An Engineering Approach for Secure Wireless Sensor Networks

Tushar Agarwal[1], Dr. Raj Kumar[2], Shashiraj Teotia[3], Dr. Sohan Garg[4]

1. Department of MCA, IIMT Meerut U.P. (INDIA)
2. Department of MCA, GKV Haridwar (UK) (INDIA)
3. Department of CS, SV Subharati University, Meerut U.P. (INDIA)
4. Department of MCA, RKGIT, Ghaziabad U.P. (India)

ABSTRACT

As wireless sensor networks continue to grow, so does the need for effective security mechanisms. Because sensor networks may interact with sensitive data and/or operate in hostile unattended environments, it is imperative that these security concerns be addressed from the beginning of the system design. Their low cost provides a means to deploy large sensor arrays in a variety of conditions capable of performing both military and civilian tasks. Wireless sensor networks are a new type of networked systems, characterized by severely constrained computational and energy resources, and an ad hoc operational environment.

The paper first introduces sensor networks, and then presents its related security problems, threats, risks and characteristics. Sensor networks refer to a heterogeneous system combining tiny sensors and actuators with general-purpose computing elements. Due to limited computation, power, and storage resources available on sensor nodes asymmetric cryptographic algorithms cannot provide security on wireless sensor networks. An adversary may launch a wide range of attacks including eavesdropping, message forgery, packet dropping, and noise injection. The unreliable communication channel and unattended operation make the security defences even harder. Additionally, the paper gives a brief introduction to proposed protocols for sensor network security applications.

Keywords: Wireless sensor networks, software engineering, security engineering

1. INTRODUCTION

Wireless sensor networks are quickly gaining popularity due to the fact that they are potentially low cost solutions to a variety of real-world challenges [1, 2]. Their low cost provides a means to deploy large sensor arrays in a variety of conditions capable of performing both military and civilian tasks. But sensor networks also introduce severe resource constraints due to their lack of data storage and power. Both of these represent major obstacles to the implementation of traditional computer security techniques in a wireless sensor network. The unreliable communication channel and unattended operation make the security defences even harder. Indeed, as pointed out in [3], wireless sensors often have the processing characteristics of machines that are decades old (or longer), and the industrial trend is to reduce the cost of wireless sensors while maintaining similar computing power. With that in mind, many researchers have begun to address the challenges of maximizing the processing capabilities and energy reserves of wireless sensor nodes while also securing them against attackers. All aspects of the wireless sensor network are being examined including secure and efficient routing [4, 7], data aggregation [8, 13], group formation [14, 16], and so on. In addition to those traditional security issues, we observe that many general-purpose sensor network techniques (particularly the early research) assumed that all nodes are cooperative and trustworthy. This is not the case for most, or much of, real-world wireless sensor networking applications, which require a certain amount of trust in the application in order to maintain proper network functionality. Researchers therefore began focusing on building a sensor trust model to solve the problems beyond the capability of cryptographic security [17, 24]. In addition, there are many attacks designed to exploit the unreliable communication channels and unattended operation of wireless sensor networks. Furthermore, due to the inherent unattended feature of wireless sensor networks, we argue that physical attacks to sensors play an important role in the operation of wireless sensor networks. Thus, we include a detailed discussion of the physical attacks and their corresponding defences [25, 34], topics typically ignored in most of the current research on sensor security. We classify the main aspects of wireless sensor network security into four major categories: characteristics, related security problems and threats & risks. The unreliable communication channel and unattended operation make the security defences even harder. Exploiting such weaknesses an adversary may launch a wide range of attacks including eavesdropping, message forgery, packet dropping, and noise injection. The paper gives a brief introduction to

some previously proposed protocols for sensor network security applications that somehow attempt to cover and control some of these to an extent.

2. CHARACTERISTICS

A wireless sensor network is a characteristic network which has many constraints compared to a traditional computer network. Due to these constraints it is difficult to directly employ the existing security approaches to the area of wireless sensor networks. Therefore, to develop useful security mechanisms while borrowing the ideas from the current security techniques, it is necessary to know and understand these constraints first [35].

2.1 Very Limited Resources

All security approaches require a certain amount of resources for the implementation, including data memory, code space, and energy to power the sensor. However, currently these resources are very limited in a tiny wireless sensor.

• Limited Memory and Storage Space A sensor is a tiny device with only a small amount of memory and storage space for the code. In order to build an effective security mechanism, it is necessary to limit the code size of the security algorithm. For example, one common sensor type (TelosB) has an 16-bit, 8 MHz RISC CPU with only 10K RAM, 48K program memory, and 1024K flash storage [36]. With such a limitation, the software built for the sensor must also be quite small. The total code space of TinyOS, the de-facto standard operating system for wireless sensors, is approximately 4K [37], and the core scheduler occupies only 178 bytes. Therefore, the code size for the all security related code must also be small.

