Authenticated Group Key Transfer Protocol Based on Secret Sharing

Lein Harn and Changlu Lin

Abstract—Key transfer protocols rely on a mutually trusted *key generation center* (KGC) to select session keys and transport session keys to all communication entities secretly. Most often, KGC encrypts session keys under another secret key shared with each entity during registration. In this paper, we propose an authenticated key transfer protocol based on secret sharing scheme that KGC can broadcast group key information to all group members at once and only authorized group members can recover the group key; but unauthorized users cannot recover the group key. The confidentiality of this transformation is information theoretically secure. We also provide authentication for transporting this group key. Goals and security threats of our proposed group key transfer protocol will be analyzed in detail.

Index Terms—Group key transfer protocol, session key, secret sharing, confidentiality, authentication.

1 Introduction

In most secure communication, the following two security functions are commonly considered:

- *Message confidentiality:* Message confidentiality ensures the sender that the message can be read only by an intended receiver.
- *Message authentication:* Message authentication ensures the receiver that the message was sent by a specified sender and the message was not altered en route.

To provide these two functions, one-time session keys need to be shared among communication entities to encrypt and authenticate messages. Thus, before exchanging communication messages, a key establishment protocol needs to distribute one-time secret session keys to all participating entities. The key establishment protocol also needs to provide confidentiality and authentication for session keys. According to [5], there are two types of key establishment protocols: *key transfer protocols* and *key agreement protocols*. Key transfer protocols rely on a mutually trusted *key generation center* (KGC) to select session keys and then transport session keys to all communication entities secretly. Most often, KGC encrypts session keys under another secret key shared with each entity during registration. In key agreement protocols, all communication entities are involved to determine session keys. The most commonly used key agreement protocol is Diffie-Hellman (DH) key agreement protocol [12]. In DH protocol, the session key is determined by exchanging public keys of two communication entities. Since the public key itself does not provide any authentication, a digital signature can be attached to the public key to provide authentication. However, DH public key distribution algorithm can only provide session key for two entities; not for a group more than two members.

- L. Harn is with the Department of Computer Science Electrical Engineering, University of Missouri-Kansas City, MO 64110-2499. E-mail: harnl@umkc.edu.
- C. Lin is with the Laboratory of Network Security and Cryptology, Fujian Normal University, Fujian 350007, P.R. China, and the State Key Laboratory of Information Security, Graduate University of Chinese Academy of Sciences, Beijing 100049, P.R. China. Email: lincl@is.ac.cn.

Manuscript received 6 Jan. 2009; revised 10 June 2009; accepted 16 July 2009; published online 11 Feb. 2009.
Recommended for acceptance by M. Eltoweissy.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2009-01-0008.
Digital Object Identifier no. 10.1109/TC.2010.40.

When a secure communication involves more than two entities, a group key is needed for all group members. Most well-known group key management protocols can be classified into two categories:

- Centralized group key management protocols: a group key generation center is engaged in managing the entire group.
- Distributed group key management protocols: there is no explicit group key distribution center, and each group member can contribute to the key generation and distribution.

The class of centralized group key management protocols is the most widely used group key management protocols. Harney et al. [15] proposed a group key management protocol that requires $O(n)$, where n is the size of group, encryptions to update a group key when a user is evicted or added if backward and forward secrecy are required. A set of scalable hierarchical structure-based group key protocols [10], [22], [27] have been proposed. Fiat and Naor [14] proposed a k-resistant protocol, i.e., coalitions of up to k users are secure, with each user storing $O(k \log k \log n)$ keys and the server broadcasting $O(k^2 \log^2 k \log n)$ messages per rekeying. Eltoweissy et al. [13] proposed a protocol based on Exclusion Basis Systems (EBS), a combinatorial formulation of the group key management problem, which allows protocol user to trade-off between the number of keys needed to be stored and the number of messages needed to be transmitted for each key update with no counter collusion solution provided.

