Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2011, Article ID 765143, 11 pages
doi:10.1155/2011/765143

Research Article

EDDK: Energy-Efficient Distributed Deterministic Key Management for Wireless Sensor Networks

Xing Zhang,[1] Jingsha He,[2] and Qian Wei[1]

[1] *College of Computer Science and Technology, Beijing University of Technology, Beijing 100124, China*
[2] *School of Software Engineering, Beijing University of Technology, Beijing 100124, China*

Correspondence should be addressed to Jingsha He, he@bjut.edu.cn

Received 31 May 2010; Revised 23 September 2010; Accepted 21 November 2010

Academic Editor: Damien Sauveron

Energy efficiency is an essential requirement for wireless sensor networks while security must also be ensured for mission-critical applications. In this paper, we present an energy-efficient distributed deterministic key management scheme (EDDK) for resource-constrained wireless sensor networks. EDDK mainly focuses on the establishment and maintenance of the pairwise keys as well as the local cluster keys and can fix some flaws in some existing key management schemes. Not only can the neighbor table constructed during key establishment provide the security for key maintenance and data transfer, but it can also be used to effectively manage the storage and update of the keys. By using the elliptic curve digital signature algorithm in EDDK, both new and mobile sensor nodes can join or rejoin a sensor network securely. Unlike some centralized and location-based key management schemes, EDDK does not depend on such infrastructure as base stations and robots and thus has a high level of flexibility. Experiments and analyses show that EDDK has a very low overhead in terms of computation, communication, and storage.

1. Introduction

A wireless sensor network (WSN) is a promising network infrastructure for many applications such as environmental monitoring, medical care, and home appliance management. It is also useful for battlefield surveillance and homeland security because WSNs can be easily deployed for those applications. However, in many hostile and tactical scenarios as well as critical commercial applications, security mechanisms are required to protect WSNs from malicious attacks. Therefore, the security of WSNs has become an important issue and a challenging design task [1]. Furthermore, security mechanisms based on cryptography would make key management a central issue to ensure the security of network services and applications in WSNs.

Many key management schemes have been proposed for WSNs over the past few years [1, 2]. From the standpoint of network structure, key management schemes can be centralized or distributed. From the standpoint of key sharing between neighboring nodes, key management schemes can be probabilistic or deterministic. For centralized key management schemes, network scalability is poor and communication overhead is high. Probabilistic schemes cannot guarantee that two neighboring nodes could successfully establish a shared key according to the underlying random graph theory. It will not be desirable if some sensor nodes cannot establish shared keys with their neighbors and are thus isolated. In addition, in probabilistic schemes, the impact of node compromise and the lack of authentication are serious issues because of the reuse of the same keys by more than one node. Deterministic schemes can solve some problems that probabilistic schemes have. But the addition of new nodes is still an issue in such schemes because the confidentiality and authenticity of new pairwise keys cannot be fully realized [3–5]. Even in the scheme in [6], the normal node's motion will be regarded as the injection of the replicas of compromised nodes. An adversary can thus cause a network to collapse by merely changing the nodes' locations in the network. Consequently, the scheme in [6] is susceptible to the so-called node location changing attack.

Due to limited battery life of sensor nodes, any security mechanisms for WSNs must be energy efficient. Specifically, the number of message transmission and the amount of computation must be kept as low as possible. Meanwhile, the size of a sensor network should not be constrained by the available storage and energy resource in each node.

In this paper, we propose a distributed deterministic key management scheme for WSNs in which the pairwise keys and the local cluster keys are set up through the broadcast information during the network initialization phase and no further message exchange is needed afterwards. Consequently, the communication overhead is very low. Furthermore, pairwise keys are also totally decentralized. Therefore, the compromise of some sensor nodes will not affect any other noncompromised pairwise keys. For the establishment of keys for new nodes and mobile nodes, we propose a composite mechanism based on the elliptic curve digital signature algorithm (ECDSA) in which resource consumption can also be kept very low.

The rest of this paper is organized as follows. In Sections 2 and 3, we discuss some related work and analyze some present deterministic key schemes. In Section 4, we present our energy-efficient distributed deterministic key management scheme (EDDK), and in Sections 5 and 6, we analyze the security and performance of EDDK. Finally, we conclude this paper in Section 7.

2. Related Work

2.1. Centralized versus Distributed Key Management Schemes. LKHW is a centralized key management scheme for WSNs based on the Logical Key Hierarchy (LKH) [7]. In the scheme, the base station is treated as a key distribution center (KDC) and all keys are logically distributed through a tree rooted at the base station. In such a centralized scheme, besides poor network scalability and high communication overhead, the base station that knows all pairwise keys can be a single point of failure for network security. In distributed key management schemes, different key controllers are used to lower security risk and to allow for better network scalability [2]. Consequently, a majority of the key management schemes proposed so far are distributed ones that also fall into the deterministic and probabilistic categories which we discuss in the following subsection.

2.2. Probabilistic versus Deterministic Key Management Schemes. Eschenauer and Gligor proposed a basic probabilistic key predistribution scheme [8]. In the scheme, each sensor is assigned a random subset of keys from a key pool before the network is deployed so that any two sensor nodes will have a certain probability of sharing at least one key. Chan et al. improved the above scheme and proposed the q-composite key predistribution scheme [9]. In the scheme, any two sensor nodes are required to share at least q predistributed keys as the basis for the establishment of a pairwise key between the two nodes. Liu and Ning proposed a framework in which pairwise keys are predistributed by using bivariate polynomials [10]. Du et al. proposed a similar pairwise key predistribution scheme [11] that uses Blom's method [12]. The main difference between the schemes in [10, 11] is that the former is based on a set of bivariate t-degree polynomials while the latter is based on the Blom's method.

Table 1: Properties of key establishment schemes.

Key setup schemes	Pairwise key	Local cluster key	New node joining	Mobility support
RKP	yes	no	yes	yes
LEAP	yes	yes	yes	no
OTMK	yes	no	yes	no
EDDK	yes	yes	yes	yes

Two representative deterministic key management schemes, that is, the LEAP protocol [3, 4], and the OMTK protocol [5], will be analyzed in the next section. In addition, Lai et al. proposed the BROadcast Session Key (BROSK) negotiation protocol [13]. Lee and Stinson proposed two deterministic schemes based on combinatorial design theory: the ID-based one-way function scheme (IOS) and the deterministic multiple space Blom's scheme (DMBS) [14]. They also discussed the use of combinatorial set systems in the design of deterministic key predistribution schemes for WSNs [15].

A comparison of the properties of the EDDK scheme with some existing key establishment schemes such as the Random Key Pre-distribution (RKP) schemes [8–11], the LEAP protocol [3, 4] and the OMTK scheme [5] is provided in Table 1. We can thus see that the RKP and the OTMK schemes do not support the establishment of local cluster keys while the LEAP and the OTMK schemes do not adapt well to node mobility.

