ISSN:2229-6093

Anupma Sangwan et al, Int. J. Comp. Tech. Appl., Vol 2 (4), 790-797

A Review of various security protocols in Wireless Sensor Network

Anupma Sangwan[1], Deepti Sindhu[2], Kulbir Singh[3]
[1] *NC Institute of Technology, Israna*
anulathwal@gmail.com,
[2] *Singhania University, Rajasthan*
deepti.sindhu@gmail.com,
[3]*Singhania University, Rajasthan*
Kulbir99@gmail.com,

Abstract

Sensor networks are highly distributed networks of small, lightweight wireless sensor nodes, deployed in large numbers to monitor the environment or system by the measurement of physical parameters such as temperature, pressure, or relative humidity, sound, vibration, motion or pollutants, at different locations. A WSN [1] is composed of a large number of low-cost sensor nodes (SNs) and one or several base stations (BS) or destination nodes. SNs are typically small wireless devices with limited computational power, equipped with a radio transceiver or other wireless communications device, radio transmission range, storage size, a small microcontroller, battery power. Sink nodes or destination nodes are distinguishing devices with powerful computation capacity, large memory size, high power energy and long communication range so as to collect data from SNs. Sink nodes act as the gateway between WSNs and the end user..

1. Introduction

An ideal sensor network protocol should provide data authentication, confidentiality, integrity of data and protection from replay. The security and efficiency being the basic parameters used to design a new sensor network protocol.

A wireless sensor node integrates one or more sensors, a processor, a communication unit, a power supply and management unit and, if need be, a security and actuation unit [2]. Depending on the sensing task for which the network is deployed, there can be multiple sensors integrated within a single node.

2. Background

As wireless sensor networks have been used in a lot of applications, monitoring nodes in the networks are needed for some applications. The important functions which need to be monitored are the communicated data between each node, the movement of nodes, etc. In recent years, the sensor has made a great progress. Sensor networks have been gaining popularity as a low cost solution to the various applications. Their low cost gives an opportunity to exploit them in the military, industrial and home applications.

Sensors have started replacing the human factor in various industrial processes. These small embedded systems have become an integral part of large networks and participate in the distributed applications. As our daily life is getting more advanced, a variety of "sensitive" data used is stored, altered, manipulated, or communicated by the means of electronic systems. Hence, arises the need to deal with the security of these systems as an important aspect. Security has been the topic of research for cryptography, computing, and networking applications. As a matter of fact, security is a metric, which has to be, implemented at each increasing step in the system design, keeping in mind other factors such as development cost, performance, and power consumed.

Security is a requirement for an increasing number of growing embedded system applications ranging from low-end products such as PDAs, wireless handsets, smart id-cards, to high-end products such as routers, firewalls, gateways, storage and web-servers. A survey among internet users has rated security as a primary concern in the adoption of new services and applications.

Sensor nodes are used in a variety of applications which require constant monitoring and detection of specific events. The three important application classes:

- Environmental data collection,
- Security monitoring, and
- Sensor node tracking.

The majority of wireless sensor network deployments will fall into one of these class templates.

2.1. Environmental data collection:

A basic environmental data collection application is one where a researcher wants to collect several sensor

readings from a set of points in an environment through a period of time in order to observe trends and mutualities [3]. In the area of environmental observation and forecasting, researchers aim at establishing an early warning system to protect the population, and on the other hand, provide researchers with the means to study certain phenomena. This is because instrumenting natural places, such as national parks, volcanos, riverbanks, rift zones, and woods with numerous networked sensor nodes can enable long-term data collection at scales and resolutions that are difficult to obtain otherwise [2]. The researcher would be interested in gathering data over several months or years in order to look for long-term and seasonal trends. For the data to be meaningful it would have to be collected at regular intervals and the nodes would remain at known locations.

2.2. Security monitoring:

The second class of sensor network application is security monitoring. Security monitoring networks are composed of nodes that are placed at fixed locations throughout an environment that continuous monitor one or more sensors to detect an anomaly. A key difference between security monitoring and environmental monitoring is that security networks are not actually collecting any data. This has a significant impact on the optimal network architecture. Each node has to frequently check the status of its sensors but it only has to transmit a data report when there is a security violation. The immediate and reliable communication of alarm messages is the primary system requirement. These are "report by exception" networks.

