SECURITY MANAGEMENT

III

Chapter 15

Key Management in Wireless Sensor Networks: Challenges and Solutions

Yun Zhou, Yanchao Zhang, and Yuguang Fang[1]

Contents

[1] This work was supported in part by the U.S. Office of Naval Research under Young Investigator Award N000140210464 and under Grant N000140210554.

15.1 Introduction

Wireless sensor networks (WSNs) [1] have been seen as a promising network infrastructure for many military applications, such as battlefield surveillance and homeland security monitoring. In those hostile tactical scenarios and important commercial applications, security mechanisms are necessary to protect WSNs from malicious attacks.

Key management is very critical to security protocols because most of the cryptographical primitives, such as encryption and authentication, in those protocols are based on the operations involving keys. Encryption requires that a key be fed into an algorithm so that plaintexts can be transformed into ciphertexts. To authenticate a packet, a *Message Authentication Code* (MAC) can be attached to the packet. However, MACs are usually computed by hashing the concatenations of packets and keys. Hence, key

management is of paramount importance for establishing security infrastructures for WSNs.

However, to establish keys is a very challenging task in WSNs due to their unique characteristics, which are different from conventional wired networks such as the Internet and other wireless networks such as *mobile ad-hoc networks* (MANETs). The openness of wireless channels renders adversaries' capability to analyze the eavesdropped packets transmitted between sensor nodes such that some key information can be exposed, from which adversaries can derive keys between sensor nodes. A sensor node is usually built with constrained resources in terms of memory, radio bandwidth, processing capability, and battery power. Strong security algorithms may not be supported by sensor platforms due to their complexity. In a hostile environment, it is infeasible to provide constant surveillance on a WSN after deployment, and sensor nodes can be captured so that all their keying materials are compromised. A WSN may have to scale up to thousands of sensor nodes, which demands simple, flexible, and scalable security protocols. However, to design such security protocols is not an easy task. Higher-level security and less computation and communication overhead are contradictory requirements in the design of security protocols for WSNs. In most cases, a trade-off must be made between security and performance.

A general approach to establishing keys in WSNs includes two steps. Before sensor nodes are deployed, each node is configured with some key materials. After those nodes are deployed into a designated terrain, they perform several rounds of communications to agree on pairwise keys associated with their key materials. Based on the algorithms used to establish pairwise keys, current solutions can be classified into symmetric key schemes and asymmetric key (or public key) schemes. The symmetric key schemes of the early stage are probabilistic, in that two nodes can use their key materials to establish a pairwise key with a certain probability. Therefore, some pairs of nodes can establish keys directly and some pairs of nodes have to establish pairwise keys through multi-hop paths. However, their probabilistic nature makes some nodes isolated, in that they are not be able to establish keys with their neighboring nodes. Then, a deterministic approach is proposed so that every pair of nodes can establish a pairwise key directly or through a multihop path. The probabilistic approach and the deterministic approach uniformly distribute key materials, and thus each node can establish direct keys with a small portion of its neighbors and must establish indirect keys with other neighbors through a multi-hop path, which may cost large power consumption. Then, location infomation is used so that nodes close to each other are configured with correlated key materials. This location-based approach can increase the probability that each pair of neighboring nodes establishes a pairwise key, thus saving energy consumption on multi-hop routing. Public key schemes

are mainly based on Diffie-Hellman and RSA. However, algorithms in the field of elliptic curves have drawn much attention recently because of their efficiency.

In this chapter, some design issues and challenges are first introduced in Section 15.2. Then, symmetric key schemes are described in Section 15.3, including probabilistic, deterministic, and location-based solutions. Public key schemes are discussed in Section 15.4. Section 15.5 sheds light on some open issues. Finally, the chapter concludes with Section 15.6.

15.2 Design Issues and Challenges

15.2.1 Cryptographical Issues

In his classic paper "Communication Theory of Secrecy Systems" [2], Shannon, who had established information theory, developed the theoretical framework for *symmetric key*-based crytography. In his cryptographical system model, there are two information sources (i.e., a message source and a key source) at the transmission end. The key source produces a particular key K from among those that are usable in the system. This key K is transmitted by some means, supposedly *not interceptible*, for example by a messenger, to the receiving end. The message source produces a message M (in the "clear"), which is enciphered by the encipherer T_K. The resulting ciphertext E is sent to the receiving end by possibly *interceptible* means, for example radio. At the receiving end, the ciphertext E and the key K are combined in the decipherer T_K^{-1} to recover the message M. The transformation T_K and its inverse T_K^{-1} are possibly known to the public.

The Diffie-Hellman [3] and RSA [4] algorithms mark the establishment of *asymmetric key*-based cryptography. Unlike a single key used in symmetric key systems, there are two keys in asymmetric key systems. The transmission end encrypts a message M into a ciphertext E by an encryption key K. The receiving end decrypts the ciphertext E to get the message M by a decryption key K^{-1}. Here, the encryption key K and the decryption key K^{-1} are different. Although the decryption key is kept secret, the encryption key is usually known to the public. Asymmetric key systems, therefore, are also called *public key* systems.

In a cryptographical system, the message source and the ciphertext space are usually accessible to an attacker. The encryption and the decryption transforms are also seen as accessible to the attacker. Although in some specific systems the cryptographical algorithms can be kept secret, this approach may increase system vulnerability because an algorithm that is not inspected carefully by critical experts may have some potential defects that can be utilized by hackers. Therefore, most "secure" algorithms are public so that they can be carefully inspected. In this case, the

security of the entire system primarily relies on the secrecy of the keys it uses.

If an attacker can find the key, the entire system is broken. The attaker can achieve this goal by cryptanalysis. Most cryptographical systems are vulnerable to cryptanalysis due to the existence of the redundancy of message source in the real world. The redundancy can always provide the attacker with a possible tool to do cryptanalysis over intercepted ciphertexts during their transmission. Moreover, the attacker knows the system being used, that is, the message space, the transformation T_i, and the probabilities of choosing various keys, and has unlimited time and manpower available for the analysis of ciphertexts. The attacker thus can use all these resources to find the key if time is not important to him. Another way is to directly intercept the key during its transmission between the message source and receiving end. Therefore, how to securely achieve key agreement between the source and sink is a very important issue.

Generally, establishing keys involves two steps. First, the source and sink should be configured with some key materials. Second, those materials are used to establish a shared symmetric key between the source and sink. In symmetric key systems, those key materials can be the shared symmetric key or parameters used to calculate a symmetric key. In asymmetric key systems, they are parameters associated with the chosen asymmetric key algorithm (e.g., Diffie-Hellman or RSA), and the source and sink can negotiate a shared symmetric key using the asymmetric key algorithm. After those nodes are deployed into a designated terrain, they perform several rounds of communications to agree on pairwise keys associated with their key materials.

15.2.2 Challenges

Although the key management problem has been investigated thoroughly in conventional wired networks such as the Internet and wireless networks such as cellular networks, WLANs, or ad hoc networks, the exising solutions can hardly be transplanted into WSNs due to their unique characteristics,

In wired networks, the key materials transmitted over shielded wired lines during the negotiation phase between the source and sink are more difficult to intercept. But a wireless channel is open to eavesdroppers. In addition to eavesdropping key materials to expose corresponding keys, adversaries can also intercept the encrypted ciphertexts so that adversaries can analyze the eavesdropped packets to get some key information, from which adversaries can derive keys between sensor nodes.

In cellular networks and WLANs, the communication pattern is one-hop, that is, between the base station or the access point and the mobile node; but in WSNs, all the nodes are involved into multi-hop communications. Most centralized secure protocols cannot be directly applied in distributed

WSNs. Although ad hoc networks bear more similarities to WSNs, the nodes in ad hoc networks are more powerful than those in and are WSNs, and are thus able to support more secure, more complex protocols.

