may be required to transmit data every time a vehicle runs a red light.

In the observer-initiated model, sensors transmit data only upon the observer's request. For example, an observer may request location information from a sensor armed with a location finding application. In the hybrid model, one or more of the first three models coexist in the network.

For continuous transmission, session hijacking is the primary integrity threat, whereas intermittent transmission paradigms may necessitate frequent session initialization and the accompanying authentication demands.

Yang, et al. introduce a peer-to-peer aggregation method that they call SDAP, short for "A Secure Hop-by-Hop Data Aggregation Protocol for sensor networks" [43]. SDAP leverages the natural property in data aggregation protocols that nodes nearest to the collection point can have the greatest impact on the accuracy of the accumulated result. Thus, SDAP generates a flat aggregation structure with fewer nodes per accumulation group and more nodes reporting to the network root. Individual node deviations are ignored or addressed with skew balancing mechanisms.

SDAP is a four step protocol. First, nodes are probabilistically separated into groups. Then, a hierarchal data aggregation protocol collects data at the base station. These data are then analyzed for suspicious content and finally each group participates in an attestation process that verifies their reported result.

The aggregation protocol itself is straightforward, based on a simple message format containing the transmitting node identity, the hop count, the encrypted aggregated data, and a MAC to authenticate the message. Each message is encrypted with a pair wise symmetric key shared between parent and children nodes.

The authors propose several attestation approaches that can provide varying confidence levels depending on the unique application requirements. Attestation begins

with an attestation request from the base station that includes random numbers, one that act as the request identifier and the other as the attestation seed.

At each group, a probabilistic algorithm establishes the group attestation path. The algorithm assures that the probability of any node being included in the attestation path is proportional to their data count, thus ensuring that higher reporting nodes are more likely to be verified. The attestation messages are constructed so that the base station can compute the node counts and verify the earlier reported result, or can detect false or altered reports.

5.2. Secure Code Update

Software binaries have been remotely installed and update since the first networked systems were introduced. In the past five years automatic software update distribution has become the norm, in fact, is uniform among major distributed software vendors. It is no wonder that secure code update is an active research and development area.

Updating code in wireless sensor networks is an interesting concept. Some sensor applications may outlive software versions or may demand frequent updates, depending on environmental changes or even on mission shifts. There is substantial work focused on protocols to allow sensor software updates.

Two approaches to secure code update for wireless sensor networks recently emerged in the research literature. The earlier, mysteriously dubbed "Sluice" [44], was developed specifically for wireless sensor networks and is constructed around the network reprogramming paradigm. In that paradigm, code updates propagate from the base station outward, with node-to-node update occurring after the original transmission.

Sluice leverages digital signatures and hash chaining to prevent malicious nodes from transporting dubious updates to non-corrupted nodes. The three phase protocol begins by the base station partitioning the code update into transmission and storage-appropriate sized pages. Sluice creates a cryptographic hash for each page before it is transmitted, and the hash is attached to the previous page before it is hashed, thus forming a hash chain. The final packet is signed rather than hashed. As each page is received, the hash chain is verified and the page is transferred to persistent storage for further distribution. Once the signature is verified, the update is installed locally and transmitted according to the reprogramming algorithm.

The second secure code update approach is based on a combination of remote software attestation and execution integrity guarantees. Secure Code Update By Attestation (SCUBA) is also targeted for sensor networks [36]. Comparatively, while Sluice provides integrity guarantees to the updated node, SCUBA provides strong security assurances to the updating node. Moreover, SCUBA is a stronger mechanism than Sluice in that it can repair infected nodes as long as the hardware is not disturbed.

SCUBA makes two simple assumptions, first that there is secure, read-only memory to store a node's identity and private key and second that there is a public key infrastructure in place. Each of these are reasonable, though neither is guaranteed and there is no widely accepted standard for the second.

We do not delve deeply into the SCUBA technology, as it is based on SWATT and ICE (ICE is actually introduced in the SCUBA paper [36]), which we reviewed in earlier sections. SCUBA simply employs ICE to ensure that the executing update code will not be interrupted or altered in any way. The update then ensures that the proper code is installed and that no malicious remnants remain. SCUBA's greatest strength is

that it guarantees that it will repair a malware infected node or that the base station will detect its inability to complete the update.

6. Chapter Summary

Identity is a fleeting concept. Without waxing philosophical, it safe to say that the complexity of approximating identity generally increases disproportionately as the identity standard strengthens. Recognizing a face or a voice may be easier than remembering a name, and all three are easier than remembering a social security number.

Network identification is similarly complex. Sometime sharing a key provides sufficient confidence, while other applications demand sophisticated protocols to ensure that a communicating principle can be granted access to a resource or can properly be associated with a previous action.

In this chapter, we describe numerous identity properties and correspondingly, many identity verification mechanisms. We show how obfuscation can protect identity properties in mobile code and we address some key distribution capabilities and limitations as they relate to network identities.

Identity in wireless sensor networks often reduces to simply knowing that a node is executing the correct software. This chapter culminates with the three subjects as they pertain to wireless sensor networks:

- Integrity guaranteed execution
- Remote software attestation and
- Secure code updates

These three capabilities advance sensor software technology and clear the way for more advanced, sensitive, and mission critical wireless sensor network applications.

References

[1] D. Gollman, “What do we mean by entity authentication?” In Proceedings of Symposium in Research in Security and Privacy, pp. 46-54. IEEE Press, 1996.

[2] P. Ramachandran and A. Yasinsac. “Limitations of On Demand Secure Routing Protocols.” IEEE Information Assurance Workshop 2004, pp. 52-59, June 10-11, 2004.

[3] Y.C. Hu, A. Perrig, and D. B. Johnson. “Wormhole Detection in Wireless Ad Hoc Networks.” Technical Report TR01-384, Department of Computer Science, Rice University, December 2001.

[4] T. Beth & Y. Desmedt, "Identification Tokens-or: Solving the Chess Grandmaster Problem", In A. J. Menezes & S.A. Vanstone, editors Proc. CRYPTO 90, pp. 169-77. Springer-Verlag, '91. LNCS, 537.

[5] Tina Suen and Alec Yasinsac, "Peer Identification in Wireless Sensor Networks Using Signal Properties", Proceedings of the 2005 IEEE Workshop on Wireless and Sensor Network Security, Nov. 7-10, 2005, Washington DC

[6] M. Barbeau, J. Hall, and E. Kranakis, "Detecting Impersonation Attacks in Future Wireless and Mobile Networks," in Workshop on Secure Mobile Ad-hoc Networks and Sensors (MADNESS 2005), vol. 4074, LNCS. Singapore: Springer-Verlag, 2006.

[7] Roger M. Needham, Michael D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers", Communications of the ACM, December 1978, Volume 21, Number 12, pp. 993-999

[8] D. E. Denning and G. M. Sacco, "Timestamps in key distribution protocols," Communications of the ACM, vol. 24, no. 8, Aug 1981, pp. 533-536

[9] M. V. D. Burmester and Y. Desmedt. "A Secure and Efficient Conference Key Distribution System", In A. D. Santis, editor, Advances in Cryptology - EUROCRYPT '94, volume 950 of Lecture Notes in Computer Science, pp. 275-86. Springer-Verlag, 1995

