Security in Wireless Sensor Networks – A Study

L. Ertaul[1], **M. Ganta**[2]
[1]Department of Mathematics and Computer Science, California State University, East Bay, Hayward, CA, USA,
Levent.Ertaul@csueastbay.edu
[2]Department of Mathematics and Computer Science, California State University, East Bay, Hayward, CA, USA,
mganta@horizon.csueastbay.edu

Abstract - *Wireless Sensor Networks (WSNs) are used in many commercial, military, industrial, research and medical applications. Because of the limited resources of the sensor nodes in the wireless environment, these networks impose special security requirements besides to the security needs in traditional networks. This study presents the security problems in WSNs and discusses the security protocols based on three different security mechanisms: Security Protocols for WSNs (SPINS), which uses symmetric key cryptography, then TinySec, which is based on Link layer encryption, and later TinyPK (the Public Key Infrastructure solution). This study also presents the security solutions offered, and also security problems not addressed, by these protocols.*

Keywords: WSNs, SPINS, TinySEc, TinyPK.

1 Introduction

Wireless Sensor Network (WSN) is a heterogeneous system with a collection of sensors distributed in irregular patterns in remote areas, and often in hostile environments, without any pre-deployed architecture, and with limited hardware resources. The number of sensors in the WSN may range from few hundred to few hundred thousands. These sensor nodes will have limited resources of power, storage, communication and processing capabilities. [1][2][3]

WSNs are used in wide variety of applications. WSN can be used in military applications with sensors operating unsupervised in the hostile environments for target identification, tracking and data collection. WSN can also be used in civil applications for disaster relief, emergency rescue, burglar alarms and smart homes, patient monitoring etc., and also in industrial applications such as environmental control, Energy Management, inventory control and structural health monitoring. WSN is also used in habitat and wild life monitoring in the research of life sciences. [2][4][5]

WSN can be Hierarchical WSN or Distributed WSN by its architecture. In *Hierarchical Wireless Sensor Network (HWSN)*, there is a hierarchy between the nodes based on their capabilities: Base stations, Cluster Heads, and Sensor Nodes. Base stations (BS) are many orders of magnitude more powerful than sensor nodes and cluster heads. A BS is typically a gateway to another network, a powerful data processing / storage center, or an access point for human interface. BSs collect sensor readings, perform costly operations on behalf of sensor nodes and manage the network. They are used as key distribution centers. Nodes with better resources, named as cluster heads, may be used to collect and merge local traffic and send it to BSs. Transmission power of a BS is usually enough to reach all sensor nodes, but sensor nodes depend on the ad-hoc communication to reach BSs. The BS accesses individual nodes using source routing. In *Distributed Wireless Sensor Network (DWSN)*, there is no fixed infrastructure, and network topology is not known prior to deployment. Most of the situations, manual deployment of WSN is impossible because of the hostile environment and/or the number of the sensors nodes is too huge. Then the deployment has to be performed by randomly scattering the sensor nodes to target area. It may be possible to provide denser sensor deployment at certain spots, but exact positions of the sensor nodes can't be controlled. Thus, network topology is not known precisely prior to deployment. [3]

This paper is organized as follows. Section 2 explains the security problems related to WSNs. In later sections, three popular WSNs security protocols, based on three different security mechanisms, are studied. Section 3 covers the Security Protocols in WSNs (SPINS). Link layer encryption protocol, TinySec, is explained in section 4. Section 5 describes the TinyPK. Section 6 concludes this study explaining the need of new protocols and why the current protocols are not satisfying all the WSN security requirements.

2 Security Requirements of WSNs

WSNs pose unique new security challenges, which prevent direct application of traditional security techniques. First, sensor devices are limited in their processing power, memory, size, and communication capabilities. These devices have very little computational power; even efficient public-key cryptography and fast symmetric ciphers must be used with care. There is considerable requirement to ensure that our security protocols use a minimal amount of the limited RAM. Additionally, communication bandwidth is extremely dear: any message expansion caused by security mechanisms comes at significant cost. Energy is the scarcest resource of all: each milliamp consumed is one milliamp closer to death, and as a result, nearly every aspect of WSNs must be designed with power consumption in mind. [4]

Second, in contrast to traditional networks, WSNs are often deployed in accessible areas to intruders, so adversaries are not restricted to using WSN's hardware, presenting a risk of physical attacks and other security problems. They can even interact with the network from a distance by using expensive radio transceivers and powerful workstations.

Third, as the wireless communication is broadcast in nature, the traditional key distribution and message authentication solutions would pose problems in the low power WSNs. In a broadcast medium, adversaries can easily eavesdrop on, intercept, inject, and alter transmitted data.

