IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 17, 2019, accepted June 28, 2019, date of publication July 10, 2019, date of current version July 26, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2927818

Protean Authentication Scheme – A Time-Bound Dynamic KeyGen Authentication Technique for IoT Edge Nodes in Outdoor Deployments

SHIJU SATHYADEVAN[1,2], KRISHNASHREE ACHUTHAN[1], ROBIN DOSS[2], AND LEI PAN[2]
[1]Centre for Cyber Security Systems and Networks, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Kollam 690525, India
[2]Centre for Cyber Security Research, Deakin University, Geelong, VIC 3220, Australia

Corresponding author: Shiju Sathyadevan (shiju.s@am.amrita.edu)

ABSTRACT The IoT edge/sensor nodes are exposed to large attack surface and could easily succumb to several well-known attacks in the wireless sensor network (WSN) domain. Authenticating edge nodes before they join a network, especially after a sleep state, is a critical step to maintain the overall trust of any given IoT local area network (IoT LAN). The low resources and computational constraints of such IoT nodes make this a challenging and non-trivial problem. As many IoT deployments are in uncontrolled environments, the IoT devices are often placed in the open so that physical access to them is inevitable. Due to easy physical access of the devices, common attacks, including cloning of devices or stealing secret keys stored in an edge node, are some of the most common attacks on the IoT deployments. This paper focuses on developing an extremely lightweight authentication scheme for constrained end-devices that are part of a given IoT LAN. Authentication occurs between the end-device and the gateway that acts as an edge computing device. The proposed authentication scheme is put through both formal and informal security verification. Voltage drop, current, and power are measured to gauge the overall impact of the security scheme. All the three parameters were measured while configuring the edge node as an end-device or as a router. Our testing results show that the impact on the resources was minimal.

INDEX TERMS Edge node authentication, IoT authentication, IoT security, time-bound IoT authentication, resource constrained devices.

I. INTRODUCTION

Securing IoT deployments in outdoor environments is a genuine challenge especially due to easy physical access to sensor nodes. Devices that can sense environmental parameters and/or can respond to them through wireless communication are qualified to be part of any given IoT LAN. The majority of these devices are sensors that sense and stream data wirelessly to a gateway or directly to a cloud-based application through middleware. Actuators are also critical members of any given IoT network. The actuators respond to service requests raised by the application hosted at the gateway or in the cloud based on the sensed parameters. IoT deployments can enable the real-time monitoring and management of any environmental domain and find application in a variety of contexts. This heavy dependency on interconnected, intelligent devices makes it imperative to ensure the level of security for such devices and the generated data. Security solutions should be designed, developed, and deployed with utmost care as frequent outage for patch updates due to identified vulnerabilities cannot be exercised in critical real-time infrastructures.

In most cases, the authentication of IoT edge nodes is achieved by sending a secret key to the platform while triggering a cross validation check. If the requester is a preregistered device and the credentials match, then it is authenticated. Usually, the hard-coded secret key of any device is stored in the devices on-board non-volatile memory. Hence, during the authentication when the client and server share the secret key, an adversary can clone the key by replicating the device for authentication purposes [1]. The following two conventional methodologies are used by attackers to break trust in an IoT LAN:

- In the first methodology, an attacker could create a replica of the genuine device using hardware tools like

The associate editor coordinating the review of this manuscript and approving it for publication was Zhengguo Sheng.

AVR Dragon [1]. Then the attacker would extract the firmware into an image file and burn the firmware image into another device of similar nature (stolen device are used in most cases). During this extraction process, the secret keys stored in the memory will also be copied over to the replicated device along with the firmware. Thereafter the cloned device can be successfully authenticated to join the network as a genuine device.

- In the second methodology, the attacker will extract the firmware and then obtain the secret key by reverse engineering the hexadecimal code embedded in the firmware. Once the secret key is obtained, the attacker can craft his/her own HTTP request based on the platform's encoding style and subsequently send the malformed data to the platform. This methodology requires reverse engineering expertise from the attacker's side [1]. To combat such attacks, we need complex authentication schemes. As more and more devices are connected to the IoT LAN, the number of parallel re-authentication cycles will also increase.

However, the key challenges in implementing a lightweight authentication security scheme for an IoT LAN are two-folds:

- The nodes in IoT networks are severely resource-constrained. Limited device memory (<= 2kb onboard RAM and <= 32kb program memory), ultra-low processing capability (<= 8bit), and low energy availability, introduces a variety of new security concerns in the IoT domain.
- Consequently, deploying the traditional security models in such constrained networks becomes almost impossible; to safeguard it against the most widely adopted attacks like eavesdropping, spoofing the IP address, node fabrication, and flooding [1].

Therefore, any practical solution would require limiting heavy computations to the gateway where the majority of complex resource-consuming computations would be performed but maintaining a feather weight computational load at the edge node. Our proposed scheme fulfils such an objective and can be summarized as below.

- Our scheme is designed to be hosted even on ultra-resource constrained edge nodes. The proposed scheme can run on end/sensor devices such as an 8 bit, 20mHz ATmega328P processor with 32KB program memory size and 2KB onboard RAM. Such devices cannot run an industry standard hash function.
- The authentication process initiates only whenever an edge node re-establishes a connection with the gateway, based on the state of the edge node. The majority of edge nodes tend to remain in the sleep state and will respond to the sense and push requests raised by the application hosted either at the gateway or on the cloud.
- Once in the streaming mode there is no need to re-authenticate until either its current active streaming mode comes to an end or the current session times itself out.
- Re-authentication is mandatory when the edge node wishes to re-join the network at a later point of time. The technologies engaged by edge nodes and their capabilities may vary depending on the deployment. Some devices are constrained in terms of resources, while others are not and could afford to run resource-hungry authentication/security schemes.
- Gateways being even richer in resources could be loaded with IDS/IPS, encryption techniques with secure key storage mechanisms. Hence securing the gateway as such is not considered as part of this paper.

II. MOTIVATION

In an IoT environment, how to establish trust among participating devices is always a fundamental and practical issue. The intention of this research is to ensure that any given IoT networks are safe against potential attacks that materialize primarily due to relatively easy access to the edge nodes planted in the open.

- Every edge node in a network should be able to authenticate itself with the coordinator/gateway node before establishing a trusted connection.
- Being low on resources the edge nodes cannot hold keys securely so that the trust parameters can be easily stolen to demolish trust relatively easily.
- Considering the constraints of the edge nodes, it is important to keep the authentication mechanism as lightweight as possible so that it requires the least computational resources and consumes fewer battery resources.
- Above all, storing a static key to perform the authentication cycle would not be ideal for IoT implementations.

A. RESEARCH CONTRIBUTIONS

The research contributions from this paper are manifold.

1. Any new authentication protocol should possess lightweight computation and a time-bound, constantly changing the key that is synchronized both at the edge node and the gateway. This would then ensure that the most common physical attacks on edge nodes such as eavesdropping, device cloning, man-in-the-middle attack, key extraction through memory dump, and so on, would be ineffective or would demand more sophistication to steal the authentication keys.
2. The proposed authentication technique engages a mechanism that would constantly change the authentication keys at regular intervals thereby reducing the attack span due to the pressing fact that the edge nodes are exposed to direct physical attacks while deployed in the open. Since the key constantly changes at configurable intervals, this new lightweight technique is named as ''Protean Authentication Scheme''.
3. The scope of this research is limited to protecting only the extremely resource-constrained edge nodes deployed primarily in outdoor environments where the devices are physically accessible almost any time.

In summary, this paper proposes a lightweight authentication scheme which is capable to (1) optimize the number of secure handshakes based on the state of the edge node without compromising the required trust level, and (2) migrate complex computation to gateway nodes with the resource awareness.

B. PAPER ORGANIZATION

The rest of this paper is organized as follows: Related work on various authentication schemes are discussed in Section III. Section IV expands on authentication design overview. Section V presents the protocol methodology in detail. Section VI explains the formal security analysis performed on the protocol followed by informal security analysis in section VII. Experiments and results are presented in Section VIII. The paper is concluded in Section IX.

III. RELATED WORK

The current practice of sharing the keys involves an exchange of the shared key unencrypted over the air [15]. This one time exchange of shared key unencrypted would result in losing the confidentiality and so an attacker could easily leverage the sniffed static key to break into the network. In most cases, the entire IoT LAN would use the same shared encrypted secret key for all gateways to node communication and vice versa. This would leave open a wider attack space as one single compromised key would mean a fully compromised network. In a time-constrained memory key model (TLMK) [15], a key, a salt value and a key lifetime are transferred to the target node. This protocol expects pre-deployment of keys and also directs all communications between devices through a central KDC. There is a heavy dependency on the KDC, and if there is a KDC outage, then the whole network would cease to function. Conventional cryptographic solutions come up with high levels of security, ignoring the requirements of constrained devices. Lightweight cryptography (LWC) [2] was developed in recent times primarily focused on devices with restricted capabilities in both hardware and software. It also works under constrained power and connectivity. Dependency on static cryptographic keys or hardware keys would eventually become the vulnerability soft spot in the whole authentication scheme. Even static authentication credentials transmitted over the air can be sniffed and can be replayed from a rogue device to be authenticated.

The proposed authentication scheme in [16] is good enough for home automation devices considering a majority of the edge nodes are connected to the main AC supply. Computation at the edge nodes is highly intense and so it will not be an ideal solution for sensor nodes with constrained resources where the frequency of key change is required and frequency of authentication required would be more often than what is proposed in [16]. Besides this shortfall, MAC values are used in the scheme which use the same secret key to generate and verify the exchanged credentials. It implies that both the sender and the receiver of a message must exchange pre-agreed keys much before initiating communications, which is similar to symmetric encryption. For the same reason, MACs do not exhibit any non-repudiation property specifically in the case of a network-wide shared secret key: The weakness of MAC is that any user who can verify a MAC is also capable of spoofing MACs for other messages.

