EAP-Sens: A Security Architecture For Wireless Sensor Networks

M. Abdul Alim
The Computer Laboratory
University of Cambridge
15 JJ Thomson Avenue, Cambridge
United Kingdom CB3 0FD
Abdul.Alim@cl.cam.ac.uk

Behcet Sarikaya
Computer Science and Operations Research
University of Montreal
Montreal, Quebec
Canada H3C 3J7
sarikaya@iro.umontreal.ca

ABSTRACT

We present the design, implementation and simulation of a security protocol based on Extensible Authentication Protocol nick-named EAP-Sens for wireless sensor networks. We use the generalized pre-shared key authentication method for authentication and key establishment. Standard EAP model is used for authenticating sensor nodes within the radio range of the authenticator. For distant nodes, we use Relay Authenticators to tunnel EAP messages to the authenticator. We have defined EAP message encapsulations for IEEE 802.15.4 standard and a key hierarchy for EAP-Sens. To analyze EAP-Sens performance on larger configurations a simulation model has been developed. We have implemented an EAP-Sens prototype for Tmote sensors and showed that EAP based security protocol is feasible in wireless sensor networks.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security And Protection*

General Terms

Design, Security, Algorithms

Keywords

Wireless Sensor Network, Security, EAP, WPAN, GPSK

1. INTRODUCTION

Wireless sensor networking is one of the most exciting and challenging research areas of recent time. Deployment of wireless sensor networks is becoming more common in a wide variety of applications for collecting and disseminating sensitive and important information [5, 22, 26, 27, 18, 17, 6, 8]. As the application of wireless sensor networks is increasing, security and reliability become an important concern because any vulnerability in the system would limit its practical use. Like mobile ad hoc networks, multi-hop wireless sensor networks count on proper operation of devices that act as *coordinators* or *forwarders* [3]. Thus device or entity authentication during the early stage of network initialization or during operation later on is one of the most important design considerations. However, designing security protocols for resource-poor sensor devices is a difficult problem [21] [14] due to inherent complexities of cryptographic algorithms.

In recent years, a number of security protocols have been proposed for wireless sensor networks [21] [14] [25] [10] [13] [23] [29] [30] [20]. Though some authors have shown that public-key security protocols are feasible in wireless sensor networks, they are slow and require complex computation, and significant amount of memory [25] [28]. On the other hand, symmetric-key methods are simple, fast, and require less memory and computing power. Moreover, the IEEE 802.15.4 standard on Low-Rate Wireless Personal Area Networks (LR-WPAN) [12] sensor devices already have symmetric-key security primitives for encryption and message authentication. Therefore, security protocol based on symmetric-key is more preferable for resource-poor sensor networks.

Although a number of security protocols have been proposed for key management in WSNs, none of them describes entity authentication and network access control except Zigbee [30] in commercial mode. Some of the above protocols assume that the master key is established by using some other entity authentication protocol or is preloaded before deployment [14] [13] [23] [29] [20] [23]. The security architecture of Zigbee uses the PAN coordinator as the authentication and key distribution center, which makes the architecture unscalable and cannot be used in mobile environment of multiple PANs where mobile nodes move from one PAN to another PAN.

In this paper, we present the design, simulation, and implementation of a generalized authentication framework nicknamed *EAP-Sens*, based on the Extensible Authentication Protocol (EAP) [2] for IEEE 802.15.4 networks. EAP-Sens uses the Generalized Pre-Shared Key (GPSK) method for entity authentication and access control and prevents unauthorized devices from joining the network. This prevents unauthorized nodes from intercepting network packets, becoming forwarders and subsequently playing denial of service (DoS) attacks, responding to a query by spoofing a legitimate node, or injecting fabricated packets into the network.

WICON'08, November 17-19, 2008, Maui, Hawaii, USA

Contributions of the paper include the extension of EAP architecture for multi-hop wireless networks, an original encapsulation of EAP packets for LR-WPAN and optimization of the GPSK authentication method for low-power sensor devices.

The paper continues in Section 2 with a brief review of security protocols available for sensor networks. In Section 3 we briefly discuss EAP mechanism, EAP-Sens design challenges, and our solutions. We present EAP-Sens implementation and operation principle in Section 4. We present security analysis in Section 5. Section 6 presents performance analysis and simulation results and Section 7 concludes the paper.

2. RELATED WORK

The earliest proposal for securing WSNs is a set of security protocols for TinyOS based wireless sensor networks, called *SPINS*, [21] to provide data confidentiality and message integrity and authentication. The first implementation of WSN security protocol, *TinySec* [14], a link-layer security architecture similar to *SPINS*, provides data encryption and authentication functions with little overhead. These protocols does not provide authentication or key management functions.

In [25], authors present an end-to-end security architecture for sensor networks based on public-key cryptography. They have implemented a small-footprint secure HTTP stack, nicknamed *Sizzle*, which runs in less than 4KB of RAM, performs a full SSL handshake in 1 second and transfers 1KB of application data over SSL in 0.4 seconds. Sizzle requires public key infrastructure (PKI) for authentication. In [10], authors proposed a public key based key management protocol for sensor devices, in which a centralized device distributes public key parameters to sensor devices and acts as public key repository.

In [23], authors proposed a security architecture for WSNs which provides pairwise key agreement, cluster key establishment, and secure communication. It uses a certificate authority (CA) to generate and distribute a symmetric-key bivariate polynomial of some degree over a finite field. Two nodes derive a pairwise key by evaluating their private polynomial which can be derived directly from the CA's polynomial and is the same for both nodes. A similar protocol is proposed in [13], where a centralized device authenticates new devices that join the cluster, generates cluster key randomly and updates it periodically. To authenticate distant nodes, cluster members enable IEEE 802.1X based port authentication. Cluster heads act as the gateway of the cluster and communicate with other cluster heads. Another similar protocol for WSN is Lightweight Security Protocol (LiSP) [20], each node shares a *master key* with the *key server* which is established by executing some sort of *entity authentication* protocol. The key server generates a set of new keys using one-way and periodically broadcasts a new key well before its use for encryption or decryption, and a client node first authenticates the received key and then recovers all previously missed keys, if any.