Most distributed group key management protocols took natural generalization of the DH key agreement protocol, for example, Ingemarsson et al. [18], Steer et al. [28], Burmester and Desmedt [9], and Steiner et al. [29] followed this approach. In 1996, Steiner et al. proposed a natural extension of DH, named the group DH key exchange [29] and later in 2001, it has been enhanced with authentication services and has proved to be secure [6]. In 2006, Bohli [8] developed a framework for robust group key agreement that provides security against malicious insiders and active adversaries in an unauthenticated point-to-point network. Then, in 2007, Bresson et al. [7] constructed a generic authenticated group DH Key exchange and the algorithm is provably secure. Also, in 2007, Katz and Yung [19] proposed the first constant-round and fully scalable group DH protocol which is provably secure in the standard model (i.e., without assuming the existence of "random oracles"). The main feature of the group DH key exchange is to establish a secret group key among all group members without relying on a mutually trusted KGC.

There are other distributed group key management protocols based on non-DH key agreement approach. Tzeng [31] proposed a conference key agreement protocol based on discrete logarithm (DL) assumption with fault tolerance in recent years. The protocol can establish a conference key even if there are several malicious participants among the conference participants. However, the protocol requires each participant to create nn-power polynomials, where n is the number of participants; this is a serious encumbrance to efficiency. In 2008, Cheng and Laih [11] modified Tseng's conference key agreement protocol based on bilinear pairing. In 2009, Huang et al. [16] proposed a noninteractive protocol based on DL assumption to improve the efficiency of Tseng's protocol. One main concern of key agreement protocols is that since all communication entities are involved to determine session keys, the time delay of setting up this group key may be too long, especially when there are a large number of group members.

Secret sharing has been used to design group key distribution protocols. There are two different approaches using secret sharing: one assumes a trusted offline server active only at initialization [4], [14], [25], [3] and the other assumes an online trusted server, called the *key generation center*, always active. The first type of approach is also called the *key predistribution scheme*. In a key predistribution scheme, a trusted authority generates and distributes secret pieces of information to all users offline. At the beginning of a conference, users belonging to a privileged subset can compute individually a secret key common to this subset. A family of forbidden subsets of users must have no information about the value of the secret. The main disadvantage of this approach is to require every user to store a large size of secrets. The second type of approach requires an online server to be active [20] and this approach is similar to the model used in the IEEE 802.11i standard [17] that employs an online server to select a group key and transport it to each group member. However, the difference between this approach and the IEEE 802.11i is that, instead of encrypting the *group temporal key* (GTK) using the *key encryption key* (KEK) from the authentication server to each mobile client separately as specified in the IEEE 8-2.11i, the trusted KGC broadcasts group key information to all group members at once. In 1989, Laih et al. [20] proposed the first algorithm based on this approach using any (t, n) secret sharing scheme to distribute a group key to a group consisting of $(t-1)$ members. Later, there are some papers [2], [21], [25] following the same concept to propose ways to distribute group messages to multiple users. In this paper, we propose a solution based on this approach and provide confidentiality and authentication for distributing group keys. Furthermore, we classify attacks into insider and outsider attacks separately, and analyze our protocol under these attacks in detail.

We list following unique features of our proposed group key transfer protocol using secret sharing scheme.

- Each user needs to register at KGC to subscribe the group key transfer service and to establish a secret with KGC. Thus, a secure channel is needed initially to share this secret with each user. Later, KGC can transport the group key and interact with all group members in a broadcast channel.
- The confidentiality of group key distribution is information theoretically secure; that is, the security of this transfer of group key to each group member does not depend on any computational assumption.
- The authentication of the group key is achieved by broadcasting a single authentication message to all group members.

The rest of this paper is organized as follows: In the next section, we provide some preliminaries. In Section 3, we describe our main objective. In Section 4, we propose our authenticated group key transfer protocol. We analyze the security of our proposed protocol in Section 5. We conclude in Section 6.

2 Preliminaries

In this section, we introduce some fundamental backgrounds.

Definition 1 (Factoring Problem). *Let us choose two large safe primes p and q (i.e., primes such that $p' = \frac{p-1}{2}$ and $q' = \frac{q-1}{2}$ are also primes) and compute $n = pq$. n is made publicly known. Factoring problem is defined to compute factors p and q such that $n = pq$.*

Definition 2 (Factoring Assumption). *It is computationally intractable to solve the Factoring Problem.*

Secret sharing schemes were introduced by both Blakley [1] and Shamir [26] independently in 1979 as a solution for safeguarding cryptographic keys and have been studied extensively in the literatures. In a secret sharing scheme, a secret s is divided into n *shares* and shared among n shareholders in such a way that, with any t or more than t shares, it is able to reconstruct this secret; but, with fewer than t shares, it cannot reconstruct the secret. Such a scheme is called a (t, n) secret sharing, denoted as (t, n)-SS.