Due to all the above reasons, the traditional security mechanisms are inadequate, for WSNs, and different security techniques are needed.

The following are the major security requirements in WSNs.

Confidentiality. It provides privacy of the wireless communication channels to prevent eavesdropping. Providing confidentiality is mandatory security requirement in many of WSN applications. Preferably, an encryption scheme should not only prevent message recovery, but also prevent adversaries from learning even partial information about the messages that have been encrypted even when the multiple encryptions of the same plaintext are seen. This feature of encryption scheme is also known as semantic security [2][4][9].

Integrity. Integrity [2][9] is ensuring that message or the entity under consideration is not altered. If an adversary modifies a message from an authorized sender while the message is in transit, the receiver should be able to detect this tampering.

Availability. Availability is ensuring that service offered by whole WSN, by any part of it, or by a single sensor node must be available whenever required. This also means robustness to communication without Denial-of-Service (DoS). WSNs are also vulnerable to resource consumption attacks. Adversaries can repeatedly send packets to drain the nodes' batteries or waste network bandwidth. [4][9][10]

Key establishment and trust setup. When setting up a WSN, one of the first requirements is to establish cryptographic keys for later use. Key-establishment techniques need to scale to networks with hundreds or thousands of nodes. [4][9][10]. The simplest solution for key establishment is a network-wide shared key. But, the compromise of a single node in a network would lead to the reveal the secret key and thus allow decryption of all network traffic. Another solution proposes to use a single shared key to establish a set of link keys, one per pair of communicating nodes, and then erase the network-wide key after setting up the session keys. However, this key-establishment process does not allow addition of new nodes after initial deployment [4]. Another option is to pre-configure the network with a shared unique symmetric key between each pair of nodes, but it doesn't offer scalability. Bootstrapping keys using a trusted base station is another option. Here, each node needs to share only a single key with the BS and set up keys with other nodes through the base station. This option makes the BS a single point of failure, but because there is only one BS, the network may incorporate tamper-resistant packaging for the BS, which can reduce the threat of physical attack. [4]. Another approach is random-key pre-distribution protocols [5] in which a large pool of symmetric keys is chosen and a random subset of the pool is distributed to each sensor node before installation. Here the advantage is that key establishment does not need a central trusted BS. But, the disadvantage of this approach is that attackers who compromised sufficiently many nodes could also reconstruct the complete key pool and break this scheme. Finally this leads that we need a secure and efficient key-distribution mechanism allowing simple key establishment for large-scale WSNs. [4]

Authentication. In the two-party communication case, data authentication can be achieved through a purely symmetric mechanism: The sender and the receiver share a secret key to compute a message authentication code (MAC) of all communicated data. When a message with a correct MAC arrives, the receiver knows that it must have sent by the actual sender. This style of authentication cannot be applied to a broadcast setting, without placing much stronger trust assumptions on the network nodes. If one sender wants to send authentic data to mutually untrusted receivers, using a symmetric MAC is insecure: any one of the receivers knows the MAC key, and hence, could impersonate the sender and forge messages to other receivers. Hence, we need an asymmetric mechanism to achieve authenticated broadcast. [2][4][9]

Freshness. Data freshness means the message received is the message sent by the source but not a replayed message sent by the adversary. Basic wireless communication is not secure. Because it is broadcast, any adversary can eavesdrop on traffic, inject new messages, and replay old messages. There are two types of freshness: weak freshness, which provides partial message ordering, but carries no delay information, and strong freshness, which provides a total order on a request–response pair, and allows for delay estimation. [2]

Resilience to node capture. In traditional networks, physical security is achieved by not allowing the physical access of the network to the unauthorized persons. As this cannot be done in the case of WSNs, new solutions need to be used for achieving this. Exposure of sensor node raises the possibility that an attacker might capture sensor nodes, extract cryptographic secrets, modify their programming, or replace them with malicious nodes under the control of the attacker. Defenses based on redundancy are particularly well suited to WSNs. Node capture is one of the most vexing problems in WSN security. No good solution is found for this yet. [4][9]

Secure Routing protocols. Routing protocols are also susceptible to node-capture attacks. Routing and data forwarding is an essential service for enabling communication in WSNs. An attacker might launch DoS attacks on the routing protocol, preventing communication. The simplest attacks involve injecting malicious routing information into the network, resulting in routing inconsistencies. Simple authentication might guard against injection attacks, but some routing protocols are susceptible to replay by the attacker of legitimate routing messages. [9]

