LU Decomposition-Based Key Predistribution Scheme for Heterogeneous Wireless Sensor Networks*

CHUNGUANG MA AND XIAORUI ZHONG
College of Computer Science and Technology
Harbin Engineering University
Harbin, 150001 P.R. China
E-mail: {machunguang; zhongxiaorui}@hrbeu.edu.cn

Since LU decomposition-based key predistribution schemes can be compressed easily, they are very appropriate for source-limited sensor networks. But existing schemes have two flaws: (1) they all focus on key establishment without considering key or key material update, thus they can’t support extra nodes addition or deletion and their supported network size is limited; (2) they are vulnerable to LU attack. To conquer those drawbacks, a new LU decomposition-based key predistribution scheme for clustered heterogeneous sensor networks is proposed. Our scheme applies perturbation technology on matrices which are generated by LU decomposition, and establishes common bit-string between nodes by exchanging public vectors. These common bit-strings shared by nodes are randomly truncated and concatenated one by one to form pairwise keys. When new nodes arrive or old nodes leave, this scheme updates key materials quickly by exploiting LU attack principle. Security and performance analysis show that our proposed scheme is feasible for sensor networks and has following two advantages: (1) It is robust to eavesdropping attack, Albrecht’s attack, LU attack and node capture attack; (2) It supports efficient key update for adding and deleting nodes while keeping original information about private matrix in secret.

Keywords: wireless sensor networks, pairwise key, key establishment, key update, LU decomposition, perturbation

1. INTRODUCTION

A wireless sensor network (WSN) is composed of a lot of resource-limited sensor nodes. Since WSNs can be deployed in various environments, including unattended or hostile environment, to transmit sensitive information, it’s essential to provide efficient security services. In such purpose, many cryptography systems are adopted to protect communication. Key predistribution is an important and efficient way to guarantee that cryptography systems can work as well as we expect. However, most of the well known key predistribution schemes such as asymmetric key distribution schemes require significant computation and communication overhead. They are not feasible to be applied directly in WSNs. Providing efficient key predistribution schemes for WSNs has become a great challenge [1].

In existing researches, there are two main categories of key predistribution schemes: one is probabilistic key predistribution (PKP) schemes and the other is deterministic key
predistribution (DKP) schemes. PKP schemes, such as E-G scheme [2] and Panyim’s scheme [3], only can establish pairwise keys between nodes with a certain probability. That means even two nodes are geographically very close to each other, it is possible that they don’t share a key. Different from PKP schemes, DKP schemes can guarantee that any two nodes in network will be able to establish at least one pairwise key. Most of existing DKP schemes are based on matrix, such as Blom’s scheme [4] and its derivative scheme [5]. Such schemes usually negotiate a pairwise key via exchanging a part of pre-loaded matrix information.

In comparison with other matrix-based DKP schemes, LU decomposition-based schemes, for instance [6-13], are compressible and flexible. LU decomposition factorizes a symmetric matrix $K$ as the product of a lower triangular matrix $L$ and an upper triangular matrix $U$. A LU decomposition-based scheme usually assigns the $i$th row of $L$ and the $i$th column of $U$ to the $i$th node. When node $i$ and $j$ try to establish a pairwise key, they only need to exchange their public vectors. However, existing LU decomposition-based schemes are suffering from LU attack [14]. When two nodes exchange their public vectors, they can also calculate part or all of the private information of each other. Detailed introduction about LU attack is described in Section 2. Besides, there is always some nodes will die or leave over time, new nodes need to be added to keep network working. But existing LU decomposition-based schemes barely provide support for node addition and deletion, the network scale is limited to the dimensionality of symmetric matrix $K$.

To address these flaws, we focus on two problems in this paper. The first one is how to apply perturbation on LU decomposition matrices to provide resilience to LU attack and generate pairwise key between nodes for a clustered sensor network. The second one is how to utilize LU attack principle to provide scalability so that key materials stored in nodes can be quickly update when new nodes arrive and old nodes leave.

The rest of this paper is organized as follows. In Section 2, we introduce existing researches about LU decomposition-based schemes. Section 3 describes our basic idea about protecting communication against LU attack, finding common information and key establishment. The detailed scheme is proposed in Section 4. Sections 5 and 6 are security and performance analysis respectively. Finally, we conclude the whole work in Section 7.

2. RELATED WORKS

The first LU decomposition-based scheme, refered to as the basic LU scheme, is proposed by Choi and Youn [6]. It first decomposes a symmetric matrix $K$ into a lower triangular matrix $L$ and an upper triangular matrix $U$. Matrix $K$ consists of keys selected from a key pool. After LU decomposition, the $i$th row of $L$, denoted by $L^i$, and the $i$th column of $U$, denoted by $U^i$, are both assigned to the $i$th node. When two nodes want to establish a key, they exchange their column vectors $U^i$ and calculate a key as $L^iU^j$. As an improved version, Park’s scheme [7] simplifies the method of constructing LU decomposition so that the time overhead is reduced. In order to decrease network-wide storage overhead, Pathan et al. [8] introduces an efficient encoding mechanism to store vectors into sensor nodes. For each vector, elements are divided into two parts: the non-zero-element part and the zero-element part. The latter one is replaced by an integer
which denotes the number of zero elements in current vector. Pathan’s scheme significantly increases the storage efficiency than those previously proposed scheme which is also based on LU decomposition. The above schemes all focus on how to establish shared key without considering the change of network topology. By combining Pathan’s scheme [8] with clustered structure and hash function, Chen’s scheme [9] provides key update and deletion when network topology changes to avoid key information to be revealed. This scheme uses Pathan’s scheme to establish shared key between cluster heads. In a cluster, nodes are pre-loaded two pair of random cluster head ID. Intra-cluster key establishment only generates pairwise keys between sensor node and its cluster head. In details, node sa provides its pre-loaded cluster head ID pair, (ch1, ch2) and (ch3, ch4), to its cluster head ch5, ch5 asks cluster heads ch1~ch4 for their shared key k12 and k34, hashes (k12, k34) to form a shared key between sa and ch5. Though Chen’s scheme can handle network topology change, it isn’t secure to reveal shared keys between cluster heads. MKPS scheme [10] tries to minimize the time overhead, it performs LU decomposition first as well, and then further decomposes U into a diagonal matrix D and a new upper triangular matrix U’. Namely, MKPS scheme constructs a symmetric matrix K in the form of LDU’, assigns rows of L and D, column of U’ to nodes. Finally, establish key by exchanging columns of U’. The polynomial LU decomposition scheme [11] replaces the numeric elements of matrices with polynomials so that keys can change with time. Zeng et al. [12] decompose a shared key into two vectors R and C. The vector R is held privately by the sensor, while the other vector C is protected and disseminated using network coding. Scheme proposed by Zhou et al. [13] is the latest LU decomposition based key predistribution scheme, which combine the Chinese remainder theorem (CRT) with LU matrix. The CRT is used to secure the communication keys between cluster head and its cluster members. The communication keys among cluster heads or between cluster head and base station are generated with LU matrix. In a summary, the general method of LU decomposition-based schemes to establish shared keys is:

**Step 1: LU Decomposition** Apply LU decomposition on a symmetric matrix K to generate a lower triangular matrix L and an upper triangular matrix U;

**Step 2: Assign Vectors** Assign L’ and U’ into the ath node sa. L’ is kept in secret while U’ is public;

**Step 3: Establish Pairwise Key** Any two nodes, sa and sb, establish a shared key by exchanging their column vectors U and calculate \( k_{sa,b} = L'U_{sa} = L'U_{sb} = k_{sa,b} \).

