ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.5, May 2007, pp.1232−1245
DOI: 10.1360/jos181232

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

传感器网络中一种可靠的对密钥更新方案*

温 蜜[1+], 陈克非[1,2], 郑燕飞[1,2], 李 晖[1]

[1](上海交通大学 计算机科学与工程系,上海 200240)

[2](上海交通大学 信息安全工程学院,上海 200240)

A Reliable Pairwise Key-Updating Scheme for Sensor Networks

WEN Mi[1+], CHEN Ke-Fei[1,2], ZHENG Yan-Fei[1,2], LI Hui[1]

[1](Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200240, China)

[2](School of Information Security and Engineering, Shanghai Jiaotong University, Shanghai 200240, China)

+ Corresponding author: Phn: +86-21-34205539, Fax: +86-21-34204405, E-mail: superwm@sjtu.edu.cn, http://cis.sjtu.edu.cn

Wen M, Chen KF, Zheng YF, Li H. A reliable pairwise key-updating scheme for sensor networks. *Journal of Software*, 2007,18(5):1232−1245. http://www.jos.org.cn/1000-9825/18/1232.htm

Abstract: This paper proposes a reliable pairwise key-updating (RPKU) scheme for clustered WSNs via predistribution and local collaboration approaches. Based on the modified version of Blom's matrix construction, this scheme can extend and shrink the pairwise keys in WSNs with the network topology changes. This scheme also presents a hierarchical key distribution method in the clustered WSNs, guaranteeing that any pair of neighboring nodes can find a common secret key between themselves. Comparison and simulation results show that the proposed scheme outperforms most of the existing pairwise key establishment schemes in terms of network security, key connectivity and scalability.

Key words: wireless sensor network; key distribution; key-updating; Blom's matrix; security

摘 要: 提出了一种基于预分发和协作的可靠的对密钥更新方案 RPKU(reliable pairwise key-updating).借助于一种改进的 Blom 密钥矩阵构造方法,该方案能够随着网络的动态变化而动态伸缩各个节点的密钥信息,从而解决了由于节点被攻击所导致的密钥泄漏和密钥连通性下降等问题.该方案还提出了一种基于分簇型传感器网络结构的密钥预分发方法,使得任意两个相邻节点间都能建立一个对密钥.仿真结果表明,与已有的密钥方案相比,该方案在安全性、密钥连通性和扩展性等方面都具有明显的优势.

关键词: 无线传感器网络;密钥分发;密钥更新;Blom 密钥矩阵;安全

中图法分类号: TP393 文献标识码: A

1 Introduction

Wireless sensor networks (WSNs) increasingly become viable solutions to many challenging problems for both

* Supported by the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No.20050248043 (高等学校博士学科点专项科研基金)

Received 2006-12-29; Accepted 2007-02-14

military and civilian applications, including target tracking, battlefield surveillance, intruder detection and scientific exploration[1]. However, deploying sensors without security in mind has often proved to be dangerous in hostile environments. To prevent the malicious node from impersonating good nodes for spreading misleading information intentionally, secret keys should be used to achieve data confidentiality, integrity and authentication between communicating parties. Most of the well known secret key management schemes such as key distribution centers schemes and public key cryptography schemes require significant computation and communication overhead, and thus cannot be readily applied to WSNs. In contrast, the key pre-distribution (KPD) approach is an attractive solution to this problem, i.e., pre-installing a limited number of keys in sensor nodes prior to actual deployment[2−8]; after the deployment, if two neighboring nodes have some common keys, they can setup a secure link by the shared keys.

Furthermore, numerous conditions may give rise to the need for adding a node in the network or evicting a node from the network. For example, due to limited energy capacity of batteries, the function lifetime of sensor networks in general is longer than the operational lifetime of single nodes. Therefore, to keep the network working, we need to add new nodes in time, without compromising the network security. Additionally, wireless sensors are not tamper resistant due to their low cost. Thus, the adversary may physically capture some sensors to compromise their stored sensitive data and communication keys. This serious attack is known as node capture attack, which makes the node's operation become under control of the adversary. This will be subversive. Therefore, the compromised nodes should be identified and evicted, and secure addition of new nodes would also be useful. To deal with node addition and node eviction, the innocent nodes should update their pairwise keys to establish new keys with the new ones and prevent the adversary from utilizing the captured keys.

Recently, the pairwise key predistribution problem has been extensively studied in the context of wireless sensor networks. However, most of the existing key predistribution schemes[2−8] just preload the keys to the new nodes from the same key pool, and the network security offered by these schemes decreases with time. They suffer because of the repeated use of fixed key information. The capture of each node increases the fraction of keys known by the adversary. When a certain number of these nodes are captured, the adversary has enough key information to compromise a large number of links making the network ineffective. The addition of new nodes to the network with keys from the same key pool will not help because the keys in the new nodes are already compromised. Although the centralized revocation approach proposed by Eschenauer[5] and the distributed revocation approach proposed by Chan[6] can improve the network security by removing captured node's keys in the rest sensor nodes, the key connectivity between sensor nodes decreases consequently.

The main contribution of this paper is a novel solution, RPKU (reliable pairwise key-updating), to the problem of pairwise key updating in WSNs, without compromising the network security and key connectivity. The solution is based on a modified version of Blom's matrix key construction, which can dynamically extend or shrink the key information to support the substantial increase and decrease in the size of the network even after deployment. We show that RPKU can perform better by taking advantage of more powerful network architecture: clustered WSNs, which consists of three types of nodes in the descending order of capabilities: (a) base stations, (b) cluster heads, and (c) sensor nodes. Base stations are many orders of magnitude more powerful than sensor nodes and cluster heads. Base stations always perform costly operations on behalf of sensor nodes and manage the network. Nodes with better resources, named as cluster heads (CH), may be used to collect and merge local data from sensor nodes and send it to base station. Sensor nodes are deployed around the neighborhood of the cluster heads.

