Chapter 17

Information Security in Wireless Sensor Networks

Abdelraouf Ouadjaout and Miloud Bagaa
Houari Boumediene University of Science and Technology
Informatics Systems Lab. Algiers, Algeria

Abdelmalik Bachir
Grenoble Institute of Technology Informatics
Lab. Grenoble, France

Yacine Challal
Université de Technologies de Compiègne, UMR CNRS 6599
Heudiasyc, Compiègne, France

Nouredine Lasla
National Institute of Informatics, Algiers, Algeria

Lyes Khelladi
Scientific and Technical Information Research Center
Basic Software Lab. Algiers, Algeria

In this chapter, we provide a comprehensive survey of security issues in wireless sensor networks. We show that the main features of WSNs, namely their limited resources, wireless communications, and close physical coupling with environment, are the main causes of the their security vulnerabilities. We discuss the main attacks stemming from these vulnerabilities, along with the solutions proposed in the literature to cope with them. The security solutions are analyzed with respect to the different layers of the network protocol stack and cover the following issues: key management, secure data dissemination, secure data aggregation, secure channel access and secure node compromise.

Keywords: Key management; data dissemination security; data aggregation security; denial of service countermeasures; energy exhaustion attacks.

17.1. Introduction

The main security issues in WSN stem from the properties that make them efficient and attractive, which are:

(1) Resource Limitations: Energy is perhaps the greatest constraint to sensor node capabilities. The energy reserve of each node should be conserved to extend its lifetime and thus that of the entire network. Most of time, the sensed information is redundant due to geographically collocated sensors. Most of this energy can be saved through data aggregation. This requires a particular attention to detect faulty injected or modified data. In addition, energy scarcity opens new opportunities for DoS (Denial of Service) attacks aiming to exhaust nodes' batteries.
(2) Wireless and Multi-hop Communication: In addition to providing easy deployment, wireless communication has the advantage of offering access to hardly accessible locations such as disastrous and hostile terrains. Unfortunately, the radio communication range of sensor nodes is limited due to energy considerations. Multi-hop communication is thus indispensable for data dissemination in WSN. This introduces many security vulnerabilities at two different levels: attacking route construction and maintenance, and attacking data payload through packet injection, modification or suppression. Moreover, the wireless communication introduces other vulnerabilities at the link layer opening the door to jamming-style DoS and energy exhaustion attacks.
(3) Close physical coupling with environment: Most of WSN applications require a close deployment of nodes inside or near the phenomena to be monitored. This close physical coupling with environment leads to frequent intentional or accidental compromising of sensor nodes. As the success of WSN applications also depends on their low cost, nodes cannot afford expensive elaborated tamper protection. Therefore, a sufficiently capable adversary can manage to extract cryptographic information from nodes. As WSN missions are typically unattended, the potential for tamper attacks is significant.

As illustrated in Figure 17.1, we distinguish five main building blocks to secure WSN: (i) key management for WSN, (ii) securing radio channel access, (iii) securing data dissemination, (iv) securing data aggregation, and (v) securing the network against sensor nodes compromise. For each building block, we explain the security vulnerabilities and survey work in the area. Also, we briefly discuss advantages and drawbacks of existing solutions.

17.2. Background: Information Security and Cryptography

In this section, we review some common information security properties and present an overview of the cryptographic mechanisms used to achieve

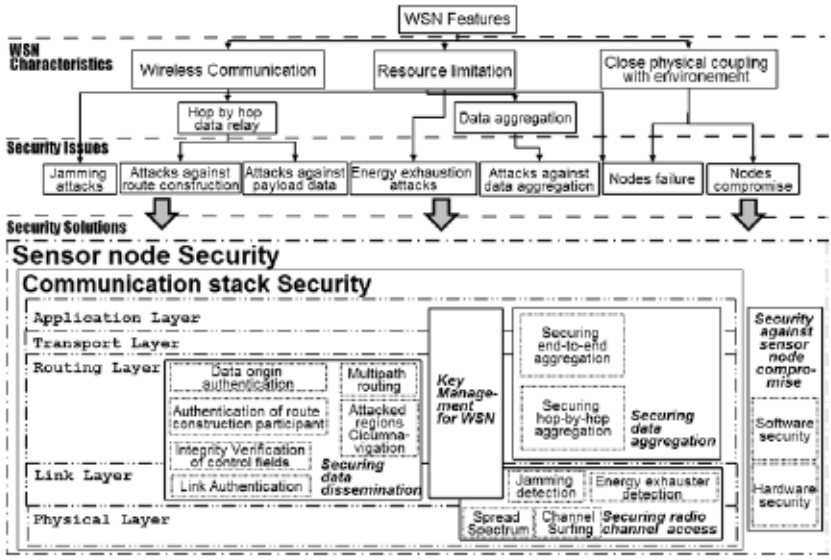


Fig. 17.1. Taxonomy of WSN security solutions.

them.[a] We consider a sender that sends a message to a receiver. For each security service, we present the cryptographic mechanism that the sender and the receiver should apply to guarantee the security service.

17.2.1. *Data integrity*

Data integrity is the property that data has not been changed, destroyed, or lost in unauthorized or accidental manner.[20] Cryptographic hash functions are typically used to assure data integrity.[21] Let us consider the following definition and then present a typical usage of cryptographic hash functions to assure data integrity:

A hash function is a computationally efficient function that maps binary strings of arbitrary length to binary strings of some fixed length, called hash-values.[21] *We denote the hash-value of a message* m *by* $h(m)$*. Cryptographic hash functions have the following properties* [20–23]: (i) *Given* m*, it is easy to compute* $h(m)$. (ii) *Given* $h(m)$*, it is hard to compute* m *such that* $h(m) = h$. (iii) *Given* m*, it is hard to find another message,* m'*, such that* $h(m) = h(m')$.

Example[b]: Suppose that you want to save a large digital document (a program or a database) from alterations that may be caused by viruses or accidental misuses. A straightforward solution would be to keep a copy of the digital document on some

[a]The given measurements are for cryptographic systems implemented on the Atmel ATMega128L (found in the Mica family: Mica2, MicaZ, Mica2Dot, ... etc.).

[b]This example is cited by many authors such as Kaufman *et al.* in Ref. 23 and Menezes *et al.* in Ref. 21

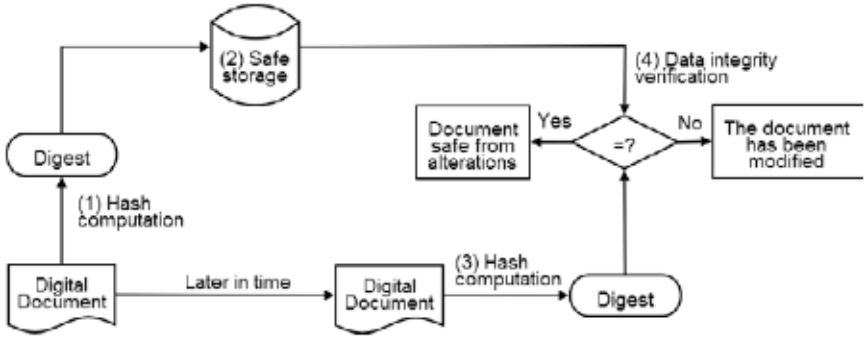


Fig. 17.2. Assuring data integrity using message digests.

Table 17.1 Measurements of some hash algorithms.

Hash algorithm	Digest size (bits)	Computation speed (bytes/ms)
MD5	128	23.81
SHA-1	160	7.93

tamper-proof backing store and periodically compare it to the active version. With a cryptographic hash function, you can save storage: you simply save the message digest of the document on the tamper-proof backing store (because the hash is small could be a piece of paper or a floppy disk) (see Figure 17.2, steps 1 and 2).

Then, periodically, you recalculate the message digest of the document (see Figure 17.2, step 3) and compare it to the original message digest (see Figure 17.2, step 4). If the message digest has not changed, you can be confident none of the data has.

Examples of hash functions are: MD2 (Message Digest 2),[24] MD5,[25] SHA-1 (Secure Hash Algorithm 1)[26] Table 17.1 gives some measurements[74] for usually used hash algorithms.

17.2.2. *Data origin authentication*

Data origin authentication is the validation that the source of data received is as claimed.[20] Message Authentication Code (MAC) is a cryptographic mechanism that can be used to assure data origin authentication and data integrity at the same time.

A Message Authentication Code (MAC) algorithm is a family of functions h_k parameterized by a secret key k, with the following properties: (i) *Given a key k and an input m, $h_k(m)$ is easy to compute.* (ii) *h_k maps an input m of an arbitrary finite bit length to an output $h_k(m)$ of fixed bit length. Furthermore, given a description of the function family h, for every fixed allowable value of k (unknown to an adversary), the following property holds:* (iii) *Given zero or more pairs $(m_1, h_k(m_1))$, it is computationally infeasible to compute any pair $(m, h_k(m))$ for any new input m.*[21]

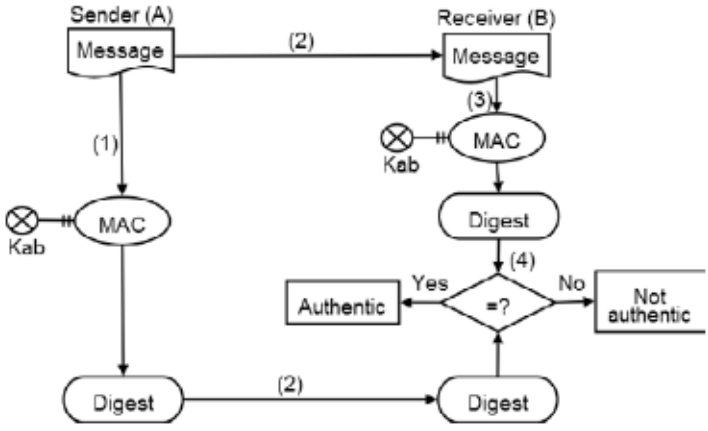


Fig. 17.3. Assuring data origin authentication using MACs.

The point of a MAC is to send something that only someone knowing the secret key can compute and verify. For example, a MAC can be constructed by concatenating a shared secret K_{AB} with the message m, and use $H(m|K_{AB})$ as the MAC (where H is a hash function).[23] Then, to assure data origin authentication, a sender (A) and a receiver (B) have to share a secret key K_{AB}. Then the sender computes the digest ($MAC(K_{AB}, m)$) corresponding to the message (m), to be sent, using the secret key (K_{AB}) (see Figure 17.3, step 1). Upon receiving the message as well as the digest, the receiver verifies the origin of the received message as follows: it recalculates the digest of the received message using the secret key K_{AB} (Figure 17.3, step 3) and compares it to the received digest (Figure 17.3, step 4). If the two digests are equal, the message is said to be authentic (has not been altered) and originates from the sender (A) since only (A) and (B) know the secret K_{AB}. Otherwise, the received message has been altered or fabricated by a sender who is not (A). An example of MAC is HMAC.[27]

17.2.3. *Data confidentiality*

Data confidentiality is the property that information is not made available or disclosed to unauthorized individuals, entities, or processes.[20] Confidentiality is guaranteed using encryption.

Encryption is a cryptographic transformation of data (called "plaintext") into a form (called "ciphertext") that conceals the data's original meaning to prevent it from being known or used. If the transformation is reversible, the corresponding reversal process is called decryption, which is a transformation that restores encrypted data to its original state.[20]

With most modern cryptography, the ability to keep encrypted information secret is based not on the cryptographic encryption algorithm, which is widely

known, but on a piece of information called a key that must be used with the algorithm to produce an encrypted result or to decrypt previously encrypted information. Depending on whether the same or different keys are used to encrypt and to decrypt the information, we distinguish between two types of encryption systems used to assure confidentiality:

17.2.3.1. *Symmetric-key encryption*

In a symmetric-key encryption system, a secret key is shared between the sender and the receiver and it is used to encrypt the message by the sender and to decrypt it by the receiver. The encryption of the message produces a non-intelligible piece of information and the decryption reproduces the original message (see Figure 17.4).

Examples of symmetric-key encryption systems are: DES,[28] AES,[29] IDEA.[23] Table 17.2 gives some measurements for usually used symmetric encryption algorithms.

17.2.3.2. *Public-key encryption*

Public-key encryption (also called asymmetric encryption) involves a pair of keys (a public key and a private key) associated with the sender. Each public key is published, and the corresponding private key is kept secret by the sender. Data encrypted with the sender's public key can be decrypted only with the sender's private key (see Figure 17.5).

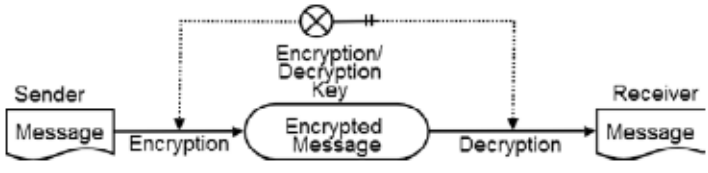


Fig. 17.4. Assuring confidentiality using symmetric-key encryption.

Table 17.2 Computation speed of some encryption algorithms.

Encryption algorithm	Encryption speed (Bytes/milli-second)
IDEA[75]	6.74
AES[74]	0.613

Fig. 17.5. Assuring confidentiality using asymmetric-key encryption.

