Self-organised Group Key Management for Ad Hoc Networks

Ling Luo, Rei Safavi-Naini, Joonsang Baek and Willy Susilo
University of Wollongong
Wollongong NSW 2522 Australia
ll82,rei,baek,wsusilo@uow.edu.au

ABSTRACT

We propose a fully distributed group key distribution protocol for ad hoc networks. The protocol uses a key pre-distribution step that is performed by each node independently and generates secure links between nodes in a neighbourhood. The key pre-distribution step also allows formation of an initiator group who will generate a session key that will be distributed to all nodes using the secure links between nodes obtained in key pre-distribution stage. We describe efficient protocols for join of new nodes and revocation of compromised nodes. We analyse the system by calculating probability of success of each operation. We evaluate security of the system against outside eavesdroppers and discuss its security against an adversary that corrupts the nodes of the network. Finally we compare our system with two competing systems and show its superior performance in some scenarios.

Keywords

Ad hoc network, Key distribution, Privacy homomorphism

1. INTRODUCTION

1.1 Background

An ad hoc network is a collection of autonomous nodes that communicate with each other, most frequently using a multi-hop wireless network [4]. Nodes do not necessarily know each other and come together to form an ad hoc group for some specific purpose. Ad hoc networks are usually deployed when it is not possible, or is too expensive, to have a fixed support infrastructure. Typical applications include communication network in battlefield, disaster recovery, emergency operation and environment monitoring. Ad hoc networks typically do not have an online trusted authority but there may be an off line one that is used during system initialisation.

A node in an ad hoc network has direct connection with a set of nodes, called *neighbouring* nodes, which are in its communication range. The number of nodes in the network is not necessarily fixed. New nodes may join the network while exiting ones may be compromised or become un-functional and so need to be revoked. Securing communication in ad hoc networks requires key management systems that provide support for dynamic properties. Key management systems in general are of three types: key distribution, key agreement and key pre-distribution. Key distribution systems [14] usually require a trusted third party that acts as a mediator between nodes of the network. In key agreement protocols [9] nodes interact with each other to compute a common key. Key agreement protocols do not usually require a trusted authority. A number of such protocols [3], [2], [13] use Diffie-Hellman style key agreement approach for groups. A shortcoming of these protocols for ad hoc networks is that they usually require a broadcast channel and hence routing infrastructure. Moreover, a leave or join would normally require the whole protocol to start over again.

Key pre-distribution systems proposed in [1] provide an attractive method for establishing secure links between nodes without requiring interactions between nodes. In these systems a trusted authority gives a set of keys to each node. The choice of the subsets is such that *privileged* subsets of users share a common key, and *forbidden* sets do not learn anything about the key. Key pre-distribution systems use combinatorial structures that can be constructed for certain parameters and so may not exist for a given scenario.

Eschenauer and Gligor [12] proposed a probabilistic key pre-distribution scheme for mobile ad hoc networks. The scheme was later extended in [8] to increase its security. The scheme requires a trusted third party to randomly select a set of keys from a secret pool of keys, for each node. This is called the *key ring* of the node. Keys have *key identifiers* and a key ring stores keys and their corresponding identifiers. A node uses a *shared key discovery protocol* with its neighbours to find keys that it shared with neighbouring nodes. These keys are used to establish a *secure link* between these nodes. It is shown that with the right set of parameters there is a good chance that all node are securely connected, that is for any two nodes there is path consisting of secure links joining the two nodes.

Chan [7] proposed a fully distributed key pre-distribution scheme with no trusted authority. In this protocol a node selects its own key ring from a public set of keys, called the *key pool*. The shared key discovery in this scheme is cryptographic and uses homomorphic encryption. This is because there is no common set of identifiers for keys. Chan's protocol uses a special method of key selection so that with high

ASIACCS '06 March 21 - 24, 2006, Taipei, Taiwan.

probability the collection of key rings form a combinatorial structure called *cover-free property*. This structure had been previously used for key distribution. Chan's contribution was to make the construction distributed and without the need for a trusted authority. It was shown [20] that Chan's argument was flawed and the success chance of establishing keys with other nodes in Chan's scheme was very small, rendering the system unworkable.

The trust and attack model in Eschenauer et al.'s scheme is different from Chan's one. In the former an adversary can capture a node and the damage caused by such a compromise is considered by authors. Their results suggest that compromise of one node does lead to the compromise of another link with probability of 0.1. In Chan's system the adversary can choose its key ring and colluding attack, where a subset of malicious nodes attempt to learn the communication between other parts of the network, is considered. Eschenauer et al.'s scheme and Chan's scheme both have the attractive properties that the size of key rings is fixed and is relatively small. Moreover both schemes support efficient join and leave operations.

1.2 Our Contributions

We propose a fully distributed probabilistic group key predistribution scheme that does not require a trusted authority. The scheme uses an *iterative approach* that allows a group of nodes to securely share a session key. The session key is generated by an *initiator* group and is forwarded to the rest of the nodes in a number of iterative steps. At each step the group of nodes that share the session key forward their keys to their neighbours, or to a selected set of nodes that are chosen according to a certain group policy. In this paper we consider the case that all nodes will share the key and this is achieved with the smallest communication overhead.

Nodes choose their own key rings. We require the nodes to be securely connected with high probability while having a reasonable size key ring. If the key pool is large, then the chance of having a common key between two randomly selected key rings will be very small. We use a key selection strategy similar to Chan that increases this chance without reducing security. We partition the key pool of size N into k subsets called *blocks*, of size m each, and require a node to select one key from each block resulting in a key ring of size k. To have a secure link, two nodes must have l shared keys. The link key can be obtained as a combination of the l keys and so finding the link key requires l keys to be correctly guessed.

The session key will be generated by a group of nodes, called the *initiator group (IG)*, that, (i) are all in the neighbourhood of each other, and (ii) they all share the same set of l keys. This means that the group has a secure broadcast channel so that nodes in the group can broadcast message to each other.This is required to ensure the session key is securely generated. Using higher ℓ will increase link security while will reduce the chance of forming the IG. We derive expressions that relate various system properties such as security, connectivity and efficiency of the system.