Shamir's (t, n)-SS. In Shamir's (t, n)-SS [26] based on Lagrange interpolating polynomial, there are n shareholders $\mathcal{U} = \{U_1, \ldots, U_n\}$ and a mutually trusted dealer D. The scheme consists of two algorithms:

1. Share generation algorithm: dealer D does the following:.
   - dealer D first picks a polynomial $f(x)$ of degree $(t-1)$ randomly: $f(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1}$, in which the secret $s = a_0 = f(0)$ and all coefficients $a_0, a_1, \ldots, a_{t-1}$ are in a finite field $\mathbb{F}_p = GF(p)$ with p elements.
   - D computes all shares: $s_i = f(i) (\bmod p)$ for $i = 1, \ldots, n$.
   - Then, D outputs a list of n shares $(s_1, s_2, \ldots, s_n)$ and distributes each share s_i to corresponding shareholder P_i privately.
2. Secret reconstruction algorithm: this algorithm takes any t shares $(s_{i_1}, \ldots, s_{i_t})$ as input, it can reconstruct the secret s as

$$\begin{aligned} s = f(0) &= \sum_{i \in A} s_i \beta_i \\ &= \sum_{i \in A} s_i \left(\prod_{j \in A - \{i\}} \frac{x_j}{x_j - x_i} \right) \quad (\bmod p), \end{aligned}$$

   where $A = \{i_1, \ldots, i_t\} \subseteq \{1, 2, \ldots, n\}$, β_i for $i \in A$ are Lagrange coefficients.

We note that the above scheme satisfies the basic security requirements of secret sharing scheme as follows: 1) with knowledge of any t or more than t shares, it can reconstruct the secret s easily; and 2) with knowledge of fewer than $(t-1)$ shares, it cannot reconstruct the secret s. Shamir's scheme is *information theoretically secure* since the scheme satisfies these two requirements without making any computational assumption. For more information on this scheme, readers can refer to the original paper [26].

In Shamir's (t, n)-SS, the secret of each shareholder is just the y-coordinate of $f(x)$ and the x-coordinate is made publicly known. However, in our proposed group key transfer protocol, for security reason, we need to keep both x-coordinate and y-coordinate as each user's secret. Furthermore, in Shamir's (t, n)-SS, the modulus p used for all computations is a prime number. In our proposed protocol, to prevent insider attack as we will explain this later, the modulus n used for computations is a composite integer. We should point out that finding a modular inverse is needed in secret reconstruction process. We can use Euclid's extended algorithm [30] to compute modular inverse without factoring the composite modulus n.

3 Objective

In this section, we first describe the model of our group key transfer protocol. Then, we present the security goals for our group transfer protocol.

3.1 Model

Group key transfer protocol relies on one trusted entity, KGC, to choose the key, which is then transported to each member involved. Each user is required to register at KGC for subscribing the key distribution service. The KGC keeps tracking all registered users and removing any unsubscribed users. During registration, KGC shares a secret with each user. In most key transfer protocol, KGC encrypts the randomly selected group key under the secret shared with each user during registration and sends the ciphertext to each group member separately. An authenticated message checksum is attached with the ciphertext to provide group key authenticity. In this approach, the confidentiality of group key is ensured using any encryption algorithm which is computationally secure. Our protocol uses secret sharing scheme to replace the encryption algorithm. A broadcast message is sent to all group members at once. The confidentiality of group key is information theoretically secure. In addition, the authentication of broadcasting message can be provided as a group authentication. This feature provides efficiency of our proposed protocol.

3.2 Goals

The main security goals for our group key transfer protocol are: 1) *key freshness*; 2) *key confidentiality*; and 3) *key authentication*.

Key freshness is to ensure that a group key has never been used before. Thus, a compromised group key cannot cause any further damage of group communication. Key confidentiality is to protect the group key such that it can only be recovered by authorized group members; but not by any un-authorized user. Key authentication is to provide assurance to authorized group members that the group key is distributed by KGC; but not by an attacker.