Zigbee [30], the industry alliance of wireless sensor devices, defines a security architecture for WPANs. In Zigbee, the PAN coordinator acts as the trust center and shares a *master key* with each of the devices in the network. If a device does not have a pre-configured master key with the trust center, the trust center sends a master key to the device in plain text. Otherwise, the trust center and the device run the Symmetric-Key Key Exchange (SKKE) protocol for mutual authentication and key agreement.

S. Zhu et al. propose an efficient security protocol for WSN called LEAP [29]. LEAP assumes different security requirements for different types of messages and uses different keys for securing these messages. It uses *individual keys* to communicate with the base station, *pairwise keys* are used to communicate with other sensor nodes, a *cluster key* is used to communicate with neighbors, and a *group key* is shared by all the nodes in the network. The individual key is preloaded into each node before being deployed. LEAP uses one-way hash chains for authenticating local broadcasts like SPINS. The most notable feature of LEAP is that it supports in-network aggregation.

3. EAP-SENS DESIGN

Extensible Authentication Protocol (EAP) [2] is an authentication framework which supports multiple authentication mechanisms. EAP uses four messages (**EAP-Request, EAP-Response, EAP-Success,** and **EAP-Failure**) and typically runs directly over the link-layer without requiring the Internet Protocol (IP) and therefore includes its own support for in-order delivery and retransmission. Actual authentication messages are exchanged between *EAP server* and *EAP peer* in **EAP-Request** and **EAP-Response** messages until successful completion of authentication or authentication fails. If authentication fails at any point, *EAP server* sends **EAP-Failure** to *EAP peer*, otherwise *EAP server* sends **EAP-Success**. In successful authentication, *EAP server* and *EAP peer* establish a Master Session Key (MSK). In 3-party pass-through mode, *EAP server* exports the MSK to *authenticator* to be used as the long-term shared secret between the *authenticator* and the *EAP peer*.

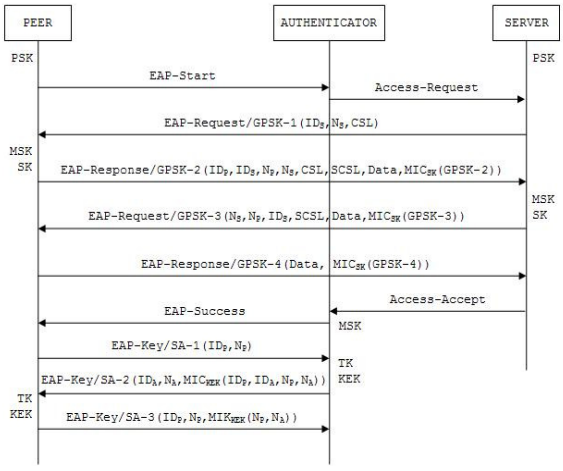

Figure 1: GPSK EAP Authentication

EAP messages between the *authenticator* and *EAP server* are transported using the authentication, authorization and accounting (AAA) protocol, e.g. RADIUS messages, such as

Access-Request and **Access-Accept**. In this case *EAP server* is co-located with AAA, e.g. RADIUS server. AAA servers are used in cellular networks to manage mobile phones by the mobile network operators. The authentication method developed in this paper allows an extension of AAA based mobile network management to wireless sensor networks such as residential or medical sensor networks.

3.1 EAP-GPSK Authentication

EAP supports multiple authentication algorithms called method. We choose the generalized pre-shared key EAP method (EAP-GPSK) [7] for our EAP-Sens design because of its simplicity, lightweightness, and good level of security [24] [7]. EAP-GPSK performs authentication between EAP server and EAP peer based on a pre-shared secret key (PSK). When EAP server receives an AAA **Access-Request** from PAN coordinator, the server initiates the protocol by sending GPSK-1 message containing *server ID* (ID_S) and *server nonce* (N_S) to the peer. Upon receipt of GPSK-1 message, the peer generates *peer nonce* (N_P) and derives Master Session Key (MSK), extended MSK ($EMSK$), session key (SK), and payload encryption key (PK) using GKDF [7] from PSK, ID_S, ID_P (*peer ID*), N_S, and N_P (*peer nonce*). The peer then replies with GPSK-2 message by repeating received GPSK-1 parameters and (ID_P), (N_P), and cipher suite selection protected with MIC computed using SK. On receipt of GPSK-2, the server derives all the keys, verifies the MIC in the message, and if MIC verification succeeds, it sends GPSK-3 message to the peer consisting of ID_S, N_S, and N_P protected with MIC of the message. The peer verifies the MIC in the message upon receipt of GPSK-3, if verification succeeds, the peer responds with GPSK-4 message. When the server receives GPSK-4 message, authentication is complete and it exports the MSK to the PAN coordinator in an AAA **Access Accept** message to be used as the long-term security association between the authenticator and the peer. Upon receipt of **Access Accept** message from the server, PAN coordinator sends **EAP-Success** message to the peer. A successful EAP-GPSK authentication exchanges is shown in Fig. 1.

