Research Article

Key Updating Methods for Combinatorial Design Based Key Management Schemes

Chonghuan Xu^{1,2,3} and Weinan Liu⁴

¹ College of Business Administration, Zhejiang Gongshang University, Hangzhou 310018, China

² Contemporary Business and Trade Research Center, Zhejiang Gongshang University, Hangzhou 310018, China

³ Contemporary Business and Collaborative Innovation Research Center, Zhejiang Gongshang University, Hangzhou 310018, China

⁴ College of Computer Science & Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China

Correspondence should be addressed to Chonghuan Xu; talentxch@gmail.com

Received 27 April 2014; Accepted 12 June 2014; Published 1 July 2014

Academic Editor: Gongfa Li

Copyright © 2014 C. Xu and W. Liu. 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.

Wireless sensor network (WSN) has become one of the most promising network technologies for many useful applications. However, for the lack of resources, it is different but important to ensure the security of the WSNs. Key management is a corner stone on which to build secure WSNs for it has a fundamental role in confidentiality, authentication, and so on. Combinatorial design theory has been used to generate good-designed key rings for each sensor node in WSNs. A large number of combinatorial design based key management schemes have been proposed but none of them have taken key updating into consideration. In this paper, we point out the essence of key updating for the unital design based key management scheme and propose two key updating methods; then, we conduct performance analysis on the two methods from three aspects; at last, we generalize the two methods to other combinatorial design based key management schemes and enhance the second method.

1. Introduction

Wireless sensor network (WSN) is a kind of large wireless network. Generally speaking, WSNs consist of resource-limited sensor nodes with a large number of functions, such as sensing, collecting, and processing, which leads to their ability to fulfill special missions [1]. WSNs have become one of the most promising network technologies, which can be widely used in different applications. Because of the sensitivity of these applications and many issues involved in WSNs, sophisticated security services are required [2]. Key management is a corner stone on which to build secure WSNs, because it is able to protect many services, such as confidentiality and authentication, which are required for secure communication in WSNs. However, the sensor nodes in WSNs are all resource-limited devices. Therefore, key management schemes like Diffie-Hellman based or public-key based schemes, which are widely used in other networks, are not very suitable for the direct application in WSNs [3]. The key management scheme, which uses symmetric

encryption technology, is still a hot research topic. Over the last decade, a lot of research works dealt with symmetric key predistribution issue for WSNs and many solutions have been proposed [4–6]. Nevertheless, in most existing solutions, the design of key rings (blocks of keys) strongly depends on the size of the WSNs, which results in either suffering from low scalability or degrading other performance metrics [7].

Combinatorial design theory is an important part of combinatorial mathematics. Combinatorial design theory deals with the existence, construction, and properties of systems of finite sets whose arrangements satisfy generalized concepts of balance and/or symmetry. Combinatorial design theory has been used in many different applications, such as cryptography and secret sharing in literatures [8–10], network design in literatures [11, 12], and key management for WSNs in literatures [7, 13–16]. Since Mitchell and Piper [13] firstly applied combinatorial designs to key distribution, a lot of other combinatorial design based key management schemes have been proposed.

Key updating is very important in key management because the more often a key is used, the less secure it will be. To the best of our knowledge, there exists no updating method for combinatorial design based key management scheme. Therefore, we propose two key updating methods to resolve the problem mentioned above.

The main contributions of this paper are as follows:

- (1) we point out the essence of key updating for combinatorial design based key management schemes which is a multigroup key distribution;
- (2) we propose two key updating methods for the unital design based key management scheme. The two key updating methods can be generalized to other kinds of combinatorial design based key management schemes. The first proposed method is distributed where each sensor node is devoted to key updating and the second one is traditional group key distribution;
- (3) we conduct the performance analysis on our proposed two methods and show their advantages and disadvantages;
- (4) we enrich the application of combinatorial design in key management for WSNs and also point out some future research directions.

The remainder of this paper is organized as follows: Section 2 presents the related works on combinatorial design based key management scheme for WSNs and shows the importance of key updating. We propose our two methods in Section 3. In Section 4, we conduct the performance analysis on the two methods from three aspects and make some comparison. In Section 5, we discuss our two proposed methods and improve them. Finally, we draw conclusions and point out our future research directions in Section 6.

