Towards an Approach for Improved Security in Wireless Networks

T Senthil Kumar
Department of Computer Science Engineering,
Amrita School of Engineering
Ettimadai, Coimbatore

Sugeerth Murugesan
Department of Computer Science Engineering,
Amrita School of Engineering
Ettimadai, Coimbatore

ABSTRACT
Wireless sensor networks are now widely used in military surveillance, industrial applications and civilian uses such as pollution control, forest fire detection, farming etc. Their emergence has posed many unique challenges to researchers. The security of such networks is of utmost importance and as such has been extensively researched. These networks are vulnerable to external threats that may try to gain unauthorized access with malicious intent. In this paper we look at a few security algorithms used on wireless sensor networks. These include SPINS, TinySec, LEAP (Localized Encryption and Authentication Protocol) and PADS (Practical Algorithm for Data Security). Areas that are covered include: architectures and routing protocols, security issues, algorithms, and performance issues for wireless sensor network design

Keywords
Tiny Sec, LEAP, PADS, SPINS, Wireless sensor networks, security algorithms, architectures, SNEP, SPINS.

1. INTRODUCTION
Sensor networks refer to a heterogeneous system combining tiny sensors and actuators with general-purpose computing elements. Wireless sensor networks use tiny, inexpensive sensor nodes characterized by low processing power, radio ranges, low energy consumption and limited and specific sensing functions. Hundreds or thousands of these sensors work together sensing real-world environments, processing this information and communicating it back to the coordinator.

Security has become a major concern for every network. Almost every day we hear about news of some company falling prey to network attacks. Some attacks are passive, meaning information is monitored; others are active, meaning the information is altered with intent to corrupt or destroy the data or the network itself. Without security measures and controls in place, your data might be subjected to an attack. Wireless sensor networks are typically characterized by limited power supplies, low bandwidth, small memory sizes and limited energy. This leads to very demanding environment to provide security.

2. SECURITY PROTOCOLS

2.1 SPINS: Security Protocols for Sensor Networks
SPINS is a suite of security building blocks. It is optimized for resource constrained environments and wireless communication. SPINS has two secure building blocks: SNEP and µTESLA.

SNEP provides data confidentiality, two-party data authentication, and data freshness.

µTESLA provides authenticated broadcast for severely resource-constrained environments. All cryptographic primitives (i.e. encryption, message authentication code (MAC), hash, random number generator) are constructed out of a single block cipher for code reuse [1]. This, along with the symmetric cryptographic primitives used reduces the overhead on the resource constrained sensor network [1]. In a broadcast medium such as a sensor network, data authentication through a symmetric mechanism cannot be applied as all the receivers know the key. µTESLA constructs authenticated broadcast from symmetric primitives, but introduces asymmetry with delayed key disclosure and one-way function key chains [13].

2.1.1 SPINS: Security Protocols for Sensor Networks
SNEP uses encryption to achieve confidentiality and message authentication code (MAC) to achieve two-party authentication and data integrity. Apart from confidentiality, another important security property is semantic security, which ensures that an eavesdropper has no information about the plaintext, even if it sees multiple encryptions of the same plaintext. The basic technique to achieve this is randomization: Before encrypting the message with a chaining encryption function (i.e. DESCBC), the sender precedes the message with a random bit string (also called the Initialization Vector). This prevents the attacker from inferring the plaintext of encrypt ted messages if it knows plaintext-ciphertext pairs encrypted with the same key. To avoid adding the additional transmission overhead of these extra bits, SNEP uses a shared counter between the sender and the receiver for the block cipher in counter mode (CTR). The communicating parties share the counter and increment it after each block. SNEP offers the following properties:

i. Semantic security
ii. Data authentication.
iii. Replay protection
iv. Data freshness.

2.1.2 µTESLA: Authenticated Broadcast
Most of the proposals for authenticated broadcast are impractical for sensor networks, as they rely on asymmetric digital signatures for the authentication. The TESLA protocol provides efficient authenticated broadcast but it is not

designed for limited computing environments. µTESLA solves the following inadequacies of TESLA in sensor networks:

TESLA authenticates the initial packet with a digital signature, which is too expensive for sensor nodes. µTESLA uses only symmetric mechanisms [7].

Disclosing a key in each packet requires too much energy for sending and receiving. µTESLA discloses the key once per epoch [7].