[10] Michael Steiner, Gene Tsudik, & Michael Waidner, "Key Agreement in Dynamic Peer Groups", IEEE Trans on Parallel and Dist Systems, Vol. 1 No. 8, Aug 2000, pp 769-80
[11] William A. Wulf, Alec Yasinsac, Katie S. Oliver and Ramesh Peri. A Technique for Remote Authentication. Proceedings of the Internet Society Symposium on Network and Distributed System Security. 1994.
[12] J. R. Douceur, "The sybil attack," in IPTPS '01: *Revised Papers from the First International Workshop on Peer-to-Peer Systems*. London, UK: Springer-Verlag, 2002, pp. 251–260.
[13] C Karlof, N Sastry, D Wagner, "TinySec: a link layer security architecture for wireless sensor networks", Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems, November 2004, pp. 1-10
[14] Newsome, J. Shi, E. Song, D. Perrig, A., "The Sybil attack in sensor networks: analysis & defenses", Third International Symposium on Information Processing in Sensor Networks, April 26-27, 2004, pp. 259- 268, ISBN: 1-58113-846-6
[15] John Marshall, Vikram Thakur, and Alec Yasinsac, "Identifying Flaws in the Secure Routing Protocol", *Proceedings of The 22nd International Performance, Computing, and Communications Conference* (IPCCC 2003) April 9-11, 2003, pp. 167-174
[16] Tadayoshi Kohno, Andre Broido, and K.C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, 2(2), April-June 2005.
[17] Laurent Eschenauer, Virgil D. Gligor, "A key-management scheme for distributed sensor networks," *Proceedings of the 9th ACM conference on Computer and communications security*, pp. 41 – 47, Washington, DC, USA, 2002
[18] Peng Ning and Donggang Liu, "Establishing pairwise keys in distributed sensor networks," *Proceedings of the 10th ACM Conference on Computer and Communications Security*, pp. 52-61, Washington D.C., USA 2003
[19] Mohamed Eltoweissy, M. Hossain Heydari, Linda Morales, and I. Hal Sudborough, "Combinatorial Optimization of Group Key Management", Journal of Network and Systems Management, Volume 12 , Issue 1, March 2004, pp. 33-50, ISSN:1064-7570
[20] Whitfield Diffie and Martin Hellman, "New Directions In Cryptography," IEEE Transactions on Information Theory, IT-22(6):644-654, November 1976
[21] David Aucsmith, "Tamper Resistant Software: An Implementation", Proceedings of the First International Workshop on Information Hiding, Pages: 317-33, 1996, ISBN:3-540-61996-8, LNCS 1174
[22] D. Lie, C. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. Mitchell, and M. Horowitz. Architectural Support for Copy and Tamper Resistant Software. In Proceedings of the 9 Int'l Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOSIX) , pages 169--177, November 2000.
[23] Toshio Ogiso ,Yusuke Sakabe,Masakazu Soshi,and Atsuko Miyaji, "Software Tamper Resistance Based on the Difficulty of Interprocedural Analysis", WISA 2002, Cheju Island, Korea, August 28-30, 2002
[24] T. Sander, C. Tschudin, "Protecting Mobile Agents against Malicious Hosts", Lecture Notes in Computer Science, Special Issue on Mobile Agents, Edited by G. Vigna, 1998
[25] C. Wang, J. Hill, J. Knight, J. Davidson, "Software Tamper Resistance: Obstructing Static Analysis of Programs", Tech Report: CS-2000-12, University of Virginia, Computer Science Department, 2000.
[26] L. D'Anna, B. Matt, A. Reisse, T. van Vleck, S. Schwab, P. LeBlanc. "Self-Protecting Mobile Agents Obfuscation Report". Network Associates Laboratories, Technical Report 03-015 (final), 6/30/ 2003.
[27] G. Karjoth, N. Asokan, and C. Gulcu, "Protecting the computation results of freeroaming agents," in Kurt Rothermel and Fritz Hohl, editors, Proc. of the Second International Workshop, Mobile Agents 98, LNCS 1477, Springer-Verlag, pp. 195-207, 1998
[28] F. Hohl (1998): Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts. In: Mobile Agents and Security. Springer-Verlag, pp. 92-113, 1998
[29] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang, "On the (im)possibility of obfuscating programs," in Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '01), J. Kilian, Ed. Santa Barbara, California: Springer-Verlag, LNCS 2139, Aug. 19-23 2001, pp. 1--18
[30] Alec Yasinsac and J. Todd McDonald, "Towards Working With Small Atomic Functions", the *Fifteenth International Workshop on Security Protocols*, Brno, Czech Republic, 18-20 April 2007, LNCS
[31] George C. Necula and Peter Lee, "Research on Proof-Carrying Code for Untrusted-Code Security," In Proceedings of the 1997 IEEE Symposium on Security and Privacy, Oakland, 1997
[32] DoD 5200.28-STD, "Trusted Computer System Evaluation Criteria (TCSEC)," 15 Aug 83

[33] Saurabh Ganeriwal, Srivaths Ravi, Anand Raghunathan, "Trusted Platform based Key Establishment & Management for Sensor Networks," unpublished manuscript, 2007

[34] M. Shaneck, Mahadevan, V. her, and Y. Kim, "Remote Software-based Attestation for Wireless Sensor Networks,", Proceedings of the *2nd European Workshop on Security and Privacy in Ad hoc and Sensor Networks*, July 13-14, 2005, Visegrad, Hungary, pp. 27-41

[35] Arvind Seshadri, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla, "Swatt: Software-based Attestation for Embedded Devices," Proceedings of the *IEEE Symposium on Security and Privacy*, May 2004

[36] Arvind Seshadri, Mark Luk, Adrian Perrig, Leendert van Doorn, Pradeep Khoslak, "SCUBA: Secure Code Update By Attestation in Sensor Networks," *ACM Workshop on Wireless Security*, September 29, 2006, Los Angeles, California, USA, pp. 85-94

[37] Arvind Seshadri, Mark Luk, Elaine Shi, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla, "Pioneer: Verifying integrity and guaranteeing execution of code on legacy platforms," In *Proceedings of ACM Symposium on Operating Systems Principles* (SOSP), pp. 1–15, October 2005.

[38] Lingxuan Hu and David Evans. "Secure Aggregation for Wireless Networks", Workshop on Security and Assurance in Ad-Hoc Networks. January 2003.

[39] Konstantinos Kalpakis, Koustuv Dasgupta and Parag Namjoshi, "Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks", Proceedings of the 2002 IEEE International Conference on Networking. August 2002.

[40] Bartosz Przydatek, Dawn Song and Adrian Perrig, "SIA: Secure Information Aggregation in Sensor Networks", Conference on Embedded Networked Sensor Systems. November 2003.

[41] Khandys Polite, "PPDA: Privacy Preserving Data Aggregation in Wireless Sensor Networks", FSU CS Technical Report #040502, May 2004

[42] Sameer Tilak, Nael B. Abu-Ghazaleh and Wendi Heinzelman, "A Taxonomy of Wireless Micro-Sensor Network Models< ACM Mobile Computing and Communication Review. April 2002.

[43] Yi Wang, Xinran Wang, Sencun Zhu, and Guohong Cao, "SDAP: A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks", *Mobihoc'06*, May 22-25, 2006

[44] Patrick E. Lanigan, Rajeev Gandhi, Priya Narasimhan, "Sluice: Secure Dissemination of Code Updates in Sensor Networks," p. 53, *26th IEEE International Conference on Distributed Computing Systems* (ICDCS'06), July 4-7, 2006

Wireless Sensor Network Security
J. Lopez and J. Zhou (Eds.)
IOS Press, 2008

On the Hardware Implementation Efficiency of Cryptographic Primitives

RODRIGO ROMAN [a], CRISTINA ALCARAZ [a], NICOLAS SKLAVOS [b]

[a] *University of Malaga, Spain, emails: {roman, alcaraz}@lcc.uma.es*
[b] *University of Patras, Greece, email: nsklavos@ieee.org*

Abstract. Security has been proven a crucial factor in the provision of data services and especially in the computer-related environments. While wired and wireless networks come to all sectors of everyday life, security tries to satisfy the growing needs for confidentiality, integrity and non-repudiation. There are many instances of security primitives and each one of them has different requirements in terms of processing power, memory, energy consumption, etc. Therefore, it is important to review the functionality of the less resource-demanding encryption algorithms in order to analyze their theoretical suitability to the existent sensor node hardware. Still, the constraints inherent to the sensor nodes advise against the total dependence on software-based implementations, even more in the case of expensive primitives.

Keywords. Security, Cryptographic Primitives, Hardware Implementations.

1. Introduction

While the wireless devices are coming to the offices and houses, the need for strong secure transport protocols seems to be one of the most important issues in the communications standards. From email services to cellular provided applications, from secure internet possibilities to banking operations, cryptography is an essential part of the today's users needs.

Recent and future computer networks have special needs for cryptography. They must support the three basic types of cryptography: Bulk Encryption, Message Authentication and Data Integrity. Most of the widely used wireless systems support all the above different types of encryption. Additionally, some systems offer to the users the choice to select among two or three alternative ciphers for each encryption operation. The user can select the best-suited algorithm according to the application needs. In most of the cases, the same encryption system implementation supports all the three different types of cryptography.

The standards for network applications and services are maturing and new specifications in security systems are being defined. This leads to a large set of possible technologies that a service provider can choose. Although organizations and forums seem to agree to the increasing need for secure systems with wide strength, security is a

black hole in the computer networks because of the implementation difficulty. The security layers of many communication protocols use outdated encryption algorithms, which have been proved unsuitable for hardware implementations, especially for constrained implementation environments.

In general, encryption algorithms use complicated arithmetic and algebraic modifications, which are not appropriate all the time for integrations with high quality implementation performance. That's why the integrations of the encryption algorithms allocate many of the system resources, in hardware or in software terms, in order to be implemented as separated components.

In many cases software applications have been developed, in order to support the security and cryptography needs. But, software solutions are not acceptable for the cases of computer-related environments, with high speed and performance specifications.

2. Security & Privacy in Hardware

In any environment, either physical or logical, there exists the need of maintaining someone or something safe, away from harm. This is the role of security. On any computer-related environment, security can be considered as a non-functional requirement that maintains the overall system usable and reliable, protecting the information and information systems. In order to comply with this non-functional requirement, such environments must assure the existence of certain properties by implementing new protocols or extending the current ones. These properties and protocols can be equally useful for another non-functional requirement, privacy, which is related to the capacity of controlling the flow of information about a certain subject or entities.

Major security properties related to information assurance are confidentiality, i.e. ensure that the information is accessed only by those who are inside the system, integrity, i.e. guarantee that the information has not been manipulated, authentication, i.e. certify that the source of the information is who claims to be, and non-repudiation, i.e. assure that no party can deny having participated in a part or the whole transaction. Other security properties focus more on the functionality of the overall system: Robustness and Availability are related to the capacity of the system of providing an acceptable level of service in the presence of problematic situations, whereas Resilience is the ability to withstand "Node Compromise" attacks that can try to take control of the whole system.

