SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2015; **8**:1794–1800
Published online 19 December 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1143

RESEARCH ARTICLE

Conference key establishment protocol using a multivariate polynomial and its applications

Lein Harn[1]* and Guang Gong[2]

[1] Department of Computer Science Electrical Engineering, University of Missouri at Kansas City, Kansas City, Missouri, U.S.A.
[2] Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada

ABSTRACT

In 1992, a non-interactive k-secure m-conference protocol based on an m-variate polynomial has been proposed. Each user needs to store a $(m-1)$-polynomial having degree k as a private share. A secret conference key involving m users can be computed by each conference member non-interactively using each private share. There is no overhead to exchange information in order to establish a conference key. However, the storage space of each user is exponentially proportional to the group size of the conference. In this paper, we propose a key establishment protocol using a multivariate polynomial in Z_N, where N is a RSA modulus. One unique feature of using this special type of polynomials for conference key protocol is that the storage space of each user is fixed and is independent to the group size of the conference. User can use their shares obtained from a key generation center initially to establish conference keys consisting of different users. Furthermore, we propose two applications to demonstrate the importance of using this special type of polynomials to design solutions. One is the private reconstruction of secret in a secret sharing scheme over network, and the other is the secure group communication. Copyright © 2014 John Wiley & Sons, Ltd.

KEYWORDS

conference key; multivariate polynomial; secret sharing; RSA assumption; group communication

***Correspondence**

Lein Harn, Department of Computer Science Electrical Engineering, University of Missouri at Kansas City, Kansas City, Missouri, U.S.A.
E-mail: lharn@umkc.edu

1. INTRODUCTION

In a secure communication involving n members ($n \geq 2$), a group key is needed to be shared among group users and uses it to encrypt and authenticate messages. A group key establishment protocol is a method to enable multiple users to share a secret group key.

The class of centralized group key management protocols is the most widely used protocols because of its efficiency in implementation. For example, the Institute of Electrical and Electronics Engineers (IEEE) 802.11i standard [1] employs an online server to select a group key and transport it to each group member. The server in the IEEE 802.11i encrypts the group temporal key using the key encryption key obtained from the authentication server, and the server transmits the encrypted message to each mobile client (group member) separately. In IEEE 802.11i, the confidentiality of group key is protected by conventional encryption algorithm, which is conditionally secure. Recently, Harn and Lin [2] have proposed an authenticated group key transfer protocol based on secret sharing. Their protocol uses a RSA modulus to resist the insider attack. One major requirement of the centralized group key management protocols is that a pre-shared key is needed between each group member and the key generation center.

The most commonly used key agreement protocol is the Diffie–Hellman (DH) key agreement protocol [3]. In DH protocol, the session key is determined by exchanging DH public keys of two communication entities. Because the public key itself does not provide any authentication, a digital signature of the public key can be used to provide authentication. However, DH protocol can provide session key only for two entities, not for a group more than two members. Computing a group DH key among a set of n group members is a special case of secure multiparty computation in which a group of n members each possesses a private input k_i and computes a function $f(k_1, k_2, \ldots, k_n)$ securely [4]. In 2007, Katz *et al.* [5] proposed the first constant round and fully scalable GDH protocol, which is provably secure in the standard model (i.e.,without assuming the existence of "random oracles"). In 2009, Brecher *et al.* [6] extended the tree-DH technique of GDH protocol with robustness, that is,with resistance to faults resulting

from possible system crashes, network failures, and misbehavior of the members. In 2011, Jarecki *et al.* [7] proposed a robust group key agreement protocol, which can tolerate up to t failed nodes. One common feature in these protocols is that secure digital signatures are generated to provide authentication of DH public keys. Because generation and verification of digital signatures take times, the computational cost of each group member is the main concern in implementing these protocols especially when there are a large number of group members. Recently, Harn *et al.* [8] proposed a group key agreement protocol, which used a one-way key confirmation and digital certificates of DH public keys to provide authentication of group keys.