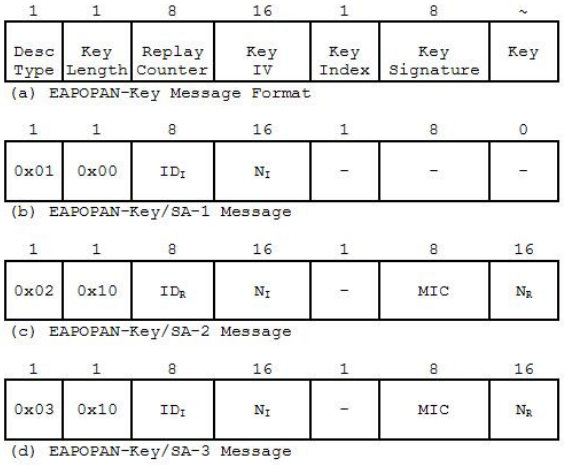

Figure 2: EAP-Key/SA Messages

3.2 Security Association Protocol

When a sensor completes EAP authentication successfully with EAP server, it (*initiator*) initiates the *security association protocol* (SAP) with the PAN coordinator (*responder*) to establish session keys. The *initiator* starts by sending *initiator ID* (ID_I)and *initiator nonce* (N_I)to the *responder* in SA-1 **EAPOPAN-Key** mesh frame as shown in Fig. 2. The *responder*, upon receipt of SA-1, generates *responder nonce* (N_R) and derives an auxiliary MSK ($AMSK$), a key encryption key (KEK), and a temporal key (TK) using the same GKDF from MSK, ID_I, N_I, ID_R (*responder ID*), and N_R. The *responder* then sends SA-2 message containing ID_R and N_R protected using MIC of the message computed using KEK. When the *initiator* receives SA-2 message, it derives all the keys and verifies the MIC of the message. If MIC verification succeeds, the *initiator* sends SA-3 message consisting of ID_I and N_I protected using MIC of the message computed using KEK. When the *responder* receives SA-3 message, the SAP is complete. The *initiator* and the *responder* use TK as the session key.

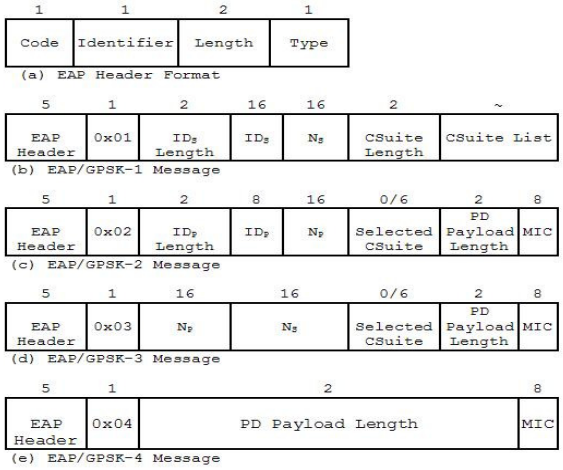

Figure 3: GPSK Packet Format

3.3 Design Challenges

Sensor devices have very little computing resources such as 8 or 16-bit processor, few KB of RAM, few hundred KB of instruction memory, low-bandwidth radio, and more importantly limited battery power. Thus designing any protocol for sensor network must consider the resource utilization and keep resource requirement as low as possible.

Designing EAP authentication framework for IEEE 802.15.4 networks is a challenging task mainly due to smaller packet size and ad hoc multi-hop wireless network. The other important issue of adapting EAP framework for IEEE 802.15.4 networks is the ad hoc multi-hop access network. EAP protocols used for access control use IEEE 802.1X 3-party model, where all mobile devices are within the radio range of the authenticator and communicate directly with it. In IEEE 802.15.4 networks, it is very likely to have devices several hops away from the PAN coordinator and communicate with the PAN coordinator via intermediate coordinators. Therefore, traditional EAP based protocols are not directly applicable to LR-WPAN networks.

The IEEE 802.15.4 protocol data units have the maximum

physical layer packet size of 127 octets and leave only 81 octets for upper layer after physical and link-layer overheads [12]. This is obviously far below what is required for the existing authentication and key exchange protocols. EAP methods use Network Access Identifier (NAI) as the server and client IDs, which can be up to 256 octets in length and cannot be transported in a single IEEE 802.15.4 Medium Access Control (MAC) frame. Another important parameter of EAP protocols is the size of random numbers, which are usually 32-octet numbers. It is not possible to carry two 32-octet random numbers in a single IEEE 802.15.4 MAC frame in addition to other protocol parameters.

We extend the IEEE 802.1X based 3-party EAP model and present a 4-party EAP model for IEEE 802.15.4 based sensor networks, where the PAN coordinator acts as the authenticator and we define a new role for coordinators called *Relay Authenticators* (RAs) to tunnel EAP messages between a supplicant and the PAN coordinator securely.

3.4 GPSK Optimization

We propose the following optimization to the EAP-GPSK standard [7] to keep GPSK messages smaller so that each of them can be transported in a single frame.

- Use sensor node's 64-bit MAC address as ID_P and EAP server's IPv6 address as ID_S.
- Use 16-octet *random numbers* as nonces instead of 32 octets.
- Do not repeat the ID_S and N_S values in GPSK-2, but use them in MIC computation.
- Do not repeat the ID_S in GPSK-3, but use it in MIC computation.
- Use IEEE 802.15.4 standard ENC-MIC-64 [12] cipher suite as the default cipher suite.

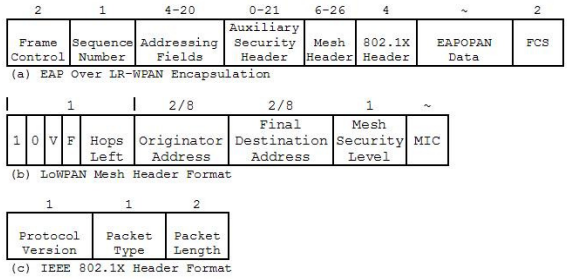

Figure 4: EAP Encapsulation

3.5 EAP Encapsulation

We use the IEEE 802.1X EAP over LAN (EAPOL) encapsulation [11] as the basis for our EAP encapsulation for LR-WPAN called EAPOPAN and is shown in Fig. 4(a). We propose to use a reserved value (e.g. 100) for the *Frame Type* subfield of the *Frame Control* field of IEEE 802.15.4 MAC header to indicate IEEE 802.1X frames. EAPOPAN messages must be transported in mesh frames defined in RFC 4944 [19] with an extended mesh header as shown in Fig. 4(b) and an IEEE 802.1X header as shown in Fig. 4(c). Fig. 2(a) shows the EAPOPAN-Key frame format which is used to transport cryptographic keys and SAP messages (Fig. 2(b) - Fig. 2(d)).