The rest of this paper is organized as follows. Section 2 overviews related work. Section 3 provides a modified version of Blom's matrix construction. In Section 4, we describe the key updating approach. Section 5 discusses our

scheme. Security analysis and performance evaluation are presented in Section 6. Finally, we conclude in Section 7.

2 Related Work

To the best of our knowledge, RPKU is the first attempt to address the problem of pairwise key updating in WSNs without compromising the network security and key connectivity. Specifically, RPKU leverages a hierarchical network infrastructure to implement efficient pairwise key initialization and management, having as goal the tradeoff among security, connectivity, and network cost.

Up to now, there are three kinds of KPDs employed in the scenarios of sensor networks, i.e. deterministic KPDs[2−4] (DKPD), probabilistic KPDs[5,6] (PKPD) and hybrid KPDs[7,8,10,15,18] (HKPD). Deterministic KPDs can certainly guarantee the connectivity and the exclusive pairwise key between any pair of nodes in a network. A naive solution is the full pairwise approach, which lets each node in the network share a unique pairwise key with every other node in the network. That means each sensor in the network with size should carry secret pairwise keys. This solution has perfect node compromise resilience. But, it has high storage overhead for each node, and due to its limited memory, if N is large it becomes impractical or even impossible to support large-scale sensor networks. Blom[2], Blundo[3], and Choil[4] propose another major class of deterministic KPDs, they supply each node with a relatively small amount of secret key information from a large network key space, based on which each pair of nodes can derive their pairwise keys. This class of solutions still suffers from the N memory cost problem and they are not perfectly resilient against node compromise attack.

Probabilistic KPDs cannot guarantee the connectivity and the exclusive key pairing between any pair of nodes in a network. The reason is that, in the random KPDs, each node is randomly supplied with some secret keys from a large key pool, and then it tries to find some direct or indirect common keys with its neighbors in certain probability. Since random key distribution is probabilistic in nature, some critical neighboring nodes could not establish pairwise key successfully[10]. Moreover, due to the case that several different pairs of sensor nodes may use the same key, some sensor nodes' compromise may affect the communication links in the rest of the network. Examples include schemes proposed by Eschenauer and Gligor[5], Chan[6].

Most of the hybrid KPDs are the combination of the deterministic KPDs and the basic key pool idea used in probabilistic KPDs, such as schemes proposed by Du[7], Liu[8] etc., aiming at the tradeoff among compromise resilience of probabilistic KPDs and scalability of deterministic KPDs. Some other hybrid KPDs use the deployment knowledge to improve the key connectivity and scalability of key distribution schemes[10,15,18]. In general, the existing pairwise key solutions suffer because they reuse the information for key distribution, lacking of pairwise key updating approaches for supporting the network changes.

3 Modified Version of Blom's Key Distribution Approach

Blom proposed a key distribution approach[2], which allows any pair of nodes in a network to be able to find a pairwise secret key. As long as no more than λ nodes are compromised, the network is perfectly secure. Du[7] gave an extended version to construct the multiple key spaces. To improve the scalability, we modify the Blom's symmetric matrix construction in this section. We briefly describe how our modified version of Blom's key distribution approach works as follows.

During a cluster pre-deployment process, the base station (BS) in WSNs (acting as a trusted dealer) first computes a $n*n$ matrix B over a finite field $GF(q)$, B is considered as public information, q is a prime, and $q<n$. One example of such a matrix is a Vandermonde matrix whose element $b_{ij}=(g^j)^i \bmod q$, where g is the primitive nonzero element of $GF(q)$ and g^j is the jth column seed. That means

$$B=\begin{bmatrix} 1 & 1 & 1 & ... & 1 \\ g & g^2 & g^3 & ... & g^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g^{n-1} & (g^2)^{n-1} & (g^3)^{n-1} & ... & (g^n)^{n-1} \end{bmatrix}.$$

This construction requires that $n^2<\phi(q)$ i.e., $n^2<q-1$. Since B is a Vandermonde matrix, it can be proved that the n columns are linearly independent when $g,g^2,g^3,\ldots,g^n$ are all distinct[9].

Next, the BS generates n row seeds $s_i,i=1,\ldots,n$, where s_i is the random prime number of $GF(q)$ and only known to the cluster key manager (CKM). And then BS creates a random $n*n$ matrix D over $GF(q)$. Each row of the D is composed of hash values of the row seeds. Differing from the construction of matrix B, the elements in the symmetric matrix D are generated as follows

for (i=1;$i\le n$;i++)

  for (j=1;$j\le n$;j++)

{if ($i>j$),$d_{ij}=H^i(s_j)$;else $d_{ij}=H^j(s_i)$;}

where d_{ij} is the element in D. An example of matrix D with size 3*3 is shown as follows

$$\begin{bmatrix} H^1(s_1) & H^2(s_1) & H^3(s_1) \\ H^2(s_1) & H^2(s_2) & H^3(s_2) \\ H^3(s_1) & H^3(s_2) & H^3(s_3) \end{bmatrix}.$$

At last, the BS computes a $n*n$ matrix $A=(DB)^T$, where T indicates a transposition of the matrix. The elements in matrix A are denoted as a_{ij}, where $a_{ij}=\sum_{\beta=1}^{n} d_{j\beta}b_{\beta i}$. The matrix B is public while the matrix D is kept secret by the base station. Since D is symmetric, the key matrix $K=AB$ can be written as

$$K=(DB)^TB=B^TD^TB=B^TDB=(AB)^T=K^T.$$

Thus K is also a symmetric matrix and $K_{ij}=K_{ji}$, where K_{ij} is the element of K at the ith row and jth column. We take K_{ij} (or K_{ji}) as the pairwise key between node i and node j. To carry out the above computation, nodes i and j should be able to compute K_{ij} and K_{ji} respectively. This can be easily achieved using the following key predistribution procedure, for node i:

1) Store the ith row of matrix A at node i, denoted as $r_i(A)$, i.e., $r_i(A)=[a_{ij}]$, for j=1,…,n,

2) Store the ith column seed g^i of matrix B at node i.