The authentication mechanism proposed in [17] proposes a new scheme to authenticate between sensor nodes, proxy, and cloud. To reduce resource consumption at the sensor nodes, the computation is passed onto the proxy with the proxy in turn connecting to the cloud. It is assumed that there is a secure channel between the proxy and the sensor nodes with a long-term secret key shared among them. Using this key, nodes can perform symmetric key encryption. In the case of most deployments where sensors are planted in the open, expecting to have a secure channel between the sensors and proxy is not a viable approach unless keys are exchanged in an encrypted fashion. In the long term, the secret key is the weakness of the proposed scheme as it is stored in the memory. A lightweight mutual verification and key exchange methods based on a hash function are mentioned in [18]. The proposed scheme is also dependent on a CA which is also an overhead. Edge nodes have fairly intense computation to be performed as part of the handshaking process which is once again a drain on the batteries. Dabbagh and Saad in [20] proposed an authentication technique using the device's unique fingerprints. The proposed scheme can differentiate between security attacks and normal changes in fingerprints because of the environmental effect on the surrounding objects. It proposes to use a Received Signal Strength-based scheme for proximity detection on single antenna devices. The scheme is proposed primarily for home automation devices where an attacker tries to interfere with the smart devices from outside the building. There are noticeable RSS variations in both the cases which are good enough to pick up possible attack build ups.

Octopus [23] is a secure and efficient scheme to allow any Fog user to mutually authenticate with any Fog server in any Fog under the authority of a Cloud service provider. The Fog user is required to store one master secret key in the registration phase only once. Using this master key the Fog user is able to mutually authenticate with any Fog server managed by the Cloud service provider. Insider attack can be prevented since the brute force attack against the master key is difficult because of its bit length. The scheme has much less computational complexity since it needs to perform very few hash invocations and symmetric key cryptography. Mutual authentication between Fog User (FU) and Fog Server (FS) is achieved. The shared key $k_{FS}FU$ is protected since it is used to encrypt along with a random nonce R_{FU} which is never known to an eavesdropper. This scheme provides confidential communication since the secret key $k_{FS}FU$ is pre-shared and is only known to FU and FS. The computation and storage costs are reduced. The key management is simple. Each session key is chosen as a fresh random string. The secret key guessing attack is not possible since it is long enough

to sustain brute force attacks. Due to the presence of nonce, hashing, and symmetric encryption/decryption, the replay of impersonation attacks are not possible. The man-in-the-middle attack is not possible due to the presence of k_{FU} and hence $k_{FS}FU$ cannot be deduced. If any *FU* is compromised, it will not affect any other fog users. In such case of FU compromise, it should be reported to the *RA*; the master key should be revoked and re-registration should be done. This scheme demands a lot of resources to ensure the correctness according to the theory so that it is not suitable for IoT environment.

The paper entitled "Energy Cost of Cryptographic Key Establishment in Wireless Sensor Network" [24] analyses and compares the energy cost of Kerberos-like key establishment based on AES encryption and a variant of the Diffie-Hellman key exchange using elliptical curve cryptography. According to this study, the overall energy cost of the former is between 39.6 mJ and 47.6 mJ, while the latter consumed energy in the range of 79.0 and 84.6 mJ. Further analysis revealed the fact that the energy consumption of Kerberos is dominated by the message transfer whereas encryption as such consumed only negligible energy. Various security challenges faced during design of IoT systems are outlined in [25]. The authors propose PUF like hardware security mechanisms for securing IoT networks. PUF also demands storage of a secret in the memory of the device. Even though some of the work recommends this secret to be stored in a highly secure manner. Despite the memory being highly secure, it engages non-volatile digital memory to store the bits. This opens up a wider attack surface. Feldhofer et al. [33] endorse that the power consumption of AES is negligible. The paper confirms that after a number of measurements the average power consumption was less than 5 mW when operated at 100 kHz and 1.5 v. An authentication scheme based on one time password (OTP) is detailed in [26]. A lightweight end-to-end authentication scheme is achieved through identity based elliptical curve cryptography. Edge nodes do not need space to store keys because the scheme uses OTP. If the device needs to communicate frequently, then the edge nodes should frequently request for OTP from the central cloud resulting in taking more time for session establishment. But the complexity of the algorithm would induce additional computational burden on the IoT devices in order to compute the OTP using public/private keys that involve multiple iterations. The group authentication protocol mentioned in [27] recommends sharing of keys among multiple nodes. But this scheme leaves multiple nodes at risk even if one of the nodes is compromised. For distributed IoT systems a certificate-based authentication process is recommended in [28]. A drawback of this protocol is that the cryptographic credentials are stored in the edge nodes itself. This leaves the protocol exposed to a device cloning attack. This scheme uses implicit certificates to accomplish an end-to-end authentication in a distributed IoT environment. Even though the scheme uses ECC to keep a check on the overall computational resources needed, it still needs a fair bit of storage at the edge node to store the implicit certificates. The scheme is also depended on a Certificate Authority (CA).

The two-way authentication scheme for IoT proposed in [29] is based on DTLS. It uses both X.509 certification and RSA based asymmetric encryption. In order to establish a session, and the scheme demands eight handshakes which in turn demands a higher computational cost and increased storage from constrained IoT devices. A group authentication protocol for IoT devices is detailed in [27]. Paillier Threshold Cryptography whose special properties such as homomorphic addition, ability to remain indistinguishable and self-binding are used in this scheme. In order to effectively authenticate the group members the scheme establishes a session key for each group authentication. The overhead of this scheme is that a new key need to be regenerated and distributed among the group members as and when a new member joins the group. A lightweight authentication based session key establishment protocol for smart homes is presented in [16]. The protocol needs the presence of a trusted server which would act as a security service provider that would provide critical security parameters and generate and distributes tokens to communicating devices in a smart home. Devices then use these authenticated tokens to establish a session key and attain mutual authentication.

A lightweight authentication protocol for IoT is proposed in [30]. It engages a two-phase authentication process that involves a static and continuous authentication phase. The static authentication phase is used to authenticate devices at the beginning of an authentication period T whereas a continuous authentication scheme is applied to each sensed data transmission during the current authentication period T defined by the gateway. Within the predefined authentication window T, the static authentication scheme will set up an authentication token for the communicating devices which will be used for continuous authentication session. A gateway can henceforth verify the legality of the sensor node within the authentication period T and when new data is transmitted between both the parties. No encryption is used in this scheme. The scheme depends on a secret key which is generated by the gateway and passed onto the nodes through a secure channel over the air unencrypted. An eavesdropper could easily sniff this key. And the keys are expected to be stored in secure storage at the edge node level which can be quite easily extracted through a memory dump operation. The paper does not attempt to measure the voltage, current, and overall power consumption of the scheme.

A pre-shared key based authentication is mentioned in [31]. Here a device is authenticated only before joining the cluster. This seems to be a lightweight protocol but the gateway is not authenticated. Moreover, the key exchange mechanism suggested in this paper seems to be vulnerable. Based on the extensible authentication protocol, the authors in [32] proposes an authentication technique. First, the two communicating parties exchange their identities in order to establish a secure communication channel. An authentication server then ensures the validity of the two entities trying

to communicate through MD5 and/or TLS protocols. This scheme is very much depended on a third-party authentication server and also requires a large number of message exchanges resulting in increased execution time and energy consumption.

IV. AUTHENTICATION DESIGN OVERVIEW

Why is there a dire need for a time bound, constantly changing key based authentication scheme in an IoT paradigm? Traditionally static key values are stored in the Non-Volatile Memory (NVM) of the device. For authentication purposes, the client shares a secret key with the server through a secure channel. The main disadvantage with the shared secret key is that an adversary can clone the device and the secret keys stored in them are copied, thereby replicating the legitimate IoT device. It is also possible to do a memory dump to extract the secret keys which are usually stored in plaintext. Keys generated using a physically unclonable function (PUF) will also prove to be futile as PUF values need to be extracted and stored in the RAM for subsequent authentication schemes, which again makes it vulnerable to all the attacks mentioned above.

Within the IoT ecosystem, edge nodes which are the ''things'' in the internet of things paradigm mostly consist of sensors and actuators that sense, react and respond according to the requests raised by the application hosted either on a gateway or at the cloud. Edge nodes being heavily resource constrained are exposed to wider attack vectors. Most of the sensor nodes tend to be in the sleep state during the majority of their life cycle. They respond to the requests initiated by the application running on the cloud or locally at the gateway. Another category of sensors continuously sense their environment but transmits the data only when there is a change in values beyond a predefined threshold. The sensors that stream continuously would be those that are deployed in critical infrastructures. Of the three categories of sensor operation modes mentioned above, the first two would need frequent authentications, whereas the third category would need one-time handshake before commencing streaming of sensed parameters. In the first two instances, whenever a device wakes up and responds to a request, it should authenticate itself to the gateway assuring that it is indeed the actual intended device that is joining the network. This scheme is designed for devices that use the first two modes of the operation, whereas continuously streaming edge nodes would need to authenticate once through this scheme. The session timeout duration in such instances can be set to a large value so that the re-authentication cycle need not be invoked frequently. Alternatively, the streaming edge nodes can be re-authenticated if the edge node stays silent for a relatively long interval of time.

Any IoT device would need to authenticate itself for transmitting the encrypted data with securely stored keys. For both authentication and encryption purposes, it always needs to store a unique identifier resilient to spoofing and corresponding encryption keys at the device memory itself. Unless these parameters are stored securely, both authentication and encryption mechanism would be futile as both could be compromised. In order to handle these challenges, both software and hardware techniques should be engaged. The software approaches are vulnerable especially when software bugs can be exploited by attackers to break into an IoT LAN. The hardware-based mechanisms to secure the keys are more prominent and difficult to break into. Adding the encryption coprocessors to edge nodes will always cost constrained energy dearly and hence should be used with caution before mass deployments.

Achieving a secure key storage mechanism still leaves open yet another challenge of securely transmitting authentication credentials over the air. An attacker who could intercept a static encrypted authentication packet can replay the same from a rogue device to get it authenticated and thereafter gaining enough privileges to hook up to the network as a genuine device. Hence defining a secure handshake process should also be treated with equal severity as the issue of secure storage of keys. Processors with built-in secure memory areas are available now which allows the MCU alone to access its content thus completely blocking external entities from reading the secure memory location.

