Scalable and Deterministic Key Agreement for Large Scale Networks

Yun Zhou, *Student Member, IEEE*, and Yuguang Fang, *Senior Member, IEEE*

***Abstract*— Key agreement is a central problem to build up secure infrastructures for networks. Public key technology may not be suitable in many networks of low-end devices, such as ad hoc networks and sensor networks, because of its computation inefficiency and the lack of central authorities in those distributed scenarios. Conventional distributed symmetric key agreement schemes lack scalability due to their large memory cost ($\mathcal{O}(N)$, where N is the total number of nodes), and their probabilistic nature cannot ensure key agreement between every pair of nodes. In this paper, we propose a novel symmetric key agreement scheme, which is scalable for large scale networks with very small memory cost per node. A t-degree $(k+1)$-variate symmetric polynomial is used to achieve key agreement between nodes. The memory cost per node for a network of N nodes is reduced to around $\sqrt[k+1]{\frac{k(k+1)!}{2}}\sqrt[k]{N}$, where $k \geq 1$. Our scheme is also deterministic in that every pair of nodes can establish a shared key.**

***Index Terms*— Public key, security, symmetric key.**

I. Introduction

TO secure end-to-end communications in a network, a shared key is required between two end nodes to support basic secure primitives, such as encryption and authentication. Hence, the two-party key agreement is critical to build up secure infrastructures for network communications. In large scale networks, such as ad hoc and sensor networks, the key agreement is very difficult. Considering the node resource constraints and the lack of fixed on-line authorities, it is not suitable to directly apply public key techniques without any progress in simple and fast algorithms. Hence, symmetric key techniques [1]–[28] are investigated in the literature because of their efficiency.

Generally, each node keeps a set of secrets and uses those secrets to establish shared keys with other nodes. This approach has a *scalability* problem. The more nodes with which a node wants to share keys, the more secrets the node needs to keep. In ad hoc networks or sensor networks, which can consist of hundreds or thousands nodes, the memory cost of each node for key establishment can be rather large, which is unaffordable for low end devices.

To achieve key agreement in large networks, most schemes [4]–[22] follow a *probabilistic* approach. Particularly, each node keeps a small subset of secrets such that a pair of neighboring nodes establishes a shared key with a certain probability. These schemes, however, can not guarantee that every pair of nodes will establish a shared key due to their probabilistic nature. In addition, their memory cost increases linearly with the network size [23].

In this paper, we propose a novel key agreement scheme, which is *scalable* for large networks with small memory cost per node. A t-degree $(k+1)$-variate symmetric polynomial is used here to achieve key agreement between nodes. We will show that for a network of N nodes our scheme has around $\sqrt[k+1]{\frac{k(k+1)!}{2}}\sqrt[k]{N}$ memory cost per node, where $k \geq 1$. Our scheme is also *deterministic* in that any pair of nodes can calculate a shared key independently or negotiate one through $k-1$ ($k \geq 1$) intermediate nodes, called *agents*.

The rest of the paper is organized as follows. Section II describes the mathematical tool used in our scheme. Details of our scheme are given in Section III. Analysis is carried out in Section IV, and some comparisons with conventional schemes are discussed in Section V. A method to enhance security is described in Section VI. The paper is finally ended in Section VII.

II. Mathematical Tool

Our scheme is based on a t-degree multivariate symmetric polynomial. A t-degree $(k+1)$-variate polynomial is defined as

$$f(x_1, x_2, \ldots, x_k, x_{k+1}) = \sum_{i_1=0}^{t} \sum_{i_2=0}^{t} \cdots \sum_{i_k=0}^{t} \sum_{i_{k+1}=0}^{t} a_{i_1,i_2,\ldots,i_k,i_{k+1}} x_1^{i_1} x_2^{i_2} \cdots x_k^{i_k} x_{k+1}^{i_{k+1}} . \quad (1)$$

All coefficients of the polynomial are chosen from a finite field $\mathbb{F}_q$, where q is a prime that is large enough to accommodate a cryptographic key. Without specific statement, all calculations in this paper are performed over the finite field $\mathbb{F}_q$.

A $(k+1)$-tuple permutation is defined as a bijective mapping

$$\sigma : \quad [1, k+1] \longrightarrow [1, k+1] . \quad (2)$$

By choosing all the coefficients according to

$$a_{i_1,i_2,\ldots,i_k,i_{k+1}} = a_{i_{\sigma(1)},i_{\sigma(2)},\ldots,i_{\sigma(k)},i_{\sigma(k+1)}} \quad (3)$$

for any permutation σ, we can obtain a symmetric polynomial in that

$$f(x_1, x_2, \ldots, x_k, x_{k+1}) = f(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(k)}, x_{\sigma(k+1)}) . \quad (4)$$

Manuscript received May 1, 2006; revised January 1, 2007 and February 1, 2007; accepted March 1, 2007. The associate editor coordinating the review of this paper and approving it for publication was M. Guizani. This work was supported in part by the US National Science Foundation under grants CNS-0626881 and CNS-0716450.

The authors are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: {yzufl, fang}@ece.ufl.edu).

Digital Object Identifier 10.1109/TWC.2007.06088.

At first, every node should have k credentials, which are *positive* and *pairwise different* integers. Suppose node u has credentials $(u_1, u_2, \ldots, u_k)$ and node v has credentials $(v_1, v_2, \ldots, v_k)$. Before node deployment, we can assign a *polynomial share* $f(u_1, u_2, \ldots, u_k, x_{k+1})$ to u and another share $f(v_1, v_2, \ldots, v_k, x_{k+1})$ to v. By assigning polynomial shares, we mean that the coefficients of t-degree univariate polynomials $f(u_1, u_2, \ldots, u_k, x_{k+1})$ and $f(v_1, v_2, \ldots, x_{k+1})$ are loaded into nodes u's and v's memory, respectively.

