A Complete Hierarchical Key Management Scheme for Heterogeneous Wireless Sensor Networks

Chien-Ming Chen,1,2 Xinying Zheng,1 and Tsu-Yang Wu1,2

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

Copyright © 2014 Chien-Ming Chen et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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 a 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
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 $L^j_i$ 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 $L^3_0$, $L^3_1$, $L^3_2$, $L^3_3$, and $H_3$. $H_3$, 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 $H_0$ 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. 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 tamper-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 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
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 $K_{L_{i},BS}$ is preloaded to $L_{i}^{j}$ and $K_{H_{i},BS}$ 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, $K_{L_{i},H_{j}}$ which is assigned to $L_{i}^{j}$ is shared between $L_{i}$ and $H_{j}$. Since BS realizes the cluster information of each cluster, it can securely send the key $K_{L_{i},H_{j}}$ encrypted with $K_{H_{i},BS}$ or $K_{L_{i},BS}$ to every H-Sensor and L-Sensor. Hence, all L-Sensors would share the keys with their attached H-Sensors. Note that the key $K_{L_{i},H_{j}}$ 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 $R_{0}, R_{1}, \ldots, R_{\log_{2}(n_{j})-1}$ and $K_{0}, K_{1}, \ldots, K_{n_{j}/2-1}$ 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 $0 \leq X \leq \log_{2}(n_{j})-1$ and $0 \leq Y \leq [n_{j}/2]-1$. $R_{X}$ 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, $KEK_{0}^{0}$ ($=R_{2} \times K_{0}$ mod $p$), $KEK_{0}^{1}$ ($=R_{2} \times K_{0} \times K_{1}$ mod $p$), and $CK_{0}$ ($=R_{2} \times K_{0} \times K_{1} \times K_{2}$ mod $p$) are on the key path 0, $KEK_{0}^{1}$, $KEK_{0}^{2}$, and $CK_{0}$ are on the key path 1, and so on. $KEK_{0}^{0}, KEK_{0}^{1},$ and $CK_{0}$ belong to key path of $L_{0}^{0}$ and $L_{0}^{1}$. 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 $L_{0}^{j}$ to a leaf node and securely sends the cluster key $CK_{0}$ and key encryption keys on the key path to $L_{j}$ using the key $K_{L_{i},H_{j}}$.

#### 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 $SI_{H_{i},BS}$ and $SI_{L_{i},BS}$ 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:

$$\forall 0 \leq j < m, \quad BS \rightarrow H_{j} : E_{K_{H_{j},BS}} \left( SI_{H_{j},BS}, SI_{L_{j},BS} \right),$$

where $SI_{H_{j},BS} = e_{j} \times W^{-1} \mod p$ and $SI_{L_{j},BS} = S \times e_{j}^{-1} \mod p$. After receiving it, each H-Sensor, for example, $H_{j}$, broadcasts $E_{CK_{0}}(SI_{L_{j},BS})$ 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 $SI_{L_{j},BS}$. Note that L-Sensors attached to the same H-Sensor will have the same secure information $SI_{L_{j},BS}$. 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 $K_{Master} = S \times r \mod p$.

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

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

For all $0 \leq j \leq m-1$, $H_{j}$ calculates

$$E_{CK_{j}}(\left( (SI_{H_{j},BS} \times (r \times W \mod p)) \mod p \right) \oplus CK_{j}) = \left( ((e_{j} \times W^{-1}) \times (r \times W \mod p)) \mod p \right) \oplus CK_{j}$$

We use bitwise XOR operation $\oplus$ to guarantee that messages can be securely sent to legitimate L-Sensors.
Step 4. When \( L_i^j \) receives \((r \times e_j \mod p) \oplus CK_j \) from \( H_j \), it computes the master key \( K_{\text{Master}} \), where

\[
K_{\text{Master}} = SL_{L_i^j,BS} \times (r \times e_j \mod p) = (S \times e_j^{-1} \mod p) \times (r \times e_j \mod p)
\]