• Power Limitation Energy is the biggest constraint to wireless sensor capabilities. We assume that once sensor nodes are deployed in a sensor network, they cannot be easily replaced (high operating cost) or recharged (high cost of sensors). Therefore, the battery charge taken with them to the field must be conserved to extend the life of the individual sensor node and the entire sensor network. When implementing a cryptographic function or protocol within a sensor node, the energy impact of the added security code must be considered. When adding security to a sensor node, we are interested in the impact that security has on the lifespan of a sensor (i.e., its battery life). The extra power consumed by sensor nodes due to security is related to the processing required for security functions (e.g., encryption, decryption, signing data, verifying signatures), the energy required to transmit the security related data or overhead (e.g., initialization vectors needed for encryption/decryption), and the energy required to store security parameters in a secure manner (e.g., cryptographic key storage).

2.2 Unreliable Communication

Certainly, unreliable communication is another threat to sensor security. The security of the network relies heavily on a defined protocol, which in turn depends on communication.

• Unreliable Transfer Normally the packet-based routing of the sensor network is connectionless and thus inherently unreliable. Packets may get damaged due to channel errors or dropped at highly congested nodes. The result is lost or missing packets. Furthermore, the unreliable wireless communication channel also results in damaged packets. Higher channel error rate also forces the software developer to devote resources to error handling. More importantly, if the protocol lacks the appropriate error handling it is possible to lose critical security packets. This may include, for example, a cryptographic key.

• Conflicts Even if the channel is reliable, the communication may still be unreliable. This is due to the broadcast nature of the wireless sensor network. If packets meet in the middle of transfer, conflicts will occur and the transfer itself will fail. In a crowded (high density) sensor network, this can be a major problem. More details about the effect of wireless communication can be found at [2].

• Latency The multi-hop routing, network congestion, and node processing can lead to greater latency in the network, thus making it difficult to achieve synchronization among sensor nodes. The synchronization issues can be critical to sensor security where the security mechanism relies on critical event reports and cryptographic key distribution. Interested readers please refer to [38] on real-time communications
in wireless sensor networks.

2.3 Unattended Operation

Depending on the function of the particular sensor network, the sensor nodes may be left unattended for long periods of time. There are three main caveats to unattended sensor nodes:

• Exposure to Physical Attacks The sensor may be deployed in an environment open to adversaries, bad weather, and so on. The likelihood that a sensor suffers a physical attack in such an environment is therefore much higher than the typical PCs, which is

located in a secure place and mainly faces attacks from a network.

• Managed Remotely Remote management of a sensor network makes it virtually impossible to detect physical tampering (i.e., through tamperproof seals) and physical maintenance issues (e.g., battery replacement). Perhaps the most extreme example of this is a sensor node used for remote reconnaissance missions behind enemy lines. In such a case, the node may not have any physical contact with friendly forces once deployed.

• No Central Management Point A sensor network should be a distributed network without a central management point. This will increase the vitality of the sensor network. However, if designed incorrectly, it will make the network organization difficult, inefficient, and fragile. Perhaps most importantly, the longer that a sensor is left unattended the more likely that an adversary has compromised the node.

3. SECURITY & CHALLENGES

In this section, we formalize the security properties [39] required by sensor networks, and show how they are directly applicable in a typical sensor network.

3.1. Data Confidentiality

Confidentiality means keeping information secret from unauthorized parties. A sensor network should not leak sensor readings to neighbouring 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 [40] 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.

3.2. 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 [40] 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 [12] 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.

3.3. Data Integrity

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

3.4. 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 [22]) 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 neighbour 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 [21], 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 [40], 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 [31], 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.

3.5. 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.

4. THREATS AND RISKS

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.

4.1. 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.

4.2. 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 [36] defines an efficient way to do so).

4.3. 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 [34]. 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.

4.4. The Sybil attack

In a Sybil attack [35], 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 neighbours 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 neighbouring 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 [23], which supports the establishment of four types of keys.

4.5. 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 neighbouring 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 neighbours. Effectively, the adversary creates a large "*sphere of influence*" [23], attracting all traffic destined for a base station from nodes several hops away from the compromised node.

4.6. Wormholes

In the wormhole attack [16], 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.

5. EXISTING SECURITY PROTOCOLS

In previous sections we have out listed the characteristics that make sensor networks highly vulnerable and also the security challenges. This section gives an insight in to the existing security protocols in the area.