Finally, when nodes i and j need to find the pairwise key between them, they first exchange their column seeds of B (since B is the public information, it can be sent in plaintext), and then they can compute K_{ij} (or K_{ji}) respectively as: $K_{ij}=\sum_{\beta=1}^{n} a_{i\beta}b_{\beta j}$. It can be proved that the above scheme is *n-secure* because all the columns in B are linearly independent. The *n-secure* property guarantees the exclusivity of the pairwise keys in the cluster.

4 Key Updating in RPKU: Our Solution

We propose a scheme for pairwise key updating in the clustered WSNs. The scheme leverages a hierarchical network infrastructure to implement a set of scalable functions for pairwise key initialization and management. The scheme assumes all nodes are pre-initialized with some key information before deployment. We also assume that nodes are loosely synchronized, and the cluster key manager (CKM) is responsible for distributing key information to cluster members. An example of the key updating will be given in Section 4.

4.1 Node addition and key information extension

The main issue in this procedure is the key information extension problem. Here we identify a real-time generation technique to solve this problem.

Real-Time generation (**RTG**): In this technique, neighbors can collaborate with each other to effectively protect and appropriately use the preloaded keys to extend their key information. For convenience, we denote the new node as node n+1. The process is presented in Fig.1, and it includes the following two tasks:

1. The key information for node n+1 (i.e.$r_{n+1}(A)$, and the n+1th column seed of matrix B) should be generated and preloaded into its memory. First, the n+1th row of matrix D should be generated. Then, according to the modified construction of Blom's matrix in Section 3, the n+1th row of matrix A will be worked out.

2. The key information, especially the rows of matrix A in the innocent nodes, should be extended. Thus, firstly, the rows of the original matrix D, $r_i(D)$ j=1,…,n, should be extended, i.e. the n+1th column of matrix D should be generated. Since D is a symmetric matrix, the elements in the n+1th column of matrix D equal to the elements in the n+1th row of matrix D. Thus $d_{i(n+1)}=d_{(n+1)i}$, and $d_{(n+1)i}$ is generated in the first step. Secondly, the rows of matrix A in the innocent nodes will be extended in terms of the method described in Section 3.

The drawback of the real-time generation is that it introduces additional communication and computation overhead for each innocent node, which does not appear in reuse approach in the previous schemes.

Algorithm 1. Real-Time key information extension
1) For the base station (BS),
it generates a new column seed g^{n+1} and stores it in node n+1.
it generates a row seed s_{n+1} of matrix D and stores it in CKM.
2) For the CKM,
it computes the (n+1)th row of matrix D as:
for j=1 to n+1 **do**
  $d_{(n+1)j}=H^{n+1}(s_j)$
end for
Then, it distributes the elements $d_{i(n+1)}$ (i=1,…,n) to the innocent nodes.
Next, it constructs $r_{n+1}(A)$ of matrix A for the node n+1 as:
for i=1 to n+1 **do**
  $b_{i(n+1)}=(g^{n+1})^{i-1}$; $a_{(n+1)i}=\sum_{j=1}^{n+1}d_{ij}b_{i(n+1)}$
end for
3) Upon receiving the elements $d_{i(n+1)}$ from CKM, each innocent node i extends its $r_i(A)$ as:
for j=1 to n+1 **do**
  $b_{ji}=(g^i)^j$;
end for
for j=1 to n **do**
  $a_{ij}=a_{ij}+d_{j(n+1)}b_{(n+1)i}$
end for
$a_{i(n+1)}=\sum_{j=1}^{n+1}d_{(n+1)j}b_{ji}$

Fig.1 Real-Time key information extension

4.2 Node eviction and key information shrinking

We assume that the compromised nodes and the energy-exhausted nodes can eventually be detected by most of its neighbors within a certain time period. To achieve this purpose, the collaborative intruder detection[13], remote code attestation[14] and schemes for residual energy scan like[16] may be used. After detection, the innocent nodes should update their key information to prevent the adversary from utilizing the captured keys. The main issue in this procedure is the key information shrinking. Also, two types of techniques can address this problem.

Regeneration: in this approach, the BS first regenerates all of the key information for the remaining nodes, including row seeds of matrix D, column seeds of matrix B and row elements of matrix A. Then, the new key information will be distributed to the innocent nodes with the help of the previous secure pairwise keys. Finally, the remaining sensor nodes can establish new pairwise key without influencing the rest network security. The

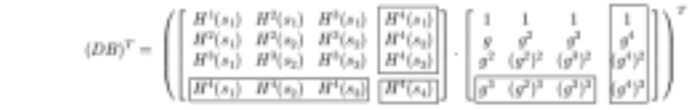
regenerating process is the same as the description in Section 3. The drawback of this approach is that it introduces substantive communication and computation overhead.

Real-Time Shrink (RTS): This technique needs the local collaboration of the neighboring innocent nodes to shrink the rows of matrix A. For clarity, we denote the evicted node as node k. The process is demonstrated in Fig.2.

