Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 406254, 14 pages
doi:10.1155/2012/406254

Research Article

An Efficient Key Management Scheme for Wireless Sensor Networks

Dahai Du, Huagang Xiong, and Hailiang Wang

School of Electronics and Information Engineering, Beijing University of Aeronautics and Astronautics, Beijing 100191, China

Correspondence should be addressed to Dahai Du, seahunter@sina.com

Received 13 April 2011; Revised 23 July 2011; Accepted 19 September 2011

Academic Editor: Ruixin Niu

Wireless sensor networks (WSNs) can be used in a wide range of environments. Due to the inherent characteristics of wireless communications, WSNs are more vulnerable to be attacked than conventional networks. Authentication and data confidentiality are critical in these settings. It is necessary to design a useful key management scheme for WSNs. In this paper, we propose a novel key management scheme called MAKM (modular arithmetic based key management). The proposed MAKM scheme is based on the congruence property of modular arithmetic. Each member sensor node only needs to store a key seed. This key seed is used to compute a unique shared key with its cluster head and a group key shared with other nodes in the same cluster. Thus, MAKM minimizes the key storage space. Furthermore, sensor nodes in the network can update their key seeds very quickly. Performance evaluation and simulation results show that the proposed MAKM scheme outperforms other key-pool-based schemes in key storage space and resilience against nodes capture. MAKM scheme can also reduce time delay and energy consumption of key establishment in large-scale WSNs.

1. Introduction

Wireless sensor networks (WSNs) consist of a large number of tiny, cheap, computational, and energy-constrained sensor nodes. They have gained numerous applications such as monitoring, target tracking, and military fields for data gathering and processing. Sensor nodes are randomly deployed in special areas to collect information. Since the data packets are transmitted over the air, it is not difficult for the adversaries to steal information by eavesdropping. To guarantee the confidentiality of information transmitted between sensor nodes, the messages should be encrypted before being transmitted. It is a great challenge to implement encryption algorithms in wireless sensor networks because of the constrained resource. Furthermore, malicious nodes may impersonate to be legitimate nodes and communicate with other valid nodes. This may subvert the whole networks. Therefore, a lightweight and efficient key management scheme should be used to solve all these problems mentioned above.

Recently, many approaches have been proposed to solve these problems. These solutions can be mainly classified into two categories: static and dynamic key management schemes. In static key management schemes, all keys are predistributed in sensor nodes. Each node chooses a set of keys from a key pool. In the bootstrapping phase, each node finds shared keys with its neighbors. If they cannot find a common key, they will set up a shared key with the help of one or more intermediate nodes. This can be found in [1–8]. These schemes are based on the probabilistic key predistribution that guarantees a high probability of sharing keys between nodes. On the other hand, in dynamic key management schemes, some keys or key seeds are predistributed in sensor nodes. The session keys are established on demand. In most cases, there are some nodes acting as group heads or gateways to establish session keys for two nodes. These intermediate nodes are usually more powerful than other member nodes in terms of energy supply, transmission range, data processing capability, storage capacity, and tamper resistance. This can be found in [9–12].

The proposed schemes are mainly based on two types of network structure. One is the peer-to-peer (P2P) network. All nodes in the network have the same capability [1–3, 7–9, 11, 12]. The other is a hierarchical structure [4–6, 10, 13],

in which some nodes are more powerful in computational capability and have larger storage capacity than common nodes. We mainly discuss a heterogeneous wireless sensor network in this paper. There are two different kinds of sensor nodes in the discussed network. The more powerful nodes act as cluster heads. The discussed WSN can be used in a wide range of applications such as military fields, aircrafts health monitor, and medical care. It can be used to collect useful and secret data. For example, sensor nodes can be deployed to monitor the status of all end systems in an aircraft. A cluster of nodes can monitor an end system.

We can find that many key-pool based schemes take much storage space. For example, in E-G scheme [1], we assume that a key ring has 100 keys and a key length is 16 Bytes (128 bits). Thus, a sensor node needs 1.6 KB to store its keys. However, a typical sensor node has only 4 KB of RAM [12]. On the other hand, dynamic key management schemes such as [11, 12] cost too much energy. Thus, it is necessary to design a key management scheme to balance the tradeoff between key storage space and energy consumption. In this paper, we present a novel solution of key management problem in WSNs. The proposed key management scheme is based on the congruence property of modular arithmetic. Each member node has a unique integer as its key seed, which can be used to calculate a shared key with the cluster head and a group key shared with all other member nodes in the same cluster. The shared group key can be used for broadcast encryption. This is especially efficient when a new node joins the network because the cluster head only needs to broadcast a key update message to finish updating the group key and its shared keys with each member node in its cluster. Other session keys are established on demand. Since each node only needs to store a single key seed, our scheme minimizes the key storage space. What is more, we add a mutual authentication during the key establishment phase, and the identity of each node is verified in our solution.

The remainder of this paper is organized as follows. Section 2 gives an introduction of pertinent related work. Section 3 details our novel key management scheme. The detailed analysis of performance evaluation is given in Section 4. Section 5 presents the simulation results. We compare our proposed scheme with other schemes by using NS-2. Finally, we draw the conclusions of this paper in Section 6.

2. Related Work

Eschenauer and Gligor [1] propose a key management scheme based on probabilistic key sharing among the nodes of random graph. Key distribution consists of three phases: key predistribution, shared-key discovery, and path key establishment. In key predistribution phase, each node randomly selects k keys from a key pool. In the shared-key discovery phase, each sensor node discovers the shared keys with its neighbors within its transmission range. If two nodes have no shared keys, they will establish a shared key via two or more links in path key establishment phase. In this scheme, once a node is compromised, the key ring will be revoked. What is more, these keys should also be removed from other node key rings. This will decrease the link connectivity of the network, and the affected nodes need to reconfigure their links.

Chan et al. [2] propose three mechanisms for sensor networks. In [2], Chan et al. propose a q-composite random key predistribution scheme. Any two nodes want to establish a pairwise key only when they have q' ($q' \geq q$) keys in common. This achieves greatly strengthened security under small-scale attack while trading off increased vulnerability in the face of a large-scale physical attack on network nodes. Multipath key reinforcement scheme is a method to strengthen the security to set up a link key via Multipath. Nodes A and B want to set up a link key. Node A sends j different random values to node B. These values are sent to B along different paths. The new link key can be computed as follows: $k'_{AB} = k_{AB} \bigoplus v_1 \bigoplus v_2 \cdots \bigoplus v_j$.They also propose a random pairwise key scheme. A unique random pairwise key is generated for a pair of nodes, and the ID of that node is also stored along with the key in the key ring. After deployment, a node can find its shared common pairwise keys with its neighbors by their node IDs.

Liu and Ning [3] introduce two pairwise key predistribution schemes: a random subset assignment scheme and a grid-based key predistribution scheme. In the random subset assignment scheme, a server generates a set of bivariate t-degree polynomials. A unique ID is assigned to each polynomial. Each node has a subset of these polynomials. Any two nodes that have polynomial shares on the same bivariate polynomial can set up a shared pairwise key directly. Otherwise, they will use path key establishment method to set up a pairwise key. A source node sends a request to its intermediate nodes to establish a pairwise key with the destination node. The intermediate nodes forward this request until a node finds a path to the destination node. In the grid-based key predistribution scheme, the server assigns each node an ID $\langle i, j \rangle$ and corresponding row and column polynomial shares. Two nodes establish a pairwise key by a polynomial share discovery phase. If there is no match between these two nodes, they will begin a path discovery phase with the help of intermediate nodes.

In [4], Du et al. give an asymmetric predistribution (AP) key management scheme. We call this scheme AP for short. There are two types of nodes in AP: H-sensors and L-sensors. H-sensor nodes are much more powerful than L-sensor nodes. H-sensor nodes act as the cluster heads. AP consists of three phases: key predistribution phase, shared-key discovery phase, and H-sensor-based pairwise key setup phase. In key predistribution phase, each H-sensor is assigned with a special key K_H and M keys shared with L-sensors. K_H is shared among H-sensors and the base station. Each L-sensor is preloaded with L ($L \ll M$) keys. In shared-key discovery phase, there are two ways for two nodes to discover their shared keys. In the distributed way, each L-sensor exchanges its key IDs with its neighbors. In the centralized way, two nodes find out their shared keys with the help of H-sensors. If two nodes have no shared keys, they will request the H-sensor to set up a pairwise key for them. In most cases, both nodes share a key with the H-sensor node at least. If these shared keys cannot be found, the H-sensor will set up a pairwise key

with the help of its neighbors from the 1st degree to the dth degree. The worst case is that none of the H-sensor neighbors have shared keys with these two nodes. The H-sensor will set up a pairwise key with the help of the base station.

In [5], Lu et al. divide sensor nodes into I classes. The higher the class a node belongs to, the more powerful it is. So the Class I node is the most powerful node. The authors adopt two key distribution schemes. One is based on a key pool, and the other is based on a polynomial pool. In this scheme, a Class 1 node (C_1) can communicate with several Class 2 (C_2) nodes securely, as a C_1 node and its neighboring C_2 nodes share at least one key. This increases the key connectivity, reliability, and resilience of the network.