In general, to send encrypted data to someone, the sender encrypts the data with that receiver's public key, and the receiver of the encrypted data decrypts it with the corresponding private key. Compared with symmetric-key encryption, public-key encryption requires more computation and is therefore not always appropriate for large amounts of data. However, it is possible to use public-key encryption to send a symmetric key, which can then be used to encrypt additional data. An example of asymmetric encryption systems is: RSA.[30]

17.2.4. *Non-repudiation with proof of origin*

Non-repudiation with proof of origin provides the recipient of data with evidence that proves the origin of the data, and thus protects the recipient against an attempt by the originator to falsely deny sending the data.[20]

Asymmetric cryptography is the basic answer for non-repudiation. With asymmetric cryptography, the piece of information sent with the message as a proof of integrity and data origin is computed using a private key held only by the sender and is verified by the receiver using the public key that corresponds to the private key. Hence, since only the sender can compute the piece of information, the latter can be used as a proof of origin to a third party and hence non-repudiation is assured. This cryptographic mechanism is called Digital Signature.

To sign a message, a sender generates a pair of private/public keys using some asymmetric cryptographic system. The sender keeps the private key secret and publishes the public key. Then the sender calculates the digest of the message to be sent using any hash function (see Figure 17.6, step 1). The digest is then cryptographically transformed using the private key (Figure 17.6, step 2). The result of this

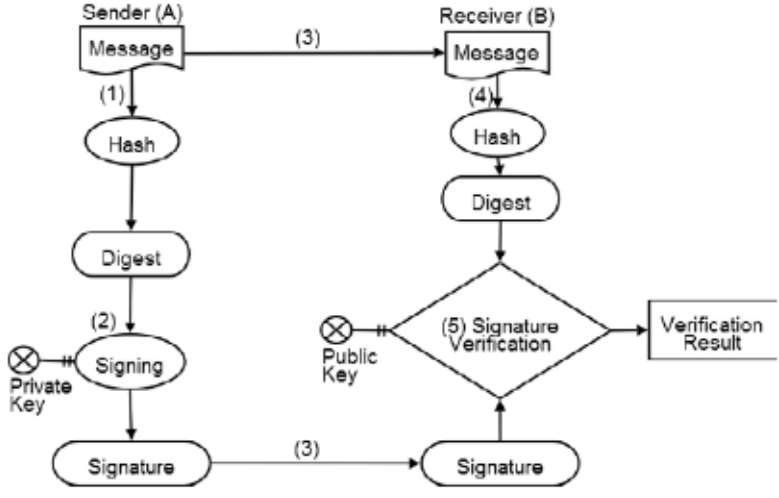


Fig. 17.6. Assuring non-repudiation using Digital Signatures.

Table 17.3 Measurements of some digital signature systems.

Digital signature	Signing speed (s)	Verification speed (s)	Public key size (bits)
RSA-1024	22.03	0.86	1024
ECC-160	1.65	3.27	160

transformation is called: the digital signature of the message. Upon receiving the message and the signature, the receiver verifies the signature using the public key as follows: first, the receiver recalculates the digest of the received message (Figure 17.6, step 4). Then, the receiver verifies the received signature using the public key (Figure 17.6, step 5). If the signature is valid then the message as well as its origin is authentic and non-repudiation is guaranteed. Otherwise the message is rejected.

Examples of digital signature schemes are: RSA,[30] DSA.[31] Table 17.3 gives some measurements[76] for usually used digital signature systems.

17.3. Key Management for WSN

Key management functions are the keystone of any communication-based system's security. In traditional networks such as the Internet, asymmetric cryptography is one of the most used key management schemes. Although asymmetric cryptography is very attractive as it provides reliable mechanisms for authentication and key distribution, its computation complexity, large memory footprints requirement and high computational overhead make it unsuitable for use in WSN. Recent researches[36,37] show, however, that it is possible to apply public key solutions to sensor networks by selecting the right algorithms and the appropriate parameters. In Refs. 34, 35 and 37, the energy cost of public key cryptography algorithms is quantified and it is shown that Elliptic Curve Cryptography (ECC)[72] has a significant advantage over RSA[30]: not only it reduces computation time but also it reduces the amount of transmitted and stored data. Some implementations of ECC for WSN can be found in Refs. 38 to 40.

17.3.1. *Literature overview*

Figure 17.7 shows taxonomy of a set of pre-distribution based key-management methods. In this taxonomy, protocols are classified into several categories according to the way neighboring nodes share common keys (probabilistic, deterministic) and to the network organization (hierarchical, flat).

17.3.1.1. *Probabilistic key management*

In a *pure probabilistic* key pre-distribution scheme,[41] each node is preloaded, prior to deployment, with a subset of keys taken from a very large *key-pool*. The main idea of this scheme is that two neighboring nodes have a certain probability of sharing

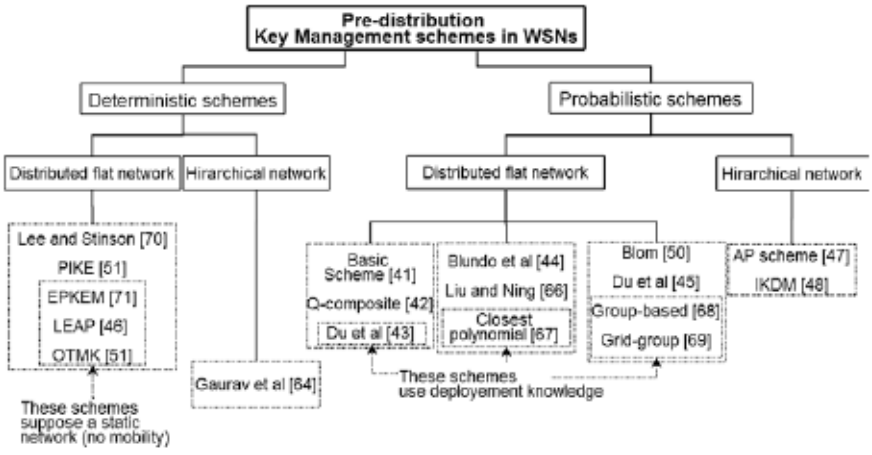


Fig. 17.7. Taxonomy of pre-distribution key management schemes in WSNs.

a common key that belongs to both key subsets of those neighbors. Although this scheme is simple and completely distributed, it has two main drawbacks. First, it requires large memory footprints to contain large subsets of preloaded keys so that the probability of neighboring nodes sharing common keys increases. Second, it cannot still provide sufficient security when the number of compromised nodes increases. To increase the resiliency of pure random key pre-distribution against compromised nodes, the *Q-composite* approach proposed in Ref. 42 allows two neighboring nodes to establish a secure communication link only if they share Q ($Q \geq 2$) common keys. Another work[43] takes advantage of deployment knowledge to improve both memory overhead and resiliency to node compromise. In this protocol, it is assumed that the deployment leads to collocated groups of nodes. Thus a sub-pool of keys is assigned to each group of collocated sensors. Each node in the same group selects a set of keys from its group sub-pool. Authors showed that two nodes in the same group will have a high probability to share a common key.

In Ref. 44, the authors use a symmetric bivariate λ-degree polynomial $\mathrm{P}(x, y)$ (such as $\mathrm{P}(x, y) = \mathrm{P}(y, x)$). Each sensor i with a unique ID, is preloaded with a polynomial share $\mathrm{P}_i(y) = \mathrm{P}(i, y)$. The common key between nodes i and j is $K_{ij} = \mathrm{P}_i(j) = \mathrm{P}_j(i)$. The solution is λ-secure, which means that keys are secure if no more than λ nodes are compromised. In order to make a solution more secure, a set F of λ-degree polynomials is generated. Each node i is preloaded with a subset F_i of the polynomial set F. Two nodes can establish a key, if they share a same polynomial in their subset of polynomials.

Du *et al.*,[45] adapted the idea of Blom's[50] symmetric key generation system in which each node i stores a column i and a row i of size $\lambda+1$ from matrices G (public matrix) and $(D*G)$ respectively, where $K = (D*G)^T * G$ is a symmetric matrix. A secret key between nodes i and j is generated as $K_{ij} = \mathrm{row}_i^* \ \mathrm{column}_j = \mathrm{row}_j^* \ \mathrm{column}_i$. The system is λ-secure, which means that keys are secure if no more than

λ nodes are compromised. To improve the resilience, the authors generate a space of w symmetric matrices $(D_1, D_2, \ldots, D_w)$ of size $\lambda + 1$ and select a random ζ distinct matrix spaces from the w matrix spaces for each node, this is similar to the assignment of random generated keys from a large key pool in Ref. 41. For each space selected by node j, the jth row is stored in the memory of this node. Two neighbouring nodes are able to calculate a common key only if they share the same matrix in their matrix spaces.

In networks with a hierarchical organization, probabilistic key management schemes take advantage of the assumption that some nodes have more resources than the others. Those powerful nodes usually elected to act as cluster heads, are preloaded with subsets of keys that are larger than the subsets of keys preloaded in the other ordinary nodes. These schemes reduce storage footprint at the ordinary nodes and achieve higher connectivity than those in flat networks.

17.3.1.2. *Deterministic key management*

In contrast to probabilistic schemes, deterministic schemes ensure that each node is able to establish a pair-wise key with its neighbors. To guarantee determinism, protocols such as LEAP[46] make use of a common transitory key that is preloaded into all nodes prior to deployment. The transitory key is used to generate session keys between neighboring nodes. To secure nodes against capture attacks, the transitory key is cleared from the memory of nodes after the generation of session keys.

In networks with a hierarchical organization, deterministic key management schemes centralize the responsibility for establishing secure links between the members of a cluster at the cluster-head. Although these schemes offer the attractive feature of being low energy, they are vulnerable to node capture attacks particularly when the cluster head is compromised.

17.3.2. *Description of some solutions*

17.3.2.1. *Random key pre-distribution*

The proposed protocol in Ref. 41 is the basis of probabilistic key pre-distribution schemes. It consists of three phases to establish a secure link between two neighboring nodes:

- Key pre-distribution phase: Each node, prior to deployment, is preloaded with a subset of keys of size K from a very large key pool of size $P \gg K$. A key identifier is associated to each key and stored into the sensor memory with the keys.
- Shared key discovery phase: After deployment, each node broadcasts the identifiers' list of its preloaded keys. Then, each node receiving this list discovers, eventually, at least a common key with its neighbor and establishes a secure link with it.

- Path key establishment phase: After the accomplishment of the above steps, two neighboring nodes may not share a key in their subsets of keys but are connected securely through one or more nodes. The latter are used to transmit securely a chosen key that will be used to secure a direct link between them.

Authors in Ref. 42 proposed a *q-composite* protocol that enhances the resilience to node capture of the basic scheme. In this protocol, two neighbouring nodes are able to establish a secure link only if they share at least q ($q > 1$) common keys in their preloaded subset of keys. For example, a pair-wise key, K, between two nodes that share q' keys ($q' \geq q$) is computed as: $K = \text{hash}\,(k1\|k2\|\ldots\|kq')$.

17.3.2.2. *LEAP*

LEAP[46] is a deterministic scheme that provides a mechanism to establish a pair-wise key between two neighbouring nodes. The protocol is based on the assumption that a sensor node, after its deployment, is secure during a time T_{min} and cannot be compromised during this period. The following steps illustrate how a newly deployed node establishes a pair-wise key with its neighbors.

- Key pre-distribution phase: The controller generates an initial key K_{IN} and preloads each node with this key. Each node u derives its master key from K_{IN}: $Ku = f_{KIN}(u)$.
- Neighbor discovery phase: After deployment, each node u broadcasts a HELLO message (1) that contains its identifier. When a node v receives the HELLO message it replies with an ACK message (2). The latter is authenticated using K_v that is calculated like this $K_v = f_{KIN}(v)$

$$u \;\; * : u \;\; (1)$$
$$v \;\; u \;\; : v, MAC(K_v, u \,|\, v) \;\; (2)$$

- Pair-wise key establishment phase: The node u calculates its pair-wise key with v: $K_{uv} = f_{Kv}(u)$. With the same formula, node v computes its pair-wise key with u.
- Key erasure phase: After the expiration of time T_{min}, the node u erases K_{IN} from its memory and hence forbids an intruder from getting this key.

After the above steps, the established pair-wise key between two nodes is used to secure communication between them.

17.3.3. *Analysis and discussion*

The main drawback of the probabilistic schemes is the high storage overhead that must be balanced against connectivity and resilience to node compromise. The deterministic schemes have fewer overheads but are more vulnerable to node compromise. For example, if the transitory initial key used in protocols such as LEAP is discovered then the entire network can be compromised. The hierarchical

schemes offer better performances than the flat ones. However, they rely on the existence of powerful nodes heading clusters, which might not be the general case of WSN. In what follows we define some basic metrics that are used to analyze the previous presented schemes.

17.3.3.1. *Connectivity*

We define the connectivity as the probability that a node can establish secure links with all of its neighbors. In the deterministic schemes all nodes are preloaded with the same secret that is used to establish a secure link with any neighboring node. Therefore, the connectivity is 100%. In the probabilistic schemes two neighboring nodes can build a secure link only if they share Q common keys in their subset of keys of size m. The smaller is Q the better is the connectivity. Thus, The basic probabilistic scheme,[41] the polynomial-based scheme[44] and Blom's matrix-based scheme[45] with $Q = 1$ have a better connectivity than the Q-composite scheme[42] with $Q > 1$.

17.3.3.2. *Resilience to node capture*

When a sensor node is captured by an adversary, its entire secret and the entire established links with its neighbors are compromised. The effect of such attack can be beyond the neighboring nodes of the compromised sensor. We define the resilience against node capture as the probability of not compromising any other link in the network when x sensor nodes are compromised.