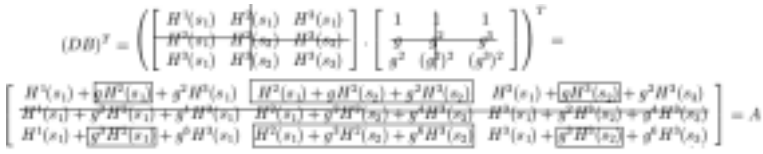
Algorithm 2. Real-Time key information shrinking
1) For the base station (BS),
it announces that the column seed g^k and the row seed s_k are both invalid.
2) For the CKM,
it computes the elements d_{ki} ($i<k$) as follows and distributes them to the innocent nodes secretly:
for j=1 to k–1 **do**
  $d_{ij}=H^k(s_i)$
end for
3) Each remaining node i deletes the elements a_{ik} from its row $r_i(A)$ and computes the elements d_{ki} ($i\geq k$) as:
  $d_{ij}=H^i(s_k)$, then evolve the other elements in $r_i(A)$ as follows:
for j=1 to n **do**
  if $i\neq j$, then
    $a_{ij}=a_{ij}-d_{ik}*b_{ki}$, where $d_{ik}=d_{ki}$
  end if
end for

Fig.2 Real-Time key information shrink

Note that, in terms of the matrix D's construction described in Section 3, d_{ki} ($i<k$) is independent of the row seed s_k, so node i ($i<k$) cannot generate d_{ki} by itself. In order to facilitate the RTS process, CKM should generate d_{ki} ($i<k$) and distribute it to node i ($i<k$) in Fig.2. In contrast, node i ($i\geq k$) can generate d_{ki} from the row seed s_k (s_k is public because it was announced invalid by BS).



(a) There is a node addition



(b) There is a node eviction in WSNs

Fig.3 Examples of key updating when (a) there is a node addition or (b) there is a node eviction in WSNs

Though this real-time key-updating mechanism also introduces additional computation overhead for each remaining node, it is much smaller than that in the regeneration approach above, and is also smaller than that in the key revocation approach in Refs.[5,6]. The performance comparison will be given in Section 6.

5 The Key Distribution Scheme in RPKU: Our Scheme

5.1 System model

In this paper, we consider a large-scale WSN consisting of some cluster heads and numerous sensor nodes which are grouped into clusters. The clusters can be formed based on various criteria such as capabilities, location, communication range, etc.[17]. Each cluster head (CH), acting as a CKM, controls a cluster and is responsible for assigning public group information and distributing some keying information to its cluster members. Each CH is

assumed to be reachable to all sensors in its cluster, either directly or by multihop. We also assume that each node is innocent before deployment, and cannot be compromised during the first several minutes after deployment since compromising a node takes some time. The network architecture is depicted in Fig.4. We require that a fraction of nodes from one cluster be interleaved with those from neighboring clusters. Due to the large distances between clusters, communication between nodes in different clusters can only be performed via nodes in these overlapping areas. This assumption is generally true, since the sensor nodes are deployed by helicopters at different deployment points in a targeted field.

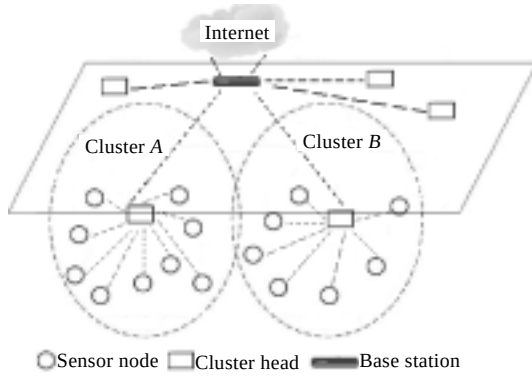


Fig.4 Clustered wireless sensor network architecture

In this paper, we mainly consider an adversary that tries to manipulate the system through compromising some network nodes. In our scheme, base stations are assumed to be trusted and tamper resistant. No trust assumptions are made on the sensors. The CHs are not assumed to be tampering proof either. They can be compromised by an adversary. However, it is assumed that the compromise of a CH is more difficult than that of a sensor. A simplifying requirement in our design is that the adversary does not have any prior knowledge of what is stored at each node and, thus, cannot selectively direct the attack to a particular node based on what that node has.

5.2 Key distribution

Our key distribution scheme can be divided into two phases: key pre-distribution phase and key agreement phase. We make the assumption that there are m clusters in the WSN, each cluster Ci (i=1,…,m) has a CH (denotes as CH_i) and n−1 sensor nodes inside. And each node in cluster Ci has a unique identification C_{ij} (j=1,…,n−1), (CH_i also can be considered as the nth member and denote as C_{in}). For ease of presentation, we define a key space as a tuple (D,B) as in Ref.[7], where D and B are as defined in Section 3. A node shares a key space (D,B) if the node carries the key information generated from (D,B). Two nodes can calculate their pairwise key if they share a common key space. To match the clustered topology of WSN, two different types of key spaces are used in our scheme. One is (D_C,B), shared by the cluster CG, where CG is the identity of the special cluster that is composed of the CHs and the BS. The other is (D_i,B), which is shared by CH_i and cluster members in Ci.

5.2.1 Key pre-distribution phase

During this phase, we need to assign key information to each node, so that after deployment, the nodes in the same cluster can find a direct secret key between them.

Step 1. Generation of Matrices: We generate m symmetric matrices $D_1,\ldots,D_m$ of size n*n and a Vandermonde matrices B of size n*n for m clusters as described in Section 2. We call each tuple (D_i,B), i=1,…,m, a key space. Then, we compute the symmetric key matrix $A_i=(D_i,B)^T$. For example, let $c_j(B)$ represents the jth column of B and $r_j(A_i)$ represents the jth row of A_i. Although $c_j(B)$ has n elements, each node only needs to store the seed g^j of this

column, which will be used to regenerate the elements in $c_j(B)$. Similarly, CH_i stores the row seed s_{ij}.

At the same time, we generate a symmetric matrix D_C of size $(m+1)*(m+1)$ for cluster CG. Its key space is denoted as (D_C,B), e.g. the jth column seed and row seed can be denoted as g^j and s_{Cj} respectively. In this special cluster, the row seeds are stored in the BS's memory.

Step 2. Key Preloading: In this phase, every node is randomly assigned key information from its shared key space. Since each cluster has a cluster head and some sensor nodes, and their responsibilities are different, thus, different secret information will be preloaded into them.

