




# A secure and efficient on-line/off-line group key distribution protocol

Keju Meng<sup>1</sup> · Fuyou Miao<sup>1</sup>  · Yue Yu<sup>1</sup>

Received: 5 January 2018 / Revised: 11 June 2018 / Accepted: 7 September 2018 /  
Published online: 4 October 2018  
© Springer Science+Business Media, LLC, part of Springer Nature 2018

## Abstract

Nowadays, group communications are getting more and more popular. In order to secure the communication, all participating users need to share a common group key in advance. The paper proposes a secure and efficient group key distribution protocol based on Shamir's secret sharing scheme. In the protocol, (1) each user only needs to send registration message in privacy, while all the other messages can be transported in public. Meanwhile, (2) the scheme supports authentication for group keys without any assumption of hard mathematics problem. Moreover, (3) the protocol introduces the notion of on-line/off-line into group key distribution and thus the speeds of group key response and recovery are greatly improved. Analyses show that our scheme is resistant to passive attack, impersonation attack and reply attack.

**Keywords** Group key distribution · Group key authentication · Secret sharing · Lagrange interpolation polynomial · On-line/off-line

**Mathematics Subject Classification** 94A60 · 94A62

## 1 Introduction

With the rapid development of network, communication patterns are not limited to 1-to-1 or 1-to- $m$  (i.e. server/clients). Group communications with  $m$ -to- $m$  pattern become more and more popular. Before a secure group communication, all participating users are required to share a group key. Group key establishment protocols have been studied in many literatures.

---

Communicated by C. Blundo.

✉ Fuyou Miao  
mfy@ustc.edu.cn  
Keju Meng  
mkj@mail.ustc.edu.cn  
Yue Yu  
yuyue204@mail.ustc.edu.cn

<sup>1</sup> University of Science and Technology of China, No. 96, JinZhai Road Baohe District, Hefei 230026, Anhui, China

According to whether a trusted key generation center (KGC) exists or not, these protocols are classified into two classes: group key agreement (or exchange) protocols and group key distribution (or transfer) protocols.

In group key agreement protocols, there is not a KGC trusted by all users. Initially, the group key agreement protocol [3] based on Diffie–Hellman algorithm [6] was proposed by Ingemarsson et al. in 1982. The protocol requires all  $n$  participants to communicate synchronously and needs  $n - 1$  rounds of interaction to establish a group key. Steiner et al. [28] extends basic Diffie–Hellman key exchange to a  $n$ -party Diffie–Hellman key exchange. In 2004, Kim et al. [16] proposed a fault-tolerant protocol named Tree-based Group Diffie–Hellman. Protocols [4,14,36] are also based on Diffie–Hellman algorithm. Moreover, there are some group key agreement protocols based on other methods. For example, Chen et al. [5] proposed a key agreement protocol based on bilinear pairing to improve the key establishment efficiency. Irshad et al. [15] proposed the protocol based on Chebyshev chaotic map without a registration center. Because there is no KGC and all the users are equal in these protocols, more communication cost is required during key establishment.

Distinct from group key agreement protocols, there exists a KGC trusted by all users in group key distribution protocols [11,17,24,31]. The group key is generated and distributed by the KGC, so that group key establishment can be accomplished more efficiently. However, it is difficult to distribute a group key for the following reasons:

1. Since a group key is transferred to multiple users, it is easier to be intercepted during distribution.
2. Group users may change from time to time. When a user leaves or joins a group, the group key needs to be updated such that users out of the group have no information about the new group key.
3. Even if a group does not change for a long time, a session key still needs to be updated after a period of time. Otherwise, it could be cracked by adversaries.

As a group-oriented cryptographic tool, secret sharing has the potential to address the above problems. Therefore, based on Shamir's  $(t, n)$  threshold secret sharing (SS) scheme, we propose an on-line/off-line group key distribution protocol.

$(t, n)$  threshold SS was first proposed by Shamir [26] and Blakley [2] separately in 1979. In a  $(t, n)$  SS scheme, a secret is divided into  $n$  shares and each share is allocated to a shareholder secretly such that any  $t$  or more than  $t$  shareholders can reconstruct the secret while less than  $t$  shareholders cannot obtain the secret. Shamir's  $(t, n)$  threshold SS and Blakley's  $(t, n)$  threshold SS are based on polynomial interpolation and hyperplane geometry respectively, while Mignotte [22] and Asmuth [1] et al.'s schemes are both based on Chinese Remainder Theorem. All these schemes are not based on any hard mathematical problems.

There are some literatures using  $(t, n)$  threshold SS to design group key distribution protocols [8,21,29,35]. Guo et al. [9] proposed a group key distribution protocol based on Asmuth–Bloom's  $(t, n)$  threshold SS. Harn and Lin proposed an authenticated group key transfer protocol [12] based on Shamir's  $(t, n)$  threshold SS. However, the protocol is not information theoretically secure since it employs large integer factorization problem to thwart Insider attack. Liu et al. [20] tried to improve the security of Harn–Lin's protocol by utilizing two hash functions, but their protocol is also based on large integer factorization problem.

In these protocols [12,20,25], one-way hash functions are used to support group key authentication. The KGC announces the hash value of group key in advance, so that users can verify whether the group key is correctly distributed by the KGC. In our proposed protocol, hash function is not only used for authentication but also utilized to compute an offset to protect the group key.

In the above distribution protocols, the KGC starts to compute and distribute group key after it has received all users' requests. However, in the scenes where quick response is required, more efficient distribution protocol is needed. Therefore, we introduce on-line/off-line method to speed the response of group key distribution protocol for the first time.

The concept of on-line/off-line was first proposed by Even et al. [7] to design an efficient digital signature scheme. The idea is that the signature generating procedure is divided into two phases. Part of signature can be computed during off-line phase (before the signed message is given), so that on-line phase (after the signed message is received) works very fast. Subsequently, Tanaka [30], Liu [19] and some others [10,13,18,27,34], extended the notion to design some other schemes. In our group key distribution protocol, the period before group key requests are sent to the KGC is the off-line phase. KGC can carry out most calculation of group key generation during this phase. When KGC receives group key request messages from users, the protocol goes into on-line phase. Then, KGC generates group key and distributes it to users. In this way, KGC has a rapid response in on-line phase.

According to the above, we summarize the characteristics of our on-line/off-line group key distribution protocol as below.

- Each user just needs to share a coordinate with the KGC in privacy during registration. And then, all messages can be transported publicly.
- All users share a common group key, and they can verify whether or not the key is sent from KGC.
- When some users drop out of a group, they will not get any information about new group keys.
- The process of group key distribution is divided into on-line and off-line phases. KGC carries out most computation during off-line phase, so that it distributes the key to users rapidly in on-line phase.
- Our protocol is resistant to passive attack, impersonation attack, and reply attack. Moreover, it is not based on any hard mathematics problems.

The rest of the paper is organized as follows: In next section, we present some preliminaries about Shamir's  $(t, n)$  threshold SS and Harn–Lin's protocol. The entities and attack models are given in Sect. 3. In Sect. 4, we propose our group key distribution protocol in detail. Correctness analysis and security analysis are given in Sects. 5 and 6 respectively. In Sect. 7, we compare our protocol to related ones. Section 8 presents some experimental data and Sect. 9 concludes the paper.

