A New Group Rekeying Scheme based on t-Packing Designs for Ad Hoc Networks*

Jianwei Chen
Key Lab of Network Security and Cryptology
Fujian Normal University
Fuzhou 350007, China
cjwxin_1@fjnu.edu.cn

Li Xu
Key Lab of Network Security and Cryptology
Fujian Normal University
Fuzhou 350007, China
xuli@fjnu.edu.cn

Yi Mu
School of Computer Science and Software Engineering
University of Wollongong
Wollongong, NSW 2522, Australia
ymu@uow.edu.au

ABSTRACT
Secure group key distribution and efficient rekeying is one of the most challenging security issues in ad hoc networks at present. In this paper, Latin squares are used to construct orthogonal arrays in order to quickly obtain t-packing designs. Based on cover-free family properties, t-packing designs are adopted in key pre-distribution phase. Then the pre-deployed keys are used for implementing secure channels between members for group key distribution. The new scheme improves the collusion-resilience of the networks using the cover-free family properties, and enhances the key-sharing connectivity of nodes which makes key management more efficient. This paper also presents in depth theory and data analysis of the new scheme in terms of network security and connectivity.

Categories and Subject Descriptors
C.2.0 [**Computer-Communication Networks**]: General – *Security and protection*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms
Security, Design

Keywords
Ad Hoc network, group rekeying, Packing Design, Cover-Free Family

1. INTRODUCTION
An ad hoc network is a collection of autonomous nodes that communicate with each other, most frequently using a multi-hop wireless network. Many applications of ad hoc networks involve collaborative computing among a large number of nodes and are thus group-oriented in nature. For deploying such applications like emergency rescue and battlefield communication, it is necessary to provide support for secure group communication.

The most efficient approach for achieving confidential group communication is to use a symmetric group key that is shared by all the nodes for data encryption. This approach however introduces the problem of group rekeying, i.e., the group key must be updated and redistributed to all the remaining nodes in a secure, reliable and timely fashion when group membership changes. Therefore this problem requires key management systems that provide support for dynamic properties. In general, Key management systems are of three types: namely key distribution, key agreement and key pre-distribution.

The traditional internet style key distribution protocols, for example Kerberos or adapted LKH [1] schemes, are infeasible for ad hoc networks because of their exclusive properties. These include communication range limitations, node movements, network dynamics and unknown network topology prior to deployment. On the other hand, contributory key agreement protocols [2,3,4], in which each node contributes an input to establish a common secret through successive pairwise message exchanges among the nodes in a secure manner using the 2-party Diffie-Hellman exchange, are not practical to ad hoc networks either. They are not robust to changing topology or intermittent links commonly occurring in an ad hoc network. In order to successfully establish a key, these protocols strictly require the underlying networks to either support broadcasting or have a relatively time-invariant topology of certain forms. Usually, all the nodes need to be online before the key establishment process is completed; if any node leaves in the midst due to link or battery outage, no common key would be established and the remaining nodes need to re-run the process from scratch. So the key agreement approach is not scalable due to the need of frequent interactive rekeying despite key freshness.

A number of recent works demonstrate that the key pre-distribution scheme (KPS) offers practical and efficient solutions to the key management problem. In KPS, each node receives a subset of keys from a key pool before deployment. Any two nodes able to find or compute non-interactively common keys within their respective subsets can use that keys as their shared secret to initiate communication. When we design a key management scheme based on KPS for ad hoc networks, the following key characteristics of the design must be considered.

- Connectivity: A network node should be able to securely communicate to its local neighbors. Here a local neighbor

* Partially supported by National Natural Science Foundation of China (No. 60502047), Scientific Program of the Educational Department of Fujian Province of China (No. JB06093).

INFOSCALE 2007, June 6-8, Suzhou, China

DOI 10.4108/infoscale.2007.929

means a network node physically located within transmission range.

- Resilience of a network: Even if many nodes are compromised by an adversary, the communications between other nodes should still be secure. In other words, a coalition of certain number of nodes cannot compute all secret keys used by other nodes.
- Small key size: Since a node has limited resource, key storage should be small. Therefore the number of keys distributed to a node should be small.
- Number of nodes: The maximum number of nodes that the scheme can support should be considered.

The rest of this paper is organized as follows. In the next section, a review of related work is introduced. In Section 3, we give the preliminaries. Then, we present the details of the new scheme in Section 4. Finally, we discuss some security and key-sharing connectivity issues in Section 5 and summarize the results in Section 6.