Cluster head CH_i: Each cluster head not only needs to share a key space (D_C,B) with its neighboring CHs and the BS, but also needs to share a key space (D_i,B) with its cluster members. That means, it needs to store two tuples $(r_i(A_C),g^j)$ and $(r_n(A_i)),g^n,s_{ij}, j=1,...,n)$ in its memory. The former is CH_i's key information from key space (D_C,B); the latter are CH_i's key information from (D_i,B) and the row seeds for cluster members.

Sensor node C_{ij}: Each sensor node in Ci needs to store a tuple $(r_i(A_i),g^j)$, which is C_{ij}'s key information from key space (D_i,B).

5.2.2 Key agreement phase

After the key pre-distribution, each node in the network carries different key information. Using these keys information, they can establish pairwise key between any two of them to secure and authenticate their communications. There are two types of key establishments in our scheme: direct and indirect key establishments.

Step 1. Direct Key Sharing Establishment: This is achievable because any two CHs share the same key space and they can establish a pairwise key directly. Similarly, any two nodes in the same cluster can establish a pairwise key directly. The proposed approach in Section 3 can be used to achieve the purpose.

Step 2. Indirect Key Establishment: If two sensors fail to establish a pairwise key directly, they must start indirect key agreement phase. During this phase, a source sensor node tries to find another node that can help to setup a common key with the destination node. As we require in Section 5.1, a fraction of nodes from one cluster are interleaved with those from neighboring clusters, the indirect key can be established via nodes in these overlapping areas. Similar to the path key establishment in Ref.[8], the source node broadcasts a request message, which includes two IDs (one for the source node in Ci and the other for the destination node in Cj). If one of the nodes in the overlapping areas of Ci and Cj receives this request and also is able to establish a direct pairwise key with both of them, it replies with a message that contains two encrypted copies of a randomly generated key: one encrypted by the pairwise key with the source node, and the other encrypted by the pairwise key with the destination node. Both the source and the destination nodes can then get the new pairwise key from this message. If two nodes need to establish a pairwise key and their clusters have no overlapping areas, the cluster heads in both clusters will help to establish the indirect key; the process is similar to what we discuss above.

5.3 Compromise or failure mitigation

In this section, we discuss the handling of cluster head and node compromise or failure. The mitigation mechanism will be the same no matter what they detect, a node compromise or a node failure. That is because both cases need to evict the detected and add the new. Thus, the key-updating procedures in Section 4 are involved and become the basis of this section. For simplicity, we focus on node compromise in the following.

5.3.1 Cluster head compromise

In case of cluster head compromise, the key information in the cluster members has to be updated. After that, the adversary cannot decrypt future messages of the cluster. We assume that the compromise of the CH will also be detected by the method in Refs.[13,14]. On identifying a compromised CH, the BS notifies the rest CHs to relinquish the inter-cluster keys they share with the compromised CH. Recovery from a compromised CH can be

handled by either deploying a new CH or by redistributing the sensors of the compromised CH's cluster among the other clusters. The choice of the method to be used may depend on the applications.

Method 1. Deploying a replacement: If it is feasible to replace the compromised cluster head CH_i with another one CH_i' that has compatible capabilities in terms of computational resources and transmission range, the recovery procedure would be a regeneration of the key information for all nodes in *Ci*. Since CH_i belongs to two clusters (*Ci* and *CG*), upon deploying the new cluster head CH_i', the cluster members in these two clusters should update their key information. Namely, (1) BS should distribute new key information to the innocent nodes in *Ci* for establishing pairwise key with the newly deployed cluster head CH_i'. That means, the BS should reform a new key space (D_i',*B*) and repeat the key distribution and key agreement processes in Section 5.2. (2) For each CH_j, $j\neq i$, CH_i is their evicted member; they need to run the node eviction process as in Section 4.1 to shrink their key information. At the same time, CH_i' is a new member of cluster *CG*; the innocent CHs need to run the addition process as in Section 4.1 to extend their key information. After the key updating, the pairwise keys in two clusters are updated to isolate the evicted CH_i, which cannot interfere with the communications in *Ci* in the future.

Method 2. Sensor Redistribution: Once a cluster head CH_i in *Ci* is compromised, sensors belonging to its cluster are orphan sensors. If the deployment of a new CH is infeasible, an alternative approach is to redistribute the orphan sensors among the other clusters. Specifically, each orphan sensor will be associated to an existing CH based on the cluster criteria, e.g., location, communication range. The BS generates new row seeds and column seeds for the orphaned sensors, and the CKM and cluster members in the cluster, which the orphan sensors added in, need to update their key information. This redistribution procedure of the orphan sensors simply would be regarded as an addition process as described in the updating mechanism in Section 4.

5.3.2 Sensor compromise

In case of a sensor node compromise, the key information in the innocent nodes also needs to be updated. Recovery from a compromised sensor node can be handled by either deploying a new sensor if necessary, or just deleting the related key information stored in the innocent sensors. The key information updating in the former approach would be regarded as a combination of a real-time shrink process and an real-time extension process, and that in the latter approach simply would be regarded as an real-time shrink process, as described in Section 4.

6 Performance Evaluation

We analyze the proposed scheme to verify that our scheme is a reliable key updating scheme against the dynamic topology changes, while keeping network security and key connectivity between sensor nodes. The three evaluation metrics of the experiments are network security, key connectivity, and scalability. The network security is measured by the resilience to compromise attack; key connectivity is measured by the probability to re-establish a shared key during the dynamic topology changes; and the scalability is analyzed by discussing the memory overhead of the related sensor nodes during the dynamic topology changes. We also analyze the communication and computation overhead for our approach.