Existing LU decomposition-based key predistribution schemes can’t provide great support to network topology change, and they are all very vulnerable to the attack launched by Zhu et al. [14]. It is proved that the result matrices of LU decomposition are linearly dependent, namely, the elements of matrices L and U satisfy \( l_{aba,mb} = l_{ba,m} \) (1 \( \leq a, b, m \leq n \)). Hence, if an adversary has already known a private row vector L’ and two public column vectors U, U’, he/she can immediately calculate the first z elements of the unknown secret row vector L = \((l_{1}, l_{2}, ..., l_{z})\), where integer z implies the number of non-zero elements of U’ and
\[ l_{b, w} = \frac{l_{a,n} u_{m,b}}{u_{m,a}}, \quad (u_{m,a} \neq 0, 1 \leq m \leq z). \] (1)

That is, if \( L U' \) equals to \( L U \) and only one of vectors \( L_1, L_2, U_1, U \) is secret, an adversary could easily obtain the secret one. More specifically, the secret \( L' \) stored in one node can be calculated by the other node after exchanging their columns of \( U \), so any key established by this exposed \( L' \) will not be a secret any longer. We call the behavior that using Eq. (1) to obtain secret keys established based on LU decomposition as LU attack in this paper. Obviously, when the number of non-zero elements of \( L_0 \) is smaller than \( z \), adversary can calculate the whole \( L \) and gain all keys shared between \( s_a \) and any other nodes. In general, the ability of an adversary to break a key depends on how many non-zero elements of secret row vectors have been revealed.

None of existing schemes which use symmetric matrix \( K \) as key source is robust to LU attack. In order to resist LU attack, the first task is to break the relationship between matrices \( L \) and \( U \). Perturbation technology can achieve this goal. It was introduced into key predistribution for the first time in 2007, by Zhang et al. [15]. The authors also provided an authentication scheme [16] in the next year. After that, Yu et al. [17] improved Blom’s scheme in terms of computational breaking complexity by using perturbation based on [15, 17]. Though an attack was launched in [19], it’s only worked to perturbation polynomials. Taking perturbation into consideration reveals a chance to save LU decomposition-based schemes from LU attack.

### 3. BASIC IDEA

First of all, the symbols involved in this paper are introduced in Table 1.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Explanation</th>
<th>Symbols</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_a )</td>
<td>the ( a )th row of matrix ( A )</td>
<td>( r )</td>
<td>Noise factor</td>
</tr>
<tr>
<td>( A^a_i )</td>
<td>the ( a )th column of matrix ( A )</td>
<td>( s_i )</td>
<td>ID of the ( i )th node</td>
</tr>
<tr>
<td>KMG</td>
<td>key material group</td>
<td>NKM</td>
<td>node’s key material</td>
</tr>
<tr>
<td>((u)mid )</td>
<td>ID of the ( u )th KMG</td>
<td>((u)m )</td>
<td>NKM which comes from the ( u )th KMG</td>
</tr>
<tr>
<td>( rdk(a, b) )</td>
<td>generate number ( r[a, b] )</td>
<td>( B(b) )</td>
<td>the simplest form of ( b )</td>
</tr>
<tr>
<td>( R(k, r) )</td>
<td>perturb the last ( r ) bits of ( k )</td>
<td>( \zeta )</td>
<td>The total number of rounds</td>
</tr>
<tr>
<td>((u)A )</td>
<td>( A ) which comes from the ( u )th KMG</td>
<td>( A^{(v)} )</td>
<td>( A ) which comes from the ( v )th round</td>
</tr>
<tr>
<td>( length(A) )</td>
<td>binary length of ( A )</td>
<td>( Len )</td>
<td>required length of key</td>
</tr>
<tr>
<td>( non(A) )</td>
<td>The number of non-zeros in vector ( A )</td>
<td>( k_{a,b} )</td>
<td>element of matrix ( K ) at position ( (a, b) )</td>
</tr>
<tr>
<td>( (A)_2 )</td>
<td>The binary representation of integer ( A )</td>
<td>( (A)_{10} )</td>
<td>The decimal representation of integer ( A )</td>
</tr>
<tr>
<td>( cm(b, p, q) )</td>
<td>delete the last ( q ) bits of ( b ) and return a ( p - q ) bits bit-string</td>
<td>( chsb_{a,b} )</td>
<td>the common bit-string between node ( s_a ) and ( s_b )</td>
</tr>
</tbody>
</table>
In order to resist LU attack, we need to break the direct relation between matrices \( L \) and \( U \) to increase the difficulty of exposing secret row vectors. Perturbation technology can help us to achieve this goal which will change two equal numbers into two different ones, but keeping them sharing a common part. However, if improper perturbation is applied, either additional computational or communication overhead is needed to extract common information. Furthermore, we try to weaken the dependence of shared key on row vector \( L_r \) by randomly cutting out a part of the common information to establish shared key, so that the impact of LU attack on our scheme can be weakened too.

Assume that only \( m \) sensor nodes \( I = \{s_1, \ldots, s_m\} \) try to establish pairwise key. \( K, L, U \) are three \( m \times m \) matrices, \( K = LU \) is a symmetric matrix, \( L \) is a lower triangular matrix and \( U \) is an upper triangular matrix. If we could find another matrix \( K' = WU \) whose elements are as same as \( K \) except the last \( r \) bits, then any two node, \( s_a \) with key material \( (L_r, U_r) \) and \( s_b \) with key material \( (L_r, U_r) \), can extract a common bit-string from \( k_{r,a} = W_{r}U_{r} \) and \( k_{r,b} = W_{r}U_{r} \). An example is given in Fig. 1.

Let \( R(k, r) \) be a function of positive integer \( k \) and \( r \), and it will change the last \( r \) bits of \( k \)'s binary representation. For example, \( R(45,2) \) could be one of 101100, 101110 and 101111. Parameter \( r \) is called noise factor. In Fig. 1, \( K \) is an initial symmetric matrix. Elements of \( K' \) equal to \( R(k_{r,b},4) \). Because \( K' = WU \) and the inverse matrix of \( U \) always exists, it’s easy to get \( W = K'U^{-1} \) and \( ns = W - L \). Matrix \( ns \) is referred to as noise matrix, and its row vector \( ns \) is called noise vector. It can be found from Fig. 1 that \( U' \), when \( U \) is transmit public, an adversary could immediately get pairwise keys between node \( s_1 \) and \( s_i (i \neq 1) \) by ask for exchanging \( U'_i \) with \( s_i \). Hence, the first row of \( K, L \) and \( U \) will be discarded by default.