## 2 Preliminaries

### 2.1 Shamir's $(t, n)$ threshold SS

Shamir's  $(t, n)$  threshold SS is based on polynomial interpolation. It consists of the following two steps:

**Share generation** At first, the dealer  $D$  randomly chooses a polynomial  $f(x)$  of degree  $t - 1$ :  $f(x) = a_0 + a_1x + \dots + a_{t-1}x^{t-1} \pmod{p}$ . The constant term  $a_0$  is regarded as the secret  $s$ , i.e.,  $s = f(0) = a_0$ . All the coefficients  $a_0, a_1, \dots, a_{t-1}$  are in the finite field  $\text{GF}(p)$ , where  $p$  is a prime number. The dealer  $D$  selects  $n$  abscissas  $\mathcal{X} = 1, 2, \dots, n$  to generate a share set  $\delta = \{s_i | f(x_i), i \in \mathcal{X}\}$ . Then, the dealer  $D$  distributes each share  $s_i$  to the corresponding shareholder  $U_i$  secretly while makes all the abscissas  $x_i$  known to the public.

**Secret reconstruction** When  $m$  ( $m \geq t$ ) legal shareholders attempt to recover the secret  $s$ , they pool shares together privately. As a result, each shareholder get a share set  $\delta_m = \{s_i | f(x_i), i \in \mathcal{X}_m\}$  associated to the abscissas set  $\mathcal{X}_m = \{1, 2, \dots, m\}$ . Then, the secret  $s$  can be computed as

$$s = f(0) = \sum_{i \in \mathcal{X}_m} s_i \prod_{j \in \mathcal{X}_m, j \neq i} \frac{-x_j}{x_i - x_j} \text{ mod } p$$

where  $s_i \prod_{j \in \mathcal{X}_m, j \neq i} \frac{-x_j}{x_i - x_j}$  is a *Lagrange component*.

**Remark 2.1** In Shamir’s  $(t, n)$  threshold SS, if a shareholder gets any  $t$  or more than  $t$  shares, it can compute the secret  $s$  easily by Lagrange Interpolation Polynomial. However, any less than  $t$  shareholders cannot obtain any information about the secret. Likewise, less than  $t - 1$  *Lagrange components* also cannot recover generation polynomial  $f(x)$  or the secret  $s$ . It plays an important role in our proposed protocol. Due to the problem without any computational assumption, Shamir’s  $(t, n)$  threshold SS is an information theoretically secure scheme.

### 2.2 Harn–Lin’s protocol

In this protocol, the KGC first picks a RSA-number  $n$ , where  $n = p * q$ . Both  $p$  and  $q$  are large safe primes. Then, KGC shares a coordinate  $(x_i, y_i)$  with each user  $U_i$  during registration, where  $i = 1, 2, \dots, m$ . The group key generation and distribution process is described as follow:

**Step 1** The initiator sends a key initiation message to KGC.

**Step 2** KGC broadcasts a response message to all users.

**Step 3** Each user sends a random challenge  $R_i$  to KGC, where  $R_i \in \mathbb{Z}_n$ .

**Step 4** KGC randomly selects a group key  $k$  and constructs an  $m$ -th degree polynomial  $f(x)$  to pass through  $(m + 1)$  coordinates,  $(0, k)$  and  $(x_i, y_i \oplus R_i)$ , for  $i = 1, 2, \dots, m$ . KGC computes  $m$  additional coordinates  $P_i$  on  $f(x)$  and authentication information  $auth = h(k, U_1, \dots, U_m, R_1, \dots, R_m)$ , where  $h(\cdot)$  is a public one-way hash function. KGC distributes  $\{auth, P_1, \dots, P_m\}$  to all users publicly.

**Step 5** Each user  $U_i$  can recover the group key  $k$  by one private coordinate  $(x_i, y_i)$  and  $m$  public coordinates  $P_i$ , for  $i = 1, 2, \dots, m$ . Then, it computes  $auth = h(k, U_1, \dots, U_m, R_1, \dots, R_m)$  to check whether or not the key is sent from KGC.

**Remark 2.2** In Harn–Lin’s protocol, KGC should choose a larger parameter  $n$  as the module of polynomials to replace the module  $p$  in Shamir’s  $(t, n)$  threshold SS, where  $n$  is a RSA-number. In this way, the coordinate  $(x_i, y_i)$  of  $U_i$  can be reused many times and kept private. Harn and Lin use the big integer factorization problem to thwart Insider attack skillfully. Because this protocol is based on not only Shamir’s  $(t, n)$  threshold SS but also hard math problem, it is not information theoretically secure. Readers can read the original paper [21] to acquire more detailed information about the problem. However, all computations have to be over  $\mathbb{Z}_n$ , and KGC cannot carry out any calculations of group key before it receives all users’ random challenges. Consequently, Harn–Lin’s protocol is inefficient.

### 3 Entities and attack models

In our group key distribution protocol, entities are classified into three types: (1) KGC, (2) User and (3) adversary. Furthermore, there are three attack models for adversaries: (1) passive attack, (2) impersonation attack, and (3) reply attack.

#### 3.1 Entities

##### 3.1.1 KGC

KGC is the entity trusted by all users and it is in charge of distributing group keys to the users. Firstly, it attests to users' identities and confers with each legal user on a private coordinate secretly. In fact, the process is similar to the shares generation in a  $(t, n)$  threshold SS scheme. Then, before users initiate a group key request, KGC goes into off-line phase to carry out part of computations about group key generation. Next, when KGC receives a group key initialization message, it goes into on-line phase to complete key generation. Finally, it sends group key distribution messages to users quickly.

##### 3.1.2 User

A user is an entity who shares a private coordinate with KGC during registration. Afterwards, users are expected to maintain their own coordinates secure all the time, because the coordinates will be reused time and again. When users need a group key, each of them submits a request to KGC. After a user receives its corresponding distribution message, it recovers a session key and verifies whether or not the key is transferred from KGC.

##### 3.1.3 Adversary

An adversary is an entity who wants to attack the protocol. Suppose that all users form a set  $\mathcal{U}$ . The adversaries are classified as Outsiders and Insiders according to whether or not they are in  $\mathcal{U}$ .

(1) Insider: If a user not only wants to obtain the group key, but also attempts to derive coordinates kept by other users in  $\mathcal{U}$ , it is named Insider.

(2) Outsider: If any others not in  $\mathcal{U}$  want to attack the protocol, they are called Outsiders. Outsiders always aim at capturing group key of  $\mathcal{U}$ , or preventing users in  $\mathcal{U}$  from obtaining a valid group key.

#### 3.2 Attack models

##### 3.2.1 Passive attack

Passive attack means that adversaries break confidentiality of a protocol by capturing messages among legal members. In our protocol, all messages during distribution phase are transported publicly, so that adversaries are able to easily capture the messages.

### 3.2.2 Impersonation attack

Impersonation attack refers to the fact that an entity pretends to be another to attack a protocol. In our protocol, it means that an Outsider sends group key request to KGC in the name of a legal user, or an adversary pretends to be KGC to distribute group key.

### 3.2.3 Reply attack

Reply attack is that an entity resends outdated messages to others. In our protocol, it is subdivided into two types:

- (1) An Outsider resends an outdated group key request to KGC.
- (2) An adversary redistributes an outdated group key to users.

## 4 Our proposed protocol

In this section, we show our group key distribution protocol in detail. It consists of three phases: (1) Preparatory phase, (2) Distribution phase, (3) Group key recovery and authentication.

### 4.1 Symbol definition

Before describing the protocol, we first define some notations as listed in Table 1.

### 4.2 Preparatory phase

#### 4.2.1 Initialization of KGC

The KGC selects a one-way hash function  $h(\cdot)$  and a random prime  $p$ . Then, it makes them publicly known.