3.6 Key Hierarchy

Each device in the network shares a 16-octet Pre-Shared Key (PSK) with the EAP server, which is preloaded into each deviceŠs memory before being deployed and is used for entity authentication only. During EAP-Sens authentication and security association, each of the devices derives a number of keys organized hierarchically for different purposes as shown in Fig. 5.

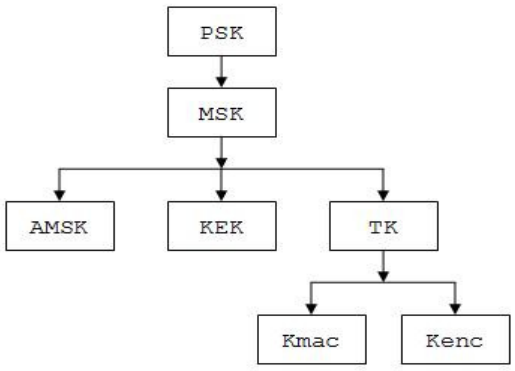

Figure 5: EAP-Sens Key Hierarchy

- **Master Session Key (MSK)**: EAP server and a supplicant derive the 64-octet MSK during EAP-Sens authentication, which is used as the shared key between the authenticator and supplicant in SAP.
- **Auxiliary MSK (AMSK)**: A 16-octet $AMSK$ is generated as part of SAP between the authenticator and a supplicant to be used as the shared key between a supplicant and its RA in SAP.
- **Key Encryption Key (KEK)**: A 16-octet KEK is generated as part of SAP between any two nodes and is used to authenticate SAP messages.
- **Temporal Key (TK)**: A 32-octet TK is generated as a result of SAP between any two nodes, which is the session key between them. The left half of TK is used as the encryption key ($Kenc$) and the right half as the authentication key ($Kmac$).

4. IMPLEMENTATION

We have first developed a simulation model of EAP-Sens on NS-2 and then implemented EAP-Sens for TinyOS, an open source operating system for sensor devices, as part of [4]. We have implemented EAP-Sens as the mesh sublayer on top of the MAC sublayer and below the IP layer. We have also implemented the MAC layer security primitives as defined in [12] because the NS-2 module for IEEE 802.15.4 did not implement security primitives.

In this section, we describe a specific instantiation of EAP-Sens realized using EAP and RADIUS. The EAP-Sens state

EAP-Sens Entity	NS-2 C Code	nesC Code
Supplicant	1,800	1,500
Relay Authenticator	2,000	1,700
Authenticator	1,500	1,400

Table 1: Approximate Lines of Code for EAP-Sens Entities

Protocol	RAM	ROM
TinySec	0.7 KB	7 KB
EAP-Sens	1.4 KB	15 KB
LEAP	1.3 KB	18 KB
Sizzle	3.5 KB	49 KB

Table 2: Memory Requirements by Different WSN Security Protocols

machines are maintained at the sensor nodes and the authentication server. Note that the RADIUS client in the IEEE 802.15.4 PAN coordinator serves as an EAP relay and keeps record of relay authenticators for supplicants which are several hops away. Since we are using the IEEE 802.15.4 security primitives for encryption, message authentication, and replay protection; EAP-Sens requires to implement the rest, i.e. EAP state machine, GPSK protocol, and SAP. To save memory, we have implemented EAP state machine partially and relied on the MAC layer for reliable transport of EAP messages. We have used the same key derivation function for deriving GPSK keys and SA keys.

For EAP-Sens implementation, we have borrowed codes from public domain EAP implementation [1] and modified according to our design. Since nesC does not support dynamic memory allocation, static arrays are used. Every node has to perform GPSK authentication with the AS and security association with the authenticator. Distant nodes also need to perform security association with corresponding RAs. Coordinators also need to keep security associations with all of their children. We have used arrays of size 10 for storing 10 security association parameters. Table 1 shows the sizes of different EAP-Sens components in terms of lines of nesC and NS-2 C codes. Note that the authenticator requires more RAM space for storing security association information of all the nodes of a network. Table 2 compares memory requirements of EAP-Sens with TinySec [14], LEAP [29], and Sizzle [25].

4.1 EAP-Sens Operation

In order to authenticate a device using symmetric-key cryptography, the authentication server and the device must have a pre-shared secret. Key pre-distribution determines how this pre-shared secret is distributed and shared throughout the network. The most common key pre-distribution technique is to install the secret into the device before deployment. In EAP-Sens, each node shares a secret key (PSK) with the authentication server which is loaded into the sensor node when it is programmed before deployment.

4.1.1 Authenticating PAN Coordinator's Neighbors

EAP-Sens uses the standard 3-party EAP model for authenticating first-hop neighbors of the PAN coordinator (see Sec. 3.1). The supplicant starts the process by sending an **EAP-Start** message to the PAN coordinator. The PAN coordinator, upon receipt of an **EAP-Start** from a supplicant, sends an AAA **Access-Request** to the AS for the supplicant. When the AS sends **EAP Identity Request**, the PAN coordinator gives proxy for the supplicant by responding with an **EAP Identity Response**. Then, the AS and the supplicant perform GPSK authentication as shown in Fig. 1 and establish an MSK . After successful GPSK authentication, AS exports the MSK to the authenticator in an **Access-Accept** AAA message. The authenticator subsequently sends an **EAP-Success** to the supplicant. The supplicant and the authenticator then perform a secure association to establish session keys as described in Sec. 3.2 and shown in Fig. 1.