Considering node \( s_2 \) and \( s_3 \), if materials assigned to them are \( (L_2, U_2) \) and \( (L_3, U_3) \)
respectively, after exchanging their column vector, $s_3$ can launch LU attack and calculate $l_{2,1} = l_{3,1}u_{1,2}/u_{1,3}$ and $l_{2,2} = l_{3,2}u_{2,2}/u_{2,3}$. $L_3$ exposed. If materials assigned to these two nodes are $(W_2^r, U_2^r)$ and $(W_3^r, U_3^r)$ respectively, calculating $w_{3,1}u_{1,2}/u_{1,3}$ can't help $s_3$ to get $w_{2,1}$ any longer, $W_2^r$ is safe. By the way, though $k_{2,3} = W_2^rU_2^r \neq W_3^rU_3^r = k_{3,2}$, $k_{3,2}$ still shares a common bit-string 11101 with $k'_{3,2}$.

Once two nodes share common information, they can use it to establish a pairwise key. The simplest way is to repeat the process of generating common bit-string until the sum of length of all common bit-strings is long enough. At this moment, a catenation of these common bit-strings can be used as a shared key. Call the $i$th common bit-string used to form the final shared key the $i$th key component. This method is not safe under LU attack, because once one of the noise vectors $v_i$ is exposed to an adversary who has already known $W_i^r$, $L_i$ will be calculated immediately, and node $s_i$ is compromised. To enhance security, we choose another way to establish shared key. Let $cm(b, p, q)$ be a function which returns a $p - q$ bits bit-string $b'$ which equals to the value of $b$ after $b$ deletes its last $q$ bits. If $B(b)$ denotes the simplest form of bit-string $b$, function $\text{length}(b)$ returns the length of $b$, and $\text{Len}$ is the required length of final shared key, parameters $b, p, q$ would satisfy the relation $r \leq q \leq \text{length}(B(b)) \leq \text{length}(b) \leq p \leq r\text{Len}$. Take binary string 0110 for example, $cm(0110, 8, 3) = 00001$, $B(0110) = 110$, $\text{length}(0110) = 4$. When nodes $s_a$ and $s_b$ want to establish a pairwise key, they could randomly pick a pair of valid $(p, q)$ and use $cm(W_{ab}^r, p, q)$ as a key component. Since parameters $p$ and $q$ are not fixed, it becomes very hard for adversaries to gain the shared keys between nodes.

4. THE PROPOSED SCHEME LU3D

In this section, a new key predistribution scheme is proposed. For the reason that this scheme is LU decomposition-based, dynamic and deterministic, we call it LU3D scheme for short.

4.1 Network Model

Assume that $N$ sensor nodes are deployed simultaneously in a monitoring area to compose a sensor network. Sensor nodes are divided into two categories: the normal ones and the advanced ones. Advanced nodes which are resource-free become cluster heads while the normal nodes which are resource-limited become cluster members. Base station acts like an interface between sensor network and the other networks. It is not only the destination of data, but also the source of control information for the whole network. Cluster heads control clusters and deliver information between base station and members. Different from cluster members, cluster heads are more powerful and secure. There are $G$ clusters in the whole network, and each cluster has $m$ members by default. Cluster heads and the base station compose a top cluster, each cluster head and its members compose a normal cluster. Network topology dynamically changes, some new nodes will apply to join the network and some member nodes may try to leave.

4.2 Initialization

Key materials should be preloaded into sensor nodes before they are deployed into
monitoring area. Initialization process is finished offline by base station. Detailed steps are listed as follow:

Step 1: Generate Key Pool and Functions Generate a large key pool \( P \). Let \( |P| = 2^q - 1 (\tilde{q} \in N^+ \land \tilde{q} \neq 1) \). Construct function \( cm(b, p, q) \) and random number generator \( rd(\min, \max) \) which will return a positive integer \( i \in [\min, \max] \). Choose a noise factor \( r \), make sure \( r \leq \min\{0.5\tilde{q}, \min_{\text{len}}\} \) where \( \min_{\text{len}} \) is the minimal key length.

Step 2: Generate Key Material Group Randomly select several keys from \( P \) to construct matrices \( K \), decompose \( K \) into \( L \) and \( U \) so that \( K, L \) and \( U \) will follow the LU decomposition rule. Calculate \( K^{v}, L^{v}, U^{v}, W^{v}, ns^{v} \) for cluster \( g \), where \( K^{v}, L^{v}, U^{v}, W^{v}, ns^{v} \) represents the \( u \)th KMG of cluster \( g \). Load “node’s key material” (NKM for short) \( (g, a, r, cm, rd, m^{e}_a, m^{f}_a) \) to member node \( a \) of cluster \( g \), where \( m^{e}_a = (v^{e}mid^{e}, v^{e}K^{e}, v^{e}U^{e}) \), integer \( a \) is the member index of \( a \) in cluster \( g \).

After initialization, each cluster head will hold two NKMs, one for top cluster, one for normal cluster. Notice that the member index \( a \) of node \( a \) is unique in a cluster and equals to the number of non-zero elements of \( L^{e} \). But it can’t be used to distinguish nodes or replace node ID.

4.3 Key Establishment

Initialized nodes are randomly scattered in monitoring area. Based on the idea mentioned in Section 3, any two nodes, \( a \) and \( b \), who are located in the same cluster can establish a shared key. Assume that in order to get a \( Len \)-bit shared key \( k_{ab} \), the process of generating key component needs to be executed \( z \) rounds. Here \( E_{k}[M] \) means that plaintext \( M \) is encrypted using the secret key \( k \) of some secure symmetric cryptosystem, and \( D_{k}[M] \) means that ciphertext \( M \) is decrypted using the secret key \( k \) of the corresponding symmetric cryptosystem. The detailed steps for establishing shared key are described as follows:

Step 1: Choose a KMG for Current Round At the \( v \)th round \( (1 \leq v \leq \tilde{q}) \), \( s_{a} \) chooses the \( u \)th KMG of cluster \( g \) and broadcasts \( (v^{e}mid^{e}, v^{e}U^{e}) \).

Step 2: Generate Random Parameters When \( s_{a} \) received the broadcast information of \( s_{a} \), it calculates a bit-string \( k^{(v)} = (v^{e}W^{e}, a^{e}U^{e}) \), \( l_{b} = \text{length}(b^{(v)}) = \text{length}(k_{ab}^{(v)}) \) and the simplest common bit-string \( cb_{ab}^{(v)} = cm(b^{(v)}, l_{b}, r) \). After that, \( s_{b} \) selects two random parameters \( q^{(v)} = rd(r, l_{b} - 1) \), \( p^{(v)} = rd(l_{b}, q^{(v)} + Len) \), and sends message \( \{E_{cb_{ab}^{(v)}}(p^{(v)}, q^{(v)}), (a^{e}U^{e}_{b})\} \) to \( s_{a} \).
Step 3: Generate Key Component  
$s_a$ calculates $b^{(i)} = s_a^{(i)}W^W, c^{(i)}U^F, cbs^{(i)} = cm(b^{(i)}, l_b, r)$, and decrypts $E_{b^{(i)}}(p^{(i)}, q^{(i)})$ to get $(p^{(i)}, q^{(i)})$. Then the $v$th key component is calculated as $k^{(v)}_{a,b} = cm(b^{(i)}, p^{(i)}, q^{(i)})$. So does $s_b$.

Step 4: Generate Pairwise Key  
Repeat Steps 1-3 $\xi$ times to make length $k^{(1)} + \ldots + length(k^{(\xi)}) \geq Len$. Let $k = k^{(1)}_a || \ldots || k^{(\xi)}_a$, then the shared key between $s_a$ and $s_b$ is $k_a b = cm(k, length(k) - Len)$.