2.3. Node tracking scenarios:

A third usage scenario commonly discussed for sensor networks is the tracking of a tagged object through a region of space monitored by a sensor network. There are many situations where one would like to track the location of valuable assets or personnel. Current inventory control systems attempt to track objects by recording the last checkpoint that an object passed through. However, with these systems it is not possible to determine the current location of an object.

3. Secure Protocols

3.1. SPINS

Sensor Protocols for Information via Negotiation (SPINS) consists of a suite of security protocols that are optimized for highly resource constrained sensor networks [6]. SPINS consists of two secure building blocks SNEP and µTESLA, which run on top of TinyOS. TinyOS has been adopted by thousands of developers worldwide, on many platforms for a broad range of wireless sensor networks. TinyOS is based on an event-driven programming model instead of multithreading. Secure Network Encryption Protocol (SNEP) provides data authentication, protection from replay attacks, and semantic security (a property which prevents an adversary from learning even partial information about a transmitted message), which is an important security property, as it prevents eavesdroppers from inferring the message content from the encrypted message; achieved as the counter value is incremented after each message, implying that the message is encrypted differently each time. The counter value is sufficiently long enough never to repeat within the lifetime of the node. In addition to integrity it provides confidentiality through encryption and authentication using a message authentication code MAC. This is similar to a checksum derived by applying an authentication scheme with a secret shared key to the message. The message can be decrypted only if the sane shared key is present. There are a number of unique advantages with SNEP. It has a very low communication overhead, adding only 8 bytes per message. Finally, it also provides data authentication, replay protection and weak message freshness [6]. To achieve data authentication, the same block cipher is used as in CBC-MAC mode.

µTESLA is the "micro" version of TESLA (Timed Efficient Stream Loss-tolerant Authentication) proposed in [8]. µTESLA ensures an authenticated broadcast, that is, nodes which receive a packet can be assured of its sender's identity. It requires a loose time synchronization between base station and nodes, with an upper bound on maximum synchronization error. For an authenticated packet to be sent, the base station computes a MAC on the packet with the key that is secret at that point in time. When a node gets a packet, it can confirm that the base Station did not yet disclose the corresponding MAC key, using its loosely synchronized clock, maximum synchronization error and the time at which the keys are to be disclosed. The node stores the packet in a buffer, aware that the MAC key is only known to the base station, and that no adversary could have altered the packet during transmission. When the keys are to be disclosed, the base station broadcasts the key to all receivers. The receiver can then verify the correctness of the key and use it to authenticate the packet stored in the buffer [6].

TESLA relies solely on this delayed disclosure, unlike its predecessor, which authenticates the initial packet using the digital signature. It has been argued that while symmetric key techniques are attractive, due to their energy efficiency, limitations have been exhibited in the flexibility of these symmetric key exchange protocols [9]. The MAC keys are derived from a chain of keys, obtained by applying a one-way function F. All nodes have an initial key K_0, which is

some key in the key-chain. The relationship between keys proceeds as $K_0 = F(K)_1$, $K_1 = F(K_2)$ and, in general, the sender chooses the last key K_2 of the chain randomly, and applies *F* repeatedly to compute all other keys: $K_i = F(K_i + 1)$. Given K_0, K_1, …., K_i, it is not possible to compute K_i+1. The key to be used changes periodically, and since nodes are synchronized to a common time within a bounded error, they can detect which key is to be used to encrypt/decrypt a packet at any time instant. Applying the SNEP building block, each node can easily perform time synchronization and retrieve an authenticated key from the chain for the "commitment in a secure and authenticated manner" [6].