The communication cost of establishing a group key will be directly related to the number of iterative steps used in the protocol. We will derive an expression to determine the number of steps in terms of the size of IG and the neighbourhood size of nodes.

We give the results of our simulation for networks. We compare efficiency of our system with Eschenauer et al.'s scheme and a group key agreement protocol.

1.3 A Motivating Scenario

Ad hoc networks are used in many application scenarios. Although support for dynamic group properties such as frequent join or leave is essential, assumptions such as existence of a trusted third party is application dependent. We assume that an ad-hoc group with no trusted authority and that all nodes in the ad hoc group need to securely communicate with each other. An example of such a scenario is an emergency operation team in which people need to be able to securely communicate with each other. An emergency worker receives a handhold device equipped with a trusted hardware that can securely run the protocol. The device does not have a secure memory and so no long term key can be held on the device. The device has a secure hardware identification number such as a identification number in a SIM card in a mobile phone, that allows the device to be identified. The devices can access a public list of keys. A group member stores the identification information of other devices that will be taken to the field. This list can be updated at a later stage by adopting a policy for joining new nodes. To be able to join the group new nodes are required to have approval of one of a current group members.

Our proposed protocol is well suited to the above scenario. Eschenauer et al.'s scheme will require devices with secure memory to hold the secret keys and a trusted authority to allocate keys. A new device can only join if it has received a key ring from the trusted authority. The authority is also required for key revocation. This is to protect against insecure join and allowing only nodes with listed identification to be part of the group.

1.4 Related Work

Key pre-distribution systems have been proposed independently by [1] and [16]. Key distribution patterns were proposed in [18]. These schemes, also known as zero-broadcast conference key, allow a group of nodes to setup a conference key without broadcasting. Systems for key management for sensor networks are discussed and compared in [6]. Key pre-distribution for ad-hoc networks has been considered in [12]. Song et al. [8] extend Eschenauer et al.'s scheme to provide higher security. In a key pre-distribution system, a trusted authority securely generates and distributes secret information to each user. There are numerous key agreement protocols [13, 3] that do not require a trusted third party. However they require high computation and do not provide flexibility that is required for ad hoc networks.

The paper is organised as follows. Section 1 provides an introduction to key management in ad hoc networks. In section 2, we give the preliminaries and in section 4, describe the new protocol. Analysis of the protocol is given in section 5. We conclude in section 6.

2. PRELIMINARIES

Chan's protocol is a probabilistic distributed construction of a cover-free family. A (w, r, d) Cover-Free Family (CFF) for a set of N points $\{p_1 \cdots p_N\}$ is a set of subsets, called *blocks*, such that union of any w blocks, will leave at least d points in the intersection of a set of r blocks, uncovered.

Chan's observation is that CFF property can be verified

via a distributed way by each node. Instead of using a well-designed CFF by a central authority to allocate keys to nodes, he allows nodes to randomly select key subsets. Nodes use a protocol to discover keys that they share with other nodes and check if CFF property holds for them. If the property holds for all nodes then it can be concluded that it is a successful construction of keys.

The Secure Key Discovery (SKD) phase in Chan's scheme uses a homomorphic encryption algorithm with certain properties. Chan proposed a homomorphic encryption protocol that satisfies these properties. In the following we give a more detailed descriptions of these results as they will be directly used in our protocol.

Homomorphic Encryption A homomorphic encryption system is an encryption system that satisfies the extra property that given encryptions $E(A)$ and $E(B)$ of two messages, output of a message $A*B$ that by $E(A*B)$ can be computed as

$$E(A * B) = E(A) * E(B)$$

where * is a group operation. Security of the encryption system uses the standard notions of security such as semantic security. Homomorphic encryption systems have been used in many protocols and allow operation on encrypted values without the knowledge of the plaintext or the keys.

Chan showed that for secure SKD, the privacy homomorphic system must satisfy the following properties.

1. Provide security against ciphertext-only attacks;
2. Given $E_K(x)$ and $E_K(y)$, encryption of $x + y$ is given by $E_K(x) + E_K(y)$;
3. Given $E_K(x)$ and a constant c, encryption of cx can be efficiently computed as $cE_K(x)$;
4. $E_K(0)$ must have a large number of different non-trivial representations.

Chan modifies a homomorphic encryption scheme due to Rivest et al[19] to ensure all the above properties.

Chan's Homomorphic Encryption

Choose two large primes p and q such that $n = pq$. The encryption is of the form $E_{p,q,r_i,s_i}(.) : \mathbb{Z}_n \rightarrow (\mathbb{Z}_n \times \mathbb{Z}_n)^l$ (where $1 \leq i \leq l$)

Encryption: To encrypt a message $m \in \mathbb{Z}_n$,

- Break the message m into l pieces (m_1, m_2 ,..., m_l) such that $m = \sum_{i=1}^{l} m_i \bmod n$
- Randomly choose $r_i < p$ and $s_i < q$, $1 \leq i \leq l$. r_i and s_i are kept secret.
- The encryption function $E_{p,q,r_i,s_i}(m) = ((m_1 r_1 \bmod p,\ m_1 s_1 \bmod q),\ (m_2 r_2 \bmod p,\ m_2 s_2 \bmod q),\ \ldots\ldots,\ (m_l r_l \bmod p,\ m_l s_l \bmod q) = ((x_1, y_1), (x_2, y_2), ..., (x_l, y_l))$

Decryption:

- Multiply components by the corresponding r_i^{-1} and s_i^{-1} respectively, i.e. $((x_1 r_1^{-1} \bmod p,\ y_1 s_1^{-1} \bmod q),\ (x_2 r_2^{-1} \bmod p,\ y_2 s_2^{-1} \bmod q),\ \ldots\ldots,\ (x_l r_l^{-1} \bmod p,\ y_l s_l^{-1} \bmod q)$
- Use Chinese Remainder Theorem to find (m_1, m_2, ... , m_l)
- Add up m_is to recover m.