4.4 New Node Arrival  
As we mentioned in Section 4.1, each cluster has $m$ original members. When a new node $s_{m+1}$ arrives, the original matrices $(K, K', L, U, W, ns)$ have to extend their dimensions to support pairwise key establishment. It is interesting that although the LU attack threatened the security of secret matrix $L$, it also provides us an efficient way to extend original matrices.

Matrices Extension  
To add a new node to a cluster needs to append a new row and column to original matrices $(K, K', L, U, W, ns)$. In order to maintain existing shared keys, elements in these matrices should be preserved. After extension, the matrix dimension will change from $m \times m$ to $(m + 1) \times (m + 1)$. Considering that matrices $L$ and $U$ are both triangular, the last element of $L_i$ and $(1 \leq i \leq m)$ are both 0. Hence, the method to extend matrix $K$ is:

Step 1: Extend $K$  
Randomly choose $m + 1$ keys from key pool $P$ and use them to form the $(m + 1)$th row and column of $K$.

Step 2: Set 0 elements  
Set $L_{i,m+1}$ and $U_{m+1,i}$ to 0 where $1 \leq i \leq m$.

Step 3: Extend $L$  
Solve the system of linear Eq. (2) and set $l_{m+1,m+1}$ to be a random number.

\[
\begin{align*}
  k_{m+1,1} &= l_{m+1,1}u_{1,1} \\
  k_{m+1,2} &= l_{m+1,1}u_{1,2} + l_{m+1,2}u_{2,2} \\
  \vdots \\
  k_{m+1,m} &= l_{m+1,1}u_{1,m} + \cdots + l_{m+1,m}u_{m,m} \\
  k_{m+1,m+1} &= \frac{k_{m+1,1}}{u_{1,1}} = \frac{l_{m+1,1}}{u_{1,1}} \\
  k_{m+1,2} &= \frac{k_{m+1,2} - l_{m+1,1}u_{1,2}}{u_{2,2}} \\
  \vdots \\
  k_{m+1,m} &= \frac{k_{m+1,m} - l_{m+1,1}u_{1,m} - \cdots - l_{m+1,m-1}u_{m-1,m}}{u_{m,m}} \\
\end{align*}
\]

Step 4: Extend $U$  
According to LU attack principle that $l_{a,b} = l_{b,a}$, the system of linear Eq. (3) can be constructed. Solve it and the extension of $U$ is finished.

Step 5: Extend $K'$  
Set $k'_{i,j} = R(k_{i,j}, r)$ where $i = m + 1 \land j \in [1, m + 1]$ or $j = m + 1 \land i \in [1, m + 1]$.

Step 6: Extend $W$ and $ns$  
$W = K'U^T$, $ns = W - L$.

All extension operations are carried out at base station too. Fig. 2 shows an extension example of matrices in Fig. 1.
LUD-BASED KEY PREDISTRIBUTION SCHEME FOR HSNs

Key Refresh

After matrices extension, each old sensor node is able to share a new pairwise key with a new member. Let $s_{new}$ be the new member’s ID. The cluster head of cluster $g$ is $s_{new}$ whose node index is $c_{new}$ in the top cluster and $c_h$ in cluster $g$ respectively. In order to avoid replay attack, we choose the round trip time from node to its cluster head as a time threshold $T$. The steps for node $s_i$ to refresh its key materials are described as follows:

**Step 1: Initialize the New Node**  
Base station chooses a cluster $g$ from $s_{new}$, and extends all original matrices of cluster $g$. Deploy $s_{new}$ into cluster $g$ after the key materials $(k_{new, m+1}, ..., k_{new, m+n})$ are stored into $s_{new}$.

**Step 2: Notice Cluster Head**  
Base station notifies the cluster head $s_{new}$ of cluster $g$ by...
sending information $B_{\text{new}} = E_{i, ch} \{m + 1, (v)_w, u, |1 \leq u \leq t\}$, where $time$ is a timestamp.

**Step 3: Notice Members** $s_g^ch$ uses key $k^*_{g, new}$ to decrypt $B_{\text{new}}$, then checks whether (1) the first element $m + 1$ equals to 1 plus the dimension of its vectors $W$ and (2) $time < T$. If it is, $s_g^ch$ extracts $(a)_{u, i, j}$, sets $(a)_{u, i, j}$ to 0 and transmits $B'_{\text{new}} = E_{i, ch} \{m + 1, (v)_{u, i, j}, time \leq t, i \neq ch\}$ to Member node $s_i$.

**Step 4: Refresh Member’s Key Materials** $s_i$ uses key $k^*_{i, ch}$ to decrypt $B'_{\text{new}}$, and checks whether (1) the first element $m + 1$ equals to 1 plus the dimension of its vectors $W$ and (2) $time < T$. If it is, $s_i$ extracts $(a)_{u, i, j}$, sets $(a)_{u, i, j}$ to 0.

When above steps are finished, every two nodes in cluster $g$ can build a secure link by establishing a pairwise key. If $s_{\text{new}}$ is a cluster head, base station only needs to generate several new key materials for new cluster and deploy $s_{\text{new}}$ into sensor network.

### 4.5 Key Revoke

Suppose that all compromised nodes and energy-exhausted nodes can eventually be detected by the cluster heads. After such events are detected, innocent nodes must update their key information for security or saving resources and starts key revoke process. The main issue of revoking a node is to inverse the process of matrices extension. Comparing with extension, inverse process is much easier.

**Revoke a normal node $s_i$ from cluster $g$:**

**Step 1: Notice Members** Cluster head $s_g^ch$ announces that member node with index $i$ is invalid by sending message $B_{\text{rvk}} = E_{i, ch} \{m - 1, time\}$ to member $s_j$.

**Step 2: Update Materials** Upon receiving the message, $s_j$ of cluster $g$ using $k^*_{i, ch}$ to decrypts $B_{\text{rvk}}$, then checks whether (1) the first element $m - 1$ equals to the dimension of its vectors $W$ minus 1 and (2) $time < T$. If it is, $s_j$ deletes the $i$th element of $(W)^r$ and $(U)^r$ where $1 \leq u \leq t$, and updates its index by $j = j - 1$ if and only if $j > i$.

**Revoke a cluster head $s_g^ch$ from cluster $g$:**

**Step 1: Add New Cluster Head to Clusters** Base station chooses a new cluster head $s_{\text{new}}^ch$, and adds it to both of the top cluster and cluster $g$ like adding a normal member by following the rules mentioned in 4.4.

**Step 2: Revoke Target from the Top Cluster** $s_g^ch$ is revoked from the top cluster by following the way of revoking a normal member.

**Step 3: Revoke Target from Cluster $g$** Cluster head $s_g^ch$ sending message $B_{\text{rvk}} = E_{i, ch} \{m, \text{chnew}, time\}$ to member $s_j$ including new cluster head $s_{\text{new}}^ch$. $s_i$ checks whether (1) the
first element \( m \) equals to the dimension of its vectors \( W \), (2) \( time < T \), and (3) the new cluster head index \( ch_{new} \) isn’t equal to the current cluster head index \( ch \). If it is, \( s_i \) delete the \( ch \)th element of \( ^{\omega}W^i \) and \( ^{\omega}U^i \) where \( 1 \leq u \leq t \), and updates its index by \( i = i - 1 \) if and only if \( i > ch \).