In the deterministic scheme[46] all nodes are preloaded with the same initial key, which will be erased after establishing a secure link with all neighbors. If an adversary can compromise a node before erasing the secret, the security of the entire network is compromised. Therefore, the resilience of this scheme is very poor from this point of view. Nevertheless, the authors showed that it is infeasible using actual means to break this initial key in a duration less than the fixed threshold before erasing the key. In the probabilistic schemes, because the preloaded keys in sensor nodes are taken from the same pool, all the probabilistic schemes are vulnerable to sensor node compromising. Furthermore, the resilience to sensor node capture changes from a scheme to another according to the number of keys needed to establish a secure link. The Q-composite scheme with a high Q presents a best resilience comparing to the other probabilistic schemes. Indeed, an attacker needs to break at least Q keys of a sensor's key pool before compromising it. On the other hand, the greater the number (m) of the preloaded keys in a sensor, the worse is the resilience. This is due to the fact that compromising a sensor leads to compromising its key ring (m keys) and hence compromising all the secure links based on those compromised keys.

17.3.3.3. *Resilience to node replication*

After compromising a sensor node from the network, the adversary may attempt to add one or more nodes to the network that use the same key information of the

compromised node. The replicated nodes can be used to inject false data, modify or delete legitimate data of other sensor nodes.

The deterministic schemes, such as Refs. 46 and 51, use the same initial key in all nodes for authentication and the establishment of pair-wise key between nodes. Because the initial key is used to establish all pair-wise keys in the network, an adversary that obtains this key and duplicates it in other nodes, can successfully establish a pair-wise key with any other node in the network. In such a case, the network will be insecure.

All probabilistic schemes must provide the node-to-node authentication to prevent node replication attacks. However, the actual resilience against node replication attack is the same for all the probabilistic schemes, and depends on the probability that a replicated node can find a common key with other legitimate nodes. In Ref. 49, the authors studied the problem of node replication attack on probabilistic key management schemes. In their study they find that WSN with probabilistic key management schemes become almost 100% insecure in the presence of one replicated sensor, and the Q-composite scheme with large Q is more resilient against replication attack compared with the other schemes.

17.4. Securing Data Dissemination

In a multi-hop communication system, routing represents a fundamental service to guarantee the proper operation of the network. This predominant position makes this component an ideal target for an adversary who wants to disturb the normal operation of the network. In addition, the intrinsic properties of a WSN, such as the limited energy and short range communications, make such networks more vulnerable to security threats than traditional wireless networks.

Generally, routing mechanisms are mainly based on a hop-by-hop collaboration among sensors in order to gather the network's data and make it available for the end users. However, an uncontrolled collaboration, as found in most existing routing protocols, constitutes the major weakness that opens a variety of doors to an adversary in order to defeat the network security.

17.4.1. *Literature overview*

To build a secure dissemination module, two important components should be considered: *route construction* and *data relay*.

17.4.1.1. *Securing route construction*

The route management engine is responsible for the construction and maintenance of the communication backbone represented by the routing topology. Since the process of route establishment aims to build the backbone of the communication

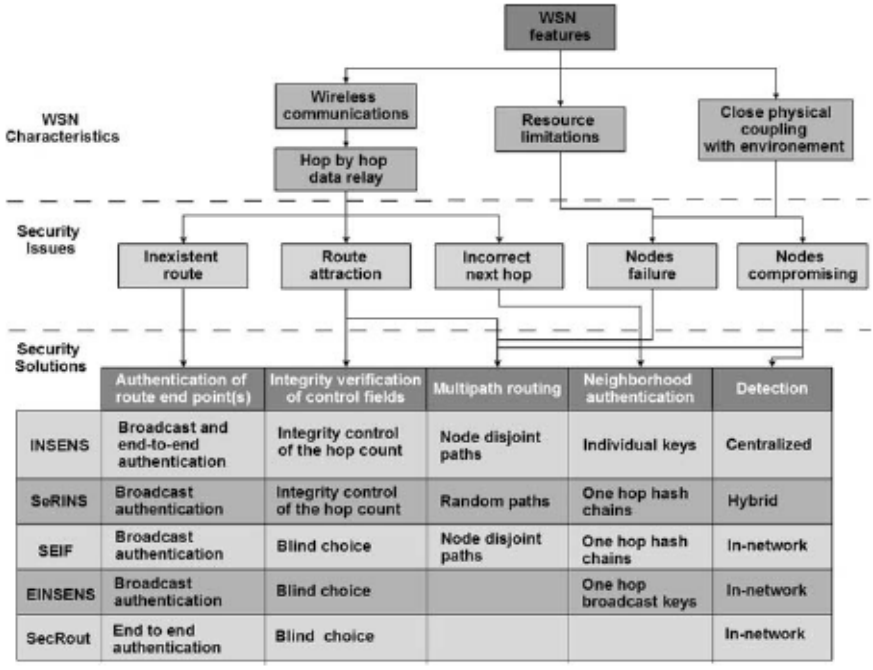


Fig. 17.8. Taxonomy of secure route construction in WSN.

system, an intruder will certainly try to disturb the normal operation of this important phase. Figure 17.8 gives a general overview of the security requirements and existing approaches to guarantee the construction of a correct routing state.

Before describing the proposed solutions, we analyze the goals of an intruder and the impact of the attacks at the routing layer:

- The first goal is to *inhibit the routing functionality* of some nodes in the network. This can take two forms: physical and logical inhibition. Logical inhibition is achieved by providing victim nodes with false routing information in order to isolate them from communicating with the sink. Loop creation and false neighborhood relationships are two examples of such attacks.

  On the other hand, attacks based on physical inhibition try to benefit from sensors' constraints to make them physically disconnected from the rest of the network. Generally, these DoS attacks try to exhaust the energy of sensors by forcing them to perform useless calculations, which will have more impact if the protocol employs some cryptographic operations during the route discovery phase.
- The second goal is to take *control over the traffic* by attracting the maximum number of routes. An intruder can "pretend" to have the best routes toward the sink, which gives him an important role in the final routing topology.

To protect a WSN against these malicious behaviors, several defense mechanisms were proposed in the literature. They can be classified into three main axes: *avoidance*, *detection* and *tolerance*.

17.4.1.1.1. Avoidance of route compromising

One of the main goals of a secure routing protocol is to provide each communicating sensor with a correct path to the corresponding sink. In order to choose one path among the discovered ones, routing protocols use some metrics. However, this enables an intruder to modify the metric and to "pretend" having the best route to obtain a predominant position among the routers. To overcome this problem, two classes of solutions were proposed:

- Some existing solutions, like EINSENS,[52] SEIF[60] and SecRout,[53] employ a radical method by using a *blind path choice* that does not reflect any meaningful metric. This approach aims at reducing attack opportunities on the path choice mechanism. EINSENS[52] and SecRout[53] are both based on choosing the first neighbor advertising a correct route as the next hop. The protocol SEIF[60] uses a more resilient blind choice since it selects a random next hop among a set of alternative routes.
- A second family of protocols ensures an integrity control of the hop count metric, used for building short paths. There exist two main approaches to guarantee this integrity. The protocol INSENS[52] employs a totally centralized approach requiring excessive communication overhead, while the protocol SeRINS[54] uses a semi-distributed solution providing a more scalable scheme.

Another avoidance method is *to verify the link state and authenticate the next hop node*, in order to prevent incoherent neighborhood information. Since a WSN may contain powerful intruders, an attacker may use a high-powered transmitter to reach a large set of nodes, to make them believe that they are neighbors of him while they are not. To defend against this Hello flooding attack, each sensor should discover its *reachable neighborhood*, consisting of neighbors having a bidirectional link using a challenge-response mechanism. We can distinguish between three existing mechanisms. In INSENS,[52] sensors are preloaded with individual keys and will generate and exchange proofs of neighborhoods that will be forwarded to the sink node. The latter can verify the proofs and infer correct neighborhood relationships. A second approach, found in EINSENS,[52] is to use *one hop broadcast keys* to authenticate the relay of the RREQ message. Using a key management protocol such as LEAP,[46] each sensor should establish a secret key shared only with its reachable neighborhood. The third approach, used by SeRINS[54] and SEIF,[60] is based on the concept of *one way hash chains* for one hop authentication.

A one way hash chain (OHC) is used as a generator of one way sequence numbers (OWS), which provides an efficient manner for authenticating a sequence of messages in a one-to-many communication. It consists of a sequence of numbers

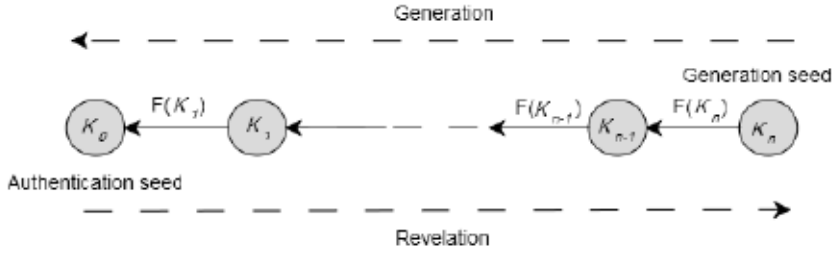


Fig. 17.9. A one way hash chain.

$(K_i)_{0\leq i\leq n}$ generated from a random seed K_n as follows: $K_i = F(K_{i+1})$, where F is a one way function (see Figure 17.9*). This entire chain is stored in the source node, whereas the eventual recipients of future messages are loaded only with the first value K_0 (named chain verifier or authentication seed). For each sent message, the source node reveals a new value from the chain in the reverse order of its generation, i.e. $K_1, K_2, \ldots, K_n$. Since nodes are initialized with K_0, they can verify the belonging of any received value to the chain by using the following property: $K_0 = F^i(K_i), \forall 0 < i \leq n$.*

Whatever the used route compromising avoidance technique, to guarantee that a discovered route really exists, it is necessary to ensure *authentication between route end points*, with defense against replay attacks. In tree-based protocols, such as INSENS,[52] SeRINS,[54] SEIF[60] and EINSENS,[52] the construction is initiated by the sink node using a global diffusion. Thus, this family of protocols employs a broadcast authentication of the sink node using one way hash chains, comparable to the μ Tesla[73] mechanism. However, in reactive protocols, like SecRout,[53] sensors represent the starting points in route discovery. Therefore, communication is one-to-one, allowing an end-to-end authentication based on shared secret keys with the sink.

17.4.1.1.2. Detection of corrupt routes

The verification of route correctness and the correction of the eventual problems is the cornerstone of each secure routing protocol. There exist three classes of mechanisms:

- In centralized solutions, like INSENS,[52] all verifications are processed at the sink node. The latter should collect neighborhood proofs from all the sensors, producing a correct cartography of the whole network.
- In in-network detection schemes, like EINSENS[52] and SEIF,[60] the verification is done hop-by-hop. The sensors perform all security controls without referring to the sink.

A trade off between the above solutions, used in SeRINS,[53] consists in delegating only partial verifications to sensors. Due to this partial information, when a node

detects a problem, it cannot make a decision without referring to the sink node. So, the role of the sink node is curative and intervenes only in presence of false routing information.

17.4.1.1.3. Fault tolerance of compromised routes

Since sensors can fail or become compromised, the discovered routes may expire and become unusable. To overcome the shortcomings of a periodical reconstruction, sensors should "anticipate" these situations by constructing alternate paths. This multi-paths solution significantly enhances the resiliency against node failure and presence of intruders. In the literature, the most important secure protocols for WSN providing multi-paths routing are INSENS,[52] SeRINS,[54] and SEIF.[60]

17.4.1.2. *Securing data relay*

Securing route construction is not sufficient to ensure secure data collection in a WSN: an intruder can be interested in falsifying the current readings, in order to make part of the network reflect an inconsistent state from the real sensed environment. This falsification can be conducted by direct injection of bogus data or by modification of forwarded packets. Another issue consists in injecting a large burst of data packets in order to drain the energy of relay nodes. This type of attack is known as a Path-based DoS attacks (PDOS). Figure 17.10 gives an in-depth view of the issues and solutions for the data relay problem.

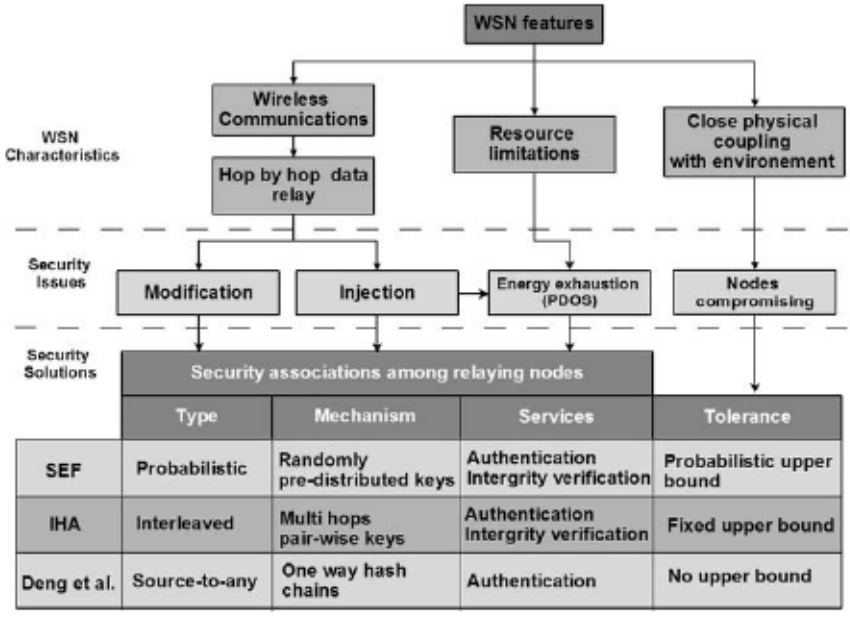


Fig. 17.10. Taxonomy of secure data relay in WSN.

