# SECURITY IN WIRELESS SENSOR NETWORKS

Mayank Saraogi
Department of Computer Science
University of Tennessee, Knoxville
saraogi AT cs.utk.edu

## Abstract

Wireless sensor networks are a new type of networked systems, characterized by severely constrained computational and energy resources, and an ad hoc operational environment. This paper studies the security aspects of these networks. The paper first introduces sensor networks, and then presents its related security problems, threats, risks and characteristics. Additionally, the paper gives a brief introduction to proposed protocols for sensor network security applications such as SPINS [1], TinySec [7] and LEAP [8].

## Introduction

Sensor networks refer to a heterogeneous system combining tiny sensors and actuators with general-purpose computing elements. These networks will consist of hundreds or thousands of self-organizing, low-power, low-cost wireless nodes deployed en masse to monitor and affect the environment. Potential applications include burglar alarms, inventory control, medical monitoring and emergency response [11], monitoring remote or inhospitable habitats [9, 10], target tracking in battlefields [12], disaster relief networks, early fire detection in forests, and environmental monitoring.

Sensor networks are typically characterized by limited power supplies, low bandwidth, small memory sizes and limited energy. This leads to a very demanding environment to provide security. Public-key cryptography is too expensive to be usable, and even fast symmetric-key ciphers must be used sparingly. Communication bandwidth is extremely dear: each bit transmitted consumes about as much power as executing 800–1000 instructions [13], and as a consequence, any message expansion caused by security mechanisms comes at significant cost.

In [5], the authors point out that it seems unlikely that Moore's law will help in the foreseeable future. Because one of the most important factors determining the value of a sensor network comes from how many sensors can be deployed, it seems likely there will be strong pressure to develop ever-cheaper sensor nodes. In other words, we expect that users will want to ride the Moore's law curve down towards ever-cheaper systems at a fixed performance point, rather than holding price constant and improving performance over time.

Thus, the resource-starved nature of sensor networks poses great challenges for security. However, in many applications the security aspects are as important as performance and low energy consumption. Besides the battlefield applications, security is critical in premise security and surveillance, building monitoring, burglar alarms, and in sensors in critical systems such as airports, hospitals.

## Sensor Network Architecture

Sensor networks often have one or more points of centralized control called base stations. A base station is typically a gateway to another network, a powerful data processing or storage center, or an access point for human interface. They can be used as a nexus to disseminate control information into the network or extract data from it. Base stations have also been referred to as sinks. The sensor nodes establish a routing forest, with a base station at the root of every tree. Base stations are many orders of magnitude more powerful than sensor nodes. Typically, base stations have enough battery power to surpass the lifetime of all sensor nodes, sufficient memory to store cryptographic keys, stronger processors, and means for communicating with outside networks.

## Communication Architecture

Generally, the sensor nodes communicate using RF, so broadcast is the fundamental communication primitive. The baseline protocols account for this property: on one hand it affects the trust assumptions, and on the other it is exploited to minimize the energy usage.

In the sensor applications developed so far, the communication patterns within the network fall into the following categories:

- Node to base station communication, e.g. sensor readings, specific alerts.
- Base station to node communication, e.g. specific requests, key updations
- Base station to all nodes, e.g. routing beacons, queries or reprogramming of the entire network
- Communication amongst a defined cluster of nodes (say, a node and all its neighbors). Clustering can reduce the total number of messages sent and thus save energy [14, 15, 16] by using in-network processing techniques such as ***data aggregation*** [24, 25] (an aggregation point can collect sensor readings from surrounding nodes and forward a single message representing an aggregate of the values) and ***passive participation*** (a node that overhears a neighboring sensor node transmitting the same reading as its own current reading can elect to not transmit the same).

## Security Issues and Goals

### 1. Data Confidentiality

Confidentiality means keeping information secret from unauthorized parties. A sensor network should not leak sensor readings to neighboring networks. In many applications (e.g. key distribution) nodes communicate highly sensitive data. The standard approach for keeping sensitive data secret is to encrypt the data with a secret key that only intended receivers possess, hence achieving confidentiality. Since public-key cryptography is too expensive to be used in the resource constrained sensor networks, most of the proposed protocols use symmetric key encryption methods. The creators of TinySec [7] argue that cipher block chaining (CBC) is the most appropriate encryption scheme for sensor networks. They found RC5 and Skipjack to be most appropriate for software implementation on embedded microcontrollers. The default block cipher in TinySec is Skipjack. SPINS uses RC6 as its cipher.