2. RELATED WORK

To date, the only practical option for the distribution of keys to nodes of ad hoc networks whose physical topology is unknown prior to deployment has been to rely on key pre-distribution. There exist a number of key pre-distribution schemes. Eschenauer and Gligor [5] proposed a random key pre-distribution scheme. Each sensor node receives a subset of random keys from the pool before deployment. Any two nodes able to find one common key within their respective subsets can use it as their shared secret to initiate communication. Based on this scheme, Chan, Perrig, and Song [6] proposed a *q*-composite random key pre-distribution scheme, which increases the security of key setup such that an attacker has to compromise many more nodes to achieve a high probability of compromising communication.

Recently, Du et al. [7] proposed another key pre-distribution scheme which substantially improves the resilience of the network compared to other schemes. This scheme exhibits a threshold property; when the number of compromised nodes is smaller than the threshold, the probability that any node other than the compromised nodes is affected is close to zero. Chan [8] proposed a fully distributed key pre-distribution scheme (DKPS) with no trusted authority for ad hoc networks. The DKPS is based on the precondition under which the key sets distributed to the network nodes can form a cover-free family. This desirable property leads to the fact that any subset of nodes can find from their key chains at least one common key not covered by a collusion of, at most, a certain number of nodes outside the subset. However, Wu and Wei [9] found that the precondition was falsely deduced. They claim that the probabilistic method (Chan used this method) cannot yield CFF practical for key distribution.

GKMPAN is an efficient group rekeying scheme for secure multicast in ad-hoc networks proposed by Zhu[10].GKMPAN also uses the probabilistic key pre-distribution technique as the underlying means to establish secure channels between nodes. However, compared to the previous schemes, GKMPAN uses the predeployed keys only as key encryption keys (KEKs) for securely distributing a group key to the nodes in the network while using the group key for securing group data communications. Thus, GKMPAN incurs much smaller communication and computational overhead in group communication. GKMPAN also includes an efficient mechanism to update the predeployed keys of nodes.

In this paper, we propose a new t-packing design based group rekeying scheme (PDGRS) for ad hoc networks. PDGRS builds on t-packing designs to pre-distribute node key-chains, and these keys are used for group rekeying. For this purpose, Latin squares are used to construct orthogonal arrays for quickly obtaining t-packing designs. The method makes the scheme mathematical model achieve cover-free family (CFF) properties [12], which improves the collusion-resilience of the networks. Moreover, updating the pre-deployed keys further prevent more compromised and revoked nodes from launching a collusive attack. Meanwhile, PDGRS enhances the key-sharing connectivity of nodes which makes keys distribution more efficient. Analysis shows that not only the key-sharing connectivity but also the collusion-resilience of the networks improves as the number of keys in a node increases compared to other existing schemes.

3. MATHEMATICAL MODEL AND RELATED DEFINITIONS

Cover-free families were first introduced by Kautz and Singleton [11] to investigate superimposed codes. Since then, cover-free families have been discussed in several equivalent formulations in subjects such as information theory, combinatorics and group testing by numberous researchers. Mitchel defined the concept of key distribution patterns, which in fact are a generalized type of CFF.

A set system is a pair (X, F), where X is a set of points and F is a set of blocks of X. The classical definitions of cover-free families [12] can be written as follows.

Definition 1. *A set system (X, F) is called a r cover-free family (or r-CFF) provided that, for any r blocks* $A_1, A_2, \cdots, A_r \in F$ *and any other block* $B_0 \in F$, *we have*

$$B_0 \not\subseteq \bigcup_{j=1}^{r} A_j \tag{1}$$

Definition 2. *A set system (X, F) is called a (r; d) cover-free family (or (r; d)-CFF) provided that, for any block* $B_0 \in F$ *and any other r blocks* $A_1, A_2, \cdots, A_r \in F$, *we have*

$$\left| B_0 \setminus \bigcup_{j=1}^{r} A_j \right| > d \tag{2}$$

The definition 2 states that the union of any *r* blocks contains at least *d* points that are not in it. Combinatorial designs can be used to constructed *r-CFF*. First we give the definition of a *t*-packing design as follows, and then other related definitions.

Definition 3. *A t-(v, k, λ) packing design is a set system (X, F), where* $|X| = v$, $|B| = k$ *for every* $B \in F$, *and every t-subset of X occurs in at most λ blocks in F.*

Definition 4. *A* $k \times v^t$ *array A with entries from V is an orthogonal array with v levels and strength t (for some t in the range* $o \le t \le k$ *) if every* $t \times v^t$ *subarray of A contains each t-tuple based on V exactly once (we assume the index λ=1) as a column. We denote such an array by* $OA(t, k, v)$.