Moreover, a wireless channel is very dynamic. Key establishment protocols can endure frequent interruptions when channel conditions vary. Although link layer protocols may have some error control mechanisms, the cost of establishing keys is inevitably increased.

The openness of a wireless channel also render adversaries the ability to launch the *denial-of-service* (DoS) attack [5]. Constant or random jamming interferences can be introduced by adversaries to corrupt normal communications, thus leading to the failure of key establishment protocols.

To save cost, sensor nodes are usually built with constrained resources. For example, the latest MICA motes [6] only use 8-bit processors. Their memory size is measured in units of kB (kilobytes). The radio interface can support approximately 40 kbits/sec, The entire mote platform is only powered by 2 AA bateries. The constraned processing capability makes the implementation of strong security algorithms a very challenging task. Considering the power limit, it is impratical to support complex protocols if a long lifetime of the WSN is desirable.

WSNs are usually deployed in hostile environments, such as battlefields or disaster locations, where fixed infrastructures are not available. After deployment, it is infeasible to provide constant surveillance of and protection for a WSN. In many security-critical scenarios, adversaries may have ability to access sensor nodes without being detected. Adversaries can use proper devices to dig into sensor hardware and find key materials. Due to cost constraints, it is also unrealistic and uneconomical to employ tamper-resistant hardware to secure the cryptographic materials in each individual node. Even if tamper-resistant devices are available, they are still not able to guarantee perfect security of secret materials [7]. Hence, adversaries can capture any node and compromise the secrets stored in that node. Furthermore, adversaries can use the compromised secrets to derive more secrets shared between other non-compromised nodes. This means that the *node compromise* attack is unavoidable in WSNs. What we can do is to reduce the impact on other normal nodes as much as possible.

A compromised node can be used as a platform to launch other tricky attacks. The adversary can let the compromised node impersonate another normal node to establish secure communications with other normal nodes. Therefore, node authentication should be considered during the key establishment procedure. If the compromised node is involved as a router between a pair of source and sink nodes, the key negotiation procedure may fail just because the compromised node intentionally drops some packets for the negotiation between the source and sink.

Scalability is another important issue. According to different application scenarios, a WSN can have from tens to thousands of sensor nodes.

Moreover, during the lifetime of the WSN, some nodes can run out of power, and some new nodes can be inserted to increase the network processing capability. Therefore, the number of nodes can vary from time to time. These node dynamics demands simple, flexible, and scalable security protocols. However, to design such security protocols is not an easy task. The reason mainly lies in that the memory cost per node increases quickly when the network size enlarges. For example, conventional key establishment schemes [8,9] cannot support large networks due to their memory cost of $N - 1$ units in a network of N nodes. To increase the scalability, many works partition the entire network into several portions and apply conventional schemes in each portion, but the security performance is reduced.

Generally, higher-level security and less computation and communication overhead are contradictory requirements in the design of security protocols for WSNs. In most cases, a trade-off must be made between security and performance.

15.3 Symmetric Key Management

Due to its computational efficiency, it is commonly believed that symmetric key technology is more viable for resource-constrained, low-end devices than public key technology. Therefore, most of the security protocols developed thus far for WSNs are based on symmetric key technology. We discuss this approach first.

15.3.1 Global Key

The simplest symmetric key scheme is to use a *global key* [10], which is shared by all the sensor nodes. Data traffic is protected by the global key. The global key is updated periodically. In each period, a sensor node is elected as a key manager that generates and distributes a new global key to all the other sensor nodes. If all the sensor nodes are trustful, this scheme can effectively prevent external adversaries from accessing critical information that is secured by the global key. However, the scheme is very vulnerable to *node compromise* in that adversaries can get the global key by compromising only one node and thereby break into the entire sensor network.

15.3.2 Key Server

A WSN is usually connected to external wired networks through a *base station* (BS). The BS can act as a key server to distribute keys for any pair of sensor nodes [11]. In particular, each sensor node shares a unique key with the BS. When two nodes need to establish a shared key, one can generate

the key and let the BS forward the key to the other node. This procedure can be secured by the keys shared between the sensor nodes and BS.

This centralized approach can reduce the impact of node compromise because compromising one node does not result in the exposure of keys between non-compromised nodes. However, the communication overhead is high because two close nodes may have to carry out handshakes through the central key server at a distant place. There is still a concern for security, in that the key server may become a potential point of failure even if the server is under careful protection. The entire network can break down if adversaries successfully corrupt the key server.

15.3.3 Full Predistribution

Usually, a WSN belongs to an authority. The authority can do configurations before deploying the network such that any pair of nodes is preloaded with a unique shared key. This scheme is the most resilient to node compromise because adversaries will not know each key unless they compromise one of the two nodes sharing the key. Moreover, each key is preloaded, so no negotiation is required between nodes. However, the memory cost is very high. In a WSN of N nodes, each node needs to store $N-1$ keys, and the overall number of keys in the WSN is $\frac{N(N-1)}{2}$. It is unaffordable on memory-constrained sensor platforms when N is very large. Therefore, this approach works only in small sensor networks.

15.3.4 Blom Scheme

A similar approach was proposed by Blom [8]. His method is based on $(N, t+1)$ *maximum distance separable* (MDS) linear codes [12]. Before a WSN is deployed, a central authority first constructs a $(t+1) \times N$ public matrix P over a finite field $\mathbb{F}_q$. Then the central authority selects a random $(t+1) \times (t+1)$ symmetric matrix S over $\mathbb{F}_q$, where S is secret and only known to the central authority. An $N \times (t+1)$ matrix $A = (S \cdot P)^T$ is computed, where $(\cdot)^T$ denotes the transpose operator. The central authority preloads the i-th row of A and the i-th column of P to node i, for $i = 1, 2, \ldots, N$. When nodes i and j need to establish a shared key, they first exchange their columns of P, and then node i computes a key K_{ij} as the product of its own row of A and the j-th column of P and node j computes K_{ji} as the product of its own row of A and the i-th column of P. Because S is symmetric, it is easy to see that:

$$\begin{aligned} K = A \cdot P &= (S \cdot P)^T \cdot P = P^T \cdot S^T \cdot P \\ &= P^T \cdot S \cdot P = (A \cdot P)^T = K^T \end{aligned} \tag{15.1}$$

Therefore, nodes pair (i, j) will use $K_{ij} = K_{ji}$, as a shared key.

The Blom scheme has a *t-secure* property, in that in a network of N nodes, the collusion of less than $t + 1$ nodes cannot reveal any key shared by other pairs of nodes. This is because as least t rows of A and t columns of P are required to solve the secret symmetric matrix S. The memory cost per node in the Blom scheme is $t + 1$. To guarantee perfect security in a WSN with N nodes, the $(N - 2)$-secure Blom scheme should be used, which means the memory cost per node is $N - 1$. Hence, the Blom scheme can provide strong security in small networks.

15.3.5 Polynomial Model

A polynomial-based key establishment scheme was described by Blundo et al. [9]. It is a special case of Blom's scheme, in that a Vandermonde matrix is used as the generator matrix of MDS code. They used a t-degree bivariate polynomial, which is defined as:

$$f(x, y) = \sum_{i=0}^{t} \sum_{j=0}^{t} a_{ij} x^i y^j \tag{15.2}$$

over a finite field $\mathbb{F}_q$, where q is a prime that is large enough to accommodate a cryptographic key. By choosing $a_{ij} = a_{ji}$, we can get a symmetric polynomial in that $f(x, y) = f(y, x)$. Each sensor node is assumed to have a unique, integer-valued, non-zero identity. For each sensor node u, a *polynomial share* $f(u, y)$ is assigned, which means the coefficients of univariate polynomials $f(u, y)$ are loaded into node u's memory. When nodes u and v need to establish a shared key, they broadcast their IDs. Subsequently, node u can compute $f(u, v)$ by evaluating $f(u, y)$ at $y = v$, and node v can also compute $f(v, u)$ by evaluating $f(v, y)$ at $y = u$. Due to the polynomial symmetry, the shared key between nodes u and v has been established as $K_{uv} = f(u, v) = f(v, u)$. Similar the Blom scheme, a t-degree bivariate polynomial is also $(t + 1)$-secure, meaning that adversaries must compromise no less than $(t + 1)$ nodes holding shares of the same polynomial to reconstruct it. But the memory cost is also the same as that of the Blom scheme. Hence, the polynomial model is also suitable in small networks.

