東海大學資訊工程學系研究所

碩士論文

指導教授：呂芳懌 博士

一個在無線感測網路中基於 SKMaS 的金鑰管理機制

A Novel Key Management Scheme based on SKMaS in wireless sensor networks

研究生：江豐慶

中 華 民 國 一 〇 四 年 六 月 二 十 九 日

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 江 豐 慶 所提之論文

一個在無線感測網路中基於 SKMaS 的金鑰管理機制

經本委員會審查，符合碩士學位論文標準。

學位考試委員會
召 集 人 [illegible] 簽章

委 員 陳金鈴

[illegible]

[illegible]

指 導 教 授 [illegible] 簽章

中華民國 104 年 6 月 29 日

中文摘要

為了在感測網路中提供一個安全的通訊環境，我們常需在與節點通訊前先認證節點。由於無線的特性，每當節點廣播一個訊息 M 時，附近的節點都會收到。為了保護這個訊息和通訊的環境，一些安全技術是必要的。所以我們在本研究中，提出了一個稱為 Novel Key Management Scheme (簡稱 NKMaS)的對稱加密安全機制，該系統中有一個 KDS (Ker Distribution Server) 負責管理金鑰，它在佈建節點前的初始化階段，將自己的獨立金鑰(放在$K_{1,1}$)，控制金鑰(放在$K_{0,0}$)，節點 i 與其他節點 j 通訊時用來加密之 key-cross i 和用來產生 keys 的 key-table i 送給節點 i； $2 \leq i, j \leq n, i \neq j$，$n$ 是整個網路中所有節點的數量(包括 KDS 本身, ID = 1)。用來和其他節點通訊的金鑰稱為 communication keys (CKs)，節點 i 還有一個自己的獨立金鑰(individual key)，是在 i 和 KDS 互傳訊息時使用。當節點 i 須和 j 通訊時，雙方會互相交換 ID，並在自己的 CKs 中找出相對應的 CK 後產生 DSK。*i*(*j*)藉由 DSK 來認證 *j*(*i*)。當節點 i 離開網路時，它的金鑰會留給後來頂替它 ID 的節點而由該節點繼續使用。但是，當節點加入網路時，沒有已使用過之 ID 可用時，KDS 會給它一個新的 ID (即$n+1$)，key-cross $n+1$，控制金鑰$K_{0,0}$， KDS 獨立金鑰$K_{1,1}$和 key-table $n+1$，然後廣播一個加入新節點之訊息內容包含讓各節點產生和節點 $n+1$ 通訊用之 CKs。系統的分析證明了本系統至少可以抵禦三種攻擊，包括 Eavesdropping attack、Forgery KDS attack 和 Forgery sensor node attack，也說明了我們的系統是安全的。

關鍵詞：Square-Key Matrix Management Scheme、 Key distribution server、新節點加入、Shared key、無線感測網路

ABSTRACT

In this paper we propose a symmetric cryptographic approach, named the Novel Key Management Scheme (NKMaS for short), in which a sensor node, called the Key Distribution Server (KDS for short), is responsible for the key management of the NKMaS. When the system starts up, the KDS establishes a key matrix *K* of $n \times n$, and sends its control key $K_{0,0}$, its individual key $K_{1,1}$, key-cross *i* and key-table *i* in which key-cross *i* as a part of *K* contains the communication keys (CKs for short) with which node i can communicate with node j, $2 \leq j \leq n, j \neq i$, and key-cross i 4×4 table used to generate CKs. With node IDs, two arbitrary valid sensor nodes, e.g., *i* and j, can individually identify the corresponding CKs, i.e., $k_{i,j}$ and $k_{j,i}$, in their own key-crosses with which to derive a dynamic shared key (DSK) for encrypting/decrypting messages transmitted between them. When i leaves the underlying network, the corresponding CKs and the individually keys currently utilized by *i* can be reused by a newly joining sensor, e.g., h. However, when h joins the network, if no such previously-used IDs are available, h will be given a new ID, i.e., $n+1$, key-cross h, $K_{0,0}$, $K_{1,1}$ and key-table h by the KDS. The KDS sends a newly-joining message which contains two seeds with which node i can generate $K_{i,n+1}$ and $K_{n+1,i}$, to $i, 2 \leq i \leq n,$ $i \neq h$. With $K_{i,n+1}$ and $K_{n+1,i}$, *i* can communicate with *h*. The lemmas and security analyzed in this paper prove that the proposed system can protect at least three common attacks, Eavesdropping attack、Forgery KDS attack and Forgery sensor node attack.

Keywords: Square-Key Matrix Management Scheme, Key distribution server, Newly joining node, Shared key, Wireless sensor network

List of Figures

List of Tables

.

List of Contents

1. Introduction

Wireless sensor networks (WSNs) are envisioned to be widely applied to commercial and military applications [1][2][3][4], such as target tracking [5], health-care [6][7], environment monitoring [8][9] and homeland security [10]. However, some WSN applications require certain security mechanisms [11] to verify the source of a message and protect the integrity of transmitted data from being maliciously modified. In order to securely authenticate a network entity and deliver messages, a secure communication environment [12][13] is required.

To build a secure WSN, Wuu et al. [14] proposed a Quorum-based Key Management Scheme. But this scheme has a problem in sensor node addition since the number of sensor nodes (or simply sensors or nodes in the following) must be odd. So each time at least two sensor nodes must be added. Furthermore, when two nodes are newly added to a sensor network, the shared keys (SKs) of some existing sensor nodes are changed. This may crash the normal operation of the whole system. We will show this later.

Generally, an asymmetric cryptographic technique [15] generates many large numbers to encrypt keys and delivered messages. But this is infeasible for WSNs, since sensor nodes are often powered by battery and provided with very limited processing capability [16][17][18][19][20]. Therefore, to achieve a high security level and support sensor node addition functionality, by which extra nodes can be easily added to a sensor network, in our previous study, a symmetric cryptographic approach, named the Square Key Matrix Management Scheme (SKMaS for short) was proposed. But when a new node, e.g., node *r*, joins the WSN, an issue concerning the message broadcasted by KDS needs to be addressed, i.e., the CKs for node *i* to encrypt the message exchange between

i and r need to be sent to i, for all is $2 \leq i \leq n$. The SKMaS puts all the CKs for all existing nodes in a message, named the newly-joining message. So when n is large, size of the message is also huge. This may waste delivery energy. For updating CKs for the new node r, each existing node in the network will receive the newly-joining message, meaning that the total length of the messages broadcasted by KDS is $n(3L + L_{ID})$, L_{ID} is the length of node ID , n is the total number of existing nodes and L is the size of a CK, e.g., $|K_{i,i}|$. When each node in the WSN receives the message, it will broadcast this message to its neighbors. Therefore, in the worst case, $n^2(3L + L_{ID})$ bytes will be sent, wasting a lot of message delivery energy.

To solve this problem, in this study, we propose a new scheme, named the Novel Key Management Scheme (NKMaS for short) for the key distribution. In the NKMaS , the newly-joining message is only $2L$ in length, rather than $n(3L + L_{ID})$ in the SKMaS, when a node newly joining the sensor network. In other words, the communication cost for a newly joining node is a constant. The main difference between the SKMaS and NKMaS is that a node in the NKMaS uses a key-table to generate CKs, which are derived from the key user inputs, for encrypting/decrypting messages sent between two nodes. In the pre-distribution phase, the KDS generates n key-tables, a key matrix K of $n \times n$ and three key seeds. The latter is used to generate CKs and individual keys for nodes. After that, the KDS allocates a key-table for node i, named key-table i, $2 \leq i \leq n$ and key-cross i which as a part of K consists of column i and row i intersected at $K_{i,i}$ where $K_{i,i}$ is the diagonal element indexed by i in K.

The rest of this paper is organized as follows. Section 2 introduces the related studies of this paper. Section 3 describes the proposed scheme. Security analysis and system simulation are presented and discussed in Section 5. Section 6 concludes this paper and addresses our future studies.

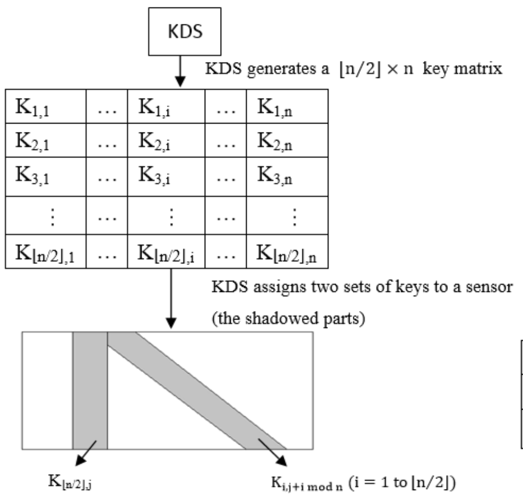
2. Related Works

Various key pre-distribution schemes used to establish secure channels for wireless sensors have been proposed in literature [14][21][22][23][24]. The key pre-distribution scheme proposed by Cheng et al. [21] introduced a $\sqrt{n} \times \sqrt{n}$ matrix as a key matrix, in which different parts of keys are assigned to different sensors where n is the total number of sensors in the system. The scheme has two phases: the key pre-distribution phase and pair-wise key setup phase. At first, the KDS randomly selects n keys from its key pool, in which more than 2^{20} distinct keys have been collected. The KDS uses the n keys to build an $m \times m$ key matrix K, where $|K| = \sqrt{n}$. The KDS assigns an element of this matrix, e.g., $K_{i,j}$, as a sensor's ID and the other entries in the ith row and jth column as the sensor's keys, to this sensor, implying that the matrix is indexed by the IDs of the involved sensors. It also means that this scheme provides the largest maximum supported network size since each element of the matrix represents one sensor node. When a sensor i would like to communicate with another sensor, e.g., j, it identifies the common keys indexed by i and j and uses them to encrypt those messages delivered between them.

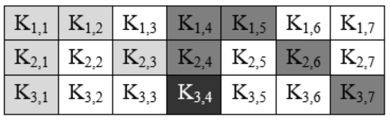
In 2008, Chien et al. [22] proposed a security scheme which based on Blom's approach [23] is a symmetric matrix based key pre-distribution system. Two arbitrary sensors in a sensing network can generate secret keys for the communication between them. Chien's system has three phases, including parameter pre-assigned phase, pair-wise key establishment phase and obsolete parameter erasure phase. In the parameter pre-assigned phase, a base station S generates the public matrix G and private symmetric matrix D, computes $A = (D \times G)^T$, stores the matrix A, a random number R_s and a shared key $K_{BS,i}$, and then assigns the key to sensor i. In the pair-wise key establishment phase, sensor *i* generates a random number R_i, computes $R_t = R_s \oplus R_i$ and sends i's ID and R_t to one of its neighbors, e.g., sensor j, $1 \leq j \leq n$ where n is the number of sensors in the concerned sensor network. After receiving this message, senor *j* computes R_t which is used as the pair-wise key. In the obsolete parameter erasure phase, sensors *i* and *j* will generate the pair-wise key by using these parameters. After that, the system erases the parameters to prevent them from being stolen, i.e., enhancing the system security.