In 1992, Blundo *et al.* [9] have proposed a non-interactive k-secure m-conference protocol based on a multivariate polynomial. Their protocol can establish a conference key of m participants. A system is said to be k-secure if any k users, pooling together their shares, have no information on keys they should not know. The key distribution center (KDC) is responsible to pick a symmetric m-variate polynomial having k degree, $F(x_1, x_2 \ldots, x_m) = \sum_{0 \le j_1, \ldots j_m \le k} a_{j_1, \ldots j_m} (x_1)^{j_1} (x_2)^{j_2} \ldots (x_m)^{j_m}$. and generates shares, $f_i(x_2 \ldots, x_m) = F(i, x_2, \ldots, x_m) = \sum_{0 \le j_2, \ldots j_m \le k} b_{j_2, \ldots j_m} (x_2)^{j_2} \ldots (x_m)^{j_m}$, for $i = 1, 2, \ldots, l$, for users. Then, later, each user can use his share to establish a conference involving m members. In [9], it has shown that the k-secure *2*-conference protocol is a special case of Blom's scheme [10]. Because each share is a polynomial involving $m-1$ variables and having degree k, each user needs to store $(k+1)^{m-1}$ coefficients from $GF(p)$. The storage space of each user is exponentially proportional to the size of conference. This makes their protocol impractical. However, the k-secure *2*-conference protocol is a special case of Blundo *et al.* approach, and this protocol is based on a symmetric bivariate polynomial. The storage space of each user is only to store $k+1$ coefficients from $GF(p)$. This special case can significantly reduce the size of stored information for each user. Since then, key distribution using symmetric bivariate polynomial has been widely adopted in communication applications, such as in the sensor network [11–16]. The general design approach of these schemes is that a server picks a symmetric bivariate polynomial and generates shares for users. Each share is sent to user secretly. Because of the property of symmetry of a bivariate polynomial, secret communication keys can be derived from these shares when secure peer-to-peer communication is needed. There are some key establishment schemes that use polynomials other than a bivariate polynomial. For example, the key establishment scheme proposed by Zhou *et al.* [17] is based on a trivariate polynomial, and the scheme proposed by Fanian *et al.* [18] is based on an m-variate polynomial. However, both schemes can only establish a secret key for two users.

In this paper, we propose a conference key establishment protocol using a multivariate polynomial in Z_N, where N is a RSA modulus. The advantage of using this type of multivariate polynomials in the design of any non-interactive k-secure m-conference protocol can limit the storage space of each user to be the coefficients of a univariate polynomial. In addition, we show that shares generated for a k-secure m-conference protocol can support any conference with size r, where $r \le m$. We propose two applications to demonstrate the importance of using this special type of polynomials to provide solutions. We list the contributions of this paper as follows.

- A non-interactive k-secure m-members n users conference key establishment protocol using a univariate polynomial in Z_N, where N is a RSA modulus, is proposed.
- The memory space of each user is limited to be $(k+1)\log_2 N$ bits, where N is the RSA modulus.
- Shares of users can be used to establish conference keys involving r users, where $r \le m$.
- We prove that the security of the proposed protocol is based on the RSA assumption.
- We show that using our proposed polynomial in two applications, a (t,n) sharing scheme (SS) and a secure group communication, significant improvement of efficiency can be achieved.

The rest of this paper is organized as follows. In Section 2, we first give the definition and describe the model of our proposed key establishment protocol, and then we propose our protocol. In Section 3, we include the performance and security analysis of our protocol. In Section 4, we introduce our design in a (t,n) SS and, in Section 5, we introduce our design in a secure group communication. We conclude in Section 6.

2. CONFERENCE KEY ESTABLISHMENT PROTOCOLS

In this section, we propose a special type of symmetric m-variate polynomial to design the k-secure m-conference protocol. The storage space of each user is the coefficients of a univariate polynomial.

2.1. Definition

In this section, we introduce a k-secure m-members n users conference key establishment protocol that each user can use his share to establish a conference involving at most m users. The following definition defines the properties of the protocol.

Definition 1. *k-secure m-members n users conference key establishment protocol* ((k, m, n) *CKEP*). *Let k, m, n be positive integers with $k, m \le n$. For a group consisting of n users, a k-secure m-members n users conference key establishment protocol has the following properties: (1) the protocol can resist up to k colluded users, and (2) the*

protocol can establish a secure conference key for any conference having r (i.e.,$r \leq m$) members.

In our proposed (*k,m,n*) *CKEP*, the KDC is responsible to register *n* users, $\{P_1, P_2, \ldots, P_n\}$, and issue a secret share, s_i, to each user, P_i, initially. In the event that any *r* users, for example $\{P_{i_1}, P_{i_2}, \ldots, P_{i_r}\}$, want to hold a secure conference, each user, P_{i_j}, can user his or her secret share, s_{i_j}, to compute a secret conference key non-interactively as

$$\begin{cases} F(s_{i_j}) = K, \text{if } P_{i_j} \in \{P_{i_1}, P_{i_2} \ldots, P_{i_r}\}; \\ F(s_{i_j}) \neq K, \text{if } P_{i_j} \notin \{P_{i_1}, P_{i_2} \ldots, P_{i_r}\}, \end{cases}$$

where *F* is a public function.