4.1.2 Authenticating Distant Nodes

A supplicant that is two or more hops away from the PAN coordinator uses its coordinator as an RA for authentication. Note that a coordinator can act as an RA only if it has already been authenticated and established security association with the PAN coordinator. After completing link-layer association with a coordinator, a sensor node starts EAP authentication by sending an **EAP-Start** message to its coordinator. The coordinator acts as an RA for the supplicant and forwards the **EAP-Start** message toward the PAN coordinator encapsulated in an **EAPOPAN** mesh frame. The RA protects the EAP message from in-flight tampering by appending mesh-layer MIC to the mesh frame computed with $Kmac$ key shared with the PAN coordinator. The RA also applies MAC sublayer security mechanism before forwarding the frame to its parent to prevent malicious nodes from injecting false EAP frames into the network or replaying old frames. EAP messages are routed to the PAN coordinator using the association tree [12], i.e. each node forwards the message to its parent. While forwarding, intermediate nodes (coordinators) set the reverse path to the supplicant by inserting an entry into the mesh routing table to be used later for forwarding EAP messages from the PAN coordinator to the supplicant.

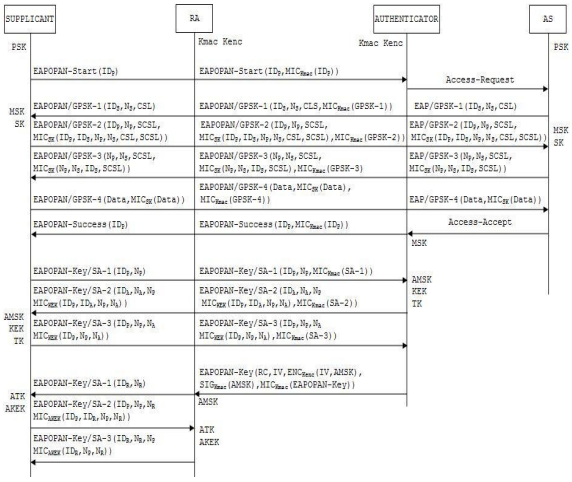

Figure 6: EAP Authentication for Distant Nodes

Upon receipt of an **EAP-Start** message via an RA, the PAN coordinator verifies the mesh-layer MIC of the mes-

sage, saves the address of the RA and the supplicant, and sends an **Access-Request** AAA message for the supplicant to the AS. When the AS sends **EAP Identity Request**, the PAN coordinator gives proxy for the supplicant by responding with **EAP Identity Response**. The AS and the supplicant then perform GPSK authentication and derive MSK as described in Section 3.1. GPSK EAP messages from the PAN coordinator to the RA are also transported in encapsulated **EAPOPAN/GPSK** mesh frames (see Fig. 3) and protected using mesh-layer MICs. After successful GPSK authentication, AS exports MSK to the PAN coordinator in **Access-Accept** AAA message. On receipt, the PAN coordinator sends an **EAP-Success** to the supplicant via the RA in an **EAPOPAN** mesh frame as shown in Fig. 6.

When a supplicant receives the **EAP-Success**, it initiates SAP with the PAN coordinator and establishes $AMSK$ and TK. SAP messages between a supplicant and the PAN coordinator are also transported in **EAPOPAN-Key/SA** mesh frames and protected using mesh layer security, see Fig. 6 and 2. After successful security association, PAN coordinator exports the $AMSK$ to the RA in **EAPOPAN-Key** mesh frame (see Fig. 2), to be used as the shared key between the RA and supplicant for SAP. The key is encrypted using the $Kenc$ shared between the PAN coordinator and the RA, the IV value used in encryption and the signature of the key are also included in the **EAPOPAN-Key** message. When the RA receives $AMSK$ for a supplicant, it initiates a SAP protocol with the supplicant to derive auxiliary session key ATK as described in Section 3.2 and shown in Fig. 6. Fig. 6 illustrates the authentication and key establishment mechanisms for supplicants that are not first-hop neighbors of the PAN coordinator.

5. SECURITY ANALYSIS

In wireless sensor networks, it is easy for an adversary to passively eavesdrop on an on-going communication, or to actively inject packets into the network, or to capture a node and obtain security keys. In this section, we will discuss different security threats and show how EAP-Sens defends those threats. We assume that GPSK provides enough security against different types of attacks against an authentication protocol [7]. We also assume that the PAN coordinator has pre-established security association with the AS and is not compromised. To be implemented on IEEE 802.15.4 devices, we have proposed some optimizations to the GPSK protocol. First we discuss security compromisations due to these optimizations.

5.0.3 *Security Threats Due to Using Smaller Nonces*

Nonce values are used by GPSK and SAP for freshness of a session. If a nonce is very small, it could be reused and an adversary could play replay attack. With 16-bit nonces, an attacker has to store and wait for $2^3 2$ successful authentication in order to replay a protocol run. In sensor networks, all the sensors will die before reaching this number of authentications.

5.0.4 *Security Threats Due to Omitting Message Parameters*

In order to defend against *denial of service* (DoS) attacks, authentication protocols do not save state information until authentication is completed. In GPSK, EAP server initiates the authentication process after confirming peer ID needs to store its own nonce for the peer indexed by peer ID. Thus, omitting server ID and *server nonce* from GPSK-2 does not introduce any vulnerability to GPSK protocol.

We assume that sensor nodes use only one EAP server and therefore peer does not need to store server ID and server nonce pairs of GPSK-1 messages from different AAA servers. GPSK-3 message carries server nonce and the client can compute MIC from this nonce and the AS's ID (only one). Since a peer does not need to save state information for the server and hence is not vulnerable to DoS attack.

Finally, since the GPSK messages are still different although some parameters are omitted, it is not possible to play reflection attack.