In our protocol, we only focus on protecting group key information broadcasted from KGC to all group members. The service request and challenge messages from users to KGC are not authenticated. Thus, an attacker can impersonate a user to request for a group key service. In addition, attacker can also modify information transmitted from users to KGC without being detected. We need to analyze security threats caused by these attacks. In our security analysis, we will conclude that none of these attacks can successfully attack to authorized group members since attackers can neither obtain the group key nor share a group key with authorized group members. User/message authentication and key confirmation can be easily incorporated into our protocol since each user has shared a secret key with KGC during registration. However, these security features are beyond the scope of our fundamental protocol. We will briefly discuss ways to provide user authentication, message authentication, and key confirmation in security analysis.

4 Our Proposed Protocol

Our authenticated group key transfer protocol consists of three processes: initialization of KGC, user registration, and group key generation and distribution. The detailed description is as follows:

Initialization of KGC. The KGC randomly chooses two safe primes p and q (i.e., primes such that $p' = \frac{p-1}{2}$ and $q' = \frac{q-1}{2}$ are also primes) and compute $n = pq$. n is made publicly known.

User Registration. Each user is required to register at KGC for subscribing the key distribution service. The KGC keeps tracking all registered users and removing any unsubscribed users. During registration, KGC shares a secret, (x_i, y_i), with each user U_i, where $x_i, y_i \in \mathbb{Z}_n^*$.

Group key generation and distribution. Upon receiving a group key generation request from any user, KGC needs to randomly selects a group key and access all shared secrets with group members. KGC needs to distribute this group key to all group members in a secure and authenticated manner. All communication between KGC and group members are in a broadcast channel. For example, we assume that a group consists of t members, $\{U_1, U_2, \ldots, U_t\}$, and shared secrets are (x_i, y_i), for $i = 1, \ldots, t$. The key generation and distribution process contains five steps.

- Step 1. The initiator sends a key generation request to KGC with a list of group members as $\{U_1, U_2, \ldots, U_t\}$.
- Step 2. KGC broadcasts the list of all participating members, $\{U_1, U_2, \ldots, U_t\}$, as a response.
- Step 3. Each participating group member needs to send a random challenge, $R_i \in \mathbb{Z}_n^*$, to KGC.
- Step 4. KGC randomly selects a group key, k, and generates an interpolated polynomial $f(x)$ with degree t to pass through $(t+1)$ points, $(0, k)$ and $(x_i, y_i \oplus R_i)$, for $i = 1, \ldots, t$. KGC also computes t additional points, P_i, for $i = 1, \ldots, t$, on $f(x)$ and $Auth = h(k, U_1, \ldots, U_t, R_2, \ldots, R_t, P_1, \ldots, P_t)$, where h is a one-way hash function. All computations on $f(x)$ are over $\mathbb{Z}_n^*$. KGC broadcasts $\{Auth, P_i\}$, for $i = 1, \ldots, t$, to all group members. All computations are performed in $\mathbb{Z}_n^*$.
- Step 5. For each group member, U_i, knowing the shared secret, $(x_i, y_i \oplus R_i)$, and t additional public points, P_i, for $i = 1, \ldots, t$, on $f(x)$, is able to compute the polynomial $f(x)$ and recover the group key $k = f(0)$. Then, U_i computes $h(k, U_1, \ldots, U_t, R_1, \ldots, R_t, P_1, \ldots, P_t)$ and checks whether this hash value is identical to $Auth$. If these two values are identical, U_i authenticates the group key is sent from KGC.

In Fig. 1, we illustrate this group key transfer protocol for a group containing three members, A, B, and C.

Remark 1. In our protocol, during registration, KGC shares a secret, (x_i, y_i), with each user U_i. Adding/removing any user does not need to update any existing shared secret. However, for distributing a secret group key involving t group members, KGC needs to broadcast a message containing $(t+1)$ elements to all group members. At the same time, each group member needs to compute a t-degree interpolating polynomial $f(x)$ to decrypt the secret group key. Thus, our proposed protocol is only suitable for distributing secret group key to a group with a small group size. If a group containing a large group size, such as applications in pay-per-view system, centralized group key distribution protocols, such as EBS protocol [13], can be used to reduce the length of broadcast message and computational load of each group member.