In 2010, Wang [24] proposes a key establishment scheme which consists of three phases, pairwise key establishment, path key establishment and group key establishment. In the pairwise key establishment phase, the base station generates a lot of keys, called key pool, and then selects some of them as the keys to produce a symmetric matrix *K*. The base station then assigns these keys to nodes. On receiving these keys, a node stores them in memory. Before communication, two nodes, e.g., nodes *i* and *j* have to establish a pairwise key. First, node *i* sends its B_{yi} in a key matrix *B* to *j*. When *j* receives this message, it computes keys K_{ji} by multiplying the matrix and the message. Node *j* replies a message including B_{yj} and $F(K_{ji})$ where the latter is a hash function on K_{ji}. Once *i* receives the message, it calculates the hash value and check to see whether *j* is vaild or not. If *i* is not one of *j*'s neighbors, they need a secure communicate channel, and then enter the path key establishment phase, in which node *i* generates a random number, computes $Q_i = r_i P$ and sends Q_i and B_{yi} to *j* over a routing path. When *j* receives i's path key request, it generates a random number, computes Q_j and calculates K_{ji} by multiplying B_{yi} and B_{yj}. At last, *j* replies the message including B_{yj}, $F(K_{ji})$ and Q_j to *i*. When receiving B_{yj} and Q_j, node *i* computes K_{ij} and checks to see whether $F(K_{ji}) = F(K_{ij})$. If they are equal, *i* will compute Q_{ij} and send $F(K_{ij})$ and $F(Q_{ij})$ to *j*. Upon receiving this message, *j* checks to see whether it is valid or not. Then both *i* and *j* own the path keys Q_{ij} and Q_{ji} which are used to authenticate each other. If some nodes want to form a group, all nodes in the group are numbered as $M_1, M_2, \dots, M_n$ according to some criteria, then M_1 and M_2 mutually exchange their 2-party pairwise key. Without the loss of its geniality, assume $n = 6$. The procedure is as follows. In the first round, M_1 and M_2 forms a new group T_{I1} by executing pairwise key establishment procedure. Then M_3 and M_4 (M_5 and M_6) form a group T_{I2} (T_{I3}) by running the same establishment procedure. In the next round, each of M_2 and M_4 broadcasts its current tree encrypted by using respective blinded keys. Then T_{II1} as a new group contains T_{I1} and T_{I2}, and T_{I3} is recalled as T_{II2}. At last, T_{II1} and T_{II2} form a new group called T_{III} which consists of all nodes. Then a group key tree has been established.

As stated above, Wuu et al. [14] proposed a Quorum-based Key Management Scheme, in which the KDS as shown in **Fig. 1a** generates a $\lfloor n/2 \rfloor \times n$ key matrix *K* and establishes a quorum system based on *K*. Each sensor, e.g., *j*, has the entire column *j* of matrix *K* and $\lfloor n/2 \rfloor$ other elements. Each belongs to one of the $\lfloor n/2 \rfloor$ columns after column *j*, meaning that each sensor has $n-1$ elements, i.e., $K_{i,j}$ and $K_{i,j+i \bmod n}, 1 \le i \le \lfloor n/2 \rfloor,\ 1 \le \mathrm{j} \le n$. As shown in **Fig. 1b,** after the deployment of sensors, two arbitrary sensors, e.g., *A* and *B*, can individually identify the common keys assigned to them so that they can mutually authenticate and securely communicate with each other. In this scheme, node addition is feasible only when some existing IDs not currently in use are available. Also, when two nodes *A* and *B* newly join the WSN, as shown in **Fig. 2**, the common keys of some nodes will be changed. For example, originally the common key of nodes 1 and 5 was $K_{1,1}$. After sensors *A* and *B* join the network, the common keys of nodes 1 and 5 becomes $K_{4,5}$. Now, the system cannot work normally. Other schemes can be found in [16][17][18][25].


KDS
KDS generates a $\lfloor n/2 \rfloor \times n$ key matrix
KDS assigns two sets of keys to a sensor
(the shadowed parts)
$K_{\lfloor n/2 \rfloor,j}$
$K_{i,j+i \bmod n}$ (i = 1 to $\lfloor n/2 \rfloor$)

$K_{1,1}$	…	$K_{1,i}$	…	$K_{1,n}$
$K_{2,1}$	…	$K_{2,i}$	…	$K_{2,n}$
$K_{3,1}$	…	$K_{3,i}$	…	$K_{3,n}$
⋮	…	⋮	…	⋮
$K_{\lfloor n/2 \rfloor,1}$	…	$K_{\lfloor n/2 \rfloor,i}$	…	$K_{\lfloor n/2 \rfloor,n}$

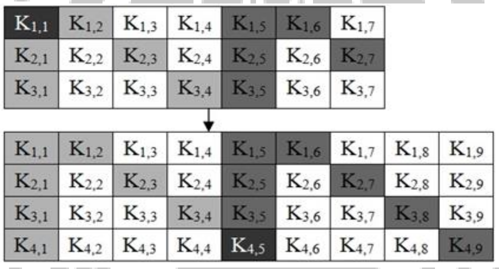


$K_{1,1}$	$K_{1,2}$	$K_{1,3}$	$K_{1,4}$	$K_{1,5}$	$K_{1,6}$	$K_{1,7}$
$K_{2,1}$	$K_{2,2}$	$K_{2,3}$	$K_{2,4}$	$K_{2,5}$	$K_{2,6}$	$K_{2,7}$
$K_{3,1}$	$K_{3,2}$	$K_{3,3}$	$K_{3,4}$	$K_{3,5}$	$K_{3,6}$	$K_{3,7}$

Fig. 1a KDS assigns each sensor node two sets of keys (the shadowed parts)

Fig. 1b Sensors A and B derive a common key

Fig.1 The KDS generates a key matrix and assigns common keys to a sensor



$K_{1,1}$	$K_{1,2}$	$K_{1,3}$	$K_{1,4}$	$K_{1,5}$	$K_{1,6}$	$K_{1,7}$
$K_{2,1}$	$K_{2,2}$	$K_{2,3}$	$K_{2,4}$	$K_{2,5}$	$K_{2,6}$	$K_{2,7}$
$K_{3,1}$	$K_{3,2}$	$K_{3,3}$	$K_{3,4}$	$K_{3,5}$	$K_{3,6}$	$K_{3,7}$

$K_{1,1}$	$K_{1,2}$	$K_{1,3}$	$K_{1,4}$	$K_{1,5}$	$K_{1,6}$	$K_{1,7}$	$K_{1,8}$	$K_{1,9}$
$K_{2,1}$	$K_{2,2}$	$K_{2,3}$	$K_{2,4}$	$K_{2,5}$	$K_{2,6}$	$K_{2,7}$	$K_{2,8}$	$K_{2,9}$
$K_{3,1}$	$K_{3,2}$	$K_{3,3}$	$K_{3,4}$	$K_{3,5}$	$K_{3,6}$	$K_{3,7}$	$K_{3,8}$	$K_{3,9}$
$K_{4,1}$	$K_{4,2}$	$K_{4,3}$	$K_{4,4}$	$K_{4,5}$	$K_{4,6}$	$K_{4,7}$	$K_{4,8}$	$K_{4,9}$

Fig. 2 New sensor nodes A (node 8) and B (node 9) join the network

3. The SKMaS

In SKMaS, we propose a symmetric cryptographic approach in which a sensor node, named the Key Distribution Server (KDS for short) which is the sensor with ID $= 1$, is responsible for the key management. When the system starts up, for each node *i* in the wireless system, the KDS delivers its own individual key $K_{1,1}$, a control key $K_{0,0}$, and a key-cross which consists of communication keys (CKs) with which i can communicate with node j, $2 \leq j \leq n, i \neq j$, to sensor node *i* for all *i*s, $2 \leq i \leq n$, where *n* is the number of nodes currently in the WSN. In key-cross i, $K_{i,i}$, called *i*'s individual key, is employed to encrypt messages delivered between *i* and the KDS.

When *i* would like to communicate with node *j*, the two nodes need to exchange their IDs with each other. With the IDs, i and j can individually identify the corresponding CKs , i.e., $K_{i,j}$ and $K_{j,i}$, from key-cross *i* and key-cross *j*, respectively. With $K_{i,j}$ and $K_{j,i}$, the two nodes can individually derive a dynamic shared key (DSK for short) for encrypting/decrypting messages transmitted between them. When *i* leaves the underlying network, the CKs and the individually keys currently used by *i* can be reused by a newly joining sensor, e.g., h. However, when *h* joins the network, if no such previously-used IDs are available, h will be given a new ID (e.g., $n+1$), key-cross $n+1$, $K_{1,1}$, and $K_{0,0}$ by the KDS. The two CKs, i.e., $K_{q,h}$ and $K_{h,q}$, by which an existing node q can communicate with h, are encrypted by using the individual key $K_{q,q}$ by the KDS so that only q rather than h can correctly decrypt the CKs, $2 \leq q \leq n, q \neq h$, based on the $n \times n$ key matrix *K* initially created by the KDS. Different parts of the key matrix are distributed to different sensors. Furthermore, due to the fast advancement of hardware technology, memory equipped in sensors is cheaper than before and the size grows rapidly in recent years. That means the memory size of a sensor no longer constitutes a problem. This further makes SKMaS feasible in practical applications.

The SKMaS consists of four working phases: the key pre-distribution, dynamic shared key establishing, key refreshment, and data transmission phases. In the key pre-distribution phase, the KDS generates a the key matrix K, in which the keys are random numbers. After that, the KDS assigns these keys to sensors during the deployment of sensor nodes. Before communicating with each other, the two sensors, e.g., nodes *i* and *j*, need send ID to each other to individually identify the CKs (recall communication keys) shared with each other, and then generates the DSK (recall Dynamic Shared Key) in the shared key establishing phase. When the sensor, e.g., *m*, newly joins the network, the KDS broadcasts the ID (i.e., *m*), and the CKs generated for *m*. Now the system enters its key refreshment phase, in which the receiving sensor accordingly updates its key information. In the data transmission phase, sensors transmit data to their neighbors, and authenticate received messages to see whether they are sent by valid sensors or not.

3.1. Key pre-distribution phase

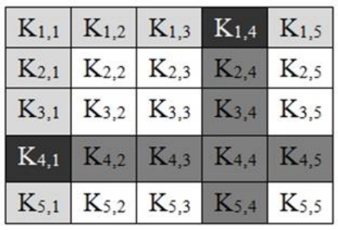
When the system starts up, for each node *i* in the wireless system, the KDS delivers $K_{1,1}$, $K_{0,0}$, and key-cross i (see Fig. 3, the shadowed areas) which consists of communication keys (CKs) with which *i* can communicate with node j, $2 \le j \le n, i \neq j$, to sensor node i, $2 \le i \le n$, where *n* is the number of nodes currently in the WSN. The steps of this phase are as follow.