Practical Problems. Large scale airdropped WSNs actually do not work for many practical reasons. First, sensors must be spaced rather closely to avoid gaps in the connectivity. Increasing the sensor node's radio transmission range would not help in reducing the number of sensors in the WSN,

because energy consumption for a single transmission is independent of the radio range. This leads to the fact that the battery is the most important parameter in deciding the security of WSN. But the energy density of the batteries is improving only few percentage points every year. Second, in battle field or terrorist watch areas, having numerous sensors will lead to higher chances of noticing the node and its capture. Third, when the sensors are air dropped, it is also hard to find the actual physical location of the sensor. Another important problem is maintenance-free operation of the network. In addition to that in any kind of WSN, used for of intruder detection, sensor node cannot distinguish between a human and an animal because of its limited resources. [10]

Hostile environments, lack of fixed infrastructure, limited resources, and broadcast communication lead to the need of special solutions for WSN security requirements, at least for better solutions. In the following sections, Security Protocols for WSNs (SPINS), TinySec and TinyPK are studied, and these protocols offer different kinds of security solutions for the above security problems in WSNs with some unanswered problems.

3 Security Protocols for WSNs (SPINS)

SPINS has two security protocols- Secure Network Encryption Protocol (SNEP) and Micro Timed Efficient Stream Loss-tolerant Authentication (μTESLA). SNEP provides data confidentiality, two-party data authentication, integrity, and evidence of data freshness. μTESLA provides authenticated broadcast for the limited resource environments like WSNs. Both the mechanisms are based on symmetric encryption since asymmetric encryption leads to higher overheads of computation, storage and communication which are not suitable for WSNs. [2]

3.1 Secure Network Encryption Protocol (SNEP)

In SNEP [2], at creation time, each node gets a *master secret key* that it shares with the base station. All other keys are derived from this master key. The two communicating parties *A* and *B* share a master secret key *X*, and they derive independent keys using the pseudorandom function *F* [11] and the master secret key X: encryption keys $K_{AB} = F_X$ *(1)* and $K_{BA} = F_X$ *(3)* for each direction of communication, and MAC keys $K'_{AB} = F_X$ *(2)* and $K'_{BA} = F_X$ *(4)* for each direction of communication.

SNEP uses *Counter exchange protocol* for bootstrapping the counters initially, and also for synchronizing the counter values as shown in Figure 1.

Since sensors and the communicating parties share the counter and increment it after each block, the sender can save energy by sending the message without the counter.

The encrypted data has the following format: Ciphertext, E is encrypted message of (M, C) with the encryption key K_{AB}, where M is the message, and the counter is *C*. The MAC_{AB} is calculated with MAC key $K'_{AB,}$ E and counter C. The complete message that *A* sends to *B* is *A*→ *B*: E and MAC_{AB}. The message is also created in the same manner for the opposite direction.

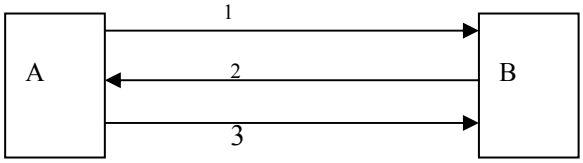

Figure.1. Counter Exchange protocol in SNEP. The messages sent are marked with the number of the step given below.

1) A →B: C_A,
2) B →A: C_B, calculated MAC of (C_A and C_B) with MAC key K'_{BA},
3) A →B: Calculated MAC of (C_A and C_B) with MAC key K'_{AB}.

The plain SNEP in Figure 2 only offers weak freshness. Since the sender increments the counter after each message, the receiver verifies weak freshness by verifying that received messages have an increasing counter, assuring that the message must have been sent after the previous message it received correctly (that had a lower counter value), but no absolute assurance to node *A* that a message was created by *B* in response to an event in node *A*. This enforces a message ordering and yields weak freshness

If strong freshness is required, Figure 3, Node *A* generates a nonce (N_A) randomly and sends it along with a request message R_A to node *B*. Then *B* returns the nonce with the encrypted response message R_B, along with the MAC of (encrypted message R_B, C_B, *and* N_A). If the MAC verifies correctly, node *A* knows that node *B* generated the response after it sent the request.

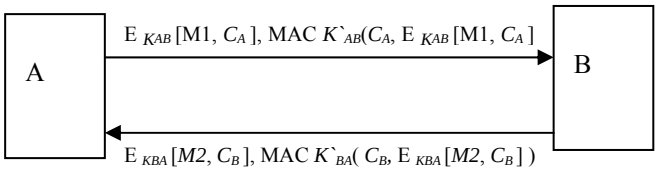

Figure.2. Plain SNEP protocol with weak freshness.

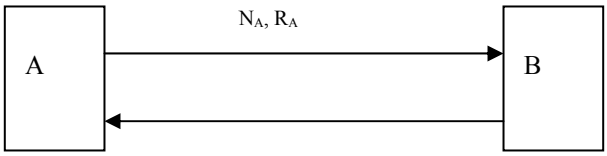

Figure.3. SNEP protocol with strong freshness.