2.2. Protocol

In this section, we propose a (*k*, *m*, *n*) *CKEP* involving *m* users, $\{P_{i_1}, P_{i_2}, \ldots, P_{i_m}\}$, where *m* is an even integer. The KDC picks a RSA modulus *N*, where *N* is the product of two large safe primes, *p* and *q*, that is, $p = 2p' + 1$ and $q = 2q' + 1$, where p' and q' are also primes. The Euler totient function is $\phi(N) = (p-1)(q-1) = 4p'q'$. Thus, we have $GCD(m-1, \phi(N)) = 1$. *p* and *q* are KDC's secrets; *N* is made publicly known. The RSA assumption assumes that it is computationally infeasible to factor the product of two large primes.

For any *m* users want to establish, a secret conference key among them can follow Scheme 1.

Scheme 1. (*k*, *m*, *n*) *CKEP* involving *m* users (*m* is even and $4 \leq m$).

Share generation

Step 1. The KDC selects a random polynomial having degree *k* as $f(x) = a_k x^k + \ldots + a_1 x + a_0 \bmod N$, where $a_i \in (0, N)$. The *m*-variate polynomial is $F(x_1, x_2, \ldots, x_m) = \prod_{i=1}^{m} f(x_i) \bmod N$.

Step 2. The KDC computes share, $s_i(x) = f(i)^{\frac{1}{m-1}} f(x) \bmod N$, for each user P_i, where *i* is a public information associated with the user, P_i, and sends each share, to user P_i secretly, for *i=1,2,…,n*.

Conference key establishment

We assume that *m* users, $\{P_{i_1}, P_{i_2}, \ldots, P_{i_m}\}$, want to establish a secret conference key among them. Each user, P_{i_j}, uses his/her share, $s_{i_j}(x) = f(i_j)^{\frac{1}{m-1}} f(x)$, to compute $K = \prod_{l=1, l \neq j}^{m} s_{i_j}(i_l) = \prod_{l=1}^{m} f(i_l) \bmod N$.

Note that $f(i)^{\frac{1}{m-1}} \bmod N$ exists because *m* is an even integer and $GCD(m-1, 4p'q') = 1$. Each user needs to store a share, which is a univariate polynomial having degree *k*. The storage space is fixed and is independent with the number of users. To compute the conference key using his or her share, each user takes $m-1$ polynomial evaluations.

Remark 1. To establish a conference key involving r users, where $r \leq m$, each user can use his or her share to compute the conference key following Scheme 2.

Scheme 2. (*k*, *m*, *n*) *CKEP* involving *r* users (*m* is even, $r \leq m$, and $4 \leq m$).

We assume that *r* users, $\{P_{i_1}, P_{i_2}, \ldots, P_{i_r}\}$, want to establish a secret conference key among them. Each user, P_{i_j}, uses his/her share $s_{i_j}(x)$ to compute the conference key as $K = \prod_{l=1, l \neq j}^{r} s_{i_j}(i_l) \cdot s_{i_j}^{m-r}(0) \bmod N$.

3. PERFORMANCE AND SECURITY ANALYSIS

3.1. Performance

Each user needs to store the coefficients of a polynomial having degree *k*. There is no communication overhead to exchange information in order to establish a conference key. Note that each user, P_i, has his or her secret share, $s_i(x) = f(i)^{\frac{1}{m-1}} f(x) \bmod N$, where $f(x)$ is a polynomial having degree *k* and each coefficient, $a_i \in Z_N$. Thus, the memory cost of each user is to store $(k+1)\log_2 N$ bits.

Horner's rule [19] can be used to evaluate polynomials. In the following discussion, we show the cost for computing a conference key involving *r* users ($r \leq m$). From Horner's rule, evaluating a polynomial of degree *k* needs *k* multiplications and $k+1$ additions. The computational cost to establish a conference key with size *r* consists of the cost of evaluating $r-1$ polynomials and evaluating an integer with power $m-r$ (i.e., needs $\log_2(m-r)$ multiplications). Overall, the computational cost, to establish a conference key involving *r* users, each user needs to evaluate $(r-1)k + \log_2(m-r) + r$ multiplications and $(r-1)(k+1)$ additions. This computation is much simpler than the RSA public-key operation, which requires approximately $1.5 \log_2 N$ multiplications.

If new users join the application, KDC can just issue new shares to new users without affecting shares of existing users. On the other hand, if there are departing users from the application, all existing shares do not need to be updated. However, if we assume that any departing user may compromise his or her share, the secrecy of shares of remaining shareholders is protected if the number of compromising shares is less than $k+1$.