To address these problems, it is important to embed a data origin authentication mechanism within the relay engine, which will allow an en-route filtering of bogus data. In existing solutions, the essential idea to implement this origin authentication is to establish *security associations among relay nodes*. One naïve approach is to establish for every path one secret key shared among forwarding nodes and the source of the data. Using this secret key, the source node can generate a MAC for each transmitted data, which allows relay nodes to verify the origin of the received data packets. However, this simple solution does not provide any resiliency in presence of malicious nodes inside the path. More sophisticated approaches were proposed to provide a stronger and more resilient data origin authentication during the relay phase:

- Instead of using one secret association among relay nodes, IHA[56] proposed to establish several associations using multi-hop pairwise keys between distant nodes in the path. Each sensor in the path discovers a multi-hop upstream and a downstream node, and establishes a secret key with each one of these associated nodes using an id-based key protocol.[44,59]
- SEF[57] tries to avoid the key establishment phase of IHA by using a random pre-distribution approach. Each sensor is preloaded with a ring of keys from a common pool. The source node sends its data with a MAC generated using a randomly chosen key. SEF relies on the fact that since keys are from the same pool, there is a certain probability that one relay sensor possesses the key used by the source node to generate the MAC.
- A more efficient solution for origin authentication is to use one way hash chains (Deng *et al.*[55]). These chains can be considered as an authentication sequence number that can be generated only by the source node, and can be verified by any node in the path. Hence, intermediate sensors can authenticate any received packet and immediately reject faulty ones.

17.4.2. *Description of some solutions*

17.4.2.1. *INSENS*

INSENS[52] is a centralized link-state protocol designed to be highly tolerant to failures and intruders. The main functional blocks of this solution can be summarized in two points:

- ***Link-state collection*:** The base station periodically starts new route construction by gathering the link-state of each sensor in the network. Since this step is prior to route establishment and sensors have to use a multi-hop communication to send their state, a temporary tree is constructed before starting the collection phase. The establishment of the tree is based on a simple flooding of a RREQ message initiated by the sink node. The latter includes in the RREQ a one way sequence number to enable a one-to-many and network-wide origin

authentication. Each sensor chooses the sending node of the first RREQ message as its parent. After this choice, the node relays the RREQ message by adding its identifier, and its neighborhood proof consisting in a MAC generated with its own secret key shared with the base station. Therefore, this hop-by-hop relay of the RREQ message allows nodes to discover the proofs of the direct neighbors and in the same time construct the temporary tree. The collected link-state information, consisting of the list of neighbors and their respective proof, is sent to the base station using a RREP message. This message is relayed from parent to parent until reaching the base station.

- ***Routing table construction*:** After receiving RREP messages, the base station starts the construction of a correct connectivity graph by verifying the coherence of the link-states. This verification is performed by recomputing the proof for each sensor and comparing it with the one provided by every "claimed" neighbor. Afterwards, the base station can apply on the resulting graph any type of path finding algorithm to build the routing tables. INSENS builds a pair of 2-hops disjoint routes for every sensor, meaning that each node in the first path should be at least at two hops far from every node in the second path. The routing table of each sensor i will contain the source node and the next hop for every path passing through i. To communicate this information without flooding the entire network, a path should be used for every node. Since the previous temporary tree is not adequate for this sink-to-node communication, INSENS employs the following solution. The base station operates by rings and sends these tables to nodes at hop h before nodes at hop $h+1$. When a node receives a table sent to another node, it has to check if the target node is included in its routing table as a source node. In this case, the receiving node just relays the message to its neighbors.

17.4.2.2. *EINSENS*

ENINSENS[52] was proposed by the same authors of INSENS to provide a better scalability. However, this new version gives a lower tolerance since it uses a single-path routing. The authors proposed a solution for this shortcoming by "emulating" a multi-path behavior using several sinks with one path to each one of them.

The distributed nature of EINSNES imposed also another constraint. To support the in-network verification, sensors must establish some key materials before starting the protocol, leading to more processing and storage requirements.

EINSENS can be considered as a secure version of the TinyOS beaconing protocol. Periodically, the sink node constructs a new spanning tree by flooding a RREQ message having the following format:

$$i \rightarrow * : i, E(BK_i, ows \| i).$$

The one way sequence number *ows* and the id of the sending node are encrypted with the broadcast key BK_i to allow a one-hop authentication among reachable

neighborhood. When a node j receives such message, it decrypts the secret part and checks if it indicates a valid new round. In this case, node j chooses i as its parent and relays the RREQ using the same encryption procedure. Consequently, each sensor will discover a valid parent resulting in a reliable communication tree rooted at the sink node.

17.4.2.3. *SEIF*

SEIF[60] is a secure multi-path protocol based on a totally in-network verification scheme. In contrast to other solutions, SEIF provides an efficient and secure method to build node disjoint paths in a totally distributed manner without any referring to the base station. Path construction in SEIF is based on the idea of branch-aware route discovery.[63] It consists in *tagging* the exchanged RREQ messages with the identifier of the first relaying node after the base station. We call these nodes *root nodes*, and their sub-trees *branches*. Using these tags, a sensor can easily decide whether two RREQ came from disjoint routes by comparing their branch id. Figure 17.11 shows an example of a branch-aware discovery process. Nodes a, b, and c represent the root nodes of the branches. When two neighboring nodes belong to two distinct branches, they discover a new disjoint path through each other. For instance, node e posses a main route through its main branch rooted at node a, and has also an alternative route through its neighbor h in the branch rooted at node b (and both routes are node disjoint).

To secure this simple but efficient mechanism, SEIF replaces the branch ids with one way hash chains in order to avoid fabrication of bogus branches. In fact, using plain text identifiers for root nodes allows an intruder to attract more routes by

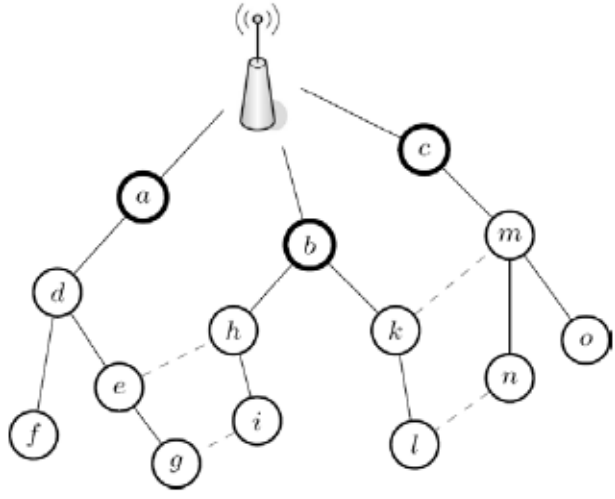


Fig. 17.11. Discovery of node disjoint paths using a branch-aware discovery.

injecting inexistent branches. The one way hash chains will prevent such malicious misbehavior by allowing legitimate sensors to authenticate the root nodes.

Before deployment, a set of hash chains are generated and stored in the base station. Each sensor i is preloaded with the first value of each chain j, which will constitute the initial value of the chain verifier $CV_{i,j}$. In addition, sensor i also maintains the position $P_{i,j}$ of that value within its corresponding chain. At each round, the base station discovers its neighborhood, maps each neighbor i to a hash chain n and sends to it the next value $V$and its position p. This message contains also a one way sequence number *ows* for round identification and the whole message will be encrypted using the secret key K_i shared between i and the base station:

$$BS \rightarrow i : E(K_i, n \parallel p \parallel V \parallel ows).$$

After verifying the one way sequence number *ows*, i authenticates the new hash value by verifying the belonging of V to the claimed chain as follows:

$$\begin{cases} p > P_{i,n} \\ CV_{i,n} = F^{p-P_{i,n}}(V) \end{cases}.$$

This authenticated V value will allow i to securely construct its branch since it will be its proof of being a valid root node, i.e. a real neighbor of the base station. Node i forwards the RREQ message having the following format:

$$i \rightarrow * : i, n, p, V, ows, oha.$$

The *oha* field represents the *one hop authentication* mechanism. It is the next value of a local one way hash chain stored in each sensor. Before deployment, each node i generates its own hash chain and will distribute its first value to the reachable neighborhood.

When a neighbor j receives the RREQ, it will verify the three chain values: V, *ows* and *oha*. Especially for the round verifier, node j should accept messages with the same *ows* in order to discover alternate paths. If these three values belong to their respective chains, j accepts i as an alternative parent since it provides a valid disjoint path form a new branch. After a random period of alternative route discovery, j relays the RREQ by selecting a random parent and relaying the RREQ with the corresponding branch authentication credentials.

SEIF also proposed an extension to find more alternative paths and increase the tolerance of the topology without requiring additional messages. Instead of tagging routes with the roots' ids, the tagging responsibility will be assigned to the neighbors of root nodes, i.e. 2-hops neighbors of the base station. By adding this second level tagging, SEIF allows root nodes as the solely intersection points between routes. Neighboring nodes of roots can become sub-roots and thereby construct their own sub-branches. A sensor will accept paths within the same branch only if they come from different sub-branches, which will increase the number of alternative routes since the basic version presented earlier discards any additional route from an already discovered branch.

17.4.2.4. *SERINS*

SeRINS[53] falls into the category of secure and tolerant routing protocols based on multi-path topologies. One major contribution of this work is its semi-distributed protection of the hop count metric using a set of one way hash chains. Three types of chains are present in SeRINS:

- As for previous protocols, one chain is used for sink authentication and round identification. At each round, the sink node reveals a new value of the chain in the reverse order of its generation.
- An "on the fly" chain is constructed during the relay of the RREQ message to prevent an intruder from decreasing its hop count. This chain is implemented by adding a field in the RREQ message named nI I that will represent the hop count as successive calls to a one way function F on a random seed value.
- The last chain is similar to the previous one and uses a field named nII that aims to detect a malicious node trying to relay the nI of its parent without applying the one way function.

Initially, the BS broadcasts a RREQ with a null hop count and a random nI:

$$BS \rightarrow * : BS, \quad h = 0, \quad nI = r, \quad nII = 0, ows.$$

When a sensor detects a valid new round, it starts an initial phase for discovering its *main parent* before relaying the RREQ. This node is selected as being the neighbor presenting the lowest hop count.[c] When the main parent is selected, the node relays the RREQ as follows:

$$i \rightarrow * : i, \quad h_p + 1, \quad F(nI_p), \quad F(nI_p \| i), ows$$

where h_p, nI_p and nII_p represent the received values from the chosen main parent.

Just after relaying the message, node i starts discovering the eventual alternate paths. When i receives a sub-sequent RREQ message $\langle j, h_j, nI_j, nII_j, ows \rangle$ indicating the same round, it has to perform two tests in order to accept the sending node as an alternative parent:

(a) If $h_j \leq h_p$, node i checks whether $F^{h_p - h_j}(nI_j) = nI_p$. This test will be successful only if both $nI_j$and nI_p were generated from the same seed, since $F^{h_p}(r) = nI_p$ and $F^{h_j}(r) = nI_j$.

(b) If $h_j = h_p + 1$, nodes i and j are at the same distance to the base station. To verify that j did not relay the nI of its parent without applying the function F on it, node i uses the nII field. Since nI of the parent of j should be equal to nI_p, i can verify if $nII_j = F(nI_p \| j)$. This means that the parent of node j is at h_p hops and that j relayed correctly the hop count metric of its parent.

[c] One main drawback of SeRINS is that this choice is performed without any security check, which represents a considerable vulnerability.

If one of the previous tests fails, node i sends an alert message to the base station. To guarantee that the attacker will not block this important message, SeRINS sends it using a global network flood. After authenticating this alert, the base station interrogates the neighbors of the suspect nodes in order to collect more information about it. If the collected hop counts are consistent, the suspect node is declared as compromised; otherwise the base station deduces that the source node of the first alert sent a false alarm and is considered as compromised.

17.4.2.5. *SecROUT*

SecRout[53] is a hierarchical routing protocol based on a reactive and sensor-initiated route discovery. Nodes are organized into clusters forming a two-level communication architecture. Cluster-heads collect data readings from their members, and aggregate them using some chosen functions (such as SUM, MAX, AVG, ... etc.). If the cluster-head does not posses a fresh route to send this aggregated report to the base station, it should start a new route discovery process. The node starts by broadcasting a RREQ message containing a MAC generated using the secret key of the cluster-head shared only with the base station. The RREQ message is relayed using a simple flooding technique until reaching the base station. During this relay, the message keeps track of the ids of the last two forwarding nodes. Using this information, each relaying node discovers the two next hops to reach the source cluster-head.

When the RREQ message reaches the base station, the latter authenticates the source node by verifying the provided MAC. If this authentication succeeds, the base station replies with a RREP message, which will travel on the reverse path of the received RREQ message. The RREP message contains also a MAC generated with the secret key of the source cluster-head. Similarly to the RREQ, the RREP message will also keep track of the last two relaying nodes, which will help to trace the path toward the base station. When this RREP reaches the source cluster-head, it verifies the MAC field using its secret key. If the MAC is correct, the node refreshes its routing table and sends the aggregated report to the newly discovered next node. The data message will be relayed hop-by-hop along the path traced by the forwarding of the RREP message.