If the credentials of node u and node v have only one element different, i.e.,

1) for some $i \in [1, k]$, $u_i \neq v_i$, and
2) for $j = 1, 2, \ldots, k$, $j \neq i$, $u_j = v_j = c_j$,

then node u and node v can have a shared key. Node u can take v_i as the input to its own share $f(u_1, u_2, \ldots, u_k, x_{k+1})$, and node v can also take u_i as the input to its share $f(v_1, v_2, \ldots, v_k, x_{k+1})$. Due to the polynomial symmetry, the desired shared key between nodes u and v has been established as

$$\begin{aligned} K_{uv} &= f(c_1, c_2, \ldots, c_{i-1}, u_i, c_{i+1}, \ldots, c_k, v_i) \\ &= f(c_1, c_2, \ldots, c_{i-1}, v_i, c_{i+1}, \ldots, c_k, u_i)\,. \end{aligned} \quad (5)$$

In fact, node u and node v achieve the key agreement by a marginal t-degree bivariate polynomial, i.e.,

$$f_i(x_i, x_{k+1}) = f(c_1, c_2, \ldots, c_{i-1}, x_i, c_{i+1}, \ldots, c_k, x_{k+1})\,. \quad (6)$$

where $i \in \{1, 2, \ldots, k\}$.

III. Key Agreement

Our key agreement scheme has three components, i.e. *share distribution*, *direct key calculation*, *indirect key negotiation*. In the share distribution part, partial information of a global t-degree $(k+1)$-variate polynomial is distributed among nodes. All the partial information cannot reveal the global polynomial but can help key agreement between nodes. Some nodes may calculate a shared key directly if they have some partial information in common. The indirect key negotiation part tells how to negotiate a shared key between two nodes with help of other nodes if they cannot calculate a direct key.

A. Network Model

We assume each node is identified by an index-tuple $(n_1, n_2, \ldots, n_k)$, where $n_i = 0, 1, \ldots, N_i - 1, i \in \{1, 2, \ldots, k\}$, and we may use the index-tuple as the *node ID*. Hence each node is mapped into a point in a k-dimension space $\mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_k$, where $n_i \in \mathcal{S}_i \subset \mathbb{Z}$ and the cardinality $|\mathcal{S}_i| = N_i$, for $i = 1, 2, \ldots, k$. The maximum number of nodes that the network can consist of is $N = \prod_{i=1}^{k} N_i$.

Our scheme targets at the key agreement between two end nodes. Hence we assume the underlying routing protocol can provide connectivity between any pair of nodes in the network.

B. Adversary Model

Due to the broadcast characteristics of radio communications, adversaries can easily eavesdrop any messages, either non-encrypted or encrypted, transmitted over the air between nodes. Moreover, due to cost constraints, it is also unrealistic and uneconomical to employ tamper-resistant hardware to secure the cryptographic materials in each individual node. Hence adversaries may capture any node and compromise the secrets stored in the node. Furthermore, adversaries can use the compromised secrets to derive more secrets shared between other non-compromised nodes. It means that the node compromise attack is unavoidable. What we can do is to reduce the impact on other normal nodes as much as possible. In our scheme, we try to reduce the probability that the keys shared between non-compromised nodes are exposed when some nodes have already been compromised. To further evaluate the impact of node compromise, we assume the probability of the compromise of a node is p.

C. Share Distribution

Before network deployment, a global t-degree $(k+1)$-variate symmetric polynomial is constructed as stated in Section II. This polynomial is used to derive shares for sensor nodes.

To achieve key agreement, every node n should have k credentials $(c_1, c_2, \ldots, c_k)$, which are positive and pairwise different as required in Section II. These credentials can be created and preloaded into nodes before deployment. However, it requires additional memory space per node. Fortunately, the k credentials can be derived from the k indices in node ID $(n_1, n_2, \ldots, n_k)$ by a bijection, i.e.,

$$\begin{cases} c_1 = n_1 + 1 \\ c_2 = n_2 + 1 + N_1 \\ c_3 = n_3 + 1 + N_1 + N_2 \\ \quad\vdots \\ c_{k-1} = n_{k-1} + 1 + N_1 + \cdots + N_{k-2} \\ c_k = n_k + 1 + N_1 + \cdots + N_{k-1} \end{cases}, \quad (7)$$

where $n_i = 0, 1, \ldots, N_i - 1$ for $i = 1, 2, \ldots, k$. Thus, the k credentials are drawn from different zones in that $c_1 \in [1, N_1]$ and $c_i \in [N_1 + \cdots + N_{i-1} + 1,\ N_1 + \cdots + N_i]$ for $i = 2, \ldots k$, which guarantee they are positive and pairwise different (Fig. 1).

For a node $(n_1, n_2, \ldots, n_k)$, a *polynomial share*

$$f_{k+1}(x_{k+1}) = f(c_1, c_2, \ldots, c_k, x_{k+1}) = \sum_{i_{k+1}=0}^{t} b_{i_{k+1}}\, x_{k+1}^{i_{k+1}} \quad (8)$$

is calculated, where

$$b_{i_{k+1}} = \sum_{i_1=0}^{t} \sum_{i_2=0}^{t} \cdots \sum_{i_k=0}^{t} a_{i_1, i_2, \ldots, i_k, i_{k+1}}\, c_1^{i_1} c_2^{i_2} \cdots c_k^{i_k} \quad (9)$$

and $(c_1, c_2, \ldots, c_k)$ is mapped from $(n_1, n_2, \ldots, n_k)$ according to the equations (7). Obviously, the share is a t-degree univariate marginal polynomial of the global polynomial and has $t+1$ coefficients. Then the polynomial share is assigned to the node. Here, the node only knows the $t+1$ coefficients of the univariate polynomial share, but not the coefficients of the original $(k+1)$-variate polynomial. Therefore, even if the marginal bivariate polynomial is exposed, the global polynomial is still safe if the degree t is chosen properly.