3.2. Analysis

In this section, we provide analysis of our proposed protocol. From Definition 1, the (*k*, *m*, *n*) *CKEP* protocol has the following properties: (a) the protocol can resist up to *k* colluded users and (b) the protocol can establish a secure

conference key for any conference having r (i.e., $r \leq m$) members. We first prove the correctness of the proposed protocol. Then, we prove the security of the protocol is based on the RSA assumption. Our security analysis addresses two types of attacks, the known shares attack and the known conference keys attack. First, we prove that the protocol is secure under the RSA assumption against attack of knowing a single share. Then, we prove that the protocol is secure against attacks of knowing fewer than $k+1$ shares and $k+1$ conference keys.

3.2.1. Correctness of the protocol

The following theorem proves the correctness of our proposed protocol.

Theorem 1. The *protocol can establish a secure conference key for any conference having r (i.e., $r \leq m$) members.*

Proof. In Scheme 2, the conference key for the set of users, $\{P_{i_1}, P_{i_2}, \ldots, P_{i_r}\}$, is

$$\begin{aligned} K &= \left(\prod_{l=1, l\neq j}^{r} s_{i_l}(i_l)\right) \cdot s_{i_j}^{m-r}(0) \bmod N \\ &= \left(\prod_{l=1, l\neq j}^{r} f(i_j)^{\frac{1}{m-1}} f(i_l)\right) \cdot f(i_j)^{\frac{m-r}{m-1}} f^{m-r}(0) \\ &= \prod_{l=1}^{r} f(i_l) \cdot f^{m-r}(0). \end{aligned}$$

This conference key can only be computed by users in the set, $\{P_{i_1}, P_{i_2}, \ldots, P_{i_r}\}$, who know one of the share in the set of shares, $\{s_{i_1}, s_{i_2}, \ldots, s_{i_r}\}$. ■

Remark 2. In our proposed protocol, the conference key is limited to conference members. Any user who has registered and received a share from KDC but not a conference member cannot obtain the conference key.

3.2.2. Known shares attack

(a) Known a single share

Definition 1. *(RSA Assumption) It is computationally infeasible to compute M given only the ciphertext $C = M^e \bmod N$ and RSA public key (N, e).*

Lemma 1. *The proposed scheme is secure under the RSA assumption to resist the attack by a single user to solve the secret polynomial, f(x), used to generate shares.*

Proof. We use the technique of proof by contradiction to prove this Lemma. Suppose to the contrary that without knowing the factoring of the RSA modulus N, given the share, $s_i(x)$, which is a univariate polynomial, $g(x) = f(i)^{\frac{1}{m-1}} f(x)$, having degree k, there exists an algorithm, Algorithm A, which can factor the polynomial, $g(x)$, to obtain the secret polynomial, $f(x)$, of the KDC. In the following discussion, we want to show that the adversary can use Algorithm A to decrypt the RSA ciphertext, $C = M^e \bmod N$, knowing only RSA public key, (N,e). Given any ciphertext, $C = M^e \bmod N$, where $C = (c_k, c_{k-1}, \ldots, c_0)_{10} \in Z_N$, the ciphertext can be represented as a polynomial, $g(x) = c_k x^k + c_{k-1} x^{k-1} + \cdots + c_0 \bmod N$, such that $C = g(10)$. The adversary can model the RSA encryption in terms of polynomials as $g(x) = f(i)^{e-1} f(x) \bmod N$, such that if $x = i = 10$, we have $g(10) = f(10)^e \bmod N \Rightarrow C = M^e \bmod N$. This implies that $M = f(10) mod N$. Thus, Algorithm A can be used to factor $g(x)$ to obtain the polynomial, $f(x)$. The plaintext is computed as $f(10) = M mod N$. This result contradicts the RSA assumption. We conclude that Algorithm A does not exist. ■

Lemma 2. *Knowing a single share cannot solve the secret polynomial, $F(x_1, x_2, \ldots, x_m) = \prod_{i=1}^{m} f(x_i) \bmod N$, used to generate conference key.*

Proof. Our proof is based on the well-known Lagrange interpolation formula. Because the polynomial, $F(x_1, x_2, \ldots, x_m) = \prod_{i=1}^{m} f(x_i) \bmod N$, is constructed by a univariate polynomial, $f(x)$, having degree k, according to the Lagrange interpolation formula, it needs at least $k+1$ or more than $k+1$ points, $(i, f(i)^m), i = 1, 2, \ldots, k+1$, to solve the polynomial by computing $\sum_{i=1}^{k+1} f(i)^m \prod_{l=1}^{m} \prod_{j=1, j\neq i}^{k+1} \frac{(x_l - j)}{(i-j)} \bmod N = \prod_{i=1}^{m} f(x_i) \bmod N$. The user, U_i, knowing only one point, $(i, f(i)^m)$, cannot solve the polynomial, $\prod_{i=1}^{m} f(x_i) \bmod N$. ■