### 2. Data Authenticity

In a sensor network, an adversary can easily inject messages, so the receiver needs to make sure that the data used in any decision-making process originates from the correct source. Data authentication prevents unauthorized

parties from participating in the network and legitimate nodes should be able to detect messages from unauthorized nodes and reject them.

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 been sent by the sender.

However, authentication for broadcast messages requires stronger trust assumptions on the network nodes. The creators of SPINS [1] contend that if one sender wants to send authentic data to mutually untrusted receivers, using a symmetric MAC is insecure since any one of the receivers know the MAC key, and hence could impersonate the sender and forge messages to other receivers. SPINS constructs authenticated broadcast from symmetric primitives, but introduces asymmetry with delayed key disclosure and one-way function key chains. LEAP [8] uses a globally shared symmetric key for broadcast messages to the whole group. However, since the group key is shared among all the nodes in the network, an efficient rekeying mechanism is defined for updating this key after a compromised node is revoked. This means that LEAP has also defined an efficient mechanism to verify whether a node has been compromised.

## 3. Data Integrity

Data integrity ensures the receiver that the received data is not altered in transit by an adversary. Note that Data Authentication can provide Data Integrity also.

## 4. Data Freshness

Data freshness implies that the data is recent, and it ensures that an adversary has not replayed old messages. A common defense (used by SNEP [1]) is to include a monotonically increasing counter with every message and reject messages with old counter values. With this policy, every recipient must maintain a table of the last value from every sender it receives. However, for RAM-constrained sensor nodes, this defense becomes problematic for even modestly sized networks. Assuming nodes devote only a small fraction of their RAM for this neighbor table, an adversary replaying broadcast messages from many different senders can fill up the table. At this point, the recipient has one of two options: ignore any messages from senders not in its neighbor table, or purge entries from the table. Neither is acceptable; the first creates a DoS attack and the second permits replay attacks.

In [5], the authors contend that protection against the replay of data packets should be provided at the application layer and not by a secure routing protocol as only the application can fully and accurately detect the replay of data packets (as opposed to retransmissions, for example). In [7], the authors reason that by using information about the network's topology and communication patterns, the application and routing layers can properly and efficiently manage a limited amount of memory devoted to replay detection.

In [1], the authors have identified 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. Weak freshness is required by sensor measurements, while strong freshness is useful for time synchronization within the network.

### 5. Robustness and Survivability

The sensor network should be robust against various security attacks, and if an attack succeeds, its impact should be minimized. The compromise of a single node should not break the security of the entire network.

## Security Threats, Types of Attacks on Sensor Networks and Countermeasures

Wireless networks are vulnerable to security attacks due to the broadcast nature of the transmission medium. Furthermore, wireless sensor networks have an additional vulnerability because nodes are often placed in a hostile or dangerous environment where they are not physically protected.

### 1. Passive Information Gathering

An intruder with an appropriately powerful receiver and well designed antenna can easily pick off the data stream. Interception of the messages containing the physical locations of sensor nodes allows an attacker to locate the nodes and destroy them. Besides the locations of sensor nodes, an adversary can observe the application specific content of messages including message IDs, timestamps and other fields. To minimize the threats of passive information gathering, strong encryption techniques needs to be used.

### 2. Subversion of a Node

A particular sensor might be captured, and information stored on it (such as the key) might be obtained by an adversary. If a node has been compromised then how to exclude that node, and that node only, from the sensor network is at issue (LEAP [8] defines an efficient way to do so).

### 3. False Node and malicious data

An intruder might add a node to the system that feeds false data or prevents the passage of true data. Such messages also consume the scarce energy resources of the nodes. This type of attack is called "*sleep deprivation torture*" in [17]. Insertion of malicious code is one of the most dangerous attacks that can occur. Malicious code injected in the network could spread to all nodes, potentially destroying the whole network, or even worse, taking over the network on behalf of an adversary. A seized sensor network can either send false observations about the environment to a legitimate user or send observations about the monitored area to a malicious user. By spoofing, altering, or replaying routing information, adversaries may be able to create routing loops, attract or repel network traffic, extend or shorten source routes, generate false error messages, partition the network, increase end-to-end latency, etc.