In the case of request-response mode, edge nodes wake up when a request is received from the gateway whereas, in the case of ''transmit on state change'' mode, edge nodes wake up and start transmitting data when the sensed parameters have values above a predefined threshold. Under such instances, it is highly desirable that devices authenticate themselves to prove its authenticity. Most of the authentication scheme demands the presence of a central key distribution center, which is distributed to various edge nodes via the gateway. The proposed scheme is designed to void a central key distribution system or any static keys throughout the authentication lifecycle. Encryption keys constantly change with each authentication cycle. New keys would be generated and issued by the gateway under two conditions

- At the end of a predetermined valid session duration

OR

- when the gateway itself enforces an authentication cycle in case the built-in IDS detects malicious behavior.

Such a model would ensure that an attacker who has access to either the authentication key or the encryption key could only use the key during the session duration. Once the session duration expires, the gateway sends in a new set of keys for all subsequent authentication/data encryption and decryption cycles. It is not easy to maintain a synchronized time clock among participating devices in the distributed network, and consequently the proposed scheme does not pose the need for having a synchronized clock for the resource-constrained edge node. For the proposed protocol, we make the following three assumptions:

A. OUR ASSUMPTIONS

1. Edge nodes are resource constrained devices powered by one or more batteries. They have limited computational capability and storage space but can handle minimal computational operations such as exclusive OR.
2. Gateways are resource-rich devices, which have enough computational and storage, capability to hold IDS and/or dynamic firewalls to safeguard against external attacks. The gateway can handle multiple edge nodes.
3. The gateway can employ an IDS or host a lightweight dynamic firewall to protect itself.

V. THE PROPOSED PROTEAN AUTHENTICATION SCHEME

The Protean Authentication scheme relies on minimal initialization vectors. It has little dependency on static stored keys over the entire authentication cycle. Every edge node is expected to have a secure key storage location to hold the initialization vectors which are set up during the initial configuration. The authentication process stores an initialization vector in the protected memory location along with the hardware ID at the edge node and a copy of the same will be maintained at the gateway, which is used only during the initial handshake process. Even if these critical parameters are exposed at a later point of time i.e. any time after the first handshake process, it is virtually impossible for an attacker to recompute and arrive at the current state of the keys generated by the proposed scheme. It prevents the attacker from quickly compromising the keys as each device in the network will have a different initialization vector, hardware ID and henceforth unique keys at any point of time within an authentication cycle. The Protean scheme is computationally less intensive on the device resources because it engages lightweight computation operations such as hash functions, XOR operations and also AES encryption at the gateway side whereas the edge nodes will only perform AES encryption. Keys exchanged can either be stored in a secure EEPROM or in a protected memory location which is not susceptible to cloning attack or memory dumps. These values change frequently as per the session duration set so that an attacker cannot easily grab all changing keys and reuse them within the valid key lifetime. It will be different for each participating node in the network. Any compromise will only be quadrant off at that device alone for a period equivalent to the session valid duration. There are MCU's that have inbuilt secure data and program memory areas with hardware encryption support making it ideal for storing passwords, encryption keys etc. Maxim DS5250 is a low cost, low power MCU designed specifically for IoT edge nodes to securely store the authentication credentials.

The Protean authentication scheme uses AES for both payload encryption as well as for encrypting the authentication credentials exchanged between the gateway and the edge node. Groschadl *et al.* [24] clearly state that energy consumption by AES encryption is negligible compared with its counterparts. Computational energy consumed by AES was only 0.1 mJ. Securely storing the credentials alone does not suffice for the security of the keys as they need to be exchanged for authentication purposes. The attackers eavesdropping on the communication can get hold of the encrypted credentials and could still use them to authenticate a rogue device by replaying the same. Hence the need for changing the keys dynamically and exchanging them securely enough is a must in the IoT paradigm. With the Protean scheme, it becomes difficult for an attacker to guess or to manipulate the sequence of events followed by the scheme as the lifetime of keys is decided dynamically by the IDS hosted at the gateway.

Each edge node has a 32-bit hardware ID (H_{id}) stored securely. An initialization vector (V_i) is assigned to each edge node and is stored along with the hardware ID. (V_i) can be manufacture specific or a large vector series generated based on a specific algorithm. Each manufacturer can use an initialization seed to generate and pre-configure their devices. (V_i) is only used for the very first authentication cycle which can be performed during the initial installation under supervised conditions. After the initialization process, both (H_{id}) and (V_i) are not used anymore in the authentication life cycle. The gateway is aware of both (H_{id}) and (V_i) of each individual edge nodes that are part of the IoT LAN. The gateway as part of its initial handshaking process would craft an authentication packet. The gateway also maintains an authentication table called auth table. The gateway uses a Random number generator and a nonce generator. The gateway expands its 32-bit hardware ID to 128-bit hardware ID and stores it in its secure vault. The gateway also generates a nonce as well as a random number in each authentication cycle and will exchange all the credentials securely with the edge node. For each auth transaction, the gateway would make an entry into the auth table with required details that would assist in retaining the Protean properties of its keys. AES 128-bit encryption is used in all those steps where encryption is applied.

A. NOTATIONS

H_{id}	Hardware ID (32 bit but expandable to 128 bit)
V_i	Initialization Vector
Rn	Random No
Nn	Nonce
KEYret	Used as current encryption key
KEYnext	Used as next encryption key
KEYretPrev	Stores previous KEYret value
KEYnextPrev	Stores previous KEYnext Value
Ekey	Encryption Key
Dkey	Decryption Key
Hid_SeqInit	A new Start sequence no is generated for each valid session
Hid_CtrlVar	CtrlVar is incremented by one to maintain a packet sequence number within a session

B. INITIAL AUTHENTICATION CYCLE FROM THE GATEWAY SIDE

During the first authentication cycle between the gateway and edge node, the gateway stores the hardware id (Hid), initialization vector (Vi), random number (Rn) and nonce (Nn). It is exchanged securely with the corresponding edge node. The gateway then computes KEYret $= h\,(Hid \oplus Nn)$ and $KEYnext = h(Hid \oplus Rn)$. It then sets KEYretPrev $= KEYret$ and $KEYnextPrev = KEYnext$ for the subsequent cycles. The gateway then initializes the encryption Key $Ekey = Hid \oplus Vi$. The initial Packet is crafted in the form $EKey(Hid, KEYret, KEYnext, Nn)$. The gateway then appends a record in its authentication table of the form $Hid, Vi, Rn, Nn, H\,(Nn)\,, KEYret, KEYnext, Ekey, SYN, ACK$. The gateway sends the encrypted packet to the edge node. Each edge node on receipt of the authentication packet will decrypt the message.

C. INITIAL EDGE NODE AUTH ACKNOWLEDGE CYCLE

The edge node computes the decryption Key ($Dkey$) as $Dkey = Hid \oplus Vi$. Using $Dkey$ the incoming packet is decrypted by the edge node and the parameters extracted. It then stores in a secure memory location the current $KEYnext$ value for decrypting the auth packet from the next cycle sent by the gateway. The decryption Key for the subsequent cycle is stored as $Dkey = CurrentKEYnext$.

The edge node sets KEYret from the current packet as the encryption key for the auth acknowledgement packet sent back by the edge node to the gateway. The encryption Key is set as $Ekey = KEYret$. The auth acknowledgement packet from the edge node to the gateway is sent in the form of $\langle\langle h(Nn), Ekey(Hid, KEYnext)\rangle\rangle$.

D. INITIAL AUTH CYCLE FROM GATEWAY SIDE

The gateway uses $h(Nn)$ to identify the latest auth entry for the corresponding edge node from the auth table and extracts its $KEYret$ value. Using the $KEYret$ value, the auth acknowledgment packet is decrypted. The edge node can use the current $Ekey$ to encrypt all subsequent packets for a full valid session duration. A valid session is the time lapse between the sense request generated by the gateway and during when the edge node is actively responding to the requests without any interruption. Once it completes the request-response cycle, the edge node goes to sleep and will only start the cycle after receiving and responding to the AUTH SYN packet from the gateway successfully.

E. SUBSEQUENT AUTH CYCLES FROM GATEWAY TO EDGE NODE

For all subsequent authentication cycles following a session expiry, there will be a constant change of keys. Our scheme will ensure that even if an attacker gets access to a key, it will only be valid for that session and one of the several devices in the network. The life span of a key or the session duration can be defined as per the criticality of the application.

Between the gateway and edge node, the following parameters are generated by the gateway and are exchanged securely with the corresponding edge node in all subsequent auth cycles. In the subsequent authentication cycles the gateway computes $KEYret = h\,(KEYretPrev \oplus Nn)$ and $KEYnext = h(KEYnextPrev \oplus Rn)$. The gateway then sets and stores the values of KEYretPrev = KEYret and KEYnextPrev = KEYnext for subsequent cycles. We then set the initial value for the packet sequence number for the new auth cycle as $\text{Hid}_{\text{SeqInit}} = \text{nextRn}$.

It also initializes $Hid\_CtrlVar$ to 1 whenever a fresh authentication cycle is triggered. Throughout a session period, $Hid\_CtrlVar$ is increment for each packet received from the edge node and is compared against the $Hid\_CtrlVar$ value sent by the edge node. This additional computation verifies the running sequence of the packets generated. Such a measure will make it difficult for an attacker to synchronize the sequence while crafting packets and transmitting from a rogue device or a captured device. During an authentication Cycle Hid_CtrlVar is set to Hid_SeqInit + 1 else Hid_CtrlVar is incremented by 1. The encryption Key Ekey is updated with the KEYnextPrev value. A packet is crafted in the form Ekey(Hid, Keyret, Keynext, Nn, , Hid_SeqInit). The gateway authentication table is updated with an entry of the form Hid, Vi, Rn, Nn, h(Nn), Hid_SeqInit, Keyret, Keynext, $E_{key,}$ SYN, ACK. The gateway then sends the encrypted packet to the edge node. The edge node on receipt of the auth packet will decrypt the same.

F. SUBSEQUENT AUTH CYCLES AT THE EDGE NODE

The decryption Key $Dkey$ is set to Key_{next} in the previous auth cycle and securely stored. Using D_{key} the incoming packet is decrypted and parameters extracted. It stores the current Key_{next} value for decrypting the next cycle of auth packet from the gateway. The decryption Key stored for the subsequent Cycle is set to the $CurrentKEYnext$. Each edge node uses the Key_{ret} to encrypt the authentication acknowledgement packet and sends it back to the gateway: Encryption Key Ekey is set as Keyret. During the authentication phase, $SeqCtr$ is reset to $Hid\_SeqInit$ and then incremented by one for the data transfer cycle within the corresponding valid session period. During an authentication cycle SeqCtr is set to Hid_SeqInit otherwise SeqCtr is incremented by one. The Authentication ACK packet is sent from the edge node to the gateway in the form h(Nn), Ekey(Keyret, Keynext, Hid_SeqInit, SeqCtr).