Chan's Protocol Chan's Protocol consists of three phases: Distributed Key Selection (DKS), Secure Shared-key Discovery (SSD), and Key Exclusion Property Test (KEPT).

DKS The key pool P of size N is chosen and is partitioned into k parts. Each node chooses one key from each part.

SSD (We give more details as this protocol is the basis of SSD in our system.) Consider two nodes A and B with key rings given as ordered sets $A = \{a_1, ..., a_k\}$ and $B = \{b_1, ..., b_k\}$, respectively. The index of the key is the partition number that the key is drawn from. That is a_i is drawn from the i^{th} partition of the key pool. A and B perform the following steps in parallel. A does the following.

1. A creates a polynomial, $f_A(x) = (x-a_1)...(x-a_k)$ and sends the coefficients of the polynomial in encrypted form to B. The encryption algorithm $E_{K_A}(.)$ is the homomorphic encryption in section 2 where K_A is A's secret key.
2. B evaluates the polynomial (with encrypted coefficients) at its own key values and computes $z_i = E_{K_A}(rf_A(b_i))$, using a randomising factor r. Due to privacy homomorphism properties of the encryption scheme, the evaluation of encrypted polynomial at a point is the same as the encryption of the value of the polynomial at that point. B sends to A, z_i, $1 \leq i \leq k$.
3. A decrypts z_i using $D_{K_A}(.)$ and obtains $rf_A(b_i)$. If the value is zero, then A concludes that A and B have a shared key. Knowing the position of the zero allows A to know the block that the shared key belongs to. Finally she deduce the value of the shared key noting that she had only one key in each block.
4. A returns an k-bit bitmap with 1 at indexes where $rf_A(b_i) = 0$ and 0 elsewhere to B.

KEPT A node collects all k-bit bitmaps of other nodes in the system, forms an incidence matrix of keys that it shares with other nodes and verifies if the CFF property hold in its incidence matrix. If it does not it re-selects its key ring and repeats the SKD phase.

3. MODEL AND ASSUMPTIONS

We assume there is no trusted authority. Nodes run the protocol correctly. We consider two types of adversarial scenarios. Firstly, there are only outside eavesdroppers but the nodes are honest and behave correctly. In the second case nodes can be compromised and use these key information to eavesdrop on the communication of other nodes. In both cases system connectivity performance is measured in terms of the success probability of forming IG and the communication cost of establishing the session key for the whole group. Security performance in the former case is measured by the computation required to find the key of a secure link between two nodes, and in the latter case by the threat posed on secure the communication of other nodes. We will discuss the types of threats that needs to be considered in

this later case and possible strategies to contain them. To control the join of malicious nodes to the group, we assume some identification mechanism, device or user level, for the nodes.

For example in disaster rescue scenario, rescuers may have name cards or some other type of credentials. These can be used to identify or verify a node physically through a distance-limited channel such as infrared interfaces, similar to the secure side channel described in [5].

Notations We will use the following notations in the rest of the paper.

- $\mathcal{K}$: The key pool;
- $G = \{u_1, ..., u_n\}$: the set of nodes;
- n: Number of nodes;
- r: Size of IG;
- N: Number of keys in the key pool;
- B_i: the i^{th} key block in the pool;
- m: Number of keys in a key block;
- k: Number of blocks we have where $N = mk$ (Note that k also denotes the size of a user's key ring);
- l: Number of keys to be shared by two nodes to have a secure link between them;
- $\mathcal{N}_i$: The set of nodes that are in communication range of a node u_i;
- $\mathcal{N}_i^s$: A subset of $\mathcal{N}_i$ with secure link to u_i.

4. PROPOSED PROTOCOL

4.1 Overview of the protocol

The protocol consists of the following phases.

- **Initial Setup Phase**: Nodes agree on parameters used in the protocol. This includes the key pool given by the set of keys and its partition into blocks, and the level of link security, l.
- **Secure Shared Group Key setup phase**: Nodes obtain a shared group key. This phase has two steps.
  - Distributed key ring selection: Each node u_i selects a key ring $\{K_1^i, K_2^i, ..., K_k^i\}$.
  - Secure shared key discovery: Nodes perform a protocol to discover their shared keys with their neighbours. Two nodes with l shared keys, are assumed securely connected.
- **Session key establishment: Form IG, construct a session key and use the iterative steps to have a session key for the whole network.**

4.2 Protocol Description

4.2.1 Setup Phase

The key pool $\mathcal{K}$, its partition and parameters, N, m, and l, are chosen by nodes. 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 set up.

The key pool may be generated by applying a randomizing function such as a hash function, on integers $1..N$. The key pool is partitioned into k parts, each of size m ($N = m \times k$). Partitions are publicly known. It is sufficient to use blocks of the form $[i, i+m-1], i = 0, \cdots N/m$, and apply the function to integers in each range to generate the key blocks.

4.2.2 Shared Group Key Setup Phase

The aim of this phase is to provide a key pre-distribution that allows, (i) good chance of forming IG, and (ii) required security level for link communication. A set of nodes are (l, r)-*secure connected* if it satisfies the two required properties for IG. That is a set of r nodes such that each node is in the communication reach of all other nodes, and every two nodes have at least the same l keys in common.

Distributed Key Selection (DKS) Phase A node u_i randomly selects one key from each key block to form a key ring $\{K_1^i, K_2^i, ..., K_k^i\}$. All the parameters for this step are public and the step can be performed by any node.

Secure Shared-key Discovery (SSD) Phase Each node runs a SSD protocol (to be described later) with each of its neighbours. The result is used to construct an incidence matrix for the keys that the node shares with its neighbours. Note that this matrix only records shared keys in a local sense: that is with current neighbours of the node. For node u_i, the incidence matrix I^i has k columns labeled by the node keys $K_1^i, \cdots K_k^i$, and one row for each neighbour $I^i(j,t) = 1$ if K_t^i is shared with node u_j in the neighbourhood of u_i, and zero otherwise.