Strong authentication techniques can prevent an adversary from impersonating as a valid node in the sensor network.

### 4. The Sybil attack

In a Sybil attack [18], a single node presents multiple identities to other nodes in the network. They pose a significant threat to geographic routing protocols, where location aware routing requires nodes to exchange coordinate information with their neighbors to efficiently route geographically addressed packets.

Authentication and encryption techniques can prevent an outsider to launch a Sybil attack on the sensor network. However, an insider cannot be prevented from participating in the network, but (s)he should only be able to do so using the identities of the nodes (s)he has compromised. Using globally shared keys

allows an insider to masquerade as any (possibly even nonexistent) node. Public key cryptography can prevent such an insider attack, but it is too expensive to be used in the resource constrained sensor networks. One solution is to have every node share a unique symmetric key with a trusted base station. Two nodes can then use a Needham-Schroeder like protocol to verify each other's identity and establish a shared key. A pair of neighboring nodes can use the resulting key to implement an authenticated, encrypted link between them. An example of a protocol which uses such a scheme is LEAP [8], which supports the establishment of four types of keys.

## 5. Sinkhole attacks

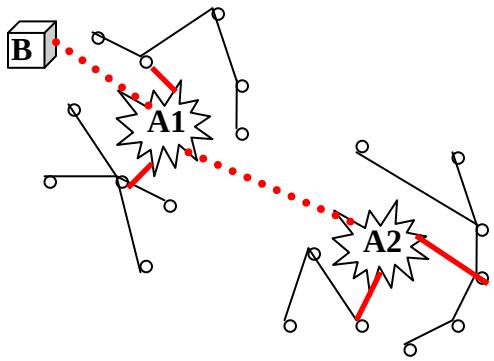
In a sinkhole attack, the adversary's goal is to lure nearly all the traffic from a particular area through a compromised node, creating a metaphorical sinkhole with the adversary at the center. Sinkhole attacks typically work by making a compromised node look especially attractive to surrounding nodes with respect to the routing algorithm. For instance, an adversary could spoof or replay an advertisement for an extremely high quality route to a base station. Due to either the real or imagined high quality route through the compromised node, it is likely each neighboring node of the adversary will forward packets destined for a base station through the adversary, and also propagate the attractiveness of the route to its neighbors. Effectively, the adversary creates a large "*sphere of influence*" [5], attracting all traffic destined for a base station from nodes several hops away from the compromised node.

## 6. Wormholes

In the wormhole attack [3], an adversary tunnels messages received in one part of the network over a low latency link and replays them in a different part. The simplest instance of this attack is a single node situated between two other nodes forwarding messages between the two of them. However, wormhole attacks more commonly involve two distant malicious nodes colluding to understate their distance from each other by relaying packets along an out-of-bound channel available only to the attacker.

An adversary situated close to a base station may be able to completely disrupt routing by creating a well-placed wormhole. An adversary could convince nodes who would normally be multiple hops from a base station that they are only one or two hops away via the wormhole. This can create a sinkhole: since the adversary on the other side of the wormhole can artificially provide a high-quality route to the base station, potentially all traffic in the surrounding area will be drawn through her if alternate routes are significantly less attractive.

The following diagram shows an example of a wormhole being used to create a sinkhole:




Adversaries A1 and A2 combine to form a sinkhole-wormhole attack. The nodes near A2 believe that the Base Station B is closer via the sinkhole A1. Hence, the wormhole convinces two distant nodes that they are neighbors by relaying packets between the two of them.

A technique for detecting wormhole attacks is presented in [20], but it requires extremely tight time synchronization and is thus infeasible for most sensor networks.

## SPINS: Security Protocols for Sensor Networks [1]

SPINS a suite of security building blocks proposed by Perig et all. 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. This, along with the symmetric cryptographic primitives used reduces the overhead on the resource constrained sensor network.

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.

### SNEP: Confidentiality, Authentication, Integrity, and Freshness

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 [21]. The basic technique to achieve this is randomization: Before encrypting the message with a chaining encryption function (i.e. DES-CBC), the sender precedes the message with a random bit string (also called the *Initialization Vector*). This prevents the attacker from inferring the plaintext of encrypted 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:

*Semantic security*: Since the counter value is incremented after each message, the same message is encrypted differently each time. The counter value is long enough that it never repeats within the lifetime of the node.