Step 1: the KDS generates n^2 random numbers to establish the $n \times n$ key matrix

K.

Step 2: the KDS assigns an ID, e.g., i, $2 \leq i \leq n$, which is the index of $K_{i,i}$, and the CKs $K_{i,j}$ and $K_{j,i}$ in K to a sensor, i.e., node $i, 2 \leq i, j \leq n, i \neq j$.

Step 3: the KDS generates $K_{0,0}$ and $K_{1,1}$, and then sends key-cross *i* to node *i* in the system for all *i*s, $2 \leq i \leq n$.



$K_{1,1}$	$K_{1,2}$	$K_{1,3}$	$K_{1,4}$	$K_{1,5}$
$K_{2,1}$	$K_{2,2}$	$K_{2,3}$	$K_{2,4}$	$K_{2,5}$
$K_{3,1}$	$K_{3,2}$	$K_{3,3}$	$K_{3,4}$	$K_{3,5}$
$K_{4,1}$	$K_{4,2}$	$K_{4,3}$	$K_{4,4}$	$K_{4,5}$
$K_{5,1}$	$K_{5,2}$	$K_{5,3}$	$K_{5,4}$	$K_{5,5}$

Fig. 3 The KDS generates the $n \times n$ key matrix *K*, in which $[K_{1,i}, K_{2,i}, \cdots, K_{i,i}, \cdots, K_{n,i}]$ and $[K_{i,1}, K_{i,2}, \cdots, K_{i,i}, \cdots, K_{i,n}]$ together called key-cross *i*, are assigned to sensor *i*.

3.2. Dynamic shared key establishing phase

After the deployment of sensors, when sensor *i* would like to communicate with sensor *j*, it sends its own ID, i.e., *i*, to *j*. With the two IDs, *i* (*j*) can identify the CK, i.e., $K_{i,j}$ and $K_{i,j}$ contained in key-cross i (key-cross *j*), $1 \leq i, j \leq n, i \neq j$. Before authenticating node *j* (receiver), node *i* (sender) generates an authentication code (*Auth*) which contains the result of performing a two dimensional operation $\oplus$ and $+_2$ with $K_{i,j}$, $K_{j,i}$ and a random number *rand* as its parameters where

$$Auth = (K_{i,j} \oplus rand) +_2 K_{j,i} \qquad (1)$$

After that, *i* delivers *rand* and *Auth* to *j* and generates the *DSK* where

$$DSK = (K_{i,j} \oplus Auth) +_2 (K_{j,i} \oplus rand) \qquad (2)$$

On receiving *rand* and *Auth*, node *j* retrieves $K_{i,j}$ and $K_{i,i}$ from its own key-cross *j*, invokes Eq. (1) to calculate *Auth*, denoted by $Auth_c$, and checks to see whether the received *Auth*, denoted by $Auth_r$, is equal to $Auth_c$ or not. If yes, meaning that *i* is a valid one, *j* invokes Eq. (2) to generate the *DSK*.

3.3. Key refreshment phase

When sensor *i* leaves a WSN, the KDS broadcasts a message (named a leaving-node message) to the remaining sensors. The format of this message is shown in **Fig. 4** in which OP_code=0 indicates that this is a leaving-node message.

$$\boxed{\text{OP\_code} = 0|\text{node ID} = i|rand|\text{H}(K_{0,0} +_2 rand||K_{1,1} \oplus rand, rand)}$$

Fig. 4 The format of a leaving-node message issued by the KDS to announce the leaving of node *i*.

When a sensor, e.g., *d*, newly joins the network, it may face two situations: with or without an available used ID in the underlying network.

1. If the situation "with an available used ID" occurs, the KDS assigns an available ID, e.g., *i*, to d (i.e., $d = i$) and retrieves the corresponding CKs, i.e., key-cross *d*, from *K*, $2 \leq i \leq n$.
2. If the situation "without an available used ID" occurs, the KDS generates a new ID d (e.g., $d = n + 1$) and key-cross *d*, in which $K_{d,i}$ and $K_{i,d}$ are used by d to securely communicate with sensor *i* for all *i*s, $1 \leq i \leq n$.

Before deploying the new sensor, the system manager has to retrieve key-cross *d* and node ID, i.e., *d* from the KDS and store them in an internal file of the node. After that, the KDS broadcasts a message, named the newly-joining message, the format of which is shown in **Fig. 5**, in which the format of *k*-msg is illustrated in **Fig. 6**. In **Fig.**

5, OP_code = 1 indicates that this is a newly-joining-node message. Upon receiving this message, sensor *i* retrieves keys $K_{0,0}$ and $K_{1,1}$ from its internal file, calculates authentication code $\mathrm{H}(K_{0,0} \oplus \mathrm{rand}||K_{1,1} +_2 \mathrm{rand}, \mathrm{rand})_r$, and checks to see whether $\mathrm{H}(K_{0,0} \oplus rand||K_{1,1} +_2 rand, rand)_c$ is equal to $\mathrm{H}(K_{0,0} \oplus rand||K_{1,1} +_2 rand, rand)_r$ or not, where script *c* denotes calculation and script *r* stands for received. If not, *i* discards this message. Otherwise *i* sequentially searches the ID fields contained in the *k*-msg field of this message. In the *k*-msg, a sensor ID, e.g., *i*, is followed by, $K_{i,i} \oplus K_{d,i}$ and $(K_{0,0} +_2 K_{i,i}) \oplus K_{i,d}$, in which $K_{d,i}$ and $K_{i,d}$ are the CKs needed to be updated by sensor *i* and added to key-cross *i*.

$$\mathrm{OP\_code} = 1|rand|k\text{-msg}|\mathrm{H}(K_{0,0} \oplus rand||K_{1,1} +_2 rand, rand)$$

Fig. 5 The format of a newly-joining node message broadcast by the KDS to all sensors. In this message, the format of *k*-msg is shown in **Fig. 6**.

ID=2	$K_{2,2} \oplus K_{n+1,2}$	$(K_{0,0} +_2 K_{2,2}) \oplus K_{2,n+2}$	$K_{n+1,2} \oplus K_{2,n+1}$
ID=3	$K_{3,3} \oplus K_{n+1,3}$	$(K_{0,0} +_2 K_{3,3}) \oplus K_{3,n+1}$	$K_{n+1,3} \oplus K_{3,n+1}$
⋮			
ID=n	$K_{n,n} \oplus K_{n+1,n}$	$(K_{0,0} +_2 K_{n,n}) \oplus K_{n,n+1}$	$K_{n+1,n} \oplus K_{n,n+1}$

Fig. 6 The format of *k*-msg, included in a newly-joining-node message (see **Fig. 5**), contains sensor ID, e.g., *i*, and the communication Keys, $K_{d,i}$, and $K_{i,d}$ needed to be added to the key-cross *i* by sensor *i*, $2 \le i \le n$, where $d = n + 1$

For example, in **Fig. 6**, if ID = 2 and $d = n + 1$, the following two fields are $K_{2,2} \oplus K_{n+1,2}$ and $(K_{0,0} +_2 K_{2,2}) \oplus K_{2,n+1}$. Only the legal KDS has the right individual key $K_{2,2}$ and control key $K_{0,0}$ to encrypt the two fields, and only the valid sensor, i.e., sensor 2, which has the two keys, is able to decrypt the two fields. Moreover, after obtaining $K_{n+1,2}$ and $K_{2,n+1}$, sensor 2 compares the $(K_{n+1,2} \oplus K_{2,n+1})$ generated by itself with the fourth field from the head field to see whether they are equal or not. If yes, the message is authenticated.

In our scheme, the addition of d, no matter whether $d = n + 1$ or $2 \leq d \leq n$, does not change those CKs currently used by all existing sensors.

3.4. Data transmission phase

After completing the authentication between two sensors, the two sensors can communicate with each other by sending a data message, the format of which is shown in **Fig. 7.**

$$\boxed{\begin{array}{c}\text{OP\_code } = 2 \mid \text{SourceID} = i \mid \text{DestinationID} = j \mid rand \mid \\ (d\text{-msg} \oplus Auth) +_2 DSK \mid \mathrm{H}'(\, i \,||\, j \,||\, rand \,||\, d\text{-msg} \oplus Auth) +_2 DSK, DSK)\end{array}}$$

Fig. 7 The format of a data message, in which *d*-msg is the data that sensor *i* would like to send to sensor *j*.

Let *x* be $(d\text{-msg} \oplus Auth) +_2 DSK$. The d-msg can be obtained by decrypting *x* where

$$d\text{-msg} = \begin{cases} (x - DSK) \oplus Auth, & \text{if } x \geq DSK \\ (x + \overline{DSK} + 1) \oplus Auth, & \text{if } x < DSK \end{cases} \qquad (3)$$

3.5. Security of a message in SKMaS

The newly-joining and leaving messages issued by the KDS possess a high security mechanism and are discussed as follows.

First, the leaving-node message shown in **Fig. 4** is secure. Since hackers do not have the control key $K_{0,0}$ and KDS's individual key $K_{1,1}$, they cannot generate correct hash authentication code $\mathrm{H}(K_{0,0}+_2\mathrm{rand}||K_{1,1} \oplus \mathrm{rand}, \mathrm{rand})$.
Second, the newly-joining message illustrated in **Fig. 5** is more secure than the leaving message since this message contains the hash authentication code $\mathrm{H}(K_{0,0} \oplus \mathrm{rand}||K_{1,1}+_2\mathrm{rand}, \mathrm{rand})$ and the self-checking code, $K_{n+1,j} \oplus K_{j,n+1}$, $2 \leq j \leq n$.
Basically, only the KDS and the intended receiving sensor node j have the individual key $K_{j,j}$, $2 \leqq j \leqq n$, with which the KDS encrypts the newly-joining node d's CKs (see $K_{j,j} \oplus K_{n+1,j}$ and $(K_{0,0}+_2K_{j,j}) \oplus rand_{j,n+1}$ fields, i.e., the 2nd and 3rd columns, in **Fig. 6** where $d = n + 1$) and j decrypts the CKs so as to obtain the correct self-checking code $K_{n+1,j} \oplus K_{j,n+1}$. Even if one of the sensor nodes, e.g., m, was captured by hackers, with the parameter that m has, hackers cannot correctly generate other nodes' self-checking codes Lemma 1 and Lemma 2 will show this. We then dare to say that a newly-joining-node message is well protected.
Lemma 1:
Let $K_{0,0}$, $K_{j,j}$, $K_{j,n+1}$ and $K_{n+1,j}$ be all n bits long, $2 \leqq j \leqq n$. The probability p of correctly generating the self-checking code $K_{n+1,j} \oplus K_{j,n+1}$ shown in **Fig. 6** by hackers is $= \frac{1}{2^n}$.