2. Related Work

Mitchell and Piper [13] firstly used combinatorial designs into key distribution. They introduced a number of new concepts and showed how the theory of incidence structures with special properties, called key distribution patterns, may be applied to key management problems. The proposed method is shown to generalize earlier work in the area.

Çamtepe and Yener [14] proposed novel deterministic and hybrid schemes based on symmetric balanced incomplete block design (SBIBD) for key predistribution in WSNs. The SBIBD decides how many and what keys are assigned into each key-chain before the WSNs are deployed. After mapping from SBIBD to key predistribution, there are $m^2 + m + 1$ key rings, which contain $k = m + 1$ keys selected from a key pool S with the size $m^2 + m + 1$. And each pair of two key rings shares exactly one common key. SBIBD which produces better connectivity with smaller key-chain size can make key distribution schemes more efficient. The main strength of the proposed scheme is that each pair of two nodes shares exactly one common key which can obtain total secure connectivity. However, the SBIBD schemes do not have a good scalability

for large WSNs because they only generate $m^2 + m + 1$ key rings to obtain a key ring with $m + 1$ keys, while the former are determined by the properties of SBIBD.

Ruj and Roy [15] also used SBIBD to guarantee intraregion secure communications in grid group WSNs. In grid group WSNs, all nodes within a particular region can link to each other directly. While sensor nodes in different regions can make use of more special resource nodes to communicate with each other. Taking the Lee distance into account, they consider the resiliency not only in terms of fraction of links broken, but also in terms of the number of nodes and regions disconnected when some sensors are compromised. But the deterministic key predistribution schemes will lead to constant-time computation cost for shared key discovery and path key establishment.

Basic schemes like [17] have perfect network resilience but their network scalability is only $O(k)$ where k is the key ring size. The SBIBD [15] and the trade [16] based ones obtain a network scalability of $O(k^2)$. The solutions proposed in [13] elevate the network scalability up to $O(k^4)$ without losing secure connectivity coverage and overall performances. The researchers apply the unital design into predistribution. They propose two schemes, a basic one and an enhanced one, which achieve a good tradeoff between scalability and connectivity.

Combinatorial design based key management schemes obtain a lot of attentions. However, to the best of our knowledge, none of the related works have taken key updating into consideration. Without key updating, the key management schemes will be less secure as time passes. In this paper, we point out the essence of key updating for combinatorial design based key management scheme and then put up our two different kinds of key updating methods.

3. The Proposed Schemes

In this section, first we briefly introduce the unital design based key management scheme [7]. The brief introduction consists of unital design and its mapping to key predistribution. Then, we point out the essence of the key updating for the unital design based key management scheme. At last, we propose two key updating methods.

3.1. Unital Based Key Management Scheme

3.1.1. Background: Unital Design. Combinatorial design theory is a part of combinatorial mathematics dealing with the existence, construction, and properties of the systems which consist of finite sets whose arrangements satisfy generalized concepts of balance and/or symmetry. A t -design (v, b, r, k, λ) is defined as follows: given a finite set X of v points (elements), we construct a family of b subsets from the finite set X , called blocks, such that each block has a certain number of points, which is k , each point is contained in r blocks, and t points are contained together in exactly λ blocks [13]. A unital design is a Steiner 2-design, where $v = m^3 + 1$, $b = m^2(m^3 + 1)/(m + 1) = m^2(m^2 - m + 1)$, $r = m^2$, $k = m + 1$,

TABLE 1: An example of a 2-(9,3,1) unital design.

	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}
E_1	1					1	1				1	
E_2						2		2	2	2		
E_3			3	3			3	3				
E_4	4			4	4				4			
E_5					5		5			5		5
E_6			6						6		6	6
E_7		7	7		7	7						
E_8	8	8						8				8
E_9		9		9						9	9	

TABLE 2: The mapping from unital to key predistribution.

Unital design	Key predistribution
X : element set	S : key pool
Size of the object set X : $\nu = m^3 + 1$	The number of keys $ S = m^3 + 1$
Blocks	Key rings
Size of block X :	The number of keys of each ring has X : $l = m^3 + 1$
Number of generated blocks	Number of generated key rings (supported nodes)
$b = m^2(m^2 - m + 1)$	$n = m^2(m^2 - m + 1)$
Each point belongs to exactly m^2 blocks	Each key appears in exactly m^2 key rings

and $\lambda = 1$. We call the unital design 2-design $(m^3 + 1, m^2(m^2 - m + 1), m^2, m + 1, 1)$, or $(m^3 + 1, m + 1, 1)$ in short form.