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

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 \( L_i^j \) desires to share a pairwise key with \( L_i^1 \), \( H_1 \) will generate this pairwise key and send \( E_{K_{L_i^j,H_1}}(\text{pairwise\_key}) \) and \( E_{K_{L_i^j,H_1}}(\text{pairwise\_key}) \) 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, \( L_i^n \) 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 \( L_0^0 \) leaves cluster 0, the keys \( CK_0 \), \( KEK_0 \), and \( KEK_0 \) 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) $H_0 \rightarrow \{L_0^0, \ldots, L_7^0\} : KEK_1^0 \oplus ((KEK_0^0) \times (K_3^{-1} \times K'_3) \mod p)$,

(ii) $H_0 \rightarrow \{L_0^1, L_7^1\} : KEK_2^0 \oplus ((KEK_0^0) \times (K_3^{-1} \times K'_3) \mod p)$,

(iii) $H_0 \rightarrow \{S_0^0\} : K_{U_{H_0}}^0 \oplus ((KEK_0^0) \times (K_3^{-1} \times K'_3) \mod p)$.

Step 2. $S_0^0, S_1^0, S_2^0$, and $S_3^0$ can obtain $K_3^{-1} \times K'_3 \mod p$ with KEK$^0_0$ and then compute the new cluster key CK$^0_0$ where

$$CK_0' = CK_0 \times (K_3^{-1} \times K'_3) \mod p \quad (4)$$

Step 3. $S_4^0$ and $S_5^0$ can obtain $K_3^{-1} \times K'_3 \mod p$ with KEK$^0_0$ and then compute the new keys CK$^0_0$ and KEK$^0_0'$ where

$$KEK_2' = KEK_2 \times (K_3^{-1} \times K'_3) \mod p \quad (5)$$

Step 4. $S_6^0$ can obtain $K_3^{-1} \times K'_3 \mod p$ with $K_{U_{H_0}}^0$ and then compute the new keys CK$^0_0$, KEK$^0_2$, and KEK$^0_0'$ where

$$KEK_6' = KEK_6 \times (K_3^{-1} \times K'_3) \mod p \quad (6)$$

Obviously, it only requires updating the value $K_3$ in this example. Since $L_0^0$ has no keys to obtain $K_3^{-1} \times K'_3 \mod p$, 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 $L_0^0$ joins this cluster and has received an individual key $K_{U_{H_0}}^0$. The procedure of updating the cluster key is described as follows.

Step 1. $H_0$ assigns $L_0^0$ 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$^0_0$, KEK$^0_2$, and KEK$^0_2'$, where

$$CK_0' = CK_0 \times (K_3^{-1} \times K'_3) \mod p \quad (7)$$

$$KEK_2' = KEK_2 \times (K_3^{-1} \times K'_3) \mod p \quad (8)$$

$$KEK_6' = KEK_6 \times (K_3^{-1} \times K'_3) \mod p \quad (9)$$

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 $L_0^0$:

(i) $H_0 \rightarrow \{L_0^0, \ldots, L_7^0\} : CK_0 \oplus (K_3^{-1} \times K'_3) \mod p$,

(ii) $H_0 \rightarrow L_0^0$:

$$\begin{align*}
(K_{U_{H_0}}^0 & \oplus (K_{U_{H_0}}^0 \times CK_0') \mod p)) \\
& \oplus (K_{U_{H_0}}^0 \times KEK_2') \mod p) \\
& \oplus (K_{U_{H_0}}^0 \times KEK_6') \mod p).
\end{align*}$$

Since $L_0^0$ does not have CK$^0_0$, it cannot obtain $K_3^{-1} \times K'_3 \mod p$ to compute the previous keys. On the contrary, all other L-Sensors in this cluster will obtain $K_3^{-1} \times K'_3 \mod p$.

Step 3. $L_0^0$ computes the new keys CK$^0_0$, KEK$^0_2$, and KEK$^0_2'$ with $K_{U_{H_0}}^0$.

Step 4. $L_0^0, L_1^0, L_2^0$, and $L_3^0$ can compute the new key CK$^0_0$ from (7) with the obtained $K_3^{-1} \times K'_3 \mod p$.

Step 5. $L_4^0$ and $L_5^0$ can compute the new keys CK$^0_2$ and KEK$^0_2'$ from (7) and (8) with the obtained $K_3^{-1} \times K'_3 \mod p$.

Step 6. $L_6^0$ can compute the new keys CK$^0_2$, KEK$^0_2$, and KEK$^0_2'$ from (7), (8), and (9) with the obtained $K_3^{-1} \times K'_3 \mod p$.