17.4.2.6. *IHA*

IHA[56] was proposed as a solution for the false data injection problem. The protocol uses the concept of multi-hop pairwise keys to achieve remote security associations among forwarding nodes, which is necessary to ensure origin authentication during the relay (see Figure 17.12). IHA assumes that the network is organized into clusters that should contain at least t members. To establish the security associations, each node should discover its t-hops neighbors along the path in both directions: upstream and downstream. For that, the base station broadcasts a Hello message relayed recursively, which will keep track of the last $t+1$ relay nodes. Therefore, each

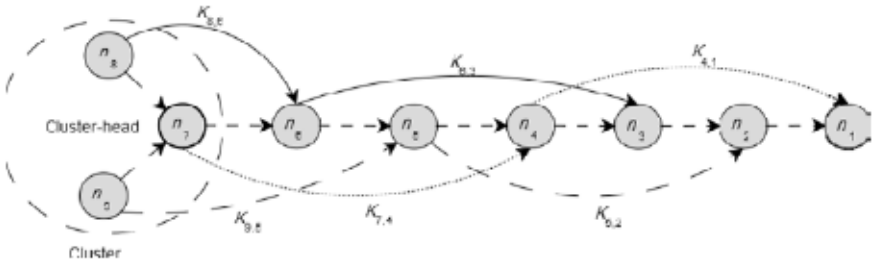


Fig. 17.12. The interleaved hop-by-hop authentication mechanism.

sensor knows the id of its upstream association node, which will be the first node in the received list in the Hello message. When a cluster-head receives the Hello message, it assigns each of the $t+1$ ids in the message to one of its cluster nodes (including itself). The cluster-head replies to the base station by sending an ACK message using the reverse path. This message is similar to the Hello message and aims to discover the associated downstream nodes in the path. When a sensor discovers a new associated node, it can use an id-based key establishment scheme to generate a pairwise key without additional messages.

IHA is based on a two-level data gathering scheme. When an event is detected in a cluster, at least t members should respond to the event by sending to the cluster-head two MACs: an *individual MAC* computed using the secret key of the node and a *pairewise MAC* computed using the pairwise key shared with the upper association node. The cluster-head collects the t individual and pairwise MACs with the ids of the members to construct the final report. To reduce the size of the packet, the individual reports can be compressed by XORing them into one MAC. When a forwarding node u receives the report, it checks the pairwise MAC of its downstream association node. If the MAC is correct and u is more than $t+1$ hops from the base station (i.e. u has an upstream association node), u replaces this MAC with its own pairwise MAC generated using the key shared with the upstream association node. When the base station receives the report, it can check the compressed individual MACs by recomputing them using the secret keys of the source nodes. The resulting MACs are XORed and compared to the received value.

17.4.2.7. *SEF*

Instead of establishing associations with multi-hop keys, SEF[57] employs a pre-deployment key assignment. Before network installation, a pool of N keys is generated and stored in the base station. This global pool is divided into n non-overlapping partitions of m keys. Keys are also assigned a unique identifier known by every node. Each deployed sensor is preloaded with k random keys from a partition chosen randomly.

When a group of sensors detect an event, they elect a center of stimulus (CoS), which can be similar to the clustering approach of IHA. Each detecting sensor

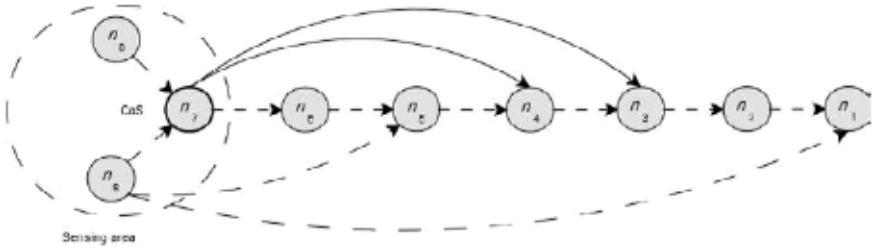


Fig. 17.13. Probabilistic security association in SEF.

chooses randomly one of its k keys and computes a MAC on the event value. The MAC and the id of the used key are sent to the CoS, which reorganizes the received MACs by the partition of their keys. The authors of SEF name the MACs generated by keys from the same partition as one category. The CoS selects T categories and picks one random MAC from each category. These MACs with the ids of the keys and the event value will form the final report. When a relaying sensor receives this report, it will check if it has one of the T keys used in the report. If there is no common key or the node possesses the right key and the MAC is correct, the packet is forwarded to the next hop.

17.4.3. *Analysis and discussion*

17.4.3.1. *Detection overhead*

The impact of a detected attack represents an important criterion for comparing existing solutions. In INSENS, forged routing information will have no consequence on sensor nodes, because nodes are no longer overburden with security checks, which are all performed at the sink level. In distributed solutions, incorrect RREQ messages will also be without negative effects, since any attack is immediately stopped by sensors. Sensors rely only on local information to successfully detect forged routing messages. Therefore, any intrusion attempt is instantly detected without additional delay. In contrast, SeRINS is a hybrid protocol in which sensors can perform only partial verifications that limit the ability of sensors to make local decisions in presence of suspect messages. Indeed, when a sensor detects a suspicious packet, it alarms the sink which must collect more information on the suspect node from its neighbors. This process is achieved via successive broadcasts, which is too expensive in large networks causing additional delay and overhead to detect the intruder.

17.4.3.2. *Tolerance*

In many situations, sensors have to be deployed in hostile or inaccessible environments, making very difficult the manual control and the individual monitoring of sensors. To maximize the network lifetime, the routing topology should tolerate as much as possible the presence of failures by preserving the connectivity of the

remaining active nodes. A disconnection occurs when the graph corresponding to the network topology contains more than one connected component. In this case, we will have some sensors without a valid route toward the base station.

When classifying routing algorithm depending on the provided tolerance, we can distinguish between three categories. *Single-path protocols*, like EINSENS and SecRout, provide the poorest resiliency due to the lack of redundancy in their topologies. *Multi-path routing approaches* provide better performance but with different levels. SeRINS builds alternative paths without considering their intersection. This uncontrolled route selection will have a negative impact on the probability of network disconnection. In fact, when a node belongs to more than one path for a given sensor, its failure will cause the disconnection of all these alternative paths. On the other hand, node-disjoint routes do not suffer from such problem since a node failure will cause at most one path to fail for every sensor in the network. Therefore, the impact of a failure on the routing topology decreases, meaning that the system will tolerate more failures, which is the case of SEIF protocol.

17.4.3.3. *Scalability*

To guarantee large scale deployments of WSN, it is important to study the scalability of the proposed solutions and the required number of control messages. Protocols requiring feedbacks from sensor nodes, such as INSENS and SecRout, suffer from a poor scalability. This is due to the one-to-one dialogue between each sensor and the base station, which leads to an excessive communication overhead when dealing with a large number of sensors. By removing the feedback phase, tree-based protocols (like EINSENS, SEIF and SeRINS) improves significantly the overall scalability by requiring only one message per sensor in order to establish the communication topology.

17.5. Secure Aggregation in WSN

Data aggregation is a collection of data readings that represents a collaborative view of a set of nodes. Generally, it is applied in a very specific area limited by the sink node. The latter sends a final aggregated data to the base station using a routing protocol. The number of source nodes sending information to the sink may be large and the information carried within the packets they transmit may be redundant. Data aggregation aims at increasing the energy saving by reducing the forwarding load of intermediate nodes. When data aggregation is used, intermediate nodes merge multiple packets into one to reduce the amount of transmitted packets. By reducing the number of transmitted messages, data aggregation also contributes to reducing the number of collisions especially in dense networks. The aims of securing data aggregation can be summarized in the following points.

- Ensuring authentication and data integrity of aggregation results: Data aggregation requires that each intermediate node be able to read and modify the data transmitted in packets. Under this condition, ensuring data integrity becomes challenging. As using a shared key between each node and the sink to secure the data contents denies the use of aggregation mechanisms, nodes use pair-wise keying instead. A pair-wise key is a common key shared between a node and its upstream node. It allows the upstream node to manipulate the received data without any control. Thereby, securing data aggregation should allow detecting corrupt aggregation operations, and/or injection of faulty data in the aggregation process.
- Preserving the benefits of data aggregation in terms of energy consumption: The main objective of data aggregation is to reduce energy consumption through minimizing data transmission. Thus, securing data aggregation should not thwart this objective through introducing supplementary computations and transmissions.

17.5.1. *Literature overview*

When an intruder node has no access to the key material of a legitimate node, all the proposed protocols can prevent data aggregation corruption. However, when an intruder obtains the key material of a legitimate node, it can carry out attacks by injecting faulty data or by falsifying the aggregation content. To avoid these problems, many protocols have been proposed in the literature.

Data aggregation protocols may be divided into two categories: *End-to-end* encrypted data, and *hop-by-hop* encrypted data, as shown in Figure 17.14.

17.5.1.1. *Protocols based on end-to-end encrypted data*

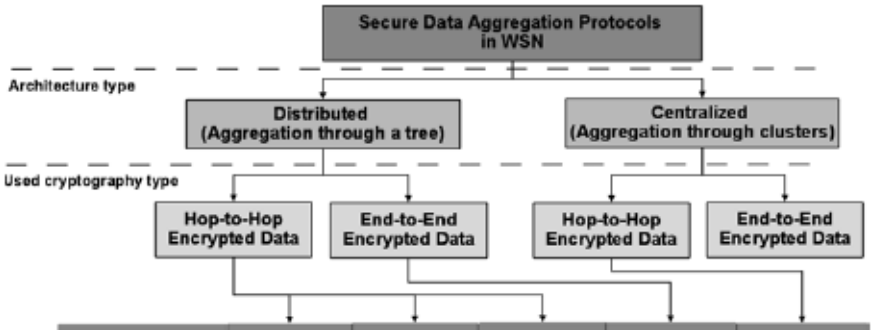
Protocols of this category make use of a shared key between each node and the sink to guarantee the integrity of the transmitted data. As the data content is encrypted, intermediate nodes use a particular encryption transformation called Privacy Homomorphism (**PH**)[8] to be able to perform aggregation without disclosing the content of data. The **PH** encryption verifies the following property:

$$E_{K1}(x_1) \oplus E_{K2}(x_2) = E_{K1 \oplus K2}(x_1 \oplus x_2),$$

where k_i are keys and x_i are data. The single point of verification in this type of protocols (such as DEPH,[1] CMT,[2] and ECEG[3]) is the sink node that holds all the keys used to encrypt data in the network. This idea has been used in many protocols with different cryptographic techniques such as modular addition[8] that uses a temporary symmetric key, or Elliptic Curve Cryptography.[72]

17.5.1.2. *Protocols based on hop-by-hop encrypted data*

In contrast to protocols based on end-to-end encrypted data, protocols based on hop-by-hop encryption make use of other mechanisms to guarantee integrity while



Protocols	SDAP	SAWN	SEDAN	CMT	SecureDAV
Energy Consumption	–	–	+ +	+ +	+
Blind Rejection	No	Yes	No	Yes	Yes
Impersonation Attack	Yes	Yes	No	Yes	No
MTTD	+ +	–	+ +	–	–
Localization of compromised node	Yes	Yes	Yes	No	Yes
Scalability	Yes	No	Yes	Yes	No

- : Bad + : Good ++ : Very good

Fig. 17.14. Taxonomy of secure data aggregation protocols.

allowing data aggregation in plain text. To ensure the integrity of data transmitted between nodes, each protocol uses a different verification mechanism. SAWN[4] proposes a two hop verification mechanism to prevent data modification by the next hop during the aggregation process. Each node generates a MAC of its data to its grandfather in the aggregation tree using a temporary key. This MAC allows the grandfather to verify aggregation value calculated by its child. The verification of MACs is delayed until the sink node discloses the series of temporary keys used to calculate the MACs. Based on the same mechanism of SAWN for verification, Bagaa *et al.* proposed SEDAN.[5] The protocol improves the energy consumption by using a new type of keys between each node and its grandfather. This type of keys eliminates the broadcasting of series of temporary keys. Both SAWN and SEDAN rely on the assumption that the network does not contain two consecutive compromised nodes.

Kui *et al.*,[6] proposed Secure Data Aggregation without Persistent cryptographic operations in WSNs (SDAP). The protocol is based on a clique topology and uses a watch dog mechanism; each node in the same clique can listen to each other, then verify the aggregation value calculated by their parent. Similarly, SecureDAV[7] is a mechanism that uses a collaborative participation of some nodes chosen randomly

from the members of a cluster to generate a signature over the average value of the cluster. This signature will be verified by the sink node.

17.5.2. *Description of some solutions*

17.5.2.1. *CMT*

The CMT protocol is based on the assumption that each node shares a symmetric key with the sink node, this key must be renewed at each aggregation process. In this protocol each node encrypts its sensed data by adding the shared key to the data, using a modular addition function. Using the same modular addition function, each aggregator node aggregates all received encrypted data. When the sink node receives the final aggregation value, it decrypts the received value through subtracting the added nodes' keys.

17.5.2.2. *SAWN*

SAWN is a secure hop-by-hop aggregation protocol based on delaying both, the aggregation and the aggregation verification processes. Instead of calculating the aggregation value at the parent level, it is delayed to the grandparent level. Moreover, the verification is delayed until receiving the final aggregation value and revealing all needed temporary keys by the sink node. This protocol is the first which introduces the two hop verification mechanism: first, each node uses its own temporary shared secret key with the sink node to calculate a Child-MAC over the data, and sends both the data and the calculated MAC to its parent. Second, the parent node sends all the received data and MACs to the grandparent node and also a Parent-MAC for the calculated aggregation value over the received data using its own temporary shared secret key. Third, in order to verify the calculated aggregation value, the grandparent node stocks all the received data, Child-MACs and Parent-MACs sent by the parent node. Finally, after receiving all child and grandchild keys, the grandparent node can verify the correctness of the calculated aggregation value.