Here is an example of a 2-(9,3,1) unital design with $m = 2$ shown in Table 1.

We can see from Table 1 that there are 12 blocks, $B_1, \dots, B_{12}$, in the above example, and 9 elements $E_1, \dots, E_9$ are 1, 2, $\dots$, 9, respectively. In addition, we can find that every block has 3 elements; for example, B_1 has 3 elements, E_1 , E_4 , and E_8 ; each element is contained in 4 blocks; for example, E_1 is contained in B_1 , B_6 , B_7 , and B_{11} , and each pair of 2 elements together is contained in exactly 1 block; for example, E_1 and E_4 together are contained only in B_1 .

3.1.2. A Basic Mapping from Unital to Key Predistribution for WSNs. Each block which has $m + 1$ points can be mapped into a sensor node in WSNs while each point can be mapped to a key. Note that point can also be called element in combinatorial design. The following table, Table 2, shows the basic mapping from unital design to key predistribution for WSNs.

The key distribution center (KDC) generates the unital blocks (key rings) and loads them to the memory of sensor nodes before these sensor nodes are deployed. Each distinct key ring will be preloaded in only one node, along with the corresponding key identifiers. When two nodes want to communicate with each other, they will exchange their key identifiers. According to their key identifiers, they can decide the communication key or they need to establish a secure path for communication. In the basic unital design mapping, each pair of two nodes shares at most one common key. Therefore, if two nodes have a common key, after exchanging the key identifiers, they will be able to determine this common key,

or if two nodes do not have a common key, they need the help of other nodes to establish secure paths.

3.2. System Model for WSNs and Updating Model for Each Sensor. We can see from the generalized definition of the unital design (a 2-design $(m^3 + 1, m^2(m^2 - m + 1), m^2, m + 1, 1)$) that there are $m^3 + 1$ keys and $m^2(m^2 - m + 1)$ sensor nodes. In addition, the definition of the unital design shows that each key is contained in m^2 sensor nodes and each sensor node has $m + 1$ keys. We can regard all the sensor nodes which have the same key as a group; therefore, there will be $m^3 + 1$ group, and in each group there will be m^2 sensor nodes. Furthermore, each sensor node belongs to $m + 1$ groups because it has $m + 1$ keys.

Therefore, the system model can be described in the following two formulas:

$$\begin{aligned}
 G_i &\longrightarrow k_i : n_{i1}, n_{i2}, \dots, n_{im^2} \quad \text{where } i = m^3 + 1, \\
 n_i &: k_{i1}, k_{i2}, \dots, k_{i(m+1)} \quad \text{where } j = m^2 \times (m^2 - m + 1).
 \end{aligned} \tag{1}$$

The first formula shows that the system model consists of $m^3 + 1$ groups. Each group corresponds to a key and there are m^2 sensor nodes in each group which is marked by the key ID. Note that $G_i \cap G_j = 1$ because, in a unital design, each pair of 2 elements is contained together in exactly 1 block.

The second formula points out that the system model is made up of $m^2(m^2 - m + 1)$ sensor nodes and each sensor node n_i has $m + 1$ keys. The $m + 1$ keys can be defined as a key ring or a key block, B_i . Note that two nodes have one common

key or not because in one unital design, each pair of B_i and B_j has the relationship that $|B_i \cap B_j| = 0$ or 1 .

According to the above system model, we can obtain the key updating model as each sensor node updates all its $m + 1$ keys. For each of sensor nodes n_i , the updating model can be described as in Figure 1.

Figure 1 shows the key updating model for each sensor. We can see, from it, sensor node n_i has $m + 1$ keys $k_{i1}, k_{i2}, \dots, k_{i(m+1)}$. If it wants to update k_{i1} , it will need to communicate with the other m^3 sensor nodes in G_1 that share the same key k_{i1} with it, and by some methods, the new updated key k'_{i1} can be obtained by all the sensor nodes in the G_1 . As each sensor node n_i has $m + 1$ keys, it needs to communicate with other groups like what it does with G_1 to update all its other keys.

We can see from the updating model for each of sensor nodes that the essence of key updating for the unital design based key management scheme is multigroup key management.

In the following subsection, we propose two key updating methods for the unital design based key management scheme, the first one is distributed which means each sensor node in the group is devoted to updating the group key, while in the second one, there is a group manager in each group distributing the broadcast to update the group key.