Yu and Guan propose a key management scheme by using deployment knowledge in [6]. The authors mainly discuss pairwise keys through one-hop links. The target field consists of many hexagon grids. Each grid is a group with the same number of sensor nodes. These groups are classified into two categories: some groups are the basic groups, others are normal groups. Each group has a public matrix G and a unique secret matrix A. Each basic group is assigned with a different matrix B. After this, each group is assigned with the matrices that have been assigned to its neighbor basic groups. After all groups have their B matrices assigned, each node selects one column from matrix G, one row from matrix A, and some rows from B matrices. The neighboring nodes can establish their pairwise keys by the rows and columns selected from the matrices. However, if two nodes cannot find a matched index, they will no longer be able to communicate with each other. This scheme increases the connectivity and strengthens the resilience against nodes capture.

Liu et al. [7] propose an asymmetric key predistribution scheme (AKPS). AKPS uses a trusted authority (TA) to distribute secret keys to each user and public keying material to keying material servers (KMSs). With the help of KMSs, two sensor nodes can establish a session key to encrypt messages. AKPS has an advantage over other schemes in that the compromise of KMSs does not disclose any information of the users' secret keys and the session keys.

Nguyen et al. propose a key management scheme considering the signal range in [8]. Each node is assigned with a subset of keys from the key pool by a key setup server. Two nodes in each other's transmission range will be assigned with a subset of common keys. This scheme also includes shared-key discovery and path key establishment phases. By using the location information of sensor nodes, it improves the connectivity and achieves better resilience than other schemes. However, this scheme depends too much on the deployment knowledge.

Zhu et al. [9] propose a localized encryption and authentication protocol (LEAP). Each node has four types of keys—an individual key shared with the base station, a pairwise key shared with another node, a cluster key shared with its neighboring nodes, and a group key shared with all the nodes in the network. The individual key $K_u^m = f_{K^m}(u)$ is preloaded into each node before deployment. K^m is a master key which is only known to the controller. In order to save the storage, the controller computes the individual key K_u^m unless it needs to communicate with node u. The controller loads each node with an initial key K_I. Each node derives a master key $K_u = f_{K_I}(u)$. With the help of the initial key, each node can verify its neighbor identities. The pairwise key is established as follows: $K_{uv} = f_{K_v}(u)$. There is no need to exchange messages between u and v in pairwise key establishment. Node u generates a cluster key K_u^c. The cluster key is encrypted by all the pairwise keys that node u shared with its neighbors, and then the encrypted keys are sent to its neighbors, respectively. When a node timer expires, it erases the initial key K_I and all the master keys of its neighbors. But its own master key K_u is maintained.

The SHELL key management scheme is proposed by Younis et al. in [10]. SHELL means scalable, hierarchical, efficient, location-aware, and light-weight. SHELL is based on EBS (exclusion basis system). Each node selects k keys from a key pool of size $k + m$. Key refresh operations are very easy in EBS. But the main drawback of EBS is that it is highly vulnerable to collusion attacks. SHELL reduces the collusion attacks by using the nodes location information. A command node, which can communicate with each gateway node directly, is used to govern the entire network. The location information of nodes is used when assigning keys. SHELL is more resilient to collusion attacks than EBS.

In [11], Jing et al. propose an identity-based public key management scheme—C4 W. C4 W is based on the elliptic curve cryptography (ECC) and combined public key [14] system. Each node can compute other node public keys from their identities. If two nodes A and B want to establish a pairwise key, the key agreement steps are as follows: (1) A and B compute each other's public key, $P_A = \mathrm{PKF}(\mathrm{ID}_A)$, $P_B = \mathrm{PKF}_{\mathrm{MAP}}(\mathrm{ID}_B)$. PKF ($\mathrm{ID}_X$) is a public key generation function. (2) A chooses a random value r and computes the shared secret $K_{AB} = \mathrm{KDF}(r \cdot S_A \cdot P_B)$, where S_A is the predistributed secret key of node A and P_B is the public key of node B. $S_A = s \cdot P_A$, s is the master key of the key generation system. (3) A computes $\mathrm{MAC}_{AB} = \mathrm{MAC}(K_{AB} \| \mathrm{ID}_A \| r)$ and sends $\mathrm{ID}_A \| r \| \mathrm{MAC}_{AB}$ to B. (4) B computes the shared secret $K'_{AB} = \mathrm{KDF}(r \cdot S_B \cdot P_A)$. We can get that $\mathrm{K}_{AB} = \mathrm{KDF}(r \cdot S_A \cdot P_B) = \mathrm{KDF}(r \cdot s \cdot P_A \cdot P_B) = \mathrm{KDF}(r \cdot s \cdot P_B \cdot P_A) = \mathrm{KDF}(r \cdot S_B \cdot P_A) = K'_{AB}$. In C4 W, no matter how many nodes are there in the networks, each node can compute another node public key with just several point additions, which lowers the energy consumption. Further, C4 W has an excellent scalability.

Szczechowiak and Collier [12] propose a key agreement scheme based on identity-based cryptograph (IBC) for wireless sensor networks. We call this scheme IBC-KM for short. A trust authority is used to predistribute a secret key, a unique identity IDx, a hashing function H, a mapping function, and a key derivation function (KDF) into each node memory. Suppose s is the master key of the trusted authority. Node A has a private key sA and node B a key sB. They can compute each other's public key by calculating $A = H(\mathrm{ID}_A)$ and $B = H(\mathrm{ID}_B)$. The pairwise key between A and B can be calculated as follow: $K_{AB} = \mathrm{KDF}(\hat{e}(sA, B)) = \mathrm{KDF}(\hat{e}(A, B)^s) = \mathrm{KDF}(\hat{e}(A, sB)) = K_{BA}$. This scheme saves much key storage space and has a high resilience against node capture. However, the key agreement protocol is pairing-based cryptography, which costs much computational and

Table 1: List of used notations.

Notation	Description
CH_t	Cluster head t
N_i	Sensor node i
K_{pt}	Public key for cluster head t
K_{st}	Secret key for cluster head t
G	A base point on elliptic curve
K_{CHAuth}	Authentication key for cluster head
K_{Authi}	Authentication key for member sensor node i
$H(\,)$	Hash function
ID_i	Identity of sensor node i
$\|\|$	Concatenation operator
$E_k(\text{Msg})$	Encrypt a message with key k
$D_k(\text{Msg})$	Decrypt a message with key k
r_m	The transmission range of a member sensor node
R_{CH}	The transmission range of a cluster head
N	Number of sensor nodes in the network
P	Key pool size
m	Key ring size

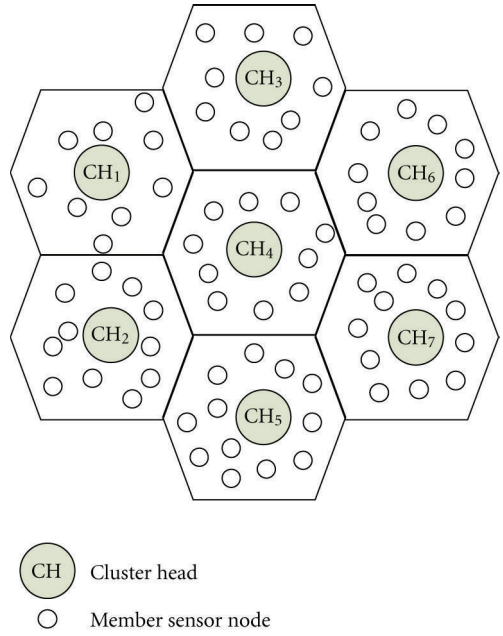

Figure 1: The structure of a heterogeneous wireless sensor network.

energy resource. Each sensor node needs much energy and time to compute the shared pairwise keys with its neighboring nodes.

Traynor et al. propose a suit of key management protocols known as LIGER in [13]. LIGER is an integration of LION and TIGER. In LION, there is no KDC in the network. Two L1 nodes establish a session key with the help of an L2 node. While in TIGER, the authentication and key establishment are achieved with the help of the KDC. There are two cases of authentication and key establishment in TIGER, one is for two L2 nodes, and the other is for an L1 node and an L2 node. Both of them use the KDC to generate a session key, and the session key is encrypted for both nodes.

Du et al. [15] propose a key management scheme using deployment knowledge. The proposed scheme is based on Eschenauer and Gligor's scheme. They divide the key pool S into $t \times n$ key pools $S_{i,j}$ $(i = 1, 2, \ldots, t, \quad j = 1, \ldots, n)$, with $S_{i,j}$ corresponding to the deployment group $G_{i,j}$. Each sensor node in group $G_{i,j}$ randomly picks m keys from its corresponding key pool $S_{i,j}$. By using the deployment knowledge, each node stores fewer keys than other key predistribution schemes, while achieving the same level of connectivity. The most important improvement of Du's scheme is that it substantially improves the network resilience against node capture. Since each node carries fewer keys, an attacker gets less information of the network when it captures a sensor node.

3. The Proposed Scheme: MAKM

From the above introduction, we can find that the following problems are not well considered in the aforementioned solutions.

(i) *Authentication*. Many proposed schemes have no authentication mechanism to verify the identity of other nodes in the network. In other schemes, the authentication is based on the common keys between two nodes. However, these schemes are based on the probabilistic key management mechanism. If two nodes have no shared keys, they could not verify each other's identity directly. Malicious nodes may pretend to be legitimate nodes and subvert the network.