In SNEP, to save code space, same encryption function is used for both encryption and decryption. The counter (CTR) mode [12] [13] [14] of block ciphers has this property. CTR-mode offers semantic security. SNEP implements RC5 as a macro and only expose interfaces to the MAC and CTR-ENCRYPT functions.

To minimize power requirements, SPINS use a MAC function as pseudo-random number generator (PRG), with the secret key X_{rand}.

In summary, SNEP offers the following security features:

Low communication overhead. Because the counter state is kept at each end point, and does not need to be sent in each message, this reduces the communication overhead.

Replay protection. The counter value in the MAC prevents replay of old messages.

Semantic security: Since the counter value is incremented after each message, the same message is encrypted differently each time.

Two-party Data authentication. The sender and the receiver share a secret key to compute a message authentication code (MAC) of all communicated data. When a message with a correct MAC arrives, the receiver knows that the message originated from the claimed sender. This also asserts the integrity of the message.

3.2 Micro Timed Efficient Stream Loss-Tolerant Authentication (μTESLA)

SNEP provides only point-to-point authentication. Authentication of broadcast messages is also an important security requirement. If convinced to accept forged or modified commands or data, sensor nodes may perform unnecessary or incorrect operations, and cannot fulfill the intended purposes of the network.

SPINS use μTESLA [2], which is an adoption of TESLA for broadcast authentication in WSNs. TESLA is a broadcast stream authentication protocol. TESLA uses delayed key disclosure mechanism where the key used to authenticate i^{th} message is disclosed along with $(i+1)^{th}$ message. The difference between TESLA and μTESLA is that TESLA uses asymmetric cryptography to bootstrap new receivers, whereas μTESLA depends on symmetric cryptography with the master key shared between the sender and each receiver to bootstrap the new receivers individually.

μTESLA provides authentication for data-broadcasts, and requires that base station and sensor nodes be loosely time synchronized. SPINS employ base station as key distribution centre.

Authenticated broadcast requires an asymmetric mechanism; otherwise any compromised receiver could forge messages from the sender. But this is suitable for traditional networks, and leads to high computation, storage, and communication overhead in case of WSN. μTESLA overcomes this problem by introducing asymmetry through a delayed disclosure of authentication keys in the key chain. Each key in the key chain is the image of the next key under the one-way hash function.

μTESLA requires sensor nodes to bootstrap from the Base Station (BS); that is, they receive the first key of the chain, which is called key chain commitment. Bootstrapping procedure requires unicast communication, and can be secured with pair-wise keys. Each node can easily perform time synchronization and retrieve an authenticated key of the key chain for the commitment in a secure and authenticated manner, using the SNEP building block.

The sender broadcasts the current key periodically in a special packet. The sender generates the one-way key chain right-to-left by repeatedly applying the one-way function *F*. The sender associates each key of the one-way key chain with a time interval. Time runs left-to-right, so the sender uses the keys of the key chain in reverse order, and computes the MAC of the packets of a time interval with the key of that time interval. The key disclosure time delay is on the order of a few time intervals.

Basically, BS randomly selects last key K_n of a chain, and applies one-way function F [15] to generate the rest of the chain $K_i = F(K_{i+1})$, $0 \le i \le n-1$, where the secret key K_i is assigned to the *i*th time interval as shown in the Figure. 4. Given K_i, every sensor node can generate the key sequence $K_1,\ldots, K_{i-1}$. However, given K_i, no one can generate K_{i+1}. The initial commitment of the key chain K_0 is provided with strong freshness and authentication by SNEP. At i^{th} time slot, BS sends message along with the MAC of the message created with the key K_i. Sensor nodes store the message until BS discloses the verification key in $(i+1)^{th}$ time slot.

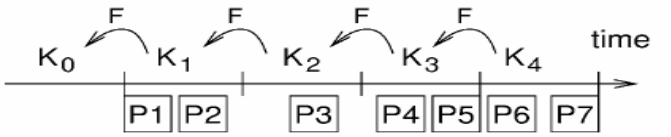

Figure. 4. μ-TESLA one-way key-chain. K_0 is the initial commitment of the key-chain, which is provided through unicast-authenticated communication through SNEP.

In a one-way key chain, the keys are self-authenticating. So the receiver can authenticate subsequent keys of the one-way key chain using one authenticated key. For example, if the node receives a new key K_i, then later that node can verify disclosed verification key K_{i+1} by using the previous key K_i as $K_i = F(K_{i+1})$.