2.3. Public Key Cryptography in WSNs. Prior studies have shown that it is feasible to apply public key cryptography to sensor networks by choosing proper algorithms with reasonable parameters and by using optimization and low-power techniques [16–18]. Gura et al. evaluated the assembly language implementations of Elliptic Curve Cryptography (ECC) and RSA on the Atmel ATmega128 processor [16] and showed that a 160-bit point multiplication of ECC only needs 0.81 seconds while a 1024-bit RSA public key operation and private key operation only need 0.43 seconds and 10.99 seconds, respectively. Du et al. developed a public key authentication scheme based on a symmetric key technique called the Merkle tree [19]. Zhang et al. proposed the notion of location-based keys by binding private keys of individual nodes to their IDs as well as locations [20].

In the method proposed by Zhou et al. [6], although ECC can be used together with timestamp to prevent an adversary from impersonating legitimate new nodes with compromised ones, the shortcomings of the method are equally obvious. Not only does the method incur communication overhead required for time synchronization, but it also limits the mobility of network nodes. This is because as soon as a legitimate node leaves its original location, it will be regarded as a malicious node [6]. As the result, the mobile node cannot establish shared keys with its new neighbors and will consequently be excluded from the network. Therefore, the access control protocol is susceptible to the so-called node location changing attack.

3. Analysis of Deterministic Key Schemes

In the LEAP protocol, a deterministic key sharing scheme based on symmetric key cryptography [3], the pairwise key establishment process between the neighboring nodes can be described as follows:

$$\begin{aligned} &u \longrightarrow *: \mathrm{ID}_u, \\ &v \longrightarrow u: \mathrm{ID}_v \mid\mid C(K_v, [\mathrm{ID}_u \mid\mid \mathrm{ID}_v]). \end{aligned} \tag{1}$$

For node u, the ACK from a neighboring node v is authenticated using the individual key K_v of node v derived through $K_v = f(K_{\mathrm{IN}}, \mathrm{ID}_v)$. Since it knows K_{IN}, node u can derive K_v and use it to verify the identity of node v. Node u can then compute a pairwise key K_{uv} with node v through $K_{uv} = f(K_v, \mathrm{ID}_u)$. Node v can also compute K_{uv} in the same way. K_{uv} can then serve as the pairwise key for nodes u and v.

In the above scheme, since the same formula is always used in computing the pairwise key, should an adversary get K_{IN} by capturing a sensor node with hardware faults, all pairwise keys in the network could be easily computed. Moreover, since the first step of the aforementioned process is not authenticated, an adversary can inject fake hello messages to launch a resource exhausting attack. The problem still remains in the improved version of the LEAP protocol, that is, the journal version LEAP+ [4].

The OTMK scheme [5] made some improvement over the LEAP protocol. When node u tries to set up pairwise keys with its neighbor nodes, it would broadcast the following message:

$$u \longrightarrow *: \mathrm{JOIN} \mid\mid E_M(\mathrm{ID}_u \mid\mid \mathrm{nonce}_u), \tag{2}$$

where nonce_u is a random number. If nodes u and v receive the JOIN request messages from each other, they will generate a pairwise key by using the following formula:

$$K_{u,v} = f(\mathrm{ID}_u \mid\mid \mathrm{ID}_v \mid\mid \mathrm{nonce}_u \mid\mid \mathrm{nonce}_v), \tag{3}$$

when $\mathrm{ID}_u < \mathrm{ID}_v$. If $\mathrm{ID}_u > \mathrm{ID}_v$, they will generate a pairwise key by using the following formula:

$$K_{v,u} = f(\mathrm{ID}_v \mid\mid \mathrm{ID}_u \mid\mid \mathrm{nonce}_v \mid\mid \mathrm{nonce}_u). \tag{4}$$

The above procedure still has some drawbacks, however. First, since all the nodes use the preconfigured master key M (M is similar to K_{IN} in the LEAP protocol) to encrypt the JOIN messages, if M is compromised through cryptanalysis, an adversary can easily deploy some malicious nodes into the network because function f is generally known. Moreover, the adversary can compute a large number of pairwise keys by eavesdropping on nonce and ID in the JOIN messages broadcast by other nodes so that it can overhear, forge, and alter network messages later.

The OTMK scheme uses the following protocol to allow new nodes to join an existing network [5]:

$$\begin{aligned} &u \longrightarrow *: \mathrm{JOIN} \mid\mid n_u \mid\mid \mathrm{ID}_u, \\ &v \longrightarrow u: \mathrm{ID}_v \| E_{K_v}(r_i \mid\mid n_v) \| \mathrm{MAC}_{K_v}(r_i \| n_u \mid\mid \mathrm{ID}_v \| n_v), \\ &u \longrightarrow v: \mathrm{ID}_u \mid\mid E_{K_v}(K_{u,v}) \mid\mid \mathrm{MAC}_{y_i}(n_v \mid\mid K_{u,v}). \end{aligned} \tag{5}$$

Since the first step above does not authenticate new node u, the protocol is vulnerable to denial of service (DoS) attacks. Let us consider the case in which a malicious node impersonates on a legitimate node u and broadcasts a JOIN message. A legitimate node v would then finish the second step. However, the malicious node may never execute the third step. By broadcasting many different spoof messages, the malicious node can incur a large amount of computation, communication, and storage overhead for node v. In addition, if an old node does not work properly after deployment due to many possible problems, an adversary can get the preloaded master key by compromising the node. The adversary can then deduce the individual keys (e.g., K_v and y_i) of all the old nodes and subsequently deploy some malicious nodes into the network. Since it has already obtained the master key, the adversary can get pairwise keys between the new and the old nodes by performing the third step in the above protocol.

We can see from the above analysis that not only is the new node joining protocol in the OTMK scheme susceptible to DoS attacks, but it is also possible for an adversary to freely deploy malicious nodes into the network and to obtain pairwise keys between the new and the old nodes.

In order to mitigate the compromise of the master key during the key setup phase, the OTMK scheme applies a mechanism that follows the same principle as that in the μTESLA protocol [21]. That is, it employs a one-way hash chain as the master key during each time period to authenticate new nodes.

$$\begin{aligned} &u \longrightarrow *: \mathrm{JOIN} \mid\mid n_u \mid\mid \mathrm{ID}_u, \\ &v \longrightarrow u: n_v \mid\mid \mathrm{ID}_v \mid\mid i \mid\mid \mathrm{MAC}_{K\,v}(n_u \mid\mid n_v \mid\mid \mathrm{ID}_v), \\ &u \longrightarrow v: \mathrm{ID}_u \mid\mid E_{K_v}(K_{u,v}) \mid\mid \mathrm{MAC}_{H_j}(n_v \mid\mid \mathrm{ID}_u \mid\mid K_{u,v}), \\ &u \longrightarrow *: H_j. \end{aligned} \tag{6}$$

The drawback of this mechanism is also obvious because it makes the protocol more complex while less secure. As long as an adversary can eavesdrop on all pairwise key setup procedures in time slot j and records $E_{K_v}(K_{u,v})$, the adversary can compute all pairwise keys in time slot j after H_j is received. Moreover, for the delayed authentication, the adversary can send a large number of joining messages to v by impersonating on u. As the result, v could store an excessive amount of messages that need to be authenticated in the third step, leading to a DoS attack.