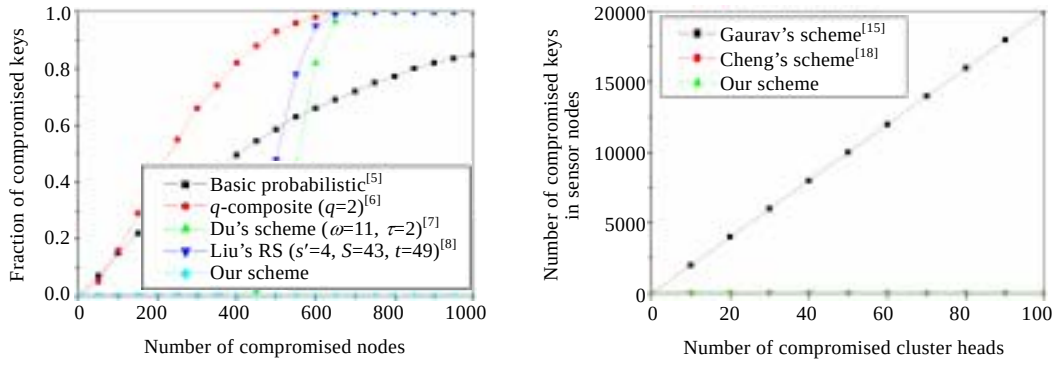
6.1 Network security analysis

Resilience means that resistance against node compromise; it requires that compromise of security credentials, which are stored on a sensor node, should not reveal information about security of any other links in the rest WSN; usually higher resilience means lower number of compromised links[11]. In this section, we study the resilience property of our scheme against nodes compromise and CHs compromise by calculating the effected fraction of communications links in the rest network due to key revealing (a secure link equals to a secure pairwise key).

Figure 5(a) shows the security performance of the basic probabilistic scheme[5], the *q*-composite scheme[6],

Liu's random subset (RS) scheme and our scheme. The figure clearly shows that as the number of compromised nodes increases, the fraction of affected communication links in our scheme remains zero. The reason is that, in our scheme each pairwise key is different from others, and any node's compromise triggers a key information updating process and, thus, the knowledge of the key information in the compromised nodes give no key information of the innocent nodes away. That means our scheme provides sufficient security, any sensor node's compromise cannot affect the secure communication between innocent nodes. Compared with other schemes, it is obvious that our scheme performs much better. In the probabilistic schemes[5,6], once the number of compromised nodes increases, the fraction of the affected communication links in the rest of the network increases quickly. The reason is that in probabilistic schemes the same key may be used or reused by several different pairs of sensors, some sensor nodes' compromise may compromise the communication links in the rest of the members. For the hybrid schemes[7,8], there are no key updating approaches, thus, when the number of compromised nodes are more than their thresholds, all pairwise keys of the entire network are compromised.

Next, we compare the performance of the LEKM[15], Cheng's scheme[18] and ours. Suppose there are N=20 000 sensors and 100 clusters in a network. Thus, in Ref.[15], each cluster head stores C=200 sensors' secret keys in its memory. Any single CH's compromise could affect the 200 sensors' secret keys. When the number of compromised cluster heads increases, the number of compromised sensors increases dramatically. In contrast, we can easily see from Fig.5(b) that as the number of compromised CHs increase, the compromised keys in the innocent member nodes in our scheme remains zero. This should owe to our compromise mitigation methods, which protect the nodes in the network from CH compromise attack. In Ref.[18], no communication between sensor nodes exists; only two bivariate polynomial shares are stored in each cluster head during the network initialization phase, and cluster heads have no idea about the sensors' secret keys. Therefore, the effected keys due to the cluster heads' compromise is also zero. But, Ref.[18] has no key updating approaches against the network topology changes. In a word, no matter how many CHs and sensors are compromised, our scheme can achieve perfect resilience.



(a) Fraction of compromised keys v.s. number of compromised nodes (N=20 000, C=200 keys, p=0.33)

(b) Fraction of compromised keys v.s. number of compromised cluster heads

Fig.5 Comparison of the resilience to compromise attack

6.2 Key connectivity

After the key-updating phase following the node compromise detection, two innocent sensor nodes in the rest network need to re-establish a direct or indirect key when the current pairwise key is compromised. In our scheme, since each pair of two communicating parties has a unique pairwise key, any sensor node's compromise cannot compromise the secure communication between innocent nodes. Thus, any two innocent nodes in the same cluster

also can establish a secret pairwise key between them with 100% certainty. Furthermore, the real-time shrink in the key updating mechanism triggered by the compromise detection disables the compromised links with the innocent nodes. Thus, node compromise can't interfere with the pairwise key setup among innocent nodes in our scheme.

Whereas, revocation of a captured node's key ring in Ref.[5] ensures that the set of (k) keys on that ring are removed network-wide. Though it disables all connectivity of the compromised node, this revocation affects a few other nodes and a part of their key ring, thus the shared key probability among the rest sensor nodes reduces. The schemes in Refs.[7,8] have no obvious key revocation mechanism, but we can also compute the re-establishing probability by assuming the compromised links are useless. Assume each node has available storage equivalent to C=200 keys, and contacts d=30 neighbor nodes and N=20 000. Figure 6 illustrates the relationship between the probability of re-establishing a pairwise key for innocent nodes and the number of compromised nodes in the network. It shows that our scheme has the highest probability to re-establish a pairwise key in the rest of WSN.

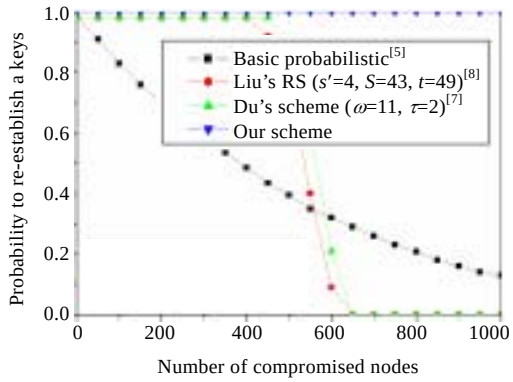


Fig.6 The probability of re-establishing a pairwise key v.s. number of compromised nodes (C=200, N=20 000, d=30)

6.3 Scalability

As far as the scalability is concerned, key distribution mechanism must support large networks, and must be flexible against dynamic changes in the topology of the network even after deployment[11]. As we introduce in Section 2, due to the limited physical memory size of sensor node, the supported network size is primarily dependent on the storage overhead of sensor nodes. Therefore, instead of calculating the maximum supported network size, we study the storage overhead of sensor nodes when the network size linearly increases and decreases.