In a sensor network environment the existence of security, that is, the need of ensuring that the existent devices and protocols comply with the major security properties is extremely important due to the associated constraints to the elements of the network and their particular behaviour. The elements of a sensor network, the sensor nodes, are highly constrained in terms of computational capabilities, memory, communication bandwidth, and battery power. Additionally, it is easy to physically access these nodes because they must be located near the physical source of the events, and they usually are not tamper-resistant due to cost constraints. Furthermore, any device can access the information exchange because the communication channel is public.

As a result, any malicious adversary can manipulate the sensor nodes, the environment, or the communication channel for its own benefit. For these reasons, it is

necessary to provide the sensor network protocols with basic security mechanisms that can guarantee a minimal protection to the services and the information flow. As a result, there is a need to protect the communication channels between the nodes, and to defend the core protocols of the network, i.e. routing, data aggregation, and time synchronization, and other non-essential protocols, such as node location, code update procedures, and other collaborative processes, against any external or internal influence.

The foundation of all these secure protocols, and the tools that can aid these systems on integrating the security properties into their operations, are the cryptographic primitives: Symmetric Key Cryptography (SKC), Public Key Cryptography (PKC), and Hash functions. These primitives alone are not enough to protect a system, since they just provide the confidentiality, integrity, authentication, and non-repudiation properties. Nevertheless, without these primitives, it would be nearly impossible to create secure and functional protocols.

There are many instances of these primitives (i.e. algorithms), and each one of them has different requirements in terms of processing power, memory and energy consumption, etc. Therefore, it is important to review the functionality of the less resource-demanding algorithms in order to analyze their theoretical suitability to the existent sensor node hardware. Still, the constraints inherent to the sensor nodes advise against the total dependence on software-based implementations, even more in the case of expensive primitives such as PKC (cf. Section 3). The time a single node spends executing these primitives could be used for other primordial tasks, or even for implementing the necessary steps that are required to provide other security properties. Consequently, it is crucial to analyse the influence of the implementation of these primitives on hardware over the behaviour of the node.

2.1. Symmetric Key Cryptography

Symmetric Key Cryptography (SKC) primitives are able to offer the necessary mechanisms for protecting the confidentiality of the information flow in a communication channel. For achieving this confidentiality level it is necessary that both the origin and destination share the same security credential (i.e. secret key), which is utilized for both encryption and decryption. As a result, any third-party that does not have such secret key cannot access the information exchange, thus the security properties of these primitives mainly resides on the complexity of their internal operations and the length of the keys.

There are two types of SKC primitives: Stream Ciphers and Block Ciphers. Stream Ciphers are simpler and faster, while Block Ciphers are more flexible and powerful. These two types are discussed more profoundly in the following sections.

2.1.1. Stream Ciphers

A stream cipher is a method of symmetric cryptography that takes as an input a flow of bits of variable size and transforms it into a ciphertext. For that purpose, these cryptographic algorithms combines, mainly by using simple operators such as XOR, those input bits with a pseudorandom key flow obtained from a cryptographic key and an initialization vector (IV). It is essential to use such IV in stream ciphers for avoiding situations where one plaintext produces the same ciphertext when encrypted with the same key.

One of the better known stream cipher algorithm, which is used by several protocols like Secure Sockets Layer (SSL) or Wi-Fi Protected Access (WPA), is RC4 [RC4_1996]. This simple algorithm was designed by Ron Rivest in 1987, and it is also known as ARC4 or ARCFOUR. Its simplicity is due to two main reasons. Firstly, it minimizes the execution time by using simple operations as XOR, AND, addition and shifting. And secondly, it uses a block size of 8-bit, consuming less memory space than others cryptosystems. Therefore, RC4 is highly suitable for those architectures with highly-constrained microcontrollers. However, RC4 must be implemented carefully for avoiding any attacks as the detected in WEP [Fluh2001], where the primitive was not properly initialized.

2.1.2. Block Ciphers

A symmetric block cipher operates on a group of bits with a fixed-size (usually 64 or 128 bits) known as blocks. A block of plaintext is transformed by using several rounds of mathematical operations into a block of ciphertext, which has the same size as the input. Besides of those operations, in every round most block ciphers also have to transform the original key, using a usually complicated process named key schedule. In cases where more than one block has to be encrypted (e.g. a plaintext that is larger than the block size), it is necessary to use a mode of operation. Moreover, if the plaintext is not a multiple of the block size, some modes of operation require padding that plaintext.

Operation modes are just a set of operations using the block cipher inputs and outputs. Some of them provide confidentiality, and a few of them provide authentication. Concretely, the earliest modes, as ECB (Electronic Codebook), CBC (Cipher Block Chaining), OFB (Output Feedback) or CBC (Cipher Block Chaining), provide mainly only confidentiality. However, operation modes such as CCM (Counter with MAC), GCM (Galois Counter Mode), OCB (Offset Codebook Mode) and others modes guarantee both security properties by including a Message Authentication Code (MAC). In addition, in order to generate some randomization in the process, these modes, except ECB, needs an initialization vector (IV) that has to be combined with the original key.

The following subsections presents a set of primitives that could be used for providing security properties in highly constrained devices. These primitives are arranged according to their complexity, and hence according to the resources that they demand to a device. Concretely, they are classified from less to more complex.

2.1.3. Skipjack

Skipjack [Skipjack_1998] is considered one of the simplest and fastest block ciphers, in comparison with other ciphers such as RC5, RC6 or AES. It was designed by the NSA (U.S. National Security Agency), and declassified in 1998. The primitive uses building blocks of 64 bits and a cryptographic key of 80 bits for encrypting data blocks of 16 bits. Therefore, it is especially suitable for constrained nodes with 16-bit microcontrollers, such as the MSP430 microcontroller family.

Skipjack alternates two stepping rules, named A and B, during 32 rounds. Concretely, each rule can be described as a linear feedback shift register with additional non linear keyed G permutation (Feistel cipher of 4 rounds). Also, the key schedule of Skipjack is straightforward, i.e. the key is cyclically repeated enough times to fill the key schedule buffer. Besides these obvious benefits regarding the simplicity

of the algorithm, this primitive does not provide the same security than others, due to its small key size.

2.1.4. RC5 and RC6

In 1994, Ron Rivest designed a simple symmetric algorithm named RC5 [RC5_1994]. This primitive has variable parameters both in block size (32, 64, 128 bits), in key size (up to 2040 bits) and in number of rounds (up to 255), although it is recommended to use a block size (b) of 64 bits, a key size of 128 bits and 20 rounds. On the other hand, its internal operations are quite simple: integer additions, bitwise XOR, and variable rotations, over two $^{b}/_{2}$ bits registers. An algorithm based on RC5 is RC6 [RC6_1998], which was designed by Ron Rivest, Matt Robshaw, Ray Sidney and Yiqun Lisa Yin in 1998. It is very similar to RC5, except that its recommended block size is 128 bits, and internally includes integer multiplication and uses four $^{b}/_{4}$ bits registers.

In spite of using simple building blocks, resulting in a small code size in their implementation, both algorithms are a bit more complex and quite slow, thus not all the highly-constrained devices could support them. Basically, they use 32-bit registers for the recommended block size (i.e. $^{b}/_{2}$ where b=64 in RC5, and $^{b}/_{4}$ where b=128 in RC6), the number of rounds is a little high, and the key schedule is quite complex in comparison with simpler ones such as in Skipjack. For that reason, the more appropriate microcontrollers are those ones with 32 bits, for example the microcontrollers PXA271 or ARM920T.

2.1.5. AES &Twofish

The Advanced Encryption Standard (AES) [AES2002] was designed in 1998 by Joan Daemen and Vincent Rijmen. It is a derivative of the Rijndael algorithm, which supports a larger range of block sizes ranging between 128 bits and 256 bits. On the other hand, AES works with a fixed block size of 128 bits and a key size of 128, 192 or 256 bits, and depending on the key size the number of rounds is 10, 12 and 14 respectively.

For encrypting, AES needs a 4 × 4 array of bytes (termed the state) over a finite field. In each round (except the last round) four distinct stages are executed: AddRoundKey, where the subkey is combined (XOR operation) with the state, SubBytes, where each byte is replaced with its entry in a fixed 8-bit lookup table, ShiftRows, where bytes in each row of the state are shifted cyclically toward the left, and MixColumns, where each column is multiplied with a fixed polynomial p(x).

Another block cipher algorithm similar to AES is Twofish [Twofish1999], designed in 1998 by Bruce Schneier, John Kelsev, Doug Whiting, David Wagner, Chris Hall and Niels Ferguson. Twofish uses 128-bit blocks with large key sizes (up to 256 bits), a complex key schedule, and four different key-dependent 8 by 8 bits S-boxes totally bijectives. Besides, it uses other elements belonging to other cipher families, concretely the pseudo-Hadamard transform (PHT(a,b)=α,β, where $\alpha = a + b \bmod 2^{32}$ and $\beta = a + 2b \bmod 2^{32}$), and the maximum distance separable (MDS) matrix (4 × 4) over GF(2^8).