4. EDDK: Energy-Efficient Distributed Deterministic Key Management Scheme

To fix the flaws present in the aforementioned key management schemes, we propose a distributed deterministic key management scheme (EDDK) for WSNs based on some of our earlier research [22–24]. EDDK mainly focuses on the establishment and maintenance of the pairwise keys as well as the local cluster keys. Unlike centralized and location-based key management schemes, EDDK does not depend on such infrastructure as base stations (BSs) and robots. Consequently, there is not any single node that shares a master key with the BS, which makes the scheme highly flexible. Therefore, EDDK can meet the requirement that there is only one resource-limited sink node in a large WSN. For the establishment and authentication of a global broadcast key, please refer to the μTESLA protocol [21].

To enhance message security in the network initialization phase, instead of using the individual key directly, each sensor node can derive a separate encryption key and a MAC key from the individual key [21]. This method can also be applied to enhance security in data transmission as well as in key update. For ease of illustration, in the following discussion, we simply use the same key to both encrypt and authenticate a message although two different keys can be used.

4.1. Notation. For the purpose of clarity, we first list and explain the symbols that will be used in our description.

(i) f and K_I are the networkwide shared pseudorandom function and the initial key, respectively, that are predistributed to every node before deployment.

(ii) K_a is node A's individual key and is computed by using formula (7) below; K_{ab} is the pairwise key that is shared by two neighboring nodes A and B and is periodically updated by them.

(iii) K_{B_a} is node A's local cluster key that is shared by node A with all its neighbors and is periodically updated by node A. This local cluster key is used to secure all local broadcast messages of a node, for example, routing control message forwarded.

(iv) P and s denote the public key and the private key of a node, respectively. The public key P_T of a trusted authority TA is predistributed to every node before deployment. But the private key s_T is not stored in the WSNs. Therefore, adversaries cannot get s_T by attacking TA directly.

(v) ID_S and ID_R are the identifiers of the sender and the receiver, respectively.

(vi) SN denotes the sequence number of a data message, that is, generally shared by two communicating nodes to fight against replay attacks and to ensure the freshness of data.

(vii) $E(K, M)$ is used to encrypt message M with key K; $C(K, M)$ is used to compute message M's message authentication code (MAC) with key K.

4.2. Key Establishment Phase. Before deployment, every sensor node is predistributed with a networkwide shared pseudorandom function f [25] and an initial key K_I to combat attacks in the network initialization phase. Through the pseudorandom function f and the initial key K_I, every node in the network can compute its individual key. For example, node E's individual key can be computed as follows:

$$K_e = f(K_I, \text{ID}_e). \tag{7}$$

After deployment, every sensor node will broadcast a network joining message to determine the neighboring relationships in the network. To prevent two or more nodes from attempting to transmit at the same time in the network initialization phase, every node would implement a binary exponential backoff algorithm to avoid collisions before broadcasting its joining message. In addition, to prevent an adversary from getting all the pairwise keys after obtaining K_I as it could in the LEAP protocol, any pairwise key that is computed through using the pseudorandom function should be updated in time.

As illustrated in Figure 1(a), we use nodes E and G to explain the pairwise key setup procedure:

$$\begin{aligned}
\text{E} \longrightarrow *&: \text{join} \mid\mid \text{ID}_e \mid\mid E(K_e, [\text{SN}_e \mid\mid K_{B_e}]) \mid\mid \\
&\quad C(K_e, [\text{ID}_e \mid\mid E(K_e, [\text{SN}_e \mid\mid K_{B_e}])]), \\
\text{G} \longrightarrow *&: \text{join} \mid\mid \text{ID}_g \mid\mid E\left(K_g, \left[\text{SN}_g \mid\mid K_{B_g}\right]\right) \mid\mid \\
&\quad C\left(K_g, \left[\text{ID}_g \mid\mid E\left(K_g, \left[\text{SN}_g \mid\mid K_{B_g}\right]\right)\right]\right),
\end{aligned} \tag{8}$$

$$K_{eg} = f\left(K_e \oplus K_g, \text{SN}_e \oplus \text{SN}_g\right). \tag{9}$$

In the above equations, join denotes the message type and || is the concatenation operator. Using (7), every receiving node can compute the sender's individual key (e.g., K_e and K_g). After verifying the correctness of the joining message, according to (9), the neighboring nodes can get the same pairwise key (e.g., K_{eg}). SN_e and SN_g are the random numbers generated by the senders that are mainly used to make the pairwise key between the neighboring nodes completely decentralized. In (8), authentication of the sender can counter resource exhaustion attacks. As long as the ID of a node is in the neighbor table, all repeated join messages will be discarded.

Our experiments show that the key establishment time is less than 10 s for a network of up to 20 neighbors. Deng et al. [5] showed that an attacker who has the experience and tools needs to spend at least tens of seconds or minutes to obtain the data of a Mica2 mote after a node is captured. Therefore, we set the key establishment timer for 10 s so that an adversary cannot impersonate a legitimate node in the key establishment phase through obtaining the initial key and the pseudorandom function. This is because when the timer runs out, key establishment will stop even under the situation in which some nodes have not established pairwise keys with some of their neighbors.

After the key establishment timer reaches its preset threshold value, a node will delete all the individual keys of its neighbors (e.g., K_e and K_g), the random numbers (e.g., SN_e and SN_g), the pseudorandom function f and the initial key K_I to improve security as well as to save storage space in the node. Therefore, even if an adversary could compromise some legitimate nodes, it still could not compute the pairwise keys and the local cluster keys of noncompromised nodes.

Note that the neighboring nodes do not have to acknowledge the setup of the pairwise keys because the measures taken later in the key maintenance phase will help the

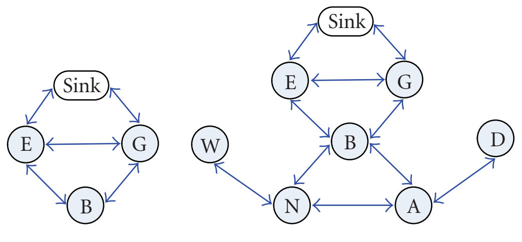

Figure 1: (a) Network initialization phase, (b) Network maintenance phase.

Table 2: Neighbor table.

2 Bytes	8 Bytes	2 Bytes	8 Bytes	2 Bytes
Node ID	Pairwise key	Pairwise sequence number	Local cluster key	Local broadcast sequence number

neighbors that failed to set up the pairwise keys successfully finish the task.