In the previous section, we discussed that when the network size linearly increases, the DKPDs[2−4] have the problem of linearly increase storage overhead and cannot support large scale WSNs. Simultaneously, in random and hybrid KPD schemes[5−8], to achieve the required network connectivity, the number of keys stored in each sensor also need to increase linearly. Therefore, the maximum supported network size is limited in Refs.[5−8]. Based on the clustered network architecture, our proposed scheme as well as Ref.[18] has better performance than random key approaches. But, its key storage overhead is still sub-linearly increased when the network size linearly increases.

Figure 7 compares the number of keys stored in related sensor node for different schemes in terms of the number of compromised nodes (which leads to the linear reduction of network size). We can easily see that the related sensor nodes in our scheme have the lowest storage overhead when the network size linearly reduced. This is due to the real-time shrink approach in the key refreshing mechanism. There, key information about the compromised node is removed from the related innocent nodes' memory. Thanks to the centralized revocation

approach, the related sensor nodes' storage overhead in Ref.[5] is also reduced. But, the related sensor nodes' storage overhead in Ref.[18] is not reduced because of their lacking of key updating approaches.

In a word, our scheme has sufficient scalability. Especially, if the maximum size of the clusters is fixed at m ($m \prec N$, N is the network size), each sensor node stores only at most a row of the key information with the length of m and a column seed no matter how large the network size is (this is practical because apart from adding some replacement sensors and other cluster's orphan sensors into the existing clusters, most of the new sensors can be organized into a new cluster, which share another key space). Therefore, the network size is only decided by the cluster heads' capacity. In our proposed network model, cluster heads have relatively large memory size and sufficient power. In addition, the length of the key information rows is flexible instead of monotonously increasing when there are nodes added or evicted. Thus, our scheme can be scalable while keep sufficient network security if the suitable key spaces and clustering algorithms are properly selected.

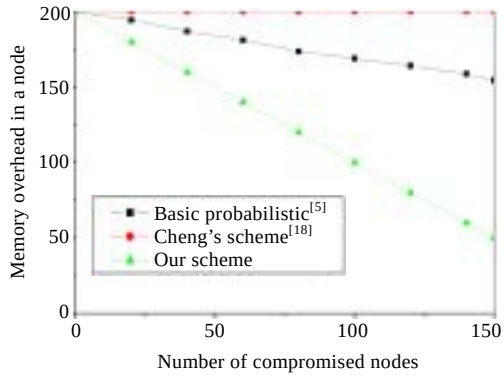


Fig.7 The memory overhead in a node v.s. number of compromised nodes

6.4 Communication and computation overhead

Since most of the existing key management schemes have no key updating approaches, there is no need to compare the total communication and computation overhead with them. Furthermore, the existing key distribution schemes usually study the communication and computation overhead in direct or indirect key establishment. Whereas, key updating is the major specialty of our scheme, and this is the first time that it has been introduced into the pairwise key management schemes, thus, instead of discussing the overhead of our scheme in the key establishment phase, we focus on discussing the overheads in the key updating phase.

In terms of communication overhead, when there is a node addition in a cluster with size n, the BS needs to broadcast a column seed and unicast a row seed to the CKM, and the CKM needs to multicast n extended elements of the matrix D to the existing member nodes to facilitate their key information extension. Thus, the communication overhead in node addition process is at most n+2. When there is a node eviction, the BS using RTS needs to announce that a column seed and a row seed are invalid, and the CKM needs to distribute k–1 elements if node k is compromised (k randomly equals to (1,…,n) with the probability of 1/n), so the CKM needs to distribute at average $\lceil (n-1)/2 \rceil$ elements every time. Thus, the communication overhead is at most $\lceil 3(n-1)/2 \rceil$.

In terms of the computational overhead, the node addition process using RTG needs n+1 hash computation, n+1 multiplication and n+1 exponentiations in the CKM, and 2n+1 multiplication and n+1 exponentiations in each sensor node. Using RTS the CKM in the deletion process needs $\lceil (n-1)/2 \rceil$ hash computations; each sensor node needs $\lceil (n-1)/2 \rceil$ hash computations, $n-1$ multiplication and exponentiations to shrink its key information.

7 Conclusions

In this paper, we propose a reliable pairwise key-updating (RPKU) scheme for clustered WSNs based on a modified version of Blom's key matrix construction. Our scheme has a number of appealing properties. Firstly, our scheme is the first one that can dynamically extend and shrink the key information to support the topology changes in large-scale networks and, thus it has good scalability. Secondly, our scheme is perfectly resilient against node and cluster head compromise. Thirdly, our scheme can keep the key connectivity with the time flows and network topology changes. Also, extensive analyses and simulations are conducted to evaluate the proposed schemes, and the results show that our scheme can achieve a good level of security, outperform most previous pairwise key management schemes, and significantly improve the network scalability and key connectivity while conservatively consuming nodes' resources.

Several research problems are worth further studying. Firstly, we will take different types of active attacks into consideration besides the node compromise attack. Secondly, we will try to find some new updating mechanism to address the key-updating problem in other network architectures, such as flat WSNs.

References:

[1] Tilak S, Abu-Ghazaleh NB, Heinzelman W. A taxonomy of wireless microsensor network models. ACM Mobile Computing and Communications Review, 2002,6(2):28−36.

[2] Blom R. An optimal class of symmetric key generation systems. In: Beth T, Cot N, Ingemarsson I, eds. Advances in Cryptology—EUROCRYPT'84. LNCS 209, Berlin, Heidelberg: Springer-Verlag, 1985. 335−338.