Proof:

To correctly generates the self-checking code $K_{n+1,j} \oplus K_{j,n+1}$, hackers need to correctly decrypt the keys $K_{j,j} \oplus K_{n+1,j}$ and $(K_{0,0}+_2K_{j,j}) \oplus K_{j,n+1}$ to obtain the random keys $K_{j,n+1}$ and $K_{n+1,j}$. However, both $K_{0,0}$ and $K_{j,j}$ are unknown to hackers. The probability of correctly generating $K_{j,j} \oplus K_{n+1,j}$ and $(K_{0,0} +_2 K_{j,j}) \oplus K_{j,n+1}$ by hackers is $\frac{1}{2^n}$. In other words, when a sensor j receives an illegal newly-joining message broadcast by hackers, the probability p with which sensor node j correctly decrypt the corresponding portion of the message, i.e., $\mathrm{ID} = i \left| K_{j,j} \oplus K_{n+1,j} \right| (K_{0,0} +_2 K_{j,j}) \oplus K_{j,n+1} | K_{n+1,j} \oplus K_{j,n+1}$, to obtain the correct values of $K_{n+1,j}$ and $K_{j,n+1}$ is $\frac{1}{2^n}$.

Hence, the probability with which hackers correctly generate the self-checking code, is $\left(\frac{1}{2^n}\right) \times \left(\frac{1}{2^n}\right) \times 2^n = \frac{1}{2^n}$, where 2^n is the number of the possible value-combinations of each of $K_{n+1,j}$ and $K_{j,n+1}$.

Lemma 2:

Let $K_{0,0}$, $K_{j,j}$, $K_{j,n+1}$ and $K_{n+1,j}$ be n bits long, $2 \leqq j \leqq n$. The probability p of recovering the correct value of $K_{j,j}$ from the corresponding portion $\mathrm{ID} = j \left| K_{j,j} \oplus K_{n+1,j} \right| (K_{0,0} +_2 K_{j,j}) \oplus K_{j,n+1} | K_{n+1,j} \oplus K_{j,n+1}$ conveyed in the illegally intercepted newly-joining message on one trial is $p = \frac{1}{2^n}$.

Proof:

On receiving the corresponding portion of the newly-joining message, the hacker may guess a pair of keys (X, Y) such that $\mathrm{X} \oplus \mathrm{Y} = (K_{n+1,j} \oplus K_{j,n+1})_{\mathrm{r}}$ where subscript r denotes that the value of $\mathrm{X} \oplus \mathrm{Y}$ is retrieved from the message. Continuously, he/she may try to obtain the individual key $K_{j,j}$ by performing $(K_{j,j})_c = (K_{n+1,j} \oplus K_{j,n+1})_{\mathrm{r}} \oplus \mathrm{X}$ and $(K_{0,0} +_2 K_{j,j})_{\mathrm{c}} = (K_{n+1,j} \oplus K_{j,n+1})_{\mathrm{r}} \oplus \mathrm{Y}$, where subscript c denotes the value calculated by the hacker. However, without knowing $K_{j,j}$ and $K_{0,0}$, the hacker cannot verify whether the calculated values of $(K_{j,j})_{\mathrm{c}}$ and $(K_{0,0} +_2 K_{j,j})_{\mathrm{c}}$ are correct or not. That is, even though the hacker receives the newly-joining message and retrieves the portion $\mathrm{ID} = j \left| K_{j,j} \oplus K_{n+1,j} \right| (K_{0,0} +_2 K_{j,j}) \oplus K_{j,n+1} | K_{n+1,j} \oplus K_{j,n+1}$, there is no way for the hacker to make sure that the obtained $K_{j,j}$ is correct or not, except by a blind guess. Hence, the probability p of recovering the correct value of $K_{j,j}$ from the portion of an illegally intercepted newly-joining message on one trial is $p = \frac{1}{2^n}$.

Furthermore, the data message *m* delivered between sensor node *i* and sensor node *j*, as shown in **Fig. 7**, is protected by the random variables rand, *Auth* and *DSK*, which are themselves different in different times of communication. These contribute two security mechanisms to a data message.

1. The hash authentication code $\mathrm{H}'(\, i \,||\, j \,||\, rand \,||(d\text{-msg} \oplus Auth) +_2 DSK, DSK)$ contained in a data message possesses three security functions, including authentication [26], integrity [27], and non-repudiation [28].
2. d-msg shown in **Fig. 7** is well protected by *Auth*, *DSK*, and two-dimensional operation $\oplus$ and $+_2$. Only sensor node *i* and sensor node *j* have the communication keys $K_{i,j}$ and $K_{j,i}$, by which they can correctly encrypt/decrypt the *d*-msg.

From the above analyses, we can see that the newly-joining message shown in **Fig. 5** and the data message illustrated in **Fig. 7** are well encrypted by the SKMaS.

4. The Proposed Scheme

The NKMaS also consists of four working phases: the key pre-distribution, dynamic shared key establishing, key refreshment, and data transmission phases. In the key pre-distribution phase, the KDS generates a $n \times n$ key matrix *K*, in which the diagonal elements are individually keys and others are CKs (recall communication keys), and only two CKs are random numbers. After that, the KDS assigns these keys to sensors during the deployment of sensor nodes for later encryption. Before communicating with each other, each pair of sensors needs to identify the CKs shared with each other, and then generates the *DSK* (recall Dynamic Shared Key) in the shared key establishing phase. When the sensor, e.g., *m*, newly joins the network, the KDS broadcasts the ID (i.e., *m*), and a newly-joining message which carries the seeds used to produce $K_{m,i}$ and $K_{i,m}$, rather than carrying encrypted CKs like that in the SKMaS. Now the system enters its key refreshment phase, in which the receiving sensor, e.g., node *i*, accordingly updates its key-cross by adding $K_{i,n+1}$ and $K_{n+1,i}$ derived from key-table *i* by inputting the two seeds to their right positions in key-cross *i*. In the data transmission phase, sensors transmit messages to their neighbors, and authenticate received messages to see whether they are sent by valid sensors or not.

4.1. Key pre-distribution phase

In the NKMaS, when the system starts up, KDS builds the key matrix *K* with the following procedure.

Step 1: KDS generates $n-1$ key-tables, i.e., {key-table 2, key-table 3,…, key-table *i*,…, key-table *n*} in which key-table *i* is generated for existing node *i*, $2 \leq i \leq n$.

Step 2: For nodes 2 and 3, KDS produces two random numbers as the $K_{2,3}$ and $K_{3,2}$ which as shown in **Fig. 8** are a part of *K*.

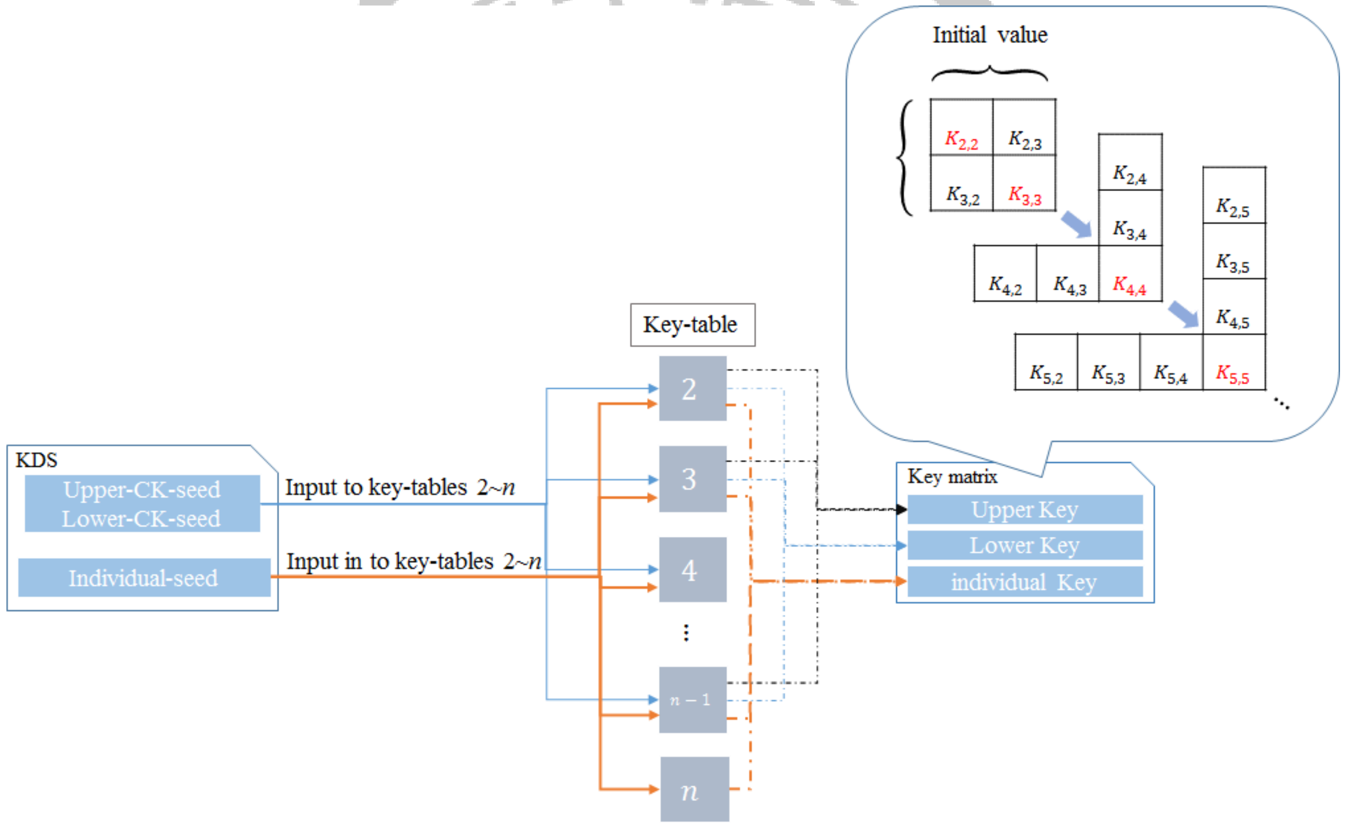


Fig. 8 The KDS input key-seeds to generate key matrix.

Step 3: KDS generates a random number as an individual-seed which is given to key-table *i* to produce $K_{i,i}$ (i.e., the individual key of node *i*) for all $is, 2 \leq i \leq n$.

Step 4: KDS generates 2 random numbers. The first called upper-CK-seed 4 is input to key-table h to generate an upper CK of K, i.e., $K_{h,4}$, and the second named lower-CK-seed 4 is also input to key-table h to produce a lower CK of K, i.e., $K_{4,h}$, for all hs, $2 \leq h \leq i-1$.

The algorithm is:

```
For ($i = 4$ to $n$)
{KDS generates an upper-CK-seed $i$ and a lower-CK-seed $i$;
For ($j = 2$ to $i-1$){
        inputs the upper-CK-seed $i$ to key-table $j$ to produce $K_{j,i}$;
        inputs the lower-CK-seed $i$ to key-table $j$ to produce $K_{i,j}$;}.
```

Step 5: KDS generates two random numbers as $K_{0,0}$ and $K_{1,1}$.