15.3.6 Random Key Predistribution

In ideal cases, every pair of nodes in a network should have a unique shared key. Although full predistribution, the Blom scheme, and the polynomial model can achieve this goal, the cost is that each node needs to store $N - 1$ keys in a network of N nodes. This is impractical for WSNs due to the memory constraints of sensor nodes when the network scale is very large. Instead, most recent research articles in this field lose the

security requirement and follow a *partial predistribution* approach, wherein key materials are predistributed such that some sensor nodes can establish shared keys directly and they can help to establish indirect shared keys between other sensor nodes.

A pioneer work following this approach is *random key predistribution* (called RKP hereafter) [13]. In RKP, each node is preloaded with a subset of m keys randomly selected from a global pool of M keys such that any pair of neighboring nodes can share at least one key with a certain probability, that is:

$$p = 1 - \frac{\binom{M-m}{m}}{\binom{M}{m}} \tag{15.3}$$

RKP is developed based on an observation that it is unnecessary to guarantee full connectivity for a sensor node with all its neighbors, as long as multi-link paths of shared keys exist among neighbors that can be used to set up a path key as needed. The rationale behind this observation is the random graph theory. When the probability p that a link exists between two nodes increases, the probability P_c that the entire graph is connected also increases. There is a required p such that P_c is almost 1. Hence, we can choose m and M such that the entire network is almost connected. In RKP, two neighboring nodes can have a shared key *directly* if their key subsets have an intersection or negotiate an *indirect* key through a *secure path*, along which every pair of neighboring nodes has a direct shared key.

A major concern of RKP is node compromise. By tampering or cryptanalysis, adversaries can compromise a node and expose its key subset. Because each key is reused by many sensor nodes, those exposed keys can be used to corrupt links between other non-compromised nodes if those links happen to be secured by the exposed keys.

Another concern is the communication overhead. Due to the memory constraint, each sensor node cannot keep too many keys. Hence, the value of p is rather small, which means that each sensor node needs to negotiate keys with a large portion of neighboring nodes through multilink paths.

15.3.7 Q-Composite RKP

To mitigate the impact of node compromise, Chan, Perrig, and Song [14] suggested to improve RKP such that any pairs of neighboring nodes are required to share at least q keys with a certain probability. Let $p(i)$ be the probability that two nodes share i keys; then:

$$p(i) = \frac{\binom{M}{i}\binom{M-i}{2(m-i)}\binom{2(m-i)}{m-i}}{\binom{M}{m}^2} \tag{15.4}$$

and the probability that two nodes share at least q keys is:

$$p = 1 - (p(0) + p(1) + \cdots + p(q-1)) \tag{15.5}$$

This scheme achieves greatly strengthened security under small-scale attacks while trading off increased vulnerability in the face of large-scale attacks on network nodes. As the amount of required key overlap increases, it becomes exponentially more difficult for an attacker with a given key set to break a link. However, to preserve the given probability p of two nodes sharing sufficient keys to establish a secure link, it is necessary to reduce the size of the global key pool M. This allows the attacker to gain a larger sample of the global key pool by breaking fewer nodes.

15.3.8 Random-Pairwise Key

Authentication is necessary to provide assurance for the identities of communicating parties. This can be achieved through the normal *challenge-response* approach based on the unique key shared by the communicating parties. In particularly, one verifier node can send an encrypted random number, called a challenge, to the other node, and that node can prove its identity by returning the decrypted result to the verifier node. The identity of the verifier node can be authenticated in the same way.

RKP and q-composite RKP can hardly provide authentication because of the reuse of the same key in many sensor nodes. To solve the problem, Chan, Perrig, and Song also proposed the *random-pairwise key* (RPK) [14] scheme. For each node, a set of M nodes is randomly selected from the entire network, and a unique pairwise key is assigned for the pair of the node and each of the nodes in the set. When the network consists of N nodes, any pair of nodes can share a pairwise key with a probability

$$p = \frac{M}{N} \tag{15.6}$$

Based on the random graph theory, the entire network can be connected as long as the probability is larger than a threshold. Hence, any pair of nodes can either share a direct key or negotiate an indirect pairwise key through a multilink secure path. The uniqueness of direct keys can be used to authenticate node identities.

In RPK, each direct key is uniquely generated and shared by a unique pair of nodes, so RPK is resilient to node compromise in terms of the secrecy of direct pairwise keys. However, the negotiation of indirect pairwise keys can introduce a lot of communication overhead due to the discovery of secure paths and handshakes between two end nodes.

15.3.9 Random Key Assignment

To discover whether the key sets of two nodes have an intersection, usually both nodes need to broadcast their key indices or find common keys through the challenge-response procedure. Such methods are not communication efficient. Pietro, Mancini, and Mei [15] improved the RKP scheme by associating the key indices of a node with the node identity. For example, each node is assigned a pseudo-random number generator $g(x, y)$, and the key indices for the node are calculated as $g(ID, i)$ for $i = 1, \ldots, M$, where ID is the node identity. In this way, other nodes can quickly find out which key is in its key set by checking its node identity.

15.3.10 Multiple-Space Key Predistribution

In the face of node compromise, the security level of RKP and its derivatives described above will deteriorate quickly, in that each time adversaries compromise one more node, more secrets such as keys are exposed.

To improve the resilience to node compromise, Du et al. [16] developed the *multiple-space key predistribution* (MSKP) scheme based on the Blom scheme. Specifically, a public matrix P and a global pool of symmetric matrices S_i for $i = 1, \ldots, w$ are constructed. Each tuple (S_i, P) is called a *key space*. For each sensor node, v spaces are randomly selected, and the node is configured with parameters derived according to the Blom scheme. Obviously, if two neighboring sensor nodes have a space in common, they can calculate a direct pairwise key according to the Blom scheme. The merit of MSKP is that it can tolerate up to a certain number of compromised node befores the security level of the network begins to deteriorate. This is due to the threshold-based security of the Blom scheme.

15.3.11 Polynomial Pool-Based Key Predistribution

Another scheme, called *polynomial pool-based key predistribution* (PPKP) [17], is basically the same as the MSKP scheme, but each Blom matrix is replaced by a t-degree bivariate polynomial. Each sensor node randomly selects v polynomials from a global pool of w polynomials. Any pair of neighboring nodes can calculate a direct pairwise key if they share the same polynomial. Like MSKP, PPKP can also provide threshold-based resilience to node compromise.

Another merit of MSKP and PPKP is that each direct key is tied to the identities of the nodes sharing it. Hence, they can provide authentication like RPK.

15.3.12 Hwang-Kim Scheme

The schemes discussed above require each sensor node keep many key materials such that two nodes share a key with a probability that can guarantee that the entire network is almost connected. The requirement may be too harsh in memory-constrained sensor networks. Hwang and Kim [18] revisited the RKP scheme and its derivatives, and proposed to reduce the amount of key materials that each node keeps while still maintaining a certain probability of sharing a key between two nodes. Their idea is to guarantee that the largest giant component of the network, instead of the entire network, is almost connected. Hence, each sensor node can keep less key materials. The probability that two nodes have a key in common is reduced, but it is still large enough for the largest network component to be connected. The trade-off is that more nodes can be isolated because they do not share keys with their neighbors.

