INSENS: Intrusion-tolerant routing for wireless sensor networks

By: Jing Deng, Richard Han, and Shivakant Mishra

J. Deng, R. Han, and S. Mishra, (2006) "INSENS: Intrusion-tolerant routing for wireless sensor networks," *Computer Communications*, vol. 29(2), pp. 216-230.

Made available courtesy of Elsevier: http://www.elsevier.com

Abstract:
This paper describes an INtrusion-tolerant routing protocol for wireless SEnsor NetworkS (INSENS). INSENS securely and efficiently constructs tree-structured routing for wireless sensor networks (WSNs). The key objective of an INSENS network is to tolerate damage caused by an intruder who has compromised deployed sensor nodes and is intent on injecting, modifying, or blocking packets. To limit or localize the damage caused by such an intruder, INSENS incorporates distributed lightweight security mechanisms, including efficient one-way hash chains and nested keyed message authentication codes that defend against wormhole attacks, as well as multipath routing. Adapting to WSN characteristics, the design of INSENS also pushes complexity away from resource-poor sensor nodes towards resource-rich base stations. An enhanced single-phase version of INSENS scales to large networks, integrates bidirectional verification to defend against rushing attacks, accommodates multipath routing to multiple base stations, enables secure joining/leaving, and incorporates a novel pairwise key setup scheme based on transitory global keys that is more resilient than LEAP. Simulation results are presented to demonstrate and assess the tolerance of INSENS to various attacks launched by an adversary. A prototype implementation of INSENS over a network of MICA2 motes is presented to evaluate the cost incurred.
Keywords: Sensor network; Security; Intrusion tolerance; Fault tolerance; Secure routing

Article:
1. Introduction
Wireless sensor networks (WSNs) are rapidly growing in their importance and relevance to both the research community and the public at large. WSNs are comprised of many small and highly resource-constrained sensor nodes that are distributed in an environment to collect sensor data and forward that data to interested users. Applications of WSNs are rapidly emerging and have become increasingly diverse, ranging from habitat monitoring [22] to indoor sensor networks [7], and from battlefield surveillance [4] to seismic monitoring of buildings.

Security is critical for a variety of sensor network applications, such as home security monitoring and military deployments. In these applications, each sensor node is highly vulnerable to many kinds of attacks, both physical and digital, due to each node's cost and energy limitations, wireless communication, and exposed location in the field. As a result, mechanisms to achieve both fault tolerance and intrusion tolerance are necessary for sensor networks.

Although intrusion tolerance has been studied in the context of wired networks [30,6,28,29,32], wireless sensor networks introduce a combination of threats that are not normally faced by wired networks. First, the broadcast nature of the wireless communication medium significantly enhances the capabilities of an adversary to eavesdrop, tamper with transmitted packets, and inject packets to initiate denial-of- service (DOS) attacks. These susceptibilities also apply to wireless LANs such as 802.11 and mobile ad hoc networks. Second, a sensor node is highly resource constrained, with limited energy lifetime, low-power micro-sensors and actuators, slow embedded processors, limited memory, and low-bandwidth radio communication. This limits the ability for sensor nodes to perform computation-intensive public key cryptography such as RSA [27,11], though elliptic curve cryptography offers a promising course of research [23]. Also, the relatively weak defenses of sensor nodes are susceptible to external attacks by much stronger adversaries equipped with more powerful computing and communication equipment. Third and perhaps the most unique, sensor nodes are distributed in the field in-situ and therefore lack physical security that is available to most wired and other forms of wireless networks. As a result, WSNs are highly susceptible to the physical compromise of one or more sensor nodes. Once compromised, the sensor node(s) can be exploited by an intruder to damage the WSN through DOS, jamming, spoofing and several other attacks.

Several salient forms of attacks on WSN routing protocols have been described, including the sinkhole attack [20], the rushing attack [18], the wormhole attack [19], and the Sybil attack [14]. These attacks try to induce incorrect routing information in the network to prevent sensor nodes from sending their data to the correct destination. In a sinkhole attack [20], a malicious node claims that it has the shortest path to a well-known destination, e.g. a base station. If a routing scheme allows sensor nodes to select their routing path based on neighborhood routing information, a sinkhole attack can result in several sensor nodes setting their routing path towards the malicious node. In a rushing attack [18], a malicious node generates a fake ROUTE REQUEST message and employs methods to have that message reach other sensor nodes before the legitimate ROUTE REQUEST message reaches there. This can result in those nodes setting the malicious node as their parent node. In a wormwhole attack [19], two malicious nodes exchange their routing information using a fast and secure channel or tunnel, and then trap or warp the routing paths of their neighbor nodes. In a Sybil attack [14], a malicious node assumes multiple fake identities and then deceives other sensor nodes using those fake identities. For example, a Sybil attack can be used to attack multipath routing or geographic routing [20], and to complicate detection of a misbehaving node [25]. A description of how these attacks can impact a routing scheme is provided in [20].

The architecture of a typical WSN is illustrated in Fig. 1. Sensor nodes organize themselves into a multi-hop wireless network that collects and forwards sensor data to an information sink, usually a base station acting as a gateway to the wired Internet. The communication pattern is relatively simple compared to a traditional wired or an adhoc wireless network. Data transmission is dominated by local communication (one or a small number of hops) between sensor nodes, and multi-hop forwarding between sensor nodes and the base station. Primarily, data is sent from sensor nodes to one or more base stations [20]. In general, the number of base stations in a WSN is significantly less than the number of sensor nodes. Also, the base stations are relatively resource-rich in terms of processing, storage, energy, and communication capabilities. The large number of resource-constrained sensor nodes and the small number of resource-rich base stations collectively form an asymmetric network. While other sensor network architectures and routing protocols for those architectures have been proposed [2], our focus in this paper is on the common asymmetric tree-structured routing architecture illustrated in Fig. 1.

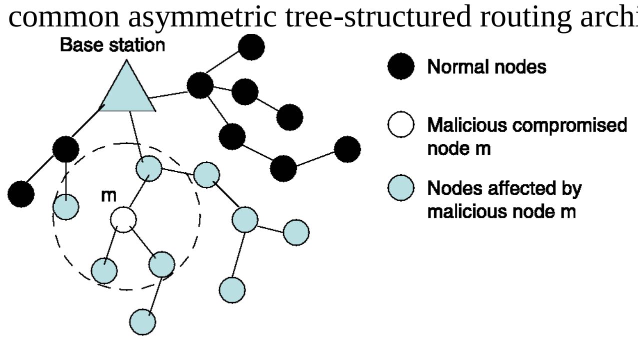

Fig. 1. The typical tree-structured hierarchy of a wireless sensor network. A malicious compromised node *m* can affect immediate neighbors as well as their downstream children. The goal of INSENS is to limit or localize the damage that can be caused by such an intruder.

This paper focuses on the design of a secure and INtrusion-tolerant routing protocol for wireless SEnsor NetworkS (INSENS). INSENS constructs secure and efficient tree-structured routing for WSNs, and is tailored for the asymmetric architecture and resource constraints of WSNs. A key objective of INSENS is to localize the damage caused by an intruder who has compromised deployed sensor nodes. Such an intruder could inject, modify, or block data packets, and in the worst case could bring down the entire sensor network, e.g. by flooding malicious packets. INSENS is therefore designed to tolerate intrusions, limiting the ability of an intruder to cause mischief through a combination of distributed lightweight security mechanisms.

The scope of INSENS is bounded in the following ways. First, INSENS is focused on securing upstream data traffic flow from leaf sensor node sources through the tree-structured routing topology to the base station sink. Arbitrary peer-to-peer communication from any sensor node to any other sensor node is beyond the scope of INSENS, and is not viewed as commonplace. Downstream traffic beyond what is needed to securely set up the upstream routing tree is not a focus of INSENS. Another assumption in INSENS is that sensor nodes can have only limited mobility after their initial deployment, which we believe to be the common case. INSENS's secure topology discovery and set up is designed to be rerun its periodically to update changes in the topology due to faults, and the same process can be applied to support limited mobility. Continuous mobility during and after set up is beyond the scope of INSENS.

The key principles in the design of INSENS are as follows:

- Intrusion tolerance

  1. Limited broadcast using one way hash chains (OHCs): INSENS permits only base stations to initiate flooding of the network, e.g. to set up routing information. Each base station stamps each of its broadcast packets with a one way hash chain number, which we term a one way sequence number. Intruders will be unable to guess the next number in the OHC and will thus be restricted in their ability to flood the network, thereby enhancing intrusion tolerance.

  2. Multipath routing: INSENS employs redundant multi-path routing to enhance intrusion tolerance. To the extent possible, multiple disjoint paths are set up from each sensor node, so that even if an intruder compromises a node or a path, alternate forwarding paths exist. The desire for intrusion tolerance must be balanced against the energy cost of multipath routing. INSENS can be configured to fall back to a secure single-path routing mechanism.