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 $K_3^{-1} \times K'_3 \mod p$ protected by KEK$^0_0$. The reason that we use the equation

$$KEK_0' \oplus ((KEK_0^0) \times (K_3^{-1} \times K'_3) \mod p) \mod p \quad (11)$$

rather than

$$(KEK_0^0) \times (K_3^{-1} \times K'_3) \mod p \quad (12)$$
is to protect KEK$^0$. More specifically, if we only use (12), $L^0$,
$L^2$, and $L^3$ can utilize the obtained $K_n^{-1} \cdot K^0$ mod $p$ to further
obtain KEK$^0$. In fact, KEK$^0$ is not revealed to $L^0$, $L^2$, and $L^3$;
therefore, an additional bitwise XOR operation @ is required.

Similarly, in Step 2 of Section 4.1.2, the reason we use (10) rather than
\[
\left( K_{L_i^0, H_0} \times CK_0 \mod p \right) \left( K_{L_i^0, H_0} \times KEK_2 \mod p \right)
\]
\[
\left( K_{L_i^0, H_0} \times KEK_0 \mod p \right)
\]  \hspace{1cm} (13)
is to protect $K_{L_i^0, H_0}$; otherwise, other cluster members can obtain
$K_{L_i^0, H_0}$ with CK$^0$, KEK$^0$, or KEK$^0$.  

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
broadcasts $r' \times W \mod p$ to all H-Sensors.

**Step 2.** While $H_j$ receives $r' \times W \mod p$, it calculates
\[
\left( S \times e_j^{-1} \mod p \right) \times (r' \times e_j \mod p)
\]  \hspace{1cm} (15)

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

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 $L_m^n$ 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 $K_{Master \_new} = S \times r' \mod p$. BS also
selects a random number $e_n \in Z_p^*$ to compute $e_n \times e_n' \mod p$ and broadcasts $(r' \times W \mod p) \parallel (E_{K_h, BS}(e_n^{-1} \times e_n' \mod p)))$
to all $H_S$.

**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
\[
\left( S \times e_j^{-1} \mod p \right) \parallel (E_{K_h, BS}(e_n^{-1} \times e_n' \mod p)))
\]  \hspace{1cm} (16)

Then, all their cluster members can obtain the new master
key $K_{Master \_new}$, where
\[
K_{Master \_new} = S \times r' \mod p
\]  \hspace{1cm} (17)

**Step 3.** (a) $H_n$ obtains $e_n^{-1} \times e_n' \mod p$ from $E_{K_h, BS}(e_n^{-1} \times e_n' \mod p)$ and further computes the new secure information $S_{H_n, BS}'$
\[
S_{H_n, BS}' = \left( S \times e_j^{-1} \mod p \right) \parallel (E_{K_h, BS}(e_n^{-1} \times e_n' \mod p)))
\]  \hspace{1cm} (18)

(b) $H_n$ obtains $(r' \times e_n' \mod p) \parallel CK_n$ from $r' \times W \mod p$ and new secure information $S_{H_n, BS}'$. Then $H_n$ broadcasts $E_{K_h, BS}(r' \times e_n' \mod p) \parallel (E_{K_h, BS}(e_n^{-1} \times e_n' \mod p))) \parallel CK_n \parallel CK_n'$. 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 $(r' \times e_n' \mod p) \parallel (E_{K_h, BS}(e_n^{-1} \times e_n' \mod p))) \parallel CK_n \parallel CK_n'$, and then compute $S_{H_n, BS}'$,
\[
S_{H_n, BS}' = \left( S \times e_j^{-1} \mod p \right) \parallel (E_{K_h, BS}(e_n^{-1} \times e_n' \mod p)))
\]  \hspace{1cm} (19)

(d) After updating $S_{L_m^n, BS}'$, all the other L-Sensors except $L_m^n$ within cluster $n$ can compute new master key $K_{Master \_new} = r' \times W \mod p$ where
\[
K_{Master \_new} = S \times r' \mod p
\]  \hspace{1cm} (20)