(ii) *Using Deployment Knowledge*. In [3, 6, 8, 10, 15], authors use the location information of sensor nodes. The target fields are divided into many equal units with the same size. These schemes achieve higher connectivity and reduce storage space with the help of deployment knowledge. However, it is hard to deploy each sensor node to a predetermined location in actual environments, especially when the sensor nodes should be deployed in military fields or dropped from a helicopter.

In order to solve the problems mentioned above, we propose a modular arithmetic-based key management scheme—MAKM. There is an offline base station which is used to preload keys for each sensor node in the network. The whole network consists of many clusters. In each cluster, there is a cluster head (CH) which is used to distribute key seeds to its member sensor nodes. A cluster head has a more powerful processing capability and is more resistant to attacks than other normal member sensor nodes. Cluster heads can communicate with their neighboring cluster heads directly. The notations used in this paper are listed in Table 1. Figure 1 depicts the network structure.

3.1. Elliptic Curve Cryptography. Public key cryptography algorithms are widely used in networks. RSA and ECC [16] are the most well-known algorithms. Compared with RSA, ECC achieves the same security with smaller key sizes,

ID_1	Msg_{update_1}	...	...	ID_{k-1}	$Msg_{update_{k-1}}$	ID_{k+1}	$Msg_{update_{k+1}}$	...	...
...	...	ID_j	Msg_{update_j}	...	...	ID_M	Msg_{update_M}		

Figure 2: Key update message format.

Table 2: Key seed table in the cluster head.

Node ID	Key seed
ID_1	$k_1 p + r_t$
ID_2	$k_2 p + r_t$
...	...
ID_j	$k_3 p + r_t$
...	...
ID_M	$k_M p + r_t$

lower memory usage, computational cost and energy consumption. ECC with 160-bit keys can provide comparable security to RSA with 1024-bit keys [16]. Thus, ECC is more suited for WSNs. In our proposed scheme, we use elliptic curve digital signature algorithm (ECDSA) [16] to verify the identity of cluster heads. ECDSA takes about 10 KB of ROM and 152 Bytes of RAM on MICAz platform, and 8 KB of ROM and 160 Bytes of RAM on TelosB platform [17], which is viable for a 128 KB ROM and 4 KB RAM sensor node [12].

3.2. Key Predistribution Phase. In this subsection, we present the key predistribution for each sensor node. In our scheme, an authentication key K_{CHAuth}, a pair of public/private key $K_{\text{pt}}/K_{\text{st}}$, and a certificate Cert_{CHt} signed by the base station are predistributed in each cluster head. $K_{\text{pt}} = K_{\text{st}} \cdot G$, where is the operation of point multiplication on elliptic curves and G is a base point on the elliptic curve. The authentication key K_{CHAuth} is used to verify member sensor node identities. K_{CHAuth} is known to all cluster heads and the base station. The public/private key pair $K_{\text{pt}}/K_{\text{st}}$ is used to establish pairwise keys among cluster heads. It is easy to calculate K_{pt} from K_{st} and G, but it is hard to compute K_{st} from K_{pt} and G due to the difficulty of elliptic curve discrete logarithm problem (ECDLP) [18]. An authentication key K_{Authi} and the public key $K_{P_{\text{BS}}}$ of the base station are predistributed in each member sensor node. $K_{P_{\text{BS}}}$ is used to verify the certificates of the cluster heads. K_{Authi} can be calculated by the following hash function:

$$K_{\text{Authi}} = H\ (\text{ID}_i || K_{\text{CHAuth}}). \tag{1}$$

$H()$ is a hash function such as SHA1 and MD5. ID_i is the identity of sensor node i.

3.3. Bootstrapping Phase. During the bootstrapping phase, the cluster heads and member sensor nodes carry out a mutual authentication in order to protect them from man-in-the-middle attack. After all sensor nodes are deployed to the target field, the cluster heads begin to broadcast HELLO messages. The HELLO message contains the identity and the certificate of the cluster head. Each member sensor node will receive at least one message from its neighboring cluster heads. Some sensor nodes may receive several messages. In this case, they will choose the cluster heads according to the signal strength. Once a sensor node has chosen a cluster head, it verifies the certificate of the cluster head by the public key of the base station. If the cluster head is valid, the sensor node sends a *JoinReq* message to its cluster head CH_t. The *JoinReq* message consists of an authentication message and the sensor node ID. With the predistributed authentication key K_{Authi}, the sensor node can compute an authentication message auth_msg_i as follows:

$$\text{auth\_msg}_i = H\ (\text{ID}_i || K_{\text{Authi}}). \tag{2}$$

When the cluster head CH_t receives the authentication message from N_i, it verifies the member sensor node identity:

$$\text{auth\_msg}_i' = H\ (\text{ID}_i || H(\text{ID}_i || K_{\text{CHAuth}})). \tag{3}$$

CH_t compares $\text{auth\_msg}_i'$ with auth_msg_i. If they are equal, CH_t sends a key seed Seed_i to node i. Assume there are M sensor nodes in cluster t. The cluster head CH_t chooses a large prime number r_t and a modulus p. The key seed Seed_j is calculated by the following equation:

$$\text{Seed}_j = k_j p + r_t \quad j = 1, 2, \ldots, M, \tag{4}$$

the value k_j is only known to the cluster head CH_t. With the authentication key K_{CHAuth} and the sensor node ID, the cluster head can compute each sensor node authentication key K_{Authi} by (1) on the fly. The key seed Seed_j is encrypted by N_i authentication key. After sensor node N_i receiving the encrypted key seed message, it decrypts this message to get the key seed, and then it deletes its authentication key and the base station public key immediately. Otherwise, the cluster head drops the *JoinReq* message. Each member sensor node in this cluster gets its key seed (Seed_j); respectively. In order to prevent the key seeds from a brute force attack, the value k_j should be a large integer, and the value p should be a large prime number. The key seed table, as shown in Table 2, is stored in the cluster head (CH_t). The authentication and key seed distribution procedure is depicted as follows:

(1) $\text{CH}_t \rightarrow *:$ HELLO, $\{\text{ID}_{\text{CH}_t}, \text{Cert}_{\text{CH}_t}\}$;

(2) $\text{N}_i \rightarrow \text{CH}_t:\ \{\text{ID}_i, \text{auth\_msg}_i\}$;

(3) $\text{CH}_t:\ \text{auth\_msg}_i' \stackrel{?}{=} \text{auth\_msg}_i$;

(4) $\text{CH}_t \rightarrow \text{N}_i:\ \text{E}_{K_{\text{Authi}}}(\text{Seed}_j)$;

(5) $\text{N}_i:\ \text{D}_{K_{\text{Authi}}}(\text{Seed}_j)$.

3.4. Key Generation. Two integers a and b are congruent if they give the same remainder when divided by a positive integer p. This is written as follows:

$$a \equiv b (\text{mod}\ p). \tag{5}$$

Table 3: Simulation environment settings.

Parameter	Value
l	280 m
N	200–1000
r_m	$\sqrt{1000}$ m
R_{CH}	150 m
P	10000
M	50–100

In our MAKM scheme, each sensor node in the network has a large integer as its key seed. According to (4) and (5), each sensor node in the same cluster can get the same remainder r_t when divided by the same modulus p.

The key $K_{N_jC_t}$ is the key between the member node (N_j) and its cluster head (CH_t), and K_{G_t} is the group key. The group key is shared by all members in the same cluster. $K_{N_jC_t}$ and K_{G_t} are computed by the following equations:

$$K_{N_jC_t} = H\left(DToStr\left(\mathrm{Seed}_j\right)\right) \quad j = 1, 2, \ldots, M, \tag{6}$$

$$K_{Gt} = H\left(DToStr\left(\mathrm{Seed}_j \mod p\right)\right) \quad j = 1, 2 \cdots M. \tag{7}$$

$DToStr()$ is a function which is used to convert an integer to a string. The output of a hash function is a fixed length string. This is the shared encryption and decryption key. Due to the congruence property of modular arithmetic, all members in the same cluster can compute the shared group key K_{G_t}. In order to give a simple example, we use some small integers. For example, let $p = 7$, $k_1 = 6$, $k_2 = 8$, $r_t = 5$; so $\mathrm{Seed}_1 = 47$ and $\mathrm{Seed}_2 = 61$. According to (7), we can get

$$\begin{aligned} K_{G_t} &= H\ (DToStr\ (47 \mod 7)) = H \quad (DToStr\ (5)) \\ &= H\ (DToStr\ (61 \mod 7)) = H \quad (DToStr\ (5)). \end{aligned} \tag{8}$$

Thus, these two nodes can get the same group key with different key seeds.