  3. Limited routing updates: Only the base station is allowed to update a node's data routing table. This is accomplished by assuming a secret pairwise key shared only between the base station and a sensor node. This inhibits many attacks directed towards routing information updates in sensor networks, e.g. the sinkhole attack [20].

- Adaptation to resource constraints

  1. Symmetric key cryptography is chosen to implement confidentiality and authentication between the base station and each resource-constrained sensor node.

  2. Complexity is pushed away from resource-poor sensor nodes and into the resource-rich base station, which is chosen as the central point for computation and dissemination of the routing tables.

- Novel mechanisms are introduced to address several specific attacks against sensor network routing. For example, lightweight bidirectional verification is applied to defend against the rushing attack. The nested message authentication code (MAC) is used as a countermeasure against the wormhole attack.

- To accommodate different sizes of sensor networks, a basic three-phase version of INSENS is presented for moderately-sized sensor networks with a single base station, while an enhanced single-phase version of INSENS is presented for large-sized sensor networks with many base stations. Multipath routing to multiple base stations also improves tolerance against base station failures or isolation of a single base station.

The paper is organized as follows. Section 2 describes related work. Section 3 discusses the network model, threat model, and assumed capabilities of sensor nodes. In Section 4, the basic INSENS protocol is described. The basic INSENS protocol is further enhanced to tolerate some more sophisticated attacks in Section 5. The

INSENS protocol has been simulated in NS2 and implemented over a network of Berkeley MICA2 motes. Section 6 describes the implementation experiences, while Section 7 evaluates the protocol based on its effectiveness in tolerating various security attacks and the costs incurred. Section 8 concludes the paper.

2. Related work

Security is a critical issue in sensor network research [3 1, 27,20]. A. Perrig et al. [27] addressed secure communication in resource-constrained sensor networks, introducing two low-level secure building blocks, SNEP and ATESLA. A. Wood and J. Stankovic [3 1] provided a survey of many kinds of denial of service attacks in sensor networks and discussed defense technologies.

C. Karlof and D. Wagner [20] analyzed security flaws of various routing protocols on WSNs, and proposed counter-measures to enhance sensor network routing. INSENS can defend against many attacks that are possible on non-secure routing protocols in sensor networks, e.g., the spoofed routing information attack, selective forwarding, sinkhole attacks, wormhole attack [19], and Sybil attack [14,25]. Karlof et al. [20] proposed a mechanism to defend against the rushing attack. The paper proposed that every node only processes beacon messages through bidirectional links as well as verified neighbor nodes. However, the paper uses a trusted base station for neighborhood verification, which is not scalable for a large sensor network. Our solution of defending against a rushing attack also proposes bidirectional verification, but instead is based on a lightweight pairwise key set up scheme for neighbor node verification. Newsome et al. proposed a set of mechanisms to defend against the Sybil attack in a sensor network, including radio resource testing and key validation in random pairwise key predistribution schemes [25]. In INSENS, the pairwise key between a base station and a sensor node can be used to defend against a Sybil attack.

Pairwise key setup is an important concept for WSNs and has been extensively studied in recent years [16,9,21, 15,33]. Our pairwise key set up scheme is used for bidirectional verification and neighborhood authentication. Our scheme is lightweight, similar to [33] and [3]. However, our threat model is stronger than [3], and our scheme is more resilient to master key compromise, when compared with [33].

While the issue of intrusion tolerance has been known for quite some time [17,5], recent increase in the need for safety-critical systems has significantly raised research activity in this area. Recent projects addressing intrusion tolerance include [6,28,29,32]. All these projects are aimed at providing intrusion tolerance capabilities in a traditional, resource-rich computing environment.

Previous work on the basic INSENS protocol [11] proposed an intrusion tolerant protocol that sets up secure tree-structured routing with multiple paths in a WSN. However, in this basic INSENS, every sensor node needs to send a feedback message to the base station, which is not scalable. In addition, the REQ message is vulnerable to rushing attacks. Another work [13] improved INSENS by employing multiple base stations and introducing bidirectional verification. In this paper, we have proposed stronger pairwise key schemes, an enhanced single-phase INSENS protocol for better scalability, improved adaptability to changes in topology so that nodes may securely join or leave, and have conducted more extensive experiments to evaluate the effectiveness and cost of redundant routing in INSENS.

3. Network framework and threat model

The design of the basic INSENS protocol targets moderately-sized WSNs of a couple hundred nodes or less. The design of the enhanced INSENS protocol targets large-sized WSNs of a thousand nodes or more, e.g. large scale battlefield deployments. We assume that each sensor node has an activity range v such that if the distance between any two sensor nodes is no more than v, they can send and receive data to and from each other. We also assume that communication channels are symmetric, i.e. if a node a can receive a message from *b*, then it can also send a message to *b*.

We assume that an adversary can pose the following threats:

- An adversary can physically capture a sensor node and is capable of compromising a sensor node to obtain all of its information, e.g. cryptographic keys and important routing information. An adversary can also reprogram a sensor node to convert it into a malicious node. However, we assume that the adversary requires some significant time to compromise a node.

- An adversary has a jamming range d, $d \geq v$. Within a circle of radius d, the adversary can generate radio signals to interfere with signals generated by other sensor nodes or base stations. However, the adversary can jam only a small part of the network, i.e. d=D, where D is the radius of the complete sensor network.
On the other hand, we assume that the adversary can receive data from any sensor node or base station, only if the distance is less than v. So, an adversary's packet acceptance range is still v, while his jamming range is greater than v. This is because it is easy to send a stronger data signal that can go beyond distance v, it is difficult to receive data from a sensor node that is further than distance v. Receiving data from nodes further than distance v requires very sensitive and expensive equipment, and we assume that the adversary does not possess them.

Finally, we assume that a base station is resource-rich and has sufficient capability to protect itself from tampering. For a moderately-sized sensor network, a base station is capable of computing and maintaining routing information of every sensor node in the network.

4. Basic INSENS protocol

The basic INSENS protocol is divided into two parts: route discovery and data forwarding. Route discovery ascertains the topology of the sensor network and sets up appropriate forwarding tables at each node by exchanging control messages. It is performed in three phases. In the first phase, the base station securely floods a request message to all reachable sensor nodes in the network, as shown in Fig. 2(a). In the second phase, sensor nodes securely send their (local) topology information using a feedback message back to the base station, as shown in Fig. 2(b). In the third phase, the base station verifies this topology information, computes the multipath forwarding tables for each sensor node, and securely unicasts those tables in a breadth-first manner to the respective nodes using a routing update message. This results in a multi-hop multipath data forwarding tree. Fig. 2(c) illustrates only the multipath routes constructed for one node, not for all nodes. The complete multipath tree will be a union of all multipath routes for single nodes. After this point, multi-hop data forwarding can commence.

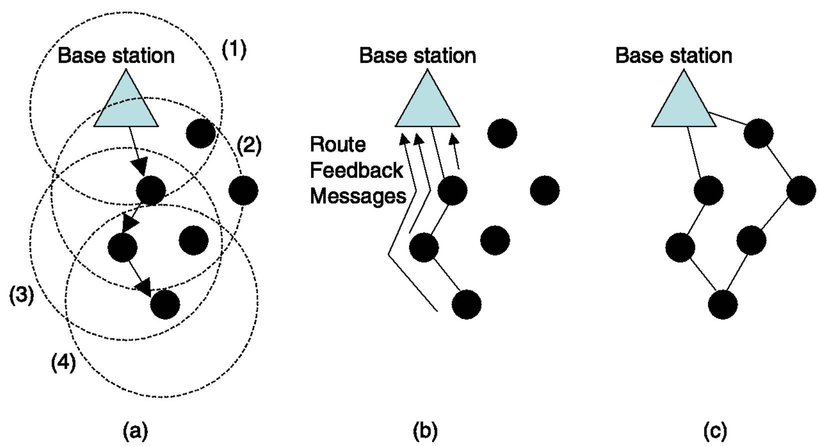

Fig. 2. Three Phases of Basic INSENS: (a) ROUTE REQUEST is flooded from the base station (only one path is shown here). (b) ROUTE REPLIES are unicast back to the base station from each sensor node, containing neighborhood topology information. (c) a routing table is securely unicast to each node, in a breadth-first manner, establishing multipath routing.

We assume that each node is preconfigured with a symmetric key that it shares only with the base station. This key is used to protect the confidentiality, authenticity and integrity of the data exchanged between the base station and each sensor node. For added security, instead of using this key K directly, each sensor node can derive a separate encryption key K_E and a MAC key K_M from the shared key K [26]. Every node is also preconfigured with a globally known one-way function F and an initial one-way hash chain number S_0. F and S_0 are used together to prevent flooding except by the base station.