Only $H_n$ obtains $e_n^{-1} \times e_n' \mod p$ by decrypting $E_{K_h, BS}(e_n^{-1} \times e_n' \mod p)$. Besides, the compromised L-Sensor $L_m^n$ cannot
obtain \( r' \times e'_i \mod p \) and \( e^{-1}_n \times e'_i \mod p \) to compute the new secure information \( SI_{L_i,BS}^{r'} \), without \( CK_0' \); 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 encryts 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 \((r' \times W \mod p)\) 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, \( L_0^n \) and \( L_1^n \), 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 \( E_{K_0^n} (\text{new_pairwise}_\text{key}) \) and \( E_{K_1^n} (\text{new_pairwise}_\text{key}) \) to \( L_0^n \) and \( L_1^n \), 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 \( L_i \), he can obtain \( K_{\text{master}} = S \times r \mod p, SI_{L_i,BS} = S \times e_j^{-1} \mod p, CK_j, \) 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 \( L_0^1 \) 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 \( L_0^1 \) 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 \( L_0^1 \) leaves, he can only decrypt the messages encrypted with \( CK_0' \).

(2) \( CK_0 \) and \( CK_1 \) are not updated: \( L_0^1 \) receives \( CK_1 \) after arriving to cluster 1. If \( L_0^1 \) 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 \( L_0^1 \) 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 \( S \times e_n^{-1} \mod p \) 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 \( x \times y \mod p \) is infeasible. Although a leaving \( L \)-Sensor knows the messages \( r \times W \mod p \) and \( S \times e_n^{-1} \mod p \), it cannot compute the current master key \( S \times r \mod p \).
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, BS computes the MAC of \( r \times W \mod p \) with \( K_{\text{Master}} \), appends it to \( r \times W \mod p \), and then broadcasts it to all \( H \)-Sensors. Each \( H \)-Sensor \( H_j \) broadcasts \( r \times e_j \mod p \) and the received message \( (r \times W \mod p \text{ and its MAC}) \) 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 \( K_{L_i^j, \text{BS}} \), \( K_{L_i^j, H_j} \), \( SL_{L_i^j, \text{BS}} \), a cluster key \( CK_j \), and a common master key \( K_{\text{Master}} \). 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 \( 128\text{-bit} \times (5 + (\lceil \log_2(s) \rceil - 1)) \). 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 generation scheme is using broadcasting. For example, in the master key generation phase, BS broadcasts \( r \times W \mod p \) 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 \( E_{K_{H_j, \text{BS}}} (SI_{H_j, \text{BS}}) \) 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 board 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 \( H_j \) by BS would pass through \( H_0 \) and \( H_5 \). 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 (\( CK_0, CK_1, \ldots, CK_{13} \)).
respectively. BS then sends these encrypted master keys to all $H$-Sensors. Therefore, $H_0$ receives seven messages which are $E_{CK_0}(K)$, $E_{CK_1}(K)$, $E_{CK_2}(K)$, $E_{CK_3}(K)$, $E_{CK_4}(K)$, $E_{CK_5}(K)$, and $E_{CK_6}(K)$. $H_0$ obtains $E_{CK_0}(K)$ and then transmits three messages ($E_{CK_1}(K)$, $E_{CK_2}(K)$, and $E_{CK_3}(K)$) to $H_2$ and transmits the remainder three messages to $H_4$. 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 ($r \times W \mod p$). Second, BS updates the master key if $L$-Sensors are compromised. The size of broadcasted message is 256 bits ($((r \times W \mod p) || (E_{K_{fix},a} (e_1^{r} \times e_2^{r} \mod p)))$).