*Data authentication*: If the MAC verifies correctly, a receiver can be assured that the message originated from the claimed sender.

*Replay protection*: The counter value in the MAC prevents replaying old messages. Note that if the counter were not present in the MAC, an adversary could easily replay messages.

*Data freshness*: If the message verified correctly, a receiver knows that the message must have been sent after the previous message it received correctly (that had a lower counter value). This enforces a message ordering and yields weak freshness.

*Low communication overhead*: The counter state is kept at each end point and does not need to be sent in each message.

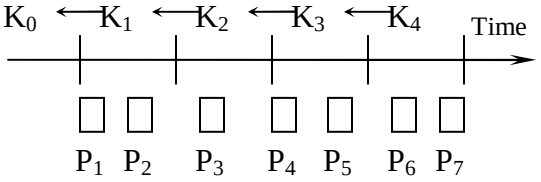
**µTESLA: Authenticated Broadcast**

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 [22, 23] 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.
- Disclosing a key in each packet requires too much energy for sending and receiving. µTESLA discloses the key once per epoch.
- It is expensive to store a one-way key chain in a sensor node. µTESLA restricts the number of authenticated senders.

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. 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. 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). 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 (which we explain below). If the key is correct, the node can now use it to authenticate the packet stored in its buffer.

Each MAC key is a key of a key chain, generated by a public one-way function F. To generate the one-way key chain, the sender chooses the last key $K_n$ of the chain randomly, and repeatedly applies F to compute all other keys: $K_i = F(K_i+1)$. 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.




For example, let the key be disclosed in 2 time intervals. Each key of the key chain corresponds to a time interval and all packets sent within one time interval are authenticated with the same key. The receiver node is loosely time synchronized and knows $K_0$ (a commitment to the key chain) in an authenticated way. Packets $P_1$ and $P_2$ sent in interval 1 contain a MAC with key $K_1$. Packet $P_3$ has a MAC using key $K_2$. So far, the receiver cannot authenticate any packets yet. Let us assume that packets $P_4$, $P_5$, and $P_6$ are all lost, as well as the packet that discloses key $K_1$, so the receiver can still not authenticate $P_1$, $P_2$, or $P_3$. In interval 4 the base station broadcasts key $K_2$, which the node authenticates by verifying $K_0 = F(F(K_2))$, and hence knows also $K_1 = F(K_2)$, so it can

authenticate packets $P_1$, $P_2$ with $K_1$, and $P_3$ with $K_2$.

Instead of adding a disclosed key to each data packet, the key disclosure is independent from the packets broadcast, and is tied to time intervals. Within the context of µTESLA, the sender broadcasts the current key periodically in a special packet.

For further details about the implementation and evaluation results of SNEP, refer to [1].

## TinySec: A Link Layer Security Architecture for Wireless Sensor Networks [7]

TinySec is a lightweight, generic security package that can be integrated into sensor network applications. It is incorporated into the official TinyOS release. In [7], the authors reason why Link Layer security is ideal for sensor networks. Sensor networks use in-network processing such as aggregation and duplicate elimination [24, 25] 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. 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 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.

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.

### Encryption

TinySec uses an 8 byte IV and cipher block chaining (CBC) [26].

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. The counter starts at 0 and the sender increases it by 1 after each message sent.

A stream cipher uses a key K and IV as a seed and stretches it into a large pseudorandom keystream $G_K(IV)$. The keystream is then xored against the message: C = (IV, $G_K(IV)$ *xor* P). 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. 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. 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 ciphertexts will leak the length (in blocks) of the longest shared prefix of P1 and P2, and nothing more. 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 [7]. 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 precomputed, which uses 104 extra bytes of RAM per key. Because of these drawbacks, the default block cipher in TinySec is Skipjack.

**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. 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 [28]. The variant used in TinySec xors the encryption of the message length with the first plaintext block.

**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, (s)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 neighbors. But Per-link keying limits passive participation and local broadcast. A less restrictive approach is for groups of neighboring 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.

For further information about the implementation and performance results of TinySec, refer to [7].

## LEAP (Localized Encryption and Authentication Protocol) [8]

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. The protocol used for establishing and updating these keys is communication and energy efficient, and minimizes the involvement of the base station. 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.