#### 4.2.2 Users registration

As described above, each user should register with KGC before it joins in the group. In this process, each user  $U_i$  is required to share a private coordinate  $(x_i, y_i)$  with KGC, where both  $x_i$  and  $y_i$  are in the finite field  $\text{GF}(p)$ . KGC should guarantee each  $x_i \neq 0$  and  $x_i \neq x_j$  for  $i \neq j$ . Each user just needs to register only once. Then, it makes its identity  $U_i$  public while keeps its coordinate  $(x_i, y_i)$  private (Fig. 1).

### 4.3 Distribution phase

#### 4.3.1 Off-line phase

**Step 1** Suppose that there are total  $m$  legal users who have registered with KGC. They form a user group  $\mathcal{U} = \{U_1, U_2, \dots, U_m\}$  and their private coordinates constitute a set  $\Omega = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ .

**Table 1** Notations

Symbol	Descriptions
$m$	Number of legal users
$h(\cdot)$	Hash function
$k$	Group key selected by KGC
$f(x)$	Group key generation function
$U_i$	The $i^{th}$ user
$\mathcal{U}$	Group of legal users $\mathcal{U} = \{U_1, U_2, \dots, U_m\}$
$\sigma$	Group communication identifier
$I$	Group key initialization message
$R_\sigma$	Response message
$M_{\sigma,i}$	Request message of $U_i$
$(x_i, y_i)$	Private and permanent coordinate of user $U_i$
$(x_i^*, y_i^*)$	Public coordinate selected by user $U_i$
$g_i(x)$	Linear function constructed by $(x_i, y_i)$ and $(x_i^*, y_i^*)$
$g_i^{-1}(y)$	Inverse of $g_i(x)$
$(x'_i, y'_i)$	Ephemeral coordinate select by KGC for user $U_i$
$\mathbf{x}_1, \dots, \mathbf{x}_m$	Public abscissas included in $R_\sigma$
$d'_i$	Original group key distribution information
$d_i$	Protected group key distribution information
$\mathcal{K}_i$	Distribution message for $U_i$
$\Delta_i$	Lagrange component of $U_i$
$k_i$	Session key computed by $U_i$

**Step 2** KGC randomly generates a polynomial  $f(x) = a_0 + a_1x + \dots + a_mx^m$  of degree  $m$  and chooses  $a_0$  as the group key  $k$ , i.e.,  $k = f(0) = a_0$ .

**Step 3** KGC picks  $2m$  different coordinates on  $f(x)$  to form other two sets  $\Omega_1 = \{(x'_1, y'_1), (x'_2, y'_2), \dots, (x'_m, y'_m)\}$  and  $\Omega_2 = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_m, \mathbf{y}_m)\}$  such that  $\Omega \cap \Omega_1 = \Omega \cap \Omega_2 = \emptyset$ .

**Step 4** KGC uses each  $x'_i$  ( $i = 1, 2, \dots, m$ ) in  $\Omega_1$  and all elements in  $\Omega_2$  to compute original group key distribution information such as

$$d'_i = \sum_{t=1}^m \mathbf{y}_t \frac{-x'_i}{\mathbf{x}_t - x'_i} \prod_{j=1, j \neq t}^m \frac{-\mathbf{x}_j}{\mathbf{x}_t - \mathbf{x}_j} \text{ mod } p$$

**Step 5** KGC computes an offset  $h(x'_i, y'_i)$  to generate a value  $d_i = d'_i + h(x'_i, y'_i) \text{ mod } p$  as protected group key distribution information.

### 4.3.2 On-line phase

**Step 1** The initiator sends a group key initialization message  $I$  to KGC.

**Step 2** When KGC receives the initialization message, it goes into on-line phase and broadcasts a message  $R_\sigma = \{\sigma, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$  as response, where  $\sigma$  is a group communication identifier selected by KGC.

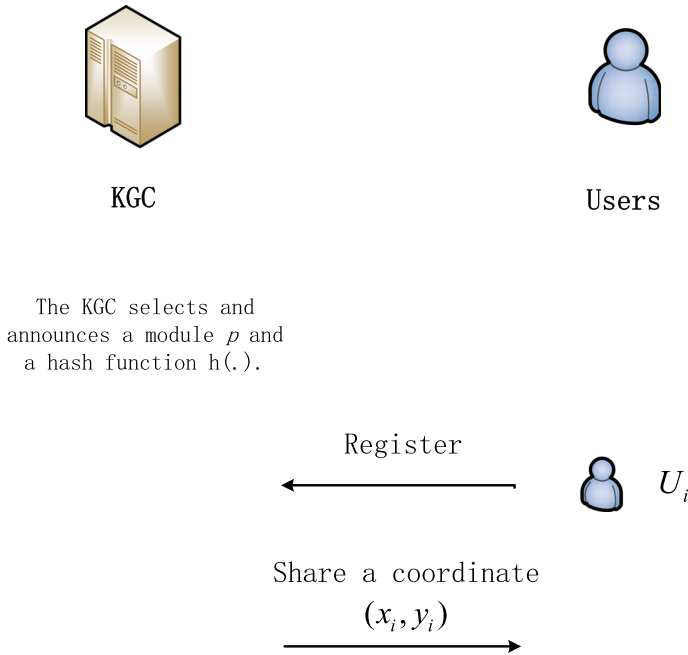


Fig. 1 Sketch of users registration

**Step 3** Each user  $U_i$  randomly picks a coordinate  $(x_i^*, y_i^*)$ , where  $x_i^* \neq x_i$ . Then it sends to KGC a request message  $M_{\sigma,i}$  including the group communication identifier  $\sigma$ , its identity  $U_i$  and the selected coordinate  $(x_i^*, y_i^*)$ , such as

$$M_{\sigma,i} = \{\sigma, U_i, (x_i^*, y_i^*)\}$$

**Step 4** For each user  $U_i$ , KGC keeps a private coordinate  $(x_i, y_i)$ , and receives  $(x_i^*, y_i^*)$  in message  $M_{\sigma,i}$ . It uses the two coordinates to construct a linear function

$$g_i(x) = y_i \frac{x - x_i^*}{x_i - x_i^*} + y_i^* \frac{x - x_i}{x_i^* - x_i} \pmod p$$

**Step 5** KGC uses  $(x'_i, y'_i)$  in  $\Omega_1$  to compute two values  $g_i(x'_i)$  and  $g_i^{-1}(y'_i)$ , where  $g_i^{-1}(y)$  is the inverse of  $g_i(x)$ .

**Step 6** KGC generates and sends distribution message  $\mathcal{K}_i = \{U_i, g_i(x'_i), g_i^{-1}(y'_i), d_i, h(k, \sigma)\}$  to the corresponding user  $U_i$ , where  $h(k, \sigma)$  is authentication information about the group key.

**Step 7** KGC completes the group key distribution and goes into off-line phase again to wait for next group key request.

#### 4.4 Group key recovery and authentication

**Step 1** Each user  $U_i$  also constructs  $g_i(x)$  and  $g_i^{-1}(y)$  by  $(x_i, y_i)$  and  $(x_i^*, y_i^*)$ . After  $U_i$  receives message  $\mathcal{K}_i$  from KGC, it recovers  $x'_i = g_i^{-1}(g_i(x'_i))$  and  $y'_i = g_i(g_i^{-1}(y'_i))$ .



