A Scalable Key Agreement Scheme 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 is not suitable because of its computation inefficiency and the lack of central authorities in distributed scenarios. Conventional distributed symmetric key agreement models try to achieve key agreement between any pair of nodes without interactions. They are lack of scalability because of their memory cost of $N-1$ in a network of N nodes, and thus only suitable and optimum in small networks. 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. We show that for a network of N nodes our scheme has only $\mathcal{O}(\sqrt[k]{N})$ memory cost per node, where $k \geq 1$. Conventional distributed models can be derived as special cases of our scheme.**

I. Introduction

Key agreement is a central problem to build up secure infrastructures for networks, because all secure primitives including encryption and authentication are based on keys. A general scenario is the two-party key agreement between any two nodes in a network. Though public key schemes outperform symmetric key schemes in terms of flexible manageability, their efficacy relies on the authenticity of public keys. Hence, public key schemes are usually applicable in the networks including fixed authorities who are in charge of the authentication of public keys. However, there are many scenarios, e.g., dynamic conferences or ad hoc networks, where such authorities are not available. In addition, public key schemes require more computation resources than symmetric key schemes. Hence, in large scale networks, symmetric key schemes are pretty suitable because of their efficiency.

To apply symmetric key techniques, one of the fundamental problems is how to achieve key agreement between two nodes in a network. In conventional *key distribution center* (KDC) and *key translation center* (KTC) models [1], all nodes have a shared key with a central trusted server, and this trusted server helps to establish a shared key between any two nodes. In this centralized network, the trusted server is a potential failure point in that the entire network is broken down if the trusted server is corrupted.

In addition to the centralized solutions, some distributed methods exist. An example is the full *pairwise key* model, in which each pair of nodes in a network of N nodes must share a distinct symmetric key. This model is *perfect secure* in that no matter how many nodes collude with each other they know nothing about the symmetric keys held by other normal nodes. However, in this model, each node must have $N-1$ symmetric keys, and the overall number of keys in the network, which may need to be centrally backed up, is then $\frac{N(N-1)}{2}$, or approximately N^2. As the size of the network increases, this number becomes unacceptably large.

In [2], Blom proposes a key distribution method that allows any pair of nodes in a network to be able to find a unique shared key. The optimality of his method is based on $(N, t+1)$ MDS linear codes [3] in that in a network of N nodes the collusion of less than $t+1$ nodes can not reveal any key held by other normal nodes, i.e., *t-secure*. The memory cost per node of the Blom scheme is $t+1$. To guarantee perfect secure in a network with N nodes, the $(N-2)$-secure Blom scheme should be used, which means the memory cost per node is $N-1$.

In [4], Blundo *et al.* suggest to use a t-degree bivariate symmetric polynomial to achieve key agreement. It is a special case of the Blom scheme in that a Vandermonde matrix is used as the generator matrix of MDS code. Like the Blom scheme, the Blundo scheme also provides perfect security while features the same memory cost of $N-1$ in a network of N nodes.

Obviously, previous distributed models are lack of scalability because of their huge memory cost of $N-1$ in a network of N nodes. Hence they are only suitable and optimum in small networks.

In this paper, we propose a novel key agreement scheme, which is scalable for large networks with small memory cost per node. We show that for a network of N nodes our scheme has only $\mathcal{O}(\sqrt[k]{N})$ memory cost per node, where $k \geq 1$. Conventional distributed models can be generated as special cases of our scheme when $k = 1$. Unlike centralized schemes where a fixed server is used, in our scheme every node may be a third trusted party to help key agreement between other two nodes, which means more robust against node corruption.

The rest of the paper is organized as follows. Section II gives our motivation to address the scalability problem of key agreement in large scale networks. Section III describes the mathematical tool we use in our scheme. Details of our scheme are given in Section IV. Analysis on network parameters, such as memory cost and security is given in Section V. A method to enhance security is described in Section VI. The paper is ended with a discussion in Section VII.