4.1. Route discovery: route request

The *route request* message is flooded in the sensor network to inform each sensor node to send its neighborhood information to the base station. The base station initiates this first phase whenever it needs to construct the routing paths of all sensor nodes. The base station broadcasts a *request message* that is received by all its neighbors. The format of this request message is

$$BS \rightarrow : REQ||BS||OHC||MAC(K_{BS},REQ||BS||OHC)$$

where REQ is the message type, BS is the ID of the base station, *OHC* is a one-way hash chain *sequence number*, and || denotes concatenation.

A sensor node that receives a request message for the first time in turn re-broadcasts the request message, but in a modified form. A node x replaces the ID in the received REQ message with its own ID x. A node x also recomputes a new MAC based on its own pairwise key shared only with the base station, as well as on the previous MAC in the received REQ message. The format of the modified request message forwarded by a sensor node x is

$$x \rightarrow : REQ||ID_x||OHC||MAC(K_x,ID_x||OHC||MAC\_of\_parent)$$

Each sensor node maintains a neighbor set and selects the first neighbor that it hears the REQ message from as its parent, i.e. when a node x receives a request message for the first time, it records the sender's id as its parent and also includes the sender in its neighbor set. When x receives a repeat REQ message (identified by the same *OHC*) any time thereafter, it includes the identity of each sender in its neighbor set, but does not rebroadcast the request message. A limited number of neighbors is kept, namely the first neighbors heard from for this REQ message, to forestall a Sybil attack.

In the request message rebroadcast by a sensor node, the MAC is recomputed based on the contents of the newly constructed REQ message, and the MAC of the parent node, i.e. the MAC embedded in the received *REQ* message. We call this form of MAC *a nested message authentication code*. The nested MAC uniquely identifies the MAC as being generated from a particular node along a particular path/sequence of nodes, thereby preventing replay of the MAC as false proof of being a neighbor in another area. More detail is provided in the description of the route feedback phase.

A *request message* must be protected against spoofing attacks, in which an adversary sends forged *request messages*. By launching such an attack, an adversary can (1) impersonate the base station and have all route feedback messages from sensor nodes directed to itself; this will allow him to learn important topology information of the network, and prevent the base station from receiving that information. (2) launch a denial of service attack by flooding the entire network.

We use one-way hash chains to address this issue. OHCs are lightweight in terms of computation and memory, and are thus ideally suited for WSNs. A one-way hash chain number OHC included in the *request message* limits an adversary's ability to flood the base station's REQ messages as follows. The base station uses a one-way function F to generate a sequence of numbers $S_0, S_1,...,S_n$, such that $S_i = F(S_{i+1})$, where $0 \leq i \leq n$. Initially, every node is pre-configured with F and S_0. In the very first route discovery phase, the base station includes *one-way hash chain sequence number* (referred to as OHC henceforth) S_1 in the first request message it broadcasts. Each node can authenticate that this OHC sequence number originated from the base station by verifying $S_0 = F(S_1)$. In general, the base station uses S_i in the ith route discovery phase. This mechanism allows a sensor node to verify that an OHC it received indeed originated from the base station, because the one-way characteristic of F ensures that only the base station can correctly generate the next OHC. This prevents an adversary from spoofing a base station and arbitrary broadcasting packets since he cannot predict the next OHC given the current OHC. Therefore, an adversary cannot arbitrarily inject forged REQ messages and flood the network.

The overall effect of these security mechanisms is that a malicious node can attack in the first phase only by jamming its neighbor nodes, dropping a request message, or launching a rushing attack (described later). The first two attacks may result in some of the malicious node's neighbor nodes not receiving a correct request message. However, the flooding mechanism limits the effectiveness of message dropping or jamming by allowing valid REQ messages to reach nodes downstream of the affected area through other paths.

4.2. Route discovery: route feedback

After forwarding a request message in phase one, each sensor node waits for some fixed period of time before starting the second phase. In the second phase, a node x unicasts a *feedback message* to the base station. This feedback message contains x's neighbor set and is protected by a keyed MAC. The format of a *feedback message* sent by node x with parent y is

$$x \rightarrow y\text{: FDBK}||\text{ID}_x||\text{E}(K_{x_E},\text{NBRx})||\text{MAC}(K_{x_M}, \text{OHC}||\text{FDBK}||\text{ID}_x||\text{E}(K_{x_E}, \text{NBR}_x))$$

Here, MAC is the message authentication code of the complete feedback message and is generated using K_{x_M}. NBR_x stores the neighbor information of x. For example, if x has k neighbor nodes, namely n_1 ... n_k, its neighborhood information is

$$\text{NBR}_x : ID_{n_1}||\text{MAC}_{n_1}||\text{ID}_{n_2}||\text{MAC}_{n_2}|| \ldots ||\text{ID}_{n_k}||\text{MAC}_{n_k}$$

When a base station receives a feedback message, it can verify the integrity of NBR_x by computing the MACs. Intermediate nodes cannot tamper with neighbor information without being detected. Since MAC_{n_i} is generated as a function of the upstream parent's MAC, the *nested* MAC is a function of the path that the REQ message has taken before arriving at node n_i. Therefore, a malicious node cannot replay this MAC in another part of the network as a proof of a (fake) neighbor. This attack is one form of wormhole attack [19], and our nested MAC is able to defend against it. Conversely, the base station will be able to reconstruct the path as nodes report their topology information and therefore verify that the MAC is consistent with the reported neighborhood and paths taken. Also, since the MAC of each neighbor is dependent upon the OHC, then an adversary will be unable to repeat the MAC as proof of being a neighbor in later rounds of REQ broadcasts. Further, a malicious node will be unable to invent nodes because it does not have the key to generate the valid MACs, thus forestalling Sybil attacks.

While the MACs enable the base station to construct a correct topology, this topology may be incomplete due to lost, dropped or tampered feedback messages. An important property of the second phase is that the feedback messages that reach the base station are guaranteed after verification to be correct and secure from tampering. Also, confidentiality of a feedback message is preserved against eavesdropping, because each node encrypts appropriate information in its feedback message.

The overall effect of these security mechanisms is that a malicious node is limited in the damage it can inflict, whether attacking by DOS attack, not forwarding a *feedback message*, or modifying the neighborhood information of nodes. These attacks will be unable to deceive the base station, but will result in some of the nodes downstream from the malicious node not being able to provide their correct neighbor information to the base station. However, a malicious node can still launch a battery-drain and/or DOS attack by persistently sending spurious feedback messages. To forestall this type of attack, SHUSH employs a simple rate-limiting mechanism that throttles the maximum rate at which a node can send messages. The result is to limit the damage that can be caused by battery-drain and DOS attacks during the feedback phase.

4.3. Route discovery: computing and propagating multipath routing tables

After sending its *request message* in the first phase, the base station waits for a certain period of time to collect all the connectivity information received via feedback mess-ages. For each feedback message, the base station verifies that its MAC is correct, and then verifies that the MAC-protected neighborhood information is also correct and consistent. The base station constructs a topology of the network from these authenticated/verified feedback mess-ages. Since some feedback messages may have been lost/dropped/tampered with, the topology constructed by the base station may be incomplete. However, it is guaranteed that this incomplete topology will still be consistent with the full toplogy of the network.

The base station computes the multipath forwarding tables of each node in the network using the topology it has constructed. While INSENS is largely agnostic to the particular criteria for choosing multiple paths, we offer the following multipath heuristic in order to proceed with our implementation of INSENS. For a sensor node A, the first path from A to the base station is chosen using Dijkstra's shortest path algorithm. To determine the second path, three sets of nodes, N_1, N_2, and N_3 are first constructed. N_1 is the set of nodes belonging to the first path, N_2 is the set of nodes belonging to N_1 and any neighbor nodes of the nodes in N_1, and N_3 is the set of nodes belonging N_2 and any neighbor nodes of the nodes in N_2. All three sets exclude *A* and the base station. The second path is then computed as follows.

1. Remove all nodes in N_3 from the network, and find the shortest path from *A* to the base station. If such a path is found, terminate the computation. The path found it is the second path.

2. Remove all nodes in N_2 from the original network. Find the shortest path from *A* to the base station. If such a path is found, terminate the computation. The path found it is the second path.

3. Remove all nodes in N_1 from the original network. Find the shortest path from *A* to the base station. If such a path is found, it is the second path. Otherwise, there is no second path from *A* to the base station.

Notice that depending on the network topology, it is possible that no second path is found. In that case, the current implementation of INSENS maintains only a single path.