15.3.13 PIKE Scheme

The probabilistic nature of the random distribution of key materials cannot guarantee that two neighboring nodes establish a shared key. To facilitate key establishment between every pair of neighboring nodes, a deterministic approach can be taken. In the *peer intermediaries for key establishment* (PIKE) scheme [19], all N sensor nodes are organized into a two-dimensional space (Figure 15.1), where the coordinate of each node is (x, y) for $x, y \in \{0, 1, 2, \ldots, \sqrt{N} - 1\}$. Each node shares unique pairwise keys with $2(\sqrt{N} - 1)$ nodes that have the same x or y coordinates in the two-dimensional space. For two nodes with no common coordinate, an intermediate node, which has common x or y coordinates with both nodes, is used as a router to forward a key for them. However, the communication overhead is rather high because the secure connectivity is only $\frac{2}{\sqrt{N}}$, which

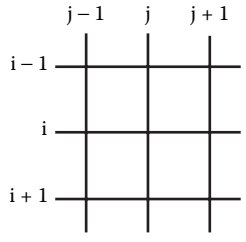

Figure 15.1 PIKE scheme. Sensor nodes are organized into a two-dimensional space.

means that each node must establish a key for almost each of its neighbors through multilink paths.

15.3.14 Grid-Based Key Predistribution

The *grid-based key predistribution* (GBKP) scheme [17] uses the same two-dimensional space as PIKE. Instead of pairwise keys in PIKE, GBKP assigns a bivariate symmetric polynomial for each set of nodes with the same x or y coordinate. Hence, direct keys can be established between nodes with the same x or y coordinate according to the polynomial model. Indirect keys can be negotiated in the same way as that in PIKE. PIKE and GBKP can guarantee that any pair of nodes shares a direct key or negotiates an indirect key through a third node. Moreover, a node can find whether it has a direct shared key with another node based on the coordinate of that node. This can provide an authentication service, in that the identity, associated with the coordinate, of a node can be challenged based on its keys that are related to its identity.

15.3.15 Scalable Key Agreement

Zhou and Fang [20] developed a scalable key agreement. They use a t-degree $(k+1)$-variate symmetric polynomial to establishe keys in a deterministic way.

A t-degree $(k+1)$-variate polynomial is defined as:

$$f(x_1, x_2, \ldots, x_k, x_{k+1}) = \sum_{i_1=0}^{t} \sum_{i_2=0}^{t} \cdots \sum_{i_k=0}^{t} \sum_{i_{k+1}=0}^{t} a_{i_1,i_2,\ldots,i_k,i_{k+1}} x_1^{i_1} x_2^{i_2} \cdots x_k^{i_k} x_{k+1}^{i_{k+1}} \tag{15.7}$$

All coefficients are chosen from a finite field $\mathbb{F}_q$, where q is a prime that is large enough to accommodate a cryptographic key.

A $(k+1)$-tuple permutation is defined as a bijective mapping:

$$\sigma : [1, k+1] \longrightarrow [1, k+1] \tag{15.8}$$

By choosing all the coefficients according to

$$a_{i_1,i_2,\ldots,i_k,i_{k+1}} = a_{i_{\sigma(1)},i_{\sigma(2)},\ldots,i_{\sigma(k)},i_{\sigma(k+1)}} \tag{15.9}$$

for any permutation σ, a symmetric polynomial can be obtained in that

$$f(x_1, x_2, \ldots, x_k, x_{k+1}) = f(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(k)}, x_{\sigma(k+1)}) \tag{15.10}$$

Each node is identified by an ID $(n_1, n_2, \ldots, n_k)$, which is the coordinate of a point in a k-dimension space $\mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_k$, where $n_i \in \mathcal{S}_i \subset \mathbb{Z}$ for $i = 1, \ldots, k$ and $\mathcal{S}_i \bigcap \mathcal{S}_j = \phi$, for $i \neq j$.

For a node $(n_1, n_2, \ldots, n_k)$ in the network, a *polynomial share*

$$f_1(x_{k+1}) = f(n_1, n_2, \ldots, n_k, x_{k+1}) = \sum_{i_{k+1}=0}^{t} b_{i_{k+1}} x_{k+1}^{i_{k+1}} \tag{15.11}$$

is calculated using the node ID as inputs to the t-degree $(k+1)$-variate symmetric polynomial.

If two nodes u with ID $(u_1, u_2, \ldots, u_k)$ and v with ID $(v_1, v_2, \ldots, v_k)$ have only one mismatch in their IDs, say $u_i \neq v_i$ for some i but $u_j = v_j = c_j$ for other $j \neq i$, then nodes u and v can calculate a shared key as:

$$K_{uv} = f(c_1, \ldots, u_i, \ldots, c_k, v_i) = f(c_1, \ldots, v_i, \ldots, c_k, u_i) \tag{15.12}$$

If two nodes have more than one mismatch in their IDs, they cannot calculate a direct key. In this case, they can negotiate an indirect key over a multi-hop path, along which every pair of neighboring nodes has already calculated a direct key.

An example of a three-dimensional ID space is given in Figure 15.2. For any edge, the pair of nodes at its two end can calculate a direct key because the two nodes have only one mismatch in their IDs. Suppose node

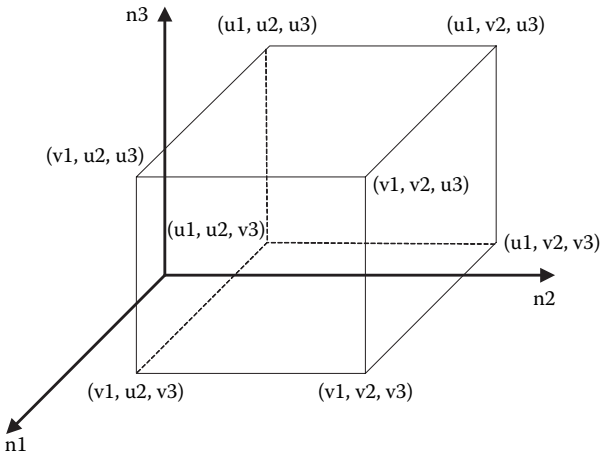

Figure 15.2 Multidimension key graph. Nodes (u_1, u_2, u_3) and (v_1, v_2, v_3) can negotiate a key through any path consisting of connected edges of the cube.

(u_1, u_2, u_3) needs to establish a shared key with node (v_1, v_2, v_3), where all three indices in their IDs are mismatching. There are three disjoint paths from node u to node v. For example, three disjoint paths are:

$$(u_1, u_2, u_3) \rightarrow (v_1, u_2, u_3) \rightarrow (v_1, v_2, u_3) \rightarrow (v_1, v_2, v_3)$$

$$(u_1, u_2, u_3) \rightarrow (u_1, u_2, v_3) \rightarrow (v_1, u_2, v_3) \rightarrow (v_1, v_2, v_3)$$

and

$$(u_1, u_2, u_3) \rightarrow (u_1, v_2, u_3) \rightarrow (u_1, v_2, v_3) \rightarrow (v_1, v_2, v_3)$$

Obviously, the above set of disjoint paths is not unique.

The dimension of the ID space k is a parameter to be controlled to achieve the trade-off between memory cost per node and scalability. To guarantee each direct key unsolvable by adversaries, no matter how many nodes are compromised, the memory cost per node is less than

$$M \leq k \log k + \log N + \left(\sqrt[k]{N} \sqrt[k+1]{\frac{k(k+1)!}{2}} + 1 \right) \log q \tag{15.13}$$

where N is the total number of nodes, q is the field size, and all the subspace S_i have the same cardinality. Obviously, the scheme has good scalability, in that the memory cost is only on the order $\mathcal{O}(\sqrt[k]{N})$ when k is fixed.