Yun Zhou is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA (Tel: (352)392-8576; Fax: (352)392-0044; Email: yzufl@ufl.edu).

Yuguang Fang is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA (Tel: (352)846-3043; Fax: (352)392-0044; Email: fang@ece.ufl.edu).

This work was supported in part by the US Office of Naval Research under grant N000140210464 (Young Investigator Award) and under grant N000140210554, and by the US National Science Foundation under grant ANI-0093241 (CAREER Award) and under grant ANI-0220287.

1-4244-0065-1/06/$20.00 ©2006 IEEE

II. Motivation

In conventional distributed key agreement schemes, all nodes are non-interactive. Every node can independently learn the keys shared with other nodes without help of other trusted parties. It has been shown in [4] that the memory cost per node in a non-interactive network is at least $N-1$ where N is the total number of nodes no matter what kind of algorithms is used to derive pairwise keys. The pairwise key model, the Blom model and the Blundo model are optimum as non-interactive schemes in terms of their memory cost. However, this memory requirement of non-interactive schemes restricts their applications in many large scale networks, such as ad hoc networks or sensor networks, which may involve hundreds to thousands of individual nodes. Obviously, to further decrease memory cost to increase the scalability in large networks, some interaction between nodes should be involved. The KDC and KTC models are two extreme cases where each node needs to keep only one key shared with the trusted server who helps key agreement between nodes. However, the security of KDC and KTC is worse than non-interactive schemes. Our scheme tries to achieve a trade-off between the interactive approach and the non-interactive approach, thus the memory cost per node can be reduced.

III. 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 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

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

IV. 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 can not 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 can not calculate a direct key.

A. Network Model

We assume each node is identified by an index-tuple $(n_1, n_2, \ldots, n_k)$, 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 $\mathcal{S}_i \subset \mathbb{Z}$ for $i = 1, \ldots, k$. Node IDs satisfy the following properties:

1) n_i is a positive integer drawn from a subspace $\mathcal{S}_i$, for $i = 1, 2, \ldots, k$;
2) Any two subspaces have no intersection, i.e., $\mathcal{S}_i \bigcap \mathcal{S}_j = \phi$, for $i, j = 1, 2, \ldots, k$ and $i \neq j$;
3) The cardinality $|\mathcal{S}_i| = N_i$, for $i = 1, 2, \ldots, k$.

Hence the maximum number of nodes in the network can be $N = \prod_{i=1}^{k} N_i$.

Each node in the network may be corrupted, and may use what he knows to learn more secrets in the network. To further evaluate the impact of node corruption, we assume the probability of the corruption of a node is p.

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. Share Distribution

During the network initialization period, a global t-degree $(k+1)$-variate symmetric polynomial $f(x_1, x_2, \ldots, x_k, x_{k+1})$ is constructed as shown in Section III. For a node $(n_1, n_2, \ldots, n_k)$ in the network, a *polynomial share*

$$\begin{aligned} f_1(x_{k+1}) &= f(n_1, n_2, \ldots, n_k, x_{k+1}) \\ &= \sum_{i_{k+1}=0}^{t} b_{i_{k+1}} \, x_{k+1}^{i_{k+1}} \, , \end{aligned} \quad (5)$$

where

$$b_{i_{k+1}} = \sum_{i_1=0}^{t} \sum_{i_2=0}^{t} \cdots \sum_{i_k=0}^{t} a_{i_1,i_2,\ldots,i_k,i_{k+1}} \, n_1^{i_1} n_2^{i_2} \cdots n_k^{i_k} \, , \quad (6)$$

is calculated by using the node ID as inputs to the t-degree $(k+1)$-variate symmetric polynomial. 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.