5. SECURITY ANALYSIS

In this paper, we focus on the effect of malicious attacks on our scheme, including eavesdropping attack, Albrecht’s attack [19], node capture attack and LU attack [14]. For comparison, 4 classic LU decomposition based schemes are selected. They are basic scheme [7], compressed LU scheme [8], MKPS scheme [10] and polynomial LU scheme [11] respectively.

5.1 Eavesdropping Attack

In our assumption, a global eavesdropper is involved in the network so that all the traffic on the network will be immediately known by the adversary.

In the 4 classic schemes, the messages exchanged between nodes are only the public vectors of matrix \( U \). No matter how many public vectors an eavesdropper get, he/she can never gain any valuable information about shared keys. Similarly, the column vectors of \( U \) are still public and exchanged between nodes in our scheme. Besides, we also transmit random parameters \( p \) and \( q \), update messages \( B_{\text{new}} \) and revoke messages \( B_{\text{rvk}} \). Parameters \( p \) and \( q \) are encrypted by the simplest common bit-string which is unknown to eavesdropper. The update messages \( B_{\text{new}} \) and \( B_{\text{rvk}} \) are encrypted by pairwise key shared between node and its cluster, so that an adversary can’t decrypt them without the right keys. Generally, our scheme is secure against eavesdropping attack as the same as the 4 classic schemes.

5.2 Albrecht’s Attack in [19]

Albrecht et al. [19] launch attack on schemes which use “perturbation polynomials” to add “noise” to polynomial-based systems that offer information-theoretic security, in an attempt to increase the resilience threshold while maintaining efficiency. Their results cast doubt on the viability of using “perturbation polynomials” for designing secure cryptographic schemes. Albrecht’s attack based on the following two conditions:

1. Shared keys are established by perturbation polynomial \( s_i(x) = F(x_i, x) + b \times h(x) + (1 - b) \times g(x) \), where \( F(\cdot, \cdot) \) is a bivariate polynomial, \( h(\cdot) \) and \( g(\cdot) \) are two hash functions, \( x_i \) is a public point preloaded in node \( i \) and \( b \) equals to 0 or 1.
2. Point \( x \) is required to be an integer satisfying \( |h(x) - g(x)| < r \), so that \( s_i(x) \) can share a common bit-string with \( F(x_i, x) \).

Under condition (1) and (2), if an adversary has compromised \( n \) nodes, gained their preloaded point \( x_1, \ldots, x_n \) and polynomials \( s_1(x), \ldots, s_n(x) \), he/she can use error correction algorithm to recover the polynomial \( f^*(z) = F(z, x^*) + b \times h(x^*) + (1 - b) \times g(x^*) \) of an-
other uncompromised node \( v \) with public point \( x^* \). At that moment, the shared key between \( v \) and any other node \( v' \) is revealed since \( k_{v,v'} \) is a part of bit-string generated by 
\[ F(x', x^*) + b \times h(x^*) + (1 - b) \times g(x^*) \]

Dose Albrecht’s attack works for our proposed schemes which also use perturbation technology to provide security? The answer is no. In fact, Albrecht’s attack tries to use error correction algorithm to recover the method of calculating pairwise key (namely, the polynomial 
\[ F(x', x^*) + b \times h(x^*) + (1 - b) \times g(x^*) \] 
on the basis of that parameters of each nodes (namely, public point \( x \)) are already known. But our proposed scheme is complete opposite. In LU3D scheme, method of establish pairwise key is known by all participators while the parameters \( W, p \) and \( q \) are kept in secret. Obviously, Albrecht’s attack cannot reveal any of these parameters, so LU3D scheme is robust to Albrecht’s attack.

5.3 Node Capture Attack and LU Attack

5.3.1 Resilience analysis

Sensor nodes are deployed in hostile environment to gather information at the risk of being captured. When a sensor node is captured by an adversary, all information stored in it will be exposed including its key materials and the shared keys with other nodes. Besides, LU decomposition based schemes could be attack by LU attack only when some secret vectors are exposed. Hence, launch node capture attack is the first step of launching LU attack. Assume that there are \( x \) compromised nodes \( C_x = \{s_1, \ldots, s_x\} \) in the network, the other normal nodes compose node set \( UC \), and node \( s_i \in C_x, s_a, s_b \in UC \). When node \( s_i \) is compromised, the noise factor \( r \), function \( cm(\bullet, \bullet, \bullet) \), random number generator \( rd(\bullet, \bullet) \), and all \( W_i^{(u)} \) \((1 \leq u \leq t)\) are known by adversary. If noise vectors \((w_i^{(u)}) \) \((1 \leq u \leq t)\) is not exposed, adversary can’t calculate \((w_i^{(u)}) \). LU attack doesn’t work because of the effect of perturbation technology. The shared key \( k_{sa} \) between uncompromised nodes \( s_a \) and \( s_b \) is secure. Otherwise, \((w_i^{(u)}) \) is exposed, the adversary can launch a LU attack to calculate secret vector elements of the other uncompromised nodes such as \( s_a \) and \( s_b \).

Let \( e(A) \) be the event that \( A \) is exposed while \( !e(A) \) represents the event that \( A \) is not exposed. First, we consider the simplest situation, namely, total round number \( t = 1 \), the total KMG number \( t = 1 \), \( K, K', L, U, W, ns \) are all \( m \times m \) matrices. For a compromised node \( s_i \in C_x \), the adversary has got \( k_{i,j}' \) and can calculate \( k_{i,j} = R(k_{i,j}', r) \) where \( 1 \leq j \leq m \). In this case, \( L' = K'U^{-1} \) and \( n_{si} = W_i^{(u)} - L' \) are both calculable. Thus the probability of getting a right \( n_{si} \) for an adversary is equal to the probability of exposing \( k_{i,j} \), namely:

\[ A_{si} = Pr(e(ns_{si})|e(W_i^{(u)})) = \frac{1}{2^t}. \]  

When \( n_{si} \) is exposed, \( L' \) is exposed too. Assume that \( v \) vectors \( L' \) have been successfully got by adversary, which forms a set \( Q_v = \{L'_{i_1}, \ldots, L'_{i_v}\} \). Let the number of non-zero elements in vector \( A \) is denoted by \( \text{non}(A) \). The maximum number of non-zero elements of row vectors in \( Q_v \) is denoted by \( \max n(Q_v) \), namely \( \max n(Q_v) = \max \{\text{non}(L'_{i_1}), \ldots, \text{non}(L'_{i_v})\} \). It should be noticed that \( \max n(Q_v) \) reflects the adversary’s attack capability. So when an adversary get \( Q_v \), the number of calculable elements of \( L'_{i_1}, L'_{i_2} \in Q_v \), equals:
LUD-Based Key Predistribution Scheme for HSNs

\[
\begin{cases}
m & , \text{non}(L_i) < \max n(Q_j) \\
\max n(Q_j) & , \text{non}(L_i) \geq \max n(Q_j)
\end{cases}
\] (5)