3.5. Addition of a New Node. In wireless sensor networks, some sensor nodes may run out of power supply over time because of the inherent energy constraint characteristic of wireless sensor nodes. This may significantly degrade the performance of the whole network. Some new sensor nodes need to be added to the network to mitigate this problem. In order to support the inclusion of the new sensor node, the key seeds of the existing nodes need to be updated to keep the backward secrecy. We assume that the new node will be deployed in cluster t. The cluster head CH_t chooses an increment Δr_t and encrypts it with its group key K_{G_t}.CH_t broadcasts this update message to all member sensor nodes in its cluster.

$$\mathrm{Msg}_{\mathrm{update}} = E_{K_{G_t}}\{\Delta r_t\}. \tag{9}$$

In addition, the key seed table in the cluster head should be refreshed by the following equation:

$$\mathrm{Seed}_j' = k_j p + r_t + \Delta r_t. \tag{10}$$

When a member sensor node in this cluster receives the update message, it decrypts the cipher text and gets the increment Δr_t. Then it refreshes its key seed by the following equation:

$$\mathrm{Seed}_j' = \mathrm{Seed}_j + \Delta r_t. \tag{11}$$

After refreshing the key seeds for all sensor nodes, the cluster head verifies the identity of the new node. If the new node is a legitimate node, it will be given a key seed by the cluster head. This procedure is identical to the bootstrapping phase as described in Subsection 3.3.

3.6. Nodes Compromise. We assume that the cluster head can detect the compromise of a sensor node in its cluster. When a sensor node (Node k) is compromised, the shared key seeds in this cluster should be updated in order to keep forward secrecy. The cluster head CH_t uses the shared key $K_{N_jC_t}$ with each member sensor node to encrypt an update message:

$$\mathrm{Msg}_{\mathrm{update}_j} = E_{K_{N_jC_t}}\{\Delta r_t\}. \tag{12}$$

$\mathrm{Msg}_{\mathrm{update}_j}$ represents the encrypted message for node j. The cipher texts of all update messages are concatenated along with their ID*s* to a string *Str_keyupdate*, which is shown in Figure 2. The cluster head broadcasts this message *Str_keyupdate* to all member nodes in its cluster. When N_j receives *Str_keyupdate*, it finds out $\mathrm{Msg}_{\mathrm{update}_j}$ from *Str_keyupdate* by ID_j and then decrypts $\mathrm{Msg}_{\mathrm{update}_j}$ to get the increment Δr_t. The cluster head and member nodes refresh the key seeds as described in Subsection 3.5 (Addition of a New Node).

Each node in the cluster can use its shared key with the cluster head to decrypt the key update message and get the increment Δr_t. However, the cluster head does not use $K_{N_kC_t}$ to encrypt this message for node k. Thus, node k cannot get the increment Δr_t.

3.7. Key Establishment between Two Nodes. Sensor nodes are used to collect and process useful data. In most cases, this data is transmitted to the cluster head via multihops. So it only needs to be encrypted by the shared key between the sensor node and its cluster head. In our proposed scheme, each node has a unique key shared with its cluster head. However, in some cases, sensor nodes need to communicate with other nodes. For example, in an aircraft, sensor nodes need to send messages to other nodes about the status of the end systems they monitor. It is necessary to set up the session key for these two nodes. They may belong to a same cluster, or two different clusters. For this purpose, we classify the key establishment between two nodes into two types, intracluster and intercluster key establishment. We will discuss how to establish a session key between two nodes in the following paragraphs.

3.7.1. Intracluster Key Establishment. If two sensor nodes are in the same cluster, we call this procedure intracluster key establishment. They establish a session key with the help

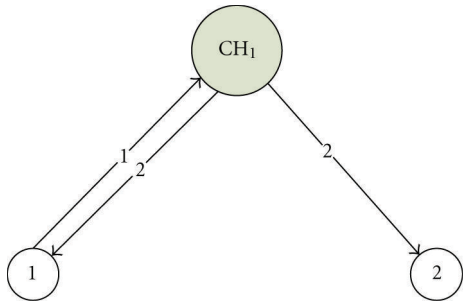

Figure 3: Intracluster key establishment.

of their cluster head, which is shown in Figure 3. The key establishment message flow for Figure 3 is as follows:

(1) $N_1 \rightarrow \mathrm{CH}_1 : \mathrm{Req}_{\mathrm{Keysetup}}, E_{K_{N_1C_1}}\{\mathrm{ID}_1, \mathrm{ID}_2\}$,

(2) $\mathrm{CH}_1 \rightarrow N_1 : E_{K_{N_1C_1}}\{K_{N_1N_2}\}, \mathrm{CH}_1 \rightarrow N_2 : E_{K_{N_2C_1}}\{K_{N_1N_2}\}$.

To establish a session key between these two nodes, the sensor node N_1 sends a key setup request to its cluster head CH_1. This message includes the IDs of these two sensor nodes. After the cluster head CH_1 receives this request, it randomly generates a session key $K_{N_1N_2}$ by a pseudo random function (PRF) and encrypts it by the shared keys with these two nodes. The encrypted session keys are sent to these two nodes, respectively. Each node receives this encrypted session key, decrypts it, and begins to communicate with each other securely.

3.7.2. Intercluster Key Establishment. On the other hand, if these two nodes belong to different clusters, we call this procedure intercluster key establishment, which is much more complex than intracluster key establishment. Figure 4 depicts this key establishment procedure. In our proposed scheme, sensor nodes are more likely to choose cluster heads to relay messages because they have a larger transmission range. For example, in Figure 4, N_1 from cluster 1 will choose the path "$N_1 - > \mathrm{CH}_1 - > \mathrm{CH}_4 - > \mathrm{CH}_7 - > N_2$" to send a message to N_2, instead of using the member sensor nodes along the path of the dash line. The key establishment message flow for Figure 4 is as follows:

(1) $N_1 \rightarrow \mathrm{CH}_1 : \mathrm{Req}_{\mathrm{Keysetup}}, E_{K_{N_1C_1}}\{\mathrm{ID}_1, \mathrm{ID}_2\}$,

(2) $\mathrm{CH}_1 \rightarrow \mathrm{CH}_4 \longrightarrow \mathrm{CH}_7 : E_{K_{C_1C_7}}\{K_{N_1N_2}\}, \mathrm{CH}_1 \rightarrow N_1 : E_{K_{N_1C_1}}\{K_{N_1N_2}\}$,

(3) $\mathrm{CH}_7 \rightarrow N_2 : E_{K_{C_7N_2}}\{K_{N_1N_2}\}$.

At first, N_1 sends a key setup request to its cluster head CH_1. When CH_1 receives this request, it randomly generates a session key $K_{N_1N_2}$. The session key is encrypted by $K_{C_1C_7}$ and $K_{N_1C_1}$, and then CH_1 sends the encrypted keys to CH_7 and N_1, respectively. N_1 uses its shared key with CH_1 to decrypt this key message and gets the key $K_{N_1N_2}$. When CH_7 receives this key, it uses $K_{C_1C_7}$ to decrypt it. CH_7 then encrypts $K_{N_1N_2}$ by the shared key with N_2 and sends it to N_2. Finally, N_2 decrypts it and gets the shared key $K_{N_1N_2}$ with N_1. In this key establishment procedure, the shared key between CH_1 and CH_7 is established on demand by using ECDH [16]. In order to protect ECDH from malicious node attacks, we add a mutual authentication in this procedure. The shared key between CH_1 and CH_7 is set up as follows:

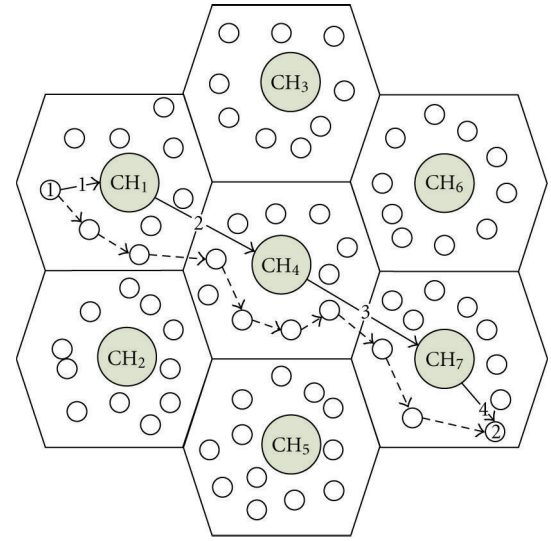

Figure 4: Intercluster key establishment.

(1) $\mathrm{CH}_1 \rightarrow \mathrm{CH}_7 : \mathrm{Req}_{\mathrm{Keysetup}}, \{\mathrm{ID}_{\mathrm{CH}_1}, K_{P_1}, \mathrm{Auth}_1 = H(K_{\mathrm{AuthCH}} \| \mathrm{ID}_{\mathrm{CH}_1} \| K_{P_1})\}$,

(2) $\mathrm{CH}_7 : \mathrm{Auth}_1' = H(K_{\mathrm{AuthCH}} \| \mathrm{ID}_{\mathrm{CH}_1} \| K_{P_1}), \mathrm{Auth}_1' \stackrel{?}{=} \mathrm{Auth}_1$. If equal, $\mathrm{CH}_7 \rightarrow \mathrm{CH}_1 : \{\mathrm{ID}_{\mathrm{CH}_7}, K_{P_7}, \mathrm{Auth}_7 = H(K_{\mathrm{AuthCH}} \| \mathrm{ID}_{\mathrm{CH}_7} \| K_{P_7})\}$; else, end,

(3) $\mathrm{CH}_1 : \mathrm{Auth}_7' = H(K_{\mathrm{AuthCH}} \| \mathrm{ID}_{\mathrm{CH}_7} \| K_{P_7}), \mathrm{Auth}_7' \stackrel{?}{=} \mathrm{Auth}_7$. If equal, $K_{C_1C_7} = K_{S_1} \cdot K_{P_7} = K_{S_1} \cdot K_{S_7} \cdot G$; else, end,

(4) $\mathrm{CH}_7 : K_{C_7C_1} = K_{S_7} \cdot K_{P_1} = K_{S_7} \cdot K_{S_1} \cdot G = K_{C_1C_7}$.