5 Security Analysis

In this section, we first consider two types of adversaries in our proposed protocol, *insider* and *outsider*. Then, we prove that our proposed protocol achieves the security goals mentioned in Section 3 and is against inside and outside attacks.

5.1 Attacks

Adversaries can be categorized into two types. The first type of adversaries are *outsiders* of a particular group. The outside attacker can try to recover the secret group key belonging to a group that the outsider is unauthorized to know. This attack is related to the confidentiality of group key. In our proposed protocol, anyone can send a request to KGC for requesting a group key service. The outside attacker may also impersonate a group user to request a group key service. In security analysis, we will show that the outside attacker gains nothing from this attack since the attacker cannot recover the group key. The second type of adversaries are *insiders* of a group who are authorized to know the secret group key; but inside attacker attempts to recover other member's secret shared with KGC. Since any insider of a group is able to recover the same group key, we need to prevent inside attacker knowing other member's secret shared with KGC.

The following theorem proves that our protocol can achieve the security goals we set previously.

Theorem 1. *The proposed protocol achieves the following security goals: 1) key freshness, 2) key confidentiality, and 3) key authentication.*

Proof. 1) Key freshness is ensured by KGC since a random group key is selected by KGC for each service request. In addition, the polynomial $f(x)$ used to recover the group key is a function of random challenge selected by each group member.

2) Key confidentiality is provided due to the security feature of a secret sharing scheme. KGC generates a tth degree polynomial $f(x)$ passing through $(t+1)$ points, $(0, k)$ and $(x_i, y_i \oplus R_i)$, for $i = 1, \ldots, t$, and makes t additional points publicly known. For each authorized group member, including the secret shared with KGC, he/she knows $(t+1)$ points in total on $f(x)$. Thus, any authorized group member is able to reconstruct the polynomial $f(x)$ and recover the group key k. However, for any unauthorized member (or *outsider*), there are only t points on $f(x)$ available. Thus, unauthorized member knows nothing about the group key. This property is information theoretically secure since there has no other computational assumption based upon.

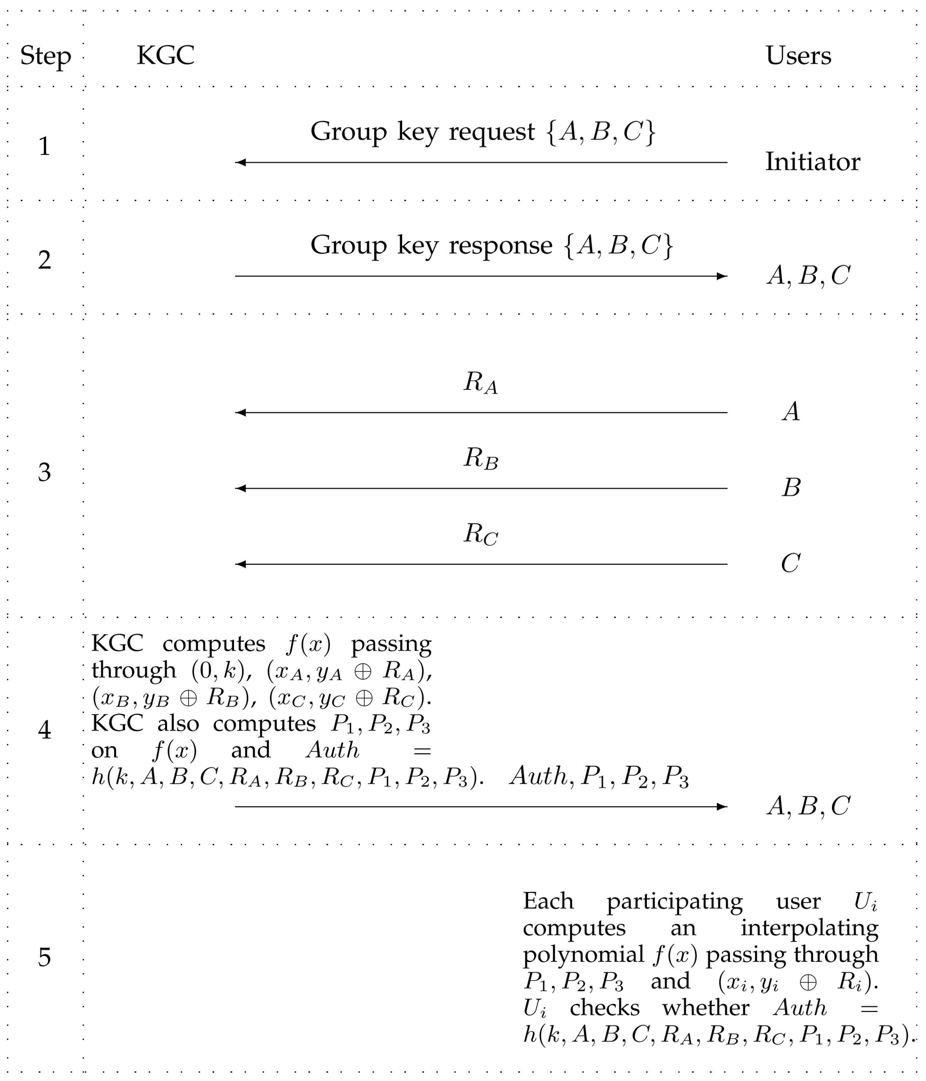
Step
KGC
Users
1
Group key request $\{A, B, C\}$
Initiator
2
Group key response $\{A, B, C\}$
A, B, C
3
R_A
A
R_B
B
R_C
C
4
KGC computes $f(x)$ passing through $(0, k)$, $(x_A, y_A \oplus R_A)$, $(x_B, y_B \oplus R_B)$, $(x_C, y_C \oplus R_C)$. KGC also computes P_1, P_2, P_3 on $f(x)$ and $Auth = h(k, A, B, C, R_A, R_B, R_C, P_1, P_2, P_3)$.
$Auth, P_1, P_2, P_3$
A, B, C
5
Each participating user U_i computes an interpolating polynomial $f(x)$ passing through P_1, P_2, P_3 and $(x_i, y_i \oplus R_i)$. U_i checks whether $Auth = h(k, A, B, C, R_A, R_B, R_C, P_1, P_2, P_3)$.