C. Direct Key Calculation

Two nodes u with ID $(u_1, u_2, \ldots, u_k)$ and v with ID $(v_1, v_2, \ldots, v_k)$ may calculate a shared key directly if the following conditions are satisfied:

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 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 as well 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

$$K_{uv} = f(c_1, \ldots, u_i, \ldots, c_k, v_i) = f(c_1, \ldots, v_i, \ldots, c_k, u_i) \,. \tag{7}$$

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

$$f_2(x_i, x_{k+1}) = f(c_1, \ldots, c_{i-1}, x_i, c_{i+1}, \ldots, c_k, x_{k+1}) \,, \tag{8}$$

where c_j for $j = 1, 2, \ldots, k, j \neq i$ are the common indices between nodes u and v.

Because all node indices of u and v are drawn from different subspaces where any two subspaces have no intersection and $u_i \neq v_i$, the $k+1$ indices $c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_k, u_i, v_i$ are pairwise different. Hence 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 may calculate a unique shared key directly if there is only one mismatch between their k-tuple IDs. This is similar to the Blom scheme and the Blundo scheme, because the two nodes have the same t-degree bivariate symmetric polynomial to calculate a pairwise key.

D. Indirect Key Negotiation

If two nodes have more than one mismatch between their IDs, they can not calculate a shared key directly, because they do not have the same bivariate symmetric polynomial. However, they may rely on some intermediate nodes as *agents* to negotiate a shared key.

1) j mismatches ($j \geq 2$): Suppose two nodes u and v have j 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 may 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 noticing that there are more 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 makes our scheme more robust than the conventional centralized models against node corruption.

2) 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 index from $(v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_{j-1}}, v_{i_j})$. Hence 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

$$N_p = j \cdot (j-1) \cdot \cdots \cdot 2 \cdot 1 = j! \,. \tag{9}$$

3) Number of disjoint secure paths: Out of the N_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 who have j mismatches in their IDs, the number of direct agent nodes of the end nodes u or v is j. Hence the number of disjoint secure paths is

$$N_d = j \,. \tag{10}$$

4) 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 and v, draw indices from u's mismatching ID at those positions, and draw indices from v's mismatching 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 and v. Hence the number of agent nodes can be calculated as

$$N_a = \binom{j}{1} + \binom{j}{2} + \cdots + \binom{j}{j-1} = 2^j - 2 \,. \tag{11}$$

5) An example: An example of 3-dimension ID space is given in Fig. 1. 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

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

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

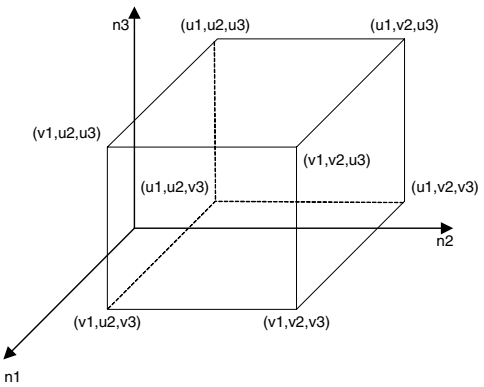

Fig. 1. An example of key graph, where nodes (u_1, u_2, u_3) and (v_1, v_2, v_3) and all 6 agent nodes form a cube in the 3-dimension ID space.

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.

V. Analysis

A. Polynomial Degree

All nodes in the network hold partial information of one t-degree $(k+1)$-variate polynomial to achieve key agreement. Some nodes may be corrupted and cooperate to expose the polynomial with the partial information they hold whereby to calculate keys between other nodes directly. Obviously, the polynomial degree t is an indication of the difficulty to expose the polynomial, and it is directly related to the memory cost per node. In this section, we will investigate how to choose the polynomial degree.