15.3.16 Location-Based Key Predistribution

In the aforementioned schemes, key materials are uniformly distributed in the entire terrain of a network. The uniform distribution makes the probability that two neighboring nodes share a direct key, called *secure connectivity*, rather small. Therefore, a lot of communication overhead is inevitable for the establishment of indirect keys. If some location information is known, two nearby sensor nodes can be preloaded intentionally with the same set of key materials. In this way, we can expect improvement in secure connectivity.

In the *location-based key predistribution* (LBKP) scheme [21], the entire sensor network is divided into many square cells. Each cell is associated with a unique t-degree bivariate polynomial. Each sensor node is preloaded with shares of the polynomials of its home cell and four other cells horizontally and vertically adjoining its home cell. After deployment, any two neighboring nodes can establish a pairwise key if they have shares of the same polynomial.

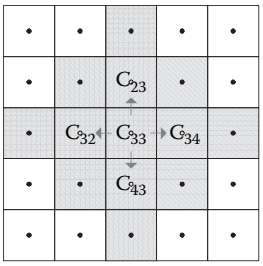

Figure 15.3 LBKP scheme. The polynomial of cell C_{33} is also assigned to cells C_{32}, C_{34}, C_{23}, and C_{43}. A node in C_{33} has some common polynomial information with other nodes in the shadow areas.

For example, in Figure 15.3, the polynomial of cell C_{33} is also assigned to cells C_{32}, C_{34}, C_{23}, and C_{43}. The polynomials of other cells are assigned in the same way. As a result, a node in C_{33} has some polynomial information in common with other nodes in the shadow areas.

15.3.17 Key Establishment Using Deployment Information

Du et al. [22] divide, the entire network into many square cells. Each cell is assigned a subset key pool S_{ij}, $i = 1, \ldots, u$ and $j = 1, \ldots, v$ out of a global key pool S. Those subset key pools are set up such that the key pools of two neighboring cells will share a portion of keys. In each cell, the RKP [13] scheme is applied.

Using deployment knowledge to achieve the same connectivity, the size of the key ring that one node holds in this scheme is much less than that in the RKP scheme. This can significantly save the memory of sensors. However, it still inherits the same weakness as RKP, in that it does not provide authentication.

15.3.18 Location-Aware Key Management

Huang et al. [23] also used square cells. To provide intra-cell connectivity, the MSKP scheme [16] is applied in each cell such that any pair of nodes having a common space in one cell can establish a shared key. The RPK scheme [14] is applied between neighboring cells in that for each sensor node, a node from each of its neighboring cells is selected and a unique key is assigned to the pair of nodes. This provides inter-cell connectivity.

The MSKP scheme provides strong resilience to node compromise. However, network connectivity is low because the randomly- distributed key spaces in each cell and the randomly- assigned pairwise keys between

cells can only guarantee that each node has shared keys with a portion of its neighbors.

15.3.19 Neighboring-Cell-Based Predistribution Model

Threshold-based schemes such as the Blom [8] and polynomial schemes [9] can tolerate only a certain number of compromised nodes. The more nodes sharing a Blom matrix or a polynomial, the more likely they are exposed due to node compromise. Therefore, if we can reduce the number of nodes sharing a Blom matrix or a polynomial, the security then increases. Following this idea, Zhou, Zhang, and Fang investigated a neighboring-cell-based predistribution model based on hexagon [24] and triangle [25] grid models.

Zhou, Zhang, and Fang [24] divided the entire network terrain into non-overlapping hexagon cells. The polynomial model [9] is used here. Unlike LBKP [21], which assigns a polynomial to each cell and its four adjacent cells, [24] assigns a polynomial to each pair of neighboring cells. For example, in Figure 15.4, cell $c0$ is assigned six polynomials, each of which is uniquely shared with one of its neighboring cells; therefore, the nodes in cell $c0$ can establish direct pairwise keys with the nodes in the shadow area. The reduced number of nodes sharing one polynomial means less chance the polynomial can be exposed by collusion of compromised nodes. Hence, this new predistribution method can improve the resilience to node compromise. At the same security level, [24] requires less memory cost for each node compared with LBKP [21]. Due to the use of deployment knowledge, the security connectivity is very high.

Later, Zhou, Zhang, and Fang [25] improved the hexagon grid model to the triangle grid model, in which the entire network is divided into non-overlapping triangle cells. The same key materials predistribution method is

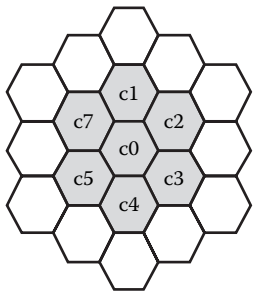

Figure 15.4 Hexagon grid-based key predistribution. Cell $c0$ is assigned six Blom matrices or polynomials, each of which is uniquely shared with one of its neighboring cells.

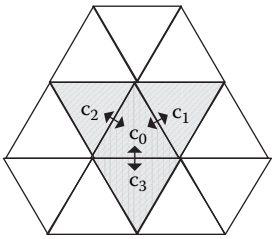

Figure 15.5 Triangle grid-based key predistribution. Cell $c0$ is assigned three Blom matrices or polynomials, each of which is uniquely shared with one of its neighboring cells.

used as in [24], in which each pair of neighboring cells is associated with a unique t-degree bivariate polynomial [9] or a Blom matrix [8]. For example, in Figure 15.5, cell $c0$ is assigned three Blom matrices or polynomials, each of which is uniquely shared with one of its neighboring cells, and the nodes in cell $c0$ can establish direct pairwise keys with the nodes in the shadow area. This scheme further reduces the memory cost of each node at the same security level compared to the hexagon or the square grid model because each node needs to keep only three polynomials or matrices. Although the security connectivity is smaller than that of hexagon grid model, it is still much larger than conventional schemes that do not use location information.

15.3.20 Group-Based Key Predistribution Framework

Liu, Ning, and Du [26] established a group-based key predistribution framework that may incorporate previous schemes. They divide all the sensor nodes into many *deployment groups*. In each group, a specific keying material distribution scheme, which could be one of the schemes discussed above, is applied to provide in-group connectivity. Also, each group picks one node and all those picked nodes form a *cross group*. There is also a specific keying material distribution scheme for each cross group. Therefore, two nodes from different deployment groups can establish a shared key through a path in a cross group.

15.3.21 LEAP

Zhu, Setia, and Jajodia proposed a *Localized Encryption and Authentication Protocol* (LEAP) [27] for sensor networks. LEAP supports the establishment of four types of keys for each sensor node: (1) an *individual key* shared with the base station, (2) a *pairwise key* shared with another sensor node,

(3) a *cluster key* shared with multiple neighboring nodes, and (4) a *group key* shared by all the nodes in the sensor network.

In this protocol, each node has an individual key that is only shared with the base station. This key is generated and preloaded into each node prior to its deployment. The individual key K_u^m for a node u is generated as $K_u^m = f_{K_s^m}(u)$ (assuming that each node has a unique ID). Here, f is a pseudo-random function and K_s^m is a master key known only to the base station.

A pairwise key refers to a key shared only between the node and one of its direct neighbors (i.e., one-hop neighbors). At first, each node u is preloaded with an initial key K_I. Then sensor node u derives a master key $K_u = f_{K_I}(u)$. When it is deployed, node u initiates a timer $T_{\min}$ and then broadcasts a *HELLO* message that contains a nonce N to each neighbor v. The reply from each neighbor v is authenticated using its master key K_v. Because node u can compute K_v using K_I, it is able to verify node v's identity independently. The handshake is as follows:

$$u \rightarrow * : \quad u, N_u$$

$$v \rightarrow u : \quad v, MAC(K_v, N_u|v)$$

Then node u computes its pairwise key with v as $K_{uv} = f_{K_v}(u)$. Node v can also compute K_{uv} independently. When a preset timer $T_{\min}$ expires, node u erases K_I and all the keys K_v it computed in the previous phase.

