Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 816549, 13 pages
http://dx.doi.org/10.1155/2014/816549

Research Article

A Complete Hierarchical Key Management Scheme for Heterogeneous Wireless Sensor Networks

Chien-Ming Chen,[1,2] Xinying Zheng,[1] and Tsu-Yang Wu[1,2]

[1] *School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen 518055, China*
[2] *Shenzhen Key Laboratory of Internet Information Collaboration, Shenzhen 518055, China*

Correspondence should be addressed to Tsu-Yang Wu; wutsuyang@gmail.com

Received 28 February 2014; Accepted 26 March 2014; Published 10 April 2014

Academic Editors: T. Cao, M. Ivanovic, and F. Yu

Heterogeneous cluster-based wireless sensor networks (WSN) attracted increasing attention recently. Obviously, the clustering makes the entire networks hierarchical; thus, several kinds of keys are required for hierarchical network topology. However, most existing key management schemes for it place more emphasis on pairwise key management schemes or key predistribution schemes and neglect the property of hierarchy. In this paper, we propose a complete hierarchical key management scheme which only utilizes symmetric cryptographic algorithms and low cost operations for heterogeneous cluster-based WSN. Our scheme considers four kinds of keys, which are an individual key, a cluster key, a master key, and pairwise keys, for each sensor node. Finally, the analysis and experiments demonstrate that the proposed scheme is secure and efficient; thus, it is suitable for heterogeneous cluster-based WSN.

1. Introduction

Recently, wireless sensor networks (WSN) become more and more popular since they have been deployed in various applications, such as military, environmental monitor, industry automation, and smart space. A WSN is composed of a large number of sensor nodes which work together by collaborating with each other. In fact, sensor nodes are constrained in computing, communication, and energy capability; therefore, energy saving and hardware complexity are necessary to be considered carefully when constructing a WSN. For example, asymmetric cryptographic algorithms such as RSA or high cost operations like modular exponentiation operations are not appropriate for designing security mechanisms for a WSN.

Generally, all sensor nodes in a WSN may be divided into several small groups which are known as clusters [1–3]. Each cluster would have a cluster head responsible for collecting and aggregating sensing data from its cluster members. A cluster-based WSN can be implemented in both homogeneous WSN and heterogeneous WSN. We first consider the case that implementing a cluster-based WSN in homogeneous WSN. Note that all sensor nodes in homogeneous WSN have the same capabilities. After being deployed, every sensor node within the same cluster elects one as a cluster head. Obviously, the workload of acting a cluster head is heavier than a sensor node; as a result, the sensor node which acts as a cluster head would run out of its battery before other sensor nodes. Although it can be solved by rotating the role of the cluster head periodically over all cluster members, the workload of being a cluster head is still heavy for a sensor node. Actually, several studies [4, 5] have demonstrated that a homogeneous ad hoc network has poor performance and scalability.

Hence, several researches have concentrated on a heterogeneous WSN which incorporate different types of sensor nodes with different capabilities. For example, a WSN may contain a small number of powerful high-end sensor nodes (H-Sensors) and a large number of low-end sensor nodes (L-Sensors). If implementing a cluster-based WSN in heterogeneous WSN, H-Sensors organize L-Sensors around them into clusters, and L-Sensors forward sensing reports to the corresponding H-Sensor. It is commonly referred to as the heterogeneous cluster-based WSN. The advantage of it is the overall hardware cost of the entire WSN that can be reduced. This is because L-Sensors, which only perform the basic functions, can be manufactured very cheap and simple.

In another aspect, security is a vital issue in various WSN applications. Thus, an efficient key management scheme is necessary. Depending on different applications and security requirements, a variety of key management schemes have been proposed for a heterogeneous WSN [6–10]. In fact, most of these schemes place more emphasis on pairwise key establishment and key predistribution. However, due to the property of a heterogeneous cluster-based WSN that makes the topology hierarchical, a hierarchical key management scheme for cluster-based heterogeneous WSN is essential. Except for an pairwise key, which is shared between two sensor nodes, an individual key, which is shared between each sensor node and the base station, a common cluster key for each cluster, and a master key for all sensor nodes are also required.

Moreover, key updating is also required to be considered for the following reasons. First, sensor nodes may be compromised. The affected keys must be updated or revoked. Second, the network topology may be dynamic. For example, sensor nodes are deployed into the sea for aquatic application. If a sensor node moves to another neighboring cluster, some keys may need to be updated. Third, the base station may desire to update the master key periodically for better security level. For the best of our knowledge, no hierarchical key management scheme with key updating functionality for a heterogeneous cluster-based WSN has been proposed.

In this paper, we propose a complete and efficient hierarchical key management system for a heterogeneous cluster-based WSN. Our design only utilizes symmetric cryptographic algorithms and low cost operations such as bitwise XOR operation and modular multiplication. This construction contains two schemes, key generating scheme and key updating scheme. In the key generating scheme, four kinds of keys, which are an individual key, pairwise keys, a cluster key, and a master key, are generated for each L-Sensor. In the key updating scheme, we place the emphasis on updating cluster keys and the master key. In order to improve better efficiency, our design reduces the usage of unicasting. In the performance evaluation, we demonstrate that the storage requirement, communication, and computation cost are reasonable. Besides, in the security analysis, we show that the proposed construction is secure and influence of compromising can be confined to the affected cluster. Finally, the experimental results demonstrate that the operations used in our design are practical.

The remainder of this paper is organized as follows. In Section 2, some preliminaries are introduced. We describe the related work, network model, attack model, and our design goals. In Section 3, we propose our key generating scheme. In Section 4, the proposed key updating scheme is introduced. Then, Sections 5 and 6 describe the security analysis and performance evaluation of our design, respectively. We further provide experiments in Section 7. Finally, Section 8 concludes.

2. Preliminaries

In this section, the related work is introduced firstly. Then we describe the network model of heterogeneous cluster-based WSN and the attack model. We also list our design goals.

2.1. Related Work. Since several studies [4, 5] demonstrated that a homogeneous WSN has poor performance and scalability; several recent works investigate a heterogeneous WSN. Duarte-Melo and Liu [11] analyzed the energy consumption and lifetime of a heterogeneous WSN. Girod et al. [12] developed tools to support a heterogeneous WSN and measurement and visualization of operational systems. Du and Lin [13] proposed a differentiated coverage algorithm which can provide different coverage degrees for different areas. Lazos and Poovendran [14] studied the problem of coverage in planar heterogeneous WSNs. Lin et al. [15] proposed an ant colony optimization-based approach to maximize the lifetime of a heterogeneous WSN. Chen et al. [16] proposed a recoverable data aggregation scheme for a heterogeneous cluster-based WSN. As shown above, heterogeneous WSN indeed received lots of attention.

In the view of key management or key distribution schemes, several researches have been proposed [6–9, 17, 18]. Most of these schemes focus on probabilistic key predistribution method. Du et al. [6] presented an asymmetric key management scheme which preloads a large number of keys in each H-Sensor while preloading a small number of keys in L-Sensors. Later, Hussain et al. [8] also proposed key predistribution scheme, which reduces the storage requirements while maintaining the same security strength. Durresi et al. [7] proposed key predistribution schemes between stationary nodes and nonstationary nodes. Traynor et al. [9] described three keying and trust models for the heterogeneous WSN. Khan et al. [17] presented a key management scheme supporting mobility in a heterogeneous WSN that consists of mobile sensor nodes with few fixed sensor nodes. Shi et al. [18] proposed a resource-efficient authentic key establishment scheme for a heterogeneous WSN.

The above schemes for heterogeneous WSNs put more emphasis on pairwise key distribution and ignore the property of hierarchy. Thus, in this paper, we propose a hierarchical key management construction with the functionality of key updating.

2.2. The Network Model of Heterogeneous Cluster-Based WSN. The network model of a heterogeneous cluster-based WSN contains three components, a base station, a small number of powerful high-end sensor nodes (H-Sensors), and a large number of low-end sensor nodes (L-Sensors). H-Sensors are expected to have more energy than L-Sensors. They organize L-Sensors into clusters, collect and aggregate sensing data from their cluster members (L-Sensors), and send the results to the base station. Besides, an H-Sensor is equipped with a tamper-resistant hardware.