1) Corrupt one subspace: Let us consider the malicious cooperation in one subspace. Suppose there are N_i nodes in the subspace $\mathcal{S}_i$, in which all nodes have same indices in other subspaces, for $i = 1, 2, \ldots, k$. Any pair of nodes in the subspace $\mathcal{S}_i$ can achieve key agreement with a t-degree bivariate polynomial $f_2(x_i, x_{k+1})$, 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 IV). It has been shown in [4] 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 can not reconstruct it. To guarantee any pair of nodes in S_i have a direct key that is unsolvable by other $N_i - 2$ nodes, an $(N_i - 2)$-secure bivariate polynomial should be used. Hence, the degree of polynomial should satisfy

$$0 \leq N_i - 2 \leq t \ , \qquad i = 1, 2, \ldots, k \ . \tag{12}$$

2) Corrupt all subspaces: Even all nodes in one subspace are corrupted, they can not expose the global t-degree $(k+1)$-variate polynomial because they only know a marginal of the global polynomial. Hence, to expose the global polynomial, more subspaces should be corrupted.

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

$$\begin{cases} f_2(u_1, u_1) = K_{11} \\ \vdots \\ f_2(u_1, u_{N_i}) = K_{1N_i} \\ f_2(u_2, u_2) = K_{22} \\ \vdots \\ f_2(u_{N_i}, u_{N_i}) = K_{N_iN_i} \end{cases} , \tag{13}$$

where u_j for $j = 1, 2, \ldots, N_i$ are the 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 with the polynomial share of the j-th node.

If all the subspaces are corrupted, the total number of equations that can be constructed is

$$\begin{aligned} N_e &= \frac{N}{N_1} \cdot \frac{N_1(N_1+1)}{2} + \frac{N}{N_2} \cdot \frac{N_2(N_2+1)}{2} + \\ &\cdots \quad + \frac{N}{N_k} \cdot \frac{N_k(N_k+1)}{2} \\ &= \frac{1}{2}(\prod_{i=1}^{k} N_i)(\sum_{i=1}^{k} N_i + k) \ , \end{aligned} \tag{14}$$

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 [4]

$$N_c = \binom{t+k+1}{k+1} \ . \tag{15}$$

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

$$N_e \leq N_c \Longrightarrow \frac{1}{2}(\prod_{i=1}^{k} N_i)(\sum_{i=1}^{k} N_i + k) \leq \binom{t+k+1}{k+1} \ . \tag{16}$$

3) Choose degree t: Given the number of nodes in the network, any polynomial degree t satisfying the aforementioned conditions (12) and (16) 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.

A common case to design a network is to let $N_i = N_1$ for $i = 1, 2, \ldots, k$, i.e., all subspaces have the same number of indices. Thus, the inequality in (16) can be changed to

$$\begin{aligned} \frac{k}{2} {N_1}^k (N_1 + 1) &\leq \left(\frac{t}{k+1} + 1\right)\left(\frac{t}{k} + 1\right)\left(\frac{t}{k-1} + 1\right) \\ &\cdots \quad \left(\frac{t}{2} + 1\right)(t+1) \ . \end{aligned} \tag{17}$$

We can prove that when

$$t \geq N_1 \sqrt[k+1]{k(k+1)!/2} \tag{18}$$

the inequality (17) can be satisfied.

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

Proof :

$$\begin{aligned}
&\left(\frac{t}{k+1}+1\right)\left(\frac{t}{k}+1\right)\cdots\left(\frac{t}{2}+1\right)(t+1)\\
>\ &\frac{t^{k+1}}{(k+1)!}+\frac{(k+1)(k+2)}{2(k+1)!}t^k\\
\geq\ &\frac{\left(N_1\sqrt[k+1]{k(k+1)!/2}\right)^{k+1}}{(k+1)!}+\\
&\left(N_1\sqrt[k+1]{k(k+1)!/2}\right)^{k}\frac{(k+1)(k+2)}{2(k+1)!}\\
=\ &\frac{k}{2}{N_1}^{k+1}+\left(\frac{k}{2}\right)^{\frac{k}{k+1}}\frac{(k+1)(k+2)}{2\sqrt[k+1]{(k+1)!}}{N_1}^{k}\\
>\ &\frac{k}{2}{N_1}^{k+1}+\frac{(k+1)(k+2)}{2\sqrt[k+1]{(k+1)^{k+1}}}{N_1}^{k}\\
=\ &\frac{k}{2}{N_1}^{k+1}+\frac{k+2}{2}{N_1}^{k}\\
>\ &\frac{k}{2}{N_1}^{k}(N_1+1)\ ,
\end{aligned}\tag{19}$$