A cluster key is a key shared by a node and all its neighbors, and is primarily used for securing locally broadcast messages, for example, routing control information, or securing sensor messages that can benefit from passive participation. Consider the case that node u wants to establish a cluster key with all its immediate neighbors $v_1, v_2, \ldots, v_m$. Node u first generates a random key K_u^c , then encrypts this key with the pairwise key of each neighbor, and then transmits the encrypted key to each neighbor v_i:

$$u \rightarrow v_i : \quad \left(K_u^c\right)_{K_{uv_i}}$$

Node v_i decrypts the key K_u^c and stores it in a table. When one of the neighbors is revoked, node u generates a new cluster key and transmits to all the remaining neighbors in the same way.

A group key is a key shared by all the nodes in the network, and is necessary when the base station is distributing a secure message (for example, a query on some event of interest or a confidential instruction) to all the nodes in the network. To tolerate node failures, the group key should be updated occasionally. They employ the μTESLA protocol [11] to distribute group keys.

Unlike the schemes discussed above, LEAP assumes the network is safe during its initialization phase. Otherwise, the pairwise key establishment procedure can be attacked by adversaries.

15.3.22 Key Infection

The schemes described above assume a strong adversary model in which adversaries are able to compromise any node in a network. In such a case, security protocols are usually heavyweighted such that they can deal with worst-case attacks. However, in many scenarios, adversaries may not have enough resources or the ability to launch such worst-case attacks. Therefore, Anderson, Chan, and Perrig assumed a weak adversary model [28] where adversaries can only eavesdrop a small portion of communications between sensor nodes during node deployment phase. Hence, a node wishing to communicate securely with other nodes simply generates a symmetric key and sends it in the clear to its neighbors.

15.4 Public Key Management

A symmetric key algorithm is efficient on low-end devices, but the schemes described in the previous section are rather complex for sensor networks in terms of communication overhead and memory cost. As a contrast, public key technology is easier to manage and more resilient to node compromise, although it is much more computationally expensive. Each node can keep its private key secret and only publish its public key; therefore, compromised nodes cannot provide any clue to the private keys of noncompromised nodes.

15.4.1 RSA Algorithm

RSA [4] is a very popular public key algorithm that can provide authentication and encryption services. At first, a node generates two large random and distinct primes p and q, each roughly the same size. Then it computes $n = pq$ and $\phi = (p-1)(q-1)$. A random integer e is selected, such that $1 < e < \phi$ and $\gcd(e, \phi) = 1$. After that, a unique integer d is calculated such that $1 < d < \phi$ and $ed = 1 \pmod{\phi}$. Hence, the node's public key is (n, e) and its private key is d.

When a node B needs to secretly send a message to node A, B can represent the message as an integer m in the interval $[0, n-1]$, compute $c = m^{e_A} \bmod n_A$ using A's public key, and send c to A. Node A can use its private key d_A to decrypt c to get $m = c^{d_A} \bmod n_A$.

To authenticate itself to node A, node B can generate a public message m and calculate a signature $s = m^{d_B} \bmod n_B$, then send a certificate $\langle m, s \rangle$

to A. Node A can verify node B's identity by checking whether m is equal to $s^{e_B} \bmod n_B$.

However, the exponential operation in RSA is very expensive, especially for large exponents. Hence, the focus of applying RSA in sensor networks is to develop efficient implementations on resource-constrained sensor platforms. RSA is rarely used to provide encryption in sensor networks. The private key operation of RSA is more expensive than public key operation because d is usually rather larger than e. *TinyPK* [29] uses RSA to provide the authentication service for external parties when they access sensor networks. In particular, each external party carries an RSA-based certificate and shows it to the sensor network. Sensors can verify the certificate and authenticate the external party. To simplify the signature verification on the sensor side, the RSA public key was chosen as $e = 3$. To avoid the secret key operation on the external party side, the certificate is calculated by a central network manager and preloaded to the external party.

15.4.2 Diffie-Hellman Algorithm

The Diffie-Hellman algorithm [3] is usually used to achieve key agreement between communication parties. At first, two nodes A and B agree on two parameters g and p, where g is a generator of $\mathbb{Z}_p$ and p is a large prime. Then node A chooses a secret integer x_A and sends $g^{x_A} \bmod p$ to B, and B also chooses a secret integer x_B and sends $g^{x_B} \bmod p$ to A. The shared key between A and B can be calculated as $K_{AB} = (g^{x_A})^{x_B} = (g^{x_B})^{x_A} = g^{x_A x_B} \bmod p$.

To make the Diffie-Hellman algorithm viable on sensor platforms, TinyPK chooses the base of exponentiation operation as $g = 2$ [29].

15.4.3 Elliptic Curve Cryptography

Recently, *elliptic curve cryptography* (ECC) [30,31] has become a very hot topic in academia and industry, and is seen as the basis for the next-generation security infrastructure. The reason is that algorithms based on ECC is more efficient than RSA and Diffie-Helman at the same security level. The fundamental operation underlying RSA is the modular exponentiation in integer rings. Its security stems from the difficulty in factorizing large integers. Currently, there only exist sub-exponential algorithms to solve the integer factorization problem.[2] ECC operates on groups of points

[2] Given a positive integer $n = pq$, where p and q are large pairwise distinct primes, find p and q.

over elliptic curves and derives its security from the hardness of the *Elliptic Curve Discrete Logarithm Problem* (ECDLP).[3] The best algorithms known for solving ECDLP are exponential. Hence, ECC can achieve the same level of security as RSA with smaller key sizes. For example, 163-bit ECC can provide comparable security to conventional 1024-bit RSA [32]. Under the same security level, the smaller key sizes of ECC offer merits of faster computational efficiency, as well as memory, energy, and bandwidth savings; thus, ECC is better suited for resource-constrained devices.

In [32], Diffie-Hellman over ECC is suggested to achieve key agreement between sensor nodes. An elliptic curve E with a generator G is chosen as a global public parameter. Node A chooses a secret integer x_A and sends x_AG to node B, and B as well chooses a secret integer x_B and sends x_BG to A. The shared key between A and B can be calculated as $K_{AB} = x_B(x_AG) = x_A(x_BG) = x_Ax_BG$.

Huang et al. [33] consider a sensor network consisting of some secure managers and many sensor nodes. An ECC-based authenticated key establishment protocol is proposed for the key establishment between secure managers and sensor nodes. To reduce the computational overhead of sensor nodes, most computationally expensive public key operations are put on the secure manager side.

15.4.4 Efficient Implementations

Computational efficiency is a critical issue in applying public key technology to sensor platforms. Gaubatz, Kaps, and Sunar [34] showed the feasibility of implementing public key technology with the right selection of algorithms and associated parameters, optimization, and low-power techniques. The conceptual implementations of the Rabin scheme [35] and NtruEncrypt scheme [36] were described as examples. In many cases, high-level programming languages cannot be optimized for specific hardware platforms, and therefore assembly languages are required to further reduce computing time. Gura et al. [37] evaluated the assembly language implementations of ECC and RSA on the Atmel ATmega128 processor [38], which is popular for sensor platforms such as Crossbow MICA Motes [6]. In their implementation, a 160-bit point multiplication of ECC requires only 0.81 seconds, while 1024-bit RSA public key operation and private key operation require 0.43 and 10.99 seconds, respectively.

[3] Given a generator G of a finite cyclic point group $\mathbb{G}$ over an elliptic curve $E(\mathbb{F}_q)$ and a point Q in the group, find an element $x \in \mathbb{F}_q$ such that $xG = Q$.