Definition 5. *A Latin square of order n is an n by n array containing symbols from some alphabet of size n, arranged so that each symbol appears exactly once in each row and exactly once in*

each column.

If n is a prime or a prime power, then there exists a complete set of (n-1) mutually orthogonal Latin squares.

4. PROPOSED SCHEME

4.1 Network Assumptions and Main Notations

We assume an ad hoc network where there are N nodes. Network nodes communicate with each other and require pairwise keys to secure their communication for group rekeying. Each node has a key-chain of k keys which are selected from a key pre-distribution phase based on packing designs before the deployment. After that any two neighbor nodes find the common keys between their key-chains using cryptography homomorphism with secure shared key discovery (SSD) [8], and these keys are used to secure their communication. When a node joins or a member node leaves a group, the group key must be updated to enforce forward or backward secrecy. In addition, the pre-deployed keys need to be renewed.

The notations in Table 1 will appear in the rest of this paper.

TABLE 1. NOTATIONS

N	Number of nodes
n	Number of neighbor nodes
P	The key pool
p	A key in the key pool
q	A prime or a prime power
$E_k(msg)$	The encryption of message *msg* with key k
$H(x, y)$	The function to compute keys of nodes
$\{f_i\}$	A family of pseudo-random functions [16]
R_u	The key-chain of node u
m	Number of keys in a key-chain

4.2 Scheme Description

The scheme consists of the following phases.

- **Initial Setup Phase** The group controller (GC) selects parameters used in the scheme.
- **Key Pre-distribution Phase** Prior to the deployment of the ad hoc network, all nodes obtain a distinct subset of keys from the GC, based on packing designs.
- **Shared-key Discovery Phase** Nodes perform a protocol to discover their shared keys with their neighbors. Two nodes with shared keys are assumed securely connected. Next these keys are used as KEKs for delivering group keys.
- **Key Update** After a node receives and verifies the group key K, it updates its own pre-deployed keys based on K.

4.2.1 Setup Phase

The key pool P and parameters q and m are chosen by the GC. The choice of these parameters will determine the security level, the number of keys that a node has to store and communication efficiency of group key setup.

The number of keys in the key pool P is q. It is one-to-one mapping between the key pool P and the finite field $GF(q)$, that is , $P = \{p_i \mid i \in GF(q)\}$.

4.2.2 Key Pre-distribution Phase

From the above parameters, the GC constructs a t-packing design that involves the following steps.

Step 1. construct mutually orthogonal Latin squares of order n according to the following theorem.

Theorem 1[13]. *Select a primitive element a from a finite field GF(n), then*

$$B_{i+1} = \begin{pmatrix} 0 & 1 & a & \cdots & a^{n-2} \\ a^i & 1+a^i & a+a^i & \cdots & a^{n-2}+a^i \\ a^{i+1} & 1+a^{i+1} & a+a^{i+1} & \cdots & a^{n-2}+a^{i+1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a^{i+n-2} & 1+a^{i+n-2} & a+a^{i+n-2} & \cdots & a^{n-2}+a^{i+n-2} \end{pmatrix}$$

, for $i = 0,1,\cdots,n-2$ *, is a complete set of orthogonal Latin squares of order n.*

Step 2. Over the complete set of orthogonal Latin squares of order n, an orthogonal array $OA(t,k,v)$ can be constructed by the way that the elements of each square are written in order in a line.

Step 3. In this step, suppose $\{s_1, s_2, \cdots, s_k\}$ is a column in the $OA(t,k,v)$. Define a block as $\{(0,s_1),(1,s_2),\cdots,(k-1,s_k)\}$ accordingly. In this way, we can obtain a $t-(ks,k,1)$ packing design from the $OA(t,k,v)$.

After a t-packing design has been constructed, each node is loaded with the following information:

1. Each node u is loaded with R_u, which contains keys computed from the equation (3), and these keys are used as KEKs. Specifically, for each node, the GC chooses a block $B = \{(j,i) \mid j = 0,1,2,\cdots q; i \in GF(q)\}$ from the t-packing design upon the input of a node id. Next the block is used to calculate the corresponding keys according to the equation (3) .

$$k_j = H(j, p_i), (j,i) \in B \tag{3}$$

2. Each node is loaded with the initial group key k_g .

Example1. We illustrate the proposed phase using an example below, involving the construction of a t-packing design.

Step1. Assume that $q = 5$, $GF(5)$={0,1,2,3,4}. And the GC generates a key pool $P = \{p_0, p_1, p_2, p_3, p_4\}$.