Theorem 2. *The proposed scheme can resist the attacks by a single user knowing only one share under the RSA assumption.*

Proof. From Lemmas 1 and 2, we can conclude that a single user cannot use his share to solve both the secret polynomial, $f(x)$, used to generate shares, and the secret polynomial, $F(x_1, x_2, \ldots, x_m)$, used to generate conference keys under the RSA assumption. Hence, our proposed scheme is secure against attacks by a single user. ■

(a) Known multiple shares

Corollary 1. *Knowing fewer than $k+1$ shares cannot solve the secret polynomial, $\prod_{i=1}^{m} f(x_i) \bmod N$, used to generate conference keys.*

Proof. From Lemma 2, we know that according to the Lagrange interpolation formula, it needs at least $k+1$ or more than $k+1$ points, $(i, f(i)^m), i = 1, 2, \ldots, k+1$, to solve the polynomial used to generate conference keys. Because each share can only generate one point on the polynomial, knowing fewer than $k+1$ shares cannot solve the polynomial used to generate all conference keys. ■

3.2.3. Known conference keys attack

Theorem 3. *Knowing fewer than $k+1$ conference keys cannot solve other conference keys.*

Proof. Knowing $k+1$ or more than $k+1$ conference keys, for example knowing $k+1$ keys in the set, $\{f(1)f(2)\ldots,f(m-1)f(m), f(1)f(2)\ldots,f(m-1)f(m+1),\ldots,f(1)f(2)\ldots,f(m-1)f(m+k)\}$, according to the Lagrange interpolation formula, can compute the interpolation polynomial, $\sum_{i=m}^{k} f(1)f(2),f(m-1)f(i)\prod_{j=m,j\neq i}^{k}\frac{(x-j)}{(i-j)} \bmod N = f(1)\cdot f(2)\cdot\ldots\cdot f(m-1)\,f(x)$. Knowing this polynomial can obtain other conference keys. For example, the conference key, $f(1)f(2)\ldots f(m-1)f(m+2k)$, can be computed from this polynomial. On the other hand, knowing fewer than $k+1$ conference keys cannot obtain the interpolation polynomial and hence cannot obtain other conference keys. Thus, the proposed scheme is secure against attack of knowing fewer than $k+1$ conference keys. ∎

4. APPLICATION TO SECRET SHARING ON NETWORKS

The secret SSs were first introduced by both Blakley [20] and Shamir [21] independently in 1979 as a solution for safeguarding cryptographic keys and have been studied extensively in the literature. SS has become one of the most basic tools in cryptographic research. In Shamir's c SS, a secret s is divided into n shares by a dealer and shares are sent to shareholders secretly. The secret s is shared among n shareholders in such a way that (a) the secret can be reconstructed with t or more than t shares, and (b) the secret cannot be obtained with fewer than t shares. Shamir's (t,n) SS is based on the linear polynomial and is unconditionally secure. There are other types of SS. For example, Blakely's scheme [20] is based on the geometry, and Mignotte's scheme [22] and Asmuth-Bloom's scheme [23] are based on the Chinese Remainder Theorem.

When the secret reconstruction process is performed over a network, if the exchanged shares among shareholders are unprotected, the reconstructed secret can also be available to attackers. In order to protect the secrecy of the secret, encryption is used to protect the revealed shares among shareholders. Therefore, a conference key is also needed in the secret reconstruction.

In Scheme 3, we show that if our proposed multivariate polynomial is used in a $(k+1,n)$ SS to generate shares for shareholders, then in the secret reconstruction process, the shares can not only be used to establish a conference key to protect shares but also can be used to recover the secret. Therefore, the efficiency of this SS is significantly improved because no additional conference key establishment protocol is needed.

Scheme 3. $(k+1,n)$ SS over networks using a multivariate polynomial (n is even).

Shares generation

Step 1. The dealer selects a random polynomial having degree k as $f(x)=a_kx^k+\ldots+a_1x+a_0 \bmod N$, where $a_i\in Z_N$, $f^n(0)\bmod N=a_0^n=s$, and s is the secret. The n-variate polynomial is $F(x_1,x_2,\ldots,x_n)=\prod_{i=1}^{n}f(x_i) \bmod N$.

Step 2. The dealer computes share, $s_i(x)=f(i)^{\frac{1}{n-1}}f(x) \bmod N$, for each user P_{i_r}, where i is a public information associated with the user, P_i, and sends each share, to user P_i secretly, for $i=1,2,\ldots,n$.

Secret reconstruction

We assume that r (i.e., $k+1\le r\le n$) shareholders, $\{P_{i_1},P_{i_2},\ldots,P_{i_r}\}$, want to reconstruct the secret.