Recalling the key component establishment process, the simplest common bit-string \(cbs_{a,b}\) between \(s_a\) and \(s_b\) is

\[
cbs_{a,b} = \text{cm}(W'_r U'_r, \text{length}(W'_r U'_r), r) = \text{cm}(L'_r U'_r, \text{length}(L'_r U'_r), r)
\]

(6)

So when one of \(L'_r\) and \(L'_s\) is exposed, \(cbs_{a,b}\) will be exposed too. In this case, the adversary must find \(p, q\) to break the key component \(k'_{a,b}\). In cluster \(g\), when \(y\) row vectors \(L'\) are exposed, if \(y \geq m - 1\), matrix \(K\) is exposed, the probability that \(cbs_{a,b}\) is secure equals 0. Otherwise,

\[
\Pr(\text{le}(cbs_{a,b}) | Q_x) = \Pr(\text{le}(L'_r U'_r) | Q_x)
\]

\[
= \Pr(\text{non}(L'_r) > \max n(Q_j) \land \text{non}(L'_s) > \max n(Q_j))
\]

\[
= \sum_{x=1}^{n} \{\Pr(\text{max n}(Q_j) = x) \times \Pr(\text{non}(L'_r) > x) \times \Pr(\text{non}(L'_s) > x | \text{non}(L'_r) > x)\}
\]

\[
= \sum_{y=1}^{n} \left(1 - \frac{m - z + 1}{m} \times \frac{m - z - 1}{m}\right)
\]

(7)

More generally, since \(k'_{a,b} = k'_{a,b}^{(1)} \parallel ... \parallel k'_{a,b}^{(t)}\), \(k'_{a,b}\) is secure until all key components are exposed. At the \(v\)th round, the security of \(k'_{a,b}\) depends on \(cbs_{a,b}^{(v)}\), \(p^{(v)}\) and \(q^{(v)}\). Let \(Pr(\text{NS}_x | C_v)\) be the probability that there are \(y\) exposed noise vectors which are belongs to the nodes in \(Q_x\). Since each node stores \(t\) NKM from \(t\) KMG, we can get

\[
\Pr(\text{NS}_x | C_v) = \sum_{x=1}^{t} \left(\frac{x!}{y!} \left(\frac{1}{2^r}\right)^y \left(1 - \frac{1}{2^r}\right)^{x - y}\right).
\]

(8)

Obviously, \(Pr(Q_v | \text{NS}_x) = 1\). Let \(Q_{(w)}\) be the number of exposed noise vectors which belongs to the \(w\)th KMG is \((w)_y\), then

\[
\Pr(\text{NS}_y | C_v) = \sum_{y=0}^{t} \left(\frac{y!}{y!} \left(\frac{1}{T}\right)^y \left(1 - \frac{1}{T}\right)^{x - y}\right)
\]

(9)

where \(T\) is the total number of KMGs. If the KMG which is chosen at the \(v\)th round is the \(w\)th one, the probability of breaking \(cbs_{a,b}^{(v)}\) is

\[
\Pr(\text{le}(cbs_{a,b}^{(v)}) | C_v) = \Pr(e(L'_r U'_r) | C_v)
\]

\[
= \sum_{v=1}^{t} \Pr(\text{NS}_v | C_v) \Pr(Q_{(w)} | C_v) \Pr(\text{le}(L'_r U'_r) | Q_{(w)})
\]

(10)

When \(cbs_{a,b}^{(w)}\) is exposed, there are two cases of breaking \(k_{a,b}^{(w)}\) which should be taken into
consideration:

(1) The establishment process of $k^{(v)}_{a,b}$ is not finished.
In this case, the adversary can decrypt message \( \{ E_g^{(v)}(p^{(v)}, q^{(v)}) \} \) and get $k^{(v)}_{a,b} = cm (cbs^{(v)}_{a,b}, p^{(v)}) - r, q^{(v)} - r)$. So $Pr(e(k^{(v)}_{a,b}) | C_v) = Pr(e(cbs^{(v)}_{a,b}) | C_v)$.

(2) The establishment process of $k^{(v)}_{a,b}$ is finished.
In this case, adversary have to guess what is the value of $p^{(v)}, q^{(v)}$ respectively. Let $E[A]$ be the mean value of $A$. Since parameters $p$ and $q$ have relationship:

\[
r \leq q^{(v)} \leq \text{length}(k^{(v)}_{a,b} - 1) < \text{length}(k^{(v)}_{a,b}) \leq p^{(v)} \leq \text{Len} + q^{(v)}
\]

cbs^{(v)}_{a,b}$ and $r$ are known, so

\[
Pr(e(p^{(v)}, q^{(v)}) | cbs^{(v)}_{a,b}) = Pr(e(p^{(v)}) | e(q^{(v)})Pr(e(q^{(v)}) | cbs^{(v)}_{a,b})
\]

\[
Pr(e(q^{(v)}) | cbs^{(v)}_{a,b}) = \frac{1}{\text{length}(cbs^{(v)}_{a,b})} \approx \frac{1}{E[\text{length}(k^{(v)}_{a,b})] - r}
\]

\[
Pr(e(p^{(v)}) | e(q^{(v)}) = \frac{1}{\text{Len} + \text{length}(cbs^{(v)}_{a,b}) - 1 - \text{length}(cbs^{(v)}_{a,b})} = \frac{1}{\text{Len} - 1}
\]

Then we get

\[
Pr(e(k^{(v)}_{a,b}) | C_v) = Pr(e(cbs^{(v)}_{a,b}) | C_v)Pr(e(p^{(v)}, q^{(v)}) | e(cbs^{(v)}_{a,b}), C_v)
\]

\[
= Pr(e(cbs^{(v)}_{a,b}) | C_v)Pr(e(p^{(v)}) | e(q^{(v)})Pr(e(q^{(v)}) | e(cbs^{(v)}_{a,b}))
\]

\[
\approx Pr(e(cbs^{(v)}_{a,b}) | C_v) \times \frac{1}{\text{Len} - 1} \times \frac{1}{E[\text{length}(k^{(v)}_{a,b})] - r}
\]

Generally speaking, the probability of breaking $k^{(v)}_{a,b}$ is

\[
Pr(e(k^{(v)}_{a,b}) | C_v) \approx \sum_{i=0}^{\frac{\text{Len}}{2}} \left( \frac{\zeta}{i} \right)^i \left( e(cbs^{(v)}_{a,b}) | C_v \right)^i
\]

5.3.2 Comparative analysis

Though the 4 classic schemes choose different ways to process LU decomposition process and results, they all directly adopt the product of $L'$ and $U'$ as the shard key between any two nodes, so they share the same resilience which is greatly affected by LU attack. Comparing to the 4 classic schemes, our scheme has clear superiority in terms of resilience against node capture attack and LU attack. For clarity, we mark the curve of the 4 classic schemes with the label “other4”, and ourt(Len, EL, $\zeta$, r) denotes that the parameters of our scheme contains required key length Len, the average length of all keys EL, the total number of rounds $\zeta$, the noise factor $r$, and $i$ which equals to 1 or 0. Parameter $i = 0$ represents that when the attack is launched, no key component has been
established and \( i = 1 \) represents all key establishment process have finished. When each cluster has \( m = 30 \) members and the total number of KMG equals 40, the analytical results is shown in Fig. 3 where \( x \) is the number of compromised nodes, and \( Pr \) is the probability that the shared key of two uncompromised nodes is exposed. As we know, the higher the \( Pr \), the weaker the resilience of scheme against node capture attack, so does LU attack.