3.3. The First Key Updating Method. In this method, each sensor node is predistributed with a device ID, that is, id_{n_i} , $m + 1$ keys, and $m + 1$ corresponding key ID. In addition, each sensor node also stores the corresponding ID of sensor nodes in each group which is made up of the corresponding key. Table 3 shows the detail of the content which each sensor node should store in the first method.

In this method, to update one key, each sensor node will send a random number which is encrypted by the group key to other sensor nodes in the same group. And then, it will receive a certain number of messages which also contain random numbers from other sensor nodes. The number of the messages is determined by the number of legal sensor nodes in the group. For the sensor n_i , it updates one of its keys, k_{i1} , as the following steps.

Step 1. Consider $n_i \rightarrow n_{11}, n_{12}, \dots, n_{1m^2}/n_i \in G_1$: $\text{id}_{k_{i1}} | \text{id}_{n_i} | k_{i1} (\text{RN}_{i1})$.

In this step, sensor node n_i broadcasts the message $\text{id}_{k_{i1}} | \text{id}_{n_i} | k_{i1} (\text{RN}_{i1})$ to other sensor nodes which have the same group key k_{i1} . Here, $\text{id}_{k_{i1}}$ is the id of the key which needs to be updated; id_{n_i} is the ID of the sensor node and $k_{i1} (\text{RN}_{i1})$ which is the decrypted RN_{i1} .

Step 2. Sensor n_i receives broadcast message from other sensor nodes. If the first part of the broadcast message, $\text{id}_{k_{i1}}$, is also contained in the content of sensor n_i , and the ID of that sensor node (the sender of the broadcast message), id_{n_i} , is in the same key group, then it begins to receive the whole broadcast message. In order to compute the new updating key, it needs to receive a certain number (the number of other legal sensor nodes in the same key group). Note that after one

FIGURE 1: The updating model for each sensor.

TABLE 3: The content which each sensor node stores.

Content
id_{n_i}
$k_{i1}, \text{id}_{k_{i1}}, \text{id}_{n_{11}, n_{12}, \dots, n_{1m^2}/n_i}$
$k_{i2}, \text{id}_{k_{i2}}, \text{id}_{n_{21}, n_{22}, \dots, n_{2m^2}/n_i}$
$\vdots$
$k_{i(m+1)}, \text{id}_{k_{i(m+1)}}, \text{id}_{n_{(m+1)1}, n_{(m+1)2}, \dots, n_{(m+1)m^2}/n_i}$

node has used up its energy, it will inform other nodes in the same key group to delete its id. Therefore, the number of the broadcast messages which a sensor node needs to receive for computing the group key will decrease when the time goes. Note that these messages contain other RN generated by other sensor nodes in the same key group.

Step 3. After it receives l messages for updating the key, it computes the new updated key as the following equation:

$$k'_{i1} = h(\text{RN}_1 | \text{RN}_2 | \dots | \text{RN}_l). \quad (2)$$

Here l ($l \leq m^3 + 1$) is the number of legal nodes. The order of each RN_i is decided by the ID of the sender which is set before deployment.

As each sensor node has $m + 1$ keys, it needs to do the above steps $m + 1$ times to update all its $m + 1$ keys. When a sensor node is about to use up its energy, it will send a revocation message to other sensor nodes to inform them to delete the corresponding information about it in the key groups. In addition, if a node is detected as a promised sensor node, the detector will inform other nodes to delete the corresponding information about the promised node. The detector will ask other sensor nodes to update the corresponding keys the promised node has.

3.4. The Second Key Updating Method. In the first method, all the sensor nodes are distributed and are all responsible for the generation of the new key, while in the second key updating method, only one sensor node which is defined as group manager (GM) is responsible for the generation of a new key.

In this key updating method, the GM stores the same content as the content each sensor node stores in the first method. Each member sensor node is predistributed with a device ID id_{n_i} , $m + 1$ keys, and $m + 1$ corresponding key ID. Note that in this scheme, each sensor node does not store

TABLE 4: The content which the member sensor nodes store.

Content
id_{n_i}
$k_{i1}, \text{id}_{k_{i1}}$
$k_{i2}, \text{id}_{k_{i2}}$
$\vdots$
$k_{i(m+1)}, \text{id}_{k_{i(m+1)}}$

the identifiers of other sensor nodes. Table 4 shows the detail of the content which the group member sensor nodes store.