Step 6: KDS gives $K_{0,0}$, $K_{1,1}$, key-cross i and key-table i to node i for all is, $2 \leq i \leq n$. Node i keeps them in its internal file.

Step 7: Deploying the $n-1$ nodes, exclusing KDS itself, in the concerned sensing field.

Step 8: Put the KDS to the center of the field.

Note that in the NKMaS, a total of $5+2(n-3)$ random numbers is produced, including $K_{2,3}$, $K_{3,2}$, individual-seed, $K_{0,0}$, $K_{1,1}$ and 2 for node i (i.e., upper-CK-seed i and lower-CK-seed i) for all is, $4 \leq i \leq n$. But for node i, how to generate $K_{j,i}$ and $K_{i,j}$ for all js, $2 \leq j \leq i-1$, from upper-CK-seed i and lower-CK-seed i, $4 \leq i \leq n$. Here if $j > i$, $K_{j,i}$ will be a lower-CK. Otherwise $K_{j,i}$ is an upper-CK. Of course, if $K_{j,i}$ is a lower-CK, $K_{i,j}$ will be an upper-CK and vice versa. An example of CK generation is that, if upper-CK-seed i shown in **Fig. 9**, i.e., FB015…= 1111 1010 0000 0101…, is input to key-table j, the generated 3AFB8… is $K_{j,i}$ which is the upper-CK since in the algorithm listed in Step 4, *j* is always smaller than *i*. Given lower-CK-seed *i* to key-table *j*, we can obtain lower-CK, i.e., $K_{i,j}$ by using the similar method. Note that as mentioned above, the length of an upper-CK-seed and a lower-CK-seed follow the length of a given key (like $|K_{i,j}|$ in key matrix *K*).

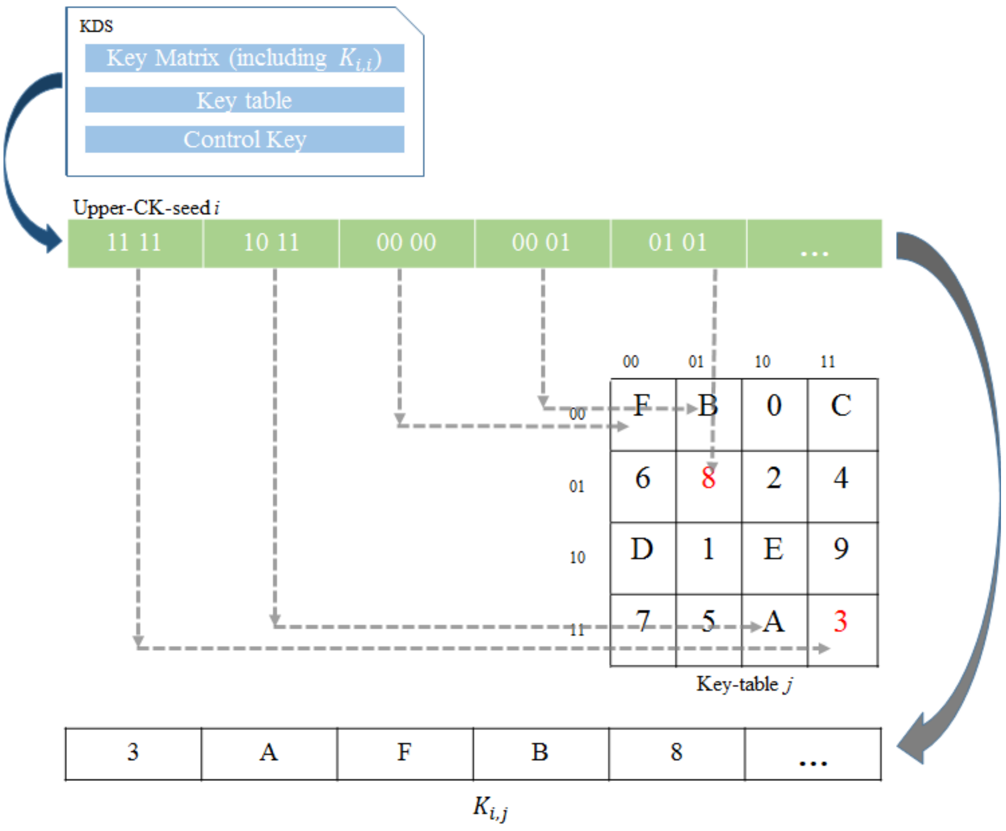


Fig. 9 The KDS inputs upper-CK-seed *i* to key-table *j* to generate $K_{j,i}$.

4.2. Dynamic shared key establishing phase

After the deployment of sensors, when sensor *i* would like to communicate with sensor *j*, it sends its own ID, i.e., *i*, to *j*. With the two IDs, *i* (*j*) can identify the CKs, i.e., $K_{i,j}$ and $K_{i,j}$ contained in key-cross *i* (key-cross *j*), $2 \leq i, j \leq n, i \neq j$. The rest of the content of this phase is described in section 3.2.

4.3. Key refreshment phase

When sensor i leaves a WSN, the KDS broadcasts a message, named a leaving-node message, to the remaining sensors. The format of this message is shown in **Fig. 4** in which OP_code=0.

On receiving the message, sensor j, $2 \leq j \leq n, j \neq i$, retrieves keys $K_{0,0}$ and $K_{1,1}$ from its internal file, calculates the authentication code $\mathrm{H}(K_{0,0} +_2 rand || K_{1,1} \oplus rand, rand)_r$, and checks to see whether $\mathrm{H}(K_{0,0} +_2 rand || K_{1,1} \oplus rand, rand)_c$ is equal to $\mathrm{H}(K_{0,0} +_2 rand || K_{1,1} \oplus rand, rand)_r$ or not, where script *c* denotes calculation and script *r* stand for received. If not, *j* discards this message. Otherwise, *j* no longer communicates with *i*. The node $\mathrm{ID} = i$ is now available and can be reused.

When a sensor, e.g., d, newly joins the network (see **Fig**. **10**), it may face two situations: with or without an available used ID in the underlying network.

1. If the situation "with an available used ID" occurs, the KDS assigns an available ID, e.g., i, to d (i.e., $d = i$), retrieves the corresponding CKs, i.e., key-cross d, from K, $2 \leq i \leq n$, and gives $K_{0,0}$, $K_{1,1}$, key-cross *i* (including *i*'s individual-key $K_{i,i}$), and key-table *i* to node *i*.

2. If the situation "without an available used ID" occurs, the KDS generates a new ID d (e.g., $d = n + 1$) and key-cross d, in which $K_{d,i}$ and $K_{i,d}$ are used by d to securely communicate with sensor *i* for all *i*s, $2 \leq i \leq n$.

In the NKMaS, the *k*-msg conveyed in the newly-joining message in the SKMaS (see **Figs. 5** and **6**) is substituted by upper-CK-seed $n + 1$ and lower-CK-seed $n + 1$. After the message is broadcast by KDS (see **Fig. 10**) and node *i* receives it, node *i* produce the two seeds and inputs them to key-table *i* to retrieve the corresponding entry values as $K_{i,n+1}$ and $K_{n+1,i}$. For example, assuming that the upper-CK-seed $n+1$ is (F, B, 0, 1, 5), the length of which is 20 $(= 4 \text{ bits} \times 5)$ bits. The corresponding binary numbers are (11 11, 10 11, 00 00, 00 01, 01 01). After that, node i retrieves the corresponding entries, i.e., 39FB8, as its $K_{i,n+1}$ from key-table *i* (see **Fig.** 11a). With the lower-CK-seed $n+1$, node *i* can generate $K_{n+1,i}$. When node *j* receives the newly-joining message, by inputting this upper-CK-seed $n+1$, i.e., (F, B, 0, 1, 5), to key-table j, it produces 89FB3 as its $K_{j,n+1}$. Note that when the length of a key of the concerned system is 1024 bit, the lengths of an upper-CK-seed and a lower-CK-seed are also 1024 bits, i.e., 256 digits, each ranging from 0 to F, are required.

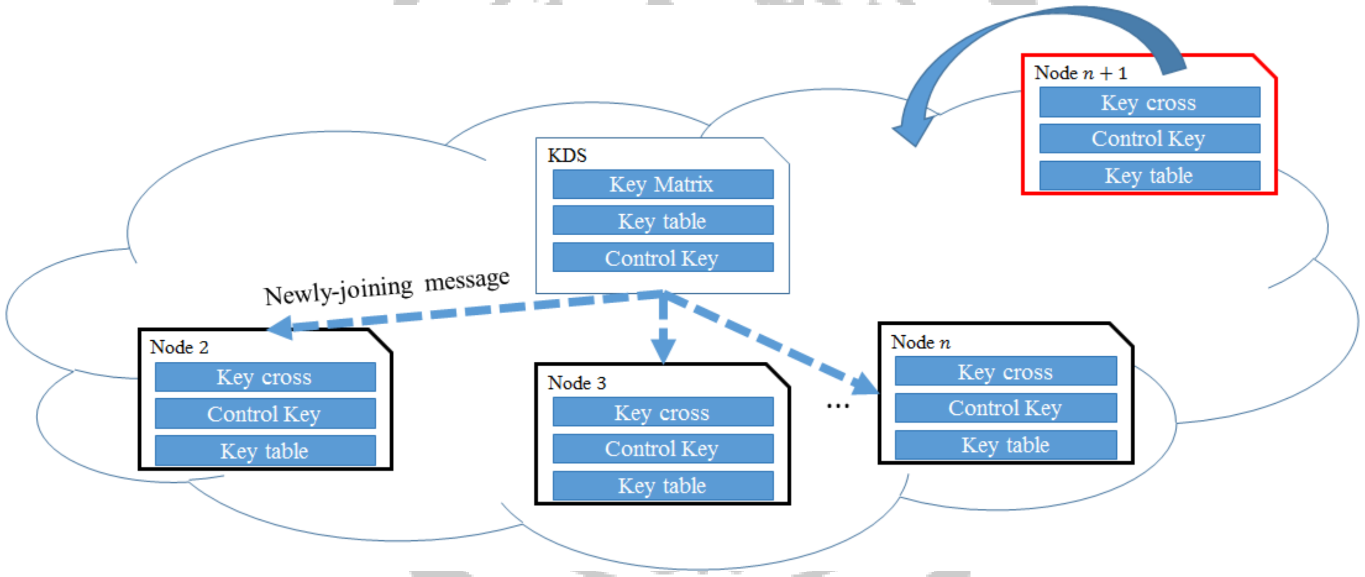


Fig. 10 A node newly joins the concerned network.

	00	01	10	11
00	F	B	0	C
01	6	8	2	4
10	D	1	E	9
11	7	5	A	3

(a) Key-table *i*

	00	01	10	11
00	F	B	0	C
01	6	3	2	4
10	D	1	E	9
11	7	5	A	8

(b) Key-table *j*

Fig. 11 The key-tables of nodes *i* and *j*

The steps for generating CKs used to encrypt and decrypt messages delivered between nodes *i* and *j* are as follows.