Step2. Construct a complete set of 4 mutually orthogonal Latin squares of order 5 as follows.

$$\begin{pmatrix} 0&1&2&4&3 \\ 1&2&3&0&4 \\ 2&3&4&1&0 \\ 4&0&1&3&2 \\ 3&4&0&2&1 \end{pmatrix} \begin{pmatrix} 0&1&2&4&3 \\ 2&3&4&1&0 \\ 4&0&1&3&2 \\ 3&4&0&2&1 \\ 1&2&3&0&4 \end{pmatrix} \begin{pmatrix} 0&1&2&4&3 \\ 4&0&1&3&2 \\ 3&4&0&2&1 \\ 1&2&3&0&4 \\ 2&3&4&1&0 \end{pmatrix} \begin{pmatrix} 0&1&2&4&3 \\ 3&4&0&2&1 \\ 1&2&3&0&4 \\ 2&3&4&1&0 \\ 4&0&1&3&2 \end{pmatrix}$$

Step3. We construct 6×25 $OA(2,6,5)$ using above mutually orthogonal Latin squares. Note that how many Latin squares we apply will determine the number of elements that a block has, that is, the number of keys m that a node has. Assume here that we use all Latin squares and write in order the elements of each square in

a line from the 3rd row of the array below. As a result, $OA(2,6,5)$ is obtained as follows.

$$\begin{pmatrix}
0&0&0&0&0&1&1&1&1&1&2&2&2&2&2&3&3&3&3&3&4&4&4&4&4\\
0&1&2&3&4&0&1&2&3&4&0&1&2&3&4&0&1&2&3&4&0&1&2&3&4\\
0&1&2&4&3&1&2&3&0&4&2&3&4&1&0&4&0&1&3&2&3&4&0&2&1\\
0&1&2&4&3&2&3&4&1&0&4&0&1&3&2&3&4&0&2&1&1&2&3&0&4\\
0&1&2&4&3&4&0&1&3&2&3&4&0&2&1&1&2&3&0&4&2&3&4&1&0\\
0&1&2&4&3&3&4&0&2&1&1&2&3&0&4&2&3&4&1&0&4&0&1&3&2
\end{pmatrix}$$

Step4. Finally the following $2-(30,6,1)$ packing design is derived .

$$\begin{pmatrix}
(0,0)&(0,0)&(0,0)&(0,0)&(0,0)&(0,1)&(0,1)&(0,1)&(0,1)&(0,1)&\cdots\\
(1,0)&(1,1)&(1,2)&(1,3)&(1,4)&(1,0)&(1,1)&(1,2)&(1,3)&(1,4)&\cdots\\
(2,0)&(2,1)&(2,2)&(2,4)&(2,3)&(2,1)&(2,2)&(2,3)&(2,0)&(2,4)&\cdots\\
(3,0)&(3,1)&(3,2)&(3,4)&(3,3)&(3,2)&(3,3)&(3,4)&(3,1)&(3,0)&\cdots\\
(4,0)&(4,1)&(4,2)&(4,4)&(4,3)&(4,4)&(4,0)&(4,1)&(4,3)&(4,2)&\cdots\\
(5,0)&(5,1)&(5,2)&(5,4)&(5,3)&(5,3)&(5,4)&(5,0)&(5,2)&(5,1)&\cdots
\end{pmatrix}$$

After the packing design is completed, the GC selects each node's, say u, block upon the input of its id. Suppose B_u={(0,0),(1,4),(2,3),(3,3),(4,3),(5,3)}. And the GC calculates its corresponding key-chain distributed to node u as equation (4) according to the equation (3).

$$R_u = \{H(0,p_0), H(1,p_4), H(2,p_3), H(3,p_3), H(4,p_3), H(5,p_3)\} \tag{4}$$

Our scheme does not require a key pre-distribution phase for every instance of network formation. Indeed, there is no limit on how many times these pre-distributed keys can be used securely because our rekeying scheme updates these keys securely after every group rekeying.

4.2.3 Shared-key Discovery Phase

After the key pre-distribution phase is completed, each node is deployed in different places. Any two neighbor nodes, say u and w, will perform SSD scheme, which uses privacy homomorphism to find common keys between R_u and R_w. The SSD scheme allows two nodes to find out common keys in their key-chains, but not to leak out to the other side any information of the keys outside the common intersection of the two key-chains.

In the Example1, after the deployment, u and w become neighbor nodes which are respectively assigned key-chains as follows.

$R_u = \{10,34,37,35,87,79\}$

$R_w = \{30,51,29,93,19,79\}$.

Then based on SSD scheme they derive their shared-key {79}。

4.2.4 Node Join

In this section, without losing generality, suppose a new node u wants to join an existing group. For example, the GC may introduce new nodes into the system to compensate for revoked nodes. To enforce forward secrecy, the following steps will be adopted.