We can see from Table 4 that this method is more efficient in storage overhead because each member sensor node stores less content than that of the first method. The second method is based on the group key distribution. Therefore, there are two kinds of sensor nodes in the system: the GM and the member sensor nodes. In this method, to update one key, the GM needs to generate the broadcast and each member sensor node can compute the group key after receiving the whole broadcast.

The GM generates and sends the broadcast as the following steps.

Step 1. The GM generates the authentication polynomial as the following equation:

$$a_j(x) = \prod_{i=1}^l (x - x_i); \quad (3)$$

l is the number of legal member sensor nodes (except the GM and revoked sensor nodes), and $l \leq m^3$.

Step 2. The GM generates the key updating polynomial as follows:

$$P_j(x) = \text{ku}_j + a_j(x); \quad (4)$$

ku_j is the key updating material.

Step 3. The GM generates the broadcast and sends it. The construction of the broadcast is as follows:

$$B_j(x) = \text{id}_k \cap P_j(x). \quad (5)$$

We can see from (5) that the broadcast consists of two parts. The first part is id_k , the ID of the key which needs to be updated. And the second part is the key updating polynomial $P_j(x)$ which is generated in the second step.

Each sensor node (except for the GM) receives the first part of the broadcast and decides whether it needs to receive the whole broadcast according to the ID of the key, id_k . If it has this key (also has this key ID), then it receives the whole broadcast and computes the corresponding key updating information according to the broadcast; otherwise, it waits for the next broadcast. The key updating information can be computed as the following equation:

$$\text{ku}'_j = P_j(x) = \text{ku}_j + a_j(x_i) = \text{ku}_j + 0 = \text{ku}_j. \quad (6)$$

TABLE 5: The storage overhead of the two methods.

	id	Key
The sensor node in the first method	$m^4 + m^3 + m + 2$	$m + 1$
The sensor node in the second method	$m + 2$	$m + 1$

Each legal member node will get the computation result of 0 when computing the authentication polynomial $a_j(x_i)$. Therefore, all the legal member nodes will get the same key update information ku_j from the above equation while other revoked sensor nodes will obtain value of $a_j(x_i)$. The different value $a_j(x_i)$ will lead to different key updating information ku_j . And then each legal member sensor node uses the key update information to update the group key as the following equation:

$$k'_{i1} = h(\text{ku}_j \mid k_{i1}). \quad (7)$$

Note that all the legal member nodes can compute the same updated key because these legal member nodes have the same group key k_{i1} and use the same key updating information ku_j to update the group key.

4. Performance Analysis

In this section, we conduct the performance analysis on the two different kinds of key updating methods. As the sensor nodes in WSNs are all resource-limited devices, we take the following three aspects into consideration: storage, computation, and communication overhead.

4.1. Storage Overhead. Table 5 shows the storage overhead of the sensor devices in the proposed two methods.

In the first method, each sensor node stores its own device ID, and $m + 1$ group information which contains one key, one key ID, and other m^3 IDs of other sensor nodes is in the same group. Therefore, the total ID number which each sensor node stores is $(m^3 + 1)(m + 1) + 1 = m^4 + m^3 + m + 2$ and the total number of keys is $m + 1$.

In the second method, the GM has the same storage overhead as the first method; however, each member sensor node only needs to store its device ID and $m + 1$ group information. The group information consists of one key and one key ID. Therefore, the total ID number which each member sensor node stores is $(m + 1) + 1 = m + 2$, and the total number of keys is also $m + 1$, the same as the first method.

We can see from Table 5 that the second method is more efficient in storage overhead than the first method. Therefore, when the sensor nodes in WSNs are more limited in memory, the second method would be preferable.

4.2. Computation Overhead. Table 6 shows the computation overhead of the sensor devices in the proposed two methods.

In the first method, each sensor node needs to generate a random number RN and do $m + 1$ hash to update its $m + 1$ keys.

TABLE 6: The computation operation of the two methods.