For our scheme and the 4 classic schemes, \( Pr \) increases with \( x \). But the \( Pr \)s of the 4 classic schemes are always bigger than 0.7 while that of our scheme is always smaller than 0.1. The curve with label \( our0(10,5,1,3) \) shows the resilience trend of our scheme when the attack launched before none of the key components is established. The curve with label \( our1(10,5,1,3) \) shows the resilience trend of our scheme when the attack launched after all key components are established. Obviously, no matter when the attack happens, our scheme can provide much higher security than the 4 classic schemes. If the attack is launched after key establishment, our scheme can achieve wonderful security.

Figs. 4 and 5 show how key length, the total number of rounds and the noise factor influence our scheme’s security. It could be verified that \( Len \) and \( EL \) won’t influence the resilience when attack launched before the key establishment phase. Even when the attack launched after the key establishment phase, their influence on resilience is very small. However, the influence of \( \xi \) and \( r \) are both very obvious and \( Len, EL < \xi < r \).
6. PERFORMANCE ANALYSIS

6.1 Efficiency of LU3D Scheme

For simplicity, we evaluate the efficiency of LU3D in terms of number of rounds, number of parameters, mean size of parameters and cost of per bit information. Here, parameters include elements in matrix, IDs, node indexes, random integers, etc. Assume that the mean size of parameters is \( l \) bits, the dimension of each matrix is \( m \).

**Storage Overhead**

Comparing with normal matrix, storing a triangular matrix can save a lot of memory. For example, the method adopted in compressed LU scheme [8] to store one row of \( L \) and one column of \( U \) is to separate each row of \( L \) and each column of \( U \) into two parts: nonzero-element part and zero-element part (might be absent). The stored values are each element in the nonzero-element part and one extra value for specifying the number of zeros in the zero-element part. We adopt the same way to store triangular matrix, since matrix \( W \) is not a triangular matrix any longer, only matrix \( U \) can be completely compressed in this paper. Hence, the average memory overhead of each normal node equals

\[
C_m = \left( \frac{m + (m + 1)}{2} + 1 \right)(t+5)l = \frac{3}{2}m(m+1)+5l
\]

where \( t \) is the number of KMGs assigned to current cluster.

**Communication Overhead**

We consider the communication efficiency in terms of number of rounds, message size transmitted between two nodes who try to establish pairwise key, the cost of receiving per bit data, denoted by \( r \), and the cost of sending per bit data, denoted by \( s \). Usually, \( r < s \). Besides, we also assume that to establish a pairwise key needs to establish key component \( \xi \) times. Thus, the average communication overhead of each normal node for key establishment is:

\[
C_{comm} = \frac{1}{2} \left( \left( 2 \times \left( \frac{1}{2} + m \right) + 3 \right)\xi(r+s) = \frac{1}{2}(m+4)\xi(r+s). \right)
\]

When new node tries to join in a cluster, extension messages should be transmitted to relevant sensor nodes. Each message contains \( 2 + t \) \( l \)-bit parameters. There are \( m \) messages need to be transmitted by cluster head. Thus, the average communication overhead of each normal node for handling the addition of a new node is \( C_{comm, join} = (t+2)lr \), and the communication overhead of each cluster head for handling the addition of a new node is \( C_{comm, join} = (2 + mt)lr + (2 + t)(m-1)ls \).

When revoking a normal node, only one message with 3 parameters needs to be transmitted through the network. Hence, the communication overhead of normal nodes and cluster heads are \( C_{comm, leave} = 3lr \) and \( C_{comm, leave} = 3lr \) respectively. When a cluster head is leaving, the computational overhead will become \( C_{comm, leave} = (t + 5)lr \) and \( C_{comm, leave} = \)
\[ C_{\text{comm,join}} + 3ls = (2 + mt)lr + (tm + 2m - t - 1)ls \] respectively.

**Computational Overhead**

Let the computational overhead for encryption or decryption be \( b \), for vector multiplication be \( c \), and for getting substring is \( d \). Then the computational overhead of each normal node for key establishment is:

\[ C_{\text{cpt}} = \xi(b + c + 2d). \]  

(18)

When a node tries to join in a cluster, the cluster head needs to decrypt one message and to encrypt \( m-1 \) messages which will be decrypted by relevant nodes. Hence the computational overhead of cluster head is \( C_{\text{cpt,join}} = mb \) and that of normal node is \( C_{\text{cpt,join}} = b \).

When a normal node is leaving, cluster head will perform encryption \( m-1 \) times and cost \( C_{\text{cpt,leave}} = (m - 1)b \). Meanwhile, normal nodes will only perform decryption once and cost \( C_{\text{cpt,leave}} = b \). When a cluster head is leaving, the computational overhead will become \( C_{\text{cpt,leave}} = C_{\text{cpt,join}} + mb = 2mb \) and \( C_{\text{cpt,leave}} = C_{\text{cpt,join}} + b = 2b \) respectively.

**6.2 Comparative Analysis**

Our normal node model is based on Micaz mote developed by Crossbow Technology Inc while some MIB520 motes from the same company are playing the role of cluster head. In most application, living longer is more important than running faster, so we choose the lightest TableLookupAES [21] as our default symmetric cipher algorithm. According to [20, 21], some involved empirical parameters are given in Table 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES</td>
<td>7.211(\mu)J/bit</td>
<td>clock cycle</td>
<td>3.5 nJ/clk</td>
</tr>
<tr>
<td>scalar multiplication</td>
<td>10 clk</td>
<td>transmission circuit</td>
<td>1.066 J/bit</td>
</tr>
<tr>
<td>battery</td>
<td>1.5 V</td>
<td>receive circuit</td>
<td>0.533(\mu)J/bit</td>
</tr>
<tr>
<td></td>
<td>2500 mAH</td>
<td>energy consumption for sleep</td>
<td>120 pJ/bit</td>
</tr>
</tbody>
</table>

Obviously, a normal AA battery can provide energy \(1.5 \times 2.5 \times 3600 = 13500 \)J, thus a normal MICAz mote equipped two AA batteries will provide energy 27000J. The length of key of AES is usually 128bits, 192bits or 256bits. The memory of MICAz mote is \( M = 10.48576 \times 10^5 \) bits. Take AES-128 for example, it leads to the comparison result of storage overhead and supported network size of each scheme as shown in Table 3, where \( d \) represents the degree of polynomials.