Step 1. The GC generates an new group key $k_g^{'}$, and broadcasts the message $E_{k_g}(k_g^{'})$ to the network.

Step 2. Every node, say v, updates every key k_i in R_v as $k_i^{'} = f_{k_i}(0)$. We denote the updated set of keys as $R_v^{'}$.

Step 3. After the key update operations, every node erases the old group key k_g .

Step 4. Finally, the GC determines u's key set R_u based on its node id. Then it loads node u with current version of R_u and the current group key over a secure channel. Such a confidential and authentic channel can be established if user physically goes to the GC or the keys can be protected by a simple blinding technique [14].

4.2.5 Node Revocation

In this part, we are going to describe the key update operations when a node leaves a group. The leaving action may happen voluntarily or when a compromised node is detected and expelled from a group. Either way, the keys must be updated to enforce backward secrecy. Let u be the node to be revoked. The following steps will be adopted.

Step 1. The GC determines l keys $\{k_1,k_2,\cdots,k_l\}$, which are the non-compromised keys that are possessed by the remaining nodes in the network, and these keys are used as KEKs. The GC then generates an new group key $k_g^{'}$. Then it broadcasts a node revocation message as equation (5) to the network.

$$\mathrm{GC} \rightarrow *: \ ID_u, \ \{E_{k_1}(k_g^{'}), E_{k_2}(k_g^{'}), \cdots, E_{k_l}(k_g^{'})\}, \ f_{k_g^{'}}(0) \tag{5}$$

Step 2. The nodes that possess one of the l keys $\{k_1,k_2,\cdots,k_l\}$ can compute the new group key $k_g^{'}$ independently. Otherwise, they can obtain it over the shared-keys with their neighbors. Node u will not receive $k_g^{'}$ even though it can impersonate a non-revoked node v by claiming node v's id, because none of the keys in R_u are used. And node u also can not derive $k_g^{'}$ from its neighbors, since the node revocation message involves its node id.

Step 3. After every node receives the new group key $k_g^{'}$, it verifies the correctness of $k_g^{'}$ by checking if $f_{k_g^{'}}(0)$ equals to that in the node revocation message. If equals, every node, say v, updates every key k_i in R_v as $k_i^{'} = f_{k_i}(0)$. We denote the updated set of keys as $R_v^{'}$.

Step 4. After the key update operations, every node erases the old group key k_g .

In step 1, the l keys chosen by the GC can be the non-compromised keys that are possessed by the maximum number of nearby remaining nodes of the GC in the network. When a node possesses none of the l keys, it can obtain the group key over the shared-keys with its neighbors. As long as the key-sharing connectivity of nodes is high, the group key will be efficiently distributed to the remaining nodes in the network.

5. ANALYSIS

In this section, we first analyze the security and the key-sharing connectivity of our scheme, then discuss the tradeoff

between security, connectivity and storage cost, finally compare the properties of our scheme with that of some other schemes.

5.1 Security Analysis

Except for forward and backward secrecy, the security of our group rekeying scheme is mainly two-fold.

Defending against Collusive Attacks From the above statement, the proposed scheme based on t-packing designs yields CFF properties.

Theorem 2. *If there is an* $OA(t,k,v)$ *, then there is a t —(kv, k, 1) packing design that contains* v^t *blocks.*

Proof. Suppose that there is a $OA(t,k,v)$ with entries from the set $\{0,1,\cdots,v-1\}$ .Define $X=\{(x,y)\,|\,0\le x\le k-1, 0\le y\le v-1\}$. For every column $(y_0,y_1,\cdots,y_{k-1})$ in the orthogonal array, define a block $B=\{(0,y_0),(1,y_1),\cdots,(k-1,y_{k-1})\}$. Let F consist of the v^t blocks thus constructed. It is easy to check that (X, F) is a $t-(kv,k,1)$ packing design.

A t-packing design is an r-CFF for certain value of r. We obtain the following construction.

Theorem 3[12]. *If there exists a t-(v, k, 1) packing design having b blocks, then there exists a (r; d)-CFF(v, b), where* $r=\lfloor (k-d-1)/(t-1)\rfloor$.

In PDGRS, q is a prime or a prime power, and there exists a complete set of (q-1) mutually orthogonal Latin squares. Using definition 1 and the above lemmas, we can easily obtain the following result.

Corollary 1. *For any prime power q and any integer t < q, then there exists an OA(t, q+1, q), such that a* $t-(kq,k,1)$ *packing design with* q^t *blocks exists, and hence there exists a* $(\left\lfloor\frac{k-d-1}{t-1}\right\rfloor,d)-CFF(qk,q^t)$ *,where* $k\le q+1$.