On the other hand, L-Sensors, which have sensing capability with limited computation, memory, and communication, are small and low-cost devices. Each L-Sensor detects a target within its detection range, uses its processing power

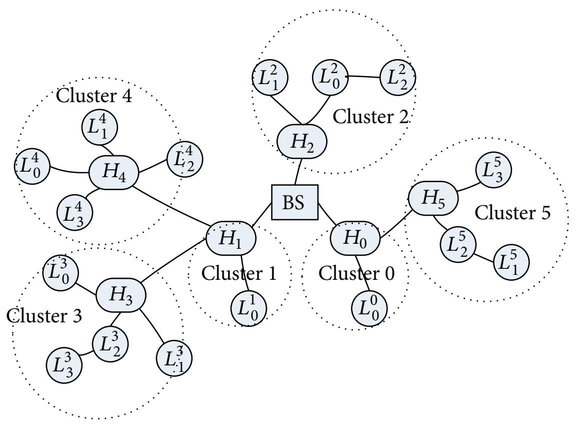


Figure 1: An example of a heterogeneous clustered-based WSN.

to locally perform simple computations, and then sends the required data to the corresponding H-Sensor. Besides, an L-Sensor is not equipped with tamper-resistant hardware; therefore, an adversary can obtain the information stored in an L-Sensor after compromising it.

Figure 1 illustrates a simple example of a heterogeneous clustered-based WSN. Note that H_i denotes the H-Sensor i and L_i^j denotes the L-Sensor; i belongs to H_j. Depending on different environments, there might be more than one level of H-Sensors between the base station and L-Sensors. In Figure 1, cluster 3 contains L_0^3, L_1^3, L_2^3, L_3^3, and H_3. H_1, an upper level H-Sensor of H_3, forwards the data sent from H_3 to the base station.

Due to the properties of a heterogeneous WSN, the communication capacity of the base station, H-Sensors and L-Sensors are different. The communication can be classified into the following categories.

(1) Within a cluster: an H-Sensor can broadcast/unicast messages to its cluster member through single hop, whereas messages sent to the H-Sensor may require multihop or still single hop depending on the distance between them.

(2) Between two neighboring H-Sensors: an H-Sensor can communicate to neighboring H-Sensors through single hop.

(3) Between the base station and H-Sensors: no doubt, messages sent to the base station require hop by hop. For example, in Figure 1, messages which are sent by H_3 to the base station would pass through H_1. On the other hand, the base station can broadcast/unicast to all H-Sensors through single hop or multihops.

2.3. Attack Models

2.3. Attack Models. Here we discuss the attack model for key management schemes in a WSN. Depending on the abilities of adversary, attacks can be categorized into two situations.

(1) Without compromising sensor nodes: an adversary can only eavesdrop packets or send false messages to legal sensor nodes without any knowledge.

(2) Compromising sensor nodes: after compromising a sensor node, an adversary can obtain the information stored in this sensor node. He may calculate other keys or secrets through the compromised information.

Note that detecting compromised sensor nodes which still act as normal sensor nodes is infeasible in all existing detection mechanisms in WSN. However, if an adversary compromises a sensor node and performs abnormal behavior or attacks, it will be detected [19–23]. Besides, in a heterogeneous WSN, it is assumed that every H-Sensor is equipped with a temper-resistant hardware; thus, considering the case that he compromises an H-Sensor is not required.

Based on the above attack models, a key management scheme for a WSN must satisfy the following security requirements.

Requirement 1: all messages, including sensing data and control messages, must be encrypted.

Requirement 2: an adversary cannot compromise the entire WSN with the compromised secrets. More specifically, he cannot derive the secrets that belong to other clusters. Furthermore, if he compromises two sensor nodes which belong to different clusters, he cannot obtain the secrets with other clusters either.

Requirement 3: if compromised sensor nodes are detected, the affected keys must be updated or revoked.

2.4. Design Goals. Four kinds of keys, individual key, master key, cluster key, and pairwise key, are generated in the proposed construction.

(1) Individual key: every L-Sensor and H-Sensor share an individual key with the base station. The base station can encrypt the secret information with the individual key if required.

(2) Cluster key: every cluster has one cluster key which is shared with all cluster members including one H-Sensor and several L-Sensors. The cluster key is utilized for encrypting the cluster traffic. An H-Sensor can securely transfer control messages to its cluster members with this key. For example, an H-Sensor may turn some cluster members into sleep mode; it can encrypt this control message with the cluster key. On the other hand, all L-Sensors within the same cluster may encrypt the sensing reports with the same cluster key. The importance of cluster key is also discussed in [24].

(3) Master key: this master key is shared between all L-Sensors and the base station within the entire network. The base station can securely broadcast the information to all L-Sensors with the master key.

(4) Pairwise key: the pairwise key is shared with two neighboring L-Sensors. In some applications, two neighboring L-Sensors may require secure channel to protect their communication. For example, if

the cluster key is compromised, L-Sensors in this cluster may encrypt the data with pairwise keys.

In this paper, we also consider how to update some of the above keys efficiently. Sensor nodes may be compromised or move to another neighboring cluster; consequently, the master key and the corresponding cluster keys must be updated. Besides, the master key may be updated periodically for security considerations.

3. The Proposed Key Generating Scheme

In this section, we propose a key generating scheme for a heterogeneous cluster-based WSN. As mentioned above, this scheme generates four kinds of keys, individual key, master key, cluster key, and pairwise key. Notations used in this paper are summarized as shown in Notation section.

3.1. Generating Each Individual Key. Actually, individual keys of each L-Sensor and H-Sensor are preloaded before being deployed. More specifically, an individual key $K_{L_i^j,\mathrm{BS}}$ is preloaded to L_i^j and $K_{H_j,\mathrm{BS}}$ is preloaded to H_j.

After deployment, H-Sensors partition all L-Sensors into several clusters. Each H-Sensor can realize which L-Sensors are organized in its cluster. Then every H-Sensor reports its cluster information to BS.

Before generating other keys, BS requires to securely assign each L-Sensor a key which is shared with the attached H-Sensor. For example, $K_{L_i^j,H_j}$ which is assigned to L_i^j is shared between L_i^j and H_j. Since BS realizes the cluster information of each cluster, it can securely send the key $K_{L_i^j,H_j}$ encrypted with $K_{H_j,\mathrm{BS}}$ or $K_{L_i^j,\mathrm{BS}}$ to every H-Sensor and L-Sensor. Hence, all L-Sensors would share the keys with their attached H-Sensors. Note that the key $K_{L_i^j,H_j}$ is transmitted only once before the stages of generating other kinds of keys.

3.2. Generating Cluster Keys. The procedure of generating cluster keys is modified from [25, 26]. To generate the cluster key, each H-Sensor constructs a binary tree and assigns its cluster members to leaf nodes. The root of the binary tree is the cluster key and intermediate nodes are key encryption keys. Note that the key encryption keys are used for updating the cluster key. Each L-Sensor knows all the keys from the parent of its corresponding leaf node up to the root. This set of keys is called the key path. The procedure of generating the cluster key is described as follows.

If there are n_j L-Sensors in cluster j, the H-Sensor H_j will generate random elements $R_0, R_1, \ldots, R_{\lceil\log_2(n_j)\rceil-1}$ and $K_0, K_1, \ldots, K_{\lceil n_j/2\rceil-1}$ to calculate key encryption keys and the cluster key. Both key encryption keys and the cluster key are composed of one random element R_X and several random elements K_Y, where $0 \le X \le \lceil\log_2(n_j)\rceil - 1$ and $0 \le Y \le \lceil n_j/2\rceil - 1$. R_X means that this key is at level X in the key tree and K_Y means that this key belongs to the key path Y. All keys on the same key path Y have the same random elements K_Y.

The example in Figure 2 shows 8 L-Sensors in cluster 0. H_0 constructs a key tree with 8 leaves and generates random elements R_0, R_1, R_2, K_0, K_1, K_2, and K_3. In Figure 2, KEK_3^0 ($=R_2 \times K_0 \bmod p$), KEK_1^0 ($=R_2 \times K_0 \times K_1 \bmod p$), and CK_0 ($=R_0 \times K_1 \times K_2 \times K_3 \bmod p$) are on the key path 0, KEK_4^0, KEK_1^0, and CK_0 are on the key path 1, and so on. KEK_5^0, KEK_2^0, and CK_0 belong to key path of L_4^0 and L_5^0. Note that p is a 128-bit prime number. Since CK_0 is at level 0 in the key tree and belongs to the key path 0, 1, 2, and 3, it is composed of R_0, K_0, K_1, K_2, and K_3. After the key tree is constructed, H_0 assigns each cluster member L_i^0 to a leaf node and securely sends the cluster key CK_0 and key encryption keys on the key path to L_i^0 using the key $K_{L_i^0,H_0}$.