5.1. SPINS Security Building Blocks[37]

To achieve the security requirements we established in Section 4 we have designed and implemented two security building blocks: SNEP and _TESLA. SNEP provides data confidentiality, two-party data authentication, integrity, and freshness. _TESLA provides authentication for data broadcast. We bootstrap the security for both mechanisms with a shared secret key between each node and the base station (see Section 2). We demonstrate in Section 8 how we can extend the trust to node-to-node interactions from the node-to base-station trust.

5.1.1. SNEP: Data Confidentiality, Authentication, Integrity, and Freshness

SNEP provides a number of unique advantages. First, it has low communication overhead since it only adds 8 bytes per message. Second, like many cryptographic protocols it uses a counter, but we avoid transmitting the counter value by keeping state at both end points. Third, SNEP achieves even semantic security, a strong security property which prevents eavesdroppers from inferring the message content from the encrypted message. Finally, the same simple and efficient protocol also gives us data authentication, replay protection, and weak message freshness.

SNEP offers the following nice 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 theMAC 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.
- *Weak 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.

5.1.2. TESLA: Authenticated Broadcast

Current proposals for authenticated broadcast are impractical for sensor networks. First, most proposals rely on asymmetric digital signatures for the authentication, which are impractical for multiple reasons. They require long signatures with high communication overhead of 50-1000 bytes per packet, very high overhead to create and verify the signature. Even previously proposed one-time signature schemes that are based on symmetric cryptography (one-way functions without trapdoors) have a high overhead. The recently proposed TESLA protocol provides efficient authenticated broadcast [30, 34]. However, TESLA is not designed for such limited computing environments as we encounter in sensor networks for three reasons. First, TESLA authenticates the initial

packet with a digital signature. Clearly, digital signatures are too expensive to compute on our sensor nodes, since even fitting the code into the memory is a major challenge. For the same reason as we mention above, onetime signatures are a challenge to use on our nodes. Standard TESLA has an overhead of approximately 24 bytes per packet. For networks connecting workstations this is usually not significant. Sensor nodes, however, send very small messages that are around 30 bytes long. It is simply impractical to disclose the TESLA key for the previous intervals with every packet: with bit keys and MACs, the TESLA-related part of the packet would constitute over 50% of the packet. Finally, the one-way key chain does not fit into the memory of our sensor node. So, pure TESLA is not practical for a node to broadcast authenticated data.

5.2. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks [40]

TinySec is a lightweight, generic security package that can be integrated into sensor network applications. It is incorporated into the official TinyOS release. In [40], the authors reason why Link Layer security is ideal for sensor networks. Sensor networks use in network processing such as aggregation and duplicate elimination [38-39] 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. 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.

5.2.1. Encryption

TinySec uses an 8 byte IV and cipher block chaining (CBC) [32]. 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 key stream GK(IV). The key stream is then xored against the message: C = (IV, GK(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 cipher texts 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 [40]. 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.

5.2.2. 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. 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 [33]. The variant used in TinySec XORs the encryption of the message length with the first plaintext block.

5.2.3. 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, (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 neighbours. But Per-link keying limits 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. For further information about the implementation and performance results of TinySec, refer to [40].

5.3. LEAP (Localized Encryption and Authentication Protocol) [42]

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 neighbourhood 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 neighbouring 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.

5.3.1. 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 behaviour by a neighbouring 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.

5.3.2. 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. 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.

5.3.3. 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. 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. 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.

5.3.4. 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 [40], 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.

6.CONCLUSION

The purpose of this research paper is to survey the current state of art on this topic identify issues for further study. An attempt is made to cover general security issues of WSN research and present achievement in security engineering with regards to earlier survey work along with the security capabilities of major implementation platforms to highlight essential available and required mechanism for the software engineers.

REFERENCES

[1] Yang Xiao,(Eds.) Wireless Sensor Network Security: A Survey. Security in Distributed, Grid, and Pervasive Computing, Auerbach Publications, CRC Press, 2006

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, August 2002.

[3] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. Spins: security protocols for sensor networks. *Wireless Networking*, 8(5):521–534, 2002.

[4] J. Deng, R. Han, and S. Mishra. INSENS: intrusion-tolerant routing in wireless sensor networks. In *Technical Report CU-CS-939-02, Department of Computer Science, University of Colorado*, 2002.

[5] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254. ACM Press, 2000.

[6] P. Papadimitratos and Z. J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Communication Networks and Distributed System Modeling and Simulation Conference (CNDS 2002)*, 2002.

[7] S. Tanachaiwiwat, P. Dave, R. Bhindwale, and A. Helmy. Poster abstract secure locations: routing on trust and isolating compromised sensors in location aware sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 324–325. ACM Press, 2003.