4. Performance Evaluation

In this section, we will discuss our proposed scheme in detail. For analysis, we compare our scheme with E-G [1], C4 W [11], and IBC-KM [12]. We focus on the evaluation of the following criteria.

(1) Key Storage Space. Key storage space is the total memory usage of keys (or key seeds) stored in the network.

(2) Resilience against Sensor Node Capture. We assume that an adversary can attack a sensor node after the network is deployed. Once a sensor node is captured, the adversary can read all secret information in it. We evaluate the resilience of the network by estimating the probability of a secure link being compromised when there are some sensor nodes captured in the network.

(3) Time Consumption of Key Establishment. In this paper, time consumption of key establishment is the time for

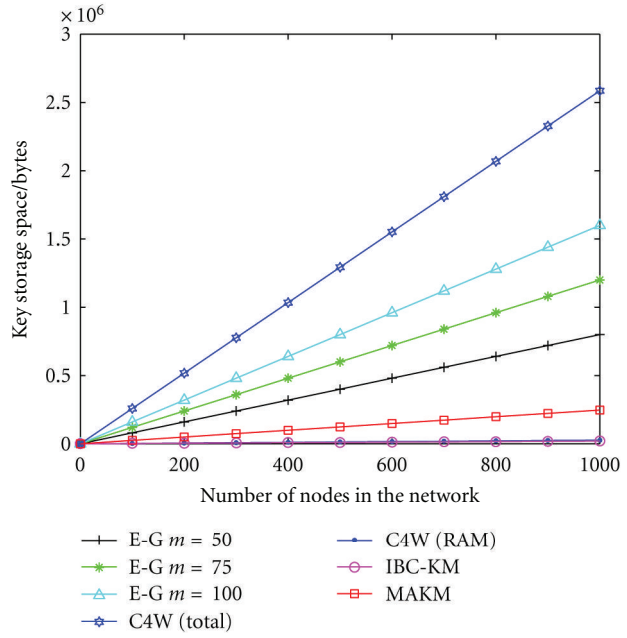

Figure 5: Key storage space versus the number of nodes in the network.

sensor nodes to establish shared keys during the network initialization.

4.1. Key Storage Space. Key storage space measures the total memory usage of keys (or key seeds) distributed in the network. In the proposed MAKM scheme, each member node only needs to store a key seed. A cluster head has an authentication key (16 bytes), a pair of public/private key (40/20 bytes) [16], a certificate (86 bytes) [19], an increment (4 bytes), and a key seed table. Other shared session keys are established on demand. Assume there are N nodes and H clusters in our test scene. The total key storage space is

$$\begin{aligned} L_{\text{total}} &= L_{MN} + L_{\text{CH}} \\ &= (N-H)\times 128 + H\times(16+40+20+86+4) \\ &\quad + (N-H)\times 128 \\ &= 256N - 90H, \end{aligned} \tag{13}$$

where L_{MN} is the total storage space of key seeds stored in the member nodes and L_{CH} is the total storage space of key seeds stored in the cluster heads. In our proposed scheme, the total ratio of cluster heads is 10%.

In E-G scheme [1], each node picks m keys from the key pool. Thus, the total number of keys in the network is mN. In C4 W [11], each node is predistributed with a secret key (20 bytes), an l-bit length mask array A_{mask} (6 bytes), and a point array A_{pk} (40*64 bytes) consisting of some randomly selected points on the elliptic curve. In IBC-KM [12], each node is only predistributed with a secret key (20 bytes).

In Figure 5, we plot the total storage space requirements of these four schemes. From this figure we can get that IBC-KM needs the least key storage space because it only stores a single secret key in its cache. C4 W needs to store a forty-point array, which consumes much storage space. However, the authors in [11] imply that this point array can be written into a sensor node ROM, which will not use the node RAM space. From Figure 5, we can obtain that C4 W needs only a little storage space without the consideration of point array stored in ROM. The storage space of E-G scheme varies according to the key ring size. E-G needs less space than C4 W, but it needs more space than IBC-KM and MAKM.

4.2. Resilience against Sensor Node Capture. Wireless sensor nodes are more likely to be captured than other wireless nodes due to the constraint capability of sensor nodes. Thus, the resilience of a wireless sensor network is very important. We will discuss the resilience of the proposed MAKM scheme in the following paragraphs.

When a sensor node is compromised, we assume that the adversary can get all materials of the node. In a key-pool-based key management scheme [1], the capture of a sensor node will disclose some keys of the network. Other sensor nodes that have a subset of keys shared with this node will be affected. When the keys stored in the captured node are revoked, it will decrease the connectivity of the network. In Eschenauer and Gligor's scheme, suppose the size of the key pool is P. Each node picks m keys from the key pool. The probability that any two nodes have exactly t keys in common is

$$p(t) = \frac{\binom{P}{t}\binom{P-t}{m-t}\binom{P-m}{m-t}}{\binom{P}{m}^2}. \tag{14}$$

In [2], Chan et al. gives the probability of compromising a secure link between two uncompromised nodes under Eschenauer and Gligor's scheme (E-G Scheme) as follows:

$$p_x(m) = \frac{\sum_{t=1}^{m}\left(1-(1-(m/P))^x\right)^t p(t)}{p_s}, \tag{15}$$

where x is the number of compromised nodes and $p_s = p(1) + p(2) + \cdots + p(m)$ is the probability of setting up a secure link between two nodes.

In our proposed MAKM scheme, the shared session keys are established on demand. Each pair of nodes shares a unique symmetric key, and this session key is blind to other member sensor nodes. Thus, the capture of member nodes has no impact on the secure links between other uncompromised nodes.

Figure 6 shows that the probability of compromised links between uncompromised nodes varies with the number of compromised nodes. The x-axis is the number of nodes captured by the attacker, while y-axis is the probability of a link between two uncompromised nodes being compromised. In our test, the key pool size P is 10000, the total number of captured nodes varies from 10 to 300, and the keys preloaded in each node varies from 15 to 55.

Figure 6 shows the relationship between the probability of a secure link being compromised and the number of captured nodes. From this figure, we obtain that an increase of the keys preloaded in a sensor node under E-G scheme

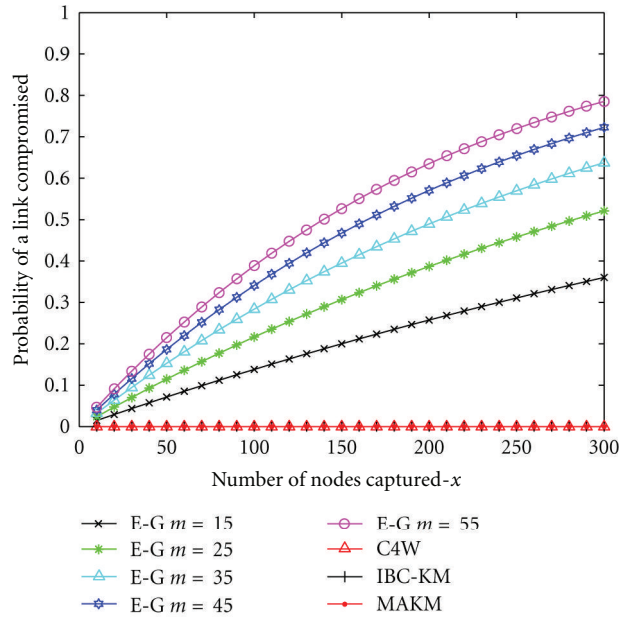

Figure 6: Probability of a communication link between two uncompromised nodes being compromised.

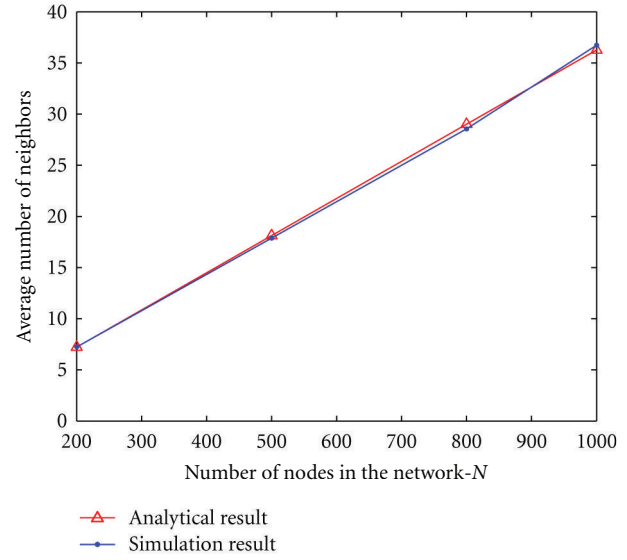

Figure 7: Average number of neighbors of a node in the network.

would make a secure link more likely to be compromised. This is because a senor node will disclose more keys when it is captured. We can also find the more nodes being captured under E-G scheme, the more likely a link being compromised. However, in C4 W, IBC-KM, and our proposed MAKM schemes, each sensor node has a unique key shared with another node. The capture of a sensor node will not affect the secure links between other uncompromised nodes. From the above analysis, we can find that the proposed MAKM scheme is very resilient against node capture.