Fig. 1. Group key transfer protocol.

3) Key authentication is provided through the value $Auth$ in step 4. $Auth$ is a one-way hash output with the secret group key and all members' random challenges as input. Since the group key is known only to authorized group members and KGC, unauthorized members cannot forge this value. Any insider also cannot forge a group key without being detected since the group key is a function of the secret shared between each group member and KGC. In addition, any replay of P_i and $Auth$ of KGC in step 4 can be detected since the group key is a function of each group member's random challenge. □

Theorem 2 (Outsider attack). *Assume that an attacker who impersonates a group member for requesting a group key service, then the attacker can neither obtain the group key nor share a group key with any group member.*

Proof. Although any attacker can impersonate a group member to issue a service request to KGC without being detected and KGC will respond by sending group key information accordingly; however, the group key can only be recovered by any group member who shares a secret with KGC. This security feature is information theoretically secure.

If the attacker tries to reuse a compromised group key by replaying previously recorded key information from KGC, this attack cannot succeed in sharing this compromised group key with any group member since the group key is a function of each member's random challenge and the secret shared between group member and KGC. A compromised group key cannot be reused if each member selects a random challenge for every conference. □

Theorem 3 (Insider attack). *Assume that the protocol runs successfully v times and the applied factoring instances are intractable, then the secret (x_i, y_i) of each group member shared with KGC remains unknown to all other group members (and outsiders).*

Proof. For a group key service request, KGC generates a tth degree polynomial $f(x)$ passing through $(t+1)$ points, $(x_i, y_i \oplus R_i)$, for $i = 1, \ldots, t$. For each authorized group member, with knowledge of the secret shared with KGC and t public information, he/she knows $(t+1)$ points on $f(x)$. Thus, any authorized group member is able to reconstruct the polynomial $f(x)$. However, the secret (x_i, y_i) of each group member shared with KGC remains unknown to outsiders.