G. SUBSEQUENT AUTH CONFIRMATION AT THE GATEWAY

The gateway uses the $h(Nn)$ to identify the device entry from the auth table and extracts its $Keyret$ value. Using the $Keyret$ value, the acknowledgment packet is decrypted. At the gateway, $Hid\_CtrlVar$ is incremented for each received packet from that device by one and compares with the incoming $SeqInit$ from the edge node. This process continues for each subsequent cycles. Unless the $SeqInit$ is in the right sequence, the gateway will not accept data packets from any given edge

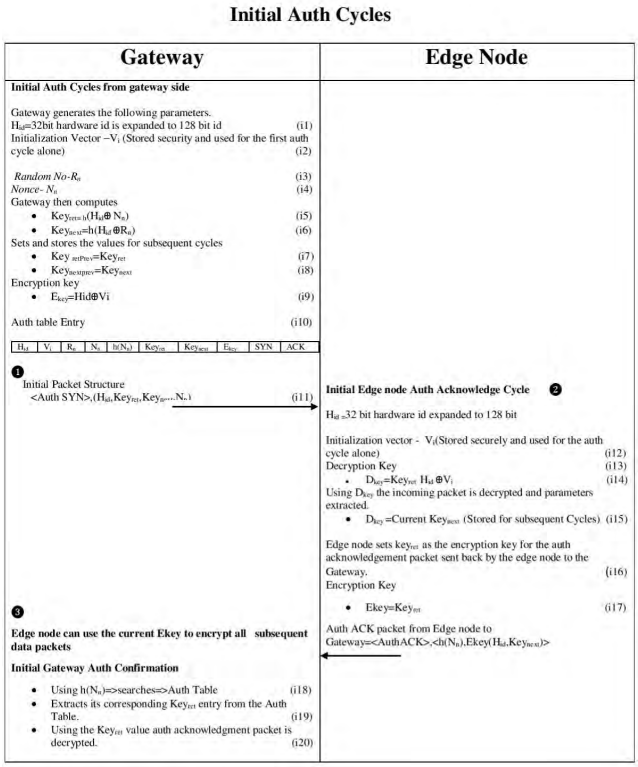

FIGURE 1. A high-level summary of the initial authentication cycle.

FIGURE 2. A high-level summary of the subsequent authentication cycle.

node. It adds an additional layer of security against attacks like replay attack, Node capture/compromise attacks, and so on.

H. SUBSEQUENT AUTH CONFIRMATION AT THE GATEWAY

Using any given session, all payloads are encrypted using the dynamically generated session limited encryption key *Ekey*. Along with the payload, *SeqInit*, *SeqCtr* are also sent in cipher text which is compared on the fly at the gateway to ensure that the packets are sent and received in an orderly and timely fashion. If sequence numbers are found out-of-sequence, then the node is suspected to be under attack and hence can be isolated from the network.

VI. FORMAL SECURITY ANALYSIS USING AVISPA TOOL

For the formal security verification the proposed protocol was simulated using extensively-applied SPAN+AVISPA. For verifying the proposed protocol, the tool has in it four backends:

1. On-the-fly Model-Checker (OFMC)
2. Constraint Logic based Attack Searcher (CL-AtSe)
3. SAT-based ModelChecker (SATMC)
4. Tree Automata based on automatic approximations for the analysis of security protocols (TA4SP).

The proposed protocol is translated into AVISPA High-Level Protocol Specification Language. The basic roles are used to represent each entity. The entities of the proposed protocol include Gateway G and End node/edge node N. The role of the user is as shown in Fig. 3. Composed roles instantiate one or more basic roles as shown in Fig. 4. Using two different channels SND and RCV, the two participating entities communicate. The session role declares all the channels used by the basic role, as shown in Fig. 5. Using the Dolev-Yao model, the intruder is modeled. In this model, the active intruder can modify, delete, or change the contents of the messages being exchanged. The analysis results of the proposed protocol using OFMC and CL-AtSe is shown in Fig. 6. We can see that OFMC and CL-AtSe find no vulnerabilities. In other words, the stated security goals are satisfied for a bounded number of sessions as specified in the environment role. Thus, we conclude that the proposed protocol is resilient to MITM and replay attacks.

VII. INFORMAL SECURITY ANALYSIS

In this section, the security analysis is performed to evaluate the security robustness of the Protean authentication protocol. The only computation that is performed at the edge node is a storage of a dynamic key that changes with each auth cycle and an AES encryption which is only performed each time a request is raised by the gateway for sensed parameter on which the device wakes up from sleep state to respond to the request raised.

Our protocol uses the hardware id and the initialization vector only at the very beginning of the authentication cycle

```
role
role_G(G:agent,N:agent,KeynextPrev:symmetric_key,SND,RCV:chann
el(dy))
played_by G
def=
      local

      State:nat,Nn:text,Hid:text,Keyret:symmetric_key,HidSeqIni
t:text,Keynext:symmetric_key,SeqCtr:text
      init
            State := 0
      transition
            1. State=0 /\ RCV(start) =|> State':=1 /\
HidSeqInit':=new() /\ Nn':=new() /\ Keynext':=new() /\
Keyret':=new() /\ Hid':=new() /\
SND({Hid'.Keyret'.Keynext'.Nn'.HidSeqInit'}_KeynextPrev)
            2. State=1 /\
RCV({Keyret.Keynext.HidSeqInit.SeqCtr'}_Keyret) =|> State':=2
end role

role
role_N(G:agent,N:agent,Keyret:symmetric_key,SND,RCV:channel(dy
))
played_by N
def=
      local

      State:nat,KeynextPrev:symmetric_key,Nn:text,Hid:text,HidS
eqInit:text,Keynext:symmetric_key,SeqCtr:text
      init
            State := 0
      transition
            1. State=0 /\
RCV({Hid'.Keyret.Keynext'.Nn'.HidSeqInit'}_KeynextPrev') =|>
State':=1 /\ SeqCtr':=new() /\
SND({Keyret.Keynext'.HidSeqInit'.SeqCtr'}_Keyret)
end role
```

FIGURE 3. The role of the gateway.

```
role
role_G(G:agent,N:agent,KeynextPrev:symmetric_key,SND,RCV:chann
el(dy))
played_by G
def=
      local

      State:nat,Nn:text,Hid:text,Keyret:symmetric_key,HidSeqIni
t:text,Keynext:symmetric_key,SeqCtr:text
      init
            State := 0
      transition
            1. State=0 /\ RCV(start) =|> State':=1 /\
HidSeqInit':=new() /\ Nn':=new() /\ Keynext':=new() /\
Keyret':=new() /\ Hid':=new() /\
SND({Hid'.Keyret'.Keynext'.Nn'.HidSeqInit'}_KeynextPrev)
            2. State=1 /\
RCV({Keyret.Keynext.HidSeqInit.SeqCtr'}_Keyret) =|> State':=2
end role

role
role_N(G:agent,N:agent,Keyret:symmetric_key,SND,RCV:channel(dy
))
played_by N
def=
      local

      State:nat,KeynextPrev:symmetric_key,Nn:text,Hid:text,HidS
eqInit:text,Keynext:symmetric_key,SeqCtr:text
      init
            State := 0
      transition
            1. State=0 /\
RCV({Hid'.Keyret.Keynext'.Nn'.HidSeqInit'}_KeynextPrev') =|>
State':=1 /\ SeqCtr':=new() /\
SND({Keyret.Keynext'.HidSeqInit'.SeqCtr'}_Keyret)
end role
```

FIGURE 4. The role of the node.

to compute the session key. Every sensor needs to be deployed physically, and at that time the initialization stage can be performed in easy steps under a controlled supervised manner. Once the initialization phase is complete these two parameters are not used any more in the computation. Session

```
role
session1(KeynextPrev:symmetric_key,G:agent,N:agent,Keyret:symm
etric_key)
def=
      local
            SND2,RCV2,SND1,RCV1:channel(dy)
      composition
            role_N(G,N,Keyret,SND2,RCV2) /\
role_G(G,N,KeynextPrev,SND1,RCV1)
end role

role
session2(KeynextPrev:symmetric_key,G:agent,N:agent,Keyret:symm
etric_key)
def=
      local
            SND2,RCV2,SND1,RCV1:channel(dy)
      composition
            role_N(G,N,Keyret,SND2,RCV2) /\
role_G(G,N,KeynextPrev,SND1,RCV1)
end role

role environment()
def=
      const
            gateway:agent,keyret:symmetric_key,node:agent,hash_
0:hash_func,keynextprev:symmetric_key,kis:symmetric_key
      intruder_knowledge = {gateway,node,kis}
      composition
            session2(keynextprev,gateway,i,kis) /\
session1(keynextprev,gateway,node,keyret)
end role
```

FIGURE 5. The session among the participants.