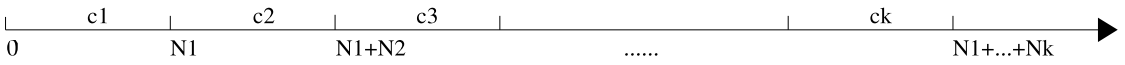


Fig. 1. Construction of positive and pairwise different credentials according to Equation (7).

D. Direct Key Calculation

According to Section II, two nodes can calculate a shared key if their credentials have $k-1$ elements in common. Due to the one-to-one mapping in the equations (7), two nodes u with ID $(u_1, u_2, \ldots, u_k)$ and v with ID $(v_1, v_2, \ldots, v_k)$ can directly calculate a shared key without any interaction if their IDs have $k-1$ indices in common.

Suppose that the i-th indices of their IDs are different. Then node u can take $v_i + 1 + N_1 + \cdots + N_{i-1}$ as the input to its own share $f(c_1, c_2, \ldots, c_k, x_{k+1})$, and node v can as well take $u_i + 1 + N_1 + \cdots + N_{i-1}$ as the input to its share $f(c_1, c_2, \ldots, c_k, x_{k+1})$. Due to the polynomial symmetry, the desired shared key between nodes u and v has been established as

$$\begin{aligned} K_{uv} &= f(c_1, \ldots, u_i + 1 + N_1 + \cdots + N_{i-1}, \\ &\qquad \ldots, c_k, v_i + 1 + N_1 + \cdots + N_{i-1}) \\ &= f(c_1, \ldots, v_i + 1 + N_1 + \cdots + N_{i-1}, \\ &\qquad \ldots, c_k, u_i + 1 + N_1 + \cdots + N_{i-1}) \,. \end{aligned} \tag{10}$$

Because all node credentials of u and v are drawn from different subspaces where any two subspaces have no intersection and $u_i \neq v_i$, the $k+1$ credentials used to calculate the shared key are pairwise different. Therefore the shared key calculated by the nodes u and v is unique, i.e., other nodes do not know the shared key. Any two nodes can directly calculate a unique shared key without any negotiation if there is only one mismatch between their k-tuple IDs.

E. Indirect Key Negotiation

If two nodes have more than one mismatch between their IDs, they cannot calculate a shared key directly. However, they can rely on some intermediate nodes as *agents* to negotiate a shared key.

Suppose two nodes u and v have j $(j \geq 2)$ mismatches in their IDs. For simplicity, let us omit all the same indices and mark the two nodes with those mismatching indices, say node u

$$(u_{i_1}, u_{i_2}, \ldots, u_{i_j})$$

and node v

$$(v_{i_1}, v_{i_2}, \ldots, v_{i_j}) \,,$$

where $i_1, i_2, \ldots, i_j \in [1, k]$ and are pairwise different. Then they can negotiate a shared key along a *secure path* consisting of agents as

$$\begin{gathered} (v_{i_1}, u_{i_2}, u_{i_3}, \ldots, u_{i_{j-1}}, u_{i_j}) \,, \\ (v_{i_1}, v_{i_2}, u_{i_3}, \ldots, u_{i_{j-1}}, u_{i_j}) \,, \\ (v_{i_1}, v_{i_2}, v_{i_3}, \ldots, u_{i_{j-1}}, u_{i_j}) \,, \\ \vdots \\ (v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_{j-1}}, u_{i_j}) \,, \end{gathered}$$

because all neighboring nodes along the path have direct keys. It is worth noting that there are many secure paths between node u and node v. Another example is

$$\begin{gathered} (u_{i_1}, u_{i_2}, u_{i_3}, \ldots, u_{i_{j-1}}, v_{i_j}) \,, \\ (u_{i_1}, u_{i_2}, u_{i_3}, \ldots, v_{i_{j-1}}, v_{i_j}) \,, \\ \vdots \\ (u_{i_1}, u_{i_2}, v_{i_3}, \ldots, v_{i_{j-1}}, v_{i_j}) \,, \\ (u_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_{j-1}}, v_{i_j}) \,. \end{gathered}$$

The existence of multiple paths indicates the strong resilience of our scheme in the face of node compromise.

IV. Analysis

In this section, we will carry out the analysis of our scheme when two nodes have j $(j \geq 2)$ mismatches in their IDs.

A. Number of Secure Paths

The number of secure paths can be calculated as follows. Each secure path is constructed in j steps. Begin from $(u_{i_1}, u_{i_2}, u_{i_3}, \ldots, u_{i_{j-1}}, u_{i_j})$. At each step one of the indices is replaced with the corresponding one from $(v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_{j-1}}, v_{i_j})$, and thus we can get an agent at the step. At the first step, any of the j indices of node u may be replaced, so there are j choices. The second step has $j-1$ choices. At the j-th step, there is only one choice left. Hence, the total number of secure paths can calculated as

$$P = j \cdot (j-1) \cdot \cdots \cdot 2 \cdot 1 = j! \,. \tag{11}$$

B. Number of Disjoint Secure Paths

Out of the P secure paths some are disjoint, i.e., any two disjoint paths have no common agent nodes except the two end nodes u and v. For nodes u and v which have j mismatches in their IDs, the number of agent nodes that are the neighbors of the end nodes u or v is j. Hence the number of disjoint secure paths is

$$P_d = j \,. \tag{12}$$

C. Number of Agent Nodes

For nodes u and v who have j mismatches in their IDs, each agent node along a secure path between the two nodes has an ID constructed in the following way. Randomly select l positions from j mismatches between u's and v's IDs, draw indices from u's ID at those positions, and draw indices from v's ID at the positions that are not selected. The ID of the agent node consists of the two sets of selected indices and the common indices between u's and v's ID. Hence the number of agent nodes can be calculated as

$$A = \binom{j}{1} + \binom{j}{2} + \cdots + \binom{j}{j-1} = 2^j - 2 \,. \tag{13}$$