Note also that each sensor node only needs to broadcast one network joining message during the key establishment phase with no further message exchange required. Moreover, the network joining message may be the route request message. The pairwise key and the local cluster key can be established at the same time when a node forwards a route request message. Thus, the communication overhead can be very low.

4.3. Data Transfer Phase. In EDDK, we use a neighbor table to maintain the keys and the sequence numbers as shown in Table 2. The Sequence Number and the Key Lifetime fields are combined into one to save memory space in the node. For ease of use, we initialize both the pairwise sequence number and the local broadcast sequence number to be (7) The sequence number would be incremented after each message is sent out. When the sequence number reaches the predefined threshold, key update will be performed. After the key is updated, the sequence number will be reset to 1 again. The use of the neighbor table can help ensure the security of data transfer.

In EDDK, every message always gets authenticated while encryption is optional. Message confidentiality becomes necessary only when some information needs to be kept secret. For example, in an intrusion detection system, the actual content of an alarm message could be empty. Receiving an alarm message indicates an intrusion. Thus, encryption is not necessary but only leading to increase in latency, computation overhead, and power consumption. Meanwhile, authentication helps to ensure that authorized nodes will not accept invalid messages injected by an adversary. Consequently, adversaries can hardly forge false alarms.

(i) Authentication mode only:

$$\begin{aligned} &\text{Unicast: } \mathrm{ID}_S \,||\, \mathrm{ID}_R \,||\, \mathrm{SN} \,||\, \text{message} \,|| \\ &\qquad C(K, [\mathrm{ID}_S \,||\, \mathrm{ID}_R \,||\, \mathrm{SN} \,||\, \text{message}]). \\ &\text{Acknowledgment: } \mathrm{ID}_S \,||\, \mathrm{ID}_R \,||\, \mathrm{SN}+1 \,|| \\ &\qquad C(K, [\mathrm{ID}_S \,||\, \mathrm{ID}_R \,||\, \mathrm{SN}+1]). \end{aligned} \tag{10}$$

$$\begin{aligned} &\text{Local Broadcast: } \mathrm{ID}_S \,||\, \mathrm{SN}_B \,||\, \text{message} \,|| \\ &\qquad C(K_B, [\mathrm{ID}_S \,||\, \mathrm{SN}_B \,||\, \text{message}]). \end{aligned} \tag{11}$$

In the above process, K is the pairwise key that is shared by the sender and the receiver, and K_B is the local cluster key of the sender. SN is the pairwise sequence number while SN_B is the local broadcast sequence number. Since the sequence numbers are used to ensure the freshness of a message and to counter replay attacks, it is not necessary to keep them secret. It is obvious that the local broadcast does not need the acknowledgment message, so it is omitted in (11).

(ii) Encryption and authentication mode:

$$\begin{aligned} &\text{Unicast: } \mathrm{ID}_S \,||\, \mathrm{ID}_R \,||\, \mathrm{SN} \,||\, E(K, \text{message}) \,|| \\ &\qquad C(K, [\mathrm{ID}_S \,||\, \mathrm{ID}_R \| \mathrm{SN} \| E(K, \text{message})]). \\ &\text{Acknowledgment: } \mathrm{ID}_S \,||\, \mathrm{ID}_R \,||\, \mathrm{SN}+1 \,|| \\ &\qquad C(K, [\mathrm{ID}_S \,||\, \mathrm{ID}_R \,||\, \mathrm{SN}+1]). \end{aligned} \tag{12}$$

$$\begin{aligned} &\text{Local Broadcast: } \mathrm{ID}_S \,||\, \mathrm{SN}_B \,||\, E(K_B, \text{message}) \,|| \\ &\qquad C(K_B, [\mathrm{ID}_S \,||\, \mathrm{SN}_B \,||\, E(K_B, \text{message})]). \end{aligned} \tag{13}$$

To save energy, a sensor node may employ early rejection in which it would turn off its radio after determining that a message is false or malicious. Upon the receipt of a message, a receiver would first examine if the sender of the message is in its neighbor table and if the SN is the same as that in its neighbor table; it would then compute the message authentication code and compare the result with the received MAC; it would finally decrypt the message only when all the previous steps have completed successfully. Hence, the use of the neighbor table not only improves the security of data transfer but also helps to reduce computation and communication overhead in the node.

4.4. Key Maintenance Phase. The security strength on the information transmitted in the network relies primarily on the protection of the keys that are used for encryption. Should an attacker get the keys, the security system would fail because the attacker can use the keys to decrypt intercepted cipher texts to get the original plaintexts or to fake network data to deceive legitimate nodes. The attacker could achieve the above goal through the means of cryptanalysis on eavesdropped messages that are transmitted over the wireless media or by compromising some sensor nodes in the network. Therefore, the sender and the receiver must update the keys used between them periodically. When a compromised node is identified, the keys that it uses must also be revoked immediately. Moreover, key management system must quickly respond to network changes to perform functions such as key establishment and maintenance of new nodes and mobile nodes.

4.4.1. Key Update. To defend against cryptanalysis and to prevent adversaries from decrypting all previous messages after one or more sensor node are compromised, all keys must be updated periodically. When the lifetime of a pairwise key (i.e., the SN) is reached, two neighboring nodes need to perform the update of the neighbor table. The node whose ID is smaller would invoke the update procedure shown in (14) and the neighbor table will be updated as the result. Similarly, when the lifetime of the local cluster key of a node (i.e., the SN_B) reaches the preset value, the node will broadcast the cluster key update message.

(i) Pairwise key update

If $\text{ID}_S < \text{ID}_R$

$$\begin{aligned}&\text{Unicast: pairwise-key} \mid\mid \text{ID}_S \mid\mid \text{ID}_R \mid\mid \text{SN} \mid\mid E(K, K') \mid\mid \\ &\qquad C(K, [\text{ID}_S \mid\mid \text{ID}_R \mid\mid \text{SN} \mid\mid E(K, K')]). \\ &\text{Acknowledgment: } \text{ID}_S \mid\mid \text{ID}_R \mid\mid \text{SN}+1 \mid\mid \\ &\qquad C(K', [\text{ID}_S \mid\mid \text{ID}_R \mid\mid \text{SN}+1]),\end{aligned} \tag{14}$$

where *pairwise-key* denotes the message type and K and K' are the old and the new pairwise keys, respectively.

(ii) Local cluster key update

$$\begin{aligned}&\text{Local Broadcast: cluster-key} \mid\mid \text{ID}_S \mid\mid \text{SN}_B \mid\mid E(K_B, K_B') \mid\mid \\ &\qquad C(K_B, [\text{ID}_S \mid\mid \text{SN}_B \mid\mid E(K_B, K_B')]),\end{aligned} \tag{15}$$

where *cluster-key* denotes the message type and K_B and K_B' are the old and the new local cluster keys, respectively.

The above key update procedures can counter replay, forgery and tampering attacks. Thus, the keys can be updated both securely and timely.