3.3. Generating the Master Key. Assume that there are m H-Sensors which are denoted as $\{H_0, H_1, \ldots, H_{m-1}\}$ in a WSN. Before generating the master key, BS needs to deliver the secure information $\mathrm{SI}_{H_j,\mathrm{BS}}$ and $\mathrm{SI}_{L_i^j,\mathrm{BS}}$ to each H-Sensor and L-Sensor. Note that this secure information is used when generating and updating other keys (see Section 4.2).

BS first chooses a 128-bit prime number p, where p is public, and two secret random numbers $S, W \in Z_p^*$. It also selects m distinct numbers which are denoted as $\{e_0, e_1, \ldots, e_{m-1}\}$ from Z_p^*. BS then securely sends the following message to all H-Sensors:

$$\forall 0 \le j < m, \quad \mathrm{BS} \longrightarrow H_j : E_{K_{H_j,\mathrm{BS}}}\left(\mathrm{SI}_{H_j,\mathrm{BS}}, \mathrm{SI}_{L_i^j,\mathrm{BS}}\right), \quad (1)$$

where $\mathrm{SI}_{H_j,\mathrm{BS}} = e_j \times W^{-1} \bmod p$ and $\mathrm{SI}_{L_i^j,\mathrm{BS}} = S \times e_j^{-1} \bmod p$. After receiving it, each H-Sensor, for example, H_j, broadcasts $E_{\mathrm{CK}_j}(\mathrm{SI}_{L_i^j,\mathrm{BS}})$ to all its cluster members. Note that CK_j is the cluster key of cluster j. As a result, all its cluster members can obtain $\mathrm{SI}_{L_i^j,\mathrm{BS}}$. Note that L-Sensors attached to the same H-Sensor will have the same secure information $\mathrm{SI}_{L_i^j,\mathrm{BS}}$. Figure 3 shows an example of a WSN with secure information.

After that, BS starts to generate the master key. The procedure of master key generation is described as follows.

Step 1. BS first selects a random number $r \in Z_p^*$ and computes a master key $K_{\mathrm{Master}} = S \times r \bmod p$.

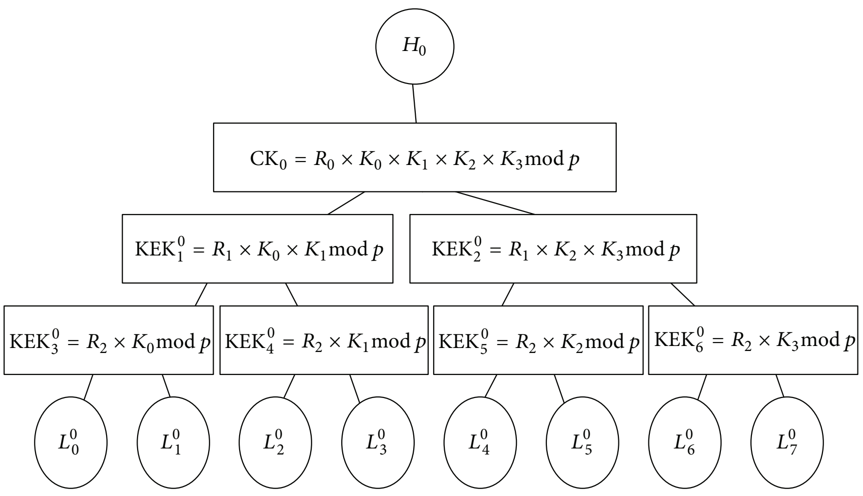
Step 2. BS broadcasts $r \times W \bmod p$ to all H-Sensors.

Step 3. When each H-Sensor receives $r \times W \bmod p$ from BS, it calculates the following equations and broadcasts the result to all its cluster members.

For all $0 \le j \le m-1$, H_j calculates

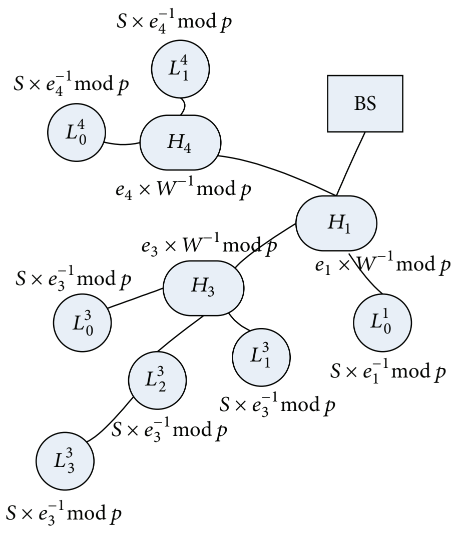
$$\begin{aligned}
&\left(\left(\mathrm{SI}_{H_j,\mathrm{BS}}\right) \times (r \times W \bmod p) \bmod p\right) \oplus \mathrm{CK}_j \\
&\quad = \left(\left(e_j \times W^{-1}\right) \times (r \times W \bmod p) \bmod p\right) \oplus \mathrm{CK}_j \qquad (2) \\
&\quad = \left(r \times e_j \bmod p\right) \oplus \mathrm{CK}_j .
\end{aligned}$$

We use bitewise XOR operation $\oplus$ to guarantee that messages can be securely sent to legitimate L-Sensors.



H_0
$\mathrm{CK}_0 = R_0 \times K_0 \times K_1 \times K_2 \times K_3 \bmod p$
$\mathrm{KEK}_1^0 = R_1 \times K_0 \times K_1 \bmod p$
$\mathrm{KEK}_2^0 = R_1 \times K_2 \times K_3 \bmod p$
$\mathrm{KEK}_3^0 = R_2 \times K_0 \bmod p$
$\mathrm{KEK}_4^0 = R_2 \times K_1 \bmod p$
$\mathrm{KEK}_5^0 = R_2 \times K_2 \bmod p$
$\mathrm{KEK}_6^0 = R_2 \times K_3 \bmod p$
L_0^0 L_1^0 L_2^0 L_3^0 L_4^0 L_5^0 L_6^0 L_7^0

Figure 2: An example of a key tree.



$S \times e_4^{-1} \bmod p$
L_1^4
$S \times e_4^{-1} \bmod p$
L_0^4
H_4
BS
$e_4 \times W^{-1} \bmod p$
H_1
$e_3 \times W^{-1} \bmod p$
$e_1 \times W^{-1} \bmod p$
$S \times e_3^{-1} \bmod p$
H_3
L_0^3
L_0^1
L_1^3
$S \times e_1^{-1} \bmod p$
L_2^3
$S \times e_3^{-1} \bmod p$
$S \times e_3^{-1} \bmod p$
L_3^3
$S \times e_3^{-1} \bmod p$

Figure 3: An example of the WSN with secure information.

Step 4. When L_i^j receives $(r \times e_j \bmod p) \oplus \mathrm{CK}_j$ from H_j, it computes the master key K_{Master}, where

$$\begin{aligned} K_{\mathrm{Master}} &= \mathrm{SI}_{L_i^j,\mathrm{BS}} \times \left(r \times e_j \bmod p\right) \\ &= \left(S \times e_j^{-1} \bmod p\right) \times \left(r \times e_j \bmod p\right) \qquad (3) \\ &= S \times r \bmod p. \end{aligned}$$

By the above steps, all L-Sensors have the same master key K_{Master}.

3.4. Generating Pairwise Keys. The pairwise key shared with two neighboring L-Sensors can be generated through their corresponding H-Sensor. For example, if L_0^1 desires to share a pairwise key with L_1^1, H_1 will generate this pairwise key and send $E_{K_{L_0^1,H_1}}$(pairwise_key) and $E_{K_{L_1^1,H_1}}$(pairwise_key) to L_0^1 and L_1^1, respectively; consequently, both L_0^1 and L_1^1 can obtain this pairwise key.

4. The Proposed Key Updating Scheme

In this section, we propose the key updating scheme to update some of the generated keys. We discuss these kinds of keys separately as follows.

(1) Individual key: since it is only shared between BS and each L-Sensor, this key is revoked automatically if an L-Sensor is compromised.

(2) Cluster key: obviously, if an L-Sensor is compromised, the corresponding cluster requires updating the cluster key. Besides, L-Sensors may move to another neighboring clusters if the deployment of L-Sensors is nonstationary; consequently, both clusters (the original cluster and the target cluster) require updating their cluster key, respectively.

(3) Master key: similarly, if L-Sensors are compromised, the master key must be updated. Besides, BS may desire to update the master key periodically for better security considerations.