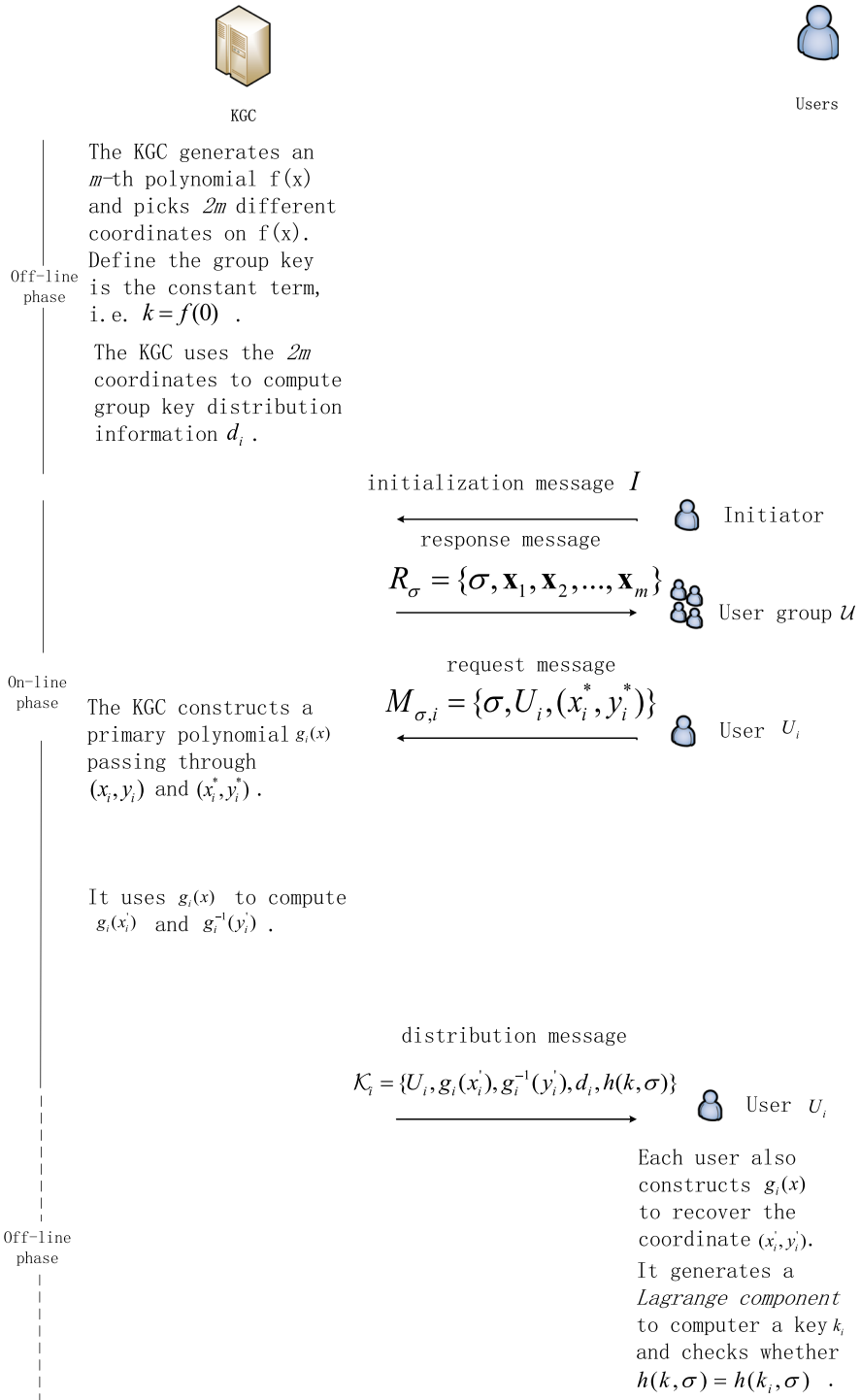


Fig. 2 Diagrammatic sketch of distribution phase, group key recovery and authentication

**Step 2** Each user  $U_i$  uses the coordinate  $(x'_i, y'_i)$ ,  $m$  public abscissas  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$  in message  $R_\sigma$ , to compute a *Lagrange component*, such as

$$\Delta_i = y'_i \prod_{j=1}^m \frac{-\mathbf{x}_j}{x'_i - \mathbf{x}_j} \text{ mod } p$$

**Step 3** The group key can be computed as

$$k_i = d_i + \Delta_i - h(x'_i, y'_i) \text{ mod } p$$

**Step 4** Each user  $U_i$  also uses the same one-way hash function to compute a value

$$h_i = h(k_i, \sigma)$$

If  $h_i = h\{k, \sigma\}$  holds, it means that the group key is the correct key sent from KGC, i.e.  $k_i = k$ . Then,  $U_i$  is allowed to use the certified key  $k$  to communicate with others in  $\mathcal{U}$ . Otherwise, users should initiate a new group key require to ensure their communication security (Fig. 2).

### 5 Correctness analysis

In this section, we give two theorems to prove the correctness of our proposed protocol: (1) all users in  $\mathcal{U}$  can obtain a common group key; (2) the group key can be validated by the authentication information  $h\{k, \sigma\}$ .

**Theorem 5.1** *All the legal users can compute a common group key  $k$ , i.e.  $k = d_i + \Delta_i - h(x'_i, y'_i) \text{ mod } p = d_j + \Delta_j - h(x'_j, y'_j) \text{ mod } p$ , where  $i \neq j$ .*

**Proof** Like secret reconstruction in Shamir’s  $(t, n)$  threshold SS, each user  $U_i$  requires at least  $m + 1$  *Lagrange components* to recover the group key, due to the generation function  $f(x)$  with degree  $m$ . However, KGC uses only  $m$  coordinates  $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_m, \mathbf{y}_m)$  and the abscissa  $x'_i$  to compute  $d'_i$ , which is the sum of  $m$  *Lagrange components*, and it sends the value  $d_i = d'_i + h(x'_i, y'_i) \text{ mod } p$  to  $U_i$ . Therefore, the user  $U_i$  need just compute its own *Lagrange component*  $\Delta_i$  and the offset  $h(x'_i, y'_i)$  to recover the group key  $k$ . That is

$$\begin{aligned} & d_i + \Delta_i - h(x'_i, y'_i) \text{ mod } p \\ &= d'_i + h(x'_i, y'_i) + \Delta_i - h(x'_i, y'_i) \text{ mod } p \\ &= d'_i + \Delta_i \text{ mod } p \\ &= y'_i \prod_{j=1}^m \frac{-\mathbf{x}_j}{x'_i - \mathbf{x}_j} + \sum_{t=1}^m y_t \frac{-x'_i}{\mathbf{x}_t - x'_i} \prod_{j=1, j \neq t}^m \frac{-\mathbf{x}_j}{\mathbf{x}_t - \mathbf{x}_j} \text{ mod } p \\ &= k \end{aligned}$$

□

**Theorem 5.2** *Each user  $U_i$  is assured that its calculated group key is sent from KGC, if  $h_i = h(k, \sigma)$  holds.*

**Proof** During *distribution phase*, only KGC knows the group key  $k$ . Thus, the valid authentication information of  $k$  can be computed by none but KGC. After *distribution phase*, each user  $U_i$  computes a key  $k_i$ . It also uses the hash function  $h(\cdot)$  with  $k_i$  and  $\sigma$  to compute  $h_i = h(k_i, \sigma)$ . Obviously, if and only if  $h_i = h(k, \sigma)$  (without regard for hash collision), its calculated key  $k_i$  is equal to  $k$  which is selected by KGC. □

## 6 Security analysis

Detailed security analyses are presented in this section. We first give a lemma as the security foundation. Then, five theorems are presented to prove that our protocol is resistant to passive attack, impersonal attack and reply attack.

Note that all the calculations are over finite field  $GF(p)$ , thus an event is deemed to be impossible if the probability of its occurrence is equal to or less than  $1/p$ .