The matrix is used to check if an IG can be formed. This is by finding a set of r rows $\mathcal{R}$ and at least l columns $\mathcal{L}$ for which an (l, r)-secure subset can be formed. Note that if u_i can find $r-1$ neighbouring nodes with whom it shares the same set of l keys, then these nodes will have the same l keys shared with each other. However their neighbourhood condition must be separately verified.

SSD protocol is similar to Chan's protocol. However, we do not require the k-bit bitmaps to be sent out and so the adversary will have less information about the shared keys.

Node u_i needs to run SSD with each u_g in its neighbourhood. Let keys for u_i and u_g be ordered sets $K^{u_i} = \{K_1^{u_i}, ..., K_k^{u_i}\}$ and $K^{u_g} = \{K_1^{u_g}, ..., K_k^{u_g}\}$ respectively. The protocol for SSD is as follows:

- u_i constructs the polynomial: $f_{u_i}(x) = (x - K_1^{u_i})$ ... $(x - K_k^{u_i})$ and sends the encrypted coefficients to u_g using encryption algorithm $E_{K_{u_i}}(.)$ in section 2.
- u_g computes $z_i = E_{K_{u_i}}(r f_{u_i}(K_i^{u_g}))$ using of the homomorphism property, where r is a random number. u_g sends back z_i.

- u_i applies $D_{K_{u_i}}(.)$ to obtain $rf_{u_i}(K_i^{u_g})$. If the value is zero, then they have a common key.
- u_i then fills one row of its incidence matrix I_g^i with corresponding 1s and 0s.

4.2.3 Session Key Generation Phase

An IG is identified. We require that all members of IG contribute to the generation of the session key. This prevents selection of weak keys. To form a session key nodes in the IG do the following.

- For all $u_i \in IG$:
  - u_i randomly selects its key share s_i;
  - u_i computes the IG broadcast key as $K_{IG} = K_1^{IG} \oplus ... \oplus K_l^{IG}$ where $\{ K_1^{IG}, ..., K_l^{IG} \}$ are keys shared by the nodes in IG;
  - u_i encrypts $E_{K_{IG}}(s_i)$ and broadcasts to the IG;
- For all $u_j \in IG, j \neq i$, u_i decrypts $D_{K^{IG}}(s_j)$. The session key is calculated as $K_S = s_1 \oplus ... \oplus s_c$

The session key generated by the initiator group is propagated to other nodes using iterative step described below.

In an iterative step, nodes send the session key to all their neighbours that they have secure link with. The iteration stops when all nodes share the session key. Note that although the system is designed for an IG of size r, in practice a larger IG might be used (if exists). Let $R = G \setminus IG$ denotes the set of nodes that do not have the session key. C_t denotes the set of nodes that in the current group after t rounds of iteration. $C_0 = IG, R_0 = R, t = 0$

- for $u_i \in C_t$
  - u_i sends K_S (session key) to all its securely connected neighbours
  - $R_{t+1} = R_t \setminus \bigcup \mathcal{N}_{i^s}$
- until $R = \emptyset$

4.2.4 Join

Suppose a new node $\hat{u}$ wants to join an existing group. The two security conditions that need to be met are (1) ensuring that the new node is trustable, and (ii) security of group communication is ensured. If the nodes are assumed trusted, then the only security requirement is provision of *forward security*. That is ensuring that the new node will not have access to the communications in the group prior to its join. This can be achieved by renewing the session key of the whole group by applying a one way function to the current session key and then forwarding the new session key to $\hat{u}$ by one of its neighbours. It is always possible to re-run the whole session key generation and distribution protocol (guarantee of a completely new start).

If nodes cannot be trusted, we require an identification mechanism for nodes that provide some level of assurance about the identity of the new node. The new node $\hat{u}$ must identify itself to one of the existing nodes in its neighbourhood, denoted by u_0. u_0 verifies the identity of $\hat{u}$ and using the shared session key, sends this information to other nodes. This adds $\hat{u}$ to the list of authorised node identities maintained by all group members. Before sending the session key to the new node, u_0 updates the session key (similar to the previous case).

4.2.5 Node Revocation

In this case no other change is necessary. Node revocation may occur because of (i) malfunctioning of a node, and (ii) node capture by the adversary. In the former case the node will be simply removed from the list of nodes and its corresponding row in the key incidence matrix will be removed. In the latter case the key ring of the node is known by the adversary who will use this information to eavesdrop on other link communication. Once a malicious node is detected by its neighbours, the neighbours will jointly send revocation request to the rest of the group. The request includes the identification information of the malicious node. Note that it is important that revocation request be generated by more than one node as a request by a single node cannot be distinguished from a fraudulent request by a malicious node. Once the revocation requests are verified, the node will be marked *revoked* in the node identifier list of all nodes and no future communication with that node will be accepted. The keys in the key ring of a revoked node are compromised keys. However since link keys use at least l keys from the key pool, as long as the overlap between the set of compromised keys and keys forming a link key is not large, security of the link can be assumed intact. A node will use its key incidence matrix to determine the set of compromised keys (possibly from more than one revoked key) to determine security of its links with its neighbours under conditions such as collusion of revoked nodes.