Example: Figure 17.15 illustrates the different steps described above: nodes A and B send their sensed data, in addition to a MAC calculated over that data using a secret key shared with the sink, to their parent C. E and F do the same with their parent G. The parents C and G calculate the aggregation value using a function f and authenticate the aggregation value with a MAC. Then, they forward the received child data in addition to the authenticated aggregation value to their parent D which repeats the process with its parent. When the sink disclosed the secret keys, the grandparent D will be able to verify the correctness of the received aggregated value. Indeed, node D recalculates the aggregation values and the MACs using the appropriate disclosed keys and compares the results with the MACs received from nodes C and G.

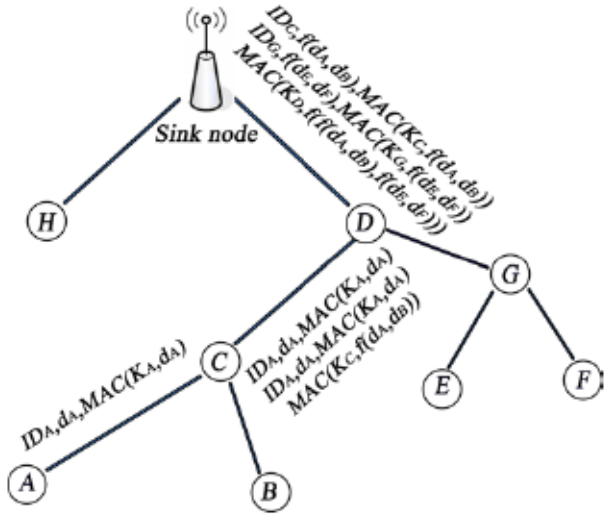


Fig. 17.15. SAWN aggregation and verification processes.

17.5.2.3. *SEDAN*

SEDAN uses the same two hop verification mechanism as SAWN. SEDAN enhances this mechanism by eliminating the use of temporary keys and hence avoids the overhead due to key broadcasting. Indeed, SEDAN introduces a new type of keys shared between each node and its grandparent. The use of this new type of keys allows a better scalability, enhances the time of detecting a malicious node and removes the blind rejection problem. Moreover, SEDAN uses a symmetric key shared between any node and its parent to ensure the origin authentication and eliminate the impersonation attack. The establishment of new type of keys in SEDAN is inspired from LEAP.[46] Each node establishes a pair-wise key shared with its one hop and two hops neighborhood using a Transitory Initial Key (TIK) preloaded into each node prior to deployment. The TIK is erased from the memory of each node after establishing the needed keys to prevent node compromising attack. Figure 17.16 illustrates the aggregation process in SEDAN.

17.5.2.4. *SecureDAV*

SecureDAV is based on a hierarchical topology. It ensures authentication and data integrity using elliptic curve cryptography. Each cluster member uses a secure link to send its data to the cluster head. After receiving all members' data by the cluster head, the latter calculates and broadcasts the average data to the cluster members. Each member compares its data with the received average data, if the difference does not exceed a defined threshold, the cluster member sends to the cluster head the average received data signed with a partial secret key. After that, the cluster head sends the average data and the combined received partial signatures to the SINK

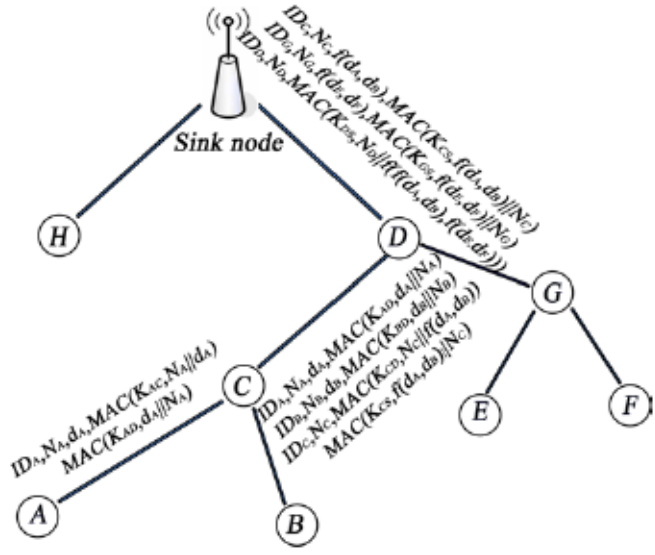
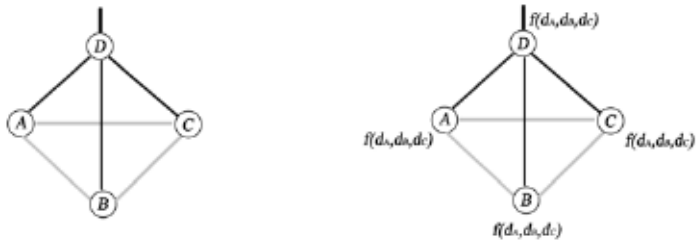


Fig. 17.16. SEDAN aggregation and authentication steps.



Each node sends its data to parent node

Each node calculates the aggregation value to control the behavior of its parent

Fig. 17.17. Aggregation and watch dog mechanism in SDAP.

node. To verify integrity, the base station can query repeatedly the cluster-heads on individual readings.

17.5.2.5. *SDAP*

SDAP uses a clique topology, where each clique is composed of a set of neighbouring nodes that elect one parent node. The parent calculates the aggregation value over all the received members' data and sends it to its parent. Because each member in the same clique can listen to the others, every node uses a watch dog mechanism to calculate the aggregation value in the purpose of controlling the correctness of the data sent by their parent (cf. Figure 17.17).

17.5.3. *Analysis and discussion*

17.5.3.1. *Blind rejection*

Blind rejection is an important parameter to evaluate secure aggregation protocols. A protocol suffering from this kind of problem cannot prevent a bogus data from infecting the global aggregation. Both SEDAN and SDAP, by stopping immediately invalid data during the aggregation process, overcome the blind rejection of the final aggregation value. However, protocols proposed in Refs. 4, 2 and 7 suffer from this phenomenon. Indeed, in these protocols, the verification is done at the sink level, and hence the final aggregation value is rejected after it has been relayed up to the sink.

17.5.3.2. *Impersonation attack*

Impersonation attack is the possibility of launching an attack by injecting false data carrying the source address of another node. If this attack happens, the pretended source will be considered as an intruder and then, will be revoked from the network.

In SAWN, when a node detects an invalid MAC, it must exclude the two downward nodes (child and grandchild) from the sensor network. However, there is no mechanism in SAWN that enables to verify the origin of a packet. This enables an intruder to launch an impersonation attack to remove legitimate nodes from the network. In SEDAN and SecureDAV, the use of the pair-wise key between a node and its upstream allows data origin authentication, and rejects any message coming from unauthenticated nodes. CMT and all end-to-end encryption protocols ruin to a local authentication mechanism, allowing the intruder to execute an impersonation attack. The SDAP protocol is vulnerable to the impersonation attack. Indeed, a malicious node can send a faulty data to one selected parent, using the identity of one chosen member node belonging to the same clique of the selected parent. To launch such an attack, the malicious node must be positioned in the neighbourhood of the parent. In this case, the parent calculates a fault aggregation value and hence will be considered as a malicious node by its child members.

17.5.3.3. *Scalability*

Depending on the mechanism used to secure data aggregation, protocols[2,4,5,7] react differently when the network size increases. The revelation of keys in SAWN is based on the assumption that the SINK node can reach all sensor nodes in the network, using only one hop broadcast. When the network size increases, this assumption will be hardly verified. Therefore, SAWN does not scale well with large networks. SecureDAV, which assumes that cluster heads send the aggregation value through only one hop to the SINK node, does not satisfy also scalability requirements. Protocols SEDAN and SDAP, however, are scalable because they rely on a distributed verification mechanism and do not make any reference to the SINK node. Furthermore, CMT that is similar to a simple aggregation process offers a best scalability too.

17.5.3.4. *Localization*

When detecting faulty data aggregation values, it is important to drop them, and also localize the compromised node to revoke it from the network. The protocols based on end-to-end encrypted data suffer from the lack of localization of the intruder, since the sink receives and verifies only the final result. However, the localization of malicious nodes in the protocols based on hop-by-hop encryption is possible because intermediate nodes have access to payload data and thus can detect the malicious nodes that falsify the aggregation.

17.5.3.5. *Resilience against aggregator node capture*

In any aggregation protocol, it is very important to verify the behaviour of aggregator nodes. A compromised aggregator node can falsify the aggregation value by rejecting the received data value from its children or simply modifying it. In the first case, all protocols, except CMT, prevent such attacks by using a simple watchdog mechanism. In the second case, SEDAN and SAWN under the assumption that two consecutive nodes cannot be compromised and by employing the two hops verification mechanism, detect any modification tentative at the parent node level. SDAP by using a watchdog mechanism in the same clique can detect this kind of attacks if the number of compromised nodes is smaller than $n/2$, with n being the number of nodes in a given clique. In SecureDAV, the aggregation values sent by each cluster head are verified by the sink node. Each cluster head sends, in addition to the mean of the received data, the mean data signatures of some of its cluster members. However, detecting the cluster head compromise cannot be guaranteed if some of the cluster members are also compromised. CMT uses the PH encryption in the purpose of detecting any faulty aggregation value. However, authors in Ref. 8 show that it is possible for an attacker to alter the encrypted aggregation value without knowledge of the plaintext, which forbids the detection of an existing compromised aggregator node.

17.6. Securing Channel Access

The wireless communication medium and the limited resources of sensor networks generate a set of vulnerabilities that make them prone to various attacks both at the physical and link layers. The wireless communication medium opens the door to jamming-style DoS attacks and the limited resources to energy exhaustion ones.

17.6.1. *Jamming attacks*

The main goal of a jammer is to break down communication links to prevent sensor nodes from exchanging information. A jammer can impede communication either by preventing nodes from accessing the channel or by letting them access the

channel but corrupting the transmitted messages so that they cannot be successfully received.

Depending on its knowledge on the network, the jammer may operate more or less efficiently. If the jammer does not know the technique used in the physical transmission, it can only jam by continuously transmitting strong noise in the suspected communication band to create noise or to cause interference. The presence of noise blocks nodes using CSMA-based channel access methods from accessing the channel as it continuously appears busy to them. With a continuous noise transmission, even TDMA-based channel access methods suffers from communication jamming as the transmitted frames suffer from interferences and thus end up in collisions.

When the jammer has more information on the network it can carry out more efficient jamming. The key idea behind such a clever jamming is to achieve targeted attacks at critical instants, such as causing deliberate collisions, while running at a low duty cycle to save energy. The paper[1] describes a number of energy-efficient jamming techniques that can be used with a large set of well-known link protocols for sensor networks.

The main counter-measures proposed to combat jamming attacks can be classified in two approaches: (i) avoidance or (ii) detection and evasion as shown in Figure 17.18.

17.6.1.1. *Jamming avoidance*

The jamming avoidance approach proposes to use robust communication techniques to compete against both intentional jamming and accidental noise.

Among these schemes, we can cite, for example, the use of Spread Spectrum with a pseudo-random sequence such as Time Hopping that is envisaged for the promising low power UWB transceivers. The use of Spread Spectrum communication can be efficient only if the pseudo-random sequence is kept secret and is long enough to be difficultly detectable. In addition to Spread Spectrum communications, sensor nodes can use directional antennas to minimize omni-directional noise and reduce the jamming-to-signal ratio at the receiver. The main issue with omni-directional antennas is that upper communication layers should adapt to maintain network connectivity. Other techniques such as power control, error correcting codes and lowering the information rate can also be used to increase the rate of successfully received messages at the receivers.

17.6.1.2. *Jamming detection and evasion*

The second counter-measure to jamming contains two parts: detection and evasion. While detection is common to many protocols and naturally executed at the lower layers (i.e. physical and link), evasion may take place even at the upper layers such as the routing layer. At the physical layer, nodes can execute a channel surfing procedure[13] upon the detection of jamming in the communication channel.

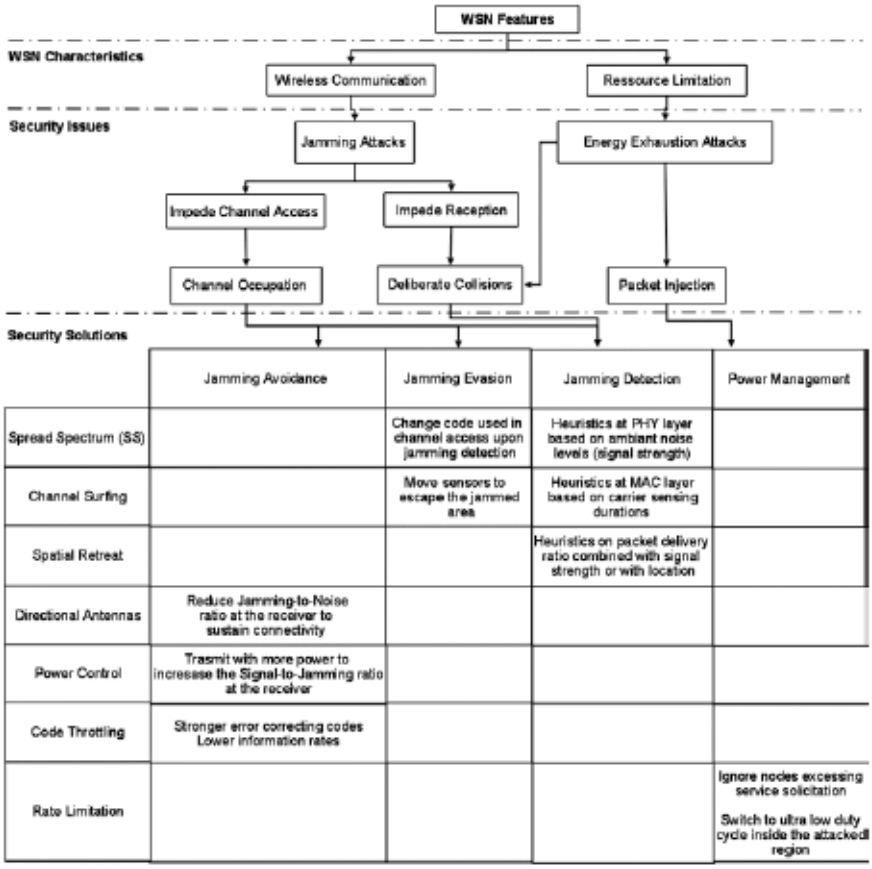


Fig. 17.18. Securing the access to radio channels in WSN: A taxonomy.