Given $k=q+1$, we have the following.

$$(\left\lfloor\frac{q-d}{t-1}\right\rfloor,d)-CFF(q^2+q,q^t) \qquad (6)$$

Corollary 2. *In the scheme PDGRS, when the number of colluding nodes is less than r, other secret keys used by any other nodes can not be completely covered.*

For example, we choose q=113, d=2 and the number of keys stored in a node m is 114, then the result r=111 is obtained. That is, at least two keys of any other legitimate nodes are secure, when the number of simultaneously colluding nodes is less than 111.

In Fig.1 we compare the number of colluding nodes (denoted as w) that PDGRS and GKMPAN[10] can tolerate by varying the number of keys in a node. We can observe that the number of colluding nodes PDGRS resists increases with m, but GKMPAN inverses. In PDGRS, w and m are in direct proportion basically. While in GKMPAN, for a fixed probability 0.01% that a node is covered, the number of colluding nodes the scheme resists decreases with m. For example, for a group size of 10,000, when m=120, the coalition of only 20 nodes can lead to have keys to cover a legitimate node. Note other schemes [5,6] have a similar result like GKMPAN.

Updating pre-deployed keys To further improve the resilience, our scheme also updates the pre-deployed keys as GKMPAN. It is critical in order to prevent more compromised and revoked nodes from launching a collusive attack in which they pool together their keys with the goal of jeopardizing other legitimate nodes. Without key updating, both the performance and security of the system will degrade greatly with the number of compromised nodes. That is, we only need to guarantee that the number of compromised or revoked nodes between two key refreshment is less than the threshold r, because the status of the system is reinstated to its original setting after every rekeying. Consequently, the security of our scheme can be strengthened largely.

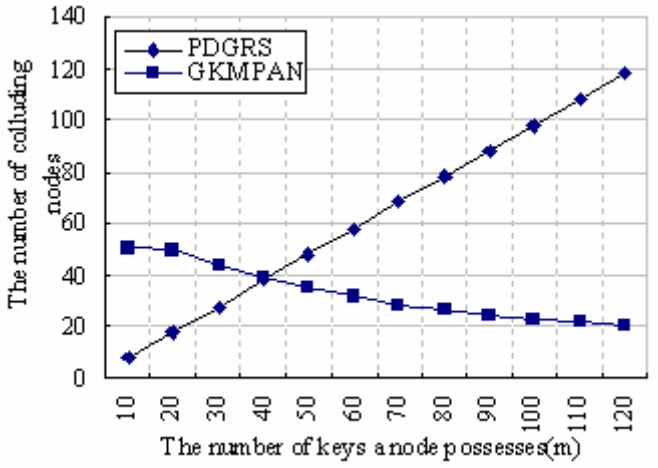

Figure 1. The number of colluding nodes that PDGRS and GKMPAN can tolerate by varying the number of keys in a node

5.2 Key-sharing Connectivity Analysis

As we have just shown, to make it possible for any node to be able find shared keys with its neighbors to secure group communication, the key sharing graph needs to be connected. In order to efficiently deliver the group key, the probability(P_c) that the key-sharing graph is connected must be as high as possible.

Using connectivity theory in a random-graph by Erdos and Renyi [15], we can obtain the necessary expected node degree d (i.e., the average number of edges connected to each node) for a network of size N when N is large in order to achieve a given global connectivity, P_c:

$$d=\frac{(N-1)}{N}\left[\ln(N)-\ln(-\ln(P_c))\right] \qquad (7)$$

Fig.2 illustrates the plot of the expected degree of a node, d, as a function of the network size, N, for various values of P_c. For example, we choose N=4000, to obtain P_c=0.999, the necessary expected node degree d is at least 16.

For a given density of network deployment, let n be the expected number of neighbors within the communication range of a node. Using the expected node degree calculated above, the required local connectivity, $P_{required}$, can be estimated as follows, $P_{required}=\frac{d}{n}$. After we have selected values for q and m , the actual local connectivity is determined by these values. We use