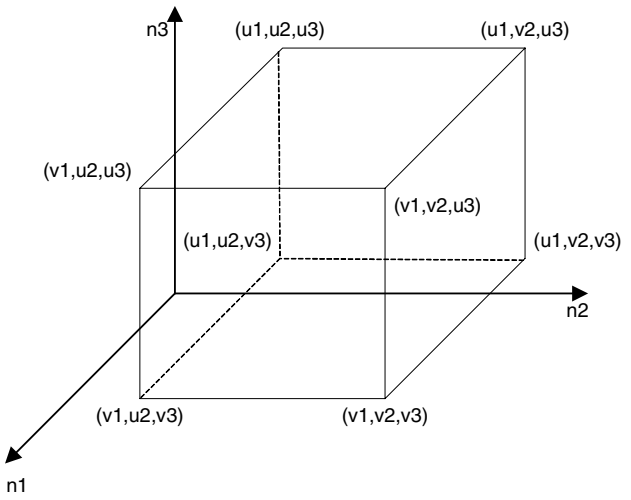


Fig. 2. An example of a key graph, where nodes (u_1, u_2, u_3) and (v_1, v_2, v_3) and all 6 agent nodes form a cube in the 3-dimension ID space.

D. An Example

An example of 3-dimension ID space is given in Fig. 2. Suppose node (u_1, u_2, u_3) needs to establish a shared key with node (v_1, v_2, v_3), where all 3 indices in their IDs are mismatching. They can determine 6 agent nodes. All these 8 nodes form a cube in the 3-dimension ID space. There are 6 paths from node u to node v, in which 3 are disjoint. For example, 3 disjoint paths are

$$(u_1, u_2, u_3) \rightarrow (v_1, u_2, u_3) \rightarrow (v_1, v_2, u_3) \rightarrow (v_1, v_2, v_3) \ ,$$

$$(u_1, u_2, u_3) \rightarrow (u_1, u_2, v_3) \rightarrow (v_1, u_2, v_3) \rightarrow (v_1, v_2, v_3) \ ,$$

and

$$(u_1, u_2, u_3) \rightarrow (u_1, v_2, u_3) \rightarrow (u_1, v_2, v_3) \rightarrow (v_1, v_2, v_3) \ .$$

Obviously, the above set of disjoint paths is not unique.

E. Security of Direct Keys

All nodes in the network hold partial information of one t-degree $(k+1)$-variate polynomial to achieve key agreement. During the network lifetime, some nodes may be compromised and then collaborate to expose the polynomial with the partial information they hold whereby to directly calculate keys between other nodes. Obviously, the polynomial degree t is an indication of the difficulty to expose the polynomial, and it is directly related to the security performance. By choosing the value of t properly, we can guarantee that no matter how many nodes are compromised, their collaboration cannot expose direct keys held between other non-compromised nodes. In this section, we will investigate how to choose the polynomial degree.

1) Node Compromise in One Subspace: Let us consider the malicious collaboration in one subspace. Though in this case the collaboration can only expose the direct keys between the non-compromised nodes in the same subspace, this is the easiest attack because adversaries only need to keep compromising the nodes in one subspace. If they randomly choose a node to compromise, they have to compromise more nodes to find all nodes in one subspace, which can require more efforts.

Suppose there are N_i nodes in the subspace $\mathcal{S}_i$, in which all nodes have same ID indices in other subspaces, for $i = 1, 2, \ldots, k$. Any pair of nodes in $\mathcal{S}_i$ can achieve key agreement with a t-degree bivariate polynomial $f_i(x_i, x_{k+1})$, which is the marginal of the global t-degree $(k+1)$-variate polynomial $f(x_1, \ldots, x_i, \ldots, x_k, x_{k+1})$ (refer to Section II). It has been shown in [3] that a t-degree bivariate polynomial is *t-secure* in that the coalition between less than $(t+1)$ nodes holding shares of the t-degree bivariate polynomial cannot reconstruct it. To guarantee any pair of nodes in S_i have a direct key that is unsolvable by the other $N_i - 2$ nodes, an $(N_i - 2)$-secure bivariate polynomial should be used. Hence, the degree of polynomial should satisfy

$$0 \leq N_i - 2 \leq t \ , \qquad i = 1, 2, \ldots, k \ . \tag{14}$$

2) Node Compromise in all Subspaces: Even when all nodes in one subspace are corrupted, they cannot expose the global t-degree $(k+1)$-variate polynomial because they only know a marginal of the global polynomial. In order to expose the direct key belonging to any pair of non-compromised nodes, adversaries must compromise enough nodes in all subspaces to expose the global polynomial.

Suppose all N_i nodes in subspace S_i are compromised, they can be used to construct $\frac{N_i(N_i+1)}{2}$ equations, i.e.,

$$\begin{cases} f_2(u_1, u_1) = K_{11} \\ \vdots \\ f_2(u_1, u_{N_i}) = K_{1N_i} \\ f_2(u_2, u_2) = K_{22} \\ \vdots \\ f_2(u_{N_i}, u_{N_i}) = K_{N_iN_i} \end{cases} , \tag{15}$$

where u_j for $j = 1, 2, \ldots, N_i$ are the ID indices in subspace $\mathcal{S}_i$. $K_{j_1,j_2}, j_1 \neq j_2$ is the direct key between the j_1-th and the j_2-th nodes in the subspace, and $K_{j,j}$ is calculated by inputting the i-th ID index of the j-th node into its own polynomial share.

If all the subspaces are compromised, the total number of equations that adversaries can construct is

$$\begin{aligned} N_e &= \frac{N}{N_1} \cdot \frac{N_1(N_1+1)}{2} + \frac{N}{N_2} \cdot \frac{N_2(N_2+1)}{2} + \\ &\cdots + \frac{N}{N_k} \cdot \frac{N_k(N_k+1)}{2} \\ &= \frac{1}{2}(\prod_{i=1}^{k} N_i)(\sum_{i=1}^{k} N_i + k) \ , \end{aligned} \tag{16}$$

where the total number of nodes in the network is $N = N_1 \cdot N_2 \cdots N_k$.