4.3. Time Consumption of Key Establishment. In the following part, we will discuss the time consumption of key establishment.

In E-G scheme, key establishment consists of two phases: the shared-key discovery phase and path key establishment phase. The shared-key discovery phase takes place after each node has finished finding shared keys with its neighbors. Each node broadcasts a list of identifiers of the keys on its key ring to all its neighbors. Two neighboring nodes will set up a secure link if they share at least one key. However, if two neighboring nodes have no keys in common, they will begin a path key establishment phase with the help of their neighbors. A path key establishment may span one or more hops. In [1], the authors point out that if a neighboring node cannot be reached via a shared key (i.e., one link or one hop), it will take at most two or three hops to contact it. We can find that the time consumption of key establishment under E-G scheme is affected by the following three factors: the number of a node neighbors N_{neighbor}, the key ring size m, and the key pool size P. Time consumption of key establishment T_{KE} can be expressed as follows:

$$T_{\text{KE}} = \max\,(T_{\text{KEi}}), \quad i = 1, 2, \ldots, N,$$
$$T_{\text{KEi}} \propto N_{\text{neighbors}}, \qquad T_{\text{KEi}} \propto \frac{P}{m}, \tag{16}$$

where T_{KEi} is the time for node i to finish its key establishment.

We assume that there are N sensor nodes uniformly distributed in an $l \times l\,\text{m}^2$ target square area. A member node transmission range is r_m. Let (X_i, Y_i) be the coordinates of node i, where $0 \le X_i, Y_i \le l$. The probability density function (p.d.f.) f for a node u can be expressed by the function given by

$$f(x_i, y_i) = \frac{1}{l^2} \quad (0 \le x_i \le l, 0 \le y_i \le l). \tag{17}$$

Theorem 1. *The expected number of neighbors of a node in the network is* $(N-1)((1/2)r_m^4 - (8/3)lr_m^3 + \pi l^2 r_m^2)/l^4$.

Proof. Suppose node i (X_i, Y_i) and node j (X_j, Y_j) are two arbitrary nodes in the network. Let $U_i = |X_i - X_j|^2$ and $V_i = |Y_i - Y_j|^2$. In [20], Yen and Yu give the p.d.f. of U_i and V_i as follows:

$$f(u) = \frac{(l/\sqrt{u} - 1)}{l^2}, \quad 0 \le u \le l^2,$$
$$f(v) = \frac{(l/\sqrt{v} - 1)}{l^2}, \quad 0 \le v \le l^2. \tag{18}$$

Since U_i and V_i are independent, the joint p.d.f. of U_i and V_i is $f(u, v) = f(u)f(v)$. Yen and Yu point out that the probability Pr of a physical link between node i and node j is

$$\begin{aligned}\text{Pr} &= \text{Pr}[u + v \le r_m^2] \\ &= \int_0^{r_m^2} \int_0^{r_m^2 - u} f(u, v)\,dv\,du \\ &= \int_0^{r_m^2} \int_0^{r_m^2 - u} \frac{(l/\sqrt{u} - 1)(l/\sqrt{v} - 1)}{l^4}\,dv\,du \\ &= \frac{(1/2)r_m^4 - (8/3)lr_m^3 + \pi l^2 r_m^2}{l^4}.\end{aligned} \tag{19}$$

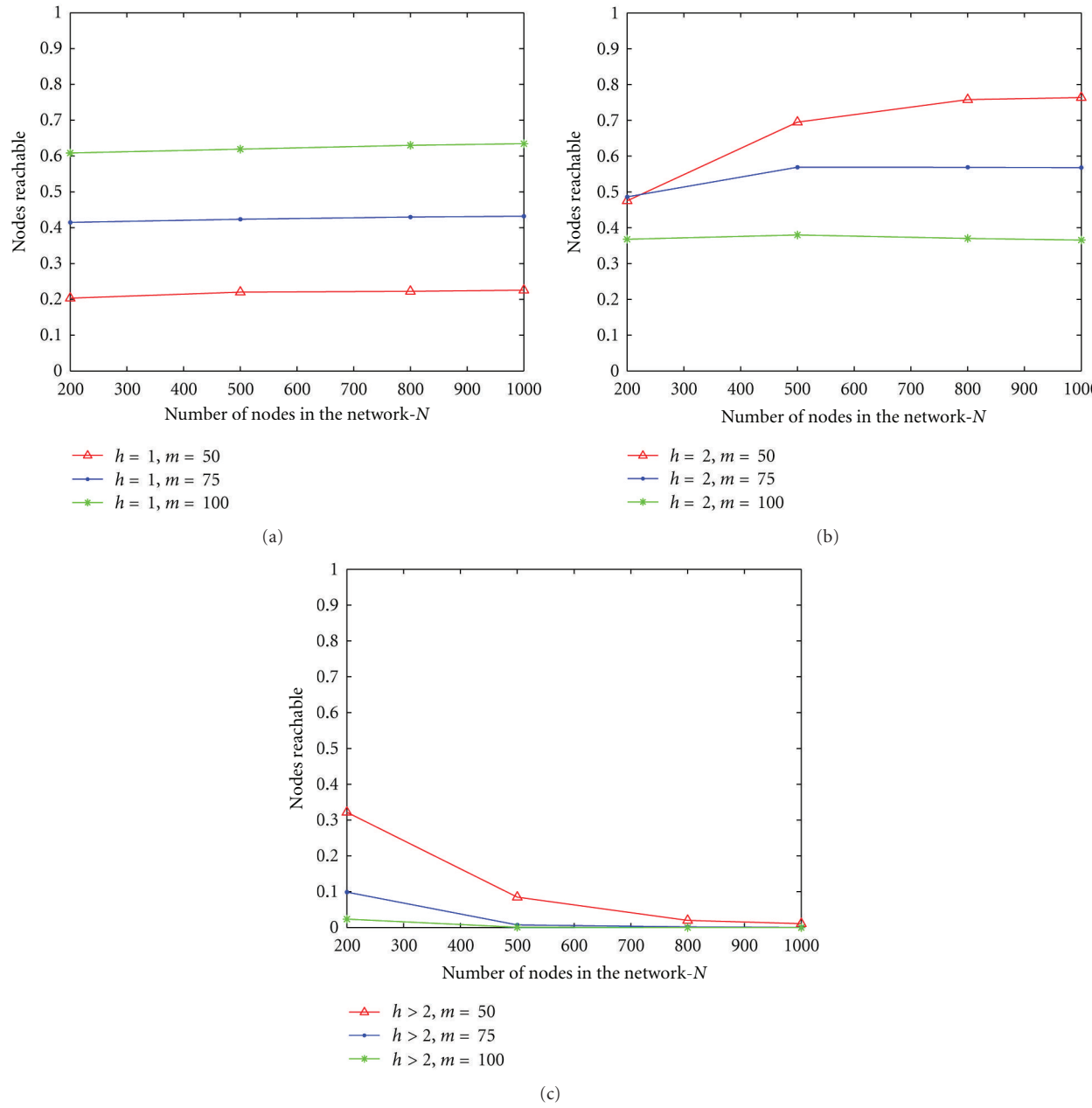

FIGURE 8: Path length to set up a pathkey under E-G scheme. (a) The percentage of two neighboring nodes that can find a shared key directly, (b) the percentage of two neighboring nodes that set up a path key via two hops, and (c) the percentage of two neighboring nodes that set up a path key via more than two hops.

We can get that the expected number of a sensor node neighbors N_{neighbor} is

$$(N-1)\Pr = \frac{(N-1)\left((1/2)r_m^4 - (8/3)lr_m^3 + \pi l^2 r_m^2\right)}{l^4} \approx \frac{\rho\left((1/2)r_m^4 - (8/3)lr_m^3 + \pi l^2 r_m^2\right)}{l^2}, \tag{20}$$

where $\rho = N/l^2$ is the node density of the network.

From (16) and (20), we can obtain that a higher node density will increase the time consumption of key establishment. This is because each node needs more time to find shared keys with more neighbors. Further, the smaller m and larger P will also increase the time consumption of key establishment. The smaller m is and the larger P is, the higher the probability that two neighboring nodes have no common keys is, and, therefore, longer paths are needed to be found to establish a path key.