(4) Pairwise key: for example, assume that two L-Sensors, L_i^n and L_j^n, have shared a pairwise key. If L_i^n is compromised, L_j^n would revoke the shared pairwise key automatically. Besides, these two L-Sensors can also update this key periodically.

4.1. Updating the Cluster Key. Here we discuss how to update cluster keys. First, we consider that an L-Sensor leaves a cluster. It is because this L-Sensor is compromised or moves to other neighboring cluster. Second, we consider an L-Sensor joins a cluster.

4.1.1. An L-Sensor Leaves a Cluster. In Figure 2, if L_6^0 leaves cluster 0, the keys CK_0, KEK_2^0, and KEK_6^0 must be updated. The detailed procedure is described in the following.

Step 1. H_0 selects a random number K_3' from Z_p^* and sends the following key update messages:

(i) $H_0 \rightarrow \{L_0^0, \ldots, L_3^0\}$: $\mathrm{KEK}_1^0 \oplus ((\mathrm{KEK}_1^0) \times (K_3^{-1} \times K_3' \bmod p) \bmod p)$,

(ii) $H_0 \rightarrow \{L_4^0, L_5^0\}$: $\mathrm{KEK}_5^0 \oplus ((\mathrm{KEK}_5^0) \times (K_3^{-1} \times K_3' \bmod p) \bmod p)$,

(iii) $H_0 \rightarrow \{S_7^0\}$: $K_{L_7^0,H_0} \oplus ((K_{L_7^0,H_0}) \times (K_3^{-1} \times K_3' \bmod p) \bmod p)$.

Step 2. S_0^0, S_1^0, S_2^0, and S_3^0 can obtain $K_3^{-1} \times K_3' \bmod p$ with KEK_1^0 and then compute the new cluster key CK_0' where

$$\begin{aligned}\mathrm{CK}_0' &= \mathrm{CK}_0 \times \left(K_3^{-1} \times K_3' \bmod p\right) \bmod p \\ &= R_0 \times K_0 \times K_1 \times K_2 \times K_3' \bmod p.\end{aligned} \tag{4}$$

Step 3. S_4^0 and S_5^0 can obtain $K_3^{-1} \times K_3' \bmod p$ with KEK_5^0 and then compute the new keys CK_0' and $\mathrm{KEK}_2^{0\prime}$ where

$$\begin{aligned}\mathrm{KEK}_2^{0\prime} &= \mathrm{KEK}_2^0 \times \left(K_3^{-1} \times K_3' \bmod p\right) \bmod p \\ &= R_1 \times K_2 \times K_3' \bmod p.\end{aligned} \tag{5}$$

Step 4. S_7^0 can obtain $K_3^{-1} \times K_3' \bmod p$ with $K_{L_7^0,H_0}$ and then compute the new keys CK_0', $\mathrm{KEK}_2^{0\prime}$, and $\mathrm{KEK}_6^{0\prime}$ where

$$\begin{aligned}\mathrm{KEK}_6^{0\prime} &= \mathrm{KEK}_6^0 \times \left(K_3^{-1} \times K_3' \bmod p\right) \bmod p \\ &= R_2 \times K_3' \bmod p.\end{aligned} \tag{6}$$

Obviously, it only requires updating the value K_3 in this example. Since L_6^0 has no keys to obtain $K_3^{-1} \times K_3' \bmod p$, it cannot compute the new cluster key and key encryption keys.

4.1.2. An L-Sensor Joins a Cluster. When an L-Sensor joins a new cluster, the target H-Sensor authenticates this joined L-Sensor. This can be accomplished by coordinating with the original H-Sensor. Then BS would deliver a key which will be shared between the target H-Sensor and this L-Sensor. After receiving this key, the H-Sensor would assign this new L-Sensor a leaf node of the key tree. To prevent this new L-Sensor from decrypting the past traffic, all the keys on this key path need to be updated.

Let us take Figure 2 as an illustration. Assume that L_6^0 joins this cluster and has received an individual key $K_{L_6^0,H_0}$. The procedure of updating the cluster key is described as follows.

Step 1. H_0 assigns L_6^0 a leaf node of the key tree. H_0 then selects a random number K_3' from Z_p^* to compute the new keys, CK_0', $\mathrm{KEK}_2^{0\prime}$, and $\mathrm{KEK}_6^{0\prime}$, where

$$\begin{aligned}\mathrm{CK}_0' &= \mathrm{CK}_0 \times \left(K_3^{-1} \times K_3'\right) \bmod p \\ &= R_0 \times K_0 \times K_1 \times K_2 \times K_3' \bmod p,\end{aligned} \tag{7}$$

$$\begin{aligned}\mathrm{KEK}_2^{0\prime} &= \mathrm{KEK}_2^0 \times \left(K_3^{-1} \times K_3'\right) \bmod p \\ &= R_1 \times K_2 \times K_3' \bmod p,\end{aligned} \tag{8}$$

$$\begin{aligned}\mathrm{KEK}_6^{0\prime} &= \mathrm{KEK}_6^0 \times \left(K_3^{-1} \times K_3'\right) \bmod p \\ &= R_2 \times K_3' \bmod p.\end{aligned} \tag{9}$$

Step 2. H_0 needs to send the following two messages; one is broadcasted to all L-Sensors within this cluster for updating the key path; one is additionally unicasted to the new L-Sensor L_6^0:

(i) $H_0 \rightarrow \{L_0^0, \ldots, L_7^0\}$: $\mathrm{CK}_0 \oplus (K_3^{-1} \times K_3' \bmod p)$,

(ii) $H_0 \rightarrow L_6^0$:

$$\begin{aligned}&\left(K_{L_6^0,H_0} \oplus \left(K_{L_6^0,H_0} \times \mathrm{CK}_0' \bmod p\right)\right) \\ &\parallel \left(K_{L_6^0,H_0} \oplus \left(K_{L_6^0,H_0} \times \mathrm{KEK}_2^{0\prime} \bmod p\right)\right) \\ &\parallel \left(K_{L_6^0,H_0} \oplus \left(K_{L_6^0,H_0} \times \mathrm{KEK}_6^{0\prime} \bmod p\right)\right).\end{aligned} \tag{10}$$

Since L_6^0 does not have CK_0, it cannot obtain $K_3^{-1} \times K_3' \bmod p$ to compute the previous keys. On the contrary, all other L-Sensors in this cluster will obtain $K_3^{-1} \times K_3' \bmod p$.

Step 3. L_6^0 computes the new keys CK_0', $\mathrm{KEK}_2^{0\prime}$, and $\mathrm{KEK}_6^{0\prime}$ with $K_{L_6^0,H_0}$.

Step 4. L_0^0, L_1^0, L_2^0, and L_3^0 can compute the new key CK_0' from (7) with the obtained $K_3^{-1} \times K_3' \bmod p$.

Step 5. L_4^0 and L_5^0 can compute the new keys CK_0' and $\mathrm{KEK}_2^{0\prime}$ from (7) and (8) with the obtained $K_3^{-1} \times K_3' \bmod p$.

Step 6. L_7^0 can compute the new keys CK_0', $\mathrm{KEK}_2^{0\prime}$, and $\mathrm{KEK}_6^{0\prime}$ from (7), (8), and (9) with the obtained $K_3^{-1} \times K_3' \bmod p$.

Similarly, only the value K_3 is required to be updated.

4.1.3. Further Discussion. Notice that in Step 1 of Section 4.1.1, for example, L_0^0 will receive $K_3^{-1} \times K_3' \bmod p$ protected by KEK_1^0. The reason that we use the equation

$$\mathrm{KEK}_1^0 \oplus \left(\left(\mathrm{KEK}_1^0\right) \times \left(K_3^{-1} \times K_3' \bmod p\right) \bmod p\right) \tag{11}$$

rather than

$$\left(\mathrm{KEK}_1^0\right) \times \left(K_3^{-1} \times K_3' \bmod p\right) \bmod p \tag{12}$$

is to protect KEK_1^0. More specifically, if we only use (12), L_4^0, L_5^9, and L_7^0 can utilize the obtained $K_3^{-1} \times K_3'$ mod p to further obtain KEK_1^0. In fact, KEK_1^0 is not revealed to L_4^0, L_5^0, and L_7^0; therefore, an additional bitewise XOR operation $\oplus$ is required.