The BS periodically discloses the next verification key to all the nodes and this period is known to all nodes. There is also a specified lag of certain intervals between the usage of a key for encryption and its disclosure to all the receivers. When the BS transmits a packet, it uses a MAC key which is still secret. The nodes which receive this packet buffer it until the appropriate verification key is disclosed. But, as soon as a packet is received, the MAC is checked to ensure that the key used in the MAC has not yet been disclosed, which implies that the only the BS which knows that yet undisclosed key could have sent the packet. The packets are decrypted once the key-disclosure packet is received from the BS. If one of the key-disclosure packets is missed, the data packets are buffered till the next time interval, and then authenticated. For instance, suppose the disclosure packet of K_j does not reach a node; it waits till it receivesK_{j+1}, then computes K_j = F(K_{j+1}) and decrypts the packets received in the previous time interval. Schemes, like µTESLA, based on delayed key disclosure, can suffer from denial of service attacks (DOS). In the subsequent interval when the message is in the buffer and the receiver waits on the disclosure time, an attacker can flood the network with arbitrary messages, claiming that they belong to the current time interval. Only in the next time interval can the nodes determine that these messages are not authentic. This type of attack can lead to buffers overflowing in the nodes and battery exhaustion as all messages are forwarded to the nodes. The use of public key cryptography would eliminate the need for such complicated protocols, increasing the security of the system, and only requiring the public key of the base station to be embedded into all of the nodes [9].

SPIN has four types: SPIN-PP, SPIN-EC, SPIN-BC, and SPIN-RL [7]. In SPIN-PP, Nodes use three types of messages ADV, REQ and DATA to communicate. ADV is used to advertise new data, REQ to request for data and DATA is the actual message itself. The protocol starts when a SPIN node obtains new data that it is willing to share. It does so by broadcasting an ADV message containing meta-data. If a neighbour is interested in the data, it sends an REQ message for the DATA and the DATA is sent to this neighbour node. The neighbour sensor node then repeats this process to its neighbours as a result of which the entire sensor area will get a copy. It starts by advertising its data to one node from other Node. Node responds by sending a request message. After receiving the requested data, node then sends out advertisements to its neighbours.

3.1.1. Secure-SPINS

Notations

With the limited computation resources available in sensor network, used symmetric key cryptography, which does not involve any cryptographic functions that require large memory and processing power to construct the Secure-SPIN protocols. The Secure-SPIN protocol is divided into three phases according to SPIN's three messages. They will use the following notation:

(a) AC: Authentication Code;
(b)K_s: sink's privacy key;
(c) K_{ses}: session key;
(d) PSAC: Personal Sensor Authentication Code;
(e) K_i: sensor node's privacy key;
(f) H(): hash function to calculate the hash value;
(g) MAC: Message Authentication Code.

3.2. LEAP

Localized Encryption and Authentication Protocol (LEAP) [10] as a key management protocol for sensor networks designed to support in- network processing, while restricting the impact of a compromised node to the network. At the time, pre-deployed keying was the most practical approach for bootstrapping secret keys in sensor nodes. This implies that the nodes were loaded into all of the sensors before they were deployed in the sensor field. This may seem primitive at this point in time, but is included to achieve thoroughness. Pairwise keys could be generated between two Nodes based on this pre-deployed keying information. The overhead is variable depending on the types of keys specified for use in the implementation. All four types may not be used for a particular application.

LEAP is a key management protocol intended for sensor networks based on symmetric key algorithms, that is, the same key is used by sender and receiver. In a network, requiring every pair of nodes to have a shared key to be used for communication between them is ideal for security, because an attack on any one node does not compromise the security of other nodes.

However, in sensor networks, the neighbours of a node may not be known in advance, hence this sharing of keys must take place after the network is deployed, which will cause a high overhead. Also, sensor networks may employ certain processing optimizations such as a node's deciding not to report an event if it overhears its neighbor reporting the same. Such optimizations will be precluded by the usage of a separate key for all nodes in the network has lower overhead, but compromise of any node affects the entire system.