In C4 W and IBC-KM, each node finds its neighbors at first. Then, the sensor node computes its neighboring node public keys according to their identities. Finally, two

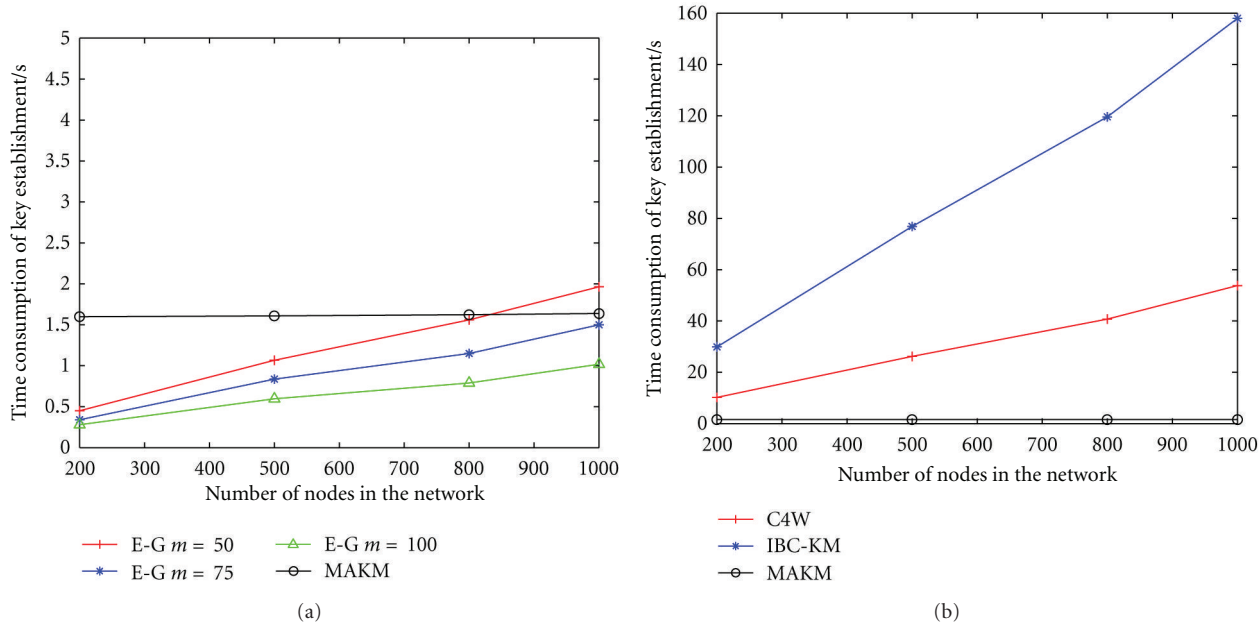

Figure 9: Time consumption of key establishment. (a) E-G and MAKM; (b) C4 W, IBC-KM and MAKM.

arbitrary neighboring nodes can compute a unique shared key. The shared key is determined by one node private key and its neighbor public key. Each sensor node establishes a set of shared keys with its neighbors. We can find that both C4 W and IBC-KM are identical to E-G and the time consumption of key establishment is affected by the number of a sensor node neighbors.

In our proposed MAKM scheme, there is no need to find shared keys. As shown in the authentication and key seed distribution procedure, we can find that the time consumption of key establishment is the time for each node to finish authentication and get a key seed from its cluster head. Thus, we can get the time consumption of key establishment T_{KE} as follows:

$$T_{KE} = \max(T_{KEi}), \quad i = 1, 2, \ldots, H, \qquad T_{KEi} \propto N_{members}, \tag{21}$$

where T_{KEi} is the time for cluster i to finish its key establishment and $N_{members}$ is the number of member sensor nodes in the cluster. From (21), we can find that key establishment time is only determined by the number of member sensor nodes in each cluster. Since the ratio of cluster heads to member nodes is a constant in our MAKM scheme, no matter how many nodes are deployed in the network, the time consumption of key establishment is the longest time of the cluster used to finish its key establishment. □

From the above analysis, we can find that the time consumption of key establishment under E-G, C4 W, and IBC-KM schemes increases with the node density of the network. In E-G, the ratio of key pool size of the network to key ring size of each node will also affect the time consumption of key establishment, while in our proposed scheme, the time consumption of key establishment is only determined by the ratio of cluster heads to member nodes. The simulation results will be shown in Section 5.

5. Simulations Study

In this section, we use NS2 to simulate our proposed MAKM scheme. We compare our MAKM with E-G, C4 W, and IBC-KM in terms of time and energy consumption of key establishment. The operation system is Linux-CentOS 5. The computer configuration is listed as follows: CPU: CoreII 1.86 G; memory: 2 G; hard disc: 250 G. The simulation environment settings used in the experiments are shown in Table 3.

5.1. Time Consumption of Key Establishment. From Section 4.3, we obtain that the time consumption of key establishment is affected by the node density and the key ring size under E-G scheme. C4 W and IBC-KM are also affected by node density. Figure 7 shows the average number of neighbors of a node in the network, where N varies from 200 to 1000. We can calculate the average number of neighbors of a node by (20). We plot the analytical and simulation results in Figure 7. We can find that the analytical result is consistent with the simulation result.

Figures 8(a), 8(b), and 8(c) show the average number of hops to set up a path key between two nodes under E-G scheme. Figure 8(a) shows the ratio of two neighboring nodes that can find a shared key directly, where $h = 1$ means there is only one hop between these two nodes. Those two nodes cannot find a common key directly and need to set up a path key via two or more hops, which are shown in Figures 8(b) and 8(c). If we increase the size of the key ring, most of the neighboring nodes can find a shared key directly or

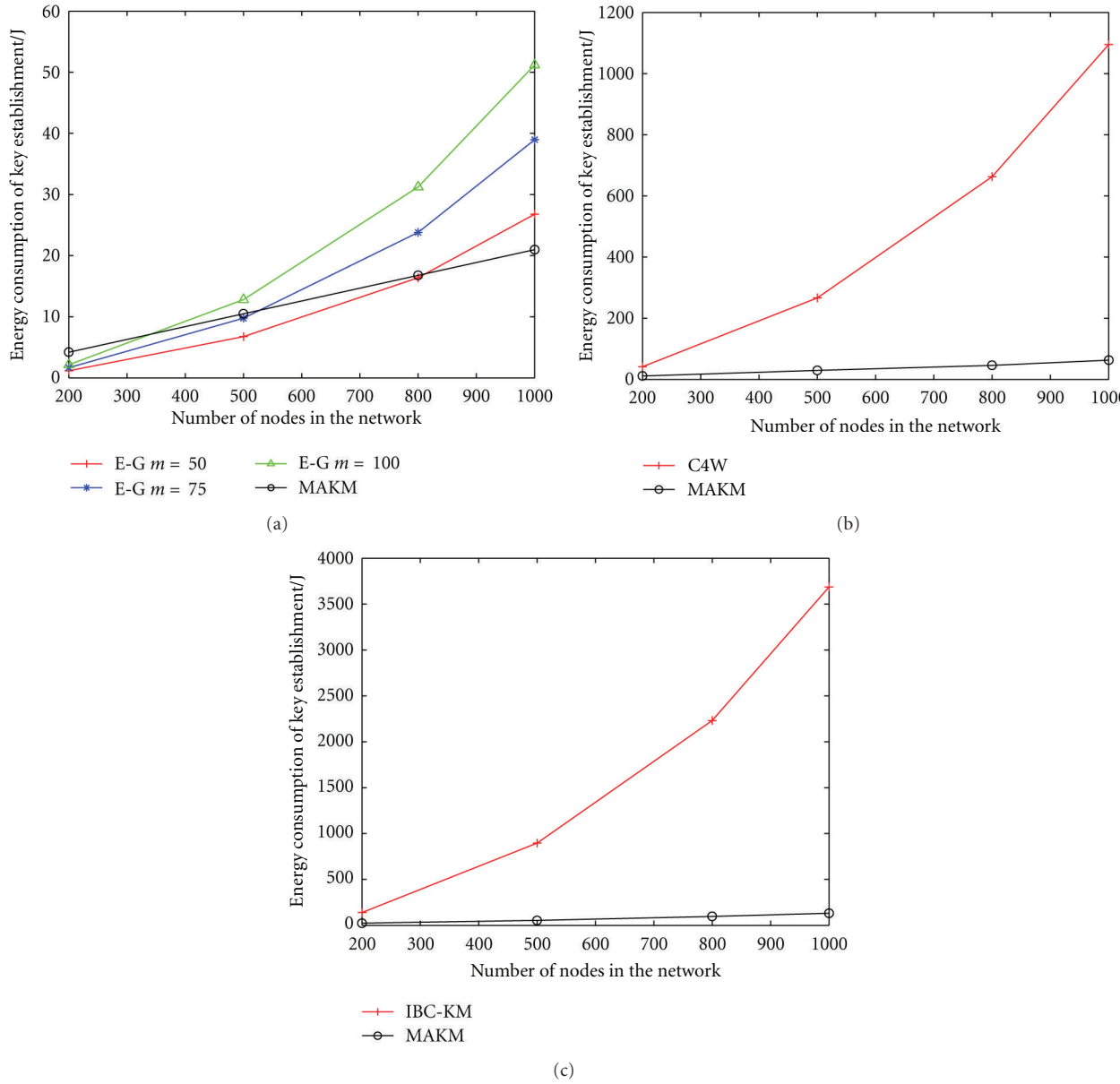

Figure 10: Energy consumption of key establishment. (a) Comparison of E-G and MAKM on TelosB platform; (b) comparison of C4 W and MAKM on MICA2 platform; (c) comparison of IBC-KM and MAKM on MICAz platform.

set up a path key within two hops. According to (16) and Figure 8, we can find that if each node is preloaded with 100 keys, it will take less time for the network to finish key establishment than each node with 75 or 50 keys.