[8] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Mobile Computing and Networking*, pages 263–270, 1999.

[9] L. Hu and D. Evans. Secure aggregation for wireless networks. In *SAINTW '03: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, page 384. IEEE Computer Society, 2003.

[10] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

[11] B. Przydatek, D. Song, and A. Perrig. Sia: Secure information aggregation in sensor networks, 2003.

[12] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 239–249. ACM Press, 2004.

[13] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route detection and filtering of injected fase data in sensor networks. In *IEEE INFOCOM 2004*, 2004.

[14] A. R. Beresford and F. Stajano. Location Privacy in Pervasive Computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.

[15] T. Kaya, G. Lin, G. Noubir, and A. Yilmaz. Secure multicast groups on ad hoc networks. In *Proceedings of the 1st ACM workshop on Security of Ad hoc and Sensor Networks (SASN '03)*, pages 94–102. ACM Press, 2003.

[16] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. *ACM Comput. Surv.*, 35(3):309–329, 2003.

[17] S. Ganeriwal and M. Srivastava. Reputation-based framework for high integrity sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, Washington DC, USA, 2004.

[18] Z. Liang and W. Shi. Enforcing cooperative resource sharing in untrusted peer-to-peer

environment. *ACM Journal of Mobile Networks and Applications (MONET)*, 10(6):771–783, 2005.

[19] Z. Liang and W. Shi. Analysis of recommendations on trust inference in the open environment. Technical Report MIST-TR-2005-002, Department of Computer Science, Wayne State University, February 2005.

[20] Z. Liang and W. Shi. PET: A PErsonalized Trust model with reputation and risk evaluation for P2P resource sharing. In *Proceedings of the HICSS-38*, Hilton Waikoloa Village Big Island, Hawaii, January 2005.

[21] K. Ren, T. Li, Z. Wan, F. Bao, R. H. Deng, and K. Kim. Highly reliable trust establishment scheme in ad hoc networks. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 45:687–699, August 2004.

[22] S. Tanachaiwiwat, P. Dave, R. Bhindwale, and A. Helmy. Location-centric isolation of misbehavior and trust routing in energy-constrained sensor networks, April 2004.

[23] Z. Yan, P. Zhang, and T. Virtanen. Trust evaluation based security solution in ad hoc networks. In *NordSec 2003, Proceedings of the Seventh Nordic Workshop on Secure IT Systems*, 2003.

[24] H. Zhu, F. Bao, R. H. Deng, and K. Kim. Computing of trust in wireless networks. In *Proceedings of 60th IEEE Vehicular Technology Conference*, Los Angles, California, September 2004.

[25] R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, Oakland, California, 1996.

[26] R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In *IWSP: International Workshop on Security Protocols, LNCS*, 1997.

[27] C. Hartung, J. Balasalle, and R. Han. Node compromise in sensor networks: The need for secure systems. Technical Report Technical Report CU-CS-988-04, Department of Computer Science, University of Colorado at Boulder, 2004.

[28] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *In 11th Annual Network and Distributed System Security Symposium*, February 2004.

[29] O. K¨omerling and M. G. Kuhn. Design principles for tamper-resistant smartcard processors. In *appeared in the USENIX Workshop on Smartcard Technology proceedings*, Chicago, Illinois, USA, May 1999.

[30] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *ACM Workshop on Wireless Security*, September 2003.

[31] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla. Swatt: Software-based attestation for embedded devices. In *In Proceedings of the IEEE Symposium on Security and Privacy*, May 2004.

[32] X. Wang, W. Gu, S. Chellappan, K.t Schoseck, and Dong Xuan. Lifetime optimization of sensor networks under physical attacks. In *Proc. of IEEE Internationl Conference on Communications*, May 2005.

[33] X. Wang, W. Gu, S. Chellappan, Dong Xuan, and Ten H. Laii. Search-based physical attacks in sensor networks: Modeling and defense. Technical report, Dept. of Computer Science and Engineering, The Ohio-State University, February 2005.

[34] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *Computer*, 35(10):54–62, 2002.

[35] D. W. Carman, P. S. Krus, and B. J. Matt. Constraints and approaches for distributed sensor network security. Technical Report 00-010, NAI Labs, Network Associates, Inc., Glenwood, MD, 2000.

[36] http://www.xbow.com/wireless home.aspx, 2006.

[37] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[38] J. A. Stankovic et al. Real-time communication and coordination in embedded sensor networks. *Proceedings of the IEEE*, 91(7):1002–1022, July 2003.

[39] Mayank Saraogi, Security in Wireless Sensor Networks; Department of Computer Science, University of Tennessee, Knoxville

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