5. ANALYSIS

5.1 Probability Analysis

Probability of forming initiator group

This is the probability of r nodes sharing at least l keys. A node randomly chooses one key from each key block. Let $P(r, 1)$ be the probability that exactly one key be shared by r nodes. There are k key blocks and so there are k possible block to choose the shared key from. For a block of size m, there are m possible choices for the key. The number of ways that r members select from a set of m keys is m^r. This means that

$$P(r, 1) = \frac{m}{m^r} = m^{1-r}$$

An initiator group must have l common keys. The common keys may come from $\binom{k}{l}$ possible subset of blocks. Then the probability that the r nodes share exactly l keys is:

$$P(r, l) = \binom{k}{l} P(r, 1)^l (1 - P(r, 1))^{k-l}$$

This is a Binomial Distribution with success probability $P = P(r, l)$ for l out of k successful events and p is $P(r, 1)$. If the probability p is small and the number k is large the distribution approaches the poisson distribution with $kP(r, 1) = \lambda$ being the expected value of having one common key for r nodes. Using this distribution we have

$$P\{X = l\} = \frac{\lambda^l e^{-\lambda}}{l!}, l = 0, 1, 2, ...$$

This gives probability of r nodes having l common keys in terms of probability r nodes having one common key. Let

$\mathcal{P}(l,r)$ denote the probability that r nodes share *at least* l keys. Then we have,

$$\mathcal{P}(l,r) = \sum_{l}^{k} P(r,l)$$

$\mathcal{P}(l,r)$ is the probability of establishing a secure link between nodes. In the following we make some observations on some graphs that show the relationship between various parameters described above.

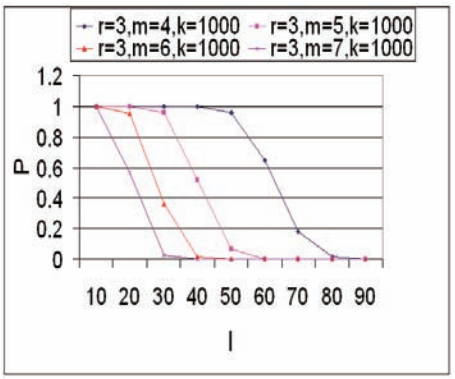

Figure 1: Relation between l and P for different m

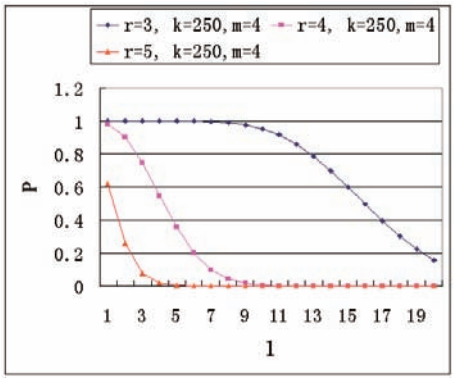

Figure 2: Relation between l and P for different r

Figure 1 graphs the relationship between l and $\mathcal{P}(l,r)$ for fixed r and k, and with varying m. It can be seen that for a fixed probability, a small increase in block size m results in large decrease in the number of shared keys l. Figure 2 gives the relationship between l and $\mathcal{P}(l,r)$ for fixed m and k, and with varying r. It can be seen that as the number of nodes in IG given by r increases, the probability of having l shared keys (forming IG) reduces. We note that this probability is for any set of r nodes. For example, $P(r,l) = 0.95$ means almost any r nodes share l keys. However as long as there is a single IG, the protocol can be completed. This means that as long as $\binom{n}{r}P(l,r)$ is bigger than 1, there will be a good chance of constructing an IG.

Number of iterative step In session key distribution, IG expands at each step. We consider the probability that nodes obtain the session key at a certain step. Let $R = G \setminus IG$ denote the set of nodes that do not have the session key, C_t denote the set of nodes that have the session key after t steps, and r_t denotes the number of nodes join the group in round t. The iteration start with $t = 1$ where the size of R is $n-r$. The number of nodes that join $C_1 = IG$ in the first round is r_1. The probability that two nodes share at least l keys is $\mathcal{P}(l,2) = \sum_{l}^{k} \mathcal{P}(l,2)$. The probability that one node has no connections to IG is $\tilde{P} = (1-\mathcal{P}(l,2))^r$. For nodes in R, the probability that r_1 (out of $n-r$) nodes have a link to IG the group is

$$P_1(r_1) = \binom{n-r}{r_1}\tilde{P}^{n-r-r_1}(1-\tilde{P})^{r_1}$$

Let $P_1(r_1) = 0.9999$. Solving the equation above, we obtain the number r_1 of nodes join the group in the first round. For the next round, the probability that one node has no connections to the group is $\tilde{P} = (1-\mathcal{P}(l,2))^{r+r_1}$. Figure 3, shows that for $\mathcal{P}(l,2) = 0.5$ more than half of nodes join the group in the first round. From the expression of $\tilde{P}$ we can see that $\tilde{P}$ becomes negligible if $|C_t|$ is greater than 50 (this is less than r_1 in above cases) for $\mathcal{P}(l,2) = 0.5$, which means nearly all other nodes join the group in the second round.

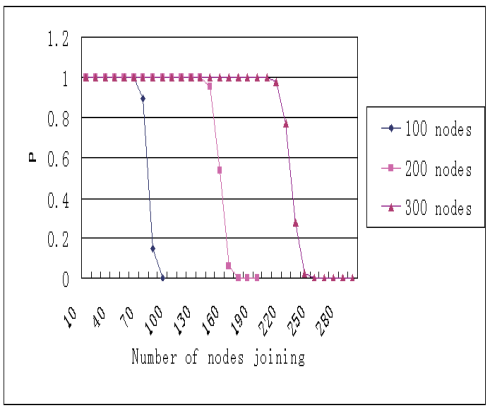

Figure 3: Number of nodes joining group in the first round

Connectivity From the Random-graph theory [11], we can analyse the effect of probability of two nodes have a link to the connectivity of the network. A random graph $G(n,p)$ is a graph of n nodes for which the probability that a link exists between two nodes is p. Given a desired probability P_c for graph connectivity, p is defined as

$$P_c = \lim_{n\to\infty} P_r[G(n,p) \text{ is connected}] = e^{-e^{-c}}$$

where

$$p = \frac{ln(n)}{n} + \frac{c}{n} \text{ and c is any real constant}$$

In our protocol, we require a initiator group to be formed with probability $\mathcal{P}(l,r)$. When $r = 2$, this is the same as p. For a certain probability of graph connectivity $P_c = 0.99999$. We can work out the value of c is about 11.5 as in [12]. This gives a small $p = 0.3$ for $n = 50$. Our protocol require more than two nodes to form initiator group, which implies that $p = \mathcal{P}(l,2)$ is close to 1 as in previous analysis. So we can guarantee to have full connectivity in the network.