Both AES and Twofish are quite fast in their encryption/decryption operations due to their small number of rounds, and also some operations can be calculated in 8-bit registers. On the other hand, they present a considerable complexity, which results in a

larger code size. Therefore, not all constrained devices can afford to implement them in software.

2.2. Public Key Cryptography

Public key cryptography (PKC), also known as asymmetric cryptography, is a form of cryptography that uses two keys: a key called secret key, which has to be kept private, and another key named public key, which is publicly known. Any operation done with the private key can only be reversed with the public key, and vice versa. This nice property makes all PKC-based algorithms useful for secure broadcasting and authentication purposes. It is also an invaluable tool for allowing the secure exchange of secret keys between previously unknown partners.

The computational cost of calculating the underlying primitive operations had hindered its application in highly-constrained devices, such as sensor nodes. However, at present there are several PKC primitives that are being considered for a sensor network environment. Obviously, it is important to know their properties to determine if they could be suitable in a constrained architecture. For example, the algorithm RSA [RSA1978] is known for offering good cryptographic operations (encrypting and signing), but limited microprocessors must pay a relatively high computational cost. For all these reasons, it is necessary to analyze what kind of PKC primitives are more suitable for the sensor nodes.

2.2.1. Elliptic Curve Cryptography

In 1985, Koblitz and Miller proposed one of the most investigated PKC primitives in the field of WSN security, named Elliptic Curve Cryptography (ECC) [ECC2000]. This interest is due to some interesting features offered by ECC that are appropriate for WSN context, such as the small key size and fast computation. ECC is based on elliptic curve structures defined as $y^2 = x^3 + ax + b$ over finite fields $\mathbb{F}_p$. Then, the base of security in this type of cryptosystem is determined by the elliptic curve discrete logarithm problem. Concretely, given the parameters a and c, the security is determined by the resolution of the equation $a^b = c$.

The basic operation in ECC is the scalar point multiplication, which can be calculated by the repetitive computation of point additions and doublings. Although point multiplications utilized in the operations of encrypting and signing are less expensive in computational terms than the primitive operations of RSA (i.e. exponentiations), they continue to be resource intensive. For this purpose, there exist several optimizations, such as the use of projective coordinates to avoid the inversion operations or the use of Shamir's trick for minimizing the verification time.

As aforementioned, ECC is able to offer the same security as other cryptosystems, like RSA, providing signatures by using Elliptic Curve DSA (ECDSA) and key establishment by using Elliptic Curve Diffie-Hellman (ECDH). Moreover, ECC works with a smaller key size (160 bits in ECC opposite 1024 bits in RSA), and its basic operation executes in a reasonable time. These properties are very appreciated by the research community in order to try their integration in sensor nodes.

2.2.2. Other cryptographic approaches

Other cryptographic approaches that could be suitable for constrained architectures are Rabin's scheme [Rabin1979], NtruEncrypt [NTRU1998] and *MQ*-schemes [MQ2005]. Overall, their execution time is quite optimal, but their memory requirements in terms of the key size are very heavy.

In 1979, Michael Rabin proposed a high-speed scheme based on the factorization problem of large prime numbers, known as Rabin's scheme. Therefore, its security is determined in the same way as for RSA. This primitive is characterized by the speed of its encryption and signature verification operation, which are built by a simple squaring operation. However, the function of Rabin generates three results: two false and one correct, and the objective is to find the correct one, hence the identification process implicates extra complexity to system. Also, the size of a single signature is quite large (512 bits).

Other asymmetric approach is NtruEncrypt and its corresponding signature scheme NtruSign, which were created by Joseph H. Silverman, Jeffrey Hoffstein, Jill Pipher and Daniel Lieman in 1996. These cryptographic primitives are basically based on a polynomial ring R, and its strength is based on the hardness of solving the Closest Vector Problem (CVP) and the Shortest Vector Problem (SVP). Due to its simple primitive operations for encryption and signing, just mere polynomial multiplications, NtruEncrypt claims to be faster than other asymmetric encryption schemes.

Finally, the most recent asymmetric approach is the multivariate public-key cryptosystems, also known as *MQ*-schemes. Their security is determined by the hardness of solving $w = V^{-1}(z) = (\omega_1,..., \omega_n) \in K^n$, given a quadratic polynomial map $V = (\gamma_1,..., \gamma_m) : K^n \rightarrow K^m$. A *MQ*-scheme is quite fast in its cryptographic operations, such as the verification of a signature. However, there is a significative storage cost for restricted environments due to the size of the keys in RAM. Concretely, the private key needs 879 bytes of storage and the public key needs 8680 bytes. Obviously, this cryptosystem can only be supported by sensor nodes with high memory capabilities.

2.3. Hash Functions

Cryptographic hash functions or hash primitives are utilized in order to compress a set of data of variable length into a set of bits of fixed length. The result is a "digital fingerprint" of the data, identified as a hash value. A cryptographic hash function must satisfy two properties: i) given a hash value *h*, it should be hard to find a message m such that hash(m) = *h*, and ii) it should be hard to find two different messages m_1 and m_2 such that $\text{hash}(m_1) = \text{hash}(m_2)$.

This kind of hash primitives are usually used to build other cryptographic primitives. For example, the Message Authentication Code (MAC) primitive provides authenticity and integrity in the messages, either by using the CBC mode of operation in a process named CBC-MAC, or by taking advantage of the existence of hash primitives. An example of hash function is SHA-1 algorithm [SHA1_2005], which takes 512 bits and returns 160 bits, and operates with additions, rotations, XOR, AND, OR and NOT. Nevertheless, the complexity required for finding a collision is only 2^{63}, hence it is recommended to use other stronger algorithms such as SHA-256. The algorithm SHA-256 takes 512 bits and returns 256 bits. Other more optimized hash function is RIPEMD-160 [RIPE1996], with an input of 512 bits and an output of 160 bits, which uses rotations, permutations, shifts, and others as internal operations.

Although all the hash functions mentioned in this section are fast, they are only suitable for microprocessors of 32 bits, because of their internal word size. Therefore, they could not be optimally supported by microcontrollers of 8 and 16 bits, whereas more powerful 32-bit microcontrollers like the PXA271 and ARM920T can optimally manage them.

3. Software Implementation Platforms

3.1. Symmetric Key Cryptography and Hash Functions

The task of protecting small, highly-constrained devices is not an easy task, since their microcontrollers are very limited in terms of computational power, memory capabilities, and so on. However, they should be able to execute several instances of Symmetric Key Cryptography (SKC) and Hash primitives without provoking a penalty in communication. That is, the time dedicated in the execution of such software primitives must be under a considerable period of time, because on the contrary it could cause a delay in the sending of packets over the radio. For avoiding this serious problem, the encryption time must be less than the *byte time*, which represents the time required for sending a single byte of data. For example, for a CC1000 transceiver with a bandwidth of 19.2kbps, the byte time is approximately 420μs, and for the 802.15.4-based CC2420 with a bandwidth of 250kps, the byte time is closer to 32μs.

One of the first studies on the encryption overhead of SKC and Hash primitives in constrained microcontrollers was made by Ganesan et. al. [Ganesan2003] in 2003. In their software implementation, they discovered that the time required for encrypting a single plaintext was much less than the time need for executing hash primitives. Nonetheless, the average code size of a single primitive is less than 4000 bytes of ROM. For example, the encryption time of a plaintext with the stream cipher RC4 was 6μs per byte, with RC5 26μs and with IDEA 21μs, whereas for digesting a single byte with the SHA-1 algorithm was necessary to wait 122μs.

Later, in 2004 SKC primitives are introduced by means of a new cryptographic package in Mica and Mica2 nodes over the “de-facto” standard OS for sensor nodes, TinyOS 1.x. This package known as TinySec [Karlof2004] provided the Skipjack primitive, which needed 48μs for encrypting a byte, and the RC5 primitive, which needed 33μs for encrypting a byte. RC5 was considerably better in encryption overhead than Skipjack. However, it did not include any hash primitives, so the integrity process was achieved through use of CBC-MAC, which generates the message authentication code using the SKC primitives.

In 2006, Wei Law et. al. [Wei2006] and Jun Choi and Song [Junchoi2006] carried out a deep analysis on the encryption overhead of other instances of SKC primitives. Their studies demonstrated that an optimized Skipjack improved in both encryption overhead (25μs) per byte and in memory overhead (2600 bytes of ROM) than the rest of algorithms (an average of 8000 bytes). Nonetheless, RC4 achieved in terms of memory overhead 428 bytes, thus it was much better than an optimized Skipjack for extremely constrained devices.

3.2. Public key Cryptography

Until 2004, almost all security studies were focused on symmetric key cryptography (SKC), since the possibility of using public key cryptography (PKC) in highly-constraint context (for example, Wireless Sensor Network (WSN)) was considered, even labelled, as "not possible". However, Gura et. al.'s studies in [Gura2004] opened a door of possibilities for the applicability of PKC in sensor networks. They ensured that Elliptic Curve Cryptography (ECC) was an appropriate technique for implementing PKC in high-constrained context, since it offers a good functionality both in computation and memory storage, minimizing energetic costs and functional complexity. The cause of this reduction is due to its small key sizes and its faster computation.