After computing redundant paths for each node, the base station computes the forwarding table for each node. These forwarding tables are unicast to the respective nodes in a breadth-first manner. The base station first unicasts the forwarding tables of all nodes that are its immediate neighbors. It then unicasts the forwarding tables of nodes that are at a distance of two hops from it, and so on. This mechanism cleverly uses the redundant routing mechanism just built for nodes closer to the base station to distribute the forwarding tables to nodes further from the base station. Standard security techniques such as SNEP [27], in combination with pairwise keys between the base station and destination nodes, can be used to unicast these forwarding tables in a secure manner, preserving end-to-end confidentiality, authenticity, and integrity of the routing information.

4.4. Data forwarding

A node maintains a forwarding table that has several entries, one for each route to which the node belongs. Each entry is a 3-tuple: $\langle destination, source, immediate\ sender\rangle$. *Destination* is the node id of the destination node to which a data packet is sent, *source* is the node id of the node that created this data packet, and *immediate* sender is the node id of the node that just forwarded this packet. For example, given a route from node *S* to *D*: $S \rightarrow a \rightarrow b \rightarrow c \rightarrow D$, the forwarding table of node *a* will contain an entry (*D*, *S*, *S*), forwarding table of *b* will contain an entry (*D*, *S*, *a*), and the forwarding table of *c* will contain an entry (*D*, *S*, *b*). With forwarding tables constructed in this way, forwarding data packets is quite simple. On receiving a data packet, a node searches for a matching entry $\langle destination, source, immediate\ sender\rangle$ in its forwarding table. If it finds a match, it forwards (broadcasts) the data packet.

Although INSENS sets up a routing path for each sensor node in the network, it does not require every node to send data all the way to the base station. For example, to conserve energy, only aggregator nodes may desire to send (processed) data to the base station [20,12].

4.5. Limitations of the basic approach

There are several limitations on security, scalability, and maintenance of the basic INSENS protocol. First, the assumption about a wireless communication channel being symmetric is not valid for many WSNs. As a result, although a node *u* can receive a *request message* from node *v*, it may not be able to send its *feedback message* to *v*. Even worse, an adversary can exploit this asymmetry to launch a *rushing attack* [18] to capture a large number of sensor nodes. In such an attack, after receiving an REQ message with the correct current OHC, the attacker floods a fake REQ message at a higher signal rate, thereby causing more nodes to view itself as the base station. Second, depending on the density of the network, the *feedback message* can be too long to fit into a single packet. Also, the overhead of forwarding these feedback messages across multiple hops, and of forwarding the routing tables across multiple hops, can be quite high if the size of sensor network is large. Third, since the base station needs to compute routing paths for each sensor node, it can get overloaded with processing if the network is large. Finally, the basic algorithm doesn't address the issue of maintaining network routing when some existing nodes fail or some new nodes join the network.

5. Enhanced INSENS protocol

The enhanced INSENS protocol incorporates several unique features and countermeasures to address the limitations of the basic INSENS protocol: (1) bidirectional verification is used to defend against the rushing attack; (2) multiple paths to multiple base stations is used to make INSENS more scalable for larger sensor networks; and (3) a set of secure maintenance mechanisms are introduced to manage node joining and leaving in a network.

5. 1. Bidirectional verification

To defend against rushing attacks, we introduce two techniques that are based on the principle of bidirectional verification. First, an *echo-back* scheme ensures that a node *x* accepts REQ messages from only those nodes that had earlier echoed back a response to *x*'s ping. A malicious node that suddenly expands its transmission power will not have echo-responded to *x*'s ping and so its unfamiliar REQ message will be ignored. Second, a REQ message is encrypted along each hop with a cluster key, so that the REQ messages that are accelerated by a rushing attack will fail to use the proper cluster key for authentication and encryption and will be dropped.

5. 1. 1. Echo-back scheme to verify neighbor nodes

An adversary is able to launch a rushing attack when a sensor node *x* fails to check whether a sender with an expanded transmission range can reciprocally receive *x*'s data. If a sensor node can detect that it cannot reach the transmitter, then that node can identify and block a rushing attack. To launch a rushing attack, an adversary's packet sending range *d* must be bigger than a normal node's sending range *v*. If each sensor node constructs a set of reachable neighbor nodes, and is only willing to receive *REQ* messages from this set of neighbor nodes, then *REQ* messages from an adversary transmitted with larger power will be ignored. Thus, the damage from a rushing attack can be restricted within a small range *v*.

To identify neighbor nodes, we introduce a simple *echo-back* scheme. In this scheme, a node *x* only forwards the *REQ* messages for the nodes that can receive a message from *x*. Those nodes are termed *x*'s reachable neighbors. We will describe how each sensor node securely finds its reachable neighbor nodes and securely identifies the *REQ* messages from its reachable neighbors. Fig. 3 shows the REQ flooding scheme, the rushing attack, and the echo-back defense. Notice that the rushing attack is not completely precluded with the echo-back defense. Multiple adversaries can cooperate to form a relay path that is shorter than the normal *REQ* propagation path. However, such a cooperative attack is much more difficult to launch than the rushing attack addressed here.

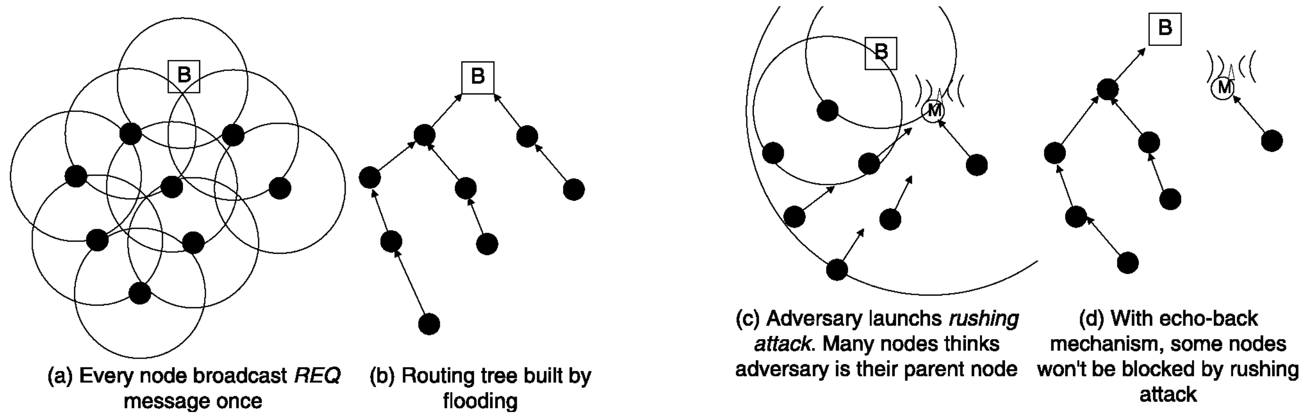

Fig. 3. Enhanced single-phase INSENS: (a) secure REQ message flooding builds a (b) secure routing tree. (c) A standard rushing attack is (d) blocked by the echo-back countermeasure.

5.1.2. Cluster key set up

To defend against a rushing attack, each REQ message forwarded by a node *x* is encrypted with a cluster key. That key is set up during the echo-back process. In this way, *x*'s reachable neighbors can decrypt and verify the *REQ* message while the adversary will not know the key and will be prevented from launching a rushing attack. Cluster key set up combined with the echo-back mechanism is performed prior to the arrival of REQ messages.

INSENS employs pairwise keys to secure the echo-back scheme. One way to establish pairwise key between neighbor nodes is to use random pre-distributed pairwise key schemes [16,9,21,15]. However, these schemes require longer time for pairwise key establishment and consume more memory. It has been shown that the memory consumption of random-key schemes increases as the size of network increases [8]. In addition, neighbor nodes need to apply certain protocols to find if they have shared keys, and that costs extra time.

Instead, INSENS employs a new variation of the *transitory global key establishment scheme* that overcomes the deficiencies of random-key schemes. A transitory global key establishment scheme, as proposed in LEAP [33], has a very limited memory footprint-a node needs to store only a global for a short time before erasing it. This global key is used to set up pairwise keys. In addition, if an adversary can compromise a node during the key establishment phase (similar to the key infection scheme proposed in [3]), he will be able to capture only the pairwise keys within his eavesdropping range. The pairwise keys in other parts of network will still be secure.

Our new method for deriving pairwise keys from a transitory global key, suggested below, is immune to some weaknesses of the LEAP scheme. In particular, if an adversary is ever able to compromise a node before the global key is erased in LEAP, then LEAP's scheme for computing pairwise keys from the global key allows the adversary to compute all pairwise keys in the network.

First, let us consider a simple pairwise key set up in which all nodes in the network are assumed to share a single global key. Each node x locally broadcasts an *echo* message to its neighbor nodes with format:

$$x \rightarrow : \mathrm{ECHO} \| E_{\mathrm{global\_key}}(\mathrm{ID}_x \| \mathrm{nonce})$$

where *ID* is the ID of sensor node *x*, *nonce* is a random number.