LEAP specifies four types of keys: individual keys, pairwise shared keys, cluster keys and group keys. Individual keys are symmetric keys which it shares with the base station and each of the nodes. For example, a node might use the individual key to notify the base station of a suspicious neighbor. The individual key is preloaded into the node before deployment, and is used for transmission of any special information between the BS and the node, such as exclusive instructions to a node, or report from a node to BS about the abnormal behavior of a neighboring node. Pairwise shared keys are symmetric keys shared between a node and each of its neighbors. Cluster keys are symmetric keys shared between a node and all of its neighbors. These cluster keys can be used for locally broadcast messages such as a routing protocol might use and are also used for updating the group key. The group key, a symmetric key shared with all nodes of the network and the base station, allows encrypted and authenticated messages to broadcast through the whole network.

It is assumed that the time required to attack a node is greater than the network establishment time, during which a node can detect all its immediate neighbours. A common initial key is loaded into each node before deployment. Each node derives a master key which depends on the common key and its unique identifier. Nodes then exchange Hello messages. The nodes then compute a shared key based on their master keys. The common key is erased in all nodes after the establishment, and by assumption, no node has been compromised up to this point. Since no adversary can get the common key, it is impossible to inject false data or decrypt the earlier exchange messages. Also, no node can later forge the master key of any other node.

3.2.1. Key Establishment

LEAP details how each of these keys are established. The key establishment protocols are meant to be lightweight and scalable. While keys may be preloaded prior to deployment in cases where the network architecture is known beforehand, many wireless sensor networks are deployed in environments where such prior knowledge is not possible and keys must be established either during an initial setup phase or on-the-fly as the network architecture changes. In addition, preloaded keys must be updated to prevent cryptanalysis attacks.

Notation

They try to use consistent notation throughout this LEAP protocol. They specifically note any place where they deviate from the following meanings.

A, B, C, S → communicating nodes in the network
NA → a nonce generated by node A
M1.M2 → message M1 concatenated with message M2
$\{M\}_K$→ message M encrypted with key K
K_{AB} → symmetric key between nodes A and B
$\{f_k\}$ →a family of pseudo-random functions
MAC (K, M) → a message authentication code (MAC) on message M using key K
A→ B: M→ message M sent from A to B
A→*: M → message M broadcast by A

- **Individual Keys**: Because the base station and each of the nodes is generally known before deployment, the protocol specifies that the individual keys should be preloaded into the nodes. To save the base station's memory, each of these individual keys is generated using a master key, K_m, and the node's unique ID. For example the key Kas shared between the base station (S) and a node (A) would be $K_{as} = f_{k_m}(A)$. Thus, when the base station receives a message from or wishes to send a message to A, the base station can generate the individual key using the stored master key.
- **Pairwise Shared Key with Single-Hop Neighbour**: Pairwise shared key are established between all immediate neighbours. A pairwise shared key with a single-hop neighbor requires four phases. First, each node calculates a master key from an initial key, K_i, preloaded into all the nodes and the nodes unique name. Second, nodes then exchange HELLO message with its name and a nonce, which are authenticated by the receivers. Then each neighbor replies to the message with a message authentication using the message authentication code (MAC) with its master key, which the initial node can check using the neighbor's identity and the initial key. Third, the two nodes calculate the pairwise shared key. Finally, once the initial setup timer has expired, the nodes erase both the initial key and their neighbors' master keys.
- **Pairwise Shared Key with Multi-Hop Neighbor**: To establish a pairwise shared key with a multi-hop neighbor requires the use of shared neighbors, called proxies. If a node, A, wishes to establish a pairwise shared key with its

multi-hop neighbor, C, A would first broadcast a QUERY message to all its neighbors with its ID and the ID of the desired node. The n shared neighbors, B_i, which are single-hop neighbors of A and singe-hop neighbors of C, send a REPLY message authenticated with the pairwise shared key. Node A splits the new key KAC into n randomly generated shares Ki such that KAC = K1xorK2xor…..xor Kn. A sends these shares to the proxies encrypted with a pairwise shared key and a verification key $f_{k_i}(0)$. The proxies then re-encrypt the share with the pairwise key shared between the proxy and C, and the proxies forward the key and the verification key to C. Finally when C receives the shares from the proxies, C verifies the shares, recreates the new key by KAC = K_1 xorK_2xor….xor K_n, and sends A. DONE message encrypted with the new key.