The number of coefficients of a t-degree $(k+1)$-variate symmetric polynomial is [3]

$$N_c = \binom{t+k+1}{k+1} \ . \tag{17}$$

TABLE I
BOUND AND PRECISE RATIOS BETWEEN t^* AND N_1

k	r	t^*/N_1
1	1	1
2	1.8171	1.7715
3	2.4495	2.3919
4	2.9926	2.9219
5	3.4878	3.4058

Therefore, to guarantee perfect security of the global polynomial, the following condition should be satisfied, i.e.,

$$N_e \leq N_c \Longrightarrow \frac{1}{2}(\prod_{i=1}^{k} N_i)(\sum_{i=1}^{k} N_i + k) \leq \binom{t+k+1}{k+1} . \quad (18)$$

3) Choose Degree t: Given the number of nodes in the network, any polynomial degree t satisfying the aforementioned conditions (14) and (18) can be chosen. Each node needs to keep a t-degree univariate polynomial, which has $t+1$ coefficients. Thus, to minimize memory cost per node, we should use the polynomial which has minimum degree satisfying the aforementioned conditions.

Here we consider a common case where $N_i = N_1$ for $i = 1, 2, \ldots, k$, i.e., all subspaces have the same number of indices. Thus, the inequality in (18) can be changed to

$$\begin{aligned} \frac{k}{2}{N_1}^k(N_1+1) &\leq \left(\frac{t}{k+1}+1\right)\left(\frac{t}{k}+1\right)\left(\frac{t}{k-1}+1\right) \\ &\cdots \left(\frac{t}{2}+1\right)(t+1) . \end{aligned} \quad (19)$$

We can prove that when

$$t \geq N_1 \sqrt[k+1]{k(k+1)!/2} \quad (20)$$

the inequality (19) can be satisfied.

Proof :

$$\begin{aligned} & \left(\frac{t}{k+1}+1\right)\left(\frac{t}{k}+1\right)\cdots\left(\frac{t}{2}+1\right)(t+1) \\ > & \frac{t^{k+1}}{(k+1)!} + \frac{(k+1)(k+2)}{2(k+1)!}t^k \\ \geq & \frac{\left(N_1 \sqrt[k+1]{k(k+1)!/2}\right)^{k+1}}{(k+1)!} + \\ & \left(N_1 \sqrt[k+1]{k(k+1)!/2}\right)^k \frac{(k+1)(k+2)}{2(k+1)!} \\ = & \frac{k}{2}{N_1}^{k+1} + \left(\frac{k}{2}\right)^{\frac{k}{k+1}} \frac{(k+1)(k+2)}{2\sqrt[k+1]{(k+1)!}}{N_1}^k \\ > & \frac{k}{2}{N_1}^{k+1} + \frac{(k+1)(k+2)}{2\sqrt[k+1]{(k+1)^{k+1}}}{N_1}^k \\ = & \frac{k}{2}{N_1}^{k+1} + \frac{k+2}{2}{N_1}^k \\ > & \frac{k}{2}{N_1}^k(N_1+1) , \end{aligned} \quad (21)$$

where $k \geq 2$. ■

Because $\binom{t+k+1}{k+1}$ is a monotonic increasing function of t, the solution of (19) should be $[t^*, \infty)$, where t^* is the minimum degree satisfying (19). Because the solution of (20) is the subset of the solution of (19), the minimum global polynomial degree t^* can be bounded as

$$t^* \leq r \cdot N_1 , \quad (22)$$

where ratio

$$r = \sqrt[k+1]{\frac{k(k+1)!}{2}} . \quad (23)$$

The second column in Table I gives some bound ratios when k is small. Fig. 3 illustrates the precise ratio of t^* to N_1 respect to N_1. We can see when N_1 becomes large, the value of t^* becomes stable and the real ratio is bounded by r. Some average ratios are given in the third column in Table I when k is small. Obviously when the condition in the inequality (18) is satisfied, the condition in the inequality (14) is automatically satisfied.

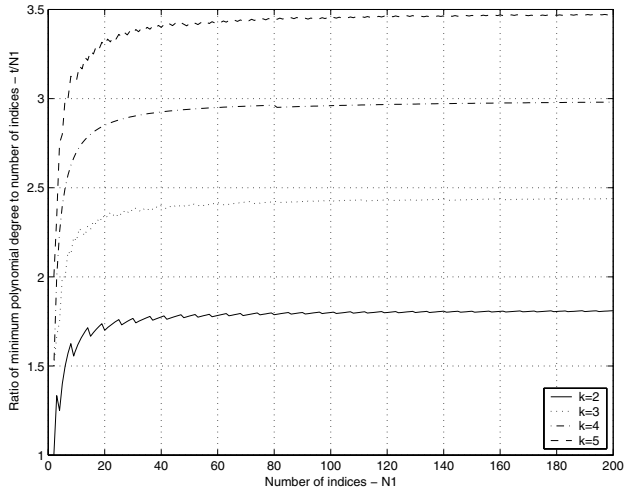


Fig. 3. The ratio of minimum required polynomial degree to number of indices in one subspace.

F. Security of Indirect Keys

For nodes u and v which have j mismatches in their IDs, the secure path between them consists of $j-1$ agent nodes. Suppose the probability that any node is corrupted is p. The probability that the exchanged indirect shared key between u and v is exposed can be calculated as

$$P_c = 1 - (1-p)^{j-1} . \quad (24)$$

Because the maximum number of mismatches in k-dimension ID space is k, the maximum probability that the exchanged key is exposed is

$$P_{c,max} = 1 - (1-p)^{k-1} . \quad (25)$$

Obviously, by tuning k, our scheme can achieve a trade-off between security and memory cost in large scale networks.