Relation between n and k In the calculation of P, we do not consider the number of nodes n. All nodes may not share the same set of common keys from DKS. Suppose there are t out of n nodes share at least one common key. The probability is $P(t)$. Because there are k keys for each user, we denote the probability of the kth key is in common between t nodes as $P(k,t)$. $P(k,t)$ is independent. Then we have:

$$P(t) = 1 - (1 - P(k,t))^k$$

To solve $P(k,t)$, we need to find out all the combinations in a particular key block. In the DKS above, all nodes choose one key randomly from one key block. We can think this case as we group n nodes into m groups. If one of the groups has r nodes, then this is one successful event. We denote the number of all the possible combinations as $P(n,m)$, where n is total number of nodes, m is the size of key block, that is, number of groups.

To calculate $P(n,m)$, we can consider the following steps. Every user need to choose one key from m keys, so there are m cases. The choice of n nodes are independent, so there are total m^n cases. To count how many cases that have t nodes in one group, we consider the following. Suppose we pick one key to put all t nodes, there are m choices. Then we pick t out of n nodes to fill in that group. After that, the rest of them should be put into $m-1$ groups. The number of cases is then

$$m\binom{n}{t}(m-1)^{n-t}$$

So for the number of as least t nodes share a key is

$$\sum_{i=t}^{n} m\binom{n}{i}(m-1)^{n-i}$$

So

$$P(k,t) = \frac{\sum_{i=t}^{n} m\binom{n}{i}(m-1)^{n-i}}{m^n}$$

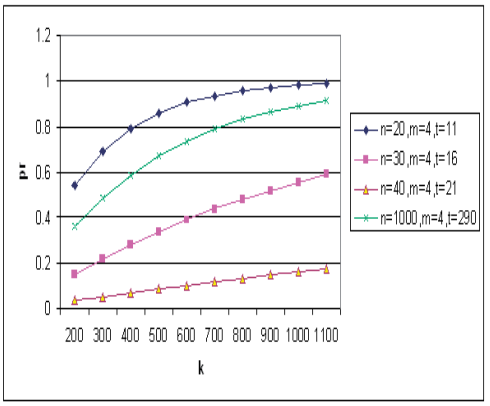

Figure 4: relation between n and k

Figure 4 shows that for a fix k, if we require t is more than 50% of n, the probability of t nodes share one key decreases as n increases (see lines with $n = 20, 30, 40$). If we let $n = t$, then $P(k)$ is equivalent to $P(r,1)$ as previously discussed. This implicitly verifies the correctness of our formula.

5.2 Security Analysis

Outside eavesdropper. Here we consider an outside eavesdropper who monitors messages passing in the network. The key rings of nodes are never revealed in the network. Messages in SSD phase are encrypted using a secure homomorphic encryption system and secrets that are chosen by nodes, and as long as the encryption system is secure, no information about the shared keys will be revealed. The messages in the session key generation and session key forwarding phases are encrypted using a semantically secure encryption algorithm and the link keys. Assuming security of the encryption algorithm, compromise of the system security requires the link keys to be found. In other words the adversary must know the keys (indices) that are shared between the two nodes forming a link.

There are $N = mk$ possible keys. The attacker knows there are l keys used to secure a link but does not know the blocks that the shared keys are coming from. So there are $\binom{k}{l}m^l$ possible choices for the attacker. With good choice of parameters, this will require an infeasible amount of computation and so the system will be secure. For example if $r = 3$, $m = 4$, $l = 20$, $k = 316$, we have the probability $\mathcal{P}(20,3) = 0.612$. There are $\binom{316}{20}4^{20} \approx 2^{150}$ possible choices which give sufficient level of security. In Eschenauer et al.'s scheme the security relies on the key length while in our case it is based on *the number of shared keys*. To compare the two systems, assume a security level of 100 bits (2^{100} search). This means the key length in Eschenauer et al.'s scheme is 100 bits, and in our system there are 2^{100} possibilities for the adversary to try. In Eschenauer et al.'s scheme if the size of the key pool is $10,000$ and each node has 75 keys, then the probability of two nodes sharing one key is 0.5 [12]. If we want to have the same number of keys per node, we can choose $m = 3, k = 75, l = 25, r = 2$. In this case the probability that two nodes share l keys will be 0.54 which is slightly higher than Eschenauer et al.'s scheme. The adversary needs to try $\binom{75}{25}3^{25} \approx 2^{104}$ combinations and so with smaller key rings the same level of security (100 bits) can be achieved.

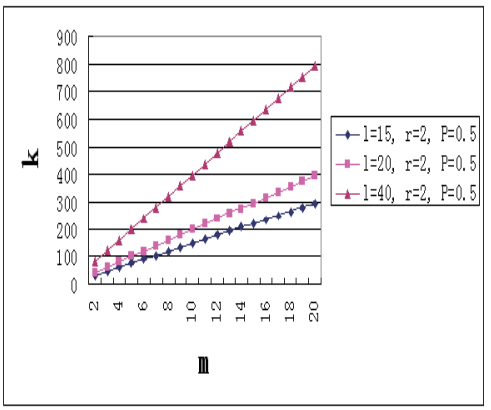

Figure 5: relation between m and k

Compromised nodes. If nodes are not trusted then to join the group, the nodes have to identify themselves using distance-limited channels ensuring physical contact with one

of the existing nodes, or use other identification mechanism such as a SIM card. neighbours around the node should be able to detect. Then the We assume compromised node is detectable by its neighbours and the identity of the node will be sent to all other nodes. The effect of a node compromise on keys and link security was discussed in Section 4.2.5.

Figure 1 and figure 2 show that if r is reduced, for example, from 4 to 3, then it is very likely to have more shared keys. This means that the IG may use a broadcast channel which uses more key from the key pool and so will be harder to find. A similar comment applies to link key.