- **Cluster Key.** The cluster key is established by a node after the pairwise key establishment. A node generates a cluster key and sends it encrypted to each neighbour with its pairwise shared key.
- **Group Key**. Group keys can be preloaded, but it should be updated once any compromised node is detected so that the intruder cannot send and read encrypted messages for the network. This could be done, in a naïve way, by the BS's sending the new group key to each node using its individual key, or on a hop-by-hop basis using cluster keys. In addition, this key should be updated regardless of node revocation to prevent cryptanalysis.

3.2.2. Problem of Pre-deployed Key Agreement

- Compromise of a single node reveals the secret key shared by all the network nodes. This would in turn disclose all future communications as well as past recorded communication by a passive adversary.
- It becomes cumbersome to add new nodes to the sensor network. Either newly added nodes should have the network-wide shared key loaded before being deployed, or all the network nodes should be securely instructed to use a new shared key.

3.2.3. Security

LEAP's goal is to satisfy the security properties of authentication and confidentiality in a wireless environment where the intruder may eavesdrop, inject packets, and replay messages [23]. The authors of LEAP also desire that the protocol will be robust and will survive in the face of security attacks and that the effects of any attacks be minimized (to a node's neighbours only). LEAP makes no claims as to defending against replay or denial of service attacks.

Table 3.1 : LEAP Security Characteristics

Protocol	C	F	I	Ava	IA	A
LEAP	Yes	No	No	No	Yes	No

C= Confidentiality, F=Freshness, I=Integrity, Ava=Availability, IA=Implicit Authentication, A=Authentication of User.

3.3. TINYSEC

Replacement for the unfinished SNEP, known as TinySec [11]. Inherently it provides similar services, including authentication, message integrity, confidentiality and replay protection. A major difference between TinySec and SNEP is that there are no counters used in TinySec.

Generally, the security of CBC-MAC is directly related to the length of the MAC. TinySec specifies a MAC of 4 Bytes, much less than the conventional 8 or 16 Bytes of previous security protocols. In the context of sensor networks this is not detrimental [11].

TinySec, a lightweight, generic security package that developers can easily integrate into sensor network applications. TinySec will cover the basic security needs of all but the most security critical applications.

In conventional networks, message authenticity, integrity, and confidentiality are usually achieved by an end-to-end security mechanism such as SSH [12], SSL [13], or IPSec [14] because the dominant traffic pattern is end-to-end communication; intermediate routers only need to view message headers and it is neither necessary nor desirable for them to have access to message bodies.

This is not the case in sensor networks. The dominant traffic pattern in sensor networks is many-to-one, with many sensor nodes communicating sensor readings or network events over a multihop topology to a central base station. However, neighboring nodes in sensor networks often witness the same or correlated environmental events, and if each node sends a packet to the base station in response, precious energy and bandwidth are wasted. To prune these redundant messages to reduce traffic and save energy, sensor networks use in-network processing such as aggregation and duplicate elimination [15, 16]. Since in-network processing requires intermediate nodes to access, modify, and suppress the contents of messages. End-to-end security mechanisms between each sensor node and the base station guarantee the authenticity, integrity, and confidentiality of these messages. End-to-end security mechanisms are also vulnerable to certain denial of service attacks. If message integrity is only checked at the final destination, the network may route packets injected by an adversary many hops before they are detected. This kind of attack will waste precious energy and bandwidth. Link-layer security architecture

can detect unauthorized packets when they are first injected into the network. Link-layer security mechanisms have been proposed for wired networks to resist similar denial of service attacks [17].

For the above reasons, they decided on link-layer security architecture for TinySec. Link-layer security mechanisms guarantee the authenticity, integrity, and confidentiality of messages between neighboring nodes, while permitting in-network processing. Despite the problems, end-to-end security mechanisms can still be useful in sensor networks and complement TinySec.

Table 3.2 : TINYSEC Security Characteristics

Protocol	C	F	I	Ava	IA	A
TINYSEC	Yes	No	No		Yes	Yes

C= Confidentiality, F=Freshness, I=Integrity, Ava=Availability, IA=Implicit Authentication, A=Authentication of User.