The node-to-base-station authenticated channel is used to bootstrap the authenticated broadcast between a new receiver and the base station. There are two ways of a node broadcasts authenticated data. In the first solution, the node broadcasts the data through the base station. It uses SNEP to send the data in an authenticated way to the base station, which subsequently broadcasts it. In the second solution, the node broadcasts the data. But, the base station keeps the one-way key chain and sends keys to the broadcasting node as needed.

SPINS don't address all the security problems in WSNs. First, problem of information leakage through covert channels is not addressed. Second, it does not deal completely with compromised sensors, it merely ensure that compromising a single sensor does not reveal the keys of all the sensors in the network. Third, it does not deal with DoS attacks. Since we operate on a wireless network, an adversary can always perform a DoS attack by jamming the wireless channel with a strong signal. Finally, Diffie-Hellman style key agreement [17] or digital signatures, to achieve non-repudiation, are not available through SPINS like with any other symmetric key mechanism.

In μTESLA, nodes are required to store a message until the authentication key is disclosed. This operation may create storage problems, and encourages DoS types of attacks. An adversary may jam key disclosure messages to saturate storages of sensor nodes. Another problem is Bootstrapping a new receiver requires unicast communication. This leads to high volume of packets in large WSN for bootstrapping a large group of new receivers and creates scalability problems.

The major barrier of using μTESLA in large WSNs lies in its difficulty to distribute the key chain commitments to a large number of sensor nodes. The essential reason for this

difficulty is the mismatch between the unicast distribution of key chain commitments and the authentication of *broadcast* messages. That is, the technique is developed for broadcast authentication, but it relies on unicast technique to distribute the initial parameters.

Multilevel μTESLA is based on μTESLA. Multilevel μTESLA satisfies several nice properties, including low overhead, tolerance of message loss, scalability to large networks, and resistance to replay attacks as well as DoS attacks. [18] We have three variations of multilevel μTESLA schemes. The first variation is named *DoS-tolerant multilevel μTESLA* and is suitable for WSNs where the base station is not very resourceful. The second variation is named *DoS-resistant multilevel μTESLA*. It is suitable for WSNs with relatively short lifetime and relatively powerful base stations. The third variation is *hybrid multilevel μTESLA*. It is a trade-off between the above two variations. It sacrifices certain immediate authentication capability to exchange for less pre-computation requirement.

4 TinySec

TinySec is the first fully implemented link layer security architecture for WSNs. TinySec is a lightweight, generic security package. TinySec offers confidentiality, message integrity, and authenticity through link layer encryption. [19]

In traditional networks, message authenticity, integrity, and confidentiality are usually achieved by an end-to-end security mechanism such as SSH [20], SSL [21], or IPSec [22]. But, this is not the case in WSNs. Neighboring nodes in these networks often have 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, WSNs use in-network processing such as aggregation and duplicate elimination. For this, intermediate nodes need to access, modify, and suppress the contents of messages. In this scenario, we cannot use end-to-end security mechanisms between each sensor node and the base station to guarantee the authenticity, integrity, and confidentiality of these messages. Another disadvantage of using end-to-end security mechanisms in WSNs is that message integrity is only checked at the final destination, then network may route packets injected by an adversary many hops before they are detected. This kind of attack will waste precious energy and bandwidth of WSNs and could lead to DoS. Link-layer security architecture can detect unauthorized packets when they are first injected into the network. Link-layer security mechanisms guarantee the authenticity, integrity, and confidentiality of messages between neighboring nodes, while permitting duplicate message elimination and data aggregation.

TinySec provides two different security options: authenticated encryption (TinySec-AE) and authentication only (TinySec-Auth). In TinySec-AE, 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 without the data payload encryption.

TinyOS packet format and TinySec packet formats in AE and Auth mode are shown in the Figure 5.

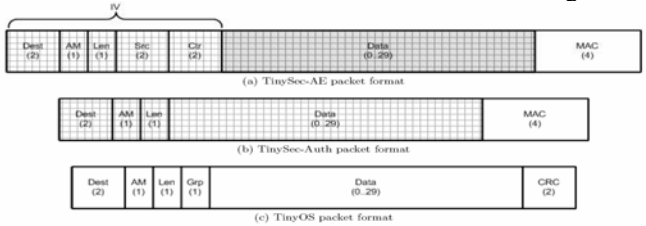

Figure. 5. The TinySec and TinyOS packet formats. The MAC protects fields, which have been hatched. In TinySec-AE, the data field, shaded gray, is encrypted.

TinySec uses symmetric key mechanisms. In some of the modes of operation using block ciphers, Initialization Vector (IV) will be given as input the encryption algorithm while creating the ciphertext. In TinySec, the IV is concatenation of destination address, active message handler type, length of the data payload, source address of the sender, and a 16-bit counter. The counter starts at 0, and the sender increases it by 1 after each message sent.