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL

/home/span/span/testsuite/results/ProteanFormalVerification.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 1 states
  Reachable  : 0 states
  Translation: 0.01 seconds
  Computation: 0.00 seconds
```

FIGURE 6. Final results.

duration can be set based on the criticality of the application. In the case of mission-critical operations like a nuclear plant control system, it is important that the session duration is kept to the minimal so that the encryption keys are recycled frequently. In less critical systems like an automated irrigation control system, the session duration can be set to a much longer duration.

A. SCALABILITY

Our scheme is scalable and supports dynamic inclusion of additional nodes to the network. Each node is preconfigured with its hardware ID and also the unique initialization

vector in the EEPROM. Similar to how MAC addresses are assigned, a range of Protean hardware ID's and initialization vectors can be assigned to each of the manufacturers who can subsequently burn them into the EEPROM at the time of manufacturing. Corresponding edge node credentials are also stored at the gateway during the initial network configuration where the edge nodes register themselves with the gateway. When the edge node is added to the network for the first time, the initialization process can be conducted under supervision, and thereafter none of the initialization parameters are used anywhere in the subsequent authentication cycle.

B. ABILITY TO RESIST REPLAY AND KNOWN KEY ATTACK

Replay attacks usually happen when an authentication ack packet is captured and transmitted again back to the gateway over the air. The Protean authentication scheme ensures that there is no single static key dependency throughout the life cycle of a device in the network. The presence of a NONCE would ensure that replay attacks are thwarted. Keys are computed at the gateway and is encrypted for an exchange through the keys generated and exchanged with the edge nodes in the previous cycle and stored securely at the edge node. Only for the initial handshaking process alone, a pre-initialized value is used at the edge node for generating the decryption and encryption keys. Thereafter neither the initialization vector nor the hardware ID is used anymore to generate the keys which are thereafter generated in an incremental manner. SeqInit and SeqCntr both offer additional security against the replay attack because the sequences should be in an incremental manner within a session and any replayed packets will be dropped from further processing at the gateway.

When an auth_ack packet is received at the gateway it is flagged at the auth table as already authenticated based on the nonce value that is exchanged. A replay attack with the same packet will be rejected as the nonce is already used for authentication. Even a random nonce is passed as part a crafted packet it will be rejected because the nonce entry is flagged as already authenticated in the auth table maintained at the gateway.

C. RESISTANCE AGAINST DEVICE IMPERSONATION ATTACK

An impersonation attack indicates that a malicious attacker may try to masquerade as a genuine edge node. If the attacker wants to masquerade as a genuine edge node then the attacker will need to forge the Auth_Ack_message ⟨⟨h(Nn), Ekey(Keyret, Keynext, Hid_SeqInit,SeqCtr) ⟩⟩ sent from the edge node to the gateway. First, the attacker should get access to the session key stored at the edge node to disassemble this message, then obtain the SeqCtr to increment it by one. With this information, the attacker builds the acknowledgement packet and sends it back to the gateway. Unless the sequence is in the correct incremental order, the gateway will not accept the packet from an edge node. Hence our scheme can resist the device impersonation attack.

D. RESISTANCE AGAINST DEVICE CLONING ATTACK

The Initialization vector is based on the hardware ID which is unique and an immutable identity. Hence compromising a single edge node will not affect the secure communication between other nodes of the network. Gateway hosts an application behavior analytics modules which are an internal IoT LAN IDS that would help identify suspicious behavior from nodes captured by an attacker. Besides, since the keys are rotated at a static or random time window it becomes cumbersome to trap the keys and re-establish a session with the gateway. The SeqInit and SeqCntr also provide extra security against device cloning or device capture attacks as the attacker will find it very difficult to generate the packets in synch with the genuine node. Every packet that is exchanged between the edge node and the gateway is encrypted with the current session key that is valid only for the genuine device.

E. RESISTING EAVESDROPPING ATTACK

There are multiple ways of how an attacker can access the session keys. Eavesdropping the authentication and/or acknowledgement packets or by reading the memory of a captured device on a constant basis. Auth and Auth_ack packets captured will not give useful information to an attacker as they are encrypted. An attacker is able to capture a node and constantly extracting the stored previous Keyret and previous Keynext can decrypt an incoming authentication packet from the gateway and extract the keys for the following cycle. The auth_ack packet from the edge node is send back to the gateway without any time delay. As soon as the gateway receives the ack packet it updates its auth table and the nonce is flagged as already authenticated. Even if an attacker crafts another packet with the same nonce the new device will not get authenticated. If the attacker uses a random nonce, then the auth table will not have a corresponding nonce entry in it. And hence, the packet will be rejected so that rogue device will not be authenticated and added to the network.

F. MESSAGE CONFIDENTIALITY, INTEGRITY AND TIMELINESS

Our scheme encrypts the data generated by the edge nodes using dynamically changing keys during a session which is generated during each authentication cycle. In the initialization phase, both the gateway and the edge devices uses their unique hardware ID and manufacturer specific initialization vector to compute the initial encryption key. Also, Keyret and Keynext are computed using hardware id and a random nonce. During the installation phase these parameters are exchanged in a closed environment and there after none of these credentials need to be used or stored at the edge node. It can be left at the edge node by encrypting it with the encryption key (Ekey) generated during the initialization phase. Ekey from initialization phase can be stored at the gateway which can be then used in case of a reconfiguration of the entire network. Subsequent authentication phases use the keys from the previous cycle, Nonce and random numbers

to compute the new set of keys used for encryption. Since the keys change for each authentication cycle, it is not easy for an intruder to extract the keys and reuse them.

Authentication cycle can be timed so that gateways can invalidate an acknowledgement cycle if it exceeds the set time limit. Maintaining a clock at the edge node is expensive on the power consumption side of resources. Instead, the gateway will evaluate the timeliness of the acknowledgment packet received even before validating the Nonce against its entry in the auth table. Any attempts to tamper with the authentication cycle will therefore be thwarted through this process.

G. NODE CAPTURE OR MAN IN THE MIDDLE ATTACKS

The node capture attacks or man-in-the-middle attacks will not be effective with this scheme. Capturing the encrypted auth packet and using the same packet to authenticate a rogue device would be ineffective. This is because in order to encrypt the auth acknowledgment packet sent by the edge node in response to an auth packet sent by the gateway, the keys can only be extracted by decrypting the incoming auth packet. Gateway computes a new session key with the help of the current active session key and the NONCE/random number combination. Even if an attacker manages to break into the secure zone where the current valid encryption key is stored and steal the same, it cannot be reused in a rogue device. This is because the auth ack packet from the captured device sent back to the gateway will complete the authentication cycle by updating the ACK flag. Reusing the same packet from a rogue node will result in rejecting the rogue device as the ACK flag for the corresponding $h(N_n)$ is already updated. Furthermore, only a single beacon is transmitted between the sender and receiver to exchange authentication/acknowledgement credentials.

H. LIGHTWEIGHT SCHEME ON EDGE NODES

The Protean scheme offloads a lot of the key computation, random number and nonce generation to the gateway in order to keep the overall computational overheads at the edge node to minimal. This conserves a lot of energy at the edge node which is one of the most constrained devices in the overall IoT ecosystem. Edge node uses key values already exchanged during the previous auth cycle to decrypt and then uses the keys from the current cycle to subsequently encrypt the acknowledgement packet. Edge nodes would only be engaged in decryption and encryption of packets. This ensures that edge nodes can continue to operate within the restricted resources. The auth table has entries for each acknowledgment cycle. For each SYN packet sent the SYN flag is set for that entry and gateway will update the ACK flag in the auth table, when an ACK is received from the node and authentication cycle is flagged successful. In the case of an Auth SYN/ACK packet loss, the process will repeat itself for a predefined number of cycles in an attempt to complete the auth cycle.