3.3.1. TinySec Design

There are two packet formats defined by TinySec. These are TinySec-Auth, for authenticated messages, and TinySec-AE, for authenticated and encrypted messages. For encryption, it uses CBC mode with cipher text stealing [11], and for authentication, CBC-MAC is used. TinySec XORs the encryption of the message length with the first plaintext block in order to make the CBC-MAC secure for variably sized messages. With authenticated encryption, TinySec encrypts the data payload and authenticates the packet with a MAC. The MAC is computed over the encrypted data and the packet header. In authentication only mode, TinySec authenticates the entire packet with a MAC, but the data payload is not encrypted.

- **Encryption:** Using semantically secure encryption typically requires two design decisions: selecting an encryption scheme and specifying the IV format. This design of TinySec uses a specially formatted 8 byte IV, and cipher block chaining (CBC) [18]. In this section, they introduce the structure of IV format and argue why CBC is the most appropriate encryption scheme for sensor networks [11].
- **Message integrity**: History has proven that using encryption without authentication is insecure [19, 20, and 21]. For example, flipping bits in unauthenticated encrypted messages can cause predictable changes in the plaintext [20], and without an authentication mechanism to guarantee integrity, receivers are unable to detect the changes. Unauthenticated messages are also vulnerable to cut-and-paste attacks [19]. In a cut-and-paste attack, an adversary breaks apart an unauthenticated encrypted message and constructs another message which decrypts to something meaningful. For example, if all the authorized nodes share a single key, an adversary can extract the encrypted data payload from a message to one node and send it to different node. Since the encrypted payload is unaltered, the second node will successfully decrypt and accept the message. To address these vulnerabilities, TinySec always authenticates messages, but encryption is optional. Message confidentiality is only necessary when there is something to keep secret. Consider a burglar alarm. The actual contents of an alarm message could be empty; receiving an alarm message signals an intrusion. Encryption is unnecessary and only increases latency, computation, and power consumption. However, most all applications require packet authenticity, meaning authorized nodes will not accept invalid messages injected by an adversary. In burglar alarm example, this means adversaries cannot trigger false alarms.

TinySec uses a cipher block chaining construction, CBC-MAC [22], for computing and verifying MACs. CBC-MAC is efficient and fast, and the fact that it relies on a block cipher as well minimizes the number of cryptographic primitives. CBC-MAC is provably secure [22], however the standard CBC-MAC construction is not secure for variably sized messages. Adversaries can forge a MAC for certain messages. Ref [22] suggests three alternatives for generating MACs for variable sized messages.

3.4. ZigBee

3.4.1. ZigBee Security Architecture

The concept of a “Trust Center” is introduced in the specification. Generally the ZigBee coordinator performs this duty. This trust center allows other devices to join the network and also distributes the keys. There are three roles played:

- trust manager, whereby authentication of devices requesting to join the network is done
- network manager, maintaining and distributing network keys, and
- configuration manager, enabling end-to-end security between devices [24].

It operates in both Residential Mode and Commercial Mode. The Trust Center running residential Mode is used for low security residential applications. Commercial Mode is designed for high-security commercial applications. In Residential Mode, the Trust Center will allow devices to join the network, but does not establish keys with the network devices. It therefore cannot periodically update keys and allows for the memory cost to be minimal, as it cannot scale

with size of the network. In commercial mode, it establishes and maintains keys and freshness counters with every device in the network, allowing centralized control and update of keys. This results in a memory cost that could scale with the size of the network [24].

There are three types of keys employed, the Master Key, the Link Key and the Network Key. Master keys are installed first, either in the factory or out of band. They are sent from the Trust Center and are the basis for long-term security between two devices. The Link key is a basis of security between two devices and the Network keys are the basis of security across the entire network. Link and Network keys, which are either installed in the factory or out of band, employ symmetrical key-key exchange (SKKE) handshake between devices. The key is transported from the Trust Center for both types of keys. This operation occurs in commercial mode, as residential mode does not allow for authentication.