In stream ciphers, if the same IV is ever used to encrypt two different packets, then it is often possible to recover both plaintexts. To guarantee that IVs are never reused requires IVs to be fairly long, say, at least 8 bytes. Since one of our goals, in WSN, is to minimize packet overhead, adding 8 additional bytes to a 30-byte packet is unacceptable. Then, the alternatives is having shorter IVs and accept that IV reuse will occur. But the stream ciphers cannot offer IV reuse, so block cipher should be used in TinySec. Examples of block ciphers include DES [23], Skipjack [24], AES [25], and RC5 [26]. Since we usually want to encrypt and authenticate messages longer than 8 or 16 bytes, block ciphers require a mode of operation to encrypt longer messages. For a k byte block cipher, a mode of operation typically breaks a message into segments of k bytes and uses the block cipher in a special way to encrypt the message block by block. Block ciphers in Cipher Block Chaining (CBC) mode [23] leaks only a small amount of information in the presence of repeated IVs, a significant improvement over a stream cipher. DES and AES are slow on sensors. The protocol implementation originally was done by RC5 and Skipjack. The default block cipher is Skipjack. When a reference is made to a TinySec key, it means a pair of Skipjack keys, one for encrypting data, and one for computing MACs.

The sender to confirm the authenticity and integrity of the message by the receiver creates message Authentication Code (MAC). The sender computes a MAC over the packet with the secret key and includes the MAC with the packet. A receiver sharing the same secret key recomputes the MAC and compares it with the received MAC value. If they are equal, the receiver accepts the packet and rejects it otherwise. MACs must be hard to forge without the secret key. This implies if an adversary alters a valid message or injects a bogus message, she or he cannot compute the corresponding MAC value, and authorized receivers will reject these messages.

TinySec uses a cipher block chaining construction, CBC, for computing and verifying MAC. CBC-MAC is efficient and fast, and it relies on a block cipher as well minimizes the number of cryptographic primitives. This works well in the limited memory available in WSN. Standard CBC-MAC construction is not secure for variably sized messages. Adversaries can forge a MAC for certain messages. The variant used in TinySec XORs the encryption of the message length with the first plaintext block.

The security of CBC-MAC is directly related to the length of the MAC. For a 4 byte MAC, an adversary has a 1 in 232 chance in blindly forging a valid MAC for a particular message. Sensor nodes do not have enough energy to receive that many messages. To detect such an attack, nodes could signal the base station when the rate of MAC failures exceeds some predetermined threshold.

The default block cipher in TinySec is Skipjack, which is suitable for sensor nodes. RC5 also can be used in TinySec, but RC5 may cause little more overhead, as it needs longer keys. DES and AES are very slow on sensor nodes. When we refer to a TinySec key, we mean a pair of Skipjack keys, one for encrypting data, and one for computing MACs.

Researchers are currently exploring key update protocols in TinySec. However, the TinySec protocol is not limited to any particular keying mechanism; any can be used in conjunction with TinySec. A keying mechanism determines how cryptographic keys are distributed and shared throughout the network. The TinySec protocol can be used in conjunction with any keying mechanism. Network-wide keying cannot protect against node capture attacks. If an adversary compromises a single node or learns the secret key, he can eavesdrop on traffic anywhere in the network. Another way is that per-link keying, a type of in-network processing where nodes take actions based on messages they overhear, and local broadcast, where nodes can cheaply send a packet to all their neighbors. Since a node cannot decrypt and authenticate messages not addressed to it, passive participation and local broadcast are incompatible with per-link keying.

TinySec's shared keys do allow for efficient, secure communications among nodes. But, for the replay protection, a counter should be sent with each message. This leads to saving one counter value, for each sender, in the receiving sensor node. If the sensor network is huge, then the each sensor node should have a big table of counter values containing last counter value from each other node of the network. This is an expensive solution in limited memory WSNs. TinySec doesn't provide replay protection of the message, one of the most important security requirements. Usage of TinySec increase the energy, bandwidth, and latency overhead due to increased packet length due to TinySec, extra computation time and energy needed for TinySec keys.

5 TinyPK

TinyPK [27] allows authentication and key agreement between a sensor network and a third party as well as between two WSNs. With symmetric encryption, proper key management is a fundamental concern. Public key (PK) technology is a widely used tool to support symmetric key management in the realm of Internet hosts and high-bandwidth interconnections. The TinyPK system demonstrates that a public-key based protocol is feasible for an extremely lightweight sensor network. Incorporating the use of TinySec or any other symmetric encryption service for mote (sensor node) networks, TinyPK provides the functionality needed for a mote and a third-party to mutually authenticate to each other and to communicate securely. TinyPK is based on the well-known RSA cryptosystem [28], using e=3 as the public exponent.