Similarly, in Step 2 of Section 4.1.2, the reason we use (10) rather than

$$\left(K_{L_6^0,H_0} \times \text{CK}_0' \bmod p\right) \parallel \left(K_{L_6^0,H_0} \times \text{KEK}_2^{0\prime} \bmod p\right) \parallel \left(K_{L_6^0,H_0} \times \text{KEK}_6^{0\prime} \bmod p\right) \tag{13}$$

is to protect $K_{L_6^0,H_0}$; otherwise, other cluster members can obtain $K_{L_6^0,H_0}$ with CK_0', $\text{KEK}_2^{0\prime}$, or $\text{KEK}_6^{0\prime}$.

4.2. Updating the Master Key. Here we discuss how to update the master key in the following situations. First, BS may update the master key periodically to improve the security level. Second, the master key is required to be updated if L-Sensors are compromised.

4.2.1. Updating the Master Key Periodically. The master key may be updated periodically, for example, monthly. The procedure of updating master key is described as follows.

Step 1. BS selects a new random number $r' \in Z_p^*$ and calculates a new master key $K_{\text{Master\_new}} = S \times r'$ mod p. BS broadcasts $r' \times W$ mod p to all H-Sensors.

Step 2. While H_j receives $r' \times W$ mod p, it calculates

$$\begin{aligned}&\left(\left(\text{SI}_{H_j,\text{BS}}\right) \times \left(r' \times W \bmod p\right) \bmod p\right) \oplus \text{CK}_j \\ &\quad = \left(r' \times e_j \bmod p\right) \oplus \text{CK}_j.\end{aligned} \tag{14}$$

H_j then broadcasts the result to all its cluster members.

Step 3. L_i^j can calculate the new master key $K_{\text{Master\_new}}$ where

$$\begin{aligned}K_{\text{Master\_new}} &= \text{SI}_{L_i^j,\text{BS}} \times \left(r' \times e_j \bmod p\right) \\ &= \left(S \times e_j^{-1} \bmod p\right) \times \left(r' \times e_j \bmod p\right) \\ &= S \times r' \bmod p.\end{aligned} \tag{15}$$

In conclusion, all L-Sensors can obtain the new master key $K_{\text{Master\_new}}$.

4.2.2. Updating the Master Key If L-Sensors Are Compromised. Assume that L-Sensor L_m^n which is attached to H_n is compromised and L_k^n denotes the L-Sensor k which is also attached to H_n, where $k \neq m$. Besides, there are m H-Sensors which are denoted as $\{H_0, H_1, \ldots, H_{m-1}\}$ in the entire WSN. The procedure of updating the master key is described as follows.

Step 1. BS selects a new random number $r' \in Z_p^*$ and calculates a new master key $K_{\text{Master\_new}} = S \times r'$ mod p. BS also selects a random number $e_n' \in Z_p^*$ to compute $e_n^{-1} \times e_n'$ mod p and broadcasts $((r' \times W \bmod p) \parallel (E_{K_{H_n,\text{BS}}}(e_n^{-1} \times e_n' \bmod p)))$ to all H-Sensors.

Step 2. Except for H_n, each H-Sensor calculates the following equations and broadcasts the result to its cluster members.

For all $0 \leq j \leq m-1$, $j \neq n$, H_j calculates

$$\begin{aligned}&\left(\left(\text{SI}_{H_j,\text{BS}}\right) \times \left(r' \times W \bmod p\right) \bmod p\right) \oplus \text{CK}_j \\ &\quad = \left(\left(e_j \times W^{-1}\right) \times \left(r' \times W \bmod p\right) \bmod p\right) \oplus \text{CK}_j \\ &\quad = \left(r' \times e_j \bmod p\right) \oplus \text{CK}_j.\end{aligned} \tag{16}$$

Then, all their cluster members can obtain the new master key $K_{\text{Master\_new}}$, where

$$\begin{aligned}K_{\text{Master\_new}} &= \text{SI}_{L_i^j,\text{BS}} \times \left(r' \times e_j \bmod p\right) \\ &= S \times r' \bmod p.\end{aligned} \tag{17}$$

Step 3. (a) H_n obtains $e_n^{-1} \times e_n'$ mod p from $E_{K_{H_n,\text{BS}}}(e_n^{-1} \times e_n' \bmod p)$ and further computes the new secure information $\text{SI}_{H_n,\text{BS}}'$ where

$$\begin{aligned}\text{SI}_{H_n,\text{BS}}' &= \left(\text{SI}_{H_n,\text{BS}}\right) \times \left(e_n^{-1} \times e_n' \bmod p\right) \bmod p \\ &= \left(e_n \times W^{-1} \bmod p\right) \times \left(e_n^{-1} \times e_n' \bmod p\right) \\ &= e_n' \times W^{-1} \bmod p.\end{aligned} \tag{18}$$

(b) H_n obtains $(r' \times e_n' \bmod p) \oplus \text{CK}_n$ from $r' \times W$ mod p and new secure information $\text{SI}_{H_n,\text{BS}}'$. Then H_n broadcasts $((r' \times e_n' \bmod p) \parallel (e_n^{-1} \times e_n' \bmod p)) \oplus \text{CK}_n \oplus \text{CK}_n'$. Note that CK_n' is the new cluster key of cluster n.

(c) Except for L_m^n, all the other L-Sensors which are also attached to H_n can derive $e_n^{-1} \times e_n'$ mod p and $r' \times e_n'$ mod p from the obtained message $((r' \times e_n' \bmod p) \parallel (e_n^{-1} \times e_n' \bmod p)) \oplus \text{CK}_n \oplus \text{CK}_n'$ and then compute $\text{SI}_{L_k^n,\text{BS}}'$, where

$$\begin{aligned}\text{SI}_{L_k^n,\text{BS}}' &= \text{SI}_{L_k^n,\text{BS}} \times \left(e_n^{-1} \times e_n' \bmod p\right)^{-1} \\ &= \left(S \times e_n^{-1} \bmod p\right) \times \left(e_n \times e_n'^{-1} \bmod p\right) \\ &= S \times e_n'^{-1} \bmod p.\end{aligned} \tag{19}$$

(d) After updating $\text{SI}_{L_k^n,\text{BS}}'$, all the other L-Sensors except L_m^n within cluster n can compute new master key $K_{\text{Master\_new}} = r' \times W$ mod p where

$$\begin{aligned}K_{\text{Master\_new}} &= \text{SI}_{L_k^n,\text{BS}}' \times \left(r' \times e_n' \bmod p\right) \\ &= \left(S \times e_n'^{-1} \bmod p\right) \times \left(r' \times e_n' \bmod p\right) \\ &= S \times r' \bmod p.\end{aligned} \tag{20}$$

Only H_n can obtain $e_n^{-1} \times e_n'$ by decrypting $E_{K_{H_n,\text{BS}}}(e_n^{-1} \times e_n' \bmod p)$. Besides, the compromised L-Sensor L_m^n cannot obtain $r' \times e'_n \bmod p$ and $e_n^{-1} \times e'_n \bmod p$ to compute the new secure information $\mathrm{SI}'_{L_k^n,\mathrm{BS}}$ without CK'_n; therefore, it is unable to obtain the new master key.

Obviously, only the L-Sensors within the affected cluster require updating the secure information before updating the master key; as a result, the influence can be confined locally.

4.2.3. Further Discussion. We have already discussed how to update the master key. It may be questioned why we need to do it in this way. A trivial idea is to let BS generate a random master key (either for master key generation or master key updating). Then BS encrypts it using each cluster key and transmits it to every H-Sensor. This method seems simpler and more straightforward.

The reason is that the proposed scheme attempts to reduce the usage of unicasting. More precisely, in the above method, BS requires to unicast each encrypted master key to corresponding H-Sensor. However, BS only needs to broadcast the same message ($r' \times W \bmod p$) to every H-Sensor in our design. Actually, using broadcast is more efficient than using unicasting. We will demonstrate it through experiments in Section 7.

4.3. Updating the Pairwise Key. Assume that two L-Sensors, L_i^n and L_j^n, have shared a pairwise key. If these two L-Sensors decide to update their shared pairwise key, H_n generates a new pairwise key and sends $E_{K_{L_i^n,H_n}}(\text{new\_pairwise\_key})$ and $E_{K_{L_j^n,H_n}}(\text{new\_pairwise\_key})$ to L_i^n and L_j^n, respectively.

5. Security Analysis

In this section, we demonstrate that the proposed construction is secure through the following analyses. We first explain that our design satisfies the following security requirements mentioned in Section 2.3.

Requirement 1: indeed, this requirement is satisfied. All kinds of messages (sensing reports and control messages) will be encrypted with the generated keys. An adversary cannot realize any information without keys.