4.4.2. Compromised Key Revocation. When one or more sensor nodes are compromised and can thus become insider attackers, measures must be taken to segregate them from the network so that they cannot eavesdrop on the information transmitted in the network and launch active attacks towards the network. We assume here that the intrusion detection system used in the network can discover compromised nodes and then inform the key management scheme to take actions to isolate these nodes. First, the neighbors of the compromised nodes should revoke the pairwise keys shared with the compromised nodes. Second, the local cluster key K_B must be updated. Here, we use the pairwise keys possessed by noncompromised nodes to encrypt the update messages:

$$\begin{aligned}&\text{Unicast: cluster-key} \mid\mid \text{ID}_S \mid\mid \text{ID}_R \mid\mid \text{SN} \mid\mid E(K, K_B) \mid\mid \\ &\qquad C(K, [\text{ID}_S \mid\mid \text{ID}_R \mid\mid \text{SN} \mid\mid E(K, K_B)]). \\ &\text{Acknowledgment: } \text{ID}_S \mid\mid \text{ID}_R \mid\mid \text{SN}+1 \mid\mid \\ &\qquad C(K, [\text{ID}_S \mid\mid \text{ID}_R \mid\mid \text{SN}+1]).\end{aligned} \tag{16}$$

As long as the noncompromised nodes are still connected, the local cluster key K_B can be successfully updated.

4.4.3. New Node Joining. To solve the problems present in the schemes in [3–5], we use the Elliptic Curve Digital Signature Algorithm (ECDSA) [26] to authenticate a new node without any negative impact on energy efficiency. In Figure 1(b), let us assume that B is the existing node and N and W are new joining nodes. When a new node tries to set up the pairwise keys with its neighbors, it broadcasts a joining message:

$$\begin{aligned}&\text{N} \longrightarrow *: \text{new-join} \mid\mid \text{ID}_n \mid\mid P_n \mid\mid T_n \mid\mid \text{SN}_n \mid\mid C_n \mid\mid c_n, \\ &\text{W} \longrightarrow *: \text{new-join} \mid\mid \text{ID}_w \mid\mid P_w \mid\mid T_w \mid\mid \text{SN}_w \mid\mid C_w \mid\mid c_w.\end{aligned} \tag{17}$$

Let us now use node N as an example to explain the fields in (17), new-join indicates that this is a new node joining message. P_n is the public key of node N. T_n is the timestamp at which the node should join the network. Node N will be treated as a new node when current time $t < T_n$. SN_n is the random number generated by node N that will be used to compute the pairwise key between two new nodes. The signature $\langle C_n, c_n \rangle$ is calculated by the trusted authority TA based on the ECDSA algorithm [26]. The calculating process is shown as follows:

$$\begin{aligned}C_n &= r_n G = (x_{cn}, y_{cn}), \\ c_n &= r_n^{-1}(H(\text{ID}_n \mid\mid P_n \mid\mid T_n) + s_T x_{cn}) (\operatorname{mod} n),\end{aligned} \tag{18}$$

where r_n is a random number, G is the generator in the cyclic group of points over the elliptic curve and has an order n of at least 160 bits, H is a hash function that can translate a binary sequence into an integer, and s_T is the private key of the TA.

The receiving nodes, both W and B, will each perform the following calculation:

$$V_n = c_n^{-1}(H(\text{ID}_n \mid\mid P_n \mid\mid T_n) G + c_n^{-1} x_{cn} P_T, \tag{19}$$

where $P_T = s_T G = (x_{pT}, y_{pT})$ and $P_n = s_n G = (x_{pn}, y_{pn})$. If $V_n = C_n$, then N's neighbors can be assured that N is a legitimate node and the verification process is illustrated below:

$$\begin{aligned} V_n &= c_n^{-1} r_n r_n^{-1} (H(\mathrm{ID}_n \,||\, P_n \,||\, T_n) + x_{cn} s_T) G \\ &= c_n^{-1} r_n c_n G = C_n. \end{aligned} \tag{20}$$

Following the same procedure, the receivers can also verify the identity of the new node W after hearing the broadcast joining messages from it. The pairwise key negotiation would fall into one of the following two scenarios: between two new nodes or between a new node and an existing node.

New nodes are preloaded with the pseudorandom function f and a new initial key K_I'. The pairwise key negotiation between two new nodes is performed using (9) in which $K_n = f(K_I', \mathrm{ID}_n)$ and $K_w = f(K_I', \mathrm{ID}_w)$. Consequently, new nodes N and W can establish a pairwise key without having to send out any more messages and $K_{nw} = f(K_n \oplus K_w, \mathrm{SN}_n \oplus \mathrm{SN}_w)$.

Let us now use nodes N and B to illustrate pairwise key negotiation between a new node and an existing one. After authenticating the identity of the new node N, the existing node B would broadcast a response message with the signature $\langle C_b, c_b \rangle$ of the TA:

$$\mathrm{B} \longrightarrow *: \text{reply} \,||\, \mathrm{ID}_b \,||\, P_b \,||\, T_b \,||\, C_b \,||\, c_b. \tag{21}$$

According to the Diffie-Hellman algorithm [27] over the elliptic curve discrete logarithm problem (ECDLP), nodes N and B could independently calculate the pairwise key using their own private keys and the other's public key as follows:

$$K_{nb} = s_n P_b = s_b P_n. \tag{22}$$

The results will be identical because $s_n P_b = s_n s_b G = s_b s_n G = s_b P_n$.

No further message exchanges are necessary during pairwise key setup beyond broadcasting the identity message by the new node N. Therefore, the communication overhead is low in the new node joining process.

Although the public-key algorithm is computationally more expensive, it is easier to manage and more resilient to node compromise than secret key algorithms. To achieve the same level of security, a smaller key size in ECC (elliptic curve cryptography) can be used, which offers the advantages of faster computation as well as savings in memory, energy, and bandwidth. Thus, ECC is more suitable for resource-constrained sensor nodes.

4.4.4. Mobile Node Joining. We assume that a sensor node can sense its own movement. If a sensor node moves to a new location, it will establish pairwise keys with its new neighbors following the aforementioned public-key-based node joining procedure, and its old neighbors will also revoke the pairwise keys with it.

In Figure 1(b), let us suppose that B is the existing node, N and W are new joining nodes, and A and D are mobile nodes. After nodes A and D move to a new location, each will broadcast a network joining message independently as shown below:

$$\begin{aligned} &\mathrm{A} \longrightarrow *: \text{rejoin} \,||\, \mathrm{ID}_a \,||\, P_a \,||\, T_a \,||\, C_a \,||\, c_a, \\ &\mathrm{D} \longrightarrow *: \text{rejoin} \,||\, \mathrm{ID}_d \,||\, P_d \,||\, T_d \,||\, C_d \,||\, c_d, \end{aligned} \tag{23}$$

where rejoin shows that this is a mobile node joining message. Since current time t is newer than both timestamps T_a and T_d, nodes A and D will be regarded as mobile nodes. After correctly verifying each other's joining message, the pairwise key computation procedure between two mobile nodes is the same as that in (22), that is, $K_{ad} = s_a P_d = s_d P_a$. Similarly, according to (21), (22), and (23), the pairwise key between mobile node A and the existing node B is $K_{ab} = s_a P_b = s_b P_a$ and, according to (17), (22), and (23), the pairwise key between mobile node A and the new node N is $K_{an} = s_a P_n = s_n P_a$. In fact, only the pairwise key between a mobile node and an existing node would normally need to be computed.