TinyPK requires a modest amount of public-key infrastructure. The first element of the infrastructure is a Certification Authority (CA), which is an entity with a private and public key pair that is trusted by all friendly units. Any third party that wishes to interact with the motes also requires its own public/private key pair and must have its public key signed (not on a hash of the data, but by transforming the data directly) by the CA's private key, thus establishing its identity. Finally, as each mote is loaded with software before being deployed to the field, it must have the CA's public key installed. Traditionally, a public key is made part of a certificate (e.g. an X.509 certificate) but TinyPK eliminates certificates as WSNs are assumed to not have the processing power or the data context to make use of certificates, e.g., no real-time access to the CA infrastructure. Without a certificate structure, there is no direct way to deal with compromise of an external party private key. TinyPK try to minimize the damage by such a compromise but there are no recovery mechanisms for compromise of private keys.

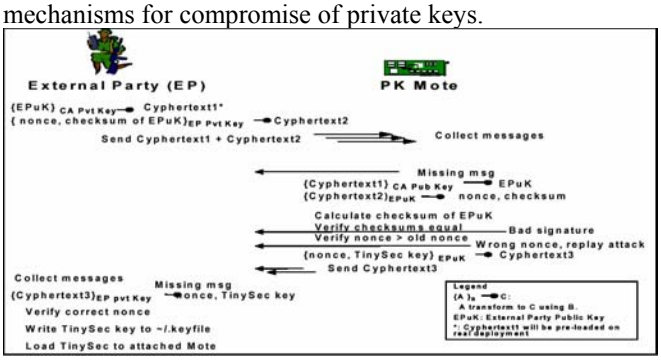

Figure. 6. TinyPK External Party Protocol Exchange.

TinyPK provides challenge-response protocol as shown in Figure 6, that authenticates the external party to the sensor network and securely transfers a session key from the sensor network to the third party. To perform authentication, the external party submits its signed public key and some text signed with its private key. Protocol operation starts when the third party provides a challenge to the sensor network. This challenge consists of two parts: The first is its own public key, signed by the CA private key; the second is a compound object consisting of a nonce (a timestamp) and a message checksum, signed with the third party's own private key. This information is not encrypted. The nonce serves to detect replay attacks, wherein a malicious party records previous valid messages and rebroadcasts them in order to provide false identification or otherwise attack a system. The

checksum is used to insure message integrity. Upon receipt of the message, a sensor node uses the preloaded CA public key to verify the first part of the challenge and extract the third party's public key. It then uses this public key to verify the second part of the message and extract the nonce and checksum. The nonce and checksum are validated. If they pass validation, the third party has successfully authenticated to the sensor network and is considered to be an authorized entity for sensor data. The sensor node now encrypts the session key plus the received nonce using the third party's public key. This combination is sent back to the third party, which decrypts it using its private key, checks that the nonce is the same as the one it sent, and if so, can record the session key for future use[29].

TinyPK currently relies on conventional modular arithmetic cryptosystems. However, there are several options for more energy efficient cryptosystems, including Elliptic Curve Cryptography (ECC) [30], and Efficient and Compact Subgroup Trace Representation (XTR) [31].

But TinyPK does not explain the revocation of the compromised private keys. It provides limited protection against DoS attacks. Another problem is that WSNs can have large number of motes, and the use of multiple session keys will be required for each mote. This is still a research topic.

6 Conclusions

In summary, SPINS do not address the information leakage through covert channels. Covert channels will have communications through unsecured procedures. It also does not deal with the DoS attacks. Besides to that, SPINS do not address compromised node issue except making sure it cannot reveal keys of other sensors. Most importantly, it cannot provide non-repudiation.

TinySec doesn't provide replay protection of the message. Each sensor node needs to have one counter value for each session. This leads to huge consumption of memory when a sensor node has multiple sessions in a large WSN. Besides to this, TinySec increases the overhead of processing time and communication because TinySec increases the TinyOS packet size.

TinyPK provides only limited protection against DoS attacks. TinyPK hasn't provided solution for compromised private keys. TinyPK does not address handling multiple session keys by a mote.

So far in all the existing solutions, there are significant tradeoffs among the WSN security parameters. However, the research in WSN security protocols can only solve some security problems, sometimes with significant tradeoffs. But they cannot eliminate all WSN security problems. Power of the battery is a big factor in removing the practical obstacles in the security of WSN. As the energy density of the battery is increasing at very low rate, we will continue to see the security problems / tradeoffs in WSN, at least for some more years.