Moreover, Gura et. al.'s studies encouraged to the research community to advance more on this topic. In fact, Malan et. al. [Malan2004] presented in the same year the first library with ECC primitives over the field F_2^p, known as EccM 2.0. It worked with a key size of 163 bit, achieving an average execution time of 34 seconds in its operations (point multiplication and secret key generation). However, 34 seconds was not fast enough for allowing the creation of secure services based on PKC. For that reason, several studies started to optimize the ECC capabilities. One of the firsts ECC optimizations was presented by Gura et. al. in [Gura2004]. They ensured that performance of ECC could be improved if it used projective coordinates instead of affine coordinates to reduce the number of expensive operations, such as the inversion. Other of their suggestions was to work over a field F_p instead of F_2^p. Such optimizations were taken into account by Batina et. al. [Batina2006] for their work on their hardware implementation of ECC (cf. Section 5.1.1).

Actually, there are several implementations using a few of the ECC optimizations previously described. For instance, Liu and Ning implemented TinyECC [Liu2007], and Wang and Li implemented WMECC [Wang2006]. Concretely, TinyECC has a set of implementations in various elliptic curve domains (secp128r1, secp128r2, secp160k1, secp160r1, secp160r2, secp192k1, and secp192r1) according to the Standards for Efficient Cryptography Group [SECG2007], while WMECC only has the implementation on the secp160r1 elliptic curve domain. All of them work under an event-oriented and component-based Operating System named TinyOS. They were codified with a component-oriented language created specifically for sensor networks, named nesC (Network Embedded System C).

Although, TinyECC and WMECC were very different both in design and in code, they were relatively similar because they have in common some optimization techniques. Firstly, for getting optimization in the modular reduction in multiplications and square, they made use of pseudo-Mersenne primes (being p a field prime, whose value is $2^n - c$). Nonetheless, TinyECC is a little better in performance than WMECC on this aspect. It can compute modular multiplication of large integers by using Hybrid Multiplication. Secondly, both implementations used projective coordinates, but TinyECC applied Jacobian representation (X,Y,Z) while WMECC utilized a combination of representation between modified Jacobian coordinates (X,Y,Z,aZ^4) and Affine coordinates (X,Y).

Lastly, both implementations utilized two methods, Shamir's trick and sliding window method, for improving the performance of scalar point multiplications. The purpose of the Shamir's trick is to execute in parallel several point multiplications, and this way it is possible to minimize the signature verification time. The sliding window

method, grouping the scalars in *s*-bit clusters, allows the reduction of the running time by pre-computing point multiplications. It is important to notice that the pre-computation phase in TinyECC is much more expensive than WMECC, since it must initialize the ECDSA system. On the contrary, WMECC uses a small window for both methods (*s=4* for sliding window method and *w=1* for Shamir's trick).

In order to analyze discrepancies between both implementations, it is necessary to compute several rounds of them for determining which is their overhead and complexity cost. Table [X] shows the results obtained by the execution of TinyECC and WMECC in the elliptic curve domain secp160r1. This table summarizes the results obtained after executing 20 rounds of the ECC primitives over the Micaz and TMote Sky nodes. The first rows of the table corresponding to the ROM and RAM size reserved for the primitives and the test program, and the rest of rows corresponding the time spent in the execution of the primitive. The difference between both nodes can be partially explained by their architecture: the microcontroller of Micaz does not spend time in computing unsigned multiplication, but the microcontroller of TMote Sky (MSP430) dedicated several cycles to them.

Comparing both implementations, the WMECC package yields a better performance than the TinyECC package: It has no initialization time for the ECDSA, and the signature and verification times are slightly faster. However, its memory footprint is simply prohibitive, and it would be impossible to develop any kind of application for a TMote Sky. Fortunately, the major penalty of the WMECC code is the SHA-1 function, and the TinyECC implementation is precisely characterized by an optimized SHA-1 code. Thanks to the component capabilities of the TinyOS operating system, it is possible to use the SHA-1 component of TinyECC in the WMECC code, resulting in a new version of the WMECC package that is significantly better. Using this new implementation, it can be perfectly possible to create PKC-based applications on a constrained node such as the TMote Sky.

Summing up, the results obtained for TinyECC and WMECC achieved the expected optimizations, reducing the running time for signing or verifying a signature to not more than 2 seconds. This indicates that there was an important improvement with respect to the results obtained by Malan et. al.'s work. As a result, it has been shown that it is possible to use software implementations of PKC in constrained sensor nodes. On the other hand, it would be better to reduce even more the execution time of these primitives for allowing the creation of efficient secure services based on asymmetric cryptography.

4. VLSI Integrations for Cryptographic Primitives

The applications increasing demand for computation power, and the power reduction requirements for portable devices, force researchers to consider that general-purpose processors are no longer an efficient solution for mobile systems. So, new hardware approaches are needed in order to implement some computational heavy and power consuming functions in order to meet the current network speed requirements. Such approaches are:

Recent Application-Specific Integrated Circuits (ASIC) technology was the solution that created better opportunities for implementing real-time and more sophisticated systems. ASIC devices guarantee better performance, with enough small dedicated size. The reliability reaches high limits and the performance reaches high

values. The implementations in these modules are characterized of tighter design stability, than any other type of hardware devices. ASICs include several custom and semi-custom hardware designs such as: Programmable Logic Devices (PLD), Gate Arrays (GA) and Standard Cells (SC). In our case ASICs can be described as follows: Custom-designed hardware, specially tailored to one particular encryption process. They require a significant initial investment for design and testing. If such a device is not produced in mass quantities, it is not economical for the market. ASICs seem to be more suitable for dedicated applications and not for an extended purposed encryption system.

Besides the original software implementation platforms and the ASICs devices integrations there is a middle ground. This area is covered by the Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs). These components provide reconfigurable logic and they are commercially available at low prices. They facilitate hardware/software codesign. Of course, these devices vary in capacity and performance. Programmable logic has several advantages over custom-hardware. It is less time-consuming, for the development and the design phase, than the custom-hardware approach. These devices are more flexible than ASICs. They can be reused for cryptanalysis of many different encryption algorithms with little extra effort.

Another solution to the implementation platform problem is smart cards. This issue has to do more with fit than with performance. In smart cards the RAM requirements are more important, than the clock's frequency. Most commodity smart cards CPU today include 128 - to 256 - bytes of on-board RAM. Each CPU family contains members with more RAM capacity and a correspondingly higher cost. Although some CPUs include enough RAM to hold the keys of the algorithms, it is often not realistic to assume that such a large fraction of RAM can be dedicated solely to the encryption function. In a smart-card operating system, encryption is a minor small part of the system and will not be able to use more than half of the available RAM. Obviously, if an encryption/decryption system does not fit on a certain CPU, with particular configuration of its components, the performance of the system is unrealistic. Even if an algorithm fits onto smart card, the encryption function will not be able to use all of the RAM capacity. For example, an algorithm that needs at about 100-bytes RAM seems to fit in a 128-bytes smart card. Of course this is a theoretical result because there are RAM requirements also for the control procedure that handles the total security process and these requirements increase the need of the memory-limited capacity. It is cleared that the devices of this category are not proper for large encryption systems with special specifications.

In general, hardware implementations have been proved better approaches compared with the software developments, in the terms of throughput, and operating frequency. Of course the covered area resources is a factor that have to be under consideration.

For all the hardware devices there are some common factors that make the implementation of the ciphers in powerful hardware engines a very hard process. The most critical of them is the large number of registers for key storage, which are used by most of the algorithms. As it has been already mentioned above, the security of ciphers is a function of two factors: the strength of the algorithm and the length of the key. While the strength of a cipher is a fixed factor since algorithm's definition, the key length is a parameter that can vary. Ciphers' introducers and cryptographers use large keys for more secure operations. This means larger number of buffers and storage units and increased memory requirements, for hardware integration. This event has finally

cost in the chip's covered area and sometimes in the I/O devices of the system. In order to face this problem RAM blocks are mainly used in hardware implementations. However, in many cases the availability of RAM is restricted. The internal memory capacity of many hardware devices is limited. The use of external RAM reduces the total system performance and increases the system's covered area. All these factors are critical items, which must especially be taken into account by the designers. The application itself defines each time the impact grade of these factors.

5. Integration on Silicon for WSN

5.1. Existing Efforts

5.1.1. Research Solutions

The sensor network paradigm is still in its infancy, and the efforts in both research and industrial environments have been focusing on the design and development of sensor node hardware platforms. On the other hand, there have been few advances in the area of security-specific hardware for sensor networks, and they have not surpassed the stage of prototypes. A possible reason could be the lack of real-world applications beyond test beds. Although it is theoretically clear that in this kind of network environments there is a need for security primitives, mainly due to the public nature of the communication channels, there are no existing applications that could force the creation of specific hardware integrations at an industrial level.