In our experiments, we choose AES [21] as the encryption algorithm. We determine the model of the sensor MICAz that we use to estimate and analyze the time consumption of key management protocols. The MICAz is based on the low-power 8-bit microcontroller ATmega128 L with a clock frequency of 7.37 MHz. Sensor nodes run TinyOS and embed an IEEE 802.15.4 compliant CC2420 transceiver with a claimed data rate of 250 kbps [19]. The transmission power is set to be 0.065 W, and the receive power is 0.072 W. Table 4 shows the time consumption of some algorithms [12, 22, 23]. Since E-G and MAKM need much less time than C4 W and IBC-KM, we draw it in Figure 9(a). The simulation results of time consumption of key establishment are shown in Figure 9.

From Figure 9(a), we can find that if there are few nodes in the network, E-G scheme needs less time to set up shared keys than MAKM scheme. However, E-G scheme will need more time to finish key establishment as the number of sensor nodes in the network increases, while the time consumption of our proposed MAKM is nearly constant. This is because the time consumption of our scheme is only determined by the ratio of cluster heads to member nodes,

and the ratio is a constant in the network. Meanwhile, all nodes can verify their cluster head certificates at the same time. However, both the node density and the key ring size will affect the time consumption of key establishment under E-G scheme. A small key ring means that more neighbors will need to set up shared keys via path key establishment phase, which will prolong the time of key establishment. Since the number of sensor nodes in a wireless sensor network is always very large in practice, our scheme can achieve better performance in time consumption of key establishment than E-G scheme in large-scale networks. Figure 9(b) shows the time consumption of key establishment of C4 W, IBC-KM, and the proposed MAKM. We can get that C4 W and IBC-KM cost much more time than MAKM. This is because pairing and point multiplication algorithms need much time in a sensor node, which is shown in Table 4. What is more, as the number of nodes in the network increases, a sensor node will need more time to establish shared keys with its neighbors.

5.2. Energy Consumption of Key Establishment. Although security is a critical issue in wireless sensor networks, it is also necessary to consider the energy consumption of sensor nodes because they are constraint in power supply. If some sensor nodes run out of energy, the performance of the whole network will be degraded. In our MAKM scheme, we have tried to provide an energy-efficient solution. We run simulations to compare the energy consumption of our MAKM key establishment scheme with E-G scheme. Table 5 shows the energy consumption of some algorithms on different sensor platforms [19, 24, 25]. Authors in [11, 12] used energy models of MICA2 and MICAz, so we choose different sensor models to compare their energy consumption of key establishment. The simulation results are shown in Figure 10.

The number of sensor nodes varies from 200 to 1000. Other parameters are listed in Table 3. The simulation results are shown in Figure 10. We can find that C4 W and IBC-KM cost much more energy than E-G and MAKM. This is due to the too much energy cost by point multiplication, pairing, and mapping algorithms. Each node needs to operate these algorithms many times, and the number of operation times is determined by the number of a sensor node neighbors. However, in MAKM, the certificate verification is operated by a member sensor node only once. Compared with E-G, MAKM costs more energy in small-scale networks. However, the energy consumption of E-G increases much more quickly than MAKM. Since more nodes are deployed into the network, more shared keys need to be established for one node. But in our proposed MAKM, the number of shared keys for a sensor node will not be affected by the density of the network. So MAKM can reduce energy consumption of key establishment in large-scale WSNs.

Table 4: Time consumption of operations on MICAz.

Algorithm	Time/ms
AES-128 encryption	1.53
AES-128 decryption	3.52
HASH	3.75
Certificate verification	938
Point multiplication	710
Pairing	2 660
Mapping	1 550

Table 5: Energy consumption of operations on different platforms.

Algorithm	Energy Consumption/mJ		
	TelosB	MICA2	MICAz
AES-128 encryption	0.009	0.02592	0.03908
AES-128 decryption	0.014	0.04482	0.08909
HASH	0.032	0.118	0.0875
Certificate verification	19	45.09	33.8
Point multiplication	—	30.02	—
Pairing	—	—	62.73
Mapping	—	—	36.55

6. Conclusions

In this paper, we propose a key management scheme by using the congruence property of modular arithmetic for heterogeneous wireless sensor networks. In our scheme, each member node in the same cluster can use a key seed to compute its unique shared key with its cluster head and a group key shared with other nodes in the same cluster. Compared with E-G, MAKM reduces much storage space. Meanwhile, it has a better resilience against node capture. The analytical and simulation results show that in large-scale WSNs, MAKM can perform better in time consumption and energy consumption of key establishment than E-G. We also compare the proposed MAKM with two typical public-key-based key management schemes. Though MAKM needs a little more RAM space to store key seeds than C4 W and IBC-KM, it can save much time and energy consumption. Further, the proposed MAKM scheme is efficient in key seed update. In E-G, once a sensor node is compromised, it will revoke all keys in the node. However, many other nodes have set up secure links by using the same keys in the compromised node. Thus, the affected nodes need to find a shared secret key by shared key discovery, even path key establishment phase. In our MAKM, the cluster head can keep forward secrecy by broadcasting a key update message.

Though it seems that ECC-based scheme consumes more energy, luckily, most computation has been finished by the offline base station. Signatures and public keys are predistributed in sensor nodes. Sensor nodes only need to verify the signatures of their cluster heads. Thus, to implement MAKM in large-scale WSNs is a good choice. It can save much storage space and reduce energy consumption compared with other schemes.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Project no. NSFC60879024).

References

[1] 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 (CCS '02*, pp. 41–47, ACM Press, Washington, DC, USA, November 2002.

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

[3] 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, ACM Press, October 2003.

[4] X. Du, Y. Xiao, M. Guizani, and H.-H. Chen, "An effective key management scheme for heterogeneous sensor networks," *Ad Hoc Networks*, vol. 5, no. 1, pp. 24–34, 2007.

[5] K. Lu, Y. Qian, M. Guizani, and H.-H. Chen, "A framework for a distributed key management scheme in heterogeneous wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 2, pp. 639–647, 2008.

[6] Z. Yu and Y. Guan, "A key management scheme using deployment knowledge for wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 10, pp. 1411–1425, 2008.

[7] Z. Liu, J. Ma, Q. Huang, and S. Moon, "Asymmetric key predistribution scheme for sensor networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 3, pp. 1366–1372, 2009.

[8] H. T. T. Nguyen, M. Guizani, M. Jo, and E.-N. Huh, "An efficient signal-range-based probabilistic key predistribution scheme in a wireless sensor network," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 5, pp. 2482–2497, 2009.

[9] 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, ACM Press, October 2003.

[10] M. F. Younis, K. Ghumman, and M. Eltoweissy, "Location-aware combinatorial key management scheme for clustered sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 8, pp. 865–882, 2006.

[11] Q. Jing, J. Hu, and Z. Chen, "C4W: an energy efficient public key cryptosystem for large-scale wireless sensor networks," in *Proceedings of the IEEE International Conference on Mobile Ad Hoc and Sensor Sysetems (MASS '06)*, pp. 827–832, October 2006.

[12] P. Szczechowiak and M. Collier, "Practical identity-based key agreement for secure communication in sensor networks," in *Proceedings of the 18th International Conference on Computer Communications and Networks (ICCCN '09)*, pp. 1–6, August 2009.

[13] P. Traynor, R. Kumar, H. Choi, G. Cao, S. Zhu, and T. L. Porta, "Efficient hybrid security mechanisms for heterogeneous sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 663–676, 2007.

[14] W. Tang, X. H. Nan, and Z. Chen, "Combined public key system," in *Proceedings of the IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM '04)*, 2004.

[15] 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 *Proceedings of the IEEE Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, pp. 586–597, March 2004.

[16] SEC1, Elliptic curve cryptography, Certicom Corporation, 2000.

[17] A. Liu and P. Ning, "TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, pp. 245–256, April 2008.

[18] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Securing mobile ad hoc networks with certificateless public keys," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 4, pp. 386–399, 2006.

[19] G. Meulenaer, F. Gosset, F. X. Standaert, and O. Pereira, "On the energy cost of communication and cryptography in wireless sensor networks," in *Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communication*, pp. 580–585, October 2008.

[20] L. Yen and C. Yu, "Link probability, network coverage, and related properties of wireless ad hoc networks," in *Proceedings of the IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, pp. 525–527, October 2004.

[21] NIST and V. A. Springfield, Advanced Encryption Standard (AES), 2001.

[22] B. Driessen, A. Poschmann, and C. Paar, "Comparison of innovative signature algorithms for WSNs," in *Proceedings of the 1st ACM Conference on Wireless Network Security*, pp. 30–35, March 2008.

[23] J. Lee, K. Kapitanova, and S. Son, "The price of security in wireless sensor networks," *Computer Networks*, vol. 54, no. 17, pp. 2967–2978, 2010.

[24] A. Wandert, N. Gura, H. Eberle, V. Gupta, and S. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications*, pp. 324–328, March 2005.

[25] K. Piotrowski, P. Langendoerfer, and S. Peter, "How public key cryptography influences wireless sensor node lifetime," in *Proceedings of the 4th ACM Workshop on Security of Ad Hoc and Sensor Networks*, pp. 169–176, October 2006.