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 $\mu$J (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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Broadcast 128-bit</th>
<th>Broadcast 256-bit</th>
<th>Unicast 128-bit</th>
<th>Number of nodes</th>
<th>Maximum (uni)</th>
<th>Average (uni)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d = 2, h = 2$</td>
<td>0.0064</td>
<td>0.0075</td>
<td>0.0852</td>
<td>2</td>
<td>0.0355</td>
<td>0.0426</td>
</tr>
<tr>
<td>$d = 2, h = 3$</td>
<td>0.0149</td>
<td>0.0174</td>
<td>0.3551</td>
<td>6</td>
<td>0.0923</td>
<td>0.0592</td>
</tr>
<tr>
<td>$d = 2, h = 4$</td>
<td>0.0320</td>
<td>0.0373</td>
<td>1.1221</td>
<td>14</td>
<td>0.2059</td>
<td>0.0802</td>
</tr>
<tr>
<td>$d = 2, h = 5$</td>
<td>0.0660</td>
<td>0.0770</td>
<td>3.1105</td>
<td>30</td>
<td>0.4332</td>
<td>0.1037</td>
</tr>
<tr>
<td>$d = 2, h = 6$</td>
<td>0.1342</td>
<td>0.1565</td>
<td>7.9964</td>
<td>62</td>
<td>0.8877</td>
<td>0.1290</td>
</tr>
<tr>
<td>$d = 2, h = 7$</td>
<td>0.2706</td>
<td>0.3155</td>
<td>19.5862</td>
<td>126</td>
<td>1.7967</td>
<td>0.1554</td>
</tr>
<tr>
<td>$d = 2, h = 8$</td>
<td>0.5433</td>
<td>0.6336</td>
<td>46.4019</td>
<td>254</td>
<td>3.6147</td>
<td>0.1827</td>
</tr>
<tr>
<td>$d = 2, h = 9$</td>
<td>1.0887</td>
<td>1.2696</td>
<td>107.3052</td>
<td>510</td>
<td>7.2507</td>
<td>0.2104</td>
</tr>
<tr>
<td>$d = 2, h = 10$</td>
<td>2.1795</td>
<td>2.5417</td>
<td>243.6559</td>
<td>1022</td>
<td>14.5228</td>
<td>0.2384</td>
</tr>
<tr>
<td>$d = 3, h = 2$</td>
<td>0.0852</td>
<td>0.0994</td>
<td>0.2983</td>
<td>3</td>
<td>0.0852</td>
<td>0.0994</td>
</tr>
<tr>
<td>$d = 3, h = 3$</td>
<td>0.2769</td>
<td>0.3229</td>
<td>1.7896</td>
<td>12</td>
<td>0.2983</td>
<td>0.1491</td>
</tr>
<tr>
<td>$d = 3, h = 4$</td>
<td>0.8520</td>
<td>0.9936</td>
<td>8.1810</td>
<td>39</td>
<td>0.9374</td>
<td>0.2098</td>
</tr>
<tr>
<td>$d = 3, h = 5$</td>
<td>2.5773</td>
<td>3.0856</td>
<td>33.1077</td>
<td>120</td>
<td>2.8548</td>
<td>0.2759</td>
</tr>
<tr>
<td>$d = 3, h = 6$</td>
<td>7.7332</td>
<td>9.0418</td>
<td>125.1444</td>
<td>363</td>
<td>8.6071</td>
<td>0.3448</td>
</tr>
<tr>
<td>$d = 3, h = 7$</td>
<td>23.2809</td>
<td>27.1501</td>
<td>453.0253</td>
<td>1092</td>
<td>25.8640</td>
<td>0.4149</td>
</tr>
</tbody>
</table>

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 sensor 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 $21.3 \mu J$ ($=14.2 + 7.1$).

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

(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.

<table>
<thead>
<tr>
<th>Operation</th>
<th>AES</th>
<th>Modular multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (ms)</td>
<td>1.8</td>
<td>3.15</td>
</tr>
<tr>
<td>Clock cycle</td>
<td>1658.88</td>
<td>2903.04</td>
</tr>
<tr>
<td>Energy (\mu J)</td>
<td>5.8735</td>
<td>10.2787</td>
</tr>
</tbody>
</table>

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 effectively 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 \( j \)

\( L_i \) : The \( L \)-Sensor \( i \) belongs to \( H_i \)

\( E_K(M) \) : A message \( M \) encrypted with the key \( K \)

\( K_{L_i,BS} \) : The predeployed key shared between \( L_i \) and BS

\( K_{H_i,BS} \) : The predeployed key shared between \( H_i \) and BS

\( K_{L_i,H_j} \) : The key shared between \( L_i \) and \( H_j \)

\( S_{H_i,BS} \) : The secure information shared between \( H_i \) and BS

\( S_{L_i,BS} \) : The secure information shared between \( L_i \) and BS

\( C_{K_j} \) : The cluster key for the cluster \( j \)

\( KEK_{K_j} \) : The key encryption key \( l \) in the cluster \( j \)

\( K_{Master} \) : The master key

\( \oplus \) : Bitwise XOR operation

\( \| \) : 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