Step 1: Node i transforms the upper-CK-seed $n+1$ (lower-CK-seed $n+1$) it receives from the KDS into upper-binary string (lower-binary string), $2 \leq i \leq n$.

Step 2: Node *i* uses this upper-binary string (lower-binary string) as the indexes to retrieve the entry values from key-table *i* and concatenates the values in their retrieved order to form the CKs, i.e., $K_{2,n+1}$ ($K_{n+1,i}$).

Step 3: After that, node *i* stores these CKs, i.e., $K_{i,n+1}$ and $K_{n+1,i}$, used to encrypt and decrypt messages exchanged between itself and the newly joining node in its key-cross *i*.

4.4. Data transmission phase

In this phase, after completing the authentication between two sensors, the two sensors can communicate with each other by sending a data message, the format of which is shown in **Fig. 7,** in which OP_code = 2, and *d*-msg is the data that the sender, e.g., node *i*, would like to send to the receiver, i.e., node *j*. The detail is shown in section 3.4.

5. Security Analysis

The NKMaS has four features, including:

1. Verifying whether a transmitted message is a valid one or not by checking a hash authentication code with a dynamic key [29], e.g., *rand* (see **Figs. (4)**, **(6)** and **(7)**).
2. The OP_code as the head of a transmitted message explicitly indicates the function of this message to improve the efficiency of the following authentication and message processing.
3. The *DSK* employed in a data message effectively improves the security level of the message since for different communication sessions, *DSK*s varies due to invoking different *rand*s.

A two dimensional operation (i.e., $+_2$ and $\oplus$) invoked by the NKMaS to encrypt/decrypt data messages enhances the security level of the WSN.

In this section, we analyze the security of the transmitted messages and describe how the NKMaS effectively defend three common attacks, including the eavesdropping [30], forgery KDS, and forgery sensor node [31] attacks, and show that the NKMaS can effectively prevent a WSN from being attacked by them.

5.1. Eavesdropping attack

Due to the wireless nature, messages sent by sensor nodes and the KDS can be accessed by a sensor located within the communication area of the sender. As described above, illegal users cannot decrypt messages protected by *DSK* derived from *rand*, $K_{i,j}$ and $K_{j,i}$ (see **Eqs. (2)** and **(3)**). In this study, different messages are dynamically encrypted by different random keys, i.e., rands, thus having a higher security level than that protected by static keys [32]. In other words, messages delivered in the data transmission phase are secure. So the eavesdropping attack does not work.

5.2. Forgery KDS attack

A forgery KDS may send faked messages intending to cheat sensors that some sensor nodes leave or newly join the network. This kind of attack can be prevented by the unique individual key $K_{i,i}$, $2 \leq i \leq n$, which is only known to sensor *i* and the KDS, and is used to encrypt messages and authenticate the integrity of messages delivered between *i* and KDS. Therefore, only the legal KDS has the right $K_{i,i}$ and only *i* can correctly use it to decrypt the messages issued by the KDS, meaning that the NKMaS can effectively defend the forgery KDS attack.

5.3. Forgery sensor node attack

If a hacker, e.g., *b*', disguising itself as a valid sensor *b*, sends a data message to *c*, since *b*' does not have $K_{b,c}$ and $K_{c,b}$, the data message cannot pass the authentication performed by *c* (see **Fig. 7**). Thus *b*' is incapable of identifying the right *DSK* for further interacting with sensor *c*. Also, a faked node cannot decrypt messages issued by a valid one because it does not own the right *DSK*.

5.4. The collision problem

In the NKMaS, we found that when two nodes, e.g., node *i* and node *j*, receive a newly-joining message that contains upper-CK-seed $n+1$ and lower-CK-seed $n+1$, if the contents of the key-tables *i* and *j* are similar, and in the two key-tables, the values of the entries indexed by the two seeds are themselves the same, then the generated CKs will be identical, i.e., $K_{i,n+1} = K_{j,n+1}$ and $K_{n+1,i} = K_{n+1,j}$. The collision probability will be discussed as follows.

1. Given two key-tables, key-table *i* and key-table *j*, let an entry pair $P_{ij}(h)$ be the entry h in key-table i, denoted by $KT_i(h)$, and entry h in key-table *j*, denoted by $KT_j(h)$, $0 \leq h \leq \mathrm{F}$, i.e., $P_{ij}(h) = \{KT_i(h),\ KT_j(h)\}$. Let P_c be the table collision probability of key-tables i and j which is defined as when values of $(16-m)$ entries in key-table i are individually the same as those of the corresponding entries in key-table *j*, i.e., a total of $(16-m)$ entries in which $KT_i(h) = KT_j(h), 1 \leq h \leq F,$ e.g., $KT_i(1110_2) = KT_j(1110_2) = F,$ and $KT_i(1111_2) = KT_i(1111_2) = 2$. The values in the remaining *m* entries are all correspondingly different, i.e., $KT_i(h) \neq KT_j(h),\ 2 \leq m \leq 15$. The detail will be discussed below. Let $EP_{ij} = \{P_{ij}(1), P_{ij}(2), \dots, P_{ij}(F)\}$.
   (a) Basically, a key-table has 16 entries. and an entry is randomly given a non-repeated value from 0~F, meaning 0~F are randomly distributed to the table. So the number of possible key-tables that can be generated is $16!$.
   (b) The number of possible *m* entries selected from the 16 entries of a key-table is C_m^{16}.
   (c) Let $f_{ij}(m)$ be the number of possible cases in which key-table *i* has *m* entries $D_{ij}(m) = \{P'_{ij}(1), P'_{ij}(2), \dots, P'_{ij}(m)\}$, $P'_{ij}(h) \in EP,$ and

$KT_i(h) \neq KT_j(h)$ for all hs, $1 \leq h \leq m$. In **Fig. 11**, the value of the entry 0101_2 (1111_2), i.e., $KT_i(0101_2)$ $(KT_i(1111_2))$, is 8 (3) which is different from that of the entry 0101_2 (1111_2), i.e., $KT_j(0101_2)$ $(KT_j(1111_2))$, i.e., 3 (8) in key-table j. Thus $m = 2$. $f_{ij}(2)$ will be counted below. Let $T(m)$ be the case in which only the value of exact one entry, e.g., entry k, in both key-table i and key-table j are the same, i.e., $KT_i(k) = KT_j(k)$, and,

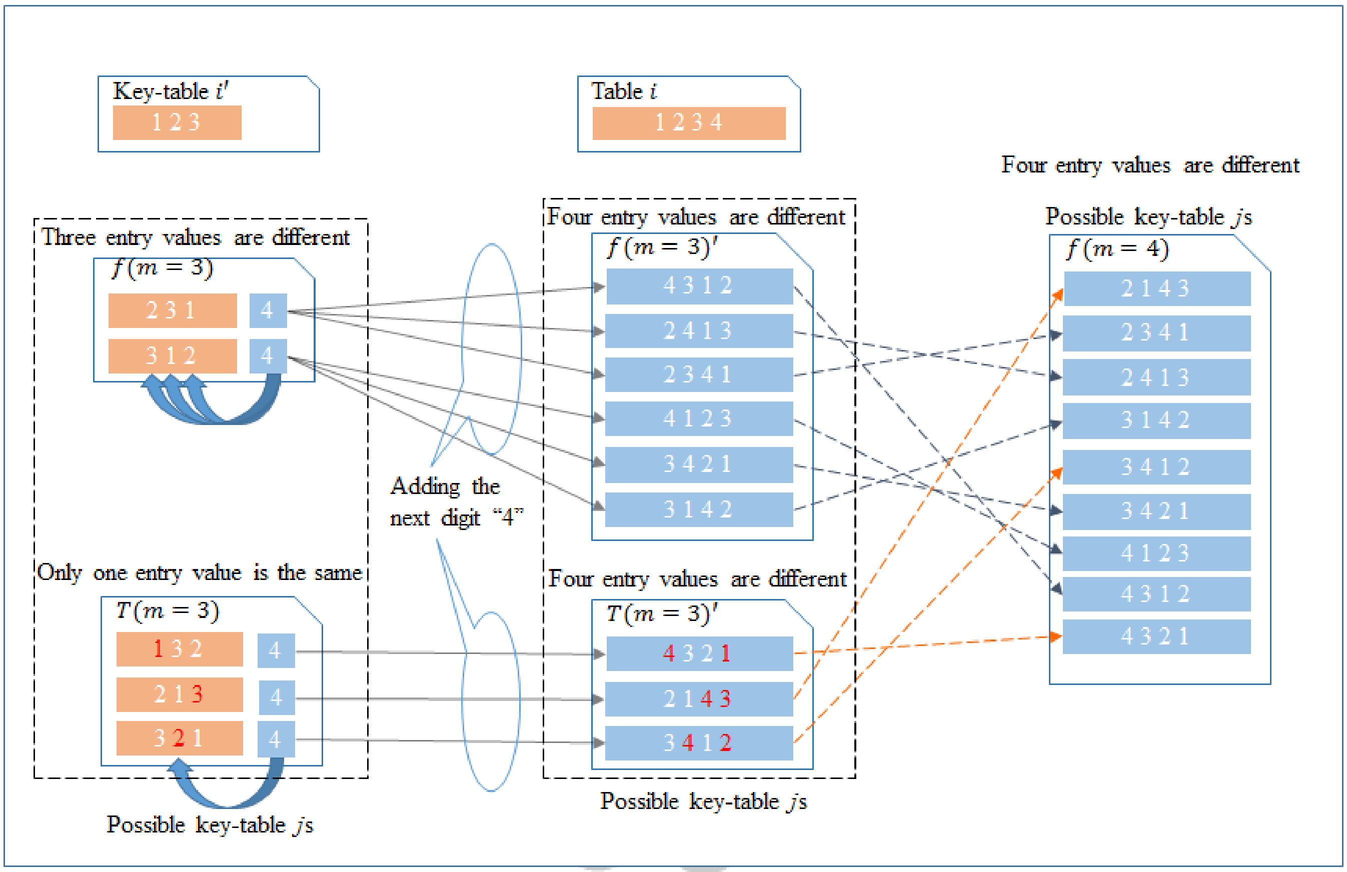


Fig. 12 All corresponding entries in both key-table i and key-table j are different, i.e., $f(m)$

$KT_i(q) \neq KT_j(q)$ for all qs, $0 \leq q \leq \mathrm{F}, q \neq k$. We give a simple example to describe this. Key table i as shown in **Fig. 12** currently has been given entry values 1 2 3, denote by X_3. On the left-hand side of **Fig.** 12, $f(m = 3) = 2$ means that there are two possibilities of key-table j, including the two cases of 2 3 1 and 3 1 2, and $T(m = 3) = 3$ means there are three possibilities of key-table j containing the three combinations of 1 3 2, 2 1 3 and 3 2 1, in each of which there is only one entry *k* with $KT_i(k) = KT_j(k)$. The digit colored with red in a combination represents that it is digit *k*. Now we input the fourth entry value. Assuming that the value without losing its generality is 4. Key-table *i* has been given 1 2 3 4, and $f(m = 3)' = 6$, which means there are 6 possibilties of key-table *j*, including the cases of 4 3 1 2, 2 4 1 3,…, and 3 1 4 2, and $T(m = 3)' = 3$, which represents that there are 3 possibilities of key-table of *j* containing the cases of 4 3 2 1, 2 1 4 3, and 3 4 1 2 in which there is only one entry *k* with $KT_i(k) = KT_j(k)$. Table j on the right-hand side in **Fig.** 12 lists the result of executing our simulation program which exhaustively searches all the possibilities of $f_{ij}(m = 4)$ cases. The results are the same to those produced from $f(m = 3)'$ and $T(m = 3)'$.