5.1 Attacks Against SAP

EAP-Sens' SAP is a 3-way challenge response authentication protocol adapted from Zigbee's SKKE [30]. Neither initiator nor responder saves state information for the other party until it completes authentication and hence SAP is free from DoS attacks. Upon receipt of SA-1 message, responder derives keys and replies with SA-2. An intruder could try to exploit this and play resource exhaustion attack against a sensor node by spoofing its parent. From the EAP-Sens protocol message sequence, a supplicant will accept SA-1 message only from its coordinator just after completing security association with the authenticator. Therefore, an adversary cannot send SA-1 message arbitrarily to a sensor node and exhaust its battery power. Finally, since all messages are different, therefore, it is free from reflection attacks. SAP uses 16-octet random numbers for message freshness, which is big enough to prevent replay attacks.

5.2 Attacks Against Relay Authenticators

An RA tunnels EAP messages for its children and does not save state information for them, therefore it is not vulnerable to DoS attack. But, an adversary could try to play resource exhaustion attack against an RA by sending false EAP messages. However, since an RA relays un-authenticated EAP messages only from its children, therefore an adversary has to perform link-layer association for each spoofed node before playing an attack, which limits an adversary from sending continuous EAP messages. To further limit false EAP messages, an RA could use a counter to count EAP messages from a child and if the counter exceeds some threshold without convergence, it could disassociate the client.

5.3 Attacks Against Supplicants

Since a supplicant accepts GPSK-1 message un-authenticated, compute keys and generate response, one could try to play DoS and resource exhaustion attacks by sending false GPSK-1 messages to a sensor. However, since a supplicant accepts this message from a specific EAP server after sending EAP-Start message via its coordinator or PAN coordinator, an adversary cannot blindly send this message continuously to a supplicant.

5.4 Node Capturing

A compromised node can be authenticated as legitimate one and can establish secure session with other nodes and then could play *byzantine* attacks. Authentication protocols cannot prevent node capturing but could minimize adverse effects of compromising nodes. An adversary could capture one or more nodes and obtain security keys. In EAP-Sens, if a supplicant is compromised, it cannot forge other nodes or prevent other nodes from being authenticated. However, if an RA is compromised, it could prevent other nodes from being authenticated or could play *byzantine* attacks but could not compromise other nodes.

5.5 Wormhole/Sinkhole Attacks

In worm-hole or sink-hole attacks, traffic is taken from one place of the network and passed to some other place. If source and destination nodes are not compromised, passing authentication protocol packets to any other nodes does not cause security compromise because from these packets no one can derive session keys.

Since both GPSK and SAP use secret shared keys for authentication and key establishment, it is not possible to play the man-in-the-middle attack. Also they use 16-octet secret keys which are generated using random number generator, it is not possible to play dictionary attack against EAP-Sens.

6. PERFORMANCE

In this section, we will analyze EAP-Sens to estimate computation and communication overheads in terms of base cryptographic operations and the number of MAC frame exchanges respectively for authentication and security association. We will then derive relation for the time complexity of EAP-Sens in terms of cryptographic computation time, MAC frame transmission time with respect to network size. We will also compare EAP-Sens complexities with other WSN security protocol complexities.

6.1 Communication Cost

Communication cost of trusted third-party based authentication protocols increases as the number of hops increases. Getting access to the network with EAP-Sens, a sensor node has to complete GPSK authentication with AS, one security association with PAN coordinator and optionally one security association with its coordinator. EAP-Sens requires 6 messages to be exchanged for a successful GPSK authentication and SAP requires 3 messages. However, for distant nodes, intermediate nodes have to relay 10 messages which include 6 GPSK, 3 SAP, and one key transport messages. If network has N nodes with d average degrees of neighbors, then an EAPOPAN frame has to be relayed by $log_d(N)$ coordinators at best. Therefore, maximum communication overhead on the network of N nodes with d average degrees of neighbors is $t_f = 10 \times log_d(N)$, where t_f is the total number of MAC frames needs to exchanged to complete an authentication run. Therefore, the communication cost increases logarithmically as the network grows.

Table 3 compares communication overheads of EAP-Sens with that of LEAP [29], Zigbee [15], and Sizzle [25], where N is the number of nodes in a network and d is the degree of neighbors or the distance in number of hops between the source and destination.

Protocol	Message Complexity
EAP-Sens	$10 \times log_d(N)$ (d = degree of neighbors)
LEAP	$(d-1)^2 \div (N-1)$ (N = number of nodes)
Zigbee	$3 \times log_d(N) + 5 \times h$ (h = hop distance)
Sizzle	$50 \times log_d(N)$ (approximated)

Table 3: A Comparison of Total Number of MAC Frames in the Network

6.2 Computation Cost

Computation cost of a security protocol consists of computation for parsing messages and computation for cryptographic primitives such as random number generation, encryption, and MIC computation. EAP-Sens uses a few simple messages and therefore time required for parsing protocol messages is negligible compared to that of cryptographic operations. Note that EAP-Sens uses AES-128 encryption as the basic cryptographic operation and the AES is an *involutary function* for a given key, i.e. the same function is used for decryption. In IEEE 802.15.4 devices, MIC is computed using AES-128 encryption function. GKDF used in EAP-Sens uses this function for generating random numbers and keys. Note that Tmote Sky motes take about 0.5 ms for encrypting a 16-octet data block offline with an additional 2 ms for initializing registers for the first time [9].

For GPSK authentication, a node has to perform MIC computation once to generate peer nonce, 10 times to derive GPSK keys, 3 for GPSK message authentication. Thus, a node performs MIC computation 14 times for GPSK authentication. In SAP protocol, a node needs to compute MIC once to generate its nonce, 5 times to derive SAP keys, twice for SAP message authentication totaling 8 times for security association with the authenticator. If a supplicant uses an RA, it needs one more SAP protocol run requiring 8 more MIC computations. Therefore, a node requires either 22 or 30 MIC computations for EAP-Sens protocol run. However, when a node uses an RA, the RA has to compute MIC once for each message it relays. Besides, the RA has to perform one AES encryption for decrypting received $AMSK$ from authenticator, two MIC computations for verifying key signature and MIC of the frame. Therefore, an RA has to compute MIC 10 times for relaying EAP-Sens messages.