Our protocol has generalised parameters m, k, ℓ, r. If $r = 2$, then our protocol is similar to pairwise key schemes [12, 10, 15]. However, our scheme does not require completing shared key discovery with all nodes. The session key will propagate through the network to allow anyone to communicate with each other. For pre-deployed sensor networks, sensor nodes may be strained to move in a designated area and contact a powerful controller for cross area communication. Our protocol can easily handle this case without such a controller. To be secure under collusion attack, when $r = 2$, we must have a large ℓ to prevent exhaust search of shared keys between two legitimate nodes.

Analysis of revocation As in section 4.2.5, we need to re-select keys that shared by more than t nodes in order to prevent collude attacks. By choosing appropriate parameters, we can have large ℓ to prevent exhaust search. Here we let $r = 2$. If there are not enough shared keys between two nodes, the link is then lost. We can require a large ℓ to avoid losing all links. This means that we need a large k. However, when there are more than one nodes revoked, the links between two nodes may be lost. We perform experiments to examine the effect of node revoking. Let $m = 4, n = 100, k = 500, t = 35$. This setting allows probability of 0.98 for $r = 3, l = 20$. We create 100 nodes and randomly select keys for them. After that, every node build a matrix of shared keys. Then we revoke some of the nodes. We do experiments for revoking 1,3,7,15,48 nodes respectively. For a chosen node, we record the number of shared keys with all other nodes, the number of total marked keys, the number of marked keys in shared keys (affected keys) and the number of effective keys. The numbers recoded are the average values of all current linked nodes. From Figure 6, we can see that the number of shared keys between two nodes does not change much after revocation. However, the number of effective shared keys does reduce to a smaller number. The line of marked keys indicates the total number marked key for a node. We can see that if more than 15 nodes collude together, many key blocks are exposed (430). There are only 17 effective keys, which means the attack can guess $(500-430)^{17} \approx 2^{100}$ times to break the shared keys. This still can be considered practically secure. As we can see, even if more nodes are revoked, the lines in figure 6 keep at stable levels. This is because every time a node is revoked, some keys that shared by revoked nodes are re-selected. So that the number of effective keys will become a constant when more nodes are revoked.

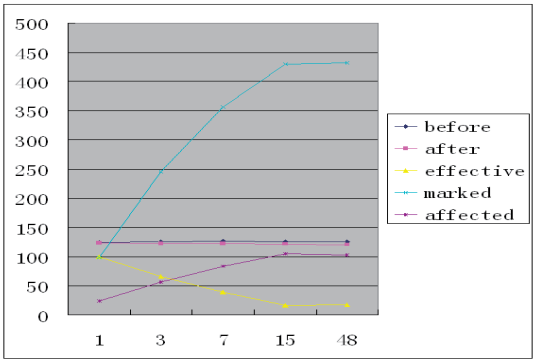

Figure 6: effect of revoking

Perfect forward security. A new node should not know about past session keys. That is perfect forward security [17]. More precisely, perfect forward secrity means that a compromise of a members long-term key cannot lead to the compromise of any short term group keys (session keys). When a new node joins the network, existing members select a new session key and send the session key to the new node. However, in some cases, nodes need to know about past information or it is ok to let new nodes use existing session key. So this renewal of session key can be arbitrary determined by current nodes in the network.

5.3 Efficiency Analysis

In our scheme, nodes in the network need to store k keys. To exchange key information, $k + 1$ encryption operations which involve calculating coefficients and k decryption operations for verifying k keys are needed. Also, k^2 operations for responding to each of exchange requests are required. Hence, $k^2 + 2k + 1$ operations are needed for each key information exchange. Consequently, $n(k^2 + 2k + 1)$ operations are required to be exchanged over the network among n nodes. Note that these operations require computation of homomorphic encryption, which is the main computationally intensive part of the scheme. Although key information exchange needs many computations, nodes do not need to complete all the exchanges before they join the network. As long as it has some other nodes' support, a new node can join the network and keep exchanging key information later on. This way, the computational overhead is relieved.

Now we compare the efficiency of our scheme to that of Eschenauer et al.'s scheme (EG) and Kim et al.'s [13] Constant-Round Authenticated Group Key Exchange (CRAGKE) scheme. Note that Eschenauer et al.'s scheme is the first one that uses the results from random graph theory and probabilistic method to manage keys in ad hoc network. CRAGKE is the latest group key agreement scheme that achieves provable security for authenticated group key establishment (AGKE), based on Computational Diffe-Hellman (CDH) assumption, with 2 round broadcasts.

Before comparing efficiency of those two schemes with ours, we note that these proposals have different assumptions. However, the common feature among the three proposals is that the group formation is dynamic, user can join or leave the group (or be revoked from the group) at any time. Table 1 summarises the assumptions that three different proposals make as to whether they require Centralised Control (CC); Initiator Group (IG); initial keys (IK); Trust Third Party (TTP); Authentication (AU), and whether nodes are ordered (ON) in a loop that is cyclic or not.

	EG	CRAGKE	our protocol
CC	yes	not always used	no
IG	no	no	yes
IK	yes	no	chosen by nodes
TTP	offline	offline	no
ON	no	cyclic	no
AU	no	yes	no

Table 1: Comparison of assumptions

Finally, efficiency of the three proposals are compared in Table 2. Specifically, we compare Storage (ST), Processing Time (PT), and Join Cost (JC) of the three proposals. In this table, n denotes the total number of nodes; k denotes the number of keys that a node has; n_j denotes the number of nodes joining the group; e denotes the cost of modular exponentiation; h denotes the computation of hash function; s denotes the cost of signature; p denotes the cost of Homomorphic encryption/decryption; a denotes the cost of addition on encrypted data of Homomorphic encryption. We ignore the cost of XOR operation in CRAGKE.