`

According to our observation on **Fig. 12** and the results generated by this simulation program, **Eq. (4)** is then derived.

$$f(m) = (m-1)f(m-1) + T(m-1) \qquad (4)$$

Table. 1 shows the result of our another simulation program which counts the number of all possible combinations that individually meet the definitions of $f(m)$ and $T(m)$. From this table, **Eq. (5)** is induced.

$$T(m-1) = f(m-1) - (-1)^{m-1} \qquad (5)$$

After substituting the term $T(m-1)$ on the right-hand side of **Eq. (4)** by **Eq. (5)**, we obtain **Eq. (6).**

$$f(m) = mf(m-1) + (-1)^m \qquad (6)$$

Table 1. The $f(m)$ and $T(m)$ given different *m*s, $2 \leq m \leq 16$.

m	$f(m)$	$T(m)$
2	1	0
3	2	3
4	9	8
5	44	45
6	265	264
7	1854	1855
8	14833	14832
9	133496	133497
10	1334961	1334960
11	14684570	14684571
12	176214841	176214840
13	2290792932	2290792933
14	32071101049	32071101048
15	481066515734	481066515735
16	7697064251745	7697064251744
17	…	…

which is exactly the same as the equation we induce from the results generated by exhaustively searching the possible combinations individually meeting the definitions of $f(m)$ and $T(m)$.

Theorem:

Let key-table A and key-table B be two of 4×4 permutation tables with values $0 \sim F$ distributed to the 16 entries, meaning that no two entries are filled in the same value.

Let $f(m)$ be the number of pairs such that $(16-m)$ corresponding entries of the two tables are individually of the same values, i.e., $KT_i(h) = KT_j(h)$, and $P_{ij}(h) \in EP_{ij} - D_{ij}(m)$ and the values in the *m* remainding entries, $2 \leq m \leq 15$, are individually different, i.e., $KT_i(h') \neq KT_j(h'),$ and $P'_{ij}(h') \in D_{ij}(m)$.

Then $f(m)$ can be defined by the following recurrence relation, i.e.,

$$f(m) = mf(m-1) + (-1)^m, 2 \leq m \leq 15, \text{with } f(2) = 1$$

Proof:

It will be proved by the method of mathematical induction. For $m = 2$, there are $16 - 2 = 14$ entries in key-table A, the values of which are correspondingly the same as those in key-table B, and the values in the remainder 2 entries e.g., entries *q* and *p*, are individually different. Obviously, there is only one possibility and assume that in key-table A, the value of *q* is *r* and that of *p* is *s*. Then in key-table B, the values of *q* and *p* should be *s* and *r*, respectively. Hence, $f(2) = 1$.

For $m = i$, the recurrence relation of $f(i) = if(i-1) + (-1)^i$ is true. Then

For $m = i + 1$, let $1, 2, 3, \ldots, i, i+1$ be a permutation of A that has been filled in to $i + 1$ entries, then the number of possible permutation of key-table B that meets $f(i+1)$ can be calculated as follows.

The $f(i+1)$ is composed of two parts. In the first part, when number of sensors S currently in the system is $i + 1$, the $(i+1)^{th}$ entry can be substituted by any existing value to form a new permutation. We will describe this below.

Assume $i = 4$, key-table A's permutation is 1 2 3 4 and key-table B's permutation is 2 1 4 3. The permutation of key-table A is entirely different from the permutation of key-table B. When $i = 5$ is added, the permutation of key-table A becomes 1 2 3 4 5 and one of key-table B's current value can be swapped with 5 to form a new entirely different permutation, e.g., the value 2 (the first digit) in key-table B's permutation is swaped with 5, resulting in the permutation of 5 1 4 3 2. There are four value in key-table B permutation so it can form four entirely different new permutation for key-table B, and so on. We can then acquire $i \times f(i)$'s new entirely different new permutation with $f(i)$ for key-table B when the $(i+1)^{th}$ digit is added.

In the second part, there are some permutation in key-table B having only one entry *k* with $KT_A(k) = KT_B(k)$, e.g., key-table A 's permutation is 1 2 3 4, and key-table B's permutation is 1 3 4 2. This means $k = 0$ since the first digits of both key-tables are 1s, whereas other entries' values are themselves different. When "5" is added, the permutation of key-table A is 1 2 3 4 5 and in key-table B, the permutation 1 3 4 2 becomes 5 3 4 2 1. Now the two permutations are entirely different, i.e., the newly joining $(i+1)^{th}$ digit is swapped with the value of entry *k*, and the number of only one value which is the same in key-table *i* and key-table *j* is $T(m = 1)$.

As above, when combining the two parts of $f(i+1)$, we can obtain

$$f(i+1) = if(i) + T(i), \text{ and } T(i) = f(i) - (-1)^i$$

where $T(i)$ is the experimental result. We substitute $T(i)$ in $f(i+1)$ with $T(i) = f(i) - (-1)^i$, then

$$f(i+1) = if(i) + \left(f(i) - (-1)^i\right) = (i+1)f(i) + (-1)^{i+1}$$

➔i.e., $f(n) = nf(n-1) + (-1)^n$ where $n = i + 1$ is also true.

By Mathematical Induction, $f(n) = nf(n-1) + (-1)^n,\ 2 \le n \le 15$ is tenable.

Hence,

$$P_c = \frac{16! \times C_m^{16} \times f(m)}{16! \times 16!} = \frac{f(m)}{m! \times (16-m)!} \tag{7}$$

where 16! is number of possible permutation of the entry values $0\sim\mathrm{F}$ in key-table *i* (key-table *j*), $16! \times 16!$ in the denominator is the total number of permutation of the two key-tables, the 16! in the numerator is the number of permutation for the 16 entry values in key-table *i*, and $C_m^{16} = \frac{16!}{m!(16-m)!}$ which represents choosing *m* entries from 16 entries ($2 \le m \le 15$) to form $D_{ij}(m)$.

2. Let P_x be the reference probability defined as the probability with which the *m* entries of different entry values in key-table *i* and key-table *j*, i.e., elements of $P_{ij}(h), P_{ij}(h) \in D_{ij}(m)$, are not referenced by the indexes contained in an upper-CK-seed, or a lower-CK-seed, meaning the indexes carried in the two seeds are all from the $(16-m)$ entries. If the length of the seed used is L ($= 4 \times P$, $P \in N^+$) bits long, i.e., *P* digits, and each entry value (0~F) of a key-table entry is 4 bits in length, then,

$$P_x = \frac{(16-m)^P}{16^P} \tag{8}$$

where 16^P is the number of all possible seeds, and $(16-m)^P$ is the case in which the indexes provided by the seed all come from the $(16-m)$ entries in key-table *i*.

Finally, the probability of common key (CK) collision, denoted by P_{cx} and defined as the CKs generated by individually inputting the seed to key-table *i* and key-table *j* are the same, is

$$P_{cx} = \frac{(16-m)^P}{16^P} \times \frac{f(m)}{m! \times (16-m)!}. \tag{9}$$

6. Simulation

Our experimental environment is developed on two identical desktop computers, the hardware and software specifications of which include: Intel i5-3450 at 3.1GHz; 16GB of memory; OS Window7 64bit with java JDK 7u21. The wireless environment is IEEE.802.11g with 54Mbps as its transmission speed.

6.1. Expression generation times

The expression generation times of the dynamic shared key establishing, key refreshment and data transmission phases on different key lengths are shown in **Table. 2**. This experiment was performed on desktop computers and Wi-Fi, the times required by employing real sensors equipped with ZeeBee as their wireless communication protocol need to be multiplied by 100 times those shown in **Table. 2**. In this table, the key refreshment phase consumes the longest time. That is why we change the CK generation method for key matrix *K*. In the SKMaS, CKs in *K* are all random numbers. The number of CKs in K is $(n-1)^2$. But the NKMaS only creates $5+2(n-3)$ random numbers. The remaining CKs other than $K_{2,3}$ and $K_{3,2}$ are produced by retrieving entries of a key-table. The time consumed is shorter.

Table. 2 The expression generation times the dynamic shared key establishing, key refreshment and data transmission phases.

length	256bit (ms)	512bit (ms)	1024bit (ms)	Note
Dynamic shared key establishing phase				
$Auth = (K_{i,j} \oplus rand_1) +_2 K_{j,i}$ (see **Eq. (1)**)	14.931	31.176	53.845	(1) retrieving $K_{i,j}$ and $K_{j,i}$ from K; (2) generating rand and perform $+_2$ and $\oplus$
$DSK = (K_{i,j} \oplus Auth) +_2 (K_{j,i} \oplus rand_1)$ (see **Eq. (2)**)	1.757	3.284	6.953	(1) retriving $K_{i,j}$, $K_{j,i}$, and *Auth* and rand produced in their generation process; (2) performing $+_2$ and $\oplus$
Key refreshment phase				
$\mathrm{H}(K_{0,0} +_2 rand_2 \vert\vert K_{1,1} \oplus rand_2, rand_2)$ (see **Fig. 4**)	15.331	31.712	58.161	(1) retrieving $K_{0,0}$ and $K_{1,1}$; (2) generating a new *rand*; (3) and performing $+_2$, $\oplus$, $\vert\vert$ and $\mathrm{H}(,)$
$\mathrm{H}(K_{0,0} \oplus rand_3 \vert\vert K_{1,1} +_2 rand_3, rand_3)$ (see **Fig. 5**)	15.364	29.031	57.228	(1) retrieving $K_{0,0}$ and $K_{1,1}$; (2) generating a new *rand*, (3) performing $+_2$, $\oplus$, $\vert\vert$ and $\mathrm{H}(,)$
Data transmission phase				
(d-msg$\oplus$ $Auth) +_2 DSK$ (see **Fig. 7**)	1.615	2.916	5.833	(1) retriving *Auth*, and *DSK*; (2) generating *d*-msg; (3) performing $+_2$ and $\oplus$
$\mathrm{H}'(\ i \vert\vert j \vert\vert rand_1 \vert\vert$($d$-msg$\oplus$ $Auth) +_2 DSK, DSK)$ (see **Fig. 7**)	14.371	28.17	53.632	(1) retrieving (d-msg$\oplus$ $Auth) +_2 DSK$; (2) generating a new rand; (3) performing $+_2$, $\oplus$, $\vert\vert$ and $\mathrm{H}(,)$

The message transmission times are shown in **Table. 3**, in which the longest time in the NKMaS was consumed in key refreshment phase. In this phase, the size of *k*-msg in the SKMaS varies depending on the number of existing sensors, i.e., *n*, in the underlying WSN. But in the NKMaS, it is a constant if the key length is fixed. The network size does not affect the size of the newly-joining message, i.e., $0.007+0.014n$ becomes $0.007+0.014$. Since no messages are transmitted in the key pre-distrbution phase and dynamic shared key establishing phase, they are not listed in **Table. 3**.