Step 1. Each shareholder, P_{i_j}, uses his/her share, $s_{i_j}(x)=f(i_j)^{\frac{1}{n-1}}f(x)$, to compute a conference key, $K=\prod_{l=1,l\neq j}^{r}s_{i_j}(i_l)\cdot s_{i_j}^{n-r}(0)\bmod N$, and $v_{i_j}=s_{i_j}^{n-1}(0)\bmod N$.

Step 2. Each shareholder, P_{i_j}, computes a ciphertext, $c_{i_j}=E_K(v_{i_j})$, and broadcasts c_{i_j} to all other shareholders, where $E_K(v_{i_j})$ denotes the encryption of v_{i_j} using the key K.

Step 3. After receiving all ciphertext, c_{i_j}, each shareholder, P_{i_j}, computes $v_{i_j}=D_K(c_{i_j})$, $j=1,2,\ldots,r$, where $D_K(c_{i_j})$ denotes the decryption of c_{i_j} using the key K.

Step 4. The secret can be obtained by computing $\sum_{j=1}^{r}v_{i_j}\prod_{l=1,l\neq j}^{r}\frac{0-i_l}{i_j-i_l} \bmod N=s$.

In a $(k+1,n)$ *SS*, the secret can be reconstructed when there are $k+1$ or more than $k+1$ shareholders participated in the secret reconstruction. Because $1\le k+1\le n$, the dealer needs to select an n-variate polynomial, $F(x_1,x_2,\ldots,x_n)$, to generate shares, which allows $k+1$ or more than $k+1$ shareholders in the secret reconstruction to establish a conference key following a (k,n,n) *CKEP*. In Step 1, the conference key is $K=\prod_{l=1,l\neq j}^{r}s_{i_j}(i_l)\cdot s_{i_j}^{n-r}(0)\bmod N=\prod_{l=1}^{r}s_{i_j}(i_l)\cdot s_{i_j}^{n-r}(0)\bmod N$, in which only shareholders in the set, $\{P_{i_1},P_{i_2},\ldots,P_{i_r}\}$, are able to compute. Thus, in Step 3, only shareholders in the set can obtain $v_{i_j}=D_K(c_{i_j})$, $j=1,2,\ldots,r$. The following theorem proves that the secret can be obtain in Step 4.

Theorem 4. *The secret can be obtained in Step 4 by all shareholders.*

Proof. In Step 1, each shareholder, P_{i_j}, uses his or her share, $s_{i_j}(x)=f(i_j)^{\frac{1}{n-1}}f(x)$, to compute $v_{i_j}=s_{i_j}^{n-1}(0)\bmod N=f(i_j)f^{n-1}(0)$. In Step 4, we have $\sum_{j=1}^{r}v_{i_j}\prod_{l=1,l\neq j}^{r}\frac{0-i_l}{i_j-i_l} \bmod N=f^{n-1}(0)\sum_{j=1}^{r}f(i_j)\prod_{l=1,l\neq j}^{r}\frac{0-i_l}{i_j-i_l} \bmod N=f^n(0)=a_0^n=s$. ∎

5. APPLICATION TO SECURE GROUP COMMUNICATIONS

User authentication and key establishment are two of the most fundamental security services in computer and

communication application. The user authentication can ensure *verifiers* that *the prover* is the real entity whom he or she claimed to be. The key establishment protocol allows all communication entities to share a session key and use it to protect the exchanged message.

All user authentication protocols [24–30] are one-to-one type of authentications in which the prover interacts with the verifier to verify the identity of the prover. For example, the RSA digital signature [30] can be used to authenticate the signer of the signature. In this approach, the verifier sends a random challenge to the prover. Then, the prover digitally signs the random challenge and returns the digital signature of the challenge to the verifier. After successfully verifying the digital signature, the verifier is convinced that the prover is the one with the identity of the public-key digital certificate used to verify the digital signature. In a group-oriented application that involves n users, each user can employ a conventional authentication protocol for $n-1$ times to authenticate other users. The complexity of this approach is $O(n)$. This complexity may become the bottleneck of a group-oriented application.

Although the trend of communications has been moved into group communication, all solutions of secure group communication are still based on the approaches designed for peer-to-peer communications. In a recent paper [31], a new type of authentication, called the *group authentication*, has been proposed, which is specially designed for the group-oriented communications. The group authentication is very efficient because it authenticates all users belonging to the same group at once. If the verification is passed successfully, all users are members; otherwise, additional user authentication is needed to identify non-members. The group authentication proposed by Harn [31] is based on a (t,n) SS. Each user needs to be registered at a group manager (GM) initially to obtain a secret share to become a group member. The GM publishes the one-way value of the secret. Later, in group authentication, all users need to release their shares to determine whether the one-way value of the recovered secret is the same as the published one-way value. However, the group authentication proposed by Harn [31] did not provide the group key distribution. An additional group key distribution protocol is needed to achieve the complete objective of secure group communications.