From the above discussion, it is clear that the computation cost of a supplicant node remains constant as the network grows. However, if link-layer security is enabled to prevent injecting false EAP messages into the network by intruders, each intermediate node has to perform one MIC computation to verify the MIC of the EAPOPAN frame before relaying it. Therefore, the total computation overhead on the network in terms of the number of MIC computations would be $t_c = 12 + 10 \times log_d(N)$ for an N node network with d average degree of neighbors.

6.3 Time Complexity

The most important parameter of evaluating the performance of an authentication protocol method is the authentication time t, defined as the time spent in authentication process. Authentication time, t which is influenced both by the computational and communication costs of the EAP method, can be expressed as $t = t_x + t_c$ where t_x is the total

flight time, i.e. the time spent over the air by the frame during the whole authentication process and t_c is the total computational time required by the authentication protocol in order to execute all cryptographic primitives and other protocol instructions such as parsing protocol messages, accessing databases, etc.

For a supplicant to complete EAP-Sens authentication successfully and get access to the network in an N node network with d average degree of neighbors, total amount of time required, t, is given by:

$$t = (10 \times log_d(N)) \times t_x + (12 + 10 \times log_d(N)) \times t_{mic}$$

From the above analysis, it is clear that total amount of authentication time t is increased logarithmically as the number of nodes in the network increases.

6.4 Simulation Results

We have simulated EAP-Sens using NS-2 for performance evaluation with the IEEE 802.15.4 MAC protocol in beacon enabled mode for estimating the amount of time it takes to complete EAP authentication for nodes at different hops away. We have used the CMU's *cmu-scene-gen* utility for generating topologies randomly. The area of the network has been chosen based on the number of nodes in the network so that the network has been connected. We ran the simulation several times and took the average with nodes at 1, 2, 3, 4, 5, and 6 hops away from the authenticator.

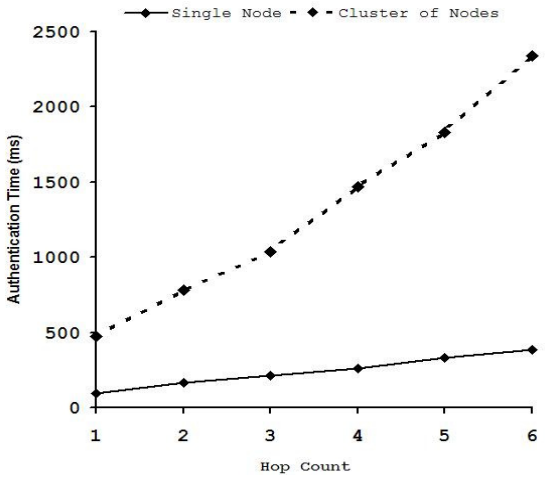

Figure 7: EAP-Sens Authentication Time

The total EAP-Sens authentication and security association time is estimated beginning at the time when a node sends **EAP-Start** message and ending at the time when it receives **EAPOPAN-Key/SA-3** message from its coordinator. Fig. 7 shows the average time it takes to complete the EAP-Sens authentication and security association successfully, where X-axis represents the hop count and the Y-axis represents the amount of time in milliseconds. The solid line represents authentication time for a single node at different distances, whereas the dotted line shows authentication time for a cluster of nodes at different hops. Note that we have chosen average cluster size of 5 nodes, that is, a coordinator can have an average of four children.

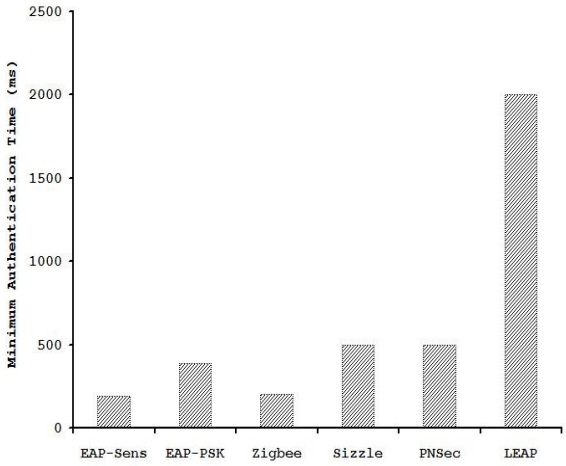

Figure 8: Comparison of Authentication Times for Different Protocols

Fig. 8 compares minimum authentication time taken by EAP-Sens with that of a couple of other protocols on IEEE 802.15.4 compliant sensor nodes. EAP-Sens takes less than 200 ms for mutual authentication compared to 400 ms for EAP-GPSK (without optimization). On the other hand, Kerberos like authentication protocol (Zigbee) takes about 200 ms (see. [16]), optimized SSL authentication (Sizzle) takes at least 500 ms (see [30]), a cluster head in PNSec authenticates all of its neighbors in about 500 ms (see [13]), and LEAP takes about 2 seconds for a pair of nodes (see [29]).

7. CONCLUSIONS

We have designed an sensor network security protocol on top of IEEE 802.15.4 security primitives for entity authentication and key management. We have simulated EAP-Sens using NS-2 for performance evaluation and showed that EAP-Sens performs better than existing WSN security protocols. We have also implemented a prototype version of EAP-Sens in TinyOS which uses an optimized version of GPSK authentication protocol in order to estimate code size and memory requirements. The results indicate that EAP-Sens can be implemented on sensor devices like Mica2, Telos, or Tmote and can be used in many sensor network applications such as medical monitoring and meter readings for utility services. We have studied the performance of EAP-Sens only in static environments, it is important to study protocol performance in mobile environments. Simulation environment needs to be extended in order to evaluate more realistic sensor deployments.