	Generate RN	Poly generation	Poly evaluation	Hash
The sensor node in the first method	1	0	0	$m + 1$
The GM in the second method	1	1	0	$m + 1$
The sensor node in the second method	0	0	$m + 1$	$m + 1$

TABLE 7: The communication overhead of the two methods.

	id	RN	Poly
The sensor node in the first method	$2(m^3 + 1)(m + 1)$	$(m^3 + 1)(m + 1)$	0
The sensor node in the second method	$m + 1$	0	$m + 1$

In the second method, the GM needs to generate a random number, that is, the updating information, and a l -degree polynomial while other sensor nodes need to do $m + 1$ polynomial evaluation operations to obtain $m + 1$ updating information. Both the GM and the member sensor nodes need to operate $m + 1$ hash to update its $m + 1$ keys.

As all the sensor nodes in both methods have $m + 1$ hash operations (when lengths of the content n and nm are small, the time required for hash operations, for example, md5 and sha1, is not obviously different compared with that of polynomial operations which need a large number of loops. For example, the time required for hashing 64-bit data by md5 in java is 1 ms while hashing 64000 bits is about 19 ms); we can compare them from the previous three items: Generate RN, Poly generation, and Poly evaluation. The first method is much more efficient in computation overhead because the Generate RN operation is much more efficient in computation than that of the Poly operation in the second method. Therefore, when the sensor nodes in WSNs are more limited in computation ability, the first method would be preferable.

4.3. Communication Overhead. Table 7 shows the communication overhead of the sensor devices in the proposed two methods.

Table 7 shows the communication overhead of the devices in two methods. In the first key updating method, in order to update one key, a sensor node needs to send a message which contains two IDs and one encrypted random number and will receive m^3 messages from other m^3 nodes in the same key group. The communication overhead for updating one key is $2(m^3 + 1)$ IDs and $m^3 + 1$ random numbers. Therefore, for each of sensor nodes to update all its $m + 1$ keys, the total communication overhead is $2(m^3 + 1)(m + 1)$ ID and $(m^3 + 1)(m + 1)$ random numbers.

In the second method, the GM needs to send (while the other sensor nodes need to receive) the broadcast which contains the ID of the key and a l -degree polynomial ($l \leq m^3 + 1$). Therefore, the total communication overhead of the GM and each member sensor node are both $m + 1$ keys and $m + 1$ l -degree polynomials. Here, we take into consideration the largest communication overhead of each sensor node; therefore, the number of legal sensor nodes or the degree of the authentication polynomial, l , is set as m^3 ($m^3 + 1$ member

sensor nodes except the GM itself). Because each sensor node needs to update all its $m + 1$ keys, the total communication overhead is $m + 1$ ID and $m + 1m^3$ -degree polynomials.

Here, we define all the IDs, RNs, and coefficients of the polynomial AS belonging to F_p . Therefore, the communication overhead of the two methods can be concluded in Table 8.

Table 8 shows that the second method is more efficient than the first method in communication overhead. Therefore, when the sensor nodes in WSNs are more limited in energy, the second method is preferred.

4.4. Comparison between the Two Methods. We can draw a conclusion from the above three subsections that the first method is better in computation while the second method is more efficient in storage and communication. In addition, the second method is more energy saving because the energy cost is linked to communication overhead (communication needs more energy than the calculation) [18]. However, the first method is more secure because in the first second the capture of one sensor node will not affect the whole WSNs with the help of other security technologies such as IDS (intrusion detect system), while in the second method, the capture of the GM will affect the whole WSNs. Note that the capture of the member sensor nodes will not affect the WSNs. In conclusion, as each proposed method has its own strong points and weak points, we should choose different key updating method for different applications.

5. Discussion

5.1. Generalization. When a combinatorial design is constructed, the number of elements and the number of blocks are both determined which will lead to the determination of the number of groups and sensor nodes. As our methods are designed for the case that there exist a certain number of groups which contain some sensor nodes and sensor nodes which have some keys, they can be generalized to other combinatorial design based key management schemes.

For example, the symmetric balanced incomplete block design (SBIBD) is a $(v, b, r, k, \text{and } \lambda)$ design, where $v = b = m^2 + m + 1$, $r = k = m + 1$, and $\lambda = 1$. There are $m^2 + m + 1$ sensor nodes and $m^2 + m + 1$ keys in the WSNs. In addition, each key is contained in $m + 1$ sensor nodes and each sensor

TABLE 8: The communication overhead of the two methods.

The sensor node in the first method	$3(m^3 + 1)(m + 1)\log^p$
The sensor node in the second method	$(m^3 + 1)(m + 1)\log^p$

node has $m + 1$ keys, which means, for a sensor node n_i , it needs to update $m + 1$ keys among $m + 1$ key groups. The sensor node n_i can use the proposed two methods to update its $m + 1$ keys.