Each edge node will have a unique hardware id and initialization vector to start with. Hence the keys generated by the edge nodes would be unique. Even if an attacker manages to compromise a node and extract its credentials through brute force attack, little damage could be done to the rest of the edge nodes unless each one of them is accessed physically and compromised.

In the case of a Zigbee based network, the network related information in a given packet is also encrypted so that device serial id or MAC along with the PAN ID is not exchanged in the open. Without the knowledge of these crucial network parameters, it is difficult to inject a rogue device into a given IoT LAN. Together with this, the Protean authentication scheme will ensure that the wiretapped secret key attacks will prove to be futile against the proposed model.

VIII. EXPERIMENTS AND RESULTS

Most of the edge nodes are battery powered. Hence every additional computation will cost the node extra power which is highly valued especially in critical infrastructures. The very purpose of the tests performed in this section is to prove that the computation proposed to run on the edge nodes is not bringing about a noticeable increase in battery consumption. The test environment consisted of a coordinator/gateway and end-devices configured to run either as a router or as an edge node. An Xbee S2 RF module was set as the coordinator. It was connected over a Radxa development board.

The Xbee S2 XB24-Z7WIT-004 is used in both the router and end-devices It is connected to a low-cost MOTE built in-house with the Atmega328P as the microcontroller. MOTEs come with the provision to slot Xbee module on the top and four sensor modules onto the sides through USB interface as shown in Fig. 7.

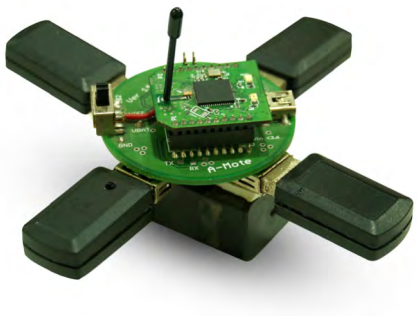

FIGURE 7. In-house MOTE used for testing.

This research made use of in-house custom hat boards that were attached to the top of the radxa board as an extension hatboard as shown in Fig, 6. The hatboard consists of the ZigBee adaptor interface and interfaces for other protocol adaptors like 6LoWPAN, BLE, WiFi etc. With the help of the hatboard and the supporting patented Gateway Software "Protocol Characterization Layer" [34] it was possible to support multiple communication protocols on the same gateway. The gateway consisted of a Radxa development board with the following specification: ARM Cortex-A9 quad-core at 1.6Ghz, 2GB DDR3 memory at 800Mhz, an 8GB Nand Flash for storage with the Micro-SD SDXC interface, 80 pin

connector with 2.54mm header including GPIO, I2C, SPI, line in, USB 2.0, PWM, ADC, on board FPC with the support of LVDS, TP and CSI.

End-devices can operate either as an edge node or as a router. When an end-device is configured as a router, it functions both as a sensing device and as a routing device. With this capability, such end-devices could allow data from other routers and edge nodes to hop through them. When configured as routers such devices cannot go into a sleep state because they become pivotal nodes in handling the vital routing functionality of the network. Routing expects the device to be in awake state at all times. In contrast, if the end-device is configured to run as an edge node, then it will act as a mere sensing end-device with no routing capabilities. Such devices can be configured to stream continuously or in request response mode or to transmit only when there is a change in state. In the latter two states, the device goes to sleep and wakes up only when prompted. This preserves energy and keeps the devices alive for a longer period of time.

In order to test the authentication scheme, the MOTEs were set up as routers as well as edge nodes. When a MOTE acts as a router it will never go to sleep and will constantly radiate sensor values; when configured as edge nodes, it wakes up once every 320 ms and waits for five seconds looking out for any requests coming from the coordinator node. The coordinator node sends out a series of requests once in four minutes requesting for sensed parameters. MOTEs will then transmit the sensed values to the coordinator and go back to sleep. Experiments were conducted using the above mentioned scenarios with and without the authentication code running on them. Edge nodes and routers are kept at line-of-sight with the coordinator so that they all send data directly to the coordinator/gateways without any involvement of intermediate routers. The gateway is connected to the direct AC while the edge nodes run out of 3v DC supply.

The voltage drainage, the current in milliampere (mA) and the current in ampere (A) were measured based on which power was computed (Voltage multiplies Current in A) as depicted in Table 1 and Table 2. Table 1 and Table 2 shows the readings for all the above mentioned parameters when executed with and without authentication code while configuring the end nodes as routers with no sleeps. Table 3 and table 4 show the readings for all the above mentioned parameters when executed with and without authentication code while configuring the end nodes as edge nodes with sleeps. Table 5 summarizes the student t-test performed on voltage drainage, current drainage and power with and without authentication code while the end node is configuring as a router and edge node. According to the results, none of the cases the mean value showed any substantial difference when executed with and without the authentication protocol.

We collected the results for the situations when the MOTEs were set as the following modes:

- Routers with authentication code, continued to remain alive till the voltage dropped to 2.44V
  - mAh: 97.74193548 [average current(mA) multiplies working time(h)(48.87096774 multiplies 2)] as shown in Table 1.
- Routers without the authentication code, continued to remain alive till the voltage dropped to 2.43V
  - mAh: 97.52903226 [average current(mA) multiplies working time(h)(48.76451613 multiplies 2)] as shown in Table 2.
- End-devices with authentication code, continued to remain alive till the voltage dropped to 2.51V
  - mAh: 150.3764706 [average current(mA) multiplies working time(h)(50.1254902 multiplies 3)] as shown in Table 3.

TABLE 1. Voltage drop, current in mA & A, power in watts in a gateway-router setup with authentication.

Time – Voltage Drainage (Gateway – Router) With Authentication				
Time	**Voltage Drainage (Volts)**	**Current (mA)**	**Current (A)**	**Power (Watt)**
1	3	50.9	0.0509	0.1527
1.04	2.66	49.4	0.0494	0.131404
1.08	2.64	49.2	0.0492	0.129888
1.12	2.64	49.2	0.0492	0.129888
1.16	2.64	49.2	0.0492	0.129888
1.2	2.63	49.2	0.0492	0.129396
1.24	2.63	49.2	0.0492	0.129396
1.28	2.63	49.2	0.0492	0.129396
1.32	2.62	49.2	0.0492	0.128904
1.36	2.6	49.1	0.0491	0.12766
1.4	2.6	48.9	0.0489	0.125184
1.44	2.55	48.8	0.0488	0.12444
1.48	2.54	48.7	0.0487	0.123698
1.52	2.53	48.7	0.0487	0.123211
1.56	2.52	48.7	0.0487	0.122724
2	2.52	48.7	0.0487	0.122724
2.04	2.52	48.7	0.0487	0.122724
2.08	2.51	48.6	0.0486	0.121986
2.12	2.51	48.6	0.0486	0.121986
2.16	2.5	48.6	0.0486	0.1215
2.2	2.49	48.5	0.0485	0.120765
2.24	2.49	48.5	0.0485	0.120765
2.28	2.48	48.4	0.0484	0.120032
2.32	2.47	48.3	0.0483	0.119301
2.36	2.46	48.3	0.0483	0.118572
2.4	2.46	48.2	0.0482	0.118572
2.44	2.45	48.2	0.0482	0.11809
2.48	2.45	48.2	0.0482	0.11809
2.52	2.44	48.1	0.0481	0.117364
2.56	2.43	48.1	0.0481	0.116883
3	2.43	48.1	0.0481	0.116883
Average Current (mA) = 48.76451613 mAh = 97.52903226				

TABLE 2. Voltage drop, current in mA & A, power in watts in a gateway-router setup without authentication.

Time – Voltage Drainage (Gateway – Router) Without Authentication				
Time	Voltage Drainage (Volts)	Current (mA)	Current (A)	Power (Watt)
1	3	50.9	0.0509	0.1527
1.04	2.89	50.4	0.0504	0.145656
1.08	2.84	50,2	0.0502	0.142568
1.12	2.7	49.5	0.0495	0.13365
1.16	2.7	49.5	0.0495	0.13365
1.2	2.66	49.3	0.0493	0.131138
1.24	2.66	49.3	0.0493	0.131138
1.28	2.66	49.3	0.0493	0.131138
1.32	2.65	49.3	0.0493	0.131138
1.36	2.64	49.2	0.0492	0.129888
1.4	2.58	48.9	0.0489	0.126162
1.44	2.56	48.8	0.0488	0.124928
1.48	2.55	48.8	0.0488	0.12444
1.52	2.55	48.8	0.0488	0.12444
1.56	2.54	48.7	0.0487	0.123698
2	2.54	48.7	0.0487	0.123698
2.04	2.53	48.6	0.0486	0.122958
2.08	2.52	48.6	0.0486	0.122472
2.12	2.52	48.6	0.0486	0.122472
2.16	2.51	48.5	0.0485	0.121735
2.2	2.5	48.5	0.0485	0.12125
2.24	2.5	48.5	0.0485	0.12125
2.28	2.49	48.4	0.0484	0.120516
2.32	2.48	48.3	0.0483	0.119784
2.36	2.47	48.3	0.0483	0.119301
2.4	2.47	48.3	0.0483	0.119301
2.44	2.46	48.2	0.0482	0.118572
2.48	2.46	48.2	0.0482	0.118572
2.52	2.45	48.2	0.0482	0.11809
2.56	2.44	48.1	0.0481	0.117364
3	2.44	48.1	0.0481	0.117364
Average Current (mA) = 48.87096774 mAh = 97.74193548				

- End-devices without the authentication code, continued to remain alive till the voltage dropped to 2.50V
  - mAh: 149.60588235[average current(mA) multiplies working time(h)(49.86862745 multiplies 3)] as shown in Table 4.