It is expensive to store a one-way key chain in a sensor node. µTESLA restricts the number of authenticated senders [7].µTESLA uses symmetric authentication but introduces asymmetry through a delayed disclosure of the symmetric keys, which results in an efficient broadcast authentication scheme. For the base station to broadcast authenticated information to the nodes, µTESLA requires that the base station and nodes are loosely time synchronized, and each node knows an upper bound on the maximum synchronization error [13]. To send an authenticated packet, the base station simply computes a MAC on the packet with a key that is secret at that point in time [13]. When a node gets a packet, it can verify that the corresponding MAC key was not yet disclosed by the base station (based on its loosely synchronized clock, its maximum synchronization error, and the time schedule at which keys are disclosed) [13]. Since a receiving node is assured that the MAC key is known only by the base station, the receiving node is assured that no adversary could have altered the packet in transit. The node stores the packet in a buffer. At the time of key disclosure, the base station broadcasts the verification key to all receivers. When a node receives the disclosed key, it can easily verify the correctness of the key. If the key is correct, the node can now use it to authenticate the packet stored in its buffer [1].

2.2 Tiny Sec: A Link Layer Security Architecture for Wireless Sensor Networks

TinySec is a lightweight, generic security package that can be integrated into sensor network applications. It is incorporated into the official TinyOS release. Sensor networks use in network processing such as aggregation and duplicate elimination to reduce traffic and save energy. Since in-network processing requires the intermediate nodes to access, modify, and suppress the contents of messages, end-to-end security mechanisms between each sensor node and the base station cannot be used to guarantee the authenticity, integrity, and confidentiality of messages [10]. 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 [10]. This kind of attack will waste energy and bandwidth. A link-layer security architecture can detect unauthorized packets when they are first injected into the network. TinySec provides the basic security properties of message authentication and integrity (using MAC), message confidentiality (through encryption), semantic security (through an Initialization Vector) and replay protection [12]. TinySec supports two different security options: authenticated encryption (TinySec-AE) and authentication only (TinySec-Auth). 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.

2.2.1 Encryption

TinySec uses an 8 byte IV and cipher block chaining (CBC).The structure of the IV is dst||AM||l||src||ctr, where dst is the destination address of the receiver, AM is the active message (AM) handler type, l is the length of the data payload, src is the source address of the sender, and ctr is a 16 bit counter [10]. The counter starts at 0 and the sender increases it by 1 after each message sent [10].

A stream cipher uses a key K and IV as a seed and stretches it into a large pseudorandom keystream GK(IV) [10]. The keystream is then xored against the message: C = (IV, GK(IV) xor P) [10]. The fastest stream ciphers are faster than the fastest block ciphers, which might make them look tempting in a resource constrained environment. However, stream ciphers have a failure mode: if the same IV is ever used to encrypt two different packets, then it is often possible to recover both plaintexts [10]. Guaranteeing that IVs are never reused requires IVs to be fairly long, say, at least 8 bytes. Since an 8-byte overhead in a 30-byte packet is unacceptable in the resource constrained sensor network, TinySec uses block cipher. Using a block cipher for encryption has an additional advantage [10]. Since the most efficient message authentication code (MAC) algorithms use a block cipher, the nodes will need to implement a block cipher in any event. Using this block cipher for encryption as well conserves code space. The advantage of using CBC is that it degrades gracefully in the presence of repeated IVs. If we encrypt two plaintexts P1 and P2 with the same IV under CBC mode, then the cipher texts will leak the length (in blocks) of the longest shared prefix of P1 and P2, and nothing more [12]. For instance, if the first block of P1 is different from the first block of P2, as will typically be the case, then the cryptanalyst learns nothing apart from this fact. CBC mode is provably secure when IVs do not repeat. However, CBC mode was designed to be used with a random IV, and has a separate leakage issue when used with a counter as the IV (note that the TinySec IV has a 16 bit counter). To fix this issue, TinySec pre-encrypts the IV. The creators of TinySec give reasons behind their choice of cipher in . Initially they found AES and Triple-DES to be slow for sensor networks. They found RC5 and Skipjack to be most appropriate for software implementation on embedded microcontrollers. Although RC5 was slightly faster, it is patented. Also, for good performance, RC5 requires the key schedule to be pre computed, which uses 104 extra bytes of RAM per key. Because of these drawbacks, the default block cipher in TinySec is Skipjack [13].

2.2.2 Message integrity

TinySec always authenticates messages, but encryption is optional. TinySec uses a cipher block chaining construction, CBC-MAC 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 we must implement in the limited memory available [10]. However the standard CBC-MAC construction is not secure for variably sized messages. Adversaries can forge a MAC for certain messages. Bellare, Kilian, and Rogaway suggest three alternatives for generating MACs for variable sized messages. The variant used in TinySec xors the encryption of the message length with the first plaintext block [10].