G. Memory Cost

The memory cost per node is mainly related to two parts, i.e., one for node ID and the other for polynomial share. Remind that each node n is identified by a k-tuple $(n_1, n_2, \ldots, n_k)$. All indices can be obtained by dividing its

node ID into k fields. In order to do this, each node needs to know how many bits are allocated for each field. Hence each node should keep the values of N_i for $i = 1, \ldots, k$. The total number of bits should be used is

$$M_{ID} = \sum_{i=1}^{k} \log N_i = \log N \ . \tag{26}$$

When all subspaces are equal sized, the memory cost for node ID is

$$M_{ID} = k \log N_1 \ . \tag{27}$$

In addition, each node in the network keeps a t-degree univariate polynomial share, which has $t+1$ coefficients drawn from the finite field $\mathbb{F}_q$. With the bound calculated in the previous section, we know the memory cost per node for polynomial share can be bounded as

$$M_p \leq \left(\sqrt[k]{N}^{\,k+1}\sqrt{\frac{k(k+1)!}{2}} + 1 \right) \log q \ . \tag{28}$$

Due to the large value of q, usually we have $M_{ID} \ll M_p$. Thus, the total memory cost is

$$\begin{aligned} M &= M_{ID} + M_p \\ &\leq \log N + \left(\sqrt[k]{N}^{\,k+1}\sqrt{\frac{k(k+1)!}{2}} + 1 \right) \log q \\ &\sim \sqrt[k]{N}\, r \log q \ . \end{aligned} \tag{29}$$

Obviously, compared with conventional probabilistic distributed models, which have memory cost at the level of $\mathcal{O}(N)$, our scheme has very small memory cost per node, which is on the order $\mathcal{O}(\sqrt[k]{N})$ when k is fixed.

H. Computation Overhead

Our scheme is based on the symmetric key technology. Each sensor node can calculate a key by using a t-degree univariate polynomial, which is a share of a global polynomial. To calculate a key, each node should calculate $2t-1$ modular multiplications over $\mathbb{F}_q^*$: $t-1$ for $x^2, \ldots, x^t$ and t for $b_1 x, b_2 x^2, \ldots, b_t x^t$. Under the symmetric key technology, the length of q is usually 64 bits or 128 bits. Suppose the total number of nodes is N and each subspace has the same number of nodes. We can estimate that the number of 64-bit or 128-bit modular multiplications each node needs to calculate is

$$C_1 = 2t^* - 1 \leq 2r \sqrt[k]{N} + 1 = 2\sqrt[k+1]{\frac{k(k+1)!}{2}}\sqrt[k]{N} + 1 \ . \tag{30}$$

Compared with conventional probabilistic schemes [9]–[18], which need computation on several polynomials, our scheme is more efficient.

I. Communication Overhead

As the establishment of direct keys between a pair of nodes does not require handshakes between them, the major communication overhead lies with the establishment of indirect keys. Just like most existing security schemes that require handshakes between end nodes to negotiate a shared key, this overhead is inevitable. However, few analytical results about the overhead are given by current schemes. Most of them rely on simulation to measure communication overhead. Here, we give an analytical estimation of the communication overhead of our scheme.

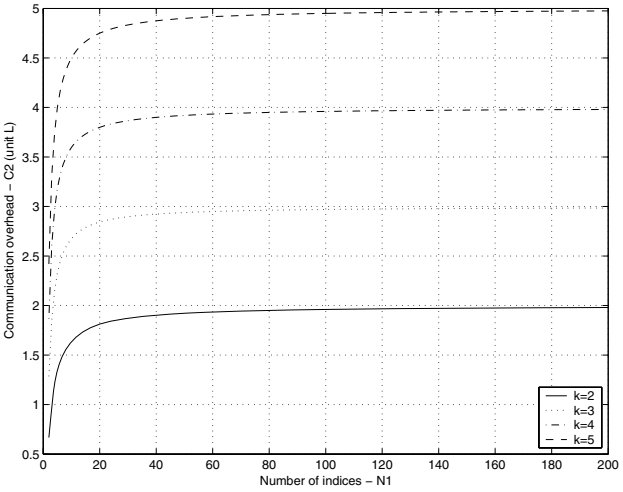


Fig. 4. The communication overhead to number of indices in one subspace.

For a pair of nodes with i, for $i = 2, \ldots, k$, mismatches in their IDs, a secure path between them involves $i-1$ agent nodes. If the average path length between a pair of nodes that have only one mismatch in their IDs is L, the average path length between a pair of nodes with i mismatches in their IDs is iL. The probability that two nodes have i mismatches in their IDs is $\binom{k}{i}(\sqrt[k]{N}-1)^i/(N-1)$. Hence the average communication overhead can be estimated as

$$\begin{aligned} C_2 &= \sum_{i=2}^{k} \frac{\binom{k}{i}(\sqrt[k]{N}-1)^i}{N-1} iL \\ &= \frac{L(\sqrt[k]{N}-1)}{N-1} \left(\sum_{i=2}^{k} \binom{k}{i} x^i \right)'_x , \qquad x = \sqrt[k]{N} - 1 \\ &= \frac{k(\sqrt[k]{N^{k-1}}-1)(\sqrt[k]{N}-1)}{N-1} L \\ &= \frac{k(N_1^{k-1}-1)(N_1-1)}{N_1^k - 1} L \ , \end{aligned} \tag{31}$$

where $N = N_1^k$. Several cases when k is small are depicted in Fig. 4.

V. Comparisons With Related Work

Centralized schemes, such as *SPINS* [1], need a trusted server to facilitate key agreement between any two nodes. The trusted server can be a potential failure point. Distributed methods are more secure. A simple method is the full *pairwise key* distribution, in which each pair of nodes in a network of N nodes is preloaded with a distinct symmetric key. Each node, however, must keep $N-1$ symmetric keys. Another two basic distributed methods are proposed by Blom [2] and Blundo *et al.* [3], which feature the same amount of memory cost as the full pairwise key approach. Those distributed methods lack scalability and thus only suitable in small networks.

Probabilistic schemes [4]–[22] can provide a certain level of scalability with the tradeoff that they can not guarantee

that every pair of nodes establish a shared key. The memory cost of those schemes increases linearly with respect to the total number of nodes if they need to achieve a certain level of security or communication efficiency [23]. Moreover, those schemes are targeted at the key establishment between neighboring nodes, while our scheme can achieve the end-to-end key agreement.