If a node *y* receives this message, it generates a random number $K_{y,x}$ as the pairwise key between *x* and *y*, and echos back a message with format

$$y \rightarrow : \mathrm{ECHOBACK} \| E_{\mathrm{global\_key}}(\mathrm{ID}_y \| \mathrm{nonce} + 1 \| K_{y,x})$$

When node x receives this message, it records node y as its verified neighbor, and compares its ID number with y's ID number. If $ID_x < ID_y$, node x and y use the random number nonce ($K_{x,y}$) generated by x as their pairwise key. Otherwise, if $ID_x > ID_y$, then they use the random number ($K_{y,x}$) generated by y as their pairwise key.

The global key is only used to encrypt the pairwise key during the echo-back process. If an adversary obtains the global key after a node has received its pairwise key, it cannot know the pairwise key. If an adversary obtains the global key before the echo-back process finishes, it can obtain the pairwise keys within its range, but is unlikely to obtain the pairwise keys outside of its range, because those nodes would have finished their echo-back by the time the adversary moves outside its current range. In this way, our transitory global scheme for computing pairwise keys is more secure and robust than LEAP.

However, if an adversary obtains the global key, it can initiate echo-back many times by sending several echo messages. The adversary can fabricate several false identities using such a Sybil attack, adding ghost nodes (with false identities) into the network. In addition, new malicious nodes can join anywhere in the network by initiating echo-back using the (compromised) global key to set up legitimate pairwise keys with legal nodes.

To prevent such attacks, a node destroys its global key from memory after a certain time that is long enough to set up pairwise keys with all its neighbors. Before a node x destroys its global key, it generates a new key $K_x = MAC(global\_key, ID_x)$, and a set of random numbers $y_1,..., y_k$, where $y_i = MAC(global\_key, r_i)$, and $r_1,...,r_k$ are random numbers. These data are used to set up keys with newly joining nodes, described later in Section 5.3. 1.

After a node x has set up pairwise keys with all of its neighbors, it sets up a single cluster key [33] for encrypted data communication with its neighbors. Node x's cluster key KC_x is a key shared by x and all of x's verified neighbors. To set up KC_x, x generates a random number KC_x, and unicasts it to all its verified neighbor nodes, encrypted with the respective pairwise keys. To forward a *REQ* message, it encrypts the message with KC_x and appends a MAC generated using KC_x. It is also possible to generate two cluster keys, one for encryption and the other to generate a MAC.

5.2. Securing multi-path multi-base station routing

In the basic INSENS protocol, if an adversary compromises a node before the second (*feedback* message) phase, it can block all its downstream nodes by simply dropping feedback messages. Bidirectional verification doesn't help here. To address this problem, we employ multiple base stations in the network, and conFig. multiple paths to multiple base stations for each sensor node. This significantly reduces the number of nodes that can be blocked by an adversary who manages to compromise a sensor node. Furthermore, with multiple base stations, a base station doesn't need to compute the routing paths for sensor nodes, nor contend with a large-sized feedback messages or routing tables. This significantly improves the scalability of INSENS and as a result, the enhanced INSENS can be used for secure routing in large sensor networks.

Given the pairwise and cluster keys, the process of setting up multiple routing paths is as follows:

Step 1: Every node uses the echo-back scheme to identify its neighbor nodes and sets up pairwise keys with its verified neighbor nodes. Then it unicasts its cluster key to each of its neighbor nodes encrypted using that neighbor's pairwise key.

Step 2: Each base station broadcasts its *REQ* message to its neighbor nodes. The format of the *REQ* message is:

$$REQ \| ID_s \| E_{KC_s}(OHC \| ID_B)$$

Here *REQ* is the message type, ID_s is the ID of the sending node s, ID_B is the ID of the base station who generated this *REQ* message, and *OHC* is that base station's one-way hash chain number.

Step 3: When a node *x* receives this *REQ* message, it checks the sender ID. If *s* is *x*'s verified neighbor, *x* decrypts and authenticates the one-way hash chain number *OHC* with *s*'s cluster key. Next, *x* uses its one-way function *F* and its cached *OHC* number of base station *B* to verify the new incoming *OHC* number. If the *OHC* is valid, it replaces its cached *OHC* number with this new value, encrypts and MACs the *OHC* with its own cluster key, and broadcasts the newly encrypted *REQ* message.

The end result is that multiple spanning trees are securely constructed, rooted in each base station. Feedback messages and downloading of routing tables are eliminated. In the enhanced INSENS protocol, the addition of the transitory global key enables nodes to trust, verify and admit neighbors locally. The flooding of the REQ messages then securely establishes direction of routing without requiring feedback to each base station. In contrast, the basic INSENS protocol assumed only pairwise keys with the base station, with no trust between neighbors. Therefore, the base station had to be involved in establishing trust between neighbors, giving rise to a need for feedback messages and down-loading of routing tables.

The effect of multiple-base station routing depends on the number of base stations and the placement of these base stations. Different applications have different constraints for the location and number of base stations. In general, base stations should be placed far away from each other to make the system resilient to node compromises. We performed a preliminary investigation of the effect of the number and placement of base stations on the extent of intrusion tolerance in a sensor network [10]. In sensor network applications, a user may replace a base station at one location and move it to another location. The question is how much processing is needed to reestablish all the routes when a base station is moved from one location to another. If there is only one base station, the base station only needs to query its neighborhood information to compute routing tables for all sensor nodes, i.e. there is no need for the first or the second phase. This is because the base station already knows the network. After computing new forwarding tables, the relocated base station can unicast them to each node [10]. If there are multiple base stations, a base station which just moved to a new location needs to only broadcast a new REQ message in the network, to trigger a construction of new routes to this base station from various nodes. These solutions assume that a base station is not continuously mobile, a scenario which is beyond the scope of INSENS.

5.3. Maintenance issues: message loss, nodes joining and leaving

In addition to securely building routing paths, INSENS addresses a number of maintenance issues. These issues include (1) *REQ* messages forwarded by different nodes may collide and as a result some nodes may not receive *REQ* messages; (2) After constructing routing tables, some nodes may run out of power or become damaged. Since these nodes cannot forward data packets, communication between some live nodes and the base station may be blocked. In addition, some new nodes may be deployed after initial network deployment.

To address the problem of lost routes due to failed or compromised nodes, INSENS employs the following procedure. If a node *u* has not received a *REQ* message for some time interval, it initiates a local repair method by sending a path request message *PREQ* to its neighbor nodes.

$u \rightarrow : \mathrm{PREQ}\|\mathrm{ID}_u\|\mathrm{jMAC}(\mathrm{KC}_u, \mathrm{PREQ}\|\mathrm{ID}_u)$

u's neighbor nodes that have recently received path information reply to *u*. For example, if neighbor node *v* received a *REQ* message, *v* sends a reply message *PRLY* to *u*:

$v \rightarrow u : \mathrm{PRLY}\|\mathrm{ID}_v\|\mathrm{MAC}(K_{uv}, \mathrm{PRLY}\|\mathrm{ID}_v)$

After *u* collects all neighbors responding affirmatively, *u* randomly selects one of these neighbor nodes as its parent node. Notice that *u* does not use any routing metrics claimed by its neighbor nodes, e.g. the distance to base station, to decide *u*'s parent node. This limits a malicious node from attracting many downstream nodes to itself by claiming that it has a shorter path to the base station. This method of local repair does not ensure that *u* chooses the shortest path to the base station. However, in a dense network, u's path should not significantly differ from its neighbor nodes.

The same method of local repair can be used when a new node joins the network. First, a new node establishes pairwise keys with its neighbor nodes. Then, it chooses any one of its neighbor nodes that have a path to base station as its parent node. Later on, when base stations flood new *REQ* messages, this node can find a different parent node.

5.3. 1. Pairwise key set up with new nodes

When a new node u is added to the network, it needs to setup pairwise keys with existing nodes. Before this key set up, u should verify whether the nodes it talks to belong to the network, and the nodes in the network need to verify that u is legitimate. We assume that the new node is configured with the global key. An existing node x will have its derived key K_x, and a set of y_i and random numbers r_i, as defined in Section 5.1.2. An existing node x can authenticate u by sending r_i and a *nonce* to u, where $1 \leq i \leq k$. If u has the global key, it can compute $y_i = MAC(global\_key, r_i)$ and sends $MAC(y_i, \text{nonce})$ back to x. To authenticate x, u asks ID_x from x, and computes K_x. Then u can verify if x belongs to the network by sending a random number R to x. It then waits for x to send back $MAC(K_x, R + 1)$. Since only the node that knows K_x can generate $MAC(K_x, R + 1)$, u can authenticate x by verifying the received message. After x and u have authenticated each other, they can set up a shared key between them. The following formula shows how a new node u and an old node x authenticate each other, and set up their pairwise key $KEY_{u,x}$.

$$u \rightarrow * : \text{JOIN}||u||R$$

$$x \rightarrow u : \text{JOINREPLY}||x||u||r_i||\text{nonce}||\text{MAC}(K_x, R + 1||r_i||\text{nonce}||x||u)$$

$$u \rightarrow x : \text{JOINVERIFY}||u||x||\text{MAC}(y_i, \text{nonce}||\text{Key}_{u,x}||u||x||u)||\text{Key}_{u,x}$$

Any node that does not have a global key will be unable to join the network. Since nodes destroy their global keys soon after setting up their pairwise keys and cluster key, compromising a node after it has finished its key set up will not gain the attacker any ability to inject new false nodes into the network. The attacker's compromised node will not be able to generate correct responses y_i to the random number challenges r_i. We assume that new nodes just introduced to a network, e.g. dropped by an airplane into an existing sensor network, will have the global key temporarily and can securely add themselves to the network.