### 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.

### 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. 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. 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.

### Cluster Key

A cluster key is a key shared by a node and all its neighbors, 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. For passive participation to be feasible, neighboring nodes should be able to decrypt and authenticate some classes of messages, e.g., sensor readings, transmitted by their neighbors. 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 neighbors use the same key for decryption or authentication of its messages.

### Pairwise Shared Key

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

Note that the use of pairwise keys precludes passive participation.

In [8], the creators of LEAP have described the schemes provided by LEAP for sensor nodes to establish and update individual keys, pairwise shared keys, cluster keys, and group keys for each node. Revocation of a compromised node and the subsequent rekeying mechanism is also described.

## Conclusion

In this paper, we introduce sensor networks, its related security problems, threats, risks and characteristics, and a brief introduction to SPINS, TinySec and LEAP. For implementation details and performance evaluation of these protocols, please refer to the [1], [7] and [8]. Adding security in a resource constrained wireless sensor network with minimum overhead provides significant challenges, and is an ongoing area of research.

## References


[1] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J. D. Tygar. SPINS: Security Protocols for Sensor Networks. *In The Seventh Annual International Conference on Mobile Computing and Networking (MobiCom 2001), 2001*.

[2] Sasha Slijepcevic, Miodrag Potkonjak, Vlasios Tsiatsis, Scott Zimbeck, Mani B. Srivastava. On Communication Security in Wireless Ad-Hoc Sensor Networks. *In The Proceedings of the Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'02), 2002*.

[3] Y.C. Hu, A. Perrig, and D. B. Johnson, "Wormhole detection in wireless ad hoc networks," *Department of Computer Science, Rice University, Tech. Rep. TR01-384, June 2002*.

[4] Jeffery Undercoffer, Sasikanth Avancha, Anupam Joshi and John Pinkston. In Security for Sensor Networks.

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

[6] 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)*.

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

[8] 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*.

[9] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, and David Culler. Wireless sensor networks for habitat monitoring. *In First ACM International Workshop on Wireless Sensor Networks and Applications, 2002*.

[10] Robert Szewczyk, Joseph Polastre, Alan Mainwaring, and David Culler. Lessons from a sensor network expedition. *In First European Workshop on Wireless Sensor Networks (EWSN '04), January 2004*.

[11] Matt Welsh, Dan Myung, Mark Gaynor, and Steve Moulton. Resuscitation monitoring with a wireless sensor network. In *Supplement to Circulation: Journal of the American Heart Association, October 2003*.

[12] G.L. Duckworth, D.C. Gilbert, and J.E. Barger. Acoustic counter-sniper system. *In SPIE International*

*Symposium on Enabling Technologies for Law Enforcement and Security, 1996*.

[13] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler and K. Pister. System architecture directions for networked sensors. *In Proceedings of ACM ASPLOS IX, November 2000*.

[14] C.Intanagonwiwat, R.Govindan and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks *In Proc. of MobiCOM'00, Boston, Massachussetts, August 2000*.

[15] C. Karlof, Y. Li, and J. Polastre. ARRIVE: An Architecture for Robust Routing *In Volatile Environments. Technical Report UCB/CSD-03-1233, University of California at Berkeley, Mar.2003*.

[16] S. Madden, R. Szewczyk, M. Franklin, and D. Culler. Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks. *In 4th IEEE Workshop on Mobile Computing Systems and Applications, June 2002*.

[17] F. Stajano, R. Anderson. "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks", *3rd AT&T Software Symposium, Middletown, NJ, October 1999*.

[18] J. R. Douceur, "The Sybil Attack," *in 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), March 2002*.

[19] R. L. Rivest, M.J.B. Robshaw, R. Sidney, and Y.L. Yin, "The RC6 Block Cipher", AES submission, Jun 1998. http://theory.lcs.mit.edu/~rivest/rc6.pdf

[20] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole detection in wireless ad hoc networks," *Department of Computer Science, Rice University, Tech. Rep. TR01-384, June 2002*.

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

[22] 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*.

[23] 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*.

[24] 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*.

[25] 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*.

[26] 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*.

[27] Bruce Schneier. Applied Cryptography, *Second Edition. John Wiley & Sons, 1996*.

[28] 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*.