Combinatorial design techniques are proposed in [24], [25]. They can ensure key sharing between any pair of nodes. In their schemes, however, each key is reused by many sensor nodes. This leads to poor resilience to node compromise in that one compromised node can expose keys belongs to other noncompromised nodes. In addition, the memory cost of their schemes is roughly $\mathcal{O}(\sqrt{N})$ where N is the total number of nodes, while the memory cost of our scheme can be $\mathcal{O}(\sqrt[k]{N})$, which is more scalable.

PIKE [23] organizes the network into a 2-dimension grid, and assigns each pair of nodes along each dimension a unique key. It is similar to our scheme in terms of security and communication overhead if we extend PIKE into a k-dimension space. Another similar work is described in [26], which also uses a k-dimension grid to model a network and k multivariate polynomials to achieve key agreement. Their memory cost, however, is at the level of $k(\sqrt[k]{N}-1)$ where N is the total number of nodes. In comparison, our scheme can achieve more memory efficiency when k increases.

Another merit of our scheme is that the communication overhead tends to be a constant ($\propto L$, where L is the average path length between a pair of nodes that have only one mismatch in their IDs) when the network size is larger than a certain threshold (refer to Fig. 4). This means our scheme can provide a good scalability. On the other hand, the communication overhead can be reduced if we could reduce the value of L. We have shown in [27], [28] that in static networks (such as sensor networks) deployment information can be used to reduce the value of L and thus reduce the communication overhead.

We [29] also developed node authentication and key establishment schemes based on the elliptic curve cryptography, which can provide higher security level than symmetric ones, but have more overhead.

VI. Security Enhancement of Indirect Keys

In our scheme, direct keys are safe because they are calculated by end nodes without any interaction. On the other hand, indirect keys may be exposed during their transmission between end nodes if any intermediate agent node is compromised. However, the existence of multiple secure paths between two nodes can be utilized to enhance the confidentiality of indirect keys. The idea is to transform an indirect key into many pieces and transmit those pieces through multiple secure paths in stead of one such that the key can be recovered if and only if all those secure paths are corrupted [30].

Suppose node u needs to negotiate an indirect key with v. Node u may randomly select a key K_{uv} and construct a new polynomial as

$$g(x) = K_{uv} + k_1 x + k_2 x^2 + \cdots + k_s x^s \ . \tag{32}$$

Then, Shamir's $(s+1, T)$ threshold secret sharing scheme [31] can be applied. Specifically, T shares can be calculated as

$$g(1), g(2), \ldots, g(T) \ , \tag{33}$$

where $T \geq s+1$.

Next, node u transmits the T shares to node v through multiple secure paths by following the method proposed in [30]. Suppose u and v have j mismatches in their IDs, which means there are j disjoint secure paths between them. Then node u may transmit T/j shares along each secure path to node v. Once node v gets $s+1$ out of T shares, it can recover the polynomial $g(x)$ and get the key K_{uv} by Lagrange interpolation.

The value T should be chosen properly such that the polynomial $g(x)$ cannot be recovered even if $j-1$ out of j secure paths are corrupted. Thus T should satisfy

$$\begin{cases} T \geq s+1 \\ T - T/j < s+1 \end{cases} \tag{34}$$

$$\Longrightarrow s+1 \leq T < \frac{j(s+1)}{j-1} \ . \tag{35}$$

By following the procedure, the key K_{uv} may be exposed only if all j secure pathes are corrupted. Hence the probability of the key exposal is reduced to

$$P_c' = {P_c}^j = (1-(1-p)^{j-1})^j \ . \tag{36}$$

The tradeoff here is the increase of communication overhead. However, we can choose the number of secure pathes here to achieve a certain level of security while maintaining an acceptable communication overhead.

VII. Conclusion

The dimension of the ID space k is a parameter we can control to achieve the trade-off between overhead per node and security performance. Conventional schemes can hardly be scalable due to their large memory cost. Our scheme is scalable in that the memory cost per node only increases proportionally to the k-th root of the total number of nodes while the security only decreases gradually. Our scheme is also deterministic compared with most probabilistic schemes.

References

[1] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: Security protocols for sensor networks," *Wireless Networks*, vol. 8, pp. 521-534, Sep. 2002.

[2] R. Blom, "An optimal class of symmetric key generation systems," in *Proc. EUROCRYPT*, 1985, pp. 335-338.

[3] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," in *Proc. Advances Cryptology*, 1992, pp. 471-486.

[4] L. Eschenauer and V. Gligor, "A key management scheme for distributed sensor networks," in *Proc. ACM CCS*, Nov. 2002, pp. 41-47.

[5] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proc. IEEE Symposium Security Privacy*, May 2003, p. 197.

[6] R. D. Pietro, L. V. Mancini, and A. Mei, "Random key-assignment for secure wireless sensor networks," in *Conf. Computer Commun. Security*, 2003, pp. 62-71.

[7] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach," in *Proc. IEEE International Conf. Network Protocols*, Nov. 2003, p. 326.

[8] M. M. Ramkumar and N. Memon, "An efficient random key pre-distribution scheme," in *Proc. IEEE Global Telecommun. Conf.*, Dec. 2004, pp. 2218-2223.

[9] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *Proc. CCS*, Oct. 2003, pp. 42-51.

[10] D. Liu and P. Ning, "Establishing pairwise keys in distributied sensor networks," in *Proc. CCS*, 2003.

[11] J. Hwang and Y. Kim, "Revisiting random key pre-distibution schemes for wireless sensor networks," in *Proc. 2nd ACM Workshop Security Ad Hoc Sensor Networks*, Oct. 2004, pp. 42-52.

[12] R. Wei and J. Wu, "Product construction of key distribution schemes for sensor networks," in *Proc. SAC*, 2004, pp. 280-293.