The mobile nodes could be normal nodes whose locations have changed due to some reasons. However, it is also possible that the mobile nodes are the replicas of some compromised nodes that are placed in the network by adversaries. Hence, the intrusion detection system in a sensor network should focus on monitoring the mobile nodes whose locations have changed. When the behavior of a mobile node matches an attack profile or deviates from a normal profile, the mobile node should be labeled as a malicious node.

In order to counter Sybil attacks and the node replication attacks, we limit the number of neighbors for any node, that is, the number of entries in the neighbor table, by using a threshold. When the number of neighbors of a node reaches the threshold value, no new neighbors can be added into the neighbor table. Thus, an affected new mobile node can not establish pairwise keys with its new neighbors and, as the result, could not communicate with them. This measure can help prevent replicas of compromised nodes from joining the network as mobile nodes.

In the above joining scenarios, after the pairwise key is established, the existing node needs to unicast its local cluster key to its new neighbors to complete the key establishment process.

5. Security Analysis

We first compare the security of EDDK with some related schemes and then analyze EDDK in terms of preventing both general and special attacks to WSNs.

5.1. Security Comparison with Related Schemes. In the LEAP+ protocol [4], when the timer expires after $T_{\min}$, each node will erase the initial key K_{IN} and all the master keys of its neighbors. However, even with tamper-proof hardware, damaged or disabled nodes may still have the initial key in the flash memory. Therefore, should an adversary physically capture such a node and obtain the initial key, all the pairwise keys in the network could be easily computed. From (8) and (9), we can see that all pairwise keys in EDDK are decentralized and cannot be inferred uniformly based on any centralized rule. Even if the adversary could know the pseudorandom function f and the initial key K_I, it would not be able to get noncompromised pairwise keys since the random number SN generated by each node in (9) is different.

The new node joining protocol in the OMTK scheme [5] not only is susceptible to DoS attacks but also makes it possible for an adversary to inject malicious nodes into the network and to obtain pairwise keys between new and existing nodes. The additional mechanism that uses a one-way hash chain as the master key during each time period to authenticate new nodes only makes the protocol more complex while less secure. To solve the above problems, we employ ECDSA to authenticate the identity of any node. Even if an adversary could compromise a node, it would not get any information about noncompromised nodes.

5.2. Preventing General Attacks on WSNs. General attacks to WSNs mainly include eavesdropping, faking, altering, and replaying network messages as well as launching DoS attacks towards the normal nodes.

In both the network initialization phase and the network maintenance phase, we use the node's individual key, the pairwise key, or the local cluster key to encrypt and authenticate network messages. Therefore, adversaries who do not know these keys cannot obtain, fake, or tamper with the contents of network messages.

Due to the use of the neighbor table, adversaries can hardly launch replay attacks. For the node joining messages, if a node exists in the neighbor table, any duplicate joining messages claimed to be from it will be discarded by the receiving node. For the data messages, by using the sequence number field in the neighbor table, any duplicate data messages will also be discarded by the receiver.

Based on (8), (10)–(17), (21), and (23), key establishment messages, node sensing messages, and key maintenance messages can be authenticated. Furthermore, key establishment messages and new and mobile node joining messages are not necessarily fed back. Thus, EDDK can effectively counter DoS attacks.

5.3. Preventing Specific Attacks to WSNs. Specific attacks to WSNs mainly include rushing attacks, Sybil attacks, node replication attacks, hello flooding attacks, acknowledge spoofing attacks, and selective forwarding attacks, to name a few [1, 28].

Every node needs to broadcast a "hello" message periodically to advertise itself. The sequence numbers and the pairwise keys stored in the neighbor table can help to authenticate the "hello" messages and the acknowledgment messages. As the result, an adversary who does not know the shared information cannot flood a large number of "hello" messages and forge acknowledgment messages.

To counter selective forwarding attacks, we adopt a measure similar to that used in [29] in which each sensor node can work under a promiscuous mode so that it can overhear data transmission among its neighboring nodes. If a neighbor of a suspected node detects that the number of messages that the suspected node fails to forward exceed a threshold, the neighbor can collaborate with other neighbors to make a decision about the suspected node. In addition, multipath routing can also help to deal with the selective forwarding problem.

Because we allow mobile nodes to rejoin the network, the replicas of compromised nodes may affect EDDK. By limiting the number of neighbors for any node and by authenticating all messages using the sequence numbers and the pairwise keys, we can prevent an intruder from launching Sybil attacks, rushing attacks, and node replication attacks to some extent. Moreover, we can use multiple paths to the sink node to counter such attacks. In a sensor network where nodes are mostly stationary, the intrusion detection system can focus on monitoring mobile nodes whose locations have just changed. If the behavior of a mobile node matches an attack profile or deviates from a normal profile, the node will be regarded as a malicious node.

6. Performance Evaluation

In the probabilistic key sharing schemes, if two nodes have to rely on a third node to help them establish a pairwise key, a large amount of computation (e.g., encryption and decryption) and communication will be required due to the fact that threshold cryptography-based schemes usually involve expensive operations such as modular multiplications. In contrast, in EDDK, every node can establish a pairwise key with each of its neighbors and thus does not need any additional help in pairwise key establishment.

We have used MATLAB 7.9.0 (R2009b) to perform the simulation to evaluate the EDDK scheme and to compare it with some other similar schemes. We have also implemented EDDK in the Crossbow's IRIS sensor nodes. We have chosen RC5 (with 12 rounds) as the block cipher to implement the encryption/decryption algorithm and to generate CBC-MAC. We have also used MAC with RC5 to provide the pseudo random function that is used to derive the individual keys as well as the pairwise keys.

We have evaluated the EDDK scheme in terms of computation overhead, communication overhead, and storage overhead.

6.1. Computation Overhead. In the network initialization phase, the computation overhead for each node includes the encryption and authentication of the local broadcast message, the verification and decryption of the received messages from neighbors, and the computation of the pseudorandom function in (8) and (9). Using an IRIS node with the XMTS300CB sensor board running the TinyOS system and a default message size of 36 bytes, we count the time cost resulting from these operations for a local broadcast message for 20 times. As shown in Figure 2, these operations do not incur too much computation overhead.