Five MOTEs were used in carrying out this experiment. Two 1.5v, 200mAh AA batteries were used in each MOTE. The following test cycles were performed. Each cycle started with a brand new battery of the same make and model. As batteries drain out, five such cycles were performed each time with brand new batteries. In each such cycles batteries were run from its peak voltage to the very minimal voltage

TABLE 3. Voltage drop, current in mA & A, power in watts in a gateway-end device setup with authentication.

Time – Voltage Drainage (Gateway – End Device) With Authentication				
Time	Voltage Drainage (Volts)	Current (mA)	Current (A)	Power (Watt)
1	3	51.3	0.0513	0.1539
1.04	2.93	50.9	0.0509	0.149137
1.08	2.92	50.8	0.0508	0.148336
1.12	2.91	50.7	0.0507	0.147537
1.16	2.91	50.7	0.0507	0.147537
1.2	2.9	50.6	0.0506	0.14674
1.24	2.88	50.6	0.0506	0.145728
1.28	2.88	50.6	0.0506	0.145728
1.32	2.87	50.6	0.0506	0.145222
1.36	2.85	50.5	0.0505	0.143925
1.4	2.85	50.5	0.0505	0.143925
1.44	2.85	50.5	0.0505	0.143925
1.48	2.83	50.4	0.0504	0.143632
1.52	2.81	50.3	0.0503	0.141343
1.56	2.81	50.3	0.0503	0.141343
2	2.79	50.2	0.0502	0.140058
2.04	2.79	50.2	0.0502	0.140058
2.08	2.78	50.1	0.0501	0.139278
2.12	2.77	50.1	0.0501	0.138777
2.16	2.76	50.1	0.0501	0.138276
2.2	2.76	50.1	0.0501	0.138276
2.24	2.75	50	0.050	0.1375
2.28	2.74	50	0.050	0.137
2.32	2.74	50	0.050	0.137
2.36	2.73	49.9	0.0499	0.136227
2.4	2.72	49.9	0.0499	0.135728
2.44	2.71	49.8	0.0498	0.134958
2.48	2.71	49.8	0.0498	0.134958
2.52	2.7	49.8	0.0498	0.13446
2.56	2.69	49.7	0.0497	0.133693
3	2.69	49.7	0.0497	0.133693
3.04	2.68	49.7	0.0497	0.133196
3.08	2.67	49.6	0.0496	0.132432
3.12	2.66	49.6	0.0496	0.131936
3.16	2.65	49.5	0.0495	0.131175
3.2	2.64	49.5	0.0495	0.13068
3.24	2.64	49.5	0.0495	0.13068
3.28	2.64	49.5	0.0495	0.13068
3.32	2.63	49.4	0.0494	0.129922
3.36	2.62	49.4	0.0494	0.129428
3.4	2.61	49.3	0.0493	0.128673
3.44	2.6	49.3	0.0493	0.12818
3.28	2.59	49.2	0.0492	0.127428
3.52	2.58	49.2	0.0492	0.126936
3.56	2.56	49.1	0.0491	0.125696
4	2.55	49	0.049	0.12485
4.04	2.54	48.9	0.0489	0.124206
4.08	2.53	48.8	0.0488	0.123464
4.12	2.52	48.8	0.0488	0.122976
4.16	2.51	48.7	0.0487	0.122237
4.2	2.5	48.6	0.0486	0.1215

TABLE 4. Voltage drop, current in mA & A, power in watts in a gateway-end device setup without authentication.

Time – Voltage Drainage (Gateway – End Device) Without Authentication				
Time	Voltage Drainage (Volts)	Current (mA)	Current (A)	Power (Watt)
1	3	51.3	0.0513	0.1539
1.04	2.95	51.2	0.0512	0.15104
1.08	2.94	51.1	0.0511	0.150234
1.12	2.93	51	0.051	0.14943
1.16	2.93	51	0.051	0.14943
1.2	2.92	50.9	0.0509	0.148678
1.24	2.89	50.8	0.0508	0.146812
1.28	2.89	50.8	0.0508	0.146812
1.32	2.89	50.8	0.0508	0.146812
1.36	2.88	50.7	0.0507	0.146016
1.4	2.87	50.7	0.0507	0.145509
1.44	2.86	50.6	0.0506	0.144716
1.48	2.85	50.6	0.0506	0.14421
1.52	2.84	50.5	0.0505	0.14342
1.56	2.83	50.5	0.0505	0.142915
2	2.82	50.4	0.0504	0.142128
2.04	2.81	50.4	0.0504	0.141624
2.08	2.8	50.4	0.0504	0.14112
2.12	2.79	50.3	0.0503	0.140337
2.16	2.78	50.3	0.0503	0.139834
2.2	2.77	50.3	0.0503	0.139331
2.24	2.76	50.2	0.0502	0.138552
2.28	2.75	50.2	0.0502	0.1385
2.32	2.75	50.2	0.0502	0.1385
2.36	2.74	50.1	0.0501	0.137274
2.4	2.74	50.1	0.0501	0.137274
2.44	2.73	50	0.050	0.1365
2.48	2.72	50	0.050	0.136
2.52	2.72	50	0.050	0.136
2.56	2.71	50	0.050	0.1355
3	2.7	49.9	0.0499	0.13473
3.04	2.69	49.9	0.0499	0.134231
3.08	2.69	49.9	0.0499	0.134231
3.12	2.68	49.8	0.0498	0.133664
3.16	2.67	49.8	0.0498	0.132966
3.2	2.66	49.8	0.0498	0.132468
3.24	2.66	49.8	0.0498	0.132468
3.28	2.66	49.8	0.0498	0.132468
3.32	2.65	49.7	0.0497	0.131705
3.36	2.64	49.7	0.0497	0.131208
3.4	2.63	49.6	0.0496	0.130448
3.44	2.62	49.6	0.0496	0.129952
3.28	2.61	49.6	0.0496	0.129456
3.52	2.59	49.5	0.0495	0.128205
3.56	2.58	49.5	0.0495	0.12771
4	2.57	49.4	0.0494	0.126958
4.04	2.56	49.4	0.0494	0.126464
4.08	2.55	49.3	0.0493	0.125715
4.12	2.53	49.1	0.0491	0.124223
4.16	2.52	49	0.049	0.122448
4.2	2.51	48.9	0.0489	0.122739
Average Current (mA) = 50.1254902 mAh = 150.3764706				

TABLE 5. Student t-test.

Student t-test		
Variables	Mean (Gateway - Router)	Mean (Gateway -Edge Node)
Voltage Drainage without Auth	2.579354839	2.741764706
Voltage Drainage with Auth	2.548387097	2.724509804
Current Drainage without Auth	48.87096774	50.1254902
Current Drainage with Auth	48.76451613	49.86862745
Power Drop without Auth	0.12616229	0.137504843
Power Drop with Auth	0.124330968	0.135946529

required to power the MOTEs which is in the range of 2.4 to 2.5V. An average of all these readings were taken as shown in Fig. 5.

- Each of the five MOTEs operates without the authentication scheme but configured as edge device.
- Each of the five MOTEs operates with the authentication scheme but configured as edge device.
- Each of the five MOTEs operates with the authentication scheme but configured as router.
- Each of the five MOTEs operates with the authentication scheme but configured as router.

Tables 1, 2, 3 and 4 depict the voltage drainage, current in milliamps and in amps and power in watts of the MOTEs with and without authentication scheme when the network is configured with one coordinator and five routers and with one coordinator and five edge nodes respectively. Routers will remain awake throughout the cycle and will stream continuously the sensed parameters. During the initial stage voltage drainage is slightly higher for authentication enabled run, but gradually voltage drainage shows uniformity. Readings look the same for current measured in mA and power computed in Watts. A student t-test performed on the data as shown in Table 2 depicts that the observed difference in sample means is not convincing enough to say that the average voltage drops with and without authentication differ significantly. In both scenarios, battery depletion reaches within the minimal operational voltage within three hours. Values in Table 3 shows the voltage drainage, current in milliamps and in amps and power in watts of the MOTE with and without authentication with the network configured to have one Zigbee coordinator and five end-devices configured as edge node with sleep enabled. Since the sleep interval is greater, the battery lasted longer than the previous scenario. Battery drainage is pretty much even in MOTEs that run with and without the authentication scheme. In both scenarios, battery depletion reached within the minimal operational voltage within around four and a half hours. A student t-test performed on the data as shown in table 4 depicts that the

FIGURE 8. Radxa board with custom built hat board.

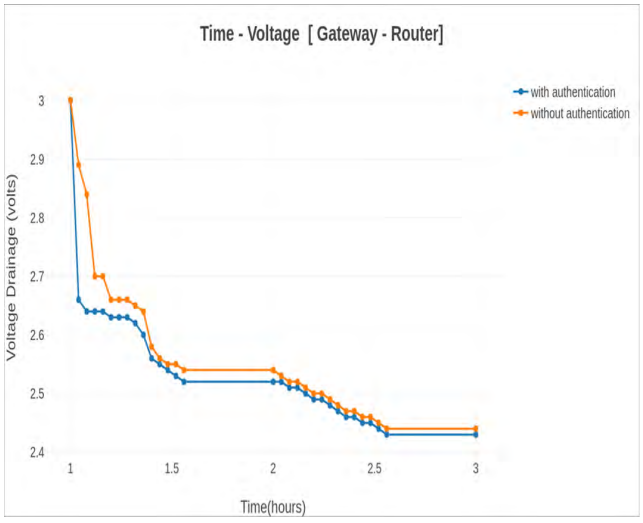

FIGURE 9. Time to voltage drain with and without the authentication code in gateway-router setup.

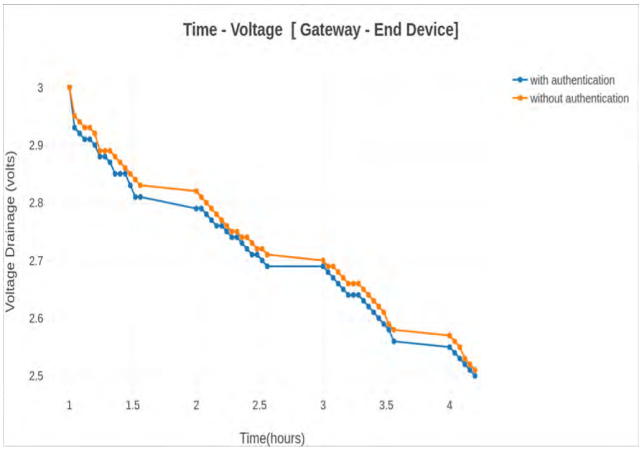

FIGURE 10. Time to voltage drain with and without the authentication code in gateway-edge node setup.

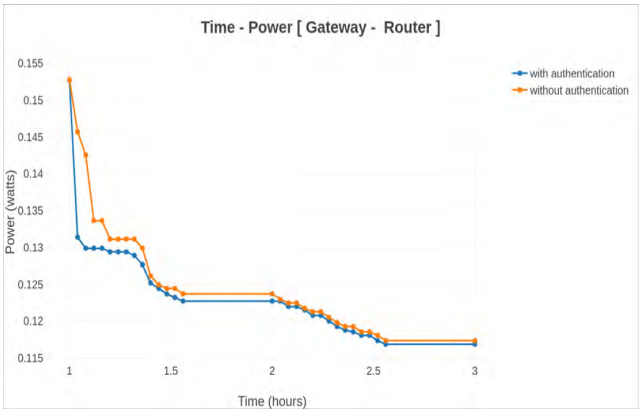

FIGURE 11. Time to power drain with and without the authentication code in gateway-router setup.

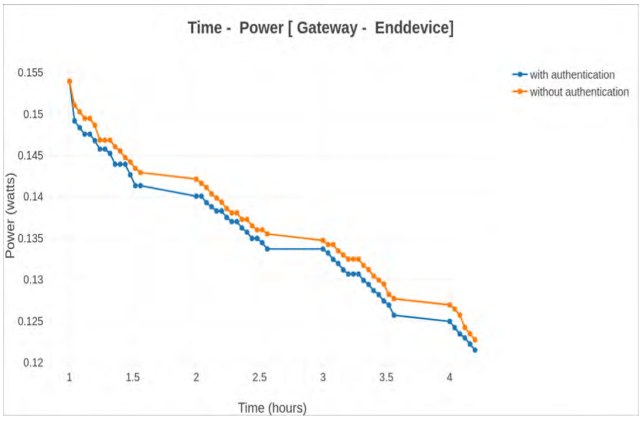

FIGURE 12. Time to power drain with and without the authentication code in gateway-edge node setup.

observed difference in sample means is convincing enough to say that the average voltage drops with and without authentication differ significantly.

End-devices are configured as a router and also as edge nodes. A router acts as an edge node and also as a medium for other edge nodes to hop its data to the nearest gateway. Under such circumstances the router would never go to sleep. The end-device is also configured to be an edge node and responds to requests from gateway. It would respond to a request from a gateway and would go to sleep saving more energy. Charts shown in Fig. 9, 10, 11 and 12 give a clear understanding that from both voltage drop and power utilization with and without authentication code is not significant enough to be

an overhead. Hence, the scheme stands tall in establishing the fact that the authentication scheme consumes negligible power but still was able to establish solid lightweight authentication mechanism between the gateways and the edge nodes.

IX. CONCLUSION AND FUTURE WORK

In the current setup of IoT deployment, the edge nodes depend on the encryption keys for secure communications. These keys are usually stored locally at the edge nodes or selected from a set of keys which are maintained and managed by a trusted distribution center. Alternatively, the encryption keys must be asymmetric keys generated by a third party KDC or through complex mathematical operations, both of which are computationally expensive on constrained IoT devices. To address these shortcomings, we developed and deployed an efficient key generation mechanism, where the authentication credentials are generated on the fly and exchanged securely between the gateway and edge nodes with minimal computational overheads on resource constrained IoT devices. Like many existing authentication schemes, our proposed mechanism does not require storing any static key value on the device. Instead, the keys are dynamically changing and shared securely to prevent message replay and device clone attacks. These two attacks are commonly seen on the IoT devices deployed in outside fields. We conducted experiments to verify the efficiency of our scheme. Our experimental results show no significant difference in terms of power drainage on the IoT devices.