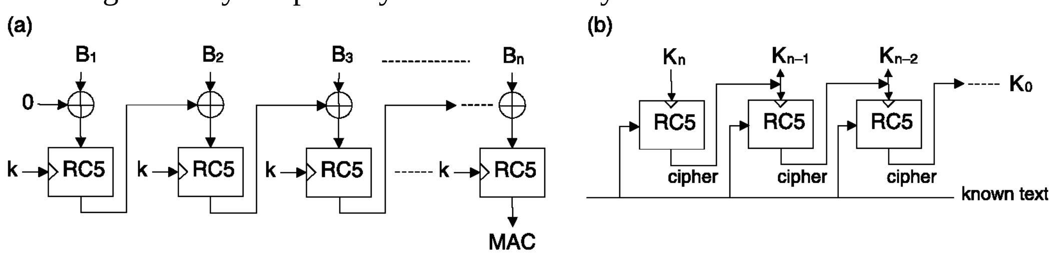

Fig. 4. Message authentication code and one-way hash chain generation. (a) CBC-based MAC generation (b) one-way hash chain generation.

6. Implementation Basic INSENS protocol

The basic INSENS protocol was implemented on a network of 10 sensor motes running TinyOS 1.0 with NesC. A base station implemented in Java receives information from the motes via a programming board, processes the information, and then sends back routing tables to each mote. The Breadth First Search (BFS) algorithm was chosen to determine two paths from each node to the base station. All compute-intensive functions are written as tasks to prevent them from blocking packets or time interrupts.

The keyed MAC plays a critical role in INSENS, and is used to authenticate each node, paths to the base station from each node, and neighbor information of each node. A standard CBC mode was used to generate each MAC given the block cipher RC5 [24]. This generator is shown in Fig. 4(a). The following criteria was used to generate a one-way hash chain: given a plaintext and the corresponding ciphertext computed using a block cipher algorithm, e.g. RC5, the key that is used to generate the ciphertext cannot be computed. Our one-way sequence number generator used by a base station to generate a one-way has chain is shown in Fig. 4(b). The base station chooses a random key S_n and uses it to encrypt a well-known plaintext and gets a cipher. This cipher is S_{n-1}. Next, the base station uses S_{n-1} as a key to encrypt the same known plain text to compute S_{n-2}. This process continues until the base station has computed S_{n-1}, S_{n-2}, … ,S_0, which is a one-way hash chain.

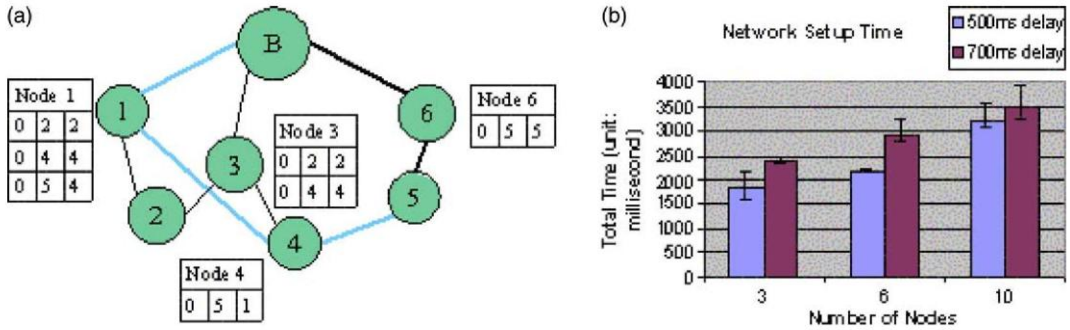

Fig. 5. (a) Routing tables built by INSENS (b) Network setup time.

In Berkeley motes running TinyOS, the default packet size is 36 bytes. However, the size of a feedback message in the basic INSENS protocol can be much larger, because it contains all neighbor information. In our implementation, one feedback message is segmented into multiples of 36 byte feedback packets. The following criteria for feedback packet segmentation is used to maintain compatibility with INSENS and prevent possible DOS attacks: every segment packet is assigned a distinct sequence number. A node must forward a packet with a lower sequence number before forwarding a packet with a higher sequence number. If a node receives packet with a higher sequence number and hasn't yet received a packet with a lower sequence number, the node drops that packet. Any tampering with the sequence number or other contents of a segment packet can be detected by the base station, because every feedback message contains a MAC.

6. 1. Network setup time

To evaluate the network setup time, we measure the time interval between the time the base station broadcasts its request message and the time it receives all '*routing table received'* messages. The network is considered to be a dense network, so that every node has several neighbors. There are several factors affecting the setup time: (1) execution time of cryptographic algorithms; (2) execution time of packet processing; and (3) waiting time that includes random delay, feedback message waiting time, and the base station waiting time. In our experiments, the base station waits at most 500 ms after receiving a feedback packet. This wait time is reset with each new feedback message. Eventually, when no new feedback messages arrive, the base station times out and computes the routing tables. Each sensor node also waits at most 500 ms for neighbor information to be collected. We also tested 700 ms timeouts for the sensor nodes (not the base station). The base station unicasts a custom routing table to each mote, and waits 100 ms between sending each routing table. We found that the total network setup time is dominated by the waiting time of the sensor nodes. In comparison, the computation time of RC5- based cryptographic algorithms is relatively short. Fig. 5(a) shows the routing tables built by INSENS at some of the sensor nodes when there are six nodes in the network. Fig. 5(b) shows the network setup time as a function of the number of nodes in the network.

7. Performance evaluation of the enhanced INSENS protocol

7. 1. Overhead of cryptographic algorithms

In the enhanced INSENS protocol, a sensor node needs to save a global key, pairwise keys, cluster keys, one-way hash chain numbers, and several random numbers for new node authentication. Suppose each key is 8 bytes (64 bits) long. If a node has n neighbor nodes, keeps l random numbers, and there are k base stations, then the node needs $8 \times (2n + k + l + 2)$ bytes to store all keys. For example, if there are 4 base stations, and a node has 10 neighbor nodes, and keeps 5 random numbers, then 248 bytes are needed to store all keys. Current sensor nodes provide 4 KB SDRAM, 128 KB flash memory, 4KB embedded EEPROM, and 128K extended EEPROM. If the keys are not changed often, they can be stored in the 4KB embedded EEPROM.

To evaluate the computing overhead of cryptographic algorithms in *REQ* flooding and destination address encryption, we implemented encryption/decryption algorithms, and one-way hash chain verification on Berkeley MICA1 sensor motes [1]. We chose RC5 (with 12 rounds) as the block cipher to implement these algorithms. Table 1 shows the performance of our implementation. The results show that it takes about 4 milliseconds to encrypt and decrypt the content of a packet, which is about 30 bytes. The delay due to one-way hash chain verification is about 4. 2 milliseconds, and SDRAM memory consumption is about 136 bytes. These results suggest that the overhead of encryption/decryption, storage requirements, and verification of one-way hash chain number on sensor nodes is reasonable.

Table 1
Overhead of RC5-based cryptographic algorithms

	Speed (msec)	Code (Bytes)	Data (Bytes)
Encryption (30)	1.94	1488	112
Decryption (30)	2.02	1518	112
One-way hash chain	4.18	1768	136

7.2. Effectiveness of multipath routing

7.2. 1. Node failure

INSENS builds multiple paths to bypass compromised nodes. With multiple independent routes available between every node and the base station, our protocol's goal is to route messages correctly in the presence of adversaries and failed nodes. We begin by assessing the impact of node failure, in which nodes that have failed can no longer forward data packets. We have performed a set of experiments to measure the number of nodes that can be blocked when a set of nodes have failed. Fig. 6 shows the average number of nodes that can be blocked as a function of the number of failed nodes. In this simulation, we measured the number of blocked nodes under three scenarios: (1) single-path routing; (2) 2-path routing in the basic INSENS protocol (single base station); and 4-path routing with 4 base stations.