[3] Blundo C, Santis AD, Herzberg A, Kutten S, Vaccaro U, Yung M. Perfectly-Secure key distribution for dynamic conferences. LNCS 740, Berlin, Heidelberg: Springer-Verlag, 1993. 471−486.

[4] Choi1 SJ, Youn1 HY. An efficient key predistribution scheme for secure distributed sensor networks. In: Enokido T, *et al*., eds. EUC Workshops 2005. IFIP Int'l Federation for Information Processing. LNCS 3823, 2005. 1088−1097.

[5] Eschenauer L, Gligor VD. A key-management scheme for distributed sensor networks. In: Proc. of the 9th ACM Conf. on Computer and Communication security. Washington: ACM Press. 2002. 41−47.

[6] Chan H, Perrig A, Song D. Random key predistribution schemes for sensor networks. In: Proc. of the 2003 IEEE Symp. on Security and Privacy (SP 2003). Berkeley, 2003. 197−213.

[7] Du W, Deng J, Han YS, Varshney PK. A pairwise key pre-distribution scheme for wireless sensor networks. In: Proc. of the 10th ACM Conf. on Computer and Communications Security. Washingtion: ACM Press, 2003. 42−51.

[8] Liu D, Ning P. Establishing pairwise keys in distributed sensor networks. ACM Trans. on Information and System Security, 2005, 8(1):41−77.

[9] MacWilliams FJ, Sloane N. The Theory of Error-Correcting Codes. North Holland, 1997.

[10] Du W, Deng J, Han YS, Varshney PK. A key management scheme for wireless sensor networks using deployment knowledge. In: Proc. of the IEEE INFOCOM 2004. Hong Kong: IEEE Press, 2004. 586−597.

[11] Camtepe SA, Yener B. Key distribution mechanisms for wireless sensor networks: A Survey. Technical Report, TR-05-07, Rensselaer Polytechnic Institute, 2005.

[12] Carman D, Kruus P, Matt B. Constraints and approaches for distributed sensor networks security. Technical Report, 00-010. NAI Labs, 2000.

[13] Wang G, Zhang W, Cao G, La Porta T. On supporting distributed collaboration in sensor networks. In: Proc. of the IEEE Military Communications Conf. (MILCOM). Boston: IEEE Press, 2003. 752−757.

[14] Shaneck M, Mahadevan K, Kher V, Kim YD. Remote software-based attestation for wireless sensors. In: Proc. of the 2nd European Workshop (ESAS 2005). LNCS 3813, Visegrad: Springer-Verlag, 2005. 27−41.

[15] Jolly G, Kuscu MC, Kokate P, Yuonis M. A low-energy management protocol for wireless sensor networks. In: Proc. of the 8th IEEE Int'l Symp. on Computers and Communication (ISCC 2003). Turkey, 2003. 335−340.

[16] Zhao YJ, Govindan R, Estrin D. Residual energy scans for monitoring wireless sensor networks. In: Proc. of the IEEE Wireless Communications and Networking Conf. (WCNC 2002). Orlando: IEEE Press, 2002. 356−362.

[17] Ren FY, Huang HN, Lin C. Wireless sensor networks. Journal of Software, 2003,14(7):1282−1291 (in Chinese with English abstract). http://www.jos.org.cn/1000- 9825/14/1282.htm

[18] Cheng Y, Agrawal DP. An improved key distribution mechanism for large-scale hierarchical wireless sensor networks. Ad Hoc Networks, 2007,5(1):35−48.

附中文参考文献:

[17] 任丰原,黄海宁,林闯.无线传感器网络.软件学报,2003,14(7):1282−1291. http://www.jos.org.cn/1000-9825/14/1282.htm

WEN Mi was born in 1979. She is a Ph.D. student of Shanghai Jiaotong University. Her current research areas are security in wireless sensor networks, etc.

ZHENG Yan-Fei was born in 1976. She is a lecturer of Shanghai Jiaotong University. Her research areas are wireless sensor networks, etc.

CHEN Ke-Fei was born in 1959. He is a professor of Shanghai Jiaotong University. His research areas are classical and modern cryptography, etc.

LI Hui was born in 1977. He is a Ph.D. student of Shanghai Jiaotong University. His current research areas are wireless sensor networks, etc.

****

全国教育游戏与虚拟现实学术会议(EGVR 2007)

征文通知

2007 年 8 月 19~21 日，大连

主办：中国图象图形学会虚拟现实专业委员会、中国教育技术协会信息技术教育专业委员会

承办：辽宁师范大学

协办：大连民族学院、中国系统仿真学会数字娱乐仿真专业委员会、浙江大学数字娱乐与动画研究中心、南京师范大学教育游戏研究中心、中国传媒大学数字技术与艺术研发中心

全国教育游戏与虚拟现实学术会议(EGVR 2007)将于 2007 年 8 月 19 日至 21 日在大连举行。会议录用论文将结集出版，优秀论文将推荐到《计算机辅助设计与图形学学报》、《中国图象图形学报》、《系统仿真学报》等国内一级学报和核心期刊发表。

征文要求：(1) 论文未被其他会议、期刊录用或发表，中文撰写；(2) 要求电子投稿（Word 版本）；(3) 论文包含题目、中英文摘要、正文、参考文献等(正式格式见录用通知)；(4) 务必写清楚论文联系人的姓名、单位、通信地址、联系电话及 E-mail。

重要日期：征稿截止：2007 年 05 月 15 日 (最后收到日期)

录用日期：2007 年 06 月 15 日 (最后发出日期)

最终稿件：2007 年 06 月 20 日 (最后收到日期)

投稿邮箱：egvr2007@lnnu.edu.cn

更多详情请登陆网站了解：http://www.egvr2007.lnnu.edu.cn