Note that 2 round broadcasts and 2 round information exchanges are needed in CRAGKE and our scheme respectively. In Eschenauer and Gligor's protocol, one node needs to broadcast its k key identifiers. In CRAGKE, a node broadcasts one message in each round (message is the result of processing). Our protocol requires one node to send $k+1$ encrypted coefficients, receive k responses and send k^2 responses.

	EG	CRAGKE	our protocol
ST	k	4	k
PT	kn	$2s+3e+4h$	$n((2k+1)p+ak^2)$
JC	N/A	$2s+3e+4h$	$n_j((2k+1)p+ak^2)$

Table 2: Efficiency comparison

Note also that when k gets larger in our protocol, the computation cost becomes high. Although the cost for computing homomorphic encryption has not been compared to exponentiation, we believe it is a time consuming operation. So our protocol is more time consuming compared to other schemes in terms of processing time. Although it has worse performance, it has advantages over other two protocols in terms of truly self-organised ability and flexibility.

5.4 Scalability and Mobility Issues

Scalability Issue. Even though the total number of nodes is unknown, our protocol can dynamically adjust parameters to accommodate more nodes effectively. Notice that since the total number of nodes n affects the number of key re-selections in the revocation phase from our analysis given in section 5.1, the number of keys stored by each node k depends on n. If n is small at the beginning, nodes can choose a small value for k to save key ring information exchange cost. When n becomes large, k can also be increased. All nodes only need to exchange information for those extra keys. This approach further reduces the cost for setup phase.

Mobility Issue. If a node moves into another range of communication during key information exchange with other nodes, it can resume later on when it moves back into the original range of communication. After the network is formed, nodes can still move around as long as there are other nodes around to pass messages. In fact, this mobility enhances the security of the system. While they are moving around, nodes can have chances to contact other nodes visually, which increases the reliability of the network.

6. CONCLUDING REMARKS

We presented a new key management protocol for practical ad hoc networks. The protocol is self-organised, scalable and flexible. We used an iterative process in a 'greedy' manner, to distribute the session key to all nodes. Group formation may use other policies that determine who can be a group member i which case extra conditions need to be checked during the session key forwarding step.

Further studies are needed to reduce the computational overhead. We could have further studies to allow big initiator group. If there is more than r nodes in one hop range before the network setup, for example, there are R nodes, $R > r$, in one hop range, then they can all choose their own shares for the group, not just leave this to r nodes. R nodes can send their shares to all other nodes through all the possible r-user groups. Recall that $\mathcal{P}(l, r)$ is for any r nodes to share l keys, there are always r-user groups to support this operation if P is relatively high. This further increases the security of session key generation. For example, as shown in figure 7, there are 7 nodes. If $r = 3$, $m = 4$, $l = 60$, $k = 1000$, we have $P = 0.64$. There are $\binom{7}{3}$ 3-user groups. So there are $35 \times 0.64 = 22$ such groups. Actually, at most $R - r + 1 = 7 - 3 + 1 = 5$ such groups can connect all nodes together. How to efficiently handle key re-generation in revocation phase is another possible direction for further study.

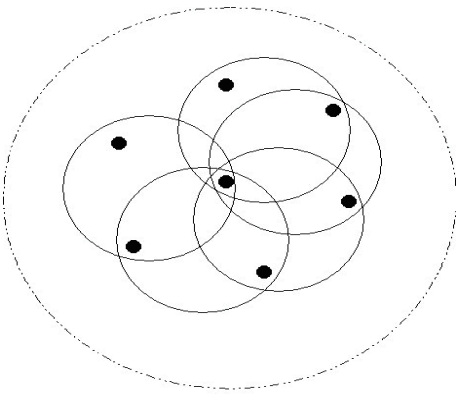

Figure 7: Number of users in the same range increase initiator group size

7. ACKNOWLEDGMENTS

The authors are grateful to anonymous referees for the valuable comments.

8. REFERENCES

[1] R. Blom. An optimal class of symmetric key generation systems. In *Proc. of the EUROCRYPT 84*

workshop on Advances in cryptology: theory and application of cryptographic techniques, pages 335–338, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[2] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably authenticated group Diffie-Hellman key exchange. In P. Samarati, editor, *Proc. of ACM-CCS 01*, page 255C264, Philadelphia, Pennsylvania, USA, November 2001. ACM, ACM Press.

[3] M. Burmester and Y. Desmedt. Efficient and secure conference-key distribution. In *Proceedings of the International Workshop on Security Protocols*, pages 119–129, London, UK, 1997. Springer-Verlag.

[4] P. C. *Ad Hoc Networking*. Addison-Wesley Newyork, 2001.

[5] S. Capkun, J.-P. Hubaux, and L. Buttyan. Mobility helps security in ad hoc networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 46–56, New York, NY, USA, 2003. ACM Press.

[6] D. W. Carman, P. S. Kruus, and B. J. Matt. Constraints and approaches for distributed sensor network security. Technical report, NAI Laboratories, 2000.

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

[8] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, page 197, Washington, DC, USA, 2003. IEEE Computer Society.

[9] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[10] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(2):228–258, 2005.

[11] P. Erdös and A. Rényi. On the evolution of random graphs. Publications of the Mathematical Institute of the Hungarian Academy of Sciences, 1960.

[12] L. Eschenauer and V. Gligor. A key management scheme for distributed sensor networks, 2002.

[13] H.-J. Kim, S.-M. Lee, and D. H. Lee. Constant-round authenticated group key exchange for dynamic groups. In *Asiacrypt 2004*, 2004.

[14] J. Kohl and B. Neuman. The kerberos network authentication service(v5). RFC1510, September 1993.

[15] D. Liu, P. Ning, and R. Li. Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(1):41–77, 2005.

[16] T. Matsumoto and H. Imai. On the key predistribution system: A practical solution to the key distribution problem. In *CRYPTO*, pages 185–193, 1987.

[17] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996.

[18] C. J. Mitchell and F. C. Piper. Key storage in secure networks. *Discrete Appl. Math.*, 21(3):215–228, 1988.

[19] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–179. Academic Press, 1978.

[20] 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/.