**Lemma 6.1** *Suppose that there are  $t$  public coordinates  $(x_1, y_1), (x_2, y_2), \dots, (x_t, y_t)$ , (each  $x_i \neq 0$ ), on a generation function  $g(x)$  with degree  $t - 1$ , such as  $g(x) = a_0 + a_1x + a_2x^2 + \dots + a_t x^t \pmod p$ , where all the parameters are in the finite field  $GF(p)$ . The probability of reconstructing generation function  $g(x)$  by the  $t$  known coordinates is only  $1/p$ .*

**Proof** Obviously, there are  $t + 1$  unknown parameters  $a_0, a_1, \dots, a_t$  for the function  $g(x)$  with degree  $t$ . Now, assume that the constant term  $a_0$  is a known quantity. It means we get an extra coordinate  $(x_0, y_0)$ , where  $x_0 = 0$  and  $y_0 = a_0$ . In this way, we can use Lagrange Interpolation Polynomial to construct a function, such as

$$G(x) = \sum_{i=0}^t y_i \prod_{j=i, j \neq i}^t \frac{x - x_j}{x_i - x_j} \pmod p$$

Due to  $a_0 \in \mathbb{Z}_p$ , there are totally  $p$  candidates for the value of  $a_0$ . Therefore, different  $p$  functions can be computed corresponding to the different  $p$  possible values of  $a_0$ , but only one function is the correct generation function  $f(x)$ . In other words, the probability of reconstructing  $t$ -th generation function  $g(x)$  by  $t$  coordinates is only  $1/p$ .  $\square$

### 6.1 Resistance to passive attack

**Model of passive attack** During *distribution phase*, all the messages are transported publicly. Thus, adversaries can easily capture response message  $R_\sigma$ , request messages  $M_{\sigma,i}$ , and distribution messages  $\mathcal{K}_i$ .

**Theorem 6.1** *Outsiders cannot obtain the group key and no adversary can derive the private coordinates of legal users by passive attack.*

**Proof** As for an Outsider, if it wants to obtain the group key, there are two methods:

- (1) The Outsider attempts to recover the generation function  $f(x)$ . In order to reconstruct  $f(x)$  of degree  $m$ , an Outsider needs at least  $m + 1$  coordinates on  $f(x)$  or  $m + 1$  valid *Lagrange components*. However, no complete coordinate on  $f(x)$  is included in these public messages. KGC broadcasts only  $m$  protected sums of *Lagrange components*. Hence, the generation function  $f(x)$  cannot be recovered.
- (2) The Outsider tries to obtain a valid *Lagrange component*  $\Delta_i$  and the corresponding offset  $h(x'_i, y'_i)$  to compute  $k = d_i + \Delta_i - h(x'_i, y'_i) \pmod p$  just like a legal user  $U_i$ . If so, it must know the exact coordinate  $(x'_i, y'_i)$ , because of the following two reasons:

- I: In equation  $\Delta_i = y'_i \prod_{j=1}^m \frac{-x_j}{x'_i - x_j} \pmod p$ ,  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$  are public while  $x'_i$  and  $y'_i$  are unknown. So, the Outsider needs to know  $x'_i$  and  $y'_i$  to compute  $\Delta_i$ .
- II: Because of unidirectional characteristic of one-way hash function, the Outsider must know the values of  $x'_i$  and  $y'_i$  to compute the offset  $h(x'_i, y'_i)$ .

However, KGC only releases  $g_i(x'_i)$  and  $g_i^{-1}(y'_i)$  included in  $\mathcal{K}_i$ . The Outsider still needs the function  $g_i(x)$  and  $g_i^{-1}(y)$  to compute  $(x'_i, y'_i)$ . During *on-line phase*, the Outsider can capture only one coordinate  $(x_i^*, y_i^*)$  on  $g_i(x)$  from  $M_{\sigma,i}$ . On the basis of Lemma 6.1., the probability of reconstructing the linear polynomial  $g_i(x)$  from only one coordinate is  $1/p$ .

If an Outsider wants to obtain a private coordinate  $(x_i, y_i)$ , the probability is much less than  $1/p$ . Because even if an Outsider gets one function  $g_i(x)$ , it still does not know which coordinate on  $g_i(x)$  is  $(x_i, y_i)$ . It must capture at least two functions  $g_i(x)$ s from different group key distribution processes, and the intersection of these function images is  $(x_i, y_i)$ . But actually, the probability of an Outsider obtaining any one function  $g_i(x)$  is only  $1/p$ , not to mention more functions.

As for an Insider, on the one hand, it is a legal user in  $\mathcal{U}$ , so that it is able to compute the group key easily from its corresponding distribution message. On the other hand, it still wants to obtain some private coordinates  $(x_i, y_i)$  of other users. For this purpose, an Insider also needs to know  $(x'_i, y'_i)$  as foundation. There are two methods that might work for an Insider:

- I: If the Insider attempts to recover the function  $g_i(x)$  by only one coordinate  $(x_i^*, y_i^*)$  like an Outsider, the probability has been given above and it is much less than  $1/p$ .
- II: The Insider uses group key  $k$  to compute  $\tau_i = (k - d_i) \bmod p$  and get an equation

$$\tau_i = \Delta_i - h(x'_i, y'_i) \bmod p$$

$$\tau_i = y'_i \prod_{j=1}^m \frac{-x_j}{x'_i - x_j} - h(x'_i, y'_i) \bmod p$$

In this equation, only  $x'_i$  and  $y'_i$  are unknown. However,  $h(x'_i, y'_i)$  is a one-way hash value. It is impossible to compute exact  $(x'_i, y'_i)$  from the equation. Otherwise, it runs counter to the unidirectional characteristic of  $h(\cdot)$ .

As a result, an Outsider could capture neither the group key nor any valid coordinates by passive attack. For an Insider, it can only obtain the group key but no other users' coordinates can be derived. □

### 6.2 Resistance to impersonation attack

**Model of impersonation attack 1** An Outsider can pretend to be a legal user  $U_i$ . And then, it also sends a request message  $M_{\sigma,i} = \{\sigma, U_i, (x_i^*, y_i^*)\}$  to KGC and receives a corresponding distribution message  $\mathcal{K}_i = \{g_i(x'_i), g_i^{-1}(y'_i), d_i, h(k, \sigma)\}$ .

**Theorem 6.2** *Outsiders can neither obtain the group key nor prevent other legal users from obtaining the group key by impersonation attack 1.*

**Proof** In order to compute the group key according to  $\mathcal{K}_i$ , the Outsider must reconstruct the function  $g_i(x) = y_i \frac{x-x_i^*}{x_i-x_i^*} + y_i^* \frac{x-x_i}{x_i^*-x_i} \bmod p$  to recover the coordinate  $(x'_i, y'_i)$ . Although the Outsider can select  $(x_i^*, y_i^*)$  by itself, it still does not know the exact coordinate  $(x_i, y_i)$ . On account of Lemma 6.1., the Outsider cannot reconstruct the function  $g_i(x)$ . Therefore, it is not able to obtain the group key by impersonation attack 1.

Meanwhile, for a legal user  $U_j$ , it can still recover the coordinate  $(x'_j, y'_j)$  to compute the group key as usual. That is to say, legal users are not influenced by the Outsider's mock request message. □

**Model of impersonation attack 2** An adversary imitates KGC to distribute group key. It first picks an  $m$ -th polynomial  $f(x)$ . After the initiator sends a group key initialization message, it broadcasts a response message  $R_\sigma = \{\sigma, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ . Then after users send request messages, it sends distribution message  $\mathcal{K}_i$  to  $U_i$ .