Although the ECC algorithm is adopted in the node mobility scenario, for achieving the same level of security, a smaller key size in ECC can offer faster computational efficiency, especially in assembly language implementations, which only require 0.81 s for completing a 160-bit point

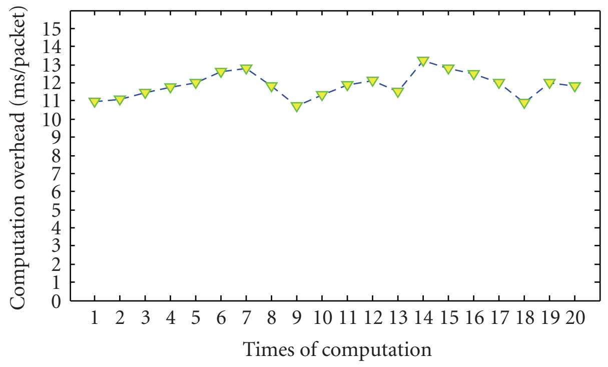

Figure 2: Computation time statistics from sending a local broadcast message to receiving the message by another node.

multiplication of ECC [16]. Node authentication needs two point multiplication operations according to (19), whereas pairwise key setup needs only one point multiplication operation according to (22) in the node mobility scenario.

6.2. Communication Overhead. A comparison of the communication overhead among the EDDK, LEAP, and OTMK schemes in the network initialization phase is shown in Figure 3 in which m denotes the average number of neighbors for a node. In the LEAP scheme and the OTMK Basic scheme I, it requires that each node unicast a reply message to the node that initially broadcast the joining message, resulting in more messages to be sent and received by the nodes in addition to incurring additional energy consumption and delaying key setup. However, in the pairwise key setup procedure of the EDDK scheme and the OTMK Simplified scheme I, each node only needs to broadcast a network joining message without requiring any reply message. So, from Figure 3, we can see that the communication overhead for each node is very low in the latter two schemes.

A comparison of communication overhead among EDDK, LEAP, and OTMK schemes in the new node joining phase is shown in Figure 4 in which n denotes the number of new nodes that simultaneously appear in the neighboring area. In the LEAP and the OTMK schemes, it requires that each existing node unicast a feedback message to the new neighbor node for the authentication of the identities of each other, resulting in additional energy consumption and key establishment delay. In addition, these two schemes failed to consider the situation in which few new nodes simultaneously appear in the neighborhood. Therefore, their communication overhead is greater than that of EDDK.

According to (8), (10)–(17), (21), and (23) in the EDDK scheme, messages sent by every node can be authenticated while the schemes in [3–5] cannot. Should a message not be authenticated by the recipient, an adversary could freely inject malicious messages into the network. Moreover, the recipient has to feed reply messages into the network and wait for further messages in order to authenticate the sender. Consequently, the communication overhead in EDDK is

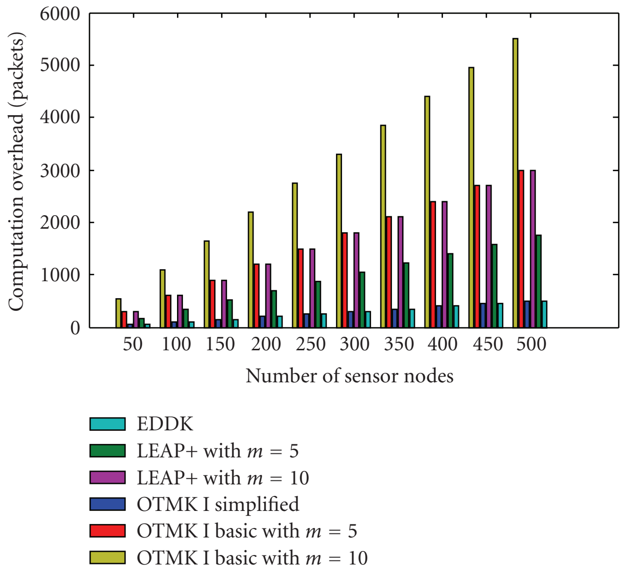

Figure 3: Comparison of communication overhead among EDDK, LEAP, and OTMK schemes in the network initialization phase.

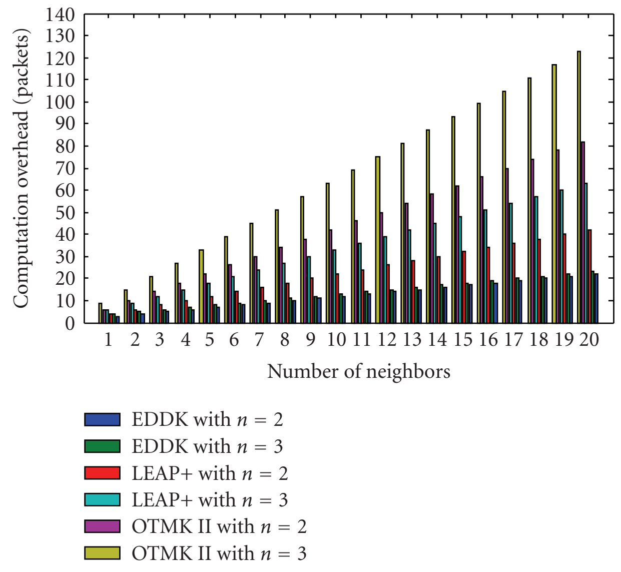

Figure 4: Comparison of communication overhead among EDDK, LEAP, and OTMK schemes in the new node joining phase.

much lower than that in the other schemes. Moreover, communication overhead is generally considered to be much worse, and therefore less desirable, than computation overhead due to higher energy consumption.

6.3. Storage Overhead. In the EDDK scheme, each node needs to store the pairwise keys and the local cluster keys related to its neighbors. Since in the mobility scenario, we employ the ECC-based public key mechanism to verify the identities of the nodes and to establish pairwise keys, each node still needs to store its own private and public keys, the TA's signature for it and the TA's public key. In our

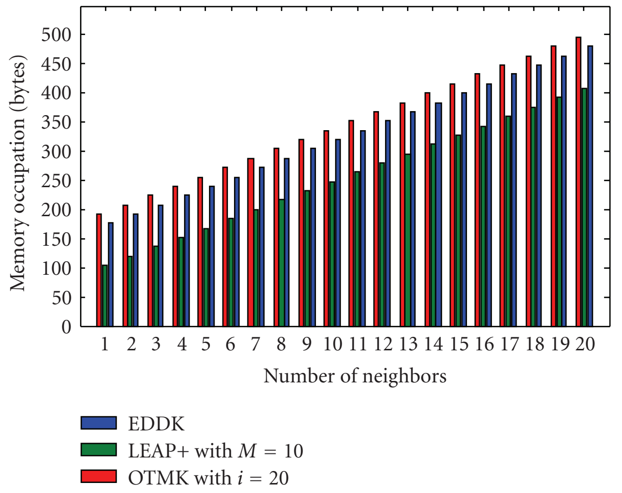

Figure 5: Comparison of storage overhead for a node among EDDK, LEAP and OTMK schemes.

current research, since we use 160-bit ECC as the underlying cryptographic foundation, the key materials related to the public key need 160-byte storage space. The LEAP, and the OTMK schemes do not adopt the public key mechanism, however. In addition to the pairwise keys and the local cluster keys, in the LEAP protocol, each node still needs to store its own individual key that is used for computing the pairwise keys and at most M initial keys: $K_{\mathrm{IN}}^1, K_{\mathrm{IN}}^2, \ldots, K_{\mathrm{IN}}^M$ that are used for deriving the individual key in different node addition scenarios. The OTMK scheme does not support the establishment of a local cluster key. In our comparison with these schemes, we assume that, in the OTMK scheme, a node already has the local cluster key. Moreover, to allow new normal nodes to join an existing network, in OMTK, a node has to generate i verifiers that contain random numbers r_i and y_i.