When schemes in [7-11] try to generate a length-fix key, they only call calculation once. Hence, their average parameter lengths are relatively fixed. Different from them, LU3D scheme can reduce its average parameter length by calling more calculation rounds. As Table 3 shows, when the average parameter length are same and \( t = 1 \), the storage overhead of LU3D scheme is only bigger than compressed LU scheme. Though increasing the number of preloaded KMG will also increase required storage, reducing average parameter length will help us to recover these overheads.
Table 3. Comparison of storage overhead and supported network size.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Storage Overhead</th>
<th>Parameter l</th>
<th>Supported Network Size (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic LU scheme</td>
<td>$2ml$</td>
<td>/</td>
<td>$\leq 4096$ (whole network)</td>
</tr>
<tr>
<td>compressed LU scheme</td>
<td>$(m + 3)l$</td>
<td>/</td>
<td>$\leq 8189$ (whole network)</td>
</tr>
<tr>
<td>MKPS scheme</td>
<td>$3ml$</td>
<td>128</td>
<td>$\leq 2730$ (whole network)</td>
</tr>
<tr>
<td>polynomial LU scheme</td>
<td>$(d + 1)(m + 1)l^2$ + $2ceil(\log_2(m - 1))$</td>
<td>$d=3$</td>
<td>$\leq 4094$ (whole network)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$d=9$</td>
</tr>
<tr>
<td>LU3D scheme</td>
<td>$\frac{3lt}{2} (m + 1) + 5l$</td>
<td>$t=1$</td>
<td>$\leq 5457$ (per cluster)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$t=2$</td>
<td>$\leq 2728$ (per cluster)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$t=1$</td>
<td>$\leq 10918$ (per cluster)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$t=2$</td>
<td>$\leq 5458$ (per cluster)</td>
</tr>
</tbody>
</table>

Table 4. Computational and communication overhead of key establishment.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Computational Cost</th>
<th>Communication Cost</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic LU scheme</td>
<td>$c$</td>
<td>$(m + 1)l(r + s)/2$</td>
<td>NO</td>
</tr>
<tr>
<td>compressed LU scheme</td>
<td>$c$</td>
<td>$(m + 1)l(r + s)/2$</td>
<td>NO</td>
</tr>
<tr>
<td>MKPS scheme</td>
<td>$2c$</td>
<td>$(m + 1)l(r + s)/2$</td>
<td>NO</td>
</tr>
<tr>
<td>polynomial LU scheme</td>
<td>$c$</td>
<td>$(m + 1)l(r + s)/2$</td>
<td>NO</td>
</tr>
<tr>
<td>LU3D scheme</td>
<td>$\xi(b + c + 2d)$</td>
<td>$(m + 4)\xi(r + s)/2$</td>
<td>YES</td>
</tr>
</tbody>
</table>

In Table 4, we summarize the computation and communication requirements of our protocol and other 4 schemes for key establishment. Schemes in [7, 8, 10, 11] share a same communication overhead, because they all establish key by exchanging a $m$-dimensional vector, but MKPS scheme costs twice computational overhead than the others for another vector multiplication. Comparing with this 4 schemes, LU3D scheme needs a higher computational and communication overhead which is closely related to rounds number $\xi$. In that case, we have to consider that whether LU3D scheme is efficient and feasible for a sensor network. To answer this question, we perform a simulation based on MICAz mote and all analysis we mentioned before in this paper.

As we know, the level of computational overhead and communication overhead eventually manifest as network lifetime, for instance, the more computational and communication overhead are, the shorter the network lifetime will be. Hence, we evaluate the lifetime of schemes by implementing LU3D schemes in simulation tool OMNET++ instead of measuring computational and communication overhead respectively. The simulation process works as follows:

At the first beginning, we construct a network topology as shown in Fig. 6. Each node is equipped with key materials, and the six nodes in the middle are default cluster heads which are powerful enough so that they can keep on providing services until the
last member node in its cluster is depleted or there is no member left. When only the cluster head is active in a cluster, we call this cluster is dead and no more nodes can be added to it. When all clusters are dead, the whole network is dead. In order to ensure there is an end of our simulation, we start this simulation by performing key establishment process in each cluster, and keep on adding node which is deleted from other clusters to current cluster instead of constructing new nodes. There is a random interval between two update events, adding or deleting a node, in a cluster, and the target node, the target cluster are randomly chosen too. So during our simulation, the number of active nodes is always equal to or less than the initial nodes number.

Parameters we adopted in simulation still come from Table 2. By setting the average interval between two update events in one cluster to 1s, 1min, 10min and 1h respectively, setting the average parameter length as 128bits, we get some results as shown in Table 5, where “AverageUI” is average update interval, “AverageR” is average number of rounds, “MinL” is the lifetime of the first dead node, “MaxL” is the lifetime of the last dead node and “AverageL” is the average lifetime of all nodes. As we see, LU3D scheme is feasible and efficient for normal sensor network applications. If there are some extra new nodes are provided or cluster heads can wait for node arrival when there is no active member left, LU3D scheme can provide a better performance.

<table>
<thead>
<tr>
<th>AverageUI</th>
<th>AverageR</th>
<th>MinL(days)</th>
<th>MaxL(days)</th>
<th>AverageL(days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1s</td>
<td>6.6898</td>
<td>15.8373</td>
<td>69.3464</td>
<td>35.8322</td>
</tr>
<tr>
<td>1min</td>
<td>5.2134</td>
<td>380.8722</td>
<td>8619.3339</td>
<td>1898.3921</td>
</tr>
<tr>
<td>10min</td>
<td>4.3933</td>
<td>8278.6451</td>
<td>29320.0564</td>
<td>19430.8296</td>
</tr>
<tr>
<td>1h</td>
<td>5.6621</td>
<td>41732.3432</td>
<td>105269.2083</td>
<td>69832.3475</td>
</tr>
</tbody>
</table>
7. CONCLUSION

Existing LU decomposition based schemes are enormously threatened by LU attack. In order to resist LU attack, a scalable LU decomposition based schemes, called LU3D, is proposed in this paper. By introducing perturbation technology, LU3D scheme provides a way to establish pairwise keys for clustered sensor networks. Perturbation is used to generate common information while random common bit-string truncation is used to establish key component. Furthermore, it also takes advantage of LU attack principle to quickly extend matrices and provide scalability. From both of theoretical and experimental aspects, we figure out how the parameters influence LU3D scheme and prove that it is robust to eavesdropping attack, Albrecht’s attack [19], node capture attack and LU attack. By comparing LU3D scheme with 4 existing schemes mentioned in [7, 8, 10, 11] in terms of storage overhead, computational overhead and communication overhead, we verify that LU3D scheme is feasible to be used in sensor networks.

Although LU3D scheme is much better than existing schemes for providing much higher security, the overhead of multi-rounds calculation is a problem which can be improved. Besides, the way cluster head adopts to transmit update message is unicast which will bring extra overhead. If we provide an authentication method for member nodes to check the source of messages, cluster head can broadcast update messages to save some energy. In the future, we will focus on further reducing the computational and communication overhead of our scheme, and designing a proper authentication scheme with key materials assigned to nodes in this paper.

REFERENCES

LUD-BASED KEY PREDISTRIBUTION SCHEME FOR HSNs


Chunguang Ma (马春光), born in 1974, got his Ph.D. degree in Cryptography from Beijing University of Posts and Telecommunications in 2005. He is a Professor at College of Computer Science and Technology, Harbin Engineering University. His current research interests include cryptology, information security, network coding, Ad hoc & WSN. Dr. Ma is a senior member of China Computer Federation, a member of council of Chinese Association for Cryptologic Research.

Xiaorui Zhong (钟晓睿), born in 1987, a Ph.D. candidate of Computer Science and Technology, Harbin Engineering University, China since 2010. She has taken a successive undergraduate, postgraduate and doctoral program since 2005. Her current research interests include security and performance evaluation in wireless networks. She is also a student member of China Computer Federation.