In channel surfing, nodes switch to another orthogonal channel to escape jamming. At the upper layers, nodes aim at circumnavigating the jammed region. They can, for example, map the jammed region[14] and route messages around it or physically move to escape jamming.[d]

Jamming detection techniques are based on heuristics. To detect a jammed channel, a node gathers information about channel state in normal situations and uses it as reference to detect jamming situations. The most common detection techniques are those based on signal strength levels and carrier sense durations. For signal strength, a node builds a statistical model describing normal energy levels in the network by gathering measurements on signal strength level during a given time interval then it uses the established model to determine whether a situation is jamming or not. Another way to detect jamming situations is to use carrier sense

[d]Note that the latter is particularity valid for sensor and actuator networks.

duration. By building a statistical model describing the distribution of carrier sense durations a node can determine whether the channel is jammed or not. Jamming can also be detected by building a statistical model for the PDR (Packet Delivery Ratio) and using it to detect pathological situations. Both of the receiver and the transmitter can measure the PDR. The receiver measures the PDR by calculating the ratio of the number of received packets that pass the CRC check to the number of all received packets. However, the transmitter determines the PDR by calculating the ratio of the number of received acknowledgments to the number of transmitted packets. While the PDR can be useful for receivers, it may be inefficient for transmitters to detect an intelligent jammer that only corrupts broadcast messages. To cope with this, SIS (Secure Implicit Sampling)[15] makes it possible for a broadcasting base station to probabilistically detect the failure to receive its broadcast, even if the attacker is intended to insert this attack to be undetectable. By soliciting authenticated acknowledgments from a subset of nodes per broadcast, the subset is unpredictable to the attacker and tunable to avoid acknowledgments implosion at the base station.

The main issue with these methods is that they can lead to false positive situations such as declaring a congestion situation to be a jamming situation. To enhance jamming detection, multimodal detection, in which detection is based on multiple criteria such as combining packet delivery ratio with signal strength or location, is used. For example, if a node detects a low packet delivery ratio then it suspects a jamming situation. The low packet delivery ratio should correspond to weak signal strength or a transmission from a far node to weaken the hypothesis of jamming.

17.6.2. *Energy exhaustion attacks*

In addition to communication jamming, the limited energy resources of sensor nodes make them prone to another type of attacks such as energy exhaustion. This type of attacks may have two scopes: network-wide or local. In contrast to network wide attacks, an attacker carrying out a localized attack aims at exhausting the energy of neighboring nodes. Although such an attack has only localized issues, it may be very dangerous and hard to cope with. This attack is particularly disastrous if the attacker is intending to exhaust critical nodes. Moreover, finding a counter measure for such an attack can be difficult as the node should know at the link layer whether the message it is receiving could be classified as an attack. The only existing solution to those attacks is rate limitation[16] in which a node decides to ignore messages from blacklisted nodes.

17.7. Security Against Sensor Node Compromise

Once deployed, WSN are expected to run without human attendance. This makes nodes highly vulnerable to compromising and physical tampering. If an adversary

captures a node, it can easily undertake a large number of attacks exploiting possible hardware or software implementation shortcomings. Preventing such attacks mandates the use of tamper-resistance mechanisms addressing two principal security issues: hardware and software security.

17.7.1. *Hardware security*

In this category, attacks deal with nodes hardware by exploiting the physical-side of implementation flaws. Two types of attacks are distinguished: invasive and non-invasive attacks. Invasive attacks, like micro-probing techniques, require access to the chip-level components of a node in order to observe, manipulate or interfere with the system internals. In contrast, non-invasive attacks do not need to have physical access to a node. For example: Side-channel attacks may allow to break a cryptographic algorithm by simply observing the node's power consumption or its timing profile, which reveals some sensitive information about the executed cryptographic primitives, or the used secret keys. The study presented in Ref. 32 shows how a simple differential power analysis can achieve side-channel attacks on MAC.

Although both types of physical attacks represent a significant threat for sensor networks, few works have been proposed to tackle them. To cope with invasive attacks, we believe that the FIPS 140-2 requirements should be applied to WSN to increase the levels of nodes tamper-resistance. For example, the use of zeroization circuitry that immediately zeroize all plain text secrets and private keys located in memory, when a valid or invalid access is detected.

For non-invasive attacks such as side-channel attacks, future research should focus on mitigating the symptoms that allow the leak of the system's side-channel information like power dissipation, timing and electromagnetic radiations. Possible solutions include randomizing the clock signal or the instruction execution sequence, introducing dummy instructions, balancing Hamming weights of the internal data and bit splitting.

17.7.2. *Software security*

In this category, attacks aim at compromising the software running on nodes by exploiting known vulnerabilities. This kind of attacks is generally based on malicious software, like viruses and worms, and can be based on various techniques. In interception-based software attacks, the goal is to passively eavesdrop sensitive data. An attacker may make use of a logical analyzer to probe the inner lines of a node. Interruption-based attacks target destabilizing the software to disrupt the system availability. This kind of DoS attacks is facilitated due to nodes limited resources (energy, computation and memory). Finally, modification-based attacks, such as buffer overflow, modify the software code to compromise its integrity. Current WSN are considerably vulnerable to software attacks. For instance, TinyOS, a widely used operating system for sensor networks, does not provide any user/process

access control to system resources. Moreover, its serial forwarder component allows an adversary to open a port to a node without any authentication mechanism. This greatly facilitates uploading malicious code or downloading sensitive information from the sensor.

We believe that research efforts should be conducted in order to adapt software security issues like secure OS bootstrapping, or secure software design and coding, to WSN software. Techniques such as software authentication and validation using remote software-based attestation,[33] the use of restricted environment such as Java Virtual Machine (JVM) to execute untrusted code, and the use of encryption wrappers to allow dynamic run-time software encryption/decryption should also be adapted for WSN.

17.8. Challenges for Future Research

The unsuitability of asymmetric cryptography has driven research in key management to symmetric cryptography and thus to key pre-distribution methods. We believe that this situation may change in the future and low overhead asymmetric cryptography methods based on Elliptic Curve Cryptography (ECC)[72] will aid to solve many challenges relating to key management and authentication.

The absence of a hardware protection to prevent tampering, expose sensor nodes to a physical compromising by an adversary. The latter can extract the key materials contained in a deployed node and launch multiple attacks. To relax the consequences of such vulnerability, authors in Ref. 62 define two properties that must be verified in the design of a tolerant key management scheme: (1) *the opaqueness property* — an adversary cannot deduce most of the keys being used in the network by compromising a small number of sensor nodes. (2) *The inoculation property* — an adversary cannot aid unauthorized sensor node to successfully join a network by compromising a small number of sensor nodes. Thus future key management schemes should tolerate the compromise of some sensors while maintaining a safe operation of the whole network.

So far, existing secure routing solutions have been focused mainly on ensuring a correct routing state for every sensor. By analyzing Figure 17.18, we remark clearly that even the best secure routing topology in terms of performance is based on the hop count metric. Nevertheless, WSN communication mechanisms should rather be based on more meaningful properties that address real problems. By choosing appropriate metrics, the resulting topology should be more "profitable" aiming at increasing the survivability and the usability of the network. To achieve this, future protocols should devise secure ways for building paths using sensor and environment oriented metrics, such as energy and link reliability.

For the data relay problem, there are still some important problems remaining unresolved by WSN security community. Existing solutions that provide countermeasures against the false reports generated by compromised source nodes are

based on data comparison by a controller node (such as a cluster-head or a Center of Stimulation). Consequently, this method can only be applicable if the controller node can collect a certain number of samples from the same sensing region. However, sparse networks do not have such correlation between readings and collected data may vary dramatically. For instance, an event detection network with sparse nodes can observe an event only from one node, and data comparison method cannot be used in this scenario. Therefore, a compromised node can easily cheat in its reports with a low probability for being detected.

The use of efficient cryptographic techniques to guarantee integrity for aggregated data should be reinforced with lightweight statistical methods to detect faulty and corrupt data. Indeed, while cryptographic mechanisms guarantee that data is transmitted without modification, there is no mean to detect faulty data originating from a corrupt node using cryptographic techniques.

The most efficient way to cope with jammers in wireless systems is hiding the communication channel by using secret pseudo-random spread spectrum codes. Although this technique, largely used in military communication, is efficient, it may not apply to most of the wireless sensor networks. Current applications envisaged for sensor networks are built upon open standards to ensure inter-operability between different manufacturers product. For example, in Zigbee Alliance,[17] the wireless communication is based on the IEEE 802.15.4[18] open standard physical specifications. Therefore, it is very vulnerable to intelligent jamming at the physical layer. In addition, even if the physical and access layer specifications are not kept secret at the instant of deployment, they can be discovered by means of traffic analyzing or node capture. Therefore, securing channel access should include jamming detection techniques that should be able to efficiently map the jamming region. When the jamming region is detected, nodes need to either evade and/or compete with the jammers. In the evasion techniques such as channel surfing or spatial retreat, most challenges are traditional and related to distributed computing. Nodes should cooperate efficiently to provide low-latency, convergent and scalable solutions. For example, when nodes physically move to evade the jammed region, they should move efficiently to generate a new advantageous topology maintaining routes between communicating nodes. The same applies for channel surfing. In addition, in channel surfing, it should be determined whether all the nodes or only the subset of them that are affected by jammers should switch to another channel. In both case, coordination is needed to maintain connectivity throughout the network.

To realize their full potential, sensor networks require connectivity to the Internet. One benefit in connecting sensor networks to the Internet using IPv6, is to take advantages of the huge (128-bit) address space of IPv6.[61] Preparing sensor networks for IP communication and integrating them into the Internet, however, requires certain features and specification to work, for example, in the adaptation of the respective link technology, specification of ad hoc networking, handling

the security issues, and auto-configuration to support ad hoc deployment. Security is one major concern in every part of the Internet, covering areas like encryption, detection of intrusion, access control, authentication, authorization, integrity protection, prevention of denial of service etc. In principle, in IP-enabled sensor networks standard security mechanisms based on IP could be applied. However, especially sensor networks are resource constraint concerning processing power and network bandwidth, putting limits on security. Therefore, new lightweight security mechanisms appropriate for sensor networks have to be used. On the other hand, adaptation of the existing solutions to take advantage of available IPv6 services with regard to security, auto-configuration and mobility management will be required.

Most of the proposed security mechanisms are designed for static networks with flat organization. However, several applications require using mobile sensors/actuators to save energy and to increase the connectivity of the network. Future security mechanisms should take into consideration the existence of mobile components in the network. This will have necessarily an impact on route construction and updates, authentication mechanisms of mobile actuators, and data dissemination which would be mostly delivered through one hop transmissions.

17.9. Conclusions

In this paper, we provided a comprehensive taxonomy of security attacks in sensor networks and their corresponding solutions. Through the survey of existing work, we noticed that research on security issues have matured over the years. However, there remain several open problems that need to be resolved to make WSN secure to the extent required by their applications. We noted that secure data dissemination is an important area in WSN security as it addresses a core service in sensor networks, namely routing protocols. The main lesson learned in this field, is that conventional solutions, early proposed for ad hoc networks are not applicable. Moreover, secure routing for specific classes of sensor networks, such as sensor-actor networks or underwater sensor networks, remains highly unexplored. Routing services in such networks differs in terms of node mobility, and traffic patterns. Hence, adequate schemes need to be developed in order to guarantee their security. Concerning data aggregation security, most of the proposed solutions are still exposed to one or more security threats. Proposing more complete solutions, based on a fully distributed approach, constitute an attractive research field. Finally, we remarked that few works have been devoted to address security issues in link and physical layers. Similarly, the domain of physical and software security of the sensor mote remains in its infancy, and an additional endeavour is mandatory to prevent sensor node compromise, which represents a significant threat for the whole mission of the network.

Problems

1. Why asymmetric cryptography can hardly be used in securing WSN? Is there a specific asymmetric cryptosystem that can be used to secure WSN? Which one, and why?
2. In probabilistic key pre-distribution scheme, each sensor is preloaded with a key ring composed of m keys randomly picked from a huge key pool. Explain why the resiliency of such a scheme is inversely proportional to the key ring size (m)?
3. Explain why SAWN and SecureDAV do not scale to very large networks, while SEDAN and SDAP scale better?
4. Explain how SEDAN and SecureDAV allow authenticating data origin? And explain why SAWN is vulnerable to impersonation attacks?
5. How can an intruder exclude/revoke a valid sensor from a network using SDAP for data aggregation?
6. Is it always possible to localize the node responsible for injecting faulty aggregation value in a WSN? Explain.
7. In SeRINS, is it possible that a legitimate node chooses an intruder as a main parent? If yes, explain what could be the impact on the security of the protocol.
8. EINSENS uses one hop broadcast keys to authenticate one hop communications during the relay of the RREQ. Explain why these types of keys are not suitable for such communications, and give an example of a possible attack.
9. In your opinion, what is the best layer for implementing an efficient solution against energy exhaustion attacks? Why most of the proposed solutions are inefficient?