Requirement 2: if an adversary compromises an L-Sensor which is denoted as L_i^j, he can obtain $K_{\text{master}} = S \times r \bmod p$, $\mathrm{SI}_{L_i^j,\mathrm{BS}} = S \times e_j^{-1} \bmod p$, CK_j, and several key encryption keys KEK. However, he is not capable of calculating cluster key and secure information belonging to other clusters. Moreover, the secrets belonging to other clusters cannot be derived even if he compromises two L-Sensors belonging to different clusters.

Requirement 3: actually, the purpose of our key updating scheme can achieve Requirement 3.

Here we analyze the proposed construction in the following aspects.

5.1. The Influence of Compromised L-Sensors. Actually, an adversary can obtain all information which is stored in an L-Sensor after compromising it. Fortunately, the adversary cannot calculate the information (cluster key and secure information) of other clusters in our design; as a result, the proposed construction ensures that the compromise of L-Sensors does not cause the compromise of the entire network.

To prevent the adversary from decrypting future traffic, the compromised individual key and the pairwise keys are revoked automatically. Besides, BS updates the affected cluster key and the master key. In our design, the adversary cannot obtain this updated information. Moreover, as an example showed in Section 4, only the affected cluster needs to update the secure information before updating the master key. Obviously, the influence of compromising can be confined to be affected cluster.

Besides, if an L-Sensor moves to another cluster, both clusters (original and targeting clusters) would update the cluster keys. It can prevent the L-Sensor from decrypting the traffic of both clusters after it leaves or before it joins. In fact, it can also effectually reduce the influence of compromising. An adversary may eavesdrop and record all packets before compromising L-Sensors. Let us consider the following situations if an L-Sensor L_1^0 moves from cluster 0 to cluster 1. Note that CK_0 and CK_1 are the cluster keys of cluster 0 and cluster 1, respectively.

(1) CK_0 and CK_1 are updated: CK_0 and CK_1 are updated to CK'_0 and CK'_1, separately. If L_1^0 which already has moved to cluster 1 is compromised, an adversary would obtain CK'_1. Hence, he cannot decrypt the messages encrypted with CK_1. Similarly, if the adversary compromises an L-Sensor within cluster 0 after L_1^0 leaves, he can only decrypt the messages encrypted with CK'_0.

(2) CK_0 and CK_1 are not updated: L_1^0 receives CK_1 after arriving to cluster 1. If L_1^0 is compromised, an adversary can decrypt the messages encrypted with CK_1. More specifically, after compromising L_0^1, he can retrieve the messages which are sent before L_1^0 joins. Similarly, the same condition happened if he compromises an L-Sensor within cluster 0.

Obviously, the influence of compromising is effectually reduced.

5.2. Confidentiality. An outsider cannot obtain the current cluster keys or master key because the cluster keys and the secure information $S \times e_n^{-1} \bmod p$ which is used for computing the master key are securely distributed to all legitimate L-Sensors. Although a leaving L-Sensor has the old secure information or the old cluster key, this L-Sensor cannot derive the new master key or the new cluster key. To prevent a leaving L-Sensor from obtaining the parameter S or e_n, we choose the length of the prime number p as 128-bit. It is large enough to ensure that deriving correct (x,y) pair from $x \times y \bmod p$ is infeasible. Although a leaving L-Sensor knows the messages $r \times W \bmod p$ and $S \times e_n^{-1} \bmod p$, it cannot compute the current master key $S \times r \bmod p$.

5.3. Integrity. To achieve message authentication and integrity, we can utilize hash algorithms, for example, SHA-1 or MD5, to compute message authentication code (MAC) for a message. For example, in the master key generation phase of the key generating scheme, BS computes the MAC of $r \times W \bmod p$ with K_{Master}, appends it to $r \times W \bmod p$, and then broadcasts it to all H-Sensors. Each H-Sensor H_j broadcasts $r \times e_j \bmod p$ and the received message ($r \times W \bmod p$ and its MAC) to its cluster members. Eventually, each L-Sensor uses the computed master key to recompute the MAC and compares it with the received MAC value. This implies that if an adversary masquerades as BS to send a message, he does not have the master key to compute the corresponding MAC value, and L-Sensors will reject this message.

6. Performance Evaluation

In this section, we show that the proposed construction is efficient in storage, communication, and computation. Besides, it is scalable because the additional overhead of increasing L-Sensors is confined to $\log_2(s)$.

6.1. The Storage Requirement. In the key generating scheme, each L-Sensor requires storing some keys and necessary information. For example, an L-Sensor L_i^j, would store $K_{L_i^j,\text{BS}}$, $K_{L_i^j,H_j}$, $\text{SI}_{L_i^j,\text{BS}}$, a cluster key CK_j, and a common master key K_{Master}. Also, several key encryption keys KEK must be stored to update the cluster key. The number of key encryption keys of L_i^j is about $\lceil \log_2(s) \rceil - 1$ where s is the number of L-Sensors in cluster j. The total storage of these secrets is 128-bit $\times\ (5 + (\lceil \log_2(s) \rceil - 1))$. Since usually there are at most one hundred L-Sensors within a cluster, total storage is about 176 bytes. Comparing with the current generation of sensor nodes (128 Kbytes in programmable flash memory in MICAz), the storage of the proposed scheme is much less. Besides, as the cluster size grows, the number of keys stored in an L-Sensor increases proportional to the number of L-Sensors within the cluster in an order of $\log_2(s)$. Note that each L-Sensor may also require to store the pairwise keys shared with its neighbors if necessary, but the total storage requirement is still reasonable.

6.2. The Communication Cost. Here we discuss the communication cost of the proposed construction. The communication cost is closely related to two factors. The first one is the message size. Obviously, the number of bits of every message is less than 128-bit. It is reasonable in a WSN. The second one is the transmission types. In fact, using broadcasting is more efficient than using unicasting in a WSN (we will show it in the next section). The majority of transmission in the key generating scheme is using broadcasting. For example, in the master key generation phase, BS broadcasts $r \times W \bmod p$ to all H-Sensors. Similarly, H-Sensors also broadcast the calculating results to its cluster members. The use of unicasting in the proposed scheme is normally involved in the initial stage of some phases. For example, BS unicasts $E_{K_{H_j,\text{BS}}}(\text{SI}_{H_j,\text{BS}})$ to all H-Sensors at the initial stage of master key generation phase; thus, this transmission would be executed only once; even the master key must be updated. Similarly, transmitting $K_{L_i^j,H_j}$ is still executed only once before generating cluster keys and the master key.

In the key updating scheme, every L-Sensor in the affected clusters only receives one message to update the cluster key. In order to avoid using unicasting, we aim to reduce the number of kinds of messages transmitted on the air. As the example shown in Section 4.1, only three kinds of messages are transmitted on the air. Similarly, only two kinds of messages are transmitted in Section 4.1. In the view of updating the master key, it only requires two broadcasts. More specifically, BS broadcasts same information to all H-Sensors, then each H-Sensor broadcasts the calculating result to all cluster members. Only the L-Sensors within the affected cluster require updating the secure information; consequently, the impact can be confined locally. As a result, updating keys incurs less communication overhead and is beneficial for the limited energy of L-Sensors.

6.3. The Computation Cost. The proposed construction utilizes symmetric cryptosystem, such as AES, and modular multiplication. AES is practical and efficient based on previous experimental studies. Another operation, modular multiplication, is always considered as an inefficient operation where the modulus is large, for example, 1024-bit moduli. However, the length of the modulus we adopted is only 128 bits. To demonstrate high effect of the modular multiplication with a 128-bit modulus, we implement it on MICAz sensor nodes. Computation results are given in Section 7.2.

7. Experiments

In our experiments, we choose MICAz sensor nodes. MICAz is capable of ATmega128L microcontroller. The architecture is 8-bit with 8 MHz computation speed. Total programmable memory storage of MICAz sensor is 128 Kbytes. For communication interface, MICAz uses ZigBee (802.15.4) to communicate with other MICAz sensors. Figure 4(a) shows one MICAz sensor. Another device is the base station. Figure 4(b) shows one MICAz sensor plugged on a MIB510 hardware interface, which is the interface of the base station. The MIB510 broad is connected to the desktop computer.

7.1. Experiment Assumptions and Design. After deployment, routing paths will be constructed. Normally, a routing path is constructed as a tree structure. In order to simplify this experiment, we assume that all H-Sensors in the routing path have the same degree. Figure 5 illustrates a constructed routing path with degree $d = 2$ and height $h = 4$.