In our proposed protocol, group key service requests and challenges from group members are not authenticated. An adversary (insider) can make several service requests to KGC and forge challenges of the target group member. For example, the adversary makes two service requests for a group containing the adversary and the target group member. The adversary also forges the same challenge, R, of the target group member for these two services. The KGC generates $f_1(x)$ and $f_2(x)$, respectively. Thus, the adversary can obtain $y_{target} \oplus R = f_1(x_{target}) = f_2(x_{target})$. By subtracting these two polynomials, the adversary obtains a tth degree polynomial as

$$\begin{aligned} g(x) &= f_1(x_{target}) - f_2(x_{target}) \\ &= a_t x^t_{target} + a_{t-1} x^{t-1}_{target} + \cdots + a_1 x_{target} + a_0 = 0 \pmod n, \end{aligned}$$

where $a_i \in \mathbb{Z}_n$ for $i = 0, 1, \ldots, t$. It is commonly believed that the adversary needs to first solve two separate equations in $g(x) = 0 \pmod p$ and $g(x) = 0 \pmod q$, respectively, in order to solve the secret x_{target}. This is an intractable problem due to factoring assumption. Some well-known modern cryptosystems are also based on the same assumption. For example, the security of Rabin's cryptosystem [23] is based on a quadratic form of this equation, that is, $M^2 - C = 0 \pmod n$ where M is the message and C is the ciphertext. It has been shown that if the adversary can solve all four quadratic roots of this equation, then the adversary can factor the composite integer n. Also, the security of RSA cryptosystem [24] is based on a special form of this equation, that is, $M^e - C = 0 \pmod n$ where e is the public key, M is the message, C is the ciphertext. The security of RSA cryptosystem is generally believed to be based on factoring assumption. □

Remark 2. In our proposed protocol, we only focus on protecting group key information broadcasted from KGC to all group members. Here, we briefly explain how to provide user authentication and authenticate messages transmitted from group members to KGC. In our model, we assume that the KGC is a mutually trusted entity and each registered user, U_i, needs to share a secret, (x_i, y_i), with KGC during registration. *User authentication* can be achieved based on the knowledge of the shared secret between each user and KGC. In step 3 of our proposed protocol, user's challenge R_i to KGC can be authenticated by KGC if each user attaches an authentication value, $h((x_i, y_i), R_i)$, along with the challenge message. Furthermore, *key confirmation* can be done by asking each group member to send a key confirmation, $h((x_i, y_i), k)$, to KGC after step 5. Then, after receiving all key confirmations from group members, KGC sends a group key confirmation, $h((x_i, y_i), k, U_1, U_2, \ldots, U_t)$, to each group member U_i.

6 Conclusions

We have proposed an efficient group key transfer protocol based on secret sharing. Every user needs to register at a trusted KGC initially and preshare a secret with KGC. KGC broadcasts group key information to all group members at once. The confidentiality of our group key distribution is information theoretically secure. We provide group key authentication. Security analysis for possible attacks is included.

References

[1] G.R. Blakley, "Safeguarding Cryptographic Keys," *Proc. Am. Federation of Information Processing Soc. (AFIPS '79) Nat'l Computer Conf.*, vol. 48, pp. 313-317, 1979.

[2] S. Berkovits, "How to Broadcast a Secret," *Proc. Eurocrypt '91 Workshop Advances in Cryptology*, pp. 536-541, 1991.

[3] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," *Proc. Eurocrypt '84 Workshop Advances in Cryptology*, pp. 335-338, 1984.

[4] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly Secure Key Distribution for Dynamic Conferences," *Information and Computation*, vol. 146, no. 1, pp. 1-23, Oct. 1998.

[5] C. Boyd, "On Key Agreement and Conference Key Agreement," *Proc. Second Australasian Conf. Information Security and Privacy (ACISP '97)*, pp. 294-302, 1997.

[6] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater, "Provably Authenticated Group Diffie-Hellman Key Exchange," *Proc. ACM Conf. Computer and Comm. Security (CCS '01)*, pp. 255-264, 2001.

[7] E. Bresson, O. Chevassut, and D. Pointcheval, "Provably-Secure Authenticated Group Diffie-Hellman Key Exchange," *ACM Trans. Information and System Security*, vol. 10, no. 3, pp. 255-264, Aug. 2007.

[8] J.M. Bohli, "A Framework for Robust Group Key Agreement," *Proc. Int'l Conf. Computational Science and Applications (ICCSA '06)*, pp. 355-364, 2006.