2.2.3 Keying Mechanism

The simplest keying mechanism is to use a single network-wide TinySec key among the authorized nodes. However, this cannot protect against node capture attacks. If an adversary compromises a single node or learns the secret

key, she/he can eavesdrop on traffic and inject messages anywhere in the network. Hence, TinySec uses a separate key for each pair of nodes who might wish to communicate. This provides better resilience against node capture attacks: a compromised node can only decrypt traffic addressed to it and can only inject traffic to its immediate neighbours. But Per-link keying limits the passive participation and local broadcast. A less restrictive approach is for groups of neighbouring nodes to share a TinySec key rather than each pair. Group keying provides an intermediate level of resilience to node capture attacks: a compromised node can decrypt all messages from nodes in its group, but cannot violate the confidentiality of other groups' messages and cannot inject messages to other groups [10].

2.3 LEAP (Localized Encryption and Authentication Protocol)

LEAP is a key management protocol for sensor networks that is designed to support in network processing, while at the same time restricting the security impact of a node compromise to the immediate network neighborhood of the compromised node. The design of the protocol is motivated by the observation that different types of messages exchanged between sensor nodes have different security requirements, and that a single keying mechanism is not suitable for meeting these different security requirements.

Hence, LEAP supports the establishment of four types of keys for each sensor node – an individual key shared with the base station, a pairwise key shared with another sensor node, a cluster key shared with multiple neighboring nodes, and a group key that is shared by all the nodes in the network [8]. The protocol used for establishing and updating these keys is communication and energy efficient, and minimizes the involvement of the base station [8]. LEAP also includes an efficient protocol for inter-node traffic authentication based on the use of one-way key chains. A salient feature of the authentication protocol is that it supports source authentication without precluding in-network processing and passive participation.

2.3.1 Individual Key

Every node has a unique key that it shares pairwise with the base station. This key is used for secure communication between a node and the base station. For example, a node may send an alert to the base station if it observes any abnormal or unexpected behavior by a neighboring node. Similarly, the base station can use this key to encrypt any sensitive information, e.g. keying material or special instruction that it sends to an individual node.

2.3.2 Group Key

This is a globally shared key that is used by the base station for encrypting messages that are broadcast to the whole group [8]. For example, the base station issues missions, sends queries and interests. Note that from the confidentiality point of view there is no advantage to separately encrypting a broadcast message using the individual key of each node [8]. However, since the group key is shared among all the nodes in the network, an efficient rekeying mechanism is necessary for updating this key after a compromised node is revoked.].

2.3.3 Cluster Key

A cluster key is a key shared by a node and all its neighbours, and it is mainly used for securing locally broadcast messages, e.g., routing control information, or securing sensor messages which can benefit from passive participation [8]. For passive participation to be feasible, neighbouring nodes should be able to decrypt and authenticate some classes of messages, e.g., sensor readings, transmitted by their neighbours [8]. This means that such messages should be encrypted or authenticated by a locally shared key. Therefore, in LEAP each node possesses a unique cluster key that it uses for securing its messages, while its immediate neighbours use the same key for decryption or authentication of its messages [12].

2.3.4 Pairwise Shared Key

Every node shares a pairwise key with each of its immediate neighbours. In LEAP, pairwise keys are used for securing communications that require privacy or source authentication [8]. For example, a node can use its pairwise keys to secure the distribution of its cluster key to its neighbours, or to secure the transmissions of its sensor readings to an aggregation node.

2.4 Practical Algorithm for Data Security (PADS)

This algorithm is primarily used for one-time pads (OTP). The message's integrity and authenticity are based on the security of the message authentication code. A 4-byte Message Authentication Code (MAC) is used, meaning an attacker would have to go through 232 attempts, at most, to get a MAC that is a match. The security of the OTP is dependent on the key that is generated. A MAC, also known as a cryptographic checksum, results from the public application of an input via a secret key. Usually of fixed length, it is attached to the input to validate the input's integrity and authenticity. For confidentiality, a new key is generated at each transmission, with the security of the protocol involved dependent upon the Key Derivation Function described by IEEE's Standard specifications for Public-Key Cryptography [1]. These factors reduce the ability of an attacker to create an OTP as a match. The time an attacker would need to create a match will be past the lifetime of a typical sensor network [1]. An example of such a method is SPINS, which is a three-part approach providing for an authentication routing protocol as well as a three-part approach providing authenticated streaming broadcasts as well as two-party data authentication, data confidentiality, and freshness [12].