5.2. The Improvement of the Second Method. The second method can be improved in two factors: less communication overhead and self-healing.

As we can see from predistribution rule, each sensor node has $m + 1$ keys. Note that the construction of the unital design makes it possible for $m^2 - m + 1$ blocks having all $m^3 + 1$ elements. For example, in the example of the 2-(9,3,1) unital in Table 1, there are $2^2 - 2 + 1 = 3$ blocks and $2^3 + 1 = 9$ keys, and B_1 , B_2 , and B_{10} have all the 9 different keys, and so do B_2 , B_7 , and B_9 . Taking advantage of this good feature, we set that a node manages $m+1$ groups, so the number of group managers and the corresponding broadcasts will drop to $m^2(m^2 - m + 1)/(m + 1)$. The first part KID of the broadcast contains all the key IDs which the GM has. Therefore, the broadcast can be described as in the following equation:

$$B_j(x) = \text{KID} \cup P_j(x). \quad (8)$$

The disadvantage is that the degree of the $P_j(x)$ will increase to $m^2(m^2 - m + 1) - m$ as the authentication polynomial contains all IDs of the legal sensor nodes the GM knows. To reduce the degree of the $P_j(x)$, we can replace the authentication polynomial with the revocation polynomial and masking polynomial which will lead to only t (the degree of the masking polynomial) revocation ability. The masking polynomial is used in [19], and the construction of $P_j(x)$ is as follows:

$$P_j(x) = r_j(x) \times \text{ku}_j + s(x, \text{sid}_j), \quad (9)$$

where $r_j(x)$ is the revocation polynomial, $s(x, \text{sid}_j)$ is the masking polynomial, and sid_j is the session ID.

The modification increases the storage overhead that each sensor node needs to store the masking polynomial and the session ID but can decrease the degree of the $P_j(x)$ to T which can be much less than $m^2(m^2 - m + 1) - m$.

Self-healing is a desirable feature for group key distribution scheme as the wireless environment and resource-limited features make the WSNs sometime an unreliable network. In unreliable networks, the member in the group may lose a message from the GM. The member can ask the GM to resend the broadcast. However, it will become a large communication cost for the GM when the resend requirement increases. To overcome the above problem, in 2002, group key distribution with the feature of self-healing is proposed, and soon self-healing group key distribution scheme becomes a hot research topic [20].

Our second method can be easily modified to a self-healing one with some more communication overhead.

FIGURE 2: The relationship between T and healing rate.

The construction of the broadcast can also refer to that in [19] which can be described as the following equation:

$$B_j = \begin{cases} \text{KID} \cup P_j(x) & j = 1, 2 \\ \text{KID} \cup P_{\max(j-T, 1, \dots, j)}(x), \dots, P_j(x) & j = 3, \dots, m. \end{cases} \quad (10)$$

Here, m is the number of all the sessions and T is the number of $P_j(x)$ which each broadcast owns to ensure the ability of self-healing the group key within T sessions. Note that the number of $P_j(x)$ in the second part of B_j increases from 1 to T . Figure 2 shows the relationship between T and healing rate in the case that there are 1001 sensor nodes (a GM controls 1000 member sensor nodes) in the WSNs where the loss probability is 10%.

As we can see from Figure 2, the broadcast which contains less $P_j(x)$ can still have a good healing rate which means that we can set T a small number for self-healing of the second method.

6. Conclusion

In this paper, we focus on how to update the key used in combinatorial design based key management schemes for WSNs. In order to better introduce combinatorial design, we give an example of unital design and its mapping to key predistribution for WSNs. Then, we propose two key updating methods for the unital design based key management scheme; one is distributed and the other is group based. The key updating methods can be generalized to other combinatorial design based key management schemes, and the group based key updating method can be modified to a self-healing version easily. We conduct the performance analysis on the two proposed methods from three aspects: storage, computation, and communication overhead. As the essence of updating keys for combinatorial design based key management schemes is the same as that of the unital design based key management scheme, our two key updating methods can be generalized into other combinatorial design based key management schemes. Referring to the construction of

$P_j(x)$ in [19], we can modify the second method to a less communication overhead version. In addition, the second method also can be modified to a self-healing one. The best contribution of this paper is that with the key updating feature, combinatorial design based key management scheme will be more secure and, thus, enriches the application of combinatorial design in key management for WSNs.