Most of the existing prototypes, which have been designed on research environments, are specialized on providing Public Key Cryptography primitives. These primitives are extremely resource-demanding for highly constrained nodes due to the complexity of their internal operations, compared to the other primitives like Symmetric Key Cryptography. As seen in section 3, the memory footprint of a software implementation of a PKC primitive is usually quite high, consuming more than $^1/_3$ of the memory of a node on certain platforms, and the execution time of a simple operation such as signature verification can last at least 2 seconds. A hardware implementation could greatly improve the usability of these primitives, enabling the creation of advanced authentication services.

Due to its benefits in key size and complexity, Elliptic Curve Cryptography (ECC) has been the primitive of choice in most asymmetric cryptography hardware prototypes as of 2007. The main purpose of these implementations is to reduce the time on executing the ECC core operation, the point multiplication, from the seconds in software implementations to just hundred of milliseconds. A majority of the prototypes are specialized on achieving an acceptable performance using as little gates as possible while functioning with an operational frequency smaller than 8 Mhz, which is the usual operational frequency for a normal sensor node.

In fact, the prototypes that have been specifically created for sensor nodes seek the provision of ECC operations to both normal and "weak" nodes, using an operational frequency as small as 500 Khz. An example is the design by Gaubatz et. al. [Gaubatz2005], where operations are performed over $\mathbb{F}_{p_{100}}$, occupying a chip area equivalent to 18720 gates in 0.13μm CMOS technology, and consuming just under 133μA in signing and encrypting messages. ECDSA signatures and ECMQV decryptions are performed at around 410ms, and ECDSA verifications and ECMQV

encryptions are performed at around 820ms. As for the specific optimizations in this design, they efficiently implement modular reduction, achieve a fast inversion algorithm, and implement all arithmetic primitives in a bitserial fashion.

A later prototype, created by Batina et. al. [Batina2006], improves the results obtained in the previous design using the same operational frequency, 500 Khz. Their Elliptic Curve Processor (ECP) included a Modular Arithmetic Logic Unit (MALU) capable of computing modular additions and multiplications, using the same cells for both purposes without having a full-length array of multiplexors. Moreover, squaring is considered a special case of multiplication, and inversion is avoided mostly by use of projective coordinates. The results are promising: using 8104 gates (12000 gates including RAM) in 0.13μm CMOS technology, one point multiplication over $F_{2^{131}}$ is performed in only 115ms, consuming less than 10μA.

On the other hand, not all ECC prototypes consider the operational frequency as a factor that should limit the design of the hardware. The existence of "heavy-duty" microcontrollers used on certain sensor node platforms, like the PIC18F6720, PXA271, and the ARM920T, justify the creation of specialized prototypes that operated at a higher frequency. Wolkerstorfer et. al. [Wolker05], as well as Kumar and Paar [Kumar2006], designed integrated chips that are able to provide an excellent performance for signing a message using ECDSA. In the case of Wolkerstorfer et. al., the chip, which has an area of 23000 gates implemented in 0.35μm CMOS technology, is able to execute a single point multiplication operation over $F_{2^{191}}$ in only 6.67ms. This design has an operational frequency of 68.5 Mhz, thus it can only be used by "heavy duty" sensor nodes. In contrast, the chip created by Kumar and Paar is slower, providing a point multiplication over $F_{2^{131}}$ in 18ms, but the number of gates is smaller, almost 12000. Its operational frequency is also smaller, around 13 Mhz, so it can be afforded by "heavy duty" and normal microcontrollers alike.

There are also other hardware prototypes that provide other asymmetric cryptography primitives, such as NTRU, Rabin, and Multivariate (cf. Section 2). The primary disadvantage of these primitives, compared with ECC, is their key and signature size. The Multivariate cryptosystems uses a private key of 879 bytes, and a public key of 8680 bytes. Also, NTRU provides a signature of 1169 bits, while Rabin generates a signature size of 512 bits. As a result, there is an energy overhead associated to the transmission of messages over the communication channel and a cost associated to the storage of keys and signatures. All of these are critical factors in the design of protocols for sensor networks. Nevertheless, these prototypes have certain advantages that can make them useful on certain scenarios.

The NTRUEncrypt prototype, developed by Gaubatz et. al. [Gaubatz2005], employs a very small number of gates, around 3000, and consumes only 6.6μA in an operational frequency of 500kHz. Encryption and verification operations are performed at around 58ms, decryptions are calculated in almost 117ms, and messages are signed in almost 234ms. As a result, most operations done on a NTRU chip are faster than any of the ECC prototypes. Also, their chip area is the smallest of all the PKC primitives. The major benefit of the Rabin prototype, which was developed by the same authors [Gaubatz2005], is the encryption and verification operations: at the same operational frequency of the NTRU prototype, 500 Khz, a message can be encrypted or verified in just 2.88ms. Still, the downside comes from the decryption and signature operations,

which are calculated in 1.089s, and from the average power consumption, which is near 49μA.

Finally, the Multivariate prototype, made by Yang et. al. [Yang2006], only provided signature generation, since it was designed for being included inside RFID tags. Nevertheless, since the target device is even more constrained than a sensor node, the benefits are higher in comparison with the other prototypes: using 17000 gates in 0.25μm CMOS technology, with an energy consumption of 25μA in an operational frequency of just 100kHz, it is possible to sign a message in 44ms. This speed is achieved by using the most optimal primitive, the circulant (LPSQR) form of enTTS(20,28), in conjunction with some optimizations like the substitution of the Lanczos' method used in the internal operations with the symmetric Gaussian elimination.

5.1.2. Industrial Efforts

Aside from the previously mentioned prototypes developed on research environments, there are, at present, very few solutions that are able to provide hardware support for cryptographic primitives at a mass scale. The most effective solution in terms of using the primitives would be to define a standard for including this kind of hardware support in the microcontrollers. However, there are no existent solutions in this area, aside from the definition of processor-specific instructions sets like MMX that could be used to improve the execution of the security primitives. On the other hand, some network standards used for wireless communication in sensor networks demand the existence of support for such primitives.

As a result, on current technology, the radio transceiver chip located on sensor nodes can be able to provide hardware support for cryptographic primitives. The reason is simple: usually, all communications have to be protected with symmetric key cryptography primitives in order to assure the confidentiality, integrity and authenticity of the packets that traverse the network. Since all network packets must traverse the radio transceiver, this is the ideal place where the hardware support could be located. Not all radio transceivers incorporate this type of support, though: only a certain group of standards requires the support of the primitives on silicon. One of those standards, widely used on sensor networks, is the IEEE 802.15.4 standard [IEEE802.15.4].

All transceiver chips that implement the 802.15.4 standard can offer a suite of AES-based symmetric key cryptography operations for assuring the confidentiality and integrity of the network packets. The available suites are AES-CTR, AES-CBC-MAC, and AES-CCM. The AES-CTR suite only provides confidentiality by using AES in Counter Mode, thus behaving as a stream cipher. AES-CBC-MAC ensures data integrity through Message Authentication Codes using AES in Cipher Block Chaining Mode (CBC-MAC), where the size of the MAC can be 32, 64 or 128 bits long. Finally, AES-CCM provides both confidentiality and integrity, by applying integrity protection using AES-CBC-MAC and then encrypting the data using AES-CTR mode.

In order to use these security suites, the application must create an ACL (access control list) entry consisting of a source address, a destination address, a suite, and the secret key of 128 bits to be used between the source and the destination. Once the standard-compliant radio transceiver receives a packet with security enabled, it looks up an entry in the ACL, and applies the security suite if a match is found. Note that, although there can be up to 255 ACL entries, the standard does not specify a minimum

size. Moreover, the only mandatory suite that the transceiver must offer by default is AES-CCM with a 64-bit MAC.

Unfortunately, the 802.15.4 standard, and all the chips that implement it, is not exempt of flaws [Sastry2004]. One of the security suites, AES-CTR, is deeply flawed, since it does not properly support replay detection and it is possible to launch Denial of Service (DoS) attacks sending a single forged packet. Also, the acknowledgement packets are not protected by a MAC, thus it is possible to forge them. Other minor problems include deleting the ACL entries when entering low power mode. As a result, chip manufacturers and application designers must take into account the inherent problems of the standard.

As an example, the most common 802.15.4 chip integrated in sensor nodes as of 2007, the Chipcon CC2420, implements all the recommended security suites (including the flawed AES-CTR), and also provides an additional simple encryption mode where a 128 bit plaintext is encrypted with a 128 bit secret key. The hardware implementation of these suites is quite fast: the CC2420 is able to apply AES-CCM to a single packet in 222μs, and the simple encryption mode works in only 14μs. However, its ACL has only two entries, thus the chip only provides "out-of-the-box" support for two neighbors. Therefore, the application developer has to provide a workaround for taking advantage of the hardware capabilities. An example would be to implement CBC-MAC in software by using the simple encryption mode [Sun2006].

Regarding MMX, this SIMD instruction set was designed by Intel and introduced in 1997 as part of the Pentium MMX microprocessors. Nowadays, this instruction set is also part of the PXA27X family of microprocessors as "Wireless MMX". This extension provides 16 data registers of 64 bits and 8 control registers of 32 bits, and is able to perform two 32-bit, four 16-bit, or eight 8-bit operations in a single instruction. These operations range from simple mathematical and logical operations like rotating, adding and multiplying to other specific operations such as calculating the vector maximum/minimum.