**Theorem 6.3** Any adversary cannot distribute a fake group key to the users by impersonation attack 2.

**Proof** Although the adversary can also select  $f(x)$  and compute distribution information  $d_i$  like KGC does in off-line phase, it still cannot generate the function  $g_i(x)$  to compute accurate values of  $g_i(x'_i)$  and  $g_i^{-1}(y'_i)$  in  $\mathcal{K}_i$  because it does not know  $(x_i, y_i)$  of  $U_i$ . If the adversary uses false values  $fg_i(x'_i)$  and  $fg_i^{-1}(y'_i)$  to construct  $\mathcal{K}_i = \{U_i, fg_i(x'_i), fg_i^{-1}(y'_i), d_i, h(k, \sigma)\}$ , users will compute a different group key from the constant term in  $f(x)$  picked by the adversary. Then, the user will be aware of receiving a fake group key when it uses the authentication information in  $\mathcal{K}_i$  to check its calculated key. In brief, an adversary is completely powerless to distribute a fake group key by impersonation attack 2. □

### 6.3 Resistance to reply attack

**Model of reply attack 1** Suppose that an Outsider obtains an outdated group key  $ok$  and a request message  $oM_{\sigma,i} = \{o\sigma, U_i, (ox_i^*, oy_i^*)\}$ . When the group requests a new key, the Outsider uses new group communication identifier  $\sigma$  to revise the outdated request message as  $M_{\sigma,i} = \{\sigma, U_i, (ox_i^*, oy_i^*)\}$ . Then, it sends  $M_{\sigma,i}$  to KGC.

**Theorem 6.4** An Outsider cannot obtain an outdated group key by reply attack 1.

**Proof** As described in Sect. 4, each group key is selected by KGC during the off-line phase. In other words, the process of key generation is completely independent of users' request messages. Even if all users utilize old coordinates  $(ox_i^*, oy_i^*)$  to send request messages, they still obtains a new group key  $k$  foreign to previous key  $ok$ . Therefore, an Outsider cannot obtain any information about the new group key by reply attack 1. □

**Model of reply attack 2** Suppose that an adversary obtains an outdated group key  $ok$ . Define an outdated request message as  $oM_{\sigma,i} = \{o\sigma, U_i, (ox_i^*, oy_i^*)\}$  and outdated distribution message as  $o\mathcal{K}_i = \{U_i, og_i(ox'_i), og_i^{-1}(oy'_i), od_i, h(ok, o\sigma)\}$ . When users utilize new coordinate  $(x_i^*, y_i^*)$  to request a new group key, the adversary resends the outdated response and distribution messages to them.

**Theorem 6.5** An adversary cannot distribute an outdated group key to users by reply attack 2.

**Proof** In order to let legal user recover group key, KGC is required to construct  $g(x)$  to transport  $(x'_i, y'_i)$ , where  $g(x)$  is generated by private coordinate  $(x_i, y_i)$  and public coordinate  $(x_i^*, y_i^*)$ . The private coordinate  $(x_i, y_i)$  kept by  $U_i$  is fixed while  $(x_i^*, y_i^*)$  is always changed for different request messages  $M_{\sigma,i}$ . As described attack in *Model of reply attack 2*, if  $(x_i, y_i)$ ,  $(x_i^*, y_i^*)$  and  $(ox_i^*, oy_i^*)$  are not in a straight line, the outdated function  $og(x)$  is different from new function  $g(x)$ . As a result, the user  $U_i$  is not able use  $g(x)$  to recover the previous coordinate  $(ox'_i, oy'_i)$  from  $\mathcal{K}_i = \{U_i, og_i(ox'_i), og_i^{-1}(oy'_i), od_i, h(ok, \sigma)\}$ , so that it cannot obtain the outdated group key. In brief, an adversary cannot distribute an outdated group key to users by reply attack 2. □

**Corollary 6.1** *If a legal user drops out of the group, it will not get any information about the new group key.*

**Proof** If a user drops out of the group but still wants to obtain new group keys, the KGC will not compute and send its corresponding distribution messages. So, it is deemed as an Outsider. In virtue of Theorems 6.1–6.5, an Outsider can not obtain any information of group keys distributed by KGC. Therefore, the corollary holds.  $\square$

## 7 Properties

In this section, we compare our proposed protocol with Harn–Lin’s protocol [12] and Liu et al.’s protocol [20], because all the three protocols are group key distribution (transfer) protocols based on Shamir’s  $(t, n)$  threshold SS. The nine aspects are considered in comparisons: (1) hard problems, (2) security, (3) on-line/off-line mechanism, (4) computation complexity for KGC, (5) computation complexity for a user, (6) communication overhead, (7) storage complexity, (8) response speed of the group key request and (9) calculation speed of group key recovery and authentication.

### 7.1 Hard problem

In all the three protocols, each user is just required to register at the KGC only once, and then it should keep the private coordinate  $(x_i, y_i)$  as a long-term secret. For keeping the private coordinate unknown to the others, both Harn–Lin’s and Liu et al.’s protocols are based on integer factorization problem. But in our protocol, the KGC distributes sums of *Lagrange components* protected by hash values instead of coordinates on the generation function  $f(x)$ , so that no hard problems are needed in our protocol.

### 7.2 Security

On account of cask principles, the security of a protocol depends on the weakest point. Because Shamir’s SS is unconditional secure, the security of Harn–Lin’s and Liu et al.’s protocols is based on large integer factorization problem, while our protocol depends on one-way hash function.

All the three protocols are resistant to passive attack and impersonation attack. Liu et al.’s protocol and our protocol can also resist reply attack. However, in the light of Nam et al.’s paper [23], Harn–Lin’s protocol is vulnerable to reply attack. For detailed attack method, reader can refer to the original paper.

### 7.3 On-line/off-line mechanism

In both Harn–Lin’s and Liu et al.’s protocols, the KGC has to wait users to send group key request messages. Only when KGC receives all messages, it starts to pick group key  $k$ , construct generation function  $f(x)$ , select coordinates on  $f(x)$  and compute authentication value. In other words, KGC cannot do anything before it receives request messages from users. Instead, the KGC in our protocol is able to carry out most computations in off-line phase (before users send request messages). As a result, KGC has a very short response time in on-line phase (after users send request messages).

**Table 2** Comparisons table

Protocol	Harn–Lin’s protocol	Liu et al.’s protocol	Our protocol
Hash function	1	2	1
Hard problem	Yes	Yes	No
Resistance to passive attack	Yes	Yes	Yes
Resistance to impersonation attack	Yes	Yes	Yes
Resistance to reply attack	No	Yes	Yes
On-line/off-line	No	No	Yes

The above comparisons are shown in the Table 2.

### 7.4 Computation complexity for KGC

In Harn–Lin’s protocol, KGC performs about  $m$  exclusive - OR operations,  $(m - 1)(m + 1)$  additive operations,  $m^2 + (m - 1)^2$  multiplicative operations and 1 hash operation to generate group key distribution messages, where  $m$  is the total number of legal users. Liu et al.’s protocol needs  $m$  more hash operations than Harn–Lin’s protocol. Hence, the computational complexity is  $O(m^2)$  of the two protocols. In our proposed protocol, the KGC performs  $m(m + 3)$  additive operations,  $m(m + 1)$  multiplicative operations and  $m + 1$  hash operations in off-line phase. Thus, the computational complexity is also  $O(m^2)$ . However, the computation is over  $\mathbb{Z}_p$  in our protocol while it is over  $\mathbb{Z}_n$  in the other protocols, where  $n$  is much greater than  $p$ . Moreover, because KGC carries out most computation in off-line phase, it just needs  $2m + 3$  additive operations,  $2m + 2$  multiplicative operations and 1 hash operation in on-line and the computational complexity is just  $O(m)$ .