Regarding future work, we will research in the following two aspects:

- (1) how to reduce the communication overhead of the first method by the construction of the combinatorial design;
- (2) looking for or coming up with another combinatorial design holding better features for key management in WSNs.

Conflict of Interests

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

Acknowledgments

This research was supported by Ministry of Education, Humanities and Social Sciences Project (13YJCZH216), Natural Science Foundation of Zhejiang Province (LQ12G01007), Foundation of Department of Education of Zhejiang Province (Y201329545), Key Laboratory of Electronic Commerce and Logistics Information Technology of Zhejiang Province (2011E10005), Innovative Group of e-Business Technology of Zhejiang Province (2010R50041), and Postgraduate Technology Innovation Project of Zhejiang Gongshang University (1120XJ1513176).

References

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–105, 2002.
- [2] Y. Zhou, Y. Fang, and Y. Zhang, "Securing wireless sensor networks: a survey," *IEEE Communications Surveys and Tutorials*, vol. 10, no. 3, pp. 6–28, 2008.
- [3] L. Pang, L. Jiao, and Y. Wang, "Authentication and key negotiation protocol between two nodes of wireless sensor networks," *Chinese Journal of Sensors and Actuators*, vol. 21, no. 8, pp. 1422–1426, 2008.
- [4] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM '04)*, pp. 586–597, March 2004.
- [5] C. Castelluccia and A. Spognardi, "RoK: a robust key pre-distribution protocol for multi-phase wireless sensor networks," in *Proceedings of the 3rd International Conference on Security and Privacy in Communications Networks (SecureComm '07)*, pp. 351–360, 2007.
- [6] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 52–61, October 2003.
- [7] W. Bechkit, Y. Challal, A. Bouabdallah, and V. Tarokh, "A highly scalable key pre-distribution scheme for wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 2, pp. 948–959, 2013.
- [8] D. R. Stinson and S. A. Vanstone, "A combinatorial approach to threshold schemes," *SIAM Journal on Discrete Mathematics*, vol. 1, no. 2, pp. 230–236, 1988.
- [9] D. R. Stinson, "Combinatorial characterizations of authentication codes," *Designs, Codes and Cryptography*, vol. 2, no. 2, pp. 175–187, 1992.
- [10] D. R. Stinson and T. Van Trung, "Some new results on key distribution patterns and broadcast encryption," *Designs, Codes and Cryptography*, vol. 14, no. 3, pp. 261–279, 1998.
- [11] B. Yener, Y. Ofek, and M. Yung, "Combinatorial design of congestion-free networks," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 989–1000, 1997.
- [12] Y. Song, A. Wool, and B. Yener, "Combinatorial design of multi-ring networks with combined routing and flow control," *Computer Networks*, vol. 41, no. 2, pp. 247–267, 2003.
- [13] C. J. Mitchell and F. C. Piper, "Key storage in secure networks," *Discrete Applied Mathematics*, vol. 21, no. 3, pp. 215–228, 1988.
- [14] S. A. Çamtepe and B. Yener, "Combinatorial design of key distribution mechanisms for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 15, no. 2, pp. 346–358, 2007.
- [15] S. Ruj and B. Roy, "Key predistribution using combinatorial designs for grid-group deployment scheme in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 6, no. 1, article 4, 2009.
- [16] S. Ruj, A. Nayak, and I. Stojmenovic, "Fully secure pairwise and triple key distribution in wireless sensor networks using combinatorial designs," in *Proceedings of the IEEE INFOCOM*, pp. 326–330, Shanghai, China, April 2011.
- [17] T. Choi, H. B. Acharya, and M. G. Gouda, "The best keying protocol for sensor networks," in *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '11)*, pp. 1–6, June 2011.
- [18] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: a survey," *Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, 2009.
- [19] S. Han, B. Tian, Y. Zhang, and J. Hu, "An efficient self-healing key distribution scheme with constant-size personal keys for wireless sensor networks," in *Proceedings of the IEEE International Conference on Communications (ICC '10)*, pp. 1–5, Cape Town, South Africa, May 2010.
- [20] B. Tian, S. Han, S. Parvin, J. Hu, and S. Das, "Self-healing key distribution schemes for wireless networks: a survey," *The Computer Journal*, vol. 54, no. 4, pp. 549–569, 2011.