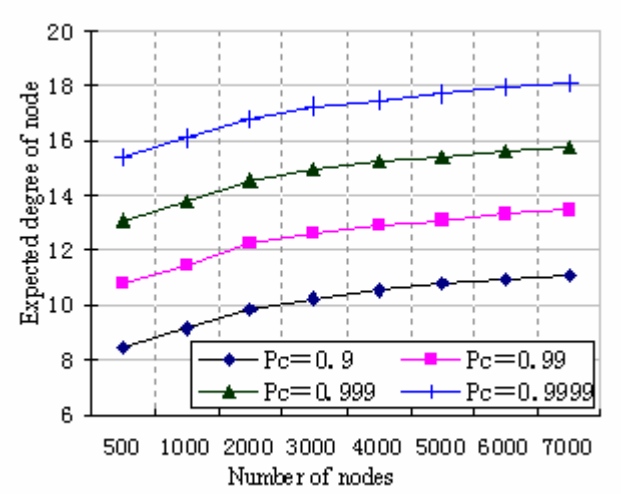

Figure 2. Expected degree of a node for varying number of nodes

P_{actual} to represent the actual local connectivity, which is the probability of any two neighboring nodes sharing at least one key. In our scheme,

$$P_{actual} = \frac{\frac{bk}{v}-1}{b-1}\cdot k = \frac{k(bk-v)}{v(b-1)} = \frac{k}{q+1} \quad (8)$$

In order to achieve the desired global connectivity P_c, we should have $P_{actual} \geq P_{required}$, and make P_{actual} become as high as possible. According to equation (8), we observe that P_{actual} increases with k for fixed q. When $k=q+1$, $P_{actual}=1$, namely, any pair of nodes can find at least a common key between them.

In Fig. 3 we compare the P_{actual} of PDGRS and GKMPAN by varying m, the number of keys in a node. In PDGRS, q=113. And the key pool size of the two schemes is equal. We can observe that the P_{actual} of them increases with m, but PDGRS outperforms GKMPAN. That is, the P_c for PDGRS is much higher than that of GKMPAN with m.

From the above analysis, however, we see that the actual local connectivity depends on the amount of space available on a node for storing keys, therefore, when the node resource is limited, we will improve the P_c by directly increasing the node degree d. PDGRS uses the following two ways to increase d. The first is that a node u can use its neighbors which have shared keys with u, to establish a secure channel with other nodes in u's one-hop communication range. We take node a (in Fig. 4(a)) as an example. In node a's one-hop communication range, node b has common keys with node a and node c respectively, but no common key exists between node a and node c. In this case, when node a wants to establish shared keys with node c, it can ask node b to act as a proxy. Suppose node a shares a key k_{ab} with node b, node c shares a key k_{bc} with node b. To forward a key k to node c, the following steps are taken.

$$a \to b : E_{k_{ab}}(k), \quad b \to c : E_{k_{bc}}(k) \quad (9)$$

The second way is to use two-hop neighbors. A two-hop neighbor of node u is a node that can be reached via one of u's one-hop (or direct) neighbors. To send a message to a two-hop neighbor, u needs to ask its direct neighbor to forward the message. We also take node a (in Fig. 4(b)) as an example. Node b has common

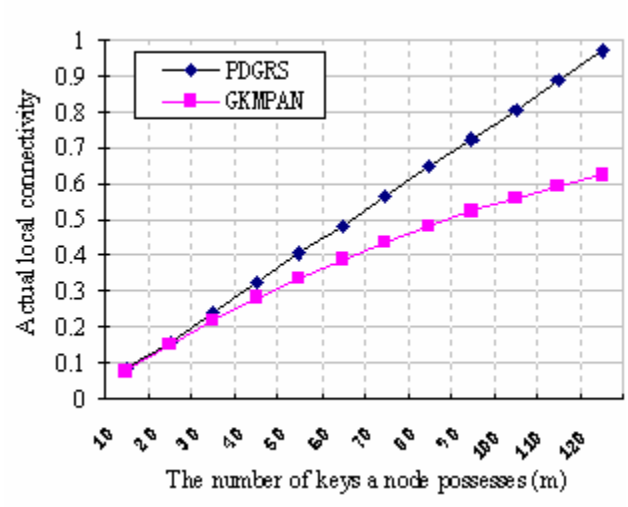

Figure 3. Comparison of the connectivity of the proposed scheme with the existing scheme

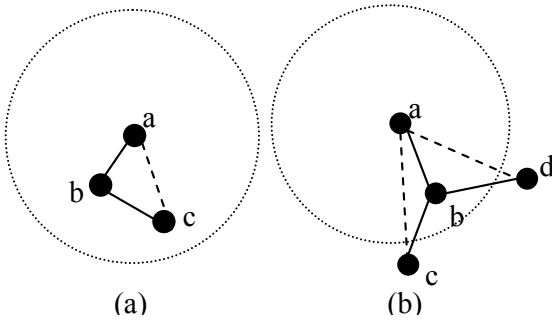

Figure 4. Establishing shared keys with more nodes (a) in one-hop communication range (b) in two-hop communication range

keys with node a and node c respectively. Node c is similar to the case above, except that node c is out of node a's one-hop but in two-hop communication range. Therefore node a asks node b to act as a proxy, not only to establish a secure channel with node c but also to forward messages to node c. For node d, it is also out of node a's one-hop communication range, but has common keys with node a. So, in this case node b only needs to forward messages. Suppose node a shares a key k_{ad} with node d. To forward a message msg to node d, the following steps are taken.