### 7.5 Computation complexity for a user

After a user receives group key distribution message from the KGC, it starts to recover and verify the group key. In Harn–Lin’s protocol, a user needs 1 exclusive - OR operation, about  $m^2$  additive operations,  $m^2$  multiplicative operations, and 1 hash operation. Liu et al.’s protocol needs 1 more hash operation than Harn–Lin’s protocol. Hence, the computational complexity are  $O(m^2)$  for the two protocols. In our protocol, a user just needs  $m + 10$  additive operations,  $m + 10$  multiplicative operations and 2 hash operations. Hence, the computational complexity is  $O(m)$ .

### 7.6 Communication overhead

In both Harn–Lin and Liu et al. protocols, users send to KGC about  $2m$  numbers and KGC transports  $3m$  numbers to users. Thus, the communication overhead is  $5m$  numbers and each number is in  $\mathbb{Z}_n$ . In our protocol, users send to KGC about  $3m$  numbers and KGC transports  $6m$  numbers to users. Hence, the communication overhead is  $9m$  numbers and each number is in  $\mathbb{Z}_p$ . However, KGC in our protocol has to sends each group key distribution message to the corresponding user one by one. KGC in the other protocols can broadcast distribution messages to all users.

## 7.7 Storage complexity

In both Harn–Lin and Liu et al. protocols, KGC just needs to store  $m$  private coordinates of users. In other words, it keeps  $2m$  numbers which are in  $\mathbb{Z}_n$ . But in our protocol, KGC has to keep  $6m$  numbers including  $m$  private coordinates  $(x_i, y_i)$ ,  $m$  ephemeral coordinates  $(x'_i, y'_i)$ ,  $m$  abscissas  $x_i$  and  $m$  protected group key distribution values  $d_i$ . Moreover, each user need 2 numbers to store its private coordinate in all the three protocols.

## 7.8 Response speed of group key request

In Harn–Lin’s protocol, before KGC receives users’ request messages, it cannot carry out any computation. After that, the computation complexity for KGC is  $O(m^2)$ . Liu et al.’s protocol needs  $m$  more hash operations than Harn–Lin’s protocol. Hence, the computational complexity is also  $O(m^2)$ . But in our proposed protocol, KGC carries out most computation in off-line phase and the computational complexity for KGC is just  $O(m)$  in off-line phase. Therefore, the response speed of group key request is faster than it in the other protocols.

## 7.9 Speed of group key recovery and authentication by users

After a user receives group key distribution message from the KGC, it starts to recover and verify the group key. In both Harn–Lin’s and Liu et al.’s protocols, the computational complexity for a user is  $O(m^2)$  while it is just  $O(m)$ . Therefore, the speed of group key recovery and authentication by users in our protocol is also faster than it in the other protocols.

## 8 Experiment

In this section, we verify the theories in 7.5 and 7.6 through experiments. Because the computational complexity of Liu et al’s protocol approximately equals Harn–Lin’s, we just compare our protocol with Harn–Lin’s. We chose a RSA-number with 1024 bits as the modulus in Harn–Lin’s protocol, where  $n = 135066410865995223349603216278805969938881475605667027524485143851526510604859533833940287150571909441798207282164471551373680419703964191743046496589274256239341020864383202110372958725762358509643110564073501508187510676594629205563685529475213500852879416377328533906109750544334999811150056977236890927563$ . And SHA-256 is used in both protocols as the hash function. Because the hash value of SHA-256 is 256 bits, we choose a 260 bits number as the modulus for our protocol, where  $p = 15476004057485807498789328096422564586042495822915855764974651495000499083147583$ .

From the perspective of cryptography, the computational complexity of factoring  $n$  is about  $e^{\sqrt{\ln(n)\ln(\ln n)}} \approx e^{70}$ . Although SHA-1 and MD5 have been proved insecurity by Wang et al. [32,33], there is still no more effective method than birthday attack to break SHA-256 currently. The computational complexity of breaking SHA-256 by birthday attack is  $2^{128}$ , which is greater than  $e^{70}$ . Therefore, our protocol is more secure than the other in this experiment.

**Remark 8.1** In our protocol, the hash value should be in  $\mathbb{Z}_p$  for security reason. Otherwise, it increases the possibility of hash function collision when different hash values modulo  $p$ . For



**Table 3** Simulation environment and conditions

CPU	Core i7-3630QM 2.40GHz
RAM	8G DDR3 1333MHz
Operation system	Windows 10 enterprise 64-bit
Programming language	Python
Programming software	Pycharm

**Table 4** Response time of group key request (s)

User number	Protocol	
	Harn–Lin’s protocol	Our protocol
50	4.571370	0.104552
100	20.673966	0.134091
150	52.657993	0.259405
200	103.841224	0.328321
250	211.968105	0.420336
300	419.852612	0.633199
350	634.937633	0.805593
400	935.913335	1.047877
450	1219.035164	1.196793
500	1593.049735	1.295323

**Table 5** Time of key recovery and authentication (s)

User number	Protocol	
	Harn–Lin’s protocol	Our protocol
50	3.947241	0.019082
100	18.507578	0.037025
150	40.831115	0.054113
200	67.566742	0.091200
250	110.791448	0.128122
300	202.860842	0.160992
350	229.557842	0.210194
400	312.193951	0.245661
450	483.446120	0.285706
500	616.605906	0.315709

efficiency reason, the smaller modulus  $p$  is, the faster KGC and users calculate. Therefore, the modulus  $p$  should be set as a number which is a bit greater than the hash value.

Table 3 shows the simulation environment and conditions of the experiment.

**Analysis 1** From Tables 4 and 5, it can be seen that, for the same user number, both response time of group key request and time of key recovery and authentication in Harn–Lin’s protocol are far longer than those in our protocol. There are two reasons:

1: The computational complexity of Harn–Lin’s protocol is  $O(m^2)$  while our protocol is  $O(m)$  in on-line phase .