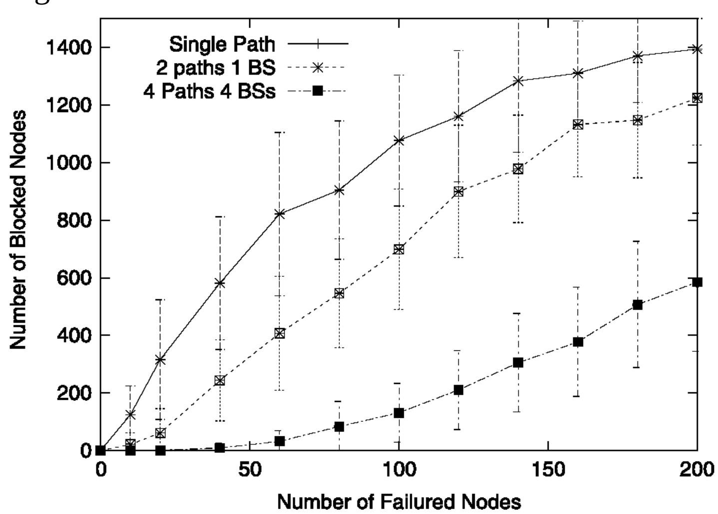

Fig. 6. Effects of node failures.

These measurements were performed for a network of 2000 nodes randomly distributed over a space in which every node has about 16 neighbor nodes in average. (The next 2 experiments use the same network configuration.) The numbers reported in this figures are averaged over 50 different combinations of nodes randomly selected to be failed nodes. This result shows that the multipath scheme used by INSENS noticeably increases the robustness of the network compared to a single-path scheme. The improvement in robustness is even more dramatic when multipath routing to multiple base stations is considered.

7.2.2. Jamming attacks

We have performed a set of experiments to analyze the effect of jamming attacks that a compromised node may launch. The jamming attack we have simulated in these experiments is comprised of repeatedly sending a jamming signal to reachable sensor nodes so that these nodes cannot send their data packets. This jamming attack is one kind of DOS attack and is quite difficult to address completely at the network level.

Fig. 7 shows the intrusion tolerance of INSENS to jamming attacks by assessing the number of nodes that can be blocked by compromised nodes launching such an attack. The number of blocked nodes in this attack depends on the effectiveness of multipath routing, the jamming range of the compromised node, the topology of the network, and the number of jammers. The x-axis records the number of compromised nodes that launch a jamming attack. The y-axis records the number of nodes that are blocked. As the jamming range increases, adversaries can block more and more nodes in the network. These figures clearly demonstrate that the multipath routing schemes increase the connectivity of sensor nodes and base stations, improving the resilience of the network to jamming attacks for all ranges and populations of jammers. All three figures show that the adversaries can block fewer nodes when the enhanced INSENS protocol is in place, compared with a simple single-path routing protocol. When a compromised node has a small jamming range, we can also see that the basic INSENS protocol is more intrusion-tolerant than a single-path routing protocol. However, as the compromised nodes are able to jam larger areas, the intrusion tolerance of the basic INSENS protocol becomes similar to the single-path routing protocol.

While the results shown here are for a random network topology, we showed in [11] that the number of blocked nodes under a jamming attack is fewer for a grid network topology because of the existence of alternative routes. Also, the effectiveness of the basic INSENS protocol was assessed in earlier work for a moderately sized network [11].

7.3. Effectiveness of secure multipath set up

To evaluate the effectiveness of the echo-back scheme with multipath to multiple base station routing, our routing path set up scheme was simulated and the number of nodes that can be blocked by an adversary was measured. An adversary can block a node *n* if it can prevent a valid *REQ* message from reaching *n*. It can do so by first compromising a sensor node and then launching a rushing attack from that node, such that *n* does not receive any valid *REQ* message.

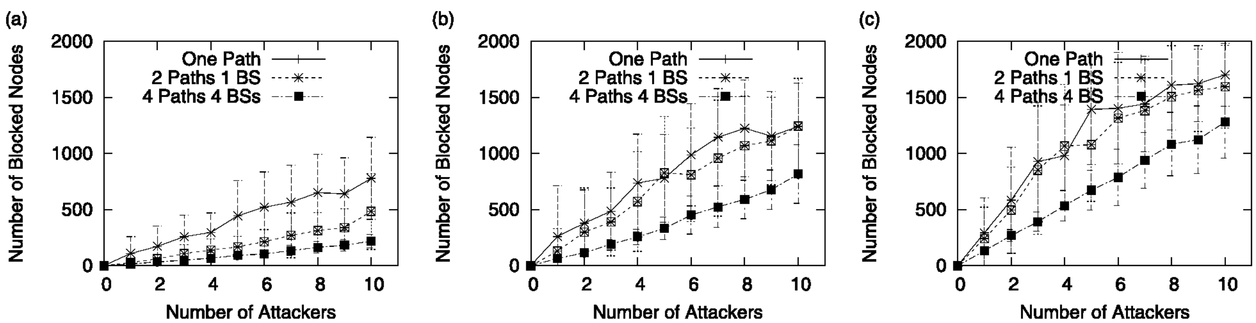

Fig. 7. Effects of Jamming Attack During Data Routing. (a) Jamming range = activity range (b) Jamming range = 2X of activity range (c) Jamming range = 3X of activity range.

Two scenarios were simulated. In the first scenario, there is one base station at the center of the network, and in the second scenario, there are four base stations at the four corners of the network. The experiments varied the transmission range of the compromised node from two to four times the data transmission range of a normal node when the echo-back approach was not used. We also experimented with the transmission range of the compromised node being the same as the data transmission range of a normal node when the echo-back approach was used.

One to ten compromised nodes were randomly selected from these 2000 nodes. The number of blocked nodes was measured given rushing attacks from the compromised nodes with transmission range varying from one, two, or four times the data transmission range of normal nodes. These experiments were repeated one hundred times. Fig. 8 shows the average number of nodes blocked by compromised nodes.

Fig. 8(a) shows the results for the single base station scenario. The echo-back approach is very effective in limiting the rushing attack. For example, if adversaries launch rushing attacks from 10 different places (i.e. 10 compromised nodes) and their packets can reach nodes four times further away than packets sent by a normal node, almost half of the nodes in the network are blocked. In comparison, when echo-back is used to defend against rushing attacks, only about 5% of the nodes in the network are blocked. Fig. 8(b) shows the results for the multiple base station scenario. From this figure, we can see again that the echo-back approach is still very effective against rushing attacks. In addition, compared with Fig. 8(a), we can see that multiple path routing to multiple base stations provides considerably more robust network connectivity than the single base station scenario, especially in combination with the echo-back defense.

7.4. Cost of intrusion tolerance

Sections 7.2 and 7.3 show that INSENS is intrusion tolerant and provides significant protection against a variety of malicious attacks during the routing set up phase as well as during the data forwarding phase. It is important to evaluate the cost of this intrusion tolerance support. Sections 6.1 and 7.1 discussed the overhead of INSENS during the initial routing setup phase. The computational and storage overheads are relatively small, while the setup time was dominated by wait times at the sensor nodes. The transmission overhead of basic INSENS setup requires flooding of the REQ messages, unicast of FDBK messages, and unicast of routing table downloads. This cost is incurred once per REQ message. In a static network, this cost may occur only once during initialization and is thus amortized across the lifetime of the network. If REQ messages are flooded more frequently, then the cost of setup is amortized across REQ interarrival times. However, the enhanced INSENS protocol considerably reduces this overhead by removing both the feedback and routing table download phases of INSENS for large sensor network. The enhanced INSENS protocol’s overhead consists of localized pairwise and cluster key set up as well as flooding of a REQ message, and is therefore on the order of other sensor network routing schemes such as TinyOS beaconing that flood route discovery messages.

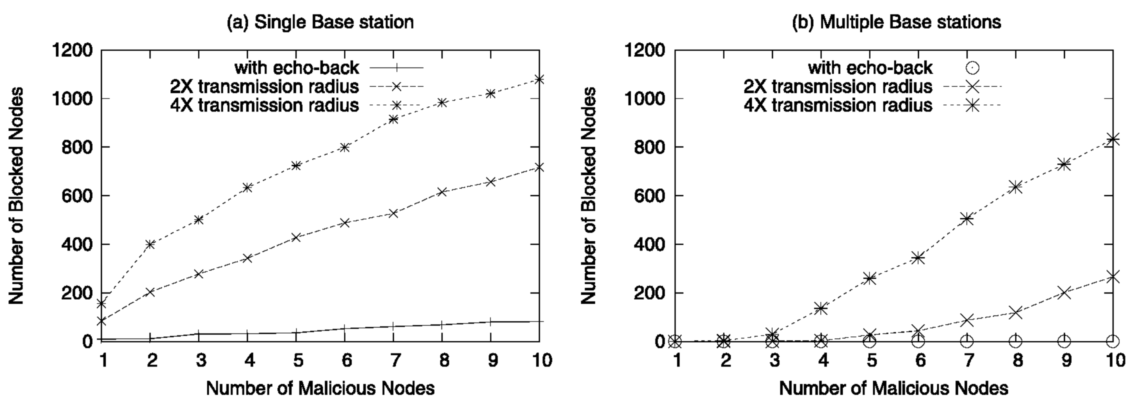

Fig. 8. Effects of rushing attack during multipath routing setup.