Some researches assume that BS is capable of transmitting data to all H-Sensors through one hop. However, this assumption is not reasonable. In our experiments, we assume that data transmitted to all H-Sensors require multihops. For example, in Figure 5, the message sent to H_6 by BS would pass through H_0 and H_2. Therefore, if BS desires to update the master key, the following two scenarios may happen using the example shown in Figure 5.

(1) Using unicasting: BS encrypts the new master key K with each cluster key $(\text{CK}_0, \text{CK}_1, \ldots, \text{CK}_{13})$,

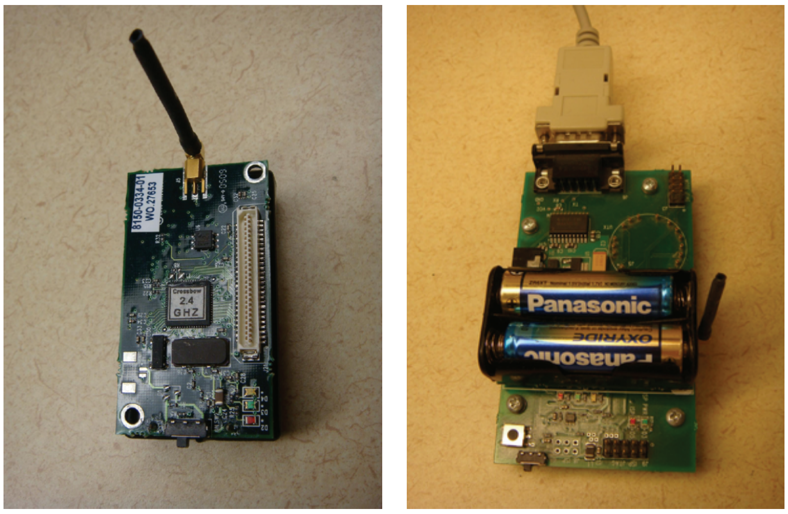

(a) MICAz sensor (b) MICAz with MIB510 broad

Figure 4: (a) Deployed sensor device and (b) base station device.

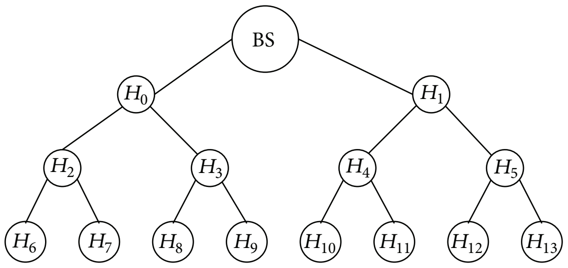


Figure 5: Example of a routing path where $d = 2$ and $h = 4$.

respectively. BS then sends these encrypted master keys to all H-Sensors. Therefore, H_0 receives seven messages which are $E_{\mathrm{CK}_0}(K)$, $E_{\mathrm{CK}_2}(K)$, $E_{\mathrm{CK}_3}(K)$, $E_{\mathrm{CK}_6}(K)$, $E_{\mathrm{CK}_7}(K)$, $E_{\mathrm{CK}_8}(K)$, and $E_{\mathrm{CK}_9}(K)$. H_0 obtains $E_{\mathrm{CK}_0}(K)$ and then transmits three messages ($E_{\mathrm{CK}_2}(K)$, $E_{\mathrm{CK}_6}(K)$, and $E_{\mathrm{CK}_7}(K)$) to H_2 and transmits the remainder three messages to H_3. Similarly, H_4 will receive three messages and then transmit two messages. Note that the size of the encrypted master key is 128 bits.

(2) Using broadcasting: the proposed master key updating method actually uses broadcasting. BS broadcasts a common message to all H-Sensors. Every H-Sensor receives only one message from the upper level H-Sensor and then broadcasts it to lower level H-Sensors. For example, H_2 receives a message from H_0 and then broadcasts it to H_6 and H_7. Note that the size of the broadcasted message depends on two situations. First, if BS desires to update the master key periodically, the size of the broadcasted message is 128 bits ($r' \times W \bmod p$). Second, BS updates the master key if L-Sensors are compromised. The size of broadcasted message is 256 bits $(((r' \times W \bmod p) \parallel (E_{K_{H_n,\mathrm{BS}}}(e_n^{-1} \times e'_n \bmod p))))$.

Table 1: Energy consumption of a MICAz sensor node.

	Transmit	Receive
Broadcast 128-bit	14.2 μJ	7.1 μJ
Broadcast 256-bit	16.56 μJ	8.28 μJ
Unicast 128-bit	14.1 μJ	7.05 μJ

Three experiments are performed in this paper.

(1) Experiment 1: we evaluate the energy consumption of an MICAz sensor node while transmitting or receiving messages with different sizes.

(2) Experiment 2: we calculate the communication overhead of the entire network. We consider the above two scenarios.

(3) Experiment 3: we evaluate the cost of AES and modulus multiplication.

In our experiments, we use MICAz sensor nodes to act as H-Sensors and calculate their energy consumption. If fact, energy consumed on powerful devices is the same as one on weak devices, such as MICAz. Besides, energy consumption measurement depends on the number of clock cycles spent [27]. Energy consumption for executing 2090 clock cycles on the ATmega128L microcontroller is equivalent to 7.4 μJ (Joule).

7.2. Experiment Results

Experiment 1. To measure energy consumed on transmitting and receiving data, the number of clock cycles is recorded when a data packet is received and sent. Broadcasting and unicasting executed 10 rounds for different length of data packets. The average results are showed in Table 1. Obviously, the values of broadcast 128-bit and unicast 128-bit are almost equal. Besides, the value of broadcasting 256-bit is larger

Table 2: Communication overheads, unit is mJ.

Type	Broadcast 128-bit	Broadcast 256-bit	Unicast 128-bit	Number of nodes	Maximum (uni)	Average (uni)
$d = 2, h = 2$	0.0064	0.0075	0.0852	2	0.0355	0.0426
$d = 2, h = 3$	0.0149	0.0174	0.3551	6	0.0923	0.0592
$d = 2, h = 4$	0.0320	0.0373	1.1221	14	0.2059	0.0802
$d = 2, h = 5$	0.0660	0.0770	3.1105	30	0.4332	0.1037
$d = 2, h = 6$	0.1342	0.1565	7.9964	62	0.8877	0.1290
$d = 2, h = 7$	0.2706	0.3155	19.5862	126	1.7967	0.1554
$d = 2, h = 8$	0.5433	0.6336	46.4019	254	3.6147	0.1827
$d = 2, h = 9$	1.0887	1.2696	107.3052	510	7.2507	0.2104
$d = 2, h = 10$	2.1795	2.5417	243.6559	1022	14.5228	0.2384
$d = 3, h = 2$	0.0852	0.0994	0.2983	3	0.0852	0.0994
$d = 3, h = 3$	0.2769	0.3229	1.7896	12	0.2983	0.1491
$d = 3, h = 4$	0.8520	0.9936	8.1810	39	0.9374	0.2098
$d = 3, h = 5$	2.5773	3.0056	33.1077	120	2.8548	0.2759
$d = 3, h = 6$	7.7532	9.0418	125.1444	363	8.6071	0.3448
$d = 3, h = 7$	23.2809	27.1501	453.0253	1092	25.8640	0.4149

but twice smaller than broadcasting 128-bit. It is because transmitting a 256-bit message still requires one package.

Experiment 2. According to the results in Experiment 1, total energy consumed for a WSN is simulated and evaluated. Table 2 lists the result of communication overhead of the above two scenarios. We consider several routing paths with different degree d and height h. For example, if a routing path is generated as a tree with degree $d = 2$ and height $h = 3$, the total energy consumption of all H-Sensors when broadcasting 128-bit/broadcasting 256-bit/unicasting 128-bit is 0.0149/0.0174/0.3551 mJ, respectively. The number of H-Sensors in this tree is 6. While performing unicasting 128-bit, the maximum energy consumption among all H-Sensors is 0.0923 mJ. Besides, the average energy consumption of all H-Sensors is 0.0592 mJ when performing unicasting 128-bit.

Since senor nodes may be deployed in a large scale environment, the number of clusters may be up to thousands. Thus, we consider several candidates with different d and h. Obviously, the communication overhead is closely related to the number of H-Sensors. If it grows to hundreds or thousands of nodes, it causes huge energy consumption.

In the view of average energy consumption of an H-Sensor, we can consider the following cases.

(1) Broadcast 128-bit: in this condition, every node receives one 128-bit message and broadcasts it. The total energy consumption of every node is 21.3 μJ (=14.2 + 7.1).

(2) Broadcast 256-bit: similarly, the total energy consumption of every node is 24.84 μJ (=16.56 + 8.28).