In Scheme 4, we show that if our proposed multivariate polynomial is used to generate shares for uses initially, later, in a group communication, shares of users can be used not only to authenticate users belonging to the same group but also to establish a secret group key. Therefore, the efficiency of the secure group communication is significantly improved because both group authentication and group key establishment can be achieved simultaneously using a multivariate polynomial.

Scheme 4. Secure group authentication and key establishment involving n users (n is even).

Share generation

Step 1. The GM selects a random polynomial having degree k as $f(x) = a_k x^k + \ldots + a_1 x + a_0 \bmod N$, where $a_i \in Z_N$, $f^n(0) \bmod N = a_0^n = s$, and s is the secret. The n-variate polynomial is $F(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} f(x_i) \bmod N$. GM makes $h(s)$ publicly known, where h is a one-way function.

Step 2. The GM computes share, $s_i(x) = f(i)^{\frac{1}{n-1}} f(x) \bmod N$, for each user P_i, where i is a public information associated with the user, P_i, and sends each share, to user P_i secretly, for $i=1,2,\ldots,n$.

Group authentication and key establishment

We assume that r (i.e., $k+1 \le r \le n$) users, $\{P_{i_1}, P_{i_2}, \ldots, P_{i_r}\}$, want to establish a secure group communication.

Step 1. Each user, P_{i_j}, uses his/her share, $s_{i_j}(x) = f(i_j)^{\frac{1}{n-1}} f(x)$, to compute a conference key, $K = \prod_{l=1, l\neq j}^{r} s_{i_j}(i_l) \cdot s_{i_j}^{n-r}(0) \bmod N$, and $v_{i_j} = s_{i_j}^{n-1}(0) \bmod N$.

Step 2. Each user, P_{i_j}, computes a ciphertext, $c_{i_j} = E_K(v_{i_j})$, and broadcasts c_{i_j} to all shareholders, where $E_K(v_{i_j})$ denotes the encryption of v_{i_j} using the key K.

Step 3. After receiving all ciphertext, c_{i_j} each user, P_{i_j}, compute $c_{i_j} = D_K(c_{i_j})$, $j = 1, 2, \ldots, r$, where $D_K(c_{i_j})$ denotes the decryption of c_{i_j} using the key K.

Step 4. Each user, P_{i_j}, computes $\sum_{j=1}^{r} v_{i_j} \prod_{l=1, l\neq j}^{r} \frac{0 - i_l}{i_j - i_l} \bmod N = s'$. If $h(s') = h(s)$, all users are group members and K is the secret group key for the group communication; otherwise, there are non-members.

In a group consisting of n users, any number of users (i.e., the number of users, r, in a group communication is $2 \le r \le n$) may form a secure group communication. Therefore, the GM needs to select an n-variate polynomial, $F(x_1, x_2, \ldots, x_n)$, to generate shares, which allows any number of users to establish a group key following a (k, n, n) *CKEP*. In Scheme 4, each user obtains a share from the GM initially. The shares can be used by users to construct a secret group key following a (k, n, n) *CKEP* protocol and recover the secret for group authentication. In Step 1, each user can use his or her share to compute a group key non-interactively and use it to encrypt the exchanged information to other users. In Step 4, if all users in the group communication are group members and compute their values in Steps 1 and 2, honestly, each group member can compute the secret and authenticate the group. At the same time, the group key obtained in Step 1 is used to protect messages of the group communication. However, if there is any non-members, the secret computed by users does not match to the published one-way secret and non-members can be detected.

6. CONCLUSION

We have proposed a (k, m, n) *CKEP* protocol for group communication. We use a special symmetric m-variate polynomial to limit the size of memory space of each user. The protocol can support any conference involving r users, where $r \le m$. We provide performance and security analysis

of the proposed protocol. In addition, we introduce two applications, which use our proposed polynomial to significantly improve the efficiency.