An algorithm that does basic embedding calculates a MAC using the static part of the packet [12]. The MAC is added to the data and a time synced key is created based on a secret key shared between the sender and the receiver. Any attacker would have to be time synced with the network or he or she would be unable to break the encryption [12]. They also use a basic detection algorithm to locate the embedded pad, remove it, and return it to its original value. The location and removal are done by the base station since it shares with the embedded sensor node the secret key.

TABLE 1. A COMPARATIVE STUDY OF SECURITY ALGORITHMS

Security Algorithm	Mechanism	Merits	Parameter For measurement
1. A Genetic Algorithm Approach for Security Authentication in the wireless Sensor	By applying the multi-objective optimization, we devise a mechanism to guarantee each node's authentication.	The mechanism doesn't base on the key or encryption algorithm. the simulation proves that this security mechanism works well and is valuable in practical.	length of binary bit string when the string length equals 60(L =30),
2. Practical Algorithm for Data Security (PAD)	It calculates a MAC using the static part of the packet. The MAC is added to the data and a time synced key is created based on a secret key shared between the sender and the receiver. Any attacker would have to be time synced with the network or he or she would be unable to break the encryption	It appears to offer better security for a smaller key size, thereby reducing processing overhead.	Three areas are evaluated: latency (the average time a packet takes to reach the base station), throughput in bits per second, and average energy Use per node.
3. Path Redundancy Based Security Algorithm(PRSA)	The algorithm uses alternative routing paths for each data transmission call to overcome the sensor network attack. To enhance network reliability, PRSA allows sensor node data to be sent on defined routing paths using various transmission modes including round robin, redundant and selective modes.	The algorithm improves sensor network resistance and reduce network vulnerability to security threats.	Parameters such as low link cost in sinkhole attack, node power, number of hops to destination, node, and a combination of node ID number and power reflect the presence of adversary nodes.
4. Advanced Encryption Standard (AES)	The standard comprises three block ciphers, AES-128, AES-192 and AES-256, adopted from a larger collection. Each of these ciphers has a 128-bit block size, with key sizes of 128, 192 and 256 bits, respectively	Provides different block and key size during the encryption process. Shows promising power consumption result during the encoding process.	Cost, Power consumption and time

3. CONCLUSION

As architectures play a key purpose in wireless sensor networks so do unique security issues such as how security affects context and design matters as well as working with confidentiality, integrity, and authenticity. Algorithms also have a role in the process of constructing a wireless sensor network. It should be noted, no single security solution is likely to address all security risks. Organizations should implement multiple approaches to completely secure wireless application access. Based on security analysis of all the wireless sensor networks, we conclude that all the above algorithms are excellent solutions that can be recommended for network attacks. Every solution has its own advantages and disadvantages.

4. REFERENCES

[1] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. Journal of Computer Security, 28:270–299, (1984).

[2] Adrian Perrig, Ran Canetti, Dawn Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In Network and Distributed System Security Symposium, NDSS '01, February 2001.

[3] Adrian Perrig, Ran Canetti, J.D. Tygar, and Dawn Song. Efficient authentication and signing of multicast streams over lossy channels. In IEEE Symposium on Security and Privacy, May 2000.

[4] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: A tiny aggregation service for ad-hoc sensor networks. In The Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), 2002.

[5] Samuel R. Madden, Robert Szewczyk, Michael J. Franklin, and David Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In Workshop on Mobile Computing and Systems Applications, 2002.

[6] 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.

[7] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. Journal of Computer and System Sciences, 61(3):362-399, December 2000.

[8] Chris Karlof David Wagner. In Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures.

[9] Wadaa, S. Olariu, L. Wilson, M. Eltoweissy, K. Jones. On Providing Anonymity in Wireless Sensor Networks.

In Proceedings of the Tenth International Conference on Parallel and Distributed Systems (ICPADS'04).

[10] Chris Karlof, Naveen Sastry, David Wagner. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. ACM SenSys 2004, November 3-5, 2004.

[11] Sencun Zhu, Sanjeev Setia, Sushil Jajodia. LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. In The Proceedings of the 10th ACM conference on Computer and communications security, 2003.

[12] Mayank Saraogi, Security in Wireless Sensor Networks.

[13] Daniel E. Burgner, Luay A. Wahsheh, Security of Wireless Sensor Networks, 2011 Eighth International Conference on Information Technology: New Generations.