15.4.5 Authentication of Public Keys

Another critical issue in applying public key technology is the authenticity of public keys. A public key should be really owned by the node that claims to have the public key. Otherwise, adversaries can easily impersonate any node by claiming its public key and launch a *man-in-the-middle* attack. For example, a malicious node C can impersonate node B to node A and also impersonate A to B if A and B cannot verify the public key of each other. In this way, node C can act as an invisible router and learn all the messages between A and B. The conventional solution to public key authentication is to use a certificate signed by a trustful *certificate authority* (CA). Therefore, node B can send its public key with corresponding certificate to node A such that A can verify the correctness of the certificate with the well-known public key of CA. Node B can verify the authenticity of A's public key by following the same procedure.

Although technical advances have made the usage of public key technology viable in WSNs, public key algorithms are still more expensive than symmetric key algorithms. The authentication of public keys can incur high energy consumption because it is likely to be performed many times. Du, Wang, and Ning [39] developed public key authentication scheme based on a symmetric key technique, the *Merkle tree* [40]. In the Merkle tree, each parent is a hash of the concatenation of its children, and each leaf is corresponding to a node and is calculated as a hash of the node ID and its public key. When a sensor wants to authenticate its public key, it attaches its public key with the siblings of the tree nodes along the path from the leaf of the sensor to the root. Other sensors verify whether they can recover the root and decide the authenticity of the public key.

15.4.6 Location-Based Keys

Based on identity-based cryptography [41], where the publicly known identity information of a node is used as its public key, Zhang et al. [42,43] proposed the notion of *location-based keys* by binding the private keys of individual nodes to both their IDs and locations.

The following parameters are chosen and preloaded to each node: a q-order cyclic group G_1 of points on an elliptic curve; a pairing over the elliptic curve $e : G_1 \times G_1 \rightarrow G_2$ that satisfies the bilinear property, that is,

$$e(P + Q, R + S) = e(P, R)e(P, S)e(Q, R)e(Q, S) \tag{15.14}$$

a hash function maps a bit string to a point in G_1 (i.e., $H_1 : \{0, 1\}^* \rightarrow G_1$); and another hash function maps a bit string to an integer, $H_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$. For a node A, an identity-based key (IBK), $IK_A = kH_1(A)$, is preloaded to A, where k is a network-wide secret parameter. After deployment, the

position of A is used to derive a location-based key (LBK), $LK_A = kH_1(pos_A)$. The LBK is encrypted by the IBK and securely transmitted to A.

When node A needs to communicate with node B, A first sends an authenticate request, including its position pos_A and a nonce n_A to B. If node A is in the transmission range of node B, B returns a reply including its own location pos_B, a random nonce n_B, and an authenticator V_B calculated as:

$$V_B = H_2(e(LK_B, H1(pos_A)) \parallel n_A \parallel n_B \parallel 0) \tag{15.15}$$

If node A finds that node B is in its transmission range, A proceeds to compute a verifier V'_B as:

$$V'_B = H_2(e(H_1(pos_B), LK_A) \parallel n_A \parallel n_B \parallel 0) \tag{15.16}$$

If and only if both A and B have the authentic LBKs corresponding to their claimed locations, they can have:

$$\begin{aligned} e(LK_B, H_1(pos_A)) &= e(H_1(pos_B), LK_A) \\ &= (e(H_1(pos_B), H_1(pos_A)))^k \end{aligned} \tag{15.17}$$

After verifying the equality of V'_B and V_B, A can ascertain that B is an authentic neighbor with the claimed location pos_B. Node A, in return, should send to B its own authenticator V_A computed as:

$$V_A = H_2(e(H_1(pos_B), LK_A) \parallel n_A \parallel n_B \parallel 1) \tag{15.18}$$

Then, node B can determine whether A is an authentic neighbor with the claimed location pos_A. Based on this three-way handshaking, nodes A and B can achieve mutual authentication and establish an authentic link between them. Meanwhile, a pairwise key between A and B is established as $(e(H_1(pos_B), H_1(pos_A)))^k$.

The location-based keys have perfect resilience to node compromise in that no matter how many nodes are compromised, the location-based keys of non-compromised nodes as well as their pairwise keys always remain secure. It has been shown in [42,43] that the solution can defend against a wide range of attacks, such as the Wormhole attack, the Sybil attack, and the node replication attack, in sensor networks.

15.5 Open Issues

In this section we discuss some problems that need to be fully studied.

15.5.1 Memory Cost

High security and lower overhead are two objectives that a key management protocol needs to achieve. Although there have been several proposals for key establishment in sensor networks, they can hardly address these two requirements, as discussed in this chapter. Strong security protocols usually require large amounts of memory cost, as well as high-speed processors and large power consumption. However, they cannot be easily supported due to the constraints on hardware resources of the sensor platform.

It is well known that in wireless environments, transmission of one bit can consume more energy than computing one bit. Therefore, communication overhead can dominate the entire power consumption. In key management protocols, direct key establishment does not require communication or only a few rounds of one-hop communications, but indirect key establishment is performed over multi-hop communications. To reduce the multi-hop communication overhead, high secure connectivity, which is the probability of direct key establishment between a pair of nodes, is desirable. However, highly secure connectivity requires more key materials in each node, which is usually impratical, especially when the network size is large.

Considering the above two issues, memory cost can be the major bottleneck in designing key management protocols. How to reduce memory cost while still maintaining a certain level of security and overhead is a very important issue.

15.5.2 End-to-End Security

The major merit of symmetric key technology is its computational efficiency. However, most current symmetric key schemes for WSNs aim at the link layer security — not the transport layer security — because it is impractical for each node to store a transport layer key for each of the other nodes in a network due to the huge number of nodes.

However, end-to-end communication at the transport layer is very common in many WSN applications. For example, to reduce unnecessary traffic, a fusion node can aggregate reports from many source nodes and forward a final report to the sink node. During this procedure, the reports between source nodes and the fusion node and the one between the fusion node and the sink node should be secured. In hostile environments, however, any node can be compromised and become malicious. If one of the intermediate nodes along a route is compromised, the message delivered along the route can be exposed or modified by the compromised node. Employing end-to-end security can effectively prevent message tampering by any malicious intermediate node.

Compared with symmetric key technology, public key technology is expensive but has flexible manageability and supports end-to-end security. A more promising approach to key establishment in WSNs is to combine the merits of both symmetric key and public key techniques, in that each node is equipped with a public key system and relies on it to establish end-to-end symmetric keys with other nodes. To achieve this goal, a critical issue is to develop more efficient public key algorithms and their implementations so that they can be widely used on sensor platforms.

How to prove the authenticity of public keys is another important problem. Otherwise, a malicious node can impersonate any other normal node by claiming its public key. Identity-based cryptography is a shortcut to avoid the problem. Currently, most identity-based cryptographical algorithms operate on elliptic curve fields, and pairing over elliptic curves is widely used in the establishment of identity-based symmetric keys. However, the pairing operation is very costly, comparable to or even more expensive than RSA. Therefore, fast algorithms and implementations are the major tasks of researchers.

15.5.3 Efficient Symmetric Key Algorithms

There is still a demand for the development of more efficient symmetric key algorithms because encryption and authentication based on symmetric keys are very frequent in the security operations of sensor nodes. For example, in the link layer security protocol TinySec [44], each packet must be authenticated, and encryption can also be triggered if critical packets are transmitted. Therefore, fast and cost-efficient symmetric key algorithms should be developed.

15.5.4 Key Update and Revocation

Once a key has been established between two nodes, the key can act as a master key and be used to derive different sub-keys for many purposes (e.g., encryption and authentication). If each key is used for a long time, it may be exposed due to cryptanalysis over the ciphertexts intercepted by adversaries. To protect the master key and those sub-keys from cryptanalysis, it is wise to update keys periodically. The period of update, however, is difficult to choose. Because the cryptanalysis capability of adversaries is unknown, it is very difficult to estimate how long it takes for adversaries to expose a key by cryptanalysis. If the key update period is too long, the corresponding key may also be exposed. If it is too short, frequent updates can incur large overhead.