[13] D. Liu and P. Ning, "Location-based pairwise key establishments for relatively static sensor networks," in *Proc. ACM Workshop Security Ad Hoc Sensor Networks*, Oct. 2003, pp. 72-82.

[14] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 586-597.

[15] D. Huang, M. Mehta, D. Medhi, and L. Harn, "Location-aware key management scheme for wireless sensor networks," in *Proc. 2nd ACM Workshop Security Ad Hoc Sensor Networks*, Oct. 2004, pp. 29-42.

[16] Z. Yu and Y. Guan, "A robust group-based key management scheme for wireless sensor networks," in *Proc. IEEE WCNC*, Mar. 2005, pp. 1915-1920.

[17] Y. Zhou, Y. Zhang, and Y. Fang, "LLK: A link-layer key establishment scheme in wireless sensor networks," in *Proc. IEEE WCNC*, Mar. 2005, pp. 1921-1926.

[18] Y. Zhou, Y. Zhang, and Y. Fang, "Key establishment in sensor networks based on triangle grid deployment model," in *Proc. IEEE Military Commun. Conf.*, Oct. 2005, pp. 347-355.

[19] Y. Zhou and Y. Fang, "A location-based naming mechanism for securing sensor networks," *Wiley's J. Wireless Commun. Mobile Computing*, *Special Issue on Wireless Networks Security*, vol. 6, pp. 347-355, 2006.

[20] D. Liu, P. Ning, and W. Du, "Group-based key pre-distribution in wireless sensor networks," in *Proc. ACM WiSe*, Sep. 2005, pp. 11-20.

[21] F. Liu and X. Cheng, "LKE: A self-configuring scheme for location-aware key establishment in wireless sensor networks," to be published.

[22] F. Liu and X. Cheng, "A self-configured key establishment scheme for large-scale sensor networks," in *Proc. IEEE MASS*, 2006, pp. 447-457.

[23] H. Chan and A. Perrig, "Pike: Peer intermediaries for key establishment in sensor networks," in *Proc. IEEE INFOCOM*, Mar. 2005, pp. 524-535.

[24] J. Lee and D. Stinson, "Deterministic key predistribution schemes for distributed sensor networks," *Selected Areas in Cryptography*, 2004, vol. 3357, pp. 294-307.

[25] S. Camtepe and B. Yener, "Combinatorial design of key distribution mechanisms for wireless sensor networks," in *Proc. 9th European Symp. Research Computer Security*, 2004, vol. 3193, pp. 293-308.

[26] F. Delgosha and F. Fekri, "Key pre-distribution in wireless sensor networks using multivariate polynomials," in *Proc. IEEE Commun. Soc. Conf. Sensor Ad Hoc Commun. Networks*, Sep. 2005, pp. 118-129.

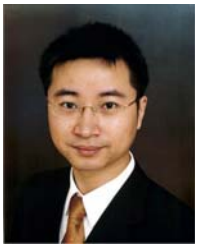
[27] Y. Zhou and Y. Fang, "Scalable link-layer key agreement in sensor networks," in *Proc. IEEE MILCOM*, Oct. 2006, pp. 1-6.

[28] Y. Zhou and Y. Fang, "A two-layer key establishment scheme for wireless sensor networks," *IEEE Trans. Mobile Computing*, vol. 6, no. 9, pp. 1009-1020, Sep. 2007.

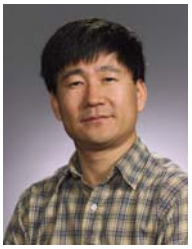
[29] Y. Zhou, Y. Zhang, and Y. Fang, "Access control in wireless sensor networks," *Elsevier Ad Hoc Networks (Special Issue on Sensor and Ad Hoc Networks)*, vol. 5, no. 1, pp. 3-13, Jan. 2007.

[30] W. Lou, W. Liu, and Y. Fang, "SPREAD: Enhancing data confidentiality in mobile ad hoc networks," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 2404-2413.

[31] A. Shamir, "How to share a secret," in *Commun. ACM*, pp. 612-613, Nov. 1979.



Yun Zhou (S'05) received a B.E. degree in electronic information engineering (2000) and an M.E. degree in communication and information systems (2003) from the Department of Electronic Engineering and Information Science at the University of Science and Technology of China, Hefei, China. He is currently pursuing the Ph.D. degree in the Department of Electrical and Computer Engineering at the University of Florida, Gainesville, USA. His research interests are in the areas of security, cryptography, wireless communications and networking, signal processing, and operating systems. He is a student member of the IEEE.



Yuguang Fang (S'92-M'94-S'96-M'97-SM'99) received a Ph.D. degree in Systems Engineering from Case Western Reserve University in January 1994 and a Ph.D degree in Electrical Engineering from Boston University in May 1997. He was an assistant professor in the Department of Electrical and Computer Engineering at the New Jersey Institute of Technology from July 1998 to May 2000. He then joined the Department of Electrical and Computer Engineering at the University of Florida in May 2000 as an assistant professor, got an early promotion to an associate professor with tenure in August 2003, and to a full professor in August 2005. He holds a University of Florida Research Foundation (UFRF) Professorship from 2006 to 2009. He has published over 200 papers in refereed professional journals and conferences. He received the National Science Foundation Faculty Early Career Award in 2001 and the Office of Naval Research Young Investigator Award in 2002. He is the recipient of the Best Paper Award at the IEEE International Conference on Network Protocols (ICNP) in 2006 and the recipient of the IEEE TCGN Best Paper Award at the IEEE High-Speed Networks Symposium, IEEE Globecom in 2002. He has served on several editorial boards of technical journals including the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE TRANSACTIONS ON MOBILE COMPUTING and *ACM Wireless Networks*. He has also been actively participating in professional conference organizations such as serving as The Steering Committee Co-Chair for QShine, the Technical Program Vice-Chair for IEEE INFOCOM'2005, Technical Program Symposium Co-Chair for IEEE Globecom'2004, and a member of the Technical Program Committee for IEEE INFOCOM (1998, 2000, 2003-2008).