$$a \to b : E_{k_{ad}}(msg), \quad b \to d : E_{k_{ad}}(msg) \quad (10)$$

6. CONCLUSIONS

Secure group rekeying has become an important component of many applications in ad hoc networks. In this paper, we have presented PDGRS, a new t-packing design based group rekeying scheme for ad hoc networks, which focuses on key distribution and update for secure group communication. Different from the previous approaches, we use Latin squares to construct orthogonal arrays in order to quickly obtain t-packing designs, which are adopted in key pre-distribution phase, and then the pre-deployed keys are used for group rekeying. The proposed scheme achieves cover-free family properties. The collusion-resilience as well as the key-sharing connectivity of networks improves with

increasing the number of the keys in a node. Moreover, updating pre-deployed keys further enhances the security of the new scheme.

In order to increase the key-sharing connectivity and enhance the security, it is necessary to increase the number of keys each node stores. However, from the viewpoint of storage, resource of node in ad hoc is smaller. Due to these conflicting requirements, the common parameter number of keys should be selected based on the application under consideration. In addition, recall that PDGRS uses a key pool, but the GC does not directly select keys distributed to nodes from the key pool. In order to satisfy the needs of a large network, a function is used to generate more keys for a number of nodes, by which PDGRS can reduce the overhead of the GC to store keys and update the pre-deployed keys.

7. REFERENCES

[1] L. Lazos and R. Poovendran. Energy-Aware Secure Multicast Communication in Ad-hoc Networks Using Geographic Location Information. In *Proc. of IEEE ICASSP'03*, Hong Kong, China, April, 2003.

[2] G. Ateniese, M. Steiner, and G. Tsudik. a "New multiparty authentication services and key agreement protocols", *IEEE Journal on Selected Areas in Communications*, 18(4):628-640, April 2000.

[3] Y. Kim, A. Perrig, and G. Tsudik. "Communication-efficient group key agreement", in *proc. IFIP SEC'01*, 2001.

[4] D. A. McGrew, and A. T. Sherman. "*Key establishment in large dynamic groups using one-way function trees*", May 1998.

[5] L. Eschenauer and V. D. Gligor.: A key-management scheme for distributed sensor networks: *Proceeding of the 9th ACM Conference on Computer and Communication security*, 41-47, 2002.

[6] H. Chan, A. Perrig, and D. Song. Random key pre-distribution schemes for sensor networks:*IEEE Symposium on Security and Privacy*, 197-213, 2003.

[7] W. Du, J. Deng, Y. S. Han and P. K. Varshney, A pairwise key pre-distribution scheme for wireless sensor networks, *Proc. of the 10th ACM conf. on Computer and communications Security*, 42-51, 2003.

[8] A. C.-F. Chan and E. S. R. Sr. Distributed symmetric key management for mobile ad hoc networks. In *Infocom 2004*, 2004.

[9] J. Wu and R. Wei. Comments on" distributed symmetric key management for mobile ad hoc networks" from infocom 2004. Cryptology ePrint Archive, Report 2005/008, 2005. http://eprint.iacr.org/.

[10] Zhu S, Setia S, Xu S, and Jajodia S. GKMPAN: An Efficient Group Rekeying Scheme for Secure Multicast in Ad-Hoc Networks[C]. In *Proc. of International Conference on Mobile and Ubiquitous Systems: Networking and Services*, 42- 51, 2004.

[11] D. Wagner, "*Cryptanalysis of an Algebraic Privacy Homomorphism*", ISC, 234-239, 2003.

[12] R. Wei. On cover-free families, *Discrete Math*., to appear.

[13] Yang Z X. Construction of Orthogonal Arrays. Jinan: Shandong People Press, 1978.

[14] Lee B, Boyd C, Dawson E, Kim K, Yang J, and Yoo S. Secure Key Issuing in ID-Based Cryptography[C], *CRPIT '04: Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation*, Australian Computer Society, Inc., 69-74, 2004.

[15] Erdos, Renyi. On random graphs I. *Publ. Math*. Debrecen, 6:290–297, 1959.

[16] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *Journal of the ACM*, 33(4): 210-217, 1986.