A related problem is key revocation. If one node is detected to be malicious, its key must be revoked. However, key revocation has not been

thoroughly investigated. Although Chan et al. [45] proposed a distributed revocation protocol, it is only based on the random-pairwise key scheme [14], and cannot easily be generalized into other key establishment protocols.

15.5.5 Node Compromise

Node compromise is the most detrimental attack on sensor networks. Because compromised nodes have all the authentic key materials, they can result in very severe damage to WSN applications and cannot be detected easily. How to counteract node compromise remains under investigation.

Most current security protocols try to defend against node compromise through careful protocol design such that the impact of node compromise can be restricted to a small area. However, a hardware approach is more promising. With advances in hardware design and manufacturing techniques, much stronger, tamper-resistant, and cheaper devices can be installed on the sensor platform to counteract node compromise.

15.6 Conclusion

Key management is the most critical component in the design of security protocols for wireless sensor networks, and has been drawing intensive interest from both academia and industry. In this chapter we surveyed current solutions to the key management issue in wireless sensor networks and shed light on future directions of the issue in wireless sensor networks. There are many challenges in the design of key management schemes due to various resource limitations and salient features of wireless sensor networks. Secure and efficient key management schemes are still under explorations.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communication Magazine*, Vol. 40, No. 8, pp. 102–114, August 2002.

[2] C.E. Shannon, "Communication theory of secrecy systems," *Bell Sys. Tech. J.*, Vol. 28, pp. 656–715, October 1949.

[3] W. Diffie and M.E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, Vol. IT-22, No. 6, pp. 644–654, 1976.

[4] R.L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, Vol. 21, No. 2, pp 120–126, February 1978.

[5] A. Wood and J. Stankovic, "Denial of service in sensor networks," *IEEE Computer*, pp. 54–62, October 2002.
[6] Crossbow Technology, http://www.xbow.com/
[7] R. Anderson and M. Kuhn, "Tamper resistance — a cautionary note," *Proc. 2nd USENIX Workshop on Electronic Commerce*, Oakland, CA, November 18–21, 1996, pp 1–11.
[8] R. Blom, "An optimal class of symmetric key generation systems," in *Proc. of EUROCRYPT '84*, 1985, pp. 335–338.
[9] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," in *Advances in Cryptology C CRYPTO 92, LNCS 740*, 1992, pp. 471–486.
[10] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti, "Secure pebblenets," *ACM Mobihoc'01*, Long Beach, CA, 2001.
[11] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, and D.E. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, Vol. 8, pp. 521–534, 2002.
[12] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error Correction Codes*, North-Holland, New York, 1977.
[13] L. Eschenauer and V. Gligor, "A key management scheme for distributed sensor networks," in *ACM CCS2002*, Washington, D.C., 2002.
[14] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 11–14, 2003, p. 197.
[15] R.D. Pietro, L.V. Mancini, and A. Mei, "Random key-assignment for secure wireless sensor networks," in *Conference on Computer and Communications Security (CCS'03)*, 2003.
[16] W. Du, J. Deng, Y.S. Han, and P.K.Varshney, "A pairwise key predistribution scheme for wireless sensor networks," in *CCS'03*, Washington, D.C., October 27–30, 2003.
[17] D. Liu and P. Ning, "Establishing pairwise keys in distributied sensor networks," *CCS'03*, Washington, D.C., 2003.
[18] J. Hwang and Y. Kim, "Revisiting random key pre-distibution schemes for wireless sensor networks," in *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04)*, October 25, 2004, Washington, D.C.
[19] H. Chan and A. Perrig, "Pike: peer intermediaries for key establishment in sensor networks," in *IEEE INFOCOM'05*, March 2005.
[20] Y. Zhou and Y. Fang, "A scalable key agreement scheme for large scale networks," *2006 IEEE International Conference on Networking, Sensing and Control (ICNSC'06)*, Fort Lauderdale, FL, April 23–25, 2006.
[21] D. Liu, and P. Ning, "Location-based pairwise key establishments for relatively static sensor networks," in *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'03)*, October 2003.
[22] W. Du, J. Deng, Y.S. Han, S. Chen, and P.K. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," in *IEEE INFOCOM 2004*, Hong Kong, March 2004.
[23] D. Huang, M. Mehta, D. Medhi, and L. Harn, "Location-aware key management scheme for wireless sensor networks," in *Proceedings of the 2nd ACM*

Workshop on Security of Ad Hoc and Sensor Networks (SASN'04), October 25, 2004, Washington, D.C.

[24] Y. Zhou, Y. Zhang, and Y. Fang, "LLK: a link-layer key establishment scheme in wireless sensor networks," *IEEE WCNC'05*, New Orleans, LA, March 2005.

[25] Y. Zhou, Y. Zhang, and Y. Fang, "Key establishment in sensor networks based on triangle grid deployment model," in *Proc. IEEE MILCOM'05*, Atlantic City, NJ, October 17–20, 2005.

[26] D. Liu, P. Ning, and W. Du, "Group-based key pre-distribution in wireless sensor networks," *ACM WiSe'05*, September 2005.

[27] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanism for large-scale distributed sensor networks," in *ACM CCS'03*, Washington, D.C., October 27–31, 2003.

[28] R. Anderson, H. Chan, and A. Perrig, "Key infection: smart trust for smart dust," in *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP'04)*, 2004.

[29] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: securing sensor networks with public key technology," *SASN'04*, Washington, D.C., October 25, 2004.

[30] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, Vol. 48, pp. 203–209, 1987.

[31] V. Miller, "Uses of elliptic curves in cryptography," *Lecture Notes in Computer Science 218: Advances in Cryptology — CRYPTO'85*. Springer-Verlag, Berlin, 1986, pp. 417–426.

[32] D.J. Malan, M. Welsh, and M.D. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," *First IEEE International Conference on Sensor and Ad Hoc Communications and Networks* (SECON'04), Santa Clara, CA, October 2004.

[33] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang, "Fast authenticated key establishment protocols for self-organizing sensor networks," *ACM WSNA'03*, San Diego, CA, 2003.

[34] G. Gaubatz, J. Kaps, and B. Sunar, "Public key cryptography in sensor networks — revisited," *ESAS'04*, 2004.

[35] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, 1996.

[36] J. Hoffstein, J. Pipher, and J.H. Silverman, "NTRU: a ring based public key cryptosystem," *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, Vol. 1433, pp. 267–288, 1998.

[37] N. Gura, A. Patel, A. Wander, H. Eberle, and S.C. Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," *CHES'04*, 2004.

[38] Atmel Corporation, *http://www.atmel.com/*

[39] W. Du, R. Wang, and P. Ning, "An efficient scheme for authenticating public keys in sensor networks," *ACM MobiHoc'05*, May 2005.

[40] R. Merkle, "Protocols for public key cryptosystem," *Proceedings of the IEEE Symposium on Research in Security and Privacy*, April 1980.

[41] D. Boneh and M. Franklin, "Identify-based encryption from the weil pairing," in *Proc. CRYPTO'01*, Ser. LNCS, Vol. 2139, Springer-Verlag, 2001, pp. 213–229.

[42] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Securing sensor networks with location-based keys," *IEEE WCNC'05*, March 2005.

[43] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Location-based compromise-tolerant security mechanisms for wireless sensor networks," *IEEE Jounal on Selected Areas in Communications*, *Special Issue on Security in Wireless Ad-Hoc Networks*, Vol. 24, No. 2, pp. 247–260, 2006.

[44] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in the *Second ACM Conference on Embedded Networked Sensor Systems (SensSys'04)*, Baltimore, MD, November 2004.

[45] H. Chan, V. Gligor, A. Perrig, and G. Muralidharan, "On the distribution and revocation of cryptographic keys in sensor networks," *IEEE Transactions on Dependable and Secure Computing*, Vol. 2, No. 3, Jul.–Sep. 2005.