where $k \geq 2$. ■

Because $\binom{t+k+1}{k+1}$ is a monotonic increasing function of t, the solution of (17) should be $[t^*, \infty)$, where t^* is the minimum degree satisfying (17). Because the solution of (18) is the subset of the solution of (17), the minimum global polynomial degree t^* can be bounded as

$$t^* \leq r \cdot N_1\ , \tag{20}$$

where ratio

$$r = \sqrt[k+1]{\frac{k(k+1)!}{2}}\ . \tag{21}$$

The second column in Table I gives some bound ratios when k is small. Figure 2 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 (16) is satisfied, the condition in the inequality (12) is automatically satisfied.

B. 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 is identified by a k-tuple $(n_1, n_2, \ldots, n_k)$ where $n_i \in \mathcal{S}_i$ and all subspaces $\mathcal{S}_i$ are disjoint. The total number of different indices is $N_1 + N_2 + \cdots + N_k$ where $N_i = |\mathcal{S}_i|$. Hence for each k-tuple ID, the number of bits should be used is

$$M_{ID} = k\log(N_1 + N_2 + \cdots + N_k)\ . \tag{22}$$

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

$$M_{ID} = k\log kN_1 = k\log k + \log N\ . \tag{23}$$

Besides, 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

$$M_p \leq \left(\sqrt[k]{N}\sqrt[k+1]{\frac{k(k+1)!}{2}}+1\right)\log q\ . \tag{24}$$

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

$$\begin{aligned}
M\ &=\ M_{ID} + M_p\\
&\leq\ k\log k + \log N + \left(\sqrt[k]{N}\sqrt[k+1]{\frac{k(k+1)!}{2}}+1\right)\log q\\
&\sim\ \sqrt[k]{N}\, r\log q\ .
\end{aligned}\tag{25}$$

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

C. Security

For nodes u and v who 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 shared key between u and v is exposed can be calculated as

$$P_c = 1 - (1-p)^{j-1}\ . \tag{26}$$

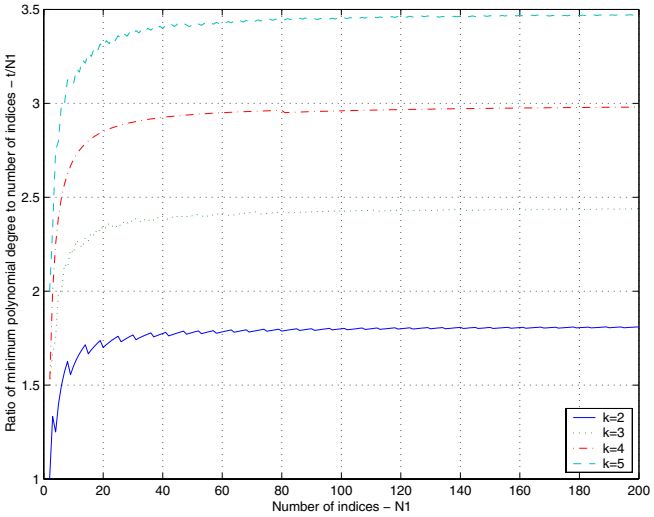

Fig. 2. The ratio of minimum required polynomial degree to number of indices in one subspace.

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

$$P_{c,max} = 1 - (1-p)^{k-1} \ . \tag{27}$$

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

VI. Enhancement

The existence of multiple secure paths between two nodes can be utilized to enhance the confidentiality of the exchanged key. The idea is to transform the key into many pieces and transmit those pieces through multiple secure paths such that the key can be recovered if and only if all those secure paths are corrupted.