Bibliography

1. J. Domingo-Ferrer, A provably secure additive and multiplicative privacy homomorphism, *ISC '02: Proceedings of the 5th International Conference on Information Security* (2002).
2. C. Castelluccia, E. Mykletun and G. Tsudik, Effcient aggregation of encrypted data, wireless sensor networks, *MobiQuitous, IEEE Computer Society* (2005) 109–117.
3. E. Mykletun, J. Girao and D. Westhoff, Public key based cryptoschemes for data concealment in wireless sensor networks, *IEEE International Conference on Communications (ICC2006)* (2006).
4. L. Hu and D. Evans, Secure aggregation for wireless networks, *Workshop on Security and Assurance in Ad Hoc Networks* (January 2003).
5. M. Bagaa, N. Lasla, A. Ouadjaout and Y. Challal, SEDAN: secure and efficient protocol for data aggregation in wireless sensor networks, *32nd IEEE Conference on Local Computer Networks (LCN 2007), Workshop on Network Security*, pp. 1053–1060.
6. K. Wua, D. Dreefa, B. Sunb and Y. Xiao, Secure data aggregation without persistent cryptographic operations in wireless sensor networks, *Ad Hoc Networks* **5**(1) (2006) 100–111.
7. A. Mahimkar and T. S. Rappaport, *SecureDAV: A Secure Data Aggregation and Verification Protocol for Sensor Networks* (2004).

8. S. Peter, K. Piotrowski and P. Langendoerfer, On concealed data aggregation for wireless sensor networks, *IEEE CCNC 2007*, Las Vegas Nevada, USA (January 2007).
9. H. Chan, A. Perrig and D. Song, Secure hierarchical in-network aggregation in sensor networks, *Proceedings of the ACM Conference on Computer and Communications Security*, New York, NY, USA (2006) 278–287.
10. H. O. Sanli, S. Ozdemir and H. Cam, SRDA: secure reference-based data aggregation protocol for wireless sensor networks, *Proceedings of the VTC Fall 2004 Conference*, Los Angeles, CA, USA (2004).
11. Y. Sang and H. Shen. Secure data aggregation in wireless sensor networks: a survey, *7th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06)* (2006) 315–320.
12. Y. Law, L. Van Hoesel, J. Doumen, P. Hartel and P. Havinga, Energy-efficient link-layer jamming attacks against three wireless sensor network MAC protocols, *Proceedings of ACM SASN*, Alexandria, Virginia (November 2005).
13. W. Xu, K. Ma, W. Trappe and Y. Zhang, Jamming sensor networks: attack and defense strategies, *IEEE Network* **20**(3) (2006) 41–47.
14. A. Wood, J. Stankovic and S. Son, *JAM: A Mapping Service for Jammed Regions in Sensor Networks*, RTSS, Cancun, Mexico (December 2003).
15. J. McCune, E. Shi, A. Perrig and M. Reiter, Detection of denial-of-message attacks on sensor network broadcasts, *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA (May 2005).
16. A. Wood and J. Stankovic, Denial of service in sensor networks, *IEEE Computer* (October 2002) 48–56.
17. www.zigbee.org.
18. IEEE 802.15.4, *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)* (2003).
19. W. Dai, *Comparison of Popular Cryptographic Algorithms*, http://www.eskimo.com/\ ∼weidai/benchmarks.html (2003).
20. R. Shirey, Internet security glossary, *RFC 2828* (May 2000).
21. A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, *Hand book of Applied Cryptography* (CRC Press, 1996).
22. B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C* (1996).
23. C. Kaufman, R. Perlman and M. Speciner, *Network Security: Private Communication in a Public World*, Printice Hall Series in Computer Networking and Distributed Systems (2002).
24. B. Kaliski, The MD2 message-digest algorithm, *RFC 1319* (1992).
25. R. Rivest, The MD5 message-digest algorithm, *RFC 1321* (1992).
26. D. Eastlake and P. Jones, US secure hash algorithm 1 (SHA1), *RFC 3174* (2001).
27. H. Krawczyk, M. Bellare and R. Canetti, HMAC: keyed-hashing for message authentication, *RFC 2104* (1997).
28. Federal Information Processing Standards Publication, Data encryption standard (DES), *FIPS PUB 46* (1993).
29. Federal Information Processing Standards Publication, Advanced encryption standard (AES), *FIPS PUB 197* (2001).
30. R. L. Rivest, A. Shamir and L. M. Adelman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM* **21**(2) (1978) 120–126.
31. Federal Information Processing Standards Publication, Digital signature standard (DSS), *FIPS PUB 186* (1994).

32. K. Okeya and T. Iwata, Side channel attacks on message authentication codes, *ESAS 2005 European Workshop on Security and Privacy in Ad Hoc and Sensor Networks* (2005).
33. M. Shanek, K. Mahadevan, V. Kher and Y. Kim, Remote software-based attestation for sensor networks, *European Workshop on Security and Privacy in Ad Hoc and Sensor Networks* (2005).
34. A. S. Wander *et al.*, Energy analysis of public-key cryptography for wireless sensor networks, *PerCom '05: Proceedings of the 3rd IEEE International Conference of Pervasive Computing and Communications* (2005).
35. N. Gura *et al.*, Comparing elliptic curve cryptography and RSA on 8-bit CPUs, *6th International Workshop on Cryptographic Hardware and Embedded Systems*, Boston, Massachusetts (2004).
36. G. Gaubatz, *et al.*, Public key cryptography in sensor networks-revisited, *ESAS '04: 1st European Workshop Security in Ad-Hoc and Sensor Networks* (2004).
37. K. Piotrowski *et al.*, How public key cryptography influences wireless sensor node lifetime, *SASN'06*, Alexandria, Virginia, USA (2006).
38. A. Liu and P. Ning, *TinyECC: Elliptic Curve Cryptography for Sensor Networks (Version 0.3)* (February 2007), available at http://discovery.csc.ncsu.edu/software/TinyECC/.
39. D. J. Malan, M. Welsh and M. D. Smith, A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography, *Proceedings of the 1st IEEE International Conference of Sensor and Ad Hoc Communications and Networks*, Santa Clara, CA (2004).
40. H. Wang, B. Sheng and Q. Li, Elliptic curve cryptography-based access control in sensor networks, *International Journal of Security and Networks* **1**(3/4) (2006).
41. L. Eschenauer and V. D. Gligor, A key-management scheme for distributed sensor networks, *Proceedings of the 9th ACM Conference on Computer and Communications Security* (2002).
42. H. Chan, A. Perrig and D. Song, Random key predistribution schemes for sensor networks, *IEEE Symposium on Security and Privacy*, Berkeley, CA (11–14 May, 2003) 197–213.
43. W. Du, J. Deng, Y. S. Han, S. Chen and P. K. Varshney, A key management scheme for wireless sensor networks using deployment knowledge, *Proceedings of IEEE INFOCOM'04* (IEEE Press, Hong Kong, 2004) 586–597.
44. C. Blundo *et al.*, Perfectly-secure key distribution for dynamic conferences, *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology* (Spring-Verlag, Berlin, 1992) 471–486.
45. Z. Yu and Y. Guan, A robust group-based key management scheme for wireless sensor networks, *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2005)*, New Orleans, LA (USA, IEEE Press, 2005) 13–17.
46. S. Zhu, S. Setia and S. Jajodia, LEAP: efficient security mechanisms for large-scale distributed sensor networks, *ACM Conference on Computer and Communications Security (CCS '03)* (2003).
47. X. Du *et al.*, An effective key management scheme for heterogeneous sensor networks, *Security Issues in Sensor and Ad Hoc Networks* (2007) 24–34.
48. X. Du *et al.*, An improved key distribution mechanism for large-scale hierarchical wireless sensor networks, *Security Issues in Sensor and Ad Hoc Networks* (2007) 35–48.
49. H. Fu *et al.*, Replication attack on random key pre-distribution schemes for wireless sensor networks, *IEEE Workshop on Information Assurance and Security*, United States Military Academy, West Point, NY (2005).

50. R. Blom, An optimal class of symmetric key generation systems, *Proceedings of the Eurocrypt 84 Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques* (Springer Verlag, 1985) 335–338.
51. J. Deng, C. Hartung, R. Han and S. Mishra, A practical study of transitory master key establishment for wireless sensor networks, *Proceedings of the 1st IEEE International Conference of Security and Privacy for Emerging Areas in Communication Networks (SecureComm '05)* (2005).
52. J. Deng, R. Han and S. Mishra, Intrusion-tolerant routing for wireless sensor networks, *Computer Communications* **29**(2) (2006) 216–230.
53. J. Yin and S. Madria, Secrout: a secure routing protocol for sensor networks, *AINA 06: Proceedings of the 20th International Conference on Advanced Information Networking and Applications*, Washington, DC, USA, IEEE Computer Society **1** (2006) 393–398.
54. S. Lee and Y. Choi, A secure alternate path routing in sensor networks, *Computer Communications* **30**(1) (2006) 153–165.
55. J. Deng, R. Han and S. Mishra, Defending against path-based DoS attacks in wireless sensor networks, *3rd ACM Workshop on Security of Ad Hoc Sensor Networks*, New York, USA (2005) 89–96.
56. S. Zhu, S. Setia, S. Jajodia and P. Ning, Interleaved hop-by-hop authentication against false data injection attacks in sensor networks, *ACM Transactions on Sensor Networks* **3**(3) (2007) 14.
57. F. Ye, H. Luo, S. Lu and L. Zhang, Statistical en-route filtering of injected false data in sensor networks, *IEEE INFOCOM'04* **4** (2004) 2446–2457.
58. L. Lamport, Password authentication with insecure communication, *Communications of the ACM* **24**(11) (November 1981) 770–772.
59. D. Liu and P. Ning, Establishing pairwise keys in distributed sensor networks, *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, Washington DC (2003).
60. A. Ouadjaout, Y. Challal, N. Lasla and M. Bagaa, SEIF: secure and efficient Intrusion-Fault tolerant routing protocol for wireless sensor networks, *IEEE-ARES* (2008).
61. K. Das, *IPv6 and Wireless Sensor Networks*, http://www.ipv6.com/articles/sensors/IPv6-Sensor-Networks.htm.
62. J. Deng, C. Hartung, R. Han and S. Mishra, A practical study of transitory master key establishment for wireless sensor networks, *Proceedings of the First IEEE International Conference of Security and Privacy for Emerging Areas in Communication Networks (SecureComm '05)* (2005).
63. W. Lou and Y. Kwon, H-spread: a hybrid multipath scheme for secure and reliable data collection in wireless sensor networks, *IEEE Transactions on Vehicular Technology* **55**(4) (2006) 1320–1330.
64. G. Jolly, M. C. Kuscu, P. Kokate, and M. Younis, A low-energy key management protocol for wireless sensor networks, *IEEE Symposium on Computers and Communications (ISCC'03)*, Kemer, Antalya, Turkey (30 June–3 July, 2003).
65. H. Chan and A. Perrig, PIKE: peer intermediaries for key establishment in sensor networks, *INFOCOM* (2005).
66. D. Liu and P. Ning, Establishing pairwise keys in distributed sensor networks, *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, Washington DC (ACM Press, 2003) 41–47.
67. D. Liu and P. Ning, Improving key pre-distribution with deployment knowledge in static sensor networks, *ACM Transactions on Sensor Networks* **1**(2) (2005) 204–239.
68. Z. Yu and Y. Guan, A robust group-based key management scheme for wireless sensor networks, *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2005)*, New Orleans, LA (USA, IEEE Press, 2005) 13–17.

69. D. Huang, M. Mehta, D. Medhi and L. Harn, Location-aware key management scheme for wireless sensor networks, *Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04)*, Washington DC (USA, ACM Press, 2004) 29–42.
70. J. Lee and D R. Stinson, Deterministic key predistribution schemes for distributed sensor networks, *Proceedings of ACM Symposium on Applied Computing 2004*, Waterloo, Canada, Lecture Notes in Computer Science, Vol. 3357 (Springer, 2004), 294–307.
71. Y. Cheng and D. P. Agrawal, Efficient pairwise key establishment and management in static wireless sensor networks, *Proceedings of the 2nd IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS'05)* (November 2005), 544–550.
72. I. F. Blake, G. Seroussi and N. P. Smart, Advances in elliptic curve cryptography, *London Mathematical Society Lecture Note Series*, No. 317 (2005).
73. A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen and D. E. Culler, Spins: security protocols for sensor networks, *Wireless Networks* **8**(5) (2002) 521–534.
74. M. Passing and F. Dressler, Experimental performance evaluation of cryptographic algorithms on sensor nodes, *3rd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (IEEE MASS 2006)* 882–887.
75. K. J. Choi and J-I. Song, Investigation of feasible cryptographic algorithms for wireless sensor network, *Proceedings of the 8th International Conference on Advanced Communication Technology (ICACT 2006)* (2006).
76. K. Piotrowski, P. Langendoerfer and S. Peter, How public key cryptography influences wireless sensor node lifetime, *Proceedings of the 4th ACM Workshop on Security of Ad Hoc and Sensor Networks, SASN'06*, ACM, New York, NY (2006).

This page intentionally left blank