(3) Unicast 128-bit: the average energy consumption is listed in Table 2. Obviously, the values are quite larger.

Experiment 3. The goal of this experiment is to evaluate the costs of AES and modulus multiplication. Execution time and energy consumed by them are recorded. For AES, we choose an AES library based on TinyOS-2.x for comparison. We also implemented modulus multiplication by ourselves. These two operations were executed on MICAz physical sensors for 100 rounds. The average results are given in Table 3. In Table 3, one multiplication over 128-bit modulus is equivalent to 1.75 AES encryptions. As a result, modulus multiplication is feasible on physical sensors.

Table 3: Time and energy consumption for different operations.

	AES	Modular multiplication
Time (ms)	1.8	3.15
Clock cycle	1658.88	2903.04
Energy (μJ)	5.8735	10.2787

7.3. Discussion. Through these experiments, we can demonstrate that the proposed key updating scheme is efficient. This is because we utilize broadcasting instead of unicasting. Actually, we must take something into consideration. First, every H-Sensor in the routing path may not have the same degree. Second, BS may have more powerful transmission capability. Let us use Figure 5 as an example. The transmission range of BS may reach the second level H-Sensors. Therefore, the total energy consumption would not equal the result shown in Table 2. However, the overall energy consumption is still high if using unicasting.

8. Conclusion

In this paper, we proposed a complete hierarchical key management construction for heterogeneous cluster-based WSN which only utilizes simple operations. It considered several kinds of keys which are necessary for WSN. Besides, some kinds of keys may require updating; an efficient key updating scheme is also proposed. In order to provide better efficiency, the majority of transmission in our design is using broadcasting. In fact, using unicasting is inevitable in designing security mechanisms for WSN. Fortunately, the usage of unicasting in our design is normally involved at the initial stage of some phases. In the security analysis, we showed that the influence of compromising is effectually reduced and confined locally. We also showed that the proposed construction is efficient in storage, communication, and computation. Finally, we gave some experiments to further demonstrate two things. First, the operations we used are simple and practical. Second, using unicasting will cause uncontrollable overhead. In conclusion, the proposed construction is appropriate for heterogeneous cluster-based WSN.

Notation

BS :	The base station
H_j:	The H-Sensor j
L_i^j:	The L-Sensor i belongs to H_j
$E_K(M)$:	A message M encrypted with the key K
$K_{L_i^j,\mathrm{BS}}$:	The predeployed key shared between L_i^j and BS
$K_{H_j,\mathrm{BS}}$:	The predeployed key shared between H_j and BS
$K_{L_i^j,H_j}$:	The key shared between L_i^j and H_j
$\mathrm{SI}_{H_j,\mathrm{BS}}$:	The secure information shared between H_j and BS
$\mathrm{SI}_{L_i^j,\mathrm{BS}}$:	The secure information shared between L_i^j and BS
CK_j:	The cluster key for the cluster j
KEK_l^j:	The key encryption key l in the cluster j
K_{Master}:	The master key
$\oplus$:	Bitewise XOR operation
$\Vert$:	Concatenation.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The work was supported in part by the Project HIT.NSRIF.2014098 by Natural Scientific Research Innovation Foundation in Harbin Institute of Technology, in part by Shenzhen Peacock Project, China, under Contract KQC201109020055A, and in part by Shenzhen Strategic Emerging Industries Program under Grant ZDSY20120613125016389.

References

[1] H.-C. Shih, J.-H. Ho, B.-Y. Liao, and J.-S. Pan, "Hierarchical gradient diffusion algorithm for wireless sensor networks," in *Recent Trends in Applied Artificial Intelligence*, pp. 480–489, Springer, 2013.

[2] M. Demirbas, A. Arora, V. Mittal, and V. Kulathumani, "A fault-local self-stabilizing clustering service for wireless ad hoc networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 9, pp. 912–922, 2006.

[3] S. Basagni, M. Mastrogiovanni, A. Panconesi, and C. Petrioli, "Localized protocols for ad hoc clustering and backbone formation: a performance comparison," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 4, pp. 292–306, 2006.

[4] S. R. Das, C. E. Perkins, and E. M. Royer, "Performance comparison of two on-demand routing protocols for ad hoc networks," in *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '00)*, vol. 1, pp. 3–12, March 2000.

[5] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[6] X. Du, Y. Xiao, S. Ci, M. Guizani, and H.-H. Chen, "A routing-driven key management scheme for heterogeneous sensor networks," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pp. 3407–3412, June 2007.

[7] A. Durresi, V. Bulusu, V. Paruchuri, M. Durresi, and R. Jain, "WSN09-4: key distribution in mobile heterogeneous sensor networks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '06)*, pp. 1–5, December 2006.

[8] S. Hussain, F. Kausar, and A. Masood, "An efficient key distribution scheme for heterogeneous sensor networks," in *Proceedings of the International Wireless Communications and Mobile Computing Conference (IWCMC '07)*, pp. 388–392, August 2007.

[9] P. Traynor, H. Choi, G. Cao, S. Zhu, and T. La Porta, "Establishing pair-wise keys in heterogeneous sensor networks," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*, pp. 1–12, April 2006.

[10] P. Traynor, R. Kumar, H. Choi, G. Cao, S. Zhu, and T. La Porta, "Efficient hybrid security mechanisms for heterogeneous sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 663–677, 2007.

[11] E. Duarte-Melo and M. Liu, "Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '02)*, vol. 1, pp. 21–25, 2002.

[12] L. Girod, T. Stathopoulos, N. Ramanathan et al., "A system for simulation, emulation, and deployment of heterogeneous sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 201–213, November 2004.

[13] X. Du and F. Lin, "Maintaining differentiated coverage in heterogeneous sensor networks," *Eurasip Journal on Wireless Communications and Networking*, vol. 5, no. 4, pp. 565–572, 2005.

[14] L. Lazos and R. Poovendran, "Stochastic coverage in heterogeneous sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 3, pp. 325–358, 2006.

[15] Y. Lin, J. Zhang, H. S.-H. Chung, W. H. Ip, Y. Li, and Y.-H. Shi, "An ant colony optimization approach for maximizing the lifetime of heterogeneous wireless sensor networks," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 42, no. 3, pp. 408–420, 2012.

[16] C.-M. Chen, Y.-H. Lin, Y.-C. Lin, and H.-M. Sun, "RCDA: recoverable concealed data aggregation for data integrity in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 4, pp. 727–734, 2012.

[17] S. U. Khan, L. Lavagno, and C. Pastrone, "A key management scheme supporting node mobility in heterogeneous sensor networks," in *Proceedings of the 6th International Conference on Emerging Technologies (ICET '10)*, pp. 364–369, October 2010.

[18] Q. Shi, N. Zhang, M. Merabti, and K. Kifayat, "Resource-efficient authentic key establishment in heterogeneous wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 73, no. 2, pp. 235–249, 2013.

[19] H.-M. Sun, C.-M. Chen, and Y.-C. Hsiao, "An efficient countermeasure to the selective forwarding attack in wireless sensor networks," in *Proceedings of the IEEE Region 10 Conference (TENCON '07)*, pp. 1–4, November 2007.

[20] C. M. Chen, Y. H. Lin, Y. H. Chen, and H. M. Sun, "Sashimi: secure aggregation via successively hierarchical inspecting of message integrity on wsn," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 4, no. 1, pp. 57–72, 2013.

[21] H.-M. Sun, S.-P. Hsu, and C.-M. Chen, "Mobile Jamming attack and its countermeasure in wireless sensor networks," in *Proceedings of the 21st International Conference on Advanced Information Networking and ApplicationsWorkshops/Symposia (AINAW '07)*, pp. 457–462, May 2007.

[22] H.-M. Sun, Y.-H. Lin, Y.-C. Hsiao, and C.-M. Chen, "An efficient and verifiable concealed data aggregation scheme in wireless sensor networks," in *Proceedings of the International Conference on Embedded Software and Systems (ICESS '08)*, pp. 19–26, July 2008.

[23] L. Kong, C. M. Chen, H. C. Shih, C. W. Lin, B. Z. He, and J. S. Pan, "An energy-aware routing protocol using cat swarm optimization for wireless sensor networks," in *Advanced Technologies, Embedded and Multimedia for Human-Centric Computing*, pp. 311–318, Springer, 2014.

[24] 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, October 2003.

[25] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Transactions on Networking*, vol. 8, no. 1, pp. 16–30, 2000.

[26] 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 Workshops (ICPPW '03)*, pp. 397–406, 2003.

[27] 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, March 2005.