For future work, we would extend this scheme to authenticate the inter-cluster and intra-cluster nodes with the potential use of distributed hyper-ledger techniques. Moreover, we would explore the intrusion detection for the IoT devices. That is, when a device is compromised, we apply anomaly detection on the gateway. The anomaly detection consists of two components; the first one detects anomalies within an IoT LAN and the second one detects anomalies of the inbound and outbound network traffic. The two detection components use different technologies including the network traffic profiling of the applications and the dynamic firewall, respectively.

REFERENCES

[1] S. Satyadevan, B. S. Kalarickal, and M. K. Jinesh, "Security, Trust and implementation limitations of prominent IoT platforms," in *Proc. FICTA*, Oct. 2014, pp. 85–95.
[2] N. Park and N. Kang, "Mutual authentication scheme in secure Internet of Things technology for comfortable lifestyle," *Sensors*, vol. 16, no. 1, p. 20, Oct. 2015.
[3] C. B. D.-K. E. Editors, *Online: Authentication in IOT Nodes Will Demand SHA and AES Solutions*. Europe: Digi-Key's European Editors, 2015.
[4] E. Asanghanwa and R. Ih, "Online: Security for intelligent, connected iot edge nodes," Atmel, Addison, TX, USA, White Paper 112015, 2016.
[5] S. Sathyadevan, S. Prabhakaranl, and K. Bipin, "A survey of security protocols in WSN and overhead evaluation," in *Proc. 3rd Int. Conf. Frontiers Intell. Comput., Theory Appl. (FICTA)*, Oct. 2015, pp. 729–738.
[6] S. Sathyadevan, K. Achuthan, and J. Poroor, "Architectural recommendations in building a network based secure, scalable and interoperable Internet of Things middleware," in *Proc. 3rd Int. Conf. Frontiers Intell. Comput., Theory Appl. (FICTA)*, Oct. 2014, pp. 429–439.
[7] O. Olawumi, K. Haataja, M. Asikainen, N. Vidgren, and P. Toivanen, "Three practical attacks against ZigBee security: Attack scenario definitions, practical experiments, Countermeasures, and lessons learned," in *Proc. 14th Int. Conf. Hybrid Intell. Syst.*, Dec. 2014, pp. 199–206.
[8] T. Zillner and S. Strobl, *Zigbee Exploited: The Good the Bad and the Ugly*. CA, USA: Blackhat, 2015.
[9] B. Stelte and G. D. Rodosek, "Thwarting attacks on ZigBee—Removal of the KillerBee stinger," in *Proc. 9th Int. Conf. Netw. Service Manage.*, Oct. 2013, pp. 219–226.
[10] G. Dini and M. Tiloca, "Considerations on security in ZigBee networks," in *Proc. IEEE Int. Conf. Sensor Netw., Ubiquitous, Trustworthy Comput.*, Jun. 2010, pp. 58–65.
[11] K. Zhao and L. Ge, "A survey on the Internet of Things security," in *Proc. 9th Int. Conf. Comput. Intell. Secur.*, Dec. 2013, pp. 663–667.
[12] S. Mangard, and F. X. Standaert, "Cryptographic hardware and embedded systems, CHES 2010," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Aug. 2016, p. 577.
[13] K. R. B. Butler, S. Ryu, P. Traynor, and P. D. McDaniel, "Leveraging identity-based cryptography for node ID assignment in structured P2P systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 12, pp. 1803–1815, Dec. 2009.
[14] J. Park, S. Shin, and N. Kang, "Mutual authentication and key agreement scheme between lightweight devices in Internet of Things," *J. Korean Inst. Commun. Inf. Sci.*, vol. 38, no. 9, pp. 707–714, 2013.
[15] V. Kimlaychuk, "Security in ad-hoc sensor networks with pre-loaded time limited memory keys," in *Proc. 4th Int. Conf. Biomed. Eng. Inform. (BMEI)*, Oct. 2011, pp. 2067–2071.
[16] P. Kumar, A. Gurtov, J. Iinatti, M. Ylianttila, and M. Sain, "Lightweight and secure session-key establishment scheme in smart home environments," *IEEE Sensors J.*, vol. 16, no. 1, pp. 254–264, Jan. 2016.
[17] J. J. Huang, W. S. Juang, C. I. Fan, and Y. F. Tseng, "Lightweight authentication scheme with dynamic group members in IoT environments," in *Proc. 13th Int. Conf. Mobile Ubiquitous Syst., Comput. Netw. Services*, Nov. 2016, pp. 88–93.
[18] J. Lee, Y. Sung, and J. Park, "Lightweight sensor authentication scheme for energy efficiency in ubiquitous computing environments," *Sensors*, vol. 16, no. 12, p. 2044, 2016.
[19] P. Porambage, A. Braeken, P. Kumar, A. Gurtov, and M. Ylianttila, "Proxy-based end-to-end key establishment protocol for the Internet of Things," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, Jun. 2015, pp. 2677–2682.
[20] Y. Sharaf-Dabbagh and W. Saad, "On the authentication of devices in the Internet of Things," in *Proc. IEEE 17th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2016, pp. 1–3.
[21] M. S. Farash, M. Turkanovic, S. Kumari, and M. Hölbl, "An efficient user authentication and key agreement scheme for heterogeneous wireless sensor network tailored for the Internet of Things environment," *Ad Hoc Netw.*, vol. 36, pp. 152–176, Jan. 2016.
[22] Y. Lu, L. Li, H. Peng, and Y. Yang, "An energy efficient mutual authentication and key agreement scheme preserving anonymity for wireless sensor networks," *Sensors*, vol. 16, p. 837, Mar. 2016.
[23] M. H. Ibrahim, "Octopus: An edge-fog mutual authentication scheme," *Int. J. Netw. Secur.*, vol. 18, no. 6, pp. 1089–1101, Nov. 2016.
[24] J. Großschädl, A. Szekely, and S. Tillich, "The energy cost of cryptographic key establishment in wireless sensor networks," in *Proc. 2nd ACM Symp. Inf., Comput. Commun. Secur.*, Mar. 2007, pp. 380–382.
[25] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of IoT systems: Design challenges and opportunities," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2014, pp. 417–423.
[26] V. L. Shivraj, M. A. Rajan, M. Singh, and P. Balamuralidhar, "One time password authentication scheme based on elliptic curves for Internet of Things (IoT)," in *Proc. 5th Nat. Symp. Inf. Technol., Towards New Smart World (NSITNSW)*, Riyadh, Saudi Arabia, Feb. 2015, pp. 1–6.
[27] P. N. Mahalle, N. R. Prasad, and R. Prasad, "Threshold cryptography-based group authentication (TCGA) scheme for the Internet of Things (IoT)," in *Proc. IEEE VITAE*, May 2014, pp. 1–5.
[28] P. Porambage, C. Schmitt, P. Kumar, A. Gurtov, and M. Ylianttila, "Two-phase authentication protocol for wireless sensor networks in distributed IoT applications," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2014, pp. 2728–2733.
[29] T. Kothmayr, C. Schmitt, W. Hu, M. Brünig, and G. Carle, "DTLS based security and two-way authentication for the Internet of Things," *Ad Hoc Netw*, vol. 11, pp. 2710–2723, Nov. 2013.

[30] Y.-H. Chuang, N.-W. Lo, C.-Y. Yang, and S.-W. Tang, "A lightweight continuous authentication protocol for the Internet of Things," *Sensors*, vol. 18, no. 4, p. 1104, Apr. 2018.
[31] M. T. Hammi, E. Livolant, P. Bellot, A. Serhrouchni, and P. Minet, "MAC sub-layer node authentication in OCARI," in *Proc. Int. Conf. Perform. Eval. Modeling Wired Wireless Netw.*, Nov. 2016, pp. 1–6.
[32] J.-C. Chen and Y.-P. Wang, "Extensible authentication protocol (EAP) and IEEE 802.1x: Tutorial and empirical experience," *IEEE Commun. Mag.*, vol. 43, no. 12, pp. supl.26–supl.32, Dec. 2005. [Online]. Available: https://ieeexplore.ieee.org/document/1561920/authors
[33] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen, "AES implementation on a grain of sand," *IEE Proc.-Inf. Secur.*, vol. 152, no. 1, pp. 13–20, Oct. 2005.
[34] S. Sathyadevan, B. Kunjumon, H. P. Pillai, and K. Achuthan, "Intelligent IoT Gateway," U.S. Patent 10 057,382 B2, Aug. 21, 2018.

SHIJU SATHYADEVAN received the M.S. degree in information technology from the University of Western Sydney, Nepean, Australia. He was a Freelance BI Consultant working with leading organizations from multitude of business domains across U.S., Europe, Asia, and Australia. He is currently an Assistant Professor with the Amrita Centre for Cyber Security, Amrita Vishwa Vidyapeetham, Amritapuri Campus, mainly focusing on research pertaining to security enhancements in cloud computing, the Internet of Things architecture and security, and big data analytic platforms. He has published over 25 conference papers in the area of cloud security, the IoT, big data, and machine learning. He also holds three patents in the area of cloud security and the IoT.

KRISHNASHREE ACHUTHAN received the Ph.D. degree from Clarkson University, NY, USA. She currently heads the Center for Cybersecurity Systems and Networks and an Amrita Technology Business Incubator (Amrita TBI) at Amrita Vishwa Vidyapeetham. She is also the Dean of PG Programs at the Amrita School of Engineering, Coimbatore. She is an Ardent Researcher with multi-disciplinary interests. She holds 29 U.S. patents and has published widely in highly acclaimed international journals.

ROBIN DOSS received the B.Eng. degree from the University of Madras, India, in 1999, and the M.Eng. and Ph.D. degrees from the Royal Melbourne Institute of Technology (RMIT), Australia, in 2000 and 2004, respectively. He joined Deakin University, Melbourne, Australia, in 2003, where he is currently the A/Head of the School (Development and International). Since 2003, he has published more than 60 papers in refereed international journals, international conference proceedings, and technical reports for industry and government. His current research interests include communication systems, protocol design, wireless networks, security, and privacy.

LEI PAN received the Ph.D. degree in computer forensics from Deakin University, Australia, in 2008, where he is currently a Lecturer with the School of Information Technology. He has published more than 30 research papers in refereed international journals and conferences, such as the *IEEE Security & Privacy*, the *Journal of Multimedia*, and *Digital Investigation*. His research interests include cybersecurity and privacy.