7 References

[1] David E. Culler, Wei Hong, "Wireless Sensor Networks: Introduction" Communications Of The ACM, June 2004/Vol. 47, No. 6, pp. 30-33.

[2] Adrian Perrig, Robert Szewczyk, J.D. Tygar, Victorwen and David E. Culler, "SPINS: Security Protocols for Sensor Networks" Wireless Networks 8, 521–534, 2002 ©2002 Kluwer Academic Publishers. Manufactured in The Netherlands.

[3] "Key Distribution Mechanisms for Wireless Sensor Networks: A survey", TR-05-07, Department of Computer Science, Rensselaer Polytechnic Institute.

[4] Adrian Perrig, John A. Stankovic, David Wagner, "Security in Wireless Sensor Networks" Communications Of The ACM, June 2004/Vol. 47, No. 6, pp. 30-33

[5] Roberto Di Pietro, Luigi V. Mancini, and Alessandro Mei, "Random-Key Assignment for Secure Wireless Sensor Networks", Proceedings of 1st ACM Workshop Security of Adhoc and Sensor Networks Fairfax, Virginia

[6] Jason Hill, Mike Horton, Ralph Kling, Lakshman Krishnamurthy, "The Platforms Enabling Wireless Sensor Networks" Communications Of The ACM, June 2004/Vol. 47, No. 6, pp. 41-46.

[7] Crossbow Technology Inc., "Motes, smart dust sensors, wireless sensor networks", http://www.xbow.com

[8] J. Hill and D. Culler, "Mica: A wireless platform for deeply embedded networks", IEEE MICRO, 22(6): 12-24, 2002.

[9] Elaine Shi, and Addrian Perrig, "Designing Secure Sensor Networks", IEEE Wireless Communications, December 2004.

[10] Chandana Gamage, Kemal Bicakci, Bruno Crispo, and Andrew S. Tanenbaum, "Security for the Mythical Air-dropped Sensor Network", Proc. 11th IEEE Symp. Computers and Communications, IEEE CS Press, 2006, pp. 41-47.

[11] O. Goldreich, S. Goldwasser and S. Micali, "How to construct random functions", Journal of the ACM 33(4) (1986) 792–807.

[12] M. Bellare, A. Desai, E. Jokipii and P. Rogaway, " A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation", in: Symposium on Foundations of Computer Science (FOCS)(1997).

[13] W. Diffie and M.E. Hellman, "Privacy and authentication: An introduction to cryptography", Proceedings of the IEEE 67(3) (1979) 397–427.

[14] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, "Handbook of AppliedCryptography" (CRC Press, 1997).

[15] R. Rivest, "The MD5 message-digest algorithm." RFC 1321, Internet Engineering Task Force (1992).

[16] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, "LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks", Copyright 2003 ACM, pages 62-72

[17] W. Diffie and M.E. Hellman, "Privacy and authentication: An introduction to cryptography", Proceedings of the IEEE 67(3) (1979) 397–427.

[18] Donggang Liu and Peng Ning, "Multilevel μTESLA: Broadcast Authentication for Distributed Sensor Networks" ACM Transactions on Embedded Computing Systems, Vol. 3, No. 4, November 2004, Pages 800–836.

[19] Chris Karlof, Naveen Sastry, David Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks" SenSys'04, November 3–5, 2004, Baltimore, Maryland, USA. Copyright 2004 ACM

[20] T. Ylonen. SSH - secure login connections over the Internet. In Proceedings of the Sixth USENIX Security Symposium, 1996.

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

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

[23] 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 (FOCS 97), 1997.

[24] Biham, E., Biryukov, A., Shamir, A. (1999). "Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials", EUROCRYPT 1999, pp12–23.

[25] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting, "Improved Cryptanalysis of Rijndael, Fast Software Encryption", 2000 pp213–230

[26] Rivest, R. L. (1994). "The RC5 Encryption Algorithm", In the Proceedings of the Second International Workshop on Fast Software Encryption (FSE) 1994, p86–96

[27] Ronald Watro, Derrick Kong, Sue-fen Cuti, Charles Gardiner, Charles Lynn1 and Peter Kruus, "TinyPK: Securing Sensor Networks with Public Key Technology" SASN'04, October 25, 2004, Washington, DC, USA.Copyright 2004 ACM

[28] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs," Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004), Boston, August 2004.

[29] Crossbow Technology, Inc., "Mote In-Network Programming User reference," http://www.xbow.com/Support/Support_pdf_files/Xnp.pdf.

[30] D. Malan, "crypto for Tiny Objects", TR-04-04, Computer Science Group, Harward University, 2004.

[31] A.K. Lenstra and E.R. Verheul, "The XTR public key system", proceedings Crypto 2000, LNCS 1880, Springer-Verlag, 2000.