3.4.2. ZigBee Security

ZigBee security specification employes a simpler and unified mode of operation of CCM (this mode is an amalgamation of both the encryption and authentication) defines key types (Master, Link, Network) and describes key setup and maintenance (Commercial, Residential) [24]. Additionally, ZigBee provides freshness through the use of freshness checks. These checks prevent replay attacks, as ZigBee devices maintain incoming and outgoing freshness counters. Whenever a new key is created, the counters are reset. It is postulated that devices that communicate once per second will not overflow their freshness counters for 136 years [24]. Message integrity and encryption are also provided under the ZigBee security specification, the operations of which are documented in [25] and [24]. Under the ZigBee specification, authentication is defined to provide assurance about the originator of a message. This prevents an attacker from mimicking the operation of another device in any attempt to compromise the network. Authentication is possible at both the network level and the device level. At the network level, authentication is achieved using a common network key, thus preventing outside attacks whilst adding very little in memory cost. Device level authentication is achieved by using unique link keys between pairs of devices. Insider and outsider attacks are now preventable, but there is a higher memory cost involved. Table 3 shows the various security characteristics of ZigBee protocol.

Table 3.3: ZIGBEE Security Characteristics

Protocol	C	F	I	Ava	IA	A
ZIGBEE	Yes	Yes	Yes	No	Yes	Yes

C= Confidentiality, F=Freshness, I=Integrity, Ava=Availability, IA=Implicit Authentication, A=Authentication of User.

3.5. SM (Security Manager)

A new method of key agreement, whereby, when a new device joins network, the Security Manager (SM) gives static domain parameters such as at the base station, the order of the curve and the elliptic curve coefficients [26]. After calculating a public key using the base point and a private key, the device sends a public key to the SM. Therefore the SM would have the public key list for all the devices in the network. They define two security levels (medium and high), based on the devices power and security policies. These two levels are defined by either normal or polynomial basis calculations. Elliptic Curve Cryptography (ECC) algorithms offer reasonable computational loads and smaller key lengths for equivalent security than other techniques. These smaller key lengths reduce the size of message buffers and reduce implementation cost of protocols. The EC-MQV (Menezes-Qu- Vanstone) scheme is more advanced than the Diffie-Hellman scheme, and the main idea is to prevent the man-in-the-middle attack and perform authentication of key holders. Under this scheme, each side of the communication holds two keys [26]. Devices in the network use initial trust parameters (pre-deployed recognition function) to establish the public key and ephemeral public key, which are in turn used for secure communication of the data payloads [26]. The overhead here will depend on the number of bits chosen for the elliptic curve system. An elliptical curve algorithm provides the same security for 160 bit key lengths as a symmetric algorithm can for 128 Byte lengths [26]. This level of security can then be increased as security needs to be increased and, therefore, allowing a variable overhead. Table 4 shows the various security characteristics of SM protocol.

Table 3.4: SM Security Characteristics

Protocol	C	F	I	Ava	IA	A
SM	Yes	No	No		Yes	Yes

C= Confidentiality, F=Freshness, I=Integrity, Ava=Availability, IA=Implicit Authentication, A=Authentication of User.

4. Conclusion

SPINS is one of the secure and efficient sensor network protocol. LEAP is a protocol that survives in the face of security attacks and that the effects of any attacks may be minimized. TINYSEC is a stronger and energy efficient protocol. In ZIGBEE protocol, concept of a “trust center” is introduced. SM uses the EC-MQV

scheme for key establishment, that is more advanced and main idea is to prevent the man-in-middle attack.

5. REFERENCES

[1] Y. Cheng, "Security mechanisms for mobile adhoc and wireless sensor network", Ph. D. thesis published in the University of Cincinnati, OHIO, 2008.

[2] W. Dargie, M. Zimmerling, "Wireless sensor networks in the context of developing countries", International journal of information and communication technology education (To appear),2009.

[3] Jason L. Hill, "System architecture for wireless sensor networks", Ph. D. thesis published in the University of California at Berkeley, Spring 2003.

[4] Chris Townsend, Steven Arms, MicroStrain, Inc., "Wireless sensor networks: principles and applications", Chapter 22 in sensor technology handbook by Jon S. Wilson, pp. 575-587.

[5] http://en.wikipedia.org/wiki/Wireless_Sensor_Networks.