8. REFERENCES

[1] HostAP Implementation. `http://hostap.epitest.fi/releases/snapshots`.

[2] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). RFC 3748, June 2004.

[3] A. Alim and B. Sarikaya. Designing Reliable Wireless EEG System with Motes. In *International Conference on Computer and Information Technology (ICCIT'06)*, Dhaka, Bangladesh, December 2006.

[4] M. A. Alim. A Security Architecture for IPv6 Enabled Wireless Medical Sensor Networks. Master's thesis, Mathematical, Computer, and Physical Sciences, University of Northern British Columbia, July 2007.

[5] Center for Future Health - University of Rochester, New York, USA. *Smart Medical Home*.

[6] T. C. Chan, J. Killeen, W. Griswold, and L. Lenert. Wireless Internet Information System for Medical Response in Disasters (WIISARD). *Academic Emergency Medicine*, 11(11):1229–1236, 2004.

[7] T. Clancy and H. Tschofenig. EAP Generalized Pre-Shared Key (EAP-GPSK). draft-ietf-emu-eap-gpsk-08.txt, December 2007.

[8] R. Fischer, L. Ohno-Machado, D. Curtis, R. Greenes, T. Stair, and J. Guttag. SMART: Scalable Medical Alert and Response Technology. In *Smart Medical Technologies Summit (SMT) 2004*, April 2004.

[9] M. Healy, T. Newe, and E. Lewis. Efficiently Securing Data on a Wireless Sensor Network. In *Journal of Physics: 14th Conference on Sensors and Their Applications*, Liverpool John Moores University, Liverpool, UK, September 2007.

[10] J. Heo and C. S. Hong. Efficient and Authenticated Key Agreement Mechanism in Low-Rate WPAN Environment. In *International Symposium on Wireless Pervasive Computing 2006*, pages 1–5, Phuket, Thailand, January 2006.

[11] IEEE Computer Society. *IEEE Standard for Local and Metropolitan Area Networks: Port-Based Network Access Control*, June 2001.

[12] IEEE Computer Society. *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, September 2006.

[13] A. Jehangir and S. M. H. de Groot. A Security Architecture for Personal Networks. In *First International Workshop on Personalized Networks (PerNets 2006)*, San Jose, California, USA, July 2006.

[14] C. Karlof, N. Sastry, and D. Wagner. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In *ACM Conf. on Embedded Networked Sensor Systems (SenSys'04)*, November 2004.

[15] M. Khan and et. al. The Cost of Security: Performance of ZigBee Key Exchange Mechanism in 802.15.4 Beacon Enabled Cluster. In *IEEE Int. Conf. on Mobile Adhoc and Sensor Systems (MASS'06)*, Vancouver, BC, October 2006.

[16] P. Ki-Woong and P. Kyu-Ho. pKASSO: Towards Seamless Authentication Providing Non-Repudiation on Resource-Constrained Devices. In *Int. Conf. on Advanced Information Networking and Applications Workshops*, Niagara Falls, ON, Canada, May 2007.

[17] K. Lorincz, D. Malan, V. Shnayder, and M. Welsh. Sensor Networks for Emergency Response: Challanges and Opportunities. *IEEE Pervasive Computing Magazine*, 3(4):16–23, October-December 2004.

[18] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton. CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care. In *International Workshop on Wearable and Implantable Body Sensor Networks*, April 2004.

[19] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, September 2007.

[20] T. Park and K. G. Shin. LiSP: A Lightweight Security Protocol for Wireless Sensor Networks. *ACM Transactions on Embedded Computing Systems*, 3(3):634 – 660, August 2004.

[21] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. In *Mobile Computing and Networking 2001*, Rome, Italy, 2001.

[22] B. Sarikaya, M. A. Alim, and S. Rezaei. Integrating Wireless EEGs into Medical Sensor Networks. In *International Wireless Communications and Mobile Computing Conference (IWCMC'06)*, Vancouver, BC, Canada, July 2006.

[23] S. Schmidt, H. Krahn, S. Fischer, and D. Watjen. A Security Architecture for Mobile Wireless Sensor Networks. In *1st European Workshop on Security in Ad-Hoc and Sensor Networks - ESAS 2004*, Heidelberg, Germany, August 2004.

[24] D. Tonesi and L. Salgarelli. Smart PSK provisioning: a key-management and authentication scheme for wireless LANs. In *IEEE 14th International Conference on Computer Communications and Networks*, pages 119–124, October 2005.

[25] V. Gupta et al. Sizzle: A Standards based End-to-end Security Architecture for the Embedded Internet. In *Int. Conf. on Pervasive Computing and Communications (PerCom)*, Kauai, HI, USA, March 2005.

[26] K. Venkatasubramanian, G. Deng, T. Mukherjee, J. Quintero, V. Annamalai, and S. K. S. Gupta. Ayushman: A Wireless Sensor Network Based Health Monitoring Infrastructure and Testbed. In *IEEE Int. Conf. on Distributed Computing in Sensor Systems (DCOSS)*, June 2005.

[27] G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, Z. He, R. Stoleru, S. Lin, and J. Stankovic. An Assisted Living Oriented Information System Based on a Residential Wireless Sensor Network. In *First Distributed Diagnosis and Home Healthcare (D2H2) Conference*, Arlington, VA, USA, April 2006.

[28] R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus. Tinypk: Securing sensor networks with public key technology. In *SASN '04: 2nd ACM Workshop on Security of Ad hoc and Sensor Networks*, pages 59–64, New York, NY, USA, 2004. ACM Press.

[29] S. Zhu, S. Setia, and S. Jajodia. LEAP+: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. *ACM Transactions on Sensor Networks*, 2(4):500–528, 2006.

[30] ZigBeeTM Alliance, http://www.zigbee.org/en. *ZigBee Specification*, December 2006.