REFERENCES

1. *IEEE 802.11i-2004*: Amendment 6: medium access control (MAC) security enhancements. The Institute of Electrical and Electronics Engineers, Inc., 2004.
2. Harn L, Lin C. Authenticated group key transfer protocol based on secret sharing. *IEEE Transactions on Computers* 2010; **59**(6):842–846.
3. Diffie W, Hellman ME. New directions in cryptography. *IEEE Transactions Information Theory* 1976; **22**(6): 644–654.
4. Ben-Or M, Goldwasser S, Wigderson A. Completeness theorems for noncryptographic fault-tolerant distributed computation, *Proc. of 20th ACM Symposium on the Theory of Computing,* 1988; pp. 1–10.
5. Katz J, Yung M. Scalable protocols for authenticated group key exchange. *Journal of Cryptology* 2007; **20**:85–113.
6. Brecher T, Bresson E, Manulis M. Fully robust tree-Diffie-Hellman group key exchange, *Proc. of the 8th International Conference on Cryptology and Network Security (CANS '09)*, 2009, LNCS, 5888, pp. 478–497.
7. Jarecki S, Kim J, Tsudik G. Flexible robust group key agreement. *IEEE Transactions on Parallel and Distributed Systems* 2011; **22**(5):879–886.
8. Harn L, Lin C. Efficient group Diffie-Hellman key agreement protocols, *Computers and Electrical Engineering*, *available online 22 January 2014*
9. Blundo C, De Santis A, Herzberg A, Kutten S, Vaccaro U, Yung M. Perfectly-secure key distribution for dynamic conferences, *Advances in Cryptology—Crypto '92,* 1993, 740, Springer-Verlag, pp. 471–486.
10. Blom R. An optimal class of symmetric key generation systems, *Advances in Cryptology—Eurocrypt '84,* 209, Springer-Verlag, pp. 335–338.
11. Liu D, Ning P. Establishing pairwise keys in distributed sensor networks, *ACM CCS*, 2003; pp. 52–61.
12. Liu D, Ning, P. Establishing pairwise keys in distributed sensor networks. *ACM Transactions on Information and System Security* 2005; **8**: 41–77.
13. Cheng Y, Agrawal Y. A improved key distribution mechanism for large-scale hierarchical wireless sensor networks. *Journal of Ad Hoc Networks* 2007; **5**(1):35–48.
14. Song G, Zhuhong Q. A compromise-resilient group rekeying scheme for hierarchical wireless sensor networks. *Wireless Communications and Networking Conference (WCNC)*, 2010; pp. 1–6.
15. Liang H, Wang C. An energy efficient dynamic key management scheme based on polynomial and cluster in wireless sensor networks. *Journal of Convergence Information Technology* 2011; **6**(5):321–328.
16. Saxena N, Tsudik G, Yi JH. Efficient node admission and certificateless secure communication in short-lived MANETs. *IEEE Transactions on Parallel and Distributed Systems* 2009; **20**(2):158–170.
17. Zhou Y, Fang Y. A two-layer key establishment scheme for wireless sensor networks. *IEEE Transactions on Mobile Computing* 2007; **6**(9):1009–1020.
18. Fanian A, Berenjkoub M, Saidi H, Gulliver TA. An efficient symmetric polynomial-based key establishment protocol for wireless sensor networks. *The ISC Int'l Journal of Information Security* 2010; **2**(2): 89–105.
19. Knuth DE. *The Art of Computer Programming, Seminumerical Algorithms*, Vol. **II**. Addison Wesley: Reading Massuchussetts, 1981.
20. Blakley GR. Safeguarding cryptographic keys. *Proceedings of AFIPS'79 Nat. Computer Conf.,* 48, AFIPS Press, 1979; pp. 313–317.
21. Shamir A. How to share a secret. *Communications Association for Computing Machinery* 1979; **22**(11): 612–613.
22. Mignotte M. How to share a secret. In *Proc. of the Workshop on Cryptography*. Springer: Heidelberg, 1983; 371–375.
23. Azimuth CA, Bloom J. A modular approach to key safeguarding. *IEEE Transactions on Information Theory* 1983; **29**(2):208–210.
24. Das ML. Two-factor user authentication in wireless sensor networks. *IEEE Transactions on Wireless Communications* 2009; **8**(3):1086–1090.
25. Downnard I. Public-key cryptography extensions into Kerberos. *IEEE Potentials* 2002; **21**(5):30–34.
26. Harn L, Ren J. Generalized digital certificate for user authentication and key establishment for secure communications. *IEEE Transactions on Wireless Communications* 2011; **10**(7):2372–2379.
27. IEEE Computer Society, '802.1X, IEEE standard for local and metropolitan area networks port-based network access control, The Institute of Electrical and Electronics Engineers, Inc., 2004.
28. Ku WC. Weaknesses and drawbacks of a password authentication protocol using neural networks for multiserver architecture. *IEEE Transactions on Neural Networks* 2005; **16**(4):1002–1005.
29. Oppliger R, Hauser R, Basin D. SSL/TLS session-aware user authentication. *Computer* 2008; **41**(3): 59–65.
30. Rivest R, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 1978; **21**(2):120–126.
31. Harn L. Group authentication. *IEEE Transactions on Computers* 2013; **62**(9):1893–1898.