These extended instruction sets were not designed with cryptographic primitives in mind, since they are focused on providing support for multimedia-based tasks such as video compression, 2D and 3D graphics, and image processing. Nonetheless, their parallelism capabilities can help to efficiently implement block ciphers such as AES or Twofish [Aoki2000], which use simple operations such as XOR over a set of 8-bit registers. In the case of AES, it has a large internal parallelism, thus there is potential for optimization. On the other hand, the Pseudo-Hadamard Transformation in Twofish somewhat complicates the parallelization process, although it is still possible to optimize certain parts of the algorithm.

Not only block ciphers can be optimized: hash functions can be also improved by the use of MMX instructions. For example, Nakajima and Matsui [Nakajima2002] explored the optimization of MD5, the RIPEMD family, the SHA family and Whirlpool, by processing message blocks in parallel using MMX registers for 32-bit oriented hash functions. This type of parallelization can be possible in embedded systems, but it has not been researched yet. There were also some optimizations for 64-bit oriented hash functions like SHA-512 and Whirlpool, but these optimizations cannot be applied to the "Wireless MMX" instruction set, because it cannot parallelize operations over 64-bit fields.

6. Alternative Solutions and Optimizations

The problem of hardware implementation is a function of two different factors: cryptographic algorithms architectures and the efficient integration of them [Sklavos2007]. All forums and organizations in the wireless communication world have specified security layers/systems and have published the selected ciphers that these systems are based on. In order high-level security to be ensured, three schemes of encryption must be applied in a communication handshake: Bulk Encryption, Message Authentication and Data Integrity. The wireless protocols have defined alternative ciphers in each type of the above schemes. Large encryption systems have been mainly implemented only in software. On the other hand, hardware devices have been used for the implementation of encryption algorithms, one per device, and for security systems with less complexity.

The previous years, the hardware integration approach to the issue of security implementation was the ASICs solution. Implementations on these modules achieve high-speed performance and have been proved confidant solutions. Although in the case of wireless protocols, this implementation aspect is proved unfeasible. The hardware integration of a set of ciphers, that a protocol defines, results in a very large circuit. Encryption algorithms implementations, that have been published until now in ASICs, cover an area of 40-60 mm2 each. For example, the WAP cipher set integration (eight algorithms in total), in one or more ASICs needs an area about 400-480 mm2, plus the space needed for the total control unit and routing allocated area. Such an ASIC device is very difficult to be designed and manufactured. Of course the cost of the chip is increased dramatically in this case.

Nowadays a flexible encryption system, which would support the operation of a set of ciphers integrated in the same module, can be implemented with hardware and software cooperation. This type of cooperation could be achieved efficiently by the principles of reconfigurable computing. A proposed solution is the design of a reconfigurable cryptographic system, which will support at least bulk and message authentication encryption. Reconfigurable computers are those machines that use the reconfigurable aspects of Reconfigurable Processing Units (RPUs) and FPGAs to implement a system/algorithm. The algorithms are partitioned into a sequence of hardware implementable objects (hardware objects). This type of objects represents the serial behavior of the algorithm and can be executed sequentially. The design technique based on hardware objects offers to the developer/designer a logic-on-demand-capability that is based on the reconfigurable computing. The appropriate software, in order to suit the application at hand, modifies the architecture of these computing platforms. This means that within the application program a software routine has been written to download a digital circuit (chip design) directly into the RPU. The main idea of these designs is the alternation among static and dynamic performance of the system.

Static circuitry is the part of the operation performance that remains in action between the different configurations of the hardware device. The part of static circuitry has to be in the maximum level of the design and special care must be taken for its optimization. General-purpose blocks, such as adders, belong to this part of performance. Another example of the static parts is the storage units. These are the parts of each system that never are never changed during different operations. Always they maintain the characteristics that the initialization process has set. On the other hand, there is the dynamic circuitry. With this term, we mean the parts of the system that change during configuration. These blocks must be minimized in order to increase

the system performance. If there are not basic common parts between the selected algorithms, the dynamic circuitry reaches high values and this is not flexible for the system performance. Dynamic circuitry increases the needs for the system's allocated resources, and on the other hand decreases its attribution. It has to be cleared that the selection ciphers with similar philosophy of functionality, finally would be proved as a critical factor of the hardware implementation.

Although, in most of the case encryption algorithms are based in different design philosophy and their functionality is completely different. In order to achieve this, the designer of a powerful security system has to choose one flexible algorithm for bulk encryption with the ability to operate as a hash function to support data integrity purposes, for example HMAC. The addition of some extra parameters in the algorithm's architecture is necessary for the efficient operation of the two encryption modes. In this way, the needs of the system resources are reduced. At the same time, we have to avoid ciphers with heavy arithmetic functions such as multiplication and modulo processes. Such arithmetic functions are proven quite difficult to be implemented in hardware devices and mainly they have no common parts with other ones.

The implementation of a security system with some common basic parts, which can be used for the implementation of ciphers' common functions, seems to be the more sophisticated alternative solution for a large encryption engine. With the term basic parts we mean "heavy" algebraic or logical components of the algorithms' architectures. In most of the cases it is difficult to implement these parts in a hardware device, with high-speed performance and minimized covered area. An example is the multiplication modulo that IDEA cipher needs. Reconfigurable computing method is proved efficient enough to solve the implementation problem of encryption engines and is suitable for the different types of architectures that ciphers' have.

The latest years, implementations on the smart card devices have been very attractive for the hardware designers. Compared with the other hardware devices like ASICs and FPGAs, smart cards have limited computing power and minimized storage capacity. Therefore, security applications which allocate a huge amount of storage or which require an extensive computation power might cause conflicts.

The persistent storage of a smart card is limited to a few kilobytes today, which prevents it from storing larger items on the card. This can be circumvented if the smart card delegates the storage of the item in an external environment. The smart card receives and processes the transmitted data. It encrypts them and saves them to the sender/receiver's device external storage units (RAM, registers of general use). Later, when these data are needed again, the smart card can request it from these storage units. By using this described method the smart card internal storage requirements can be reduced significantly. However, we have to take care that we do not create another bottleneck: the communication speed of the smart card is not very high and so we should have to handle the transmission of the same data back and forth, with special care.

Another limitation of smart cards is the small processing power. The appropriate data modifications, due to encryption/decryption, may possibly exceed the computing power of a smart card. In this case it will take unacceptably long to finish the appropriate data transformation. Thus, it is important to minimize the amount of computation power, which the smart card has to pay for the requested tasks. For such applications, it is better the design to be kept as simple as possible. The requested task can be divided in smaller parts with no hard processing specifications. The requested

round keys for encryption/decryption can be generated in the initialization procedure and not at the same time with the encryption round transformation (on the fly key generation). In this way, we avoid to spend extra processing power for the key expansion unit during encryption/decryption. The same methodology can be followed for the appropriate specified constants generation.

7. Future Directions

The needs for personal network communications systems are growing rapidly. Coupled to this increase is the telecommunication-related crime. In wireless networks, an invader with suitable receiver can intercept the transfer data. Security is a primary requirement of all wireless cryptographic protocols. Cryptographic algorithms are meant to provide secure communications applications. However, if the system is not designed properly, it may fail. Although there are many well know ciphers, with different specifications and characteristics, the security of some of them is under consideration. Many works, from different research groups, have been published in technical literature in which cryptanalysis methods have been applied in order to find out any existing black holes in the security strength of the encryption algorithms. From many points of view, such attempts offer valuable knowledge in the growth and the improvement of cryptography. Encryption algorithms have to perform efficiently in a variety of current and future applications, doing different encryption tasks. The algorithms should be used to encrypt streaming audio and video data in real time. They would have to work correctly in 32 and 64-bit CPUs. Many of today's applications run with smart cards based on 8-bit CPUs, such as burglar alarms, pay-TV and general other applications. All hardware implementations have to be efficient, with less allocated area resources. This means simplicity in algorithm's architectures with enough "clever" data transformation components. A wireless protocol implementation demands low power devices and fast computation components, which imply that the number and complexity of the encryption operations should be kept as simple as possible. A basic transformation in the operation of the encryption algorithms is needed, including modifications in the data blocks and key sizes.

The ciphers of the future have to be key agile. Many applications need a small amount of text to be encrypted with keys that are frequently changed. Many well know applications, like IPsec, use this way of algorithm's operation. Although the most widely used mode of operation is encryption with the same key for all the amount of transport data, the previous mode is also very useful for future applications. Ciphers that require subkeys precomputation have a lower key agility due to the computation time, and they also require extra RAM to hold the subkeys. This RAM requirement does not exist in the implementations of algorithms, which compute their keys during the encryption/decryption operation. Cellular phones technology demands hard specifications of the cryptography science. Ciphers have to be compatible with wireless devices restricted standards in hardware resources. New mobile phones will have proper encryption part built in them. In these devices, there is not enough room for a large integrated security layer. A solution in order to decrease the required hardware resources is to use the ciphers for both bulk encryption and data integrity, with a simple change of their operating mode. All the above, make the effort of designing new security algorithms for wireless applications a real hard process. Both AES and SHA-2

standards are very good steps for the design of communications security schemes, for the next decades.