[9] M. Burmester and Y.G. Desmedt, "A Secure and Efficient Conference Key Distribution System," *Proc. Eurocrypt '94 Workshop Advances in Cryptology*, pp. 275-286, 1994.

[10] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast Security: A Taxonomy and Some Efficient Constructions," *Proc. IEEE INFOCOM '99*, vol. 2, pp. 708-716, 1999.

[11] J.C. Cheng and C.S. Laih, "Conference Key Agreement Protocol with Non Interactive Fault-Tolerance Over Broadcast Network," *Int'l J. Information Security*, vol. 8, no. 1, pp. 37-48, 2009.

[12] W. Diffie and M.E. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory*, vol. IT-22, no. 6, pp. 644-654, Nov. 1976.

[13] M. Eltoweissy, M.H. Heydari, L. Morales, and I.H. Sudborough, "Combinatorial Optimization of Group Key Management," *J. Network and Systems Management*, vol. 12, no. 1, pp. 33-50, 2004.

[14] A. Fiat and M. Naor, "Broadcast Encryption," *Proc. 13th Ann. Int'l Cryptology Conf. Advances in Cryptology (Crypto '93)*, pp. 480-491, 1994.

[15] H. Harney, C. Muckenhirn, and T. Rivers, "Group Key Management Protocol (GKMP) Architecture," RFC 2094, July 1997.

[16] K.H. Huang, Y.F. Chung, H.H. Lee, F. Lai, and T.S. Chen, "A Conference Key Agreement Protocol with Fault-Tolerant Capability," *Computer Standards and Interfaces*, vol. 31, pp. 401-405, Jan. 2009.

[17] *IEEE 802.11i-2004: Amendment 6: Medium Access Control (MAC) Security Enhancements*, 2004.

[18] I. Ingemarsson, D.T. Tang, and C.K. Wong, "A Conference Key Distribution System," *IEEE Trans. Information Theory*, vol. IT-28, no. 5, pp. 714-720, Sept. 1982.

[19] J. Katz and M. Yung, "Scalable Protocols for Authenticated Group Key Exchange," *J. Cryptology*, vol. 20, pp. 85-113, 2007.

[20] C. Laih, J. Lee, and L. Harn, "A New Threshold Scheme and Its Application in Designing the Conference Key Distribution Cryptosystem," *Information Processing Letters*, vol. 32, pp. 95-99, 1989.

[21] C.H. Li and J. Pieprzyk, "Conference Key Agreement from Secret Sharing," *Proc. Fourth Australasian Conf. Information Security and Privacy (ACISP '99)*, pp. 64-76, 1999.

[22] A. Perrig, D. Song, and J.D. Tygar, "Elk, A New Protocol for Efficient Large-Group Key Distribution," *Proc. IEEE Symp. Security and Privacy*, pp. 247-262, 2001.

[23] M.O. Rabin, "Digitized Signatures and Public-Key Functions As Intractable As Factorization," Technical Report LCS/TR-212, MIT Laboratory for Computer Science, 1979.

[24] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *Comm. ACM*, vol. 21, pp. 120-126, 1978.

[25] G. Saze, "Generation of Key Predistribution Schemes Using Secret Sharing Schemes," *Discrete Applied Math.*, vol. 128, pp. 239-249, 2003.

[26] A. Shamir, "How to Share a Secret," *Comm. ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[27] A.T. Sherman and D.A. McGrew, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees," *IEEE Trans. Software Eng.*, vol. 29, no. 5, pp. 444-458, May 2003.

[28] D.G. Steer, L. Strawczynski, W. Diffie, and M.J. Wiener, "A Secure Audio Teleconference System," *Proc. Eighth Ann. Int'l Cryptology Conf. Advances in Cryptology (Crypto '88)*, pp. 520-528, 1988.

[29] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman Key Distribution Extended to Group Communication," *Proc. Third ACM Conf. Computer and Comm. Security (CCS '96)*, pp. 31-37, 1996.

[30] D.R. Stinson, *Cryptography Theory and Practice*, second ed., CRC Press, 2002.

[31] W.G. Tzeng, "A Secure Fault-Tolerant Conference Key Agreement Protocol," *IEEE Trans. Computers*, vol. 51, no. 4, pp. 373-379, Apr. 2002.