Table. 3 The message transmission times of the key refreshment and data transmission phases.

Phase	Message	256bit(μs)	512bit(μs)	1024bit(μs)
The key refreshment	The leaving message (see **Fig. 4**)	7	12	21
	The newly-joining message (see **Figs. 5** and **6**)	7+14	12+29	21+57
The data transmission	The data message (see **Fig. 7**)	12	22	41

6.2. The energy consumption

In this following, the energy consumed by the SKMaS and NKMaS was measured given different numbers of network nodes. Network Simulator tool NS2 is used. The minimum message (a upper-CK seed or lower-CK seed) size is 1KB. The parameter settings employed are shown in Table 4.

Table. 4 The parameter settings of the following experimens.

Setting	Value
Message size	2KB
Transmission speed	2KB/s
Energy cost when a node sends a message	5W
Energy cost when a node receives a message	2W
Energy cost when a node is idle	0.05W
Energy initially assigned to a node	3000J

In the NKMaS, the size of the newly-joining message, denoted by L, is independent from the number of nodes in the network. However, the size of the message sent by KDS in the SKMaS is $n(3L + L_{ID})$, which increases when the number of nodes in the network is higher.

The measurement results are shown in **Table. 5**. In this table, we can see the relation between the number of employed nodes and message sizes in the SKMaS and NKMaS. In theory, the energy consumed by the NKMaS multiplies the numbers of nodes, i.e., *n*, needed to be equal to the energy consumed by the SKMaS. But in **Table. 5,** it is higher than that consumed by the SKMaS. The reason is that we use UDP as the transmission protocol. It will not resend lost messages when the messages are dropped, and the probability with which a huge message is dropped is often higher that of a small. After a message is dropped, it will not continuously consume energy.

Table. 5 The energy consumed by the NKMaS and SKMaS on different numbers of existing nodes.

n	NKMaS(J)	SKMaS(J)
10	27.991928	161.629308
20	41.224396	580.124415
30	57.562310	1255.589979
40	70.058999	2178.796106
50	84.580101	3364.386500
60	96.646534	4813.971057
70	112.366328	6511.985765
80	126.400871	8510.371767
90	145.990014	10789.807046
100	161.367712	13299.770440

7. Conclusions and Future Studies

In this paper, we design and analyze a square key matrix management scheme with which to securely protect wireless sensor networks. To increase the resiliency of sensor networks, our scheme supports an efficient sensor-node-addition mechanism to deal with the dilemma in which since a sensor network does not have available used IDs, adding extra sensor nodes will change the DSKs used by other nodes and may aggravate or even crash the whole system. We also evaluate and show that the proposed system can effectively defend three common attacks. The system enhances the security and resiliency of the sensor networks without conducting tremendous amount of computation and complicated cryptographic techniques.

In the future, we would like to improve the reliability and derive working model for the proposed system and derive the reliability and behavior models for the NKMaS so that users can predict the reliability and behavior before using it. These constitute our future studies.

References

[1] Leu F.Y.: Emerging Security Technologies and Applications. In: Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, September 2011, vol. 2, no. 3, pp. 1-3.

[2] Tanveer A. Zia and Albert Y. Zomaya: A Lightweight Security Framework for Wireless Sensor Networks. In: Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol. 2, no. 3, September 2011, pp. 53-73.

[3] Durisic, M.P. and Tafa, Z., Dimic, G. and Milutinovic, V.: A Survey of Military Applications of Wireless Sensor Networks. In: Mediterranean Conference on Embedded Computing, June 2012, pp. 196-199.

[4] Bekmezci, I. and Alagoz, F.: Energy E_cient, Delay Sensitive, Fault Tolerant Wireless Sensor Network for Military Monitoring. In: IEEE International Conference on Sensors Applications Symposium, February 2008, pp. 172-177.

[5] Ranasingha, M.C. and Murthi, M.N. and Premaratne, K. and Fan, X.: Transmission Rate Allocation in Multi-Sensor Target Tracking. In: IEEE International Conference on Acoustics, Speech and Signal Processing, March/April 2008, pp. 3021-3024.

[6] Chen Y.M, Shen W., Huo H.W. and Xu Y.Z.: A Smart Gateway for Health Care System Using Wireless Sensor Network. In: International Conference on Sensor Technologies and Applications, July 2010, pp. 545-550.

[7] Johansson, A., Shen, W. and Xu, Y.: An ANT Based Wireless Body Sensor Biofeedback Network for Medical E-Health Care. In: International Conference on Wireless Communications, Networking and Mobile Computing, 2011, pp. 1-5.

[8] Kong Y.F. and Jiang P.: Development of Data Video Base Station in Wa-ter Environment Monitoring Oriented Wireless Sensor Networks. In: International Conference on Embedded Software and Systems Symposia, July 2008, pp. 281-286.

[9] Freeman, J.D. and Simi, S.: Remote Monitoring of Indoor Environment using Mobile Robot Based Wireless Sensor Network. In: International Conference on Computer Science and Education, August 2011, pp. 1080-1084.

[10] Elmagid, A.A.A., Ramadan, R.A., El-Ghanam, S.M., Al-tabbakh, S.M., Kamh, S.A. and Marie, M.: Radiation Detection Based Heterogeneous Wireless Sensor Network. In: International Conference on Engineering and Technology, October 2012, pp. 1-6.

[11] Jian B., Luo C.Y., Guo Y.H. and Li W.: A New Key Pre-distribution Scheme for Wireless Sensor Networks. In: International Symposium on Information Engineering and Electronic Commerce, May 2009, pp. 333-336.

[12] Sharmila Deva Selvi1 S., Sree Vivek1 S., Vivek Krishna Pradhan., and Pandu Rangan C.: E_cient Certi_cateless Online/O_ine Signature. In: Journal of Internet Services and Information Security, November 2012, vol 2, issue 3/4, pp. 77-92.

[13] Claycomb W., Shin D.: Extending Formal Analysis of Mobile Device Authentication. In: Journal of Internet Services and Information Security, May 2011, vol 1, issue 1, pp. 86-102.

[14] Wuu L.C., Hung C.H. and Chang C.M.: Quorum-based Key Management Scheme in Wireless Sensor Networks. In: International Conference on Ubiquitous Information Management and Communication, 2012, No. 15.

[15] Huang Y.L. and Leu F.Y.: Constructing a Secure Point-to-Point Wireless Environment by Integrating Di_e-Hellman PKDS RSA and Stream Ciphering for

Users Known to Each Other. In: Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol. 2, no. 3, September 2011, pp. 96-107.

[16] Castiglione, A., Cattaneo, G., Cembalo, M., De Santisa, A., Faruolo, P., Petagna, F., Petrillo, U.F.: Engineering a secure mobile messaging framework. In: Computers and Security, vol. 31, September 2012, pp. 771-781.

[17] Castiglione, A., Cattaneo, G., Maio, G.D., Petagna, F.: Secure End-to-End Communication over 3G Telecommunication Networks. In: International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2011, pp. 520-526.

[18] De Santis, A., Castiglione, A., Cattaneo, G., Cembalo, M., Petagna, F., Petrillo, U.F.: An Extensible Framework for E_cient Secure SMS. In: International Conference on Complex, Intelligent and Software Intensive Systems, February 2010, pp. 15-18.

[19] Hwu J.S., Hsu S.H, Lin Y.B, Chen R.J.: End-to-end security mechanisms for SMS. In: International Journal of Security and Networks, 2006, vol 1, no. 3/4, pp. 177-183.

[20] Agoyi, M., Seral, D.: An Asymmetric Encryption Approach. In: International Conference on Wireless and Mobile Communications, September 2010, pp. 448-452.

[21] Cheng, Y. and Agrawal, D.P.: E_cient Pairwise Key Establishment and Management in Static Wireless Sensor Networks. In: IEEE International Conference on Mobile Ad hoc and Sensor Systems, 2005, pp. 544-550.

[22] Hung Y.C., Chen R.C., and Shen A.: Efficient key pre-distribution for sensor nodes with strong connectivity and low storage space. In: International Conference

on Advanced Information Networking and Applications, 2008, pp. 327-333.

[23] Blom R.: An optimal class of symmetric key generation systems. In: Advances in cryptology: theory and application of cryptographic techniques, 1985, pp. 335-338.

[24] Wang E.K., Ye Y.: An Efficient and Secure Key Establishment Scheme for Wireless Sensor Network. In: International Symposium on Intelligent Information Technology and Security Informatics, 2010, pp. 511-516.

[25] Castiglione, A., Marco Cepparulo, M., De Santis, A., Palmieri, F.: Towards a Lawfully Secure and Privacy Preserving Video Surveillance System. In: International Conference on E-Commerce and Web Technologies, 2010, pp. 73-84.

[26] Chen R.: Research on Security Authentication of Hierarchy Distributed Wireless Sensor Network. In: International Conference on Computer and Automation Engineering, February 2010, pp. 275-278.

[27] Stephens, B., Tectura, Corp. and Bellevue, W.A.: Security Architecture for Aeronautical Networks. In: Digital Avionics Systems Conference, October 2004, vol. 2.

[28] Xue H., Zhang H., Qing S. and Yu R.: Automated Design of Non-Repudiation Security Protocols. In: International Conference on Wireless Communications, Networking and Mobile Computing, September 2007, pp. 2318-2321.

[29] He X.B., Niedermeier, M., Meer, H.D.: Dynamic key management in wireless sensor networks: A survey. In:Journal of Network and Computer Applications, 2013, vol. 36, issue 2, pp. 611-622.

[30] Huang, X., Ahmed, M. and Sharma, D.: Timing Control for Protecting from Internal Attacks in Wireless Sensor Networks. In: International Conference on Information Networking, February 2012, pp. 7-12.

[31] Li Z. and Ravishankar, C.V.: A Fault Localized Scheme for False Report Filtering in Sensor Networks. In: International Conference on Pervasive Services, July 2005, pp. 59-69.

[32] Liu, D. and Ning, P.: Improving Key Pre-Distribution with Deployment Knowledge in Static Sensor Networks. In: ACM Transactions on Sensor Networks, 2005, vol.1, pp. 204-239.