The technology growth gives many promises for the security future. If the strength of the applied cryptography that is used in wireless industry were increased enough, the protocol security would be efficient to withstand the attempts of the attackers. Today many ciphers can support the defense of the communications links against external invaders. On the other hand, the implementation of these is a hard process and sometimes cannot meet the wireless network requirements. This is due to the fact that that today's used ciphers have been designed some years ago and for general cryptography reasons. They are not specialized for the wireless communications. Security improvement needs strong, flexible encryption algorithms with efficient performance. New encryption algorithms must be designed for applications of any kind. On the other hand, every algorithm should be demonstrated in software before committing to hardware.

8. References

[AES2002] J. Daemen, V. Rijmen. "The Design of Rijndael". Springer, ISBN 3-540-42580-2, 2002.

[Aoki2000] K.Aoki and H. Lipmaa. "Fast implementations of the AES candidates". Proceedings of 3rd AES conference, New York (USA), April 2000.

[Batina2006] L. Batina, N. Mentens, K. Sakiyama, B. Preneel, I. Verbauwhede. "Low-Cost Elliptic Curve Cryptography for Wireless Sensor Networks". In Proceedings of the 3rd European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS 2006), Hamburg (Germany), September 2006.

[ECC2000] I. Blake, G. Seroussi, N. P. Smart. "Elliptic Curves in Cryptography". Cambridge University Press, ISBN 0-521-65374-6, 2000.

[Fluh2007] S. Fluhrer, I. Mantin, A. Shamir. "Weaknesses in the Key Scheduling Algorithm of RC4". In proceedings of the 8th Annual Workshop on Selected Areas in Cryptography (SAC 2001), Toronto (Canada), August 2001.

[Ganesan2003] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, M. Sichitiu. "Analyzing and Modeling Encryption Overhead for Sensor Network Nodes". In Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA 2003), San Diego (USA), September 2003.

[Gaubatz2005] G. Gaubatz, J.-P. Kaps, E. Öztürk, B. Sunar. "State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks". In Proceedings of the 2nd IEEE International Workshop on Pervasive Computing and Communication Security (PerSec 2005), Hawaii (USA), March 2005.

[Gura2004] N. Gura, A. Patel, A. Wander. "Comparing elliptic curve cryptography and RSA on 8-bit CPUs". In Proceedings of the 2004 Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004), Cambridge (USA), August 2004.

[IEEE802.15.4] IEEE 802.15 WPAN™ Task Group 4 (TG4). http://www.ieee802.org/15/pub/TG4.html

[Junchoi2006] K. Jun Choi, J.-I. Song. "Investigation of Feasible Cryptographic Algorithms for Wireless Sensor Network". Proceedings of the 8th International Conference on Advanced Communication Technology (ICACT 2006). Phoenix Park (Korea), February 2006.

[Karlof2004] C. Karlof, N. Sastry, D. Wagner. “TinySec: a link layer security architecture for wireless sensor networks”. Proceedings of 2nd International Conference on Embedded Networked Sensor Systems (SenSys 2004), Baltimore (USA), November 2004.

[Kumar2006] S. Kumar, C. Paar. “Are standards compliant elliptic curve cryptosystems feasible on RFID?”. In Proceedings of Workshop on RFID Security, Graz (Austria), July 2006.

[Liu2007] A. Liu, P. Kampanakis, P. Ning. “TinyECC: Elliptic Curve Cryptography for Sensor Networks (Version 0.3)”. http://discovery.csc.ncsu.edu/software/TinyECC/, February 2007.

[Malan2004] D. J. Malan, M. Welsh, M. D. Smith. “A Public-key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography”. In Proceedings of 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004), Santa Clara (USA), October 2004.

[MQ2005] C. Wolf, B. Preneel. “Taxonomy of Public Key Schemes Based on the Problem of Multivariate Quadratic Equations”. Cryptology ePrint Archive, Report 2005/077.

[Nakajima2002] Junko Nakajima, Mitsuru Matsui. “Performance Analysis and Parallel Implementation of Dedicated Hash Functions”. Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology (Eurocrypt 2002), Amsterdam (Holland), April-May 2002.

[NTRU1998] J. Hoffstein, J. Pipher, J. H. Silverman. “NTRU: a Ring based Public Key Cryptosystem”. In proceedings of the 3rd Algorithmic Number Theory Symposium (ANTS 1998), Portland (USA), June 1998.

[Rabin1979] M. O. Rabin. “Digitalized Signatures and Public Key Functions as Intractable as Factorization”. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology (1979).

[RC4_1996] B. Schneier. “Applied Cryptography, 2nd edition”. Wiley, ISBN 0-471-12845-7, 1996.

[RC5_1994] R. L. Rivest. “The RC5 Encryption Algorithm”. Proceedings of the 2nd International Workshop on Fast Software Encryption (FSE 1994), Leuven (Belgium), December 1994.

[RC6_1998] R. L. Rivest, M. J. B. Robshaw, R. Sidney, Y. L. Yin. “The RC6 Block Cipher, v1.1”. August 1998, http://theory.lcs.mit.edu/~rivest/

[RIPE1996] H. Dobbertin, A. Bosselaers, B. Preneel. “RIPEMD-160, a strengthened version of RIPEMD”. Proceedings of the 3rd International Workshop on Fast Software Encryption (FSE 1996), Cambridge (UK), February 1996.

[RSA1978] R. Rivest, A. Shamir, L. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. Communications of the ACM, vol. 21, no. 2, pp. 120–126, 1978.

[Sastry2004] N. Sastry, D. Wagner. “Security considerations for IEEE 802.15.4 networks”. In Proceedings of 2004 ACM Workshop on Wireless security (Wise 2004), Philadelphia (USA), October 2004.

[SECG2007] SECG - Standards for Efficient Cryptography Group. http://www.secg.org/

[SHA1_2005] X. Wang, A. Yao, F. Yao. “New Collision search for SHA-1”. Rump Session of the 25th Annual International Cryptology Conference (CRYPTO 2005), Santa Barbara (USA), August 2005.

[Skipjack1998] NIST-CSRC. “SKIPJACK and KEA Algorithm Specifications, version 2”. 29 May 1998, http://csrc.nist.gov/CryptoToolkit/

[Sklavos2007] N. Sklavos, X. Zhang, Handbook of Wireless Security: From Specifications to Implementations, CRC-Press, A Taylor and Francis Group, ISBN: 084938771X, 2007.

[Sun2006] K. Sun, P. Ning, C. Wang, A. Liu, Y. Zhou. “TinySeRSync: Secure and Resilient Time Synchronization in Wireless Sensor Networks”. In Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS'06), Alexandria (USA), November 2006.

[Twofish1999] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson. “The Twofish Encryption Algorithm: A 128-Bit Block Cipher”. Wiley, ISBN 0-471-35381-7, 1999.

[Wang2006] H. Wang, Q. Li. “Efficient Implementation of Public Key Cryptosystems on MICAz and TelosB Motes”. Technical Report WM-CS-2006-07, College of William & Mary, October 2006.

[Wei2006] Y. W. Law, J. Doumen, P. Hartel. “Survey and Benchmark of Block Ciphers for Wireless Sensor Networks”. ACM Transactions on Sensor Networks, vol. 2, no. 1, pp 65-93, February 2006.

[Wolker2005] J. Wolkerstorfer. “Scaling ECC Hardware to a Minimum”. In ECRYPT workshop - Cryptographic Advances in Secure Hardware - CRASH 2005. Leuven (Belgium), September 2005. Invited Talk.

[Yang2006] B.-Y. Yang, C.-M. Cheng, B.-R. Chen, J.-M. Chen. “Implementing Minimized Multivariate Public-Key Cryptosystems on Low-Resource Embedded Systems”. In Proceedings of the 3rd International Conference on Security in Pervasive Computing (SPC 2006), York (UK), April 2006.

This page intentionally left blank

Wireless Sensor Network Security
J. Lopez and J. Zhou (Eds.)
IOS Press, 2008

Author Index

Ács, G.	164	Krontiris, I.	142
Aigner, M.	44	Lopez, J.	1
Aivaloglou, E.	223	Mitrokotsa, A.	251
Alcaraz, C.	291	Preneel, B.	77
Batina, L.	77	Roman, R.	291
Benenson, Z.	22	Roy, S.	204
Buttyán, L.	164	Salajegheh, M.	142
Çamtepe, S.A.	110	Setia, S.	204
Cholewinski, P.M.	22	Seys, S.	77
Dimitriou, T.	142	Sklavos, N.	291
Feldhofer, M.	44	Soroush, H.	142
Freiling, F.C.	22	Tillich, S.	44
Gritzalis, S.	223	Verbauwhede, I.	77
Jajodia, S.	204	Yasinsac, A.	273
Karygiannis, A.	251	Yener, B.	110
Katsikas, S.	223	Zhou, J.	1

This page intentionally left blank