In the data forwarding phase, INSENS requires multiple copies of data messages to be sent. This is the price of multipath routing for reliability and intrusion tolerance. However, it is important to note that in any reasonably large network, communication of sensor data to the base station takes place in hierarchies. A small number of nodes are designated as aggregator nodes that collect data from all sensor nodes in their vicinity, aggregate this data, and send the aggregated data the base stations. In effect, only the aggregator nodes communicate with the base stations. Since the number of aggregator nodes is significantly smaller, sending multiple copies of data messages does not translate into a similar increase in overhead.

8. Conclusion

This paper described INSENS, an INtrusion-tolerant routing protocol for wireless SEnsor NetworkS. The key objective of an INSENS network is to tolerate damage caused by an intruder who has compromised deployed sensor nodes and is intent on injecting, modifying, or blocking packets. The basic INSENS protocol securely and efficiently constructs tree-structured routing for WSNs in a three-phase process: the base station floods route requests; each sensor node unicasts back a route feedback messages containing neighborhood topology information; and the base station verifies this topology information and then unicasts multipath routing tables bread-first to each sensor node. Basic INSENS incorporates efficient one-way hash chains as one-way sequence numbers to limit the ability of an adversary to flood the network. Nested keyed message authentication codes are used to uniquely and securely associate a MAC with a node, a particular path, and a specific OHC number, thereby defending against replay attacks through worm-holes. Multipath routing improves intrusion tolerance. Adapting to WSN characteristics, the design of INSENS also pushes complexity away from resource-poor sensor nodes towards resource-rich base stations. An enhanced single-phase version of INSENS scales to large networks, accommodates multipath routing to multiple base stations, integrates bidirectional verification to defend against rushing attacks, enables secure joining/leaving, and incorporates a novel pairwise key setup scheme based on transitory global keys that is more resilient than LEAP. A prototype implementation of basic INSENS over network of motes and a simulation of enhanced INSENS in NS2 show that INSENS tolerates malicious attacks launched by intruder nodes, and limits the damage an intruder can cause.

References

[1] TinyOS website, http://webs.cs.berkeley.edu/tos/.
[2] K. Akkaya, M. Younis, A survey on routing protocols for wireless sensor networks, To appear in Journal of Ad Hoc Networks.
[3] R. Anderson, H. Chan, A. Perrig, Key infection: smart trust for smart dust, In 12th IEEE International Conference on Network Protocols, Berlin, Germany, October 2004.
[4] U.A.F. ARGUS Advanced Remote Ground Unattended Sensor Systems, Department of Defense, Argus, http://www.globalsecurity. org/intell/systems/arguss.htm.
[5] L. Blain, Y. Deswarte, An intrusion tolerant security server for an open distributed system, In First European Symposium in Computer Security, Toulouse, France, 1990.
[6] C. Cachin, J.A. Poritz, Secure intrusion-tolerant replication on the internet, In IEEE International Conference on Dependable - Systems and Networks (DSN'02), Washington DC, USA, June 2002.
[7] J. Carlson, R. Han, S. Lao, C. Narayan, S. S. ghani, Rapid prototyping of mobile input devices using wireless sensor nodes, In WMCSA'03, Monterey, CA, USA, October 2003.
[8] H. Chan, A. Perrig, PIKE: peer intermediaries for key establishment in sensor networks, In Proceedings of IEEE Infocom, March 2005.
[9] H. Chan, A. Perrig, D. Song, Random key predistribution schemes for sensor networks, In IEEE Symposium on Security and Privacy, May 2003.
[10] J. Deng, R. Han, S. Mishra. Enhancing base station security in wireless sensor networks, Technical Report CU-CS-951-03, Department of Computer Science, University of Colorado, April 2003.
[11] J. Deng, R. Han, S. Mishra, The performance evaluation of intrusion-tolerant routing in wireless sensor networks, In IEEE 2nd International Workshop on Information Processing in Sensor Networks (IPSN'03), Palo Alto, CA, USA, April 2003.
[12] J. Deng, R. Han, S. Mishra, Security support for in-network processing in wireless sensor networks, In First ACM Workshop on Security of Ad Hoc and Sensor Networks, in association with the 10th ACM Conference on Computer and Communications Security, Fairfax, VA, USA, October 2003.
[13] J. Deng, R. Han, S. Mishra, Intrusion tolerance and anti-traffic analysis strategies in wireless sensor networks, In IEEE International Conference on Dependable Systems and Networks (DSN' 04), Florence, Italy, June 2004.
[14] J. Douceur, The sybil attack, In First International Workshop on Peer-to-Peer Systems, volume 2429 of Lecture Notes in Computer Science, Cambridge, MA, USA, Springer, Berlin, March 2002.
[15] W. Du, J. Deng, Y.S. Han, S. Chen, P. Varshney, A key management scheme for wireless sensor networks using deployment knowledge, In IEEE 23rd International Conference on Computer Communication, Hong Kong, China, March 2004.
[16] L. Eschenauer, V. Gligor, A key-management scheme for distributed sensor networks. In Conference on Computer and Communications Security, (CCS'02), Washington DC, USA, November 2002.
[17] J.-M. Fray, Y. Deswarte, D. Powell, Intrusion-tolerance using fine-grain fragmentation-scattering, In 1986 IEEE Symposium on Security and Privacy, Oakland, CA, USA, April 1986.

[18] Y. Hu, A. Perrig, D. Johnson, Rushing attacks and defense in wireless ad hoc network routing protocols, In Second ACM Workshop on Wireless Security (WiSe’03), San Diego, CA, USA, September 2003.
[19] Y.-C. Hu, A. Perrig, D.B. Johnson, Packet leashes: A defense against wormhole attacks in wireless networks, In Proceedings of IEEE Infocom 2003, April 2003.
[20] Secure routing in wireless sensor networks: Attacks and counter-measures, Ad Hoc Networks 1 (2–3) (2003).
[21] D. Liu, P. Ning, Establishing pairwise keys in distributed sensor networks, In CCS’03, Washington DC, USA, October 2003.
[22] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson, Wireless sensor networks for habitat monitoring, In First ACM Workshop on Wireless Sensor Networks and Applications (WSNA’02), pp. 88–97, 2002.
[23] D.J. Malan, M. Welsh, M.D. Smith, A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography, In First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks(SECON’04), Santa Clara, CA, USA, October 2004.
[24] A. Menezes, P. Oorschot, S. Vanstone, Handbook of Applied Cryptography, CRC Press, Boca Raton, 1996.
[25] J. Newsome, R. Shi, D. Song, A. Perrig, The sybil attack in sensor networks: analysis and defenses, In Proceedings of IEEE International Conference on Information Processing in Sensor Networks (IPSN 2004), April 2004.
[26] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J. Tygar. Spins: security protocols for sensor networks, In Seventh Annual International Conference on Mobile Computing and Networking, Rome, Italy, July 2001,pp.189–199.
[27] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J. Tygar, Spins: Security protocols for sensor networks, Wireless Networks Journal (WINET) 8 (5) (2002) 521–534.
[28] H.V. Ramasamy, P. Pandey, J. Lyons, M. Cukier, W.H. Sanders, Quantifying the cost of providing intrusion tolerance in group communication systems, In DSN’02, Washington DC, USA, June 2002.
[29] D. Sames, B. Matt, B. Niebuhr, G. Tally, B. Whitmore, D. Bakken, Developing a heterogeneous intrusion tolerant corba system. In DSN’02, Washington DC, USA, June 2002.
[30] F. Wang, R. Uppalli, Sitar: a scalable intrusion-tolerant architecture for distributed services-a technology summary, In Volume II of Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX III), April 2003.
[31] A. Wood, Denial of service in sensor networks, IEEE Computer 35 (10) (2002) 54–62.
[32] T.J. Wu, M. MalKin, D. Boneh, Building intrusion tolerant applications, In Eighth USENIX Security Symposium, Washington DC, USA, August 1999, pp. 79–91.
[33] S. Zhu, S. Setia, S. Jajodia. Leap: Efficient security mechanisms for large-scale distributed sensor networks. In 10th ACM Conference on Computer and Communications Security, Washington D.C, USA, October 2003.