When node u needs to negotiate a key with v, u constructs a new subspace $\mathcal{S}_{k+1} \subset \mathbb{Z}$ where $\mathcal{S}_{k+1} \bigcap \mathcal{S}_i = \phi$ for $i = 1, 2, \ldots, k$. Remember u has a polynomial share

$$f_1(x_{k+1}) = \sum_{i_{k+1}=0}^{t} b_{i_{k+1}} \ x_{k+1}^{i_{k+1}} \ . \tag{28}$$

Node u may randomly select $s \in \mathcal{S}_{k+1}$ and calculate a key $K_{uv} = sb_0$, thus a new polynomial can be formed as

$$f_1'(x_{k+1}) = K_{uv} + b_1 x_{k+1} + b_2 x_{k+1}^2 + \cdots + + b_t x_{k+1}^t \ . \tag{29}$$

Then, Shamir's $(t+1, T)$ threshold secret sharing scheme [5] can be applied. Specifically, T shares can be calculated as

$$f_1'(1), f_1'(2), \ldots, f_1'(T) \ , \tag{30}$$

where $T \geq t+1$.

Next, node u transmits the T shares to node v through multiple secure paths by following the method proposed in [6]. 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 $t+1$ out of T shares, he can recover the polynomial $f_1'(x_{k+1})$ and get the key K_{uv} by Lagrange interpolation.

The value T should be chosen properly such that the polynomial $f_1'(x_{k+1})$ can not be recovered even if $j-1$ out of j secure paths are corrupted. Thus T should satisfy

$$\begin{cases} T \geq t+1 \\ T - T/j < t+1 \end{cases} \tag{31}$$

$$\Longrightarrow t+1 \leq T < \frac{j(t+1)}{j-1} \ . \tag{32}$$

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

$$P_c' = {P_c}^j = (1 - (1-p)^{j-1})^j \ . \tag{33}$$

VII. Discussion and Conclusion

The dimension of the ID space k is a parameter we can control to achieve the trade-off between memory cost per node and security performance. The conventional pairwise key model, the Blom model and the Blundo model are special cases where $k = 1$. They are perfect secure, but have memory cost of $N-1$, which is not suitable when number of nodes increases. Our scheme are 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 deceases gradually.

Conventional key agreement models have been finding many applications in large scale networks, especially ad hoc networks and sensor networks [7]–[9]. Due to the large number of nodes in those networks, it is not realistic to apply those models directly. A general way is to partition the entire network into many portions and apply conventional models in each portion. But the performance is not good in terms of large memory cost and low security. However our scheme is pretty suitable in those scenarios because of its good scalability. We will continue to investigate our scheme in those scenarios in our future work.

References

[1] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[2] R. Blom, "An optimal class of symmetric key generation systems," in *Proc. of EUROCRYPT '84*, pages 335-338, 1985.

[3] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correction Codes*, North-Holland, New York, 1977.

[4] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro,and M. Yung, "Perfectly-secure key distribution for dynamic conferences," in *Advances in Cryptology C CRYPTO 92, LNCS 740*, pages 471-486, 1992.

[5] A. Shamir, "How to share a secret," in *Communications of the ACM*, 22911):612-613, November, 1979.

[6] Wenjing Lou, Wei Liu and Yuguang Fang, "SPREAD: Enhancing data confidentiality in mobile ad hoc networks," in *IEEE INFOCOM'04*, HongKong, China, Mar 2004.

[7] Haowen Chan, Adrian Perrig, and Dawn Song, "Random key predistribution schemes for sensor networks," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, p.197, May 11-14, 2003.

[8] W. Du, J. Deng, Y. S. Han, and P. K.Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *CCS'03*, Washington, DC, October 27-30, 2003.

[9] D. Liu, and P. Ning, "Establishing pairwise keys in distributied sensor networks," *CCS'03*, Washington, DC, 2003.