A comparison of storage overhead for a node among EDDK, LEAP and OTMK schemes is shown in Figure 5 in which we can see that although EDDK has adopted the public key mechanism ECC, the storage overhead for a node in EDDK is no more than that in the LEAP and the OTMK schemes.

We can thus conclude that the EDDK scheme is more advantageous in computation, communication, and storage and is thus more suitable for resource-constrained wireless sensor networks.

7. Conclusions

In this paper, we proposed an energy-efficient distributed deterministic key management scheme (EDDK) in which pairwise keys and local cluster keys of sensor nodes can be established and maintained securely. Furthermore, pairwise keys are totally decentralized. Hence, the compromise of any sensor node will not affect any other noncompromised pairwise keys. By broadcasting only one identity authentication message, a node can set up a pairwise key with a neighboring node both in the network initialization phase and in the node mobility scenario. Therefore, the communication overhead is very low. In addition, not only can the neighbor table mechanism used in EDDK ensure security for key management and data transmission, but it can also help nodes effectively store and update keys. The use of the elliptic curve digital signature algorithm in EDDK provides the support for the establishment of pairwise keys and local cluster keys under the node mobility scenario. Simulation and analysis has shown that EDDK is more advantageous in computation, communication, and storage than other similar schemes.

Acknowledgments

This research work in this paper has been partially supported by a Grant from the Science Foundation of Beijing Education Commission (Grant no. KM201010005027). The authors would also like to thank the anonymous reviewers for their valuable comments on the paper.

References

[1] Y. Zhou, Y. Fang, and Y. Zhang, "Securing wireless sensor networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 3, pp. 6–28, 2008.

[2] Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 8, no. 2, pp. 2–22, 2006.

[3] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 62–72, Washington, DC, USA, October 2003.

[4] S. Zhu, S. Setia, and S. Jajodia, "LEAP+: efficient security mechanisms for large-scale distributed sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 4, pp. 500–528, 2006.

[5] J. Deng, C. Hartung, R. Han, and S. Mishra, "A practical study of transitory master key establishment for wireless sensor networks," in *Proceedings of the 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm '05)*, pp. 289–299, Athens, Greece, September 2005.

[6] Y. Zhou, Y. Zhang, and Y. Fang, "Access control in wireless sensor networks," *Ad Hoc Networks*, vol. 5, no. 1, pp. 3–13, 2007.

[7] R. Di Pietro, L. V. Mancini, Y. W. Law, S. Etalle, and P. Havinga, "LKHW: a directed diffusion-based secure multicast scheme for wireless sensor networks," in *Proceedings of the International Conference on Parallel Processing*, pp. 397–406, Los Alamitos, Calif, USA, October 2003.

[8] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 41–47, Washington, DC, USA, November 2002.

[9] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of IEEE Symposium on Security And Privacy*, pp. 197–213, Oakland, Calif, USA, May 2003.

[10] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 52–61, Washington, DC, USA, October 2003.

[11] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Transactions on Information and System Security*, vol. 8, no. 2, pp. 228–258, 2005.

[12] R. Blom, "An optimal class of symmetric key generation systems," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '84)*, pp. 335–338, Paris, France, April 1984.

[13] B. Lai, S. Kim, and I. Verbauwhede, "Scalable session key construction protocols for wireless sensor networks," in *Proceedings of IEEE Workshop on Large Scale RealTime and Embedded Systems*, 2002.

[14] J. Lee and D. R. Stinson, "Deterministic key predistribution schemes for distributed sensor networks," in *Proceedings of the 11th International Workshop on Selected Areas in Cryptography (SAC '04)*, pp. 294–307, Waterloo, Canada, August 2004.

[15] J. Lee and D. R. Stinson, "A combinatorial approach to key predistribution for distributed sensor networks," in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC '05)*, pp. 1200–1205, Orleans, La, USA, March 2005.

[16] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and RSA on 8-Bit CPUs," in *Proceedings of the 6th International workshop on Cryptographic Hardware and Embedded Systems (CHES '04)*, pp. 119–132, Cambridge, Mass, USA, August 2004.

[17] A. S. Wandert, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom '05)*, pp. 324–328, Kauai Island, Hawaii, USA, March 2005.

[18] G. Gaubatz, J.-P. Kaps, and B. Sunar, "Public key cryptography in sensor networks-revisited," in *Proceedings of the 1st European Workshop on Security in Ad-Hoc and Sensor Networks*, pp. 2–18, Heidelberg, Germany, August 2005.

[19] W. Du, R. Wang, and P. Ning, "An efficient scheme for authenticating public keys in sensor networks," in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC '05)*, pp. 58–67, Urbana-Champaign, Ill, USA, May 2005.

[20] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Location-based compromise-tolerant security mechanisms for wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 247–260, 2006.

[21] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[22] X. Zhang, J. He, and Q. Wei, "An energy-efficient dynamic key management scheme in wireless sensor networks," in *Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms, and Networks (I-SPAN '09)*, pp. 238–242, Kaohsiung, Taiwan, December 2009.

[23] X. Zhang, J. He, and Q. Wei, "Neighbor-list based pairwise key management scheme in wireless sensor networks," in *Proceedings of the 5th International Conference on Active Media Technology*, pp. 522–528, Beijing, China, October 2009.

[24] X. Zhang, J. He, and Q. Wei, "Security considerations on node mobility in wireless sensor networks," in *Proceedings of the 4th International Conference on Computer Sciences and Convergence Information Technology (ICCIT '09)*, pp. 1143–1146, Seoul, Korea, November 2009.

[25] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *Journal of the ACM*, vol. 33, no. 4, pp. 792–807, 1986.

[26] S. Vanstone, "Responses to NIST's proposal," *CACM*, vol. 35, no. 7, pp. 50–52, 1992.

[27] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[28] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," in *Proceedings of the 1st International Workshop on Sensor Network Protocols and Applications*, pp. 113–127, Anchorage, Alaska, USA, May 2003.

[29] G. Wang, W. Zhang, G. Cao, and T. La Porta, "On supporting distributed collaboration in sensor networks," in *Proceedings of IEEE Military Communications Conference (MILCOM '03)*, pp. 752–757, Boston, Mass, USA, October 2003.