[6] A. Perrig, R. Szewczk, J. D. Tygar, V. Wen and D. E. Culler, "SPINS: Security Protocols for Sensor Networks", wireless networking, vol.8, pp. 521-534, Netherlands, Sept 2002.

[7] J. Kulik, W. R. Heinzelman, and H. Balakrishnan, "Adaptive protocol for information disseminating in wireless sensor networks" In Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, pp. 174-185, 1999.

[8] A. Perrig, R. Canneti, J. D. Tygar, and D. Song, "The TESLA broadcast authentication protocol", CryptoBytes, vol.5, no.2, pp. 2-13, 2002

[9] J.-P.Kaps, "Cryptography for ultra-low power devices", Ph. D. thesis, at Worcester Polytechnic Institute, 2006.

[10] S. Zhu, S. Setia, and S. Jajodia, "Leap: efficient security mechanisms for large-scale distributed sensor networks", In Proceedings of 10th ACM Conference on Computer and Communications Security, pp. 62-72, New York, USA, 2003.

[11] C.Karlof, N.Sastry, D.Wagner, "TinySec: A link layer security architecture for wireless sensor networks", In Proceedings of the 2nd International conference on embedded networked sensor systems, pp.162–175, Baltimore, MD, USA, November 2004.

[12] I-H. Huang, W. J. Tzeng, S.-W. Wang, C.-Z. Yang, "Design and implementation of a mobile SSH protocol", pp. 1-4, TENCON, Nov. 2006.

[13] OpenSSL. http://www.openssl.org.

[14] Security architecture for the Internet Protocol, RFC 2401, November 1998.

[15] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong, "TAG: A tiny aggregation service for ad-hoc sensor networks", In Proceedings of the 5th symposium on operating system design and implementation (OSDI), pp. 131-146, December 9-11, 2002.

[16] Samuel R. Madden, Robert Szewczyk, Michael J. Franklin, and David Culler, "Supporting aggregate queries over ad-hoc wireless sensor networks", 4th IEEE Workshop on mobile computing and systems applications, pp. 49, 2002.

[17] Lyes Khelladi, Yacine Challal, Abdelmadjid Bouabdallah, Nadjib Badache, "On security issues and challenges in embedded systems: challenges and solutions", International journal of information and computer security 2, vol. 2, pp. 140-174, 2008.

[18] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption: analysis of the DES modes of operation", In Proceedings of 38th annual symposium on foundations of computer science, 1997.

[19] Steven M. Bellovin, "Probable plaintext cryptanalysis of the IP Sec protocols", IEEE, pp. 52-59, 1997.

[20] Nikita Borisov, Ian Goldberg, and David Wagner, "Intercepting mobile communications: The insecurity of 802.11", In 7th annual International conference on mobile computing and networking , pp. 180-189, Rome, Italy, 2001.

[21] Hugo Krawczyk, "The order of encryption and authentication for protecting communications", In Proceedings of the 21st annual International cryptology conference on Advances in Cryptology, vol. 2139, pp. 310-331, January 2001.

[22] Mihir Bellare, Joe Kilian, and Phillip Rogaway, "The security of the cipher block chaining message authentication code", Journal of computer and system sciences, vol.61, no.3, pp. 362-399, December 2000.

[23] D. Boyle, T. Newe, "Security protocols for use with wireless sensor networks: A survey of security architecture", In Proceedings of the Third International conference on wireless and mobile communications, pp. 54, 2007.

[24] ZigBee Alliance ZigBee Security Specification Overview [online], available: http://www.zigbee.org/en/events/documents/december2005_open_house presentations/zigbee_security_layer_technical_overview.pdf

[25] YueFeng Ma, "Application of SoC Zigbee technology in the remote reading meter system", In Proceedings of world academy of science and technology , vol. 21, pp. 277-279, May 2007.

[26] Heo, J., Hong, "Efficient and authenticated key agreement mechanism in low-rate WPAN environment", International Symposium on wireless pervasive computing, pp. 1-5, Phuket, Thailand 16 – 18 January 2006, IEEE 2006