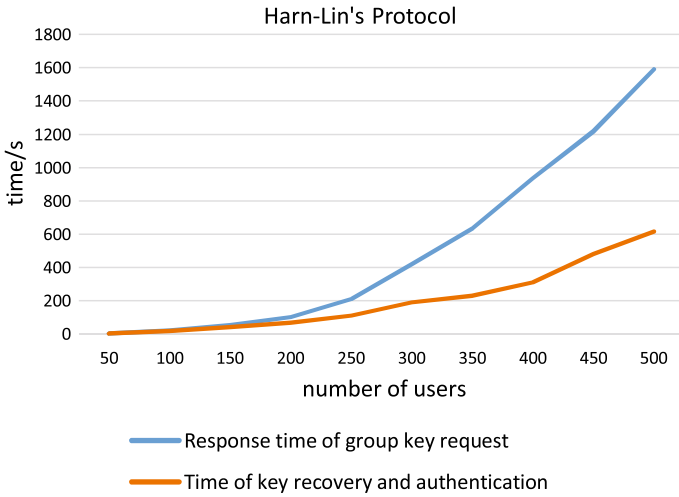


Fig. 3 Time of Harn-Lin's protocol

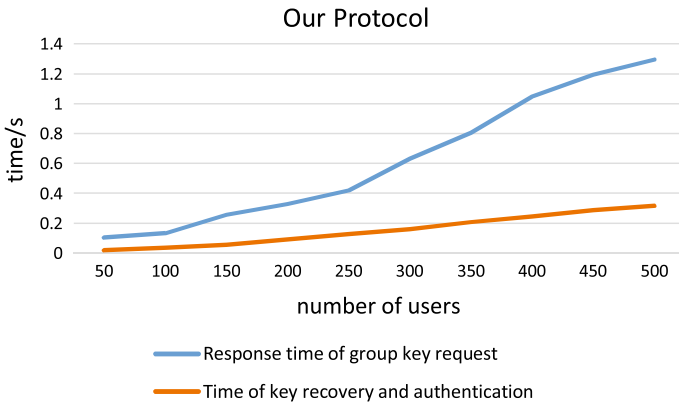


Fig. 4 Time of our protocol

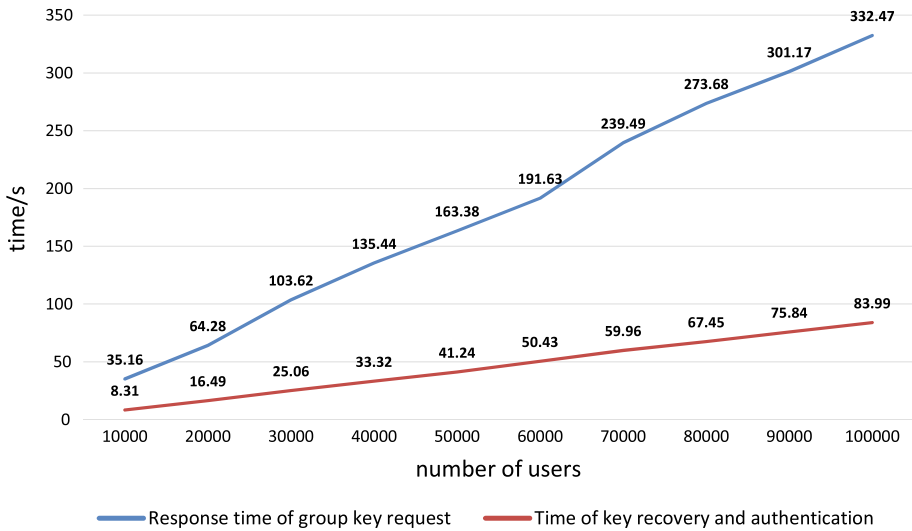
2: More importantly, the computation of Harn-Lin's protocol is over  $\mathbb{Z}_n$ , where  $n$  is 1024 bits. But it of our protocol is over  $\mathbb{Z}_p$ , where  $p$  is 260 bits.

**Analysis 2** Figures 3 and 4 show that, with the rise of user number, the speed of time increase in Harn-Lin's protocol is faster than our protocol. This means that the computational complexity of Harn-Lin's protocol is higher than our protocol.

Moreover, in order to show the high-efficiency of our protocol, we use massive users to collect more data about time of group key request and key recovery in our protocol. The results are shown in Fig. 5.

## 9 Conclusion

In this paper, we propose a secure on-line/off-line group key distribution protocol. In the proposed protocol, only users' registration messages need to be transported in privacy, while



**Fig. 5** Time of our protocol with massive users

all the other messages can be transported in public. In terms of safety, the protocol is resistant to passive attack, impersonation attack and reply attack. And it supports authentication function. More importantly, KGC can carry out most computations of group key generation in off-line phase. Consequently, the response speed of group key request and calculation speed of group key recovery and authentication can be greatly improved.

**Acknowledgements** This work is supported by National Natural Science Foundation of China under 61572454, 61572453, 61472382, 61520106007.

## References

1. Asmuth C., Bloom J.: A modular approach to key safeguarding. *IEEE Trans. Inf. Theory* **29**(2), 208–210 (1983).
2. Blakley G.R.: Safeguarding cryptographic keys. In: *Proceedings of the National Computer Conference*, vol. 48, pp. 313–317 (1979).
3. Boyd C.: On key agreement and conference key agreement. In: *Australasian Conference on Information Security and Privacy*, pp. 294–302. Springer, Berlin (1997).
4. Bresson E., Chevassut O., Pointcheval D.: Provably secure authenticated group Diffie–Hellman key exchange. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **10**(3), 10 (2007).
5. Chen L.Q., Sun C.F., Xu C.J.: An authenticated group key agreement scheme for wireless sensor networks based on bilinear pairings. *Adv. Mater. Res.* **846**, 876–882 (2014).
6. Diffie W., Hellman M.: New directions in cryptography. *IEEE Trans. Inf. Theory* **22**(6), 644–654 (1976).
7. Even S., Goldreich O., Micali S.: On-line/off-line digital signatures. In: *Conference on the Theory and Application of Cryptology*, pp. 263–275. Springer, New York (1989).
8. Fiat A., Naor M.: Broadcast encryption. In: *Annual International Cryptology Conference*, pp. 480–491. Springer, Berlin (1993).
9. Guo C., Chang C.C.: An authenticated group key distribution protocol based on the generalized Chinese remainder theorem. *Int. J. Commun. Syst.* **27**(1), 126–134 (2014).
10. Guo F., Mu Y., Chen Z.: Identity-based online/offline encryption. In: *International Conference on Financial Cryptography and Data Security*, pp. 247–261. Springer, Berlin (2008).
11. Harn L., Hsu C.F.: A practical hybrid group key establishment for secure group communications. *Comput. J.* **60**(11), 1582–1589 (2017).

12. Harn L., Lin C.: Authenticated group key transfer protocol based on secret sharing. *IEEE Trans. Comput.* **59**(6), 842–846 (2010).
13. Hohenberger S., Waters B.: Online/offline attribute-based encryption. In: *International Workshop on Public Key Cryptography*, pp. 293–310. Springer, Berlin (2014).
14. Hsu C., Zeng B., Zhang M.: A novel group key transfer for big data security. *Appl. Math. Comput.* **249**, 436–443 (2014).
15. Irshad A., Sher M., Chaudhary S.A.: An efficient and anonymous multi-server authenticated key agreement based on chaotic map without engaging Registration Centre. *J. Supercomput.* **72**(4), 1623–1644 (2016).
16. Kim Y., Perrig A., Tsudik G.: Tree-based group key agreement. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **7**(1), 60–96 (2004).
17. Kumar V., Kumar R., Pandey S.K.: A computationally efficient centralized group key distribution protocol for secure multicast communications based upon RSA public key cryptosystem. *J. King Saud Univ. Comput. Inf. Sci.* (2018). <https://doi.org/10.1016/j.jksuci.2017.12.014>.
18. Lai J., Mu Y., Guo F.: Efficient identity-based online/offline encryption and signcryption with short ciphertext. *Int. J. Inf. Secur.* **16**(3), 299–311 (2017).
19. Liu J.K., Baek J., Zhou J.: Efficient online/offline identity-based signature for wireless sensor network. *Int. J. Inf. Secur.* **9**(4), 287–296 (2010).
20. Liu Y., Cheng C., Cao J.: An improved authenticated group key transfer protocol based on secret sharing. *IEEE Trans. Comput.* **62**(11), 2335–2336 (2013).
21. Mayer A., Yung M.: Generalized secret sharing and group-key distribution using short keys. In: *Proceedings of the Compression and Complexity of Sequences 1997*, pp. 30–44. IEEE Computer Society, Los Alamitos (1997).
22. Mignotte M.: How to share a secret. In: *Workshop on Cryptography*. Springer, Berlin, pp. 371–375 (1982).
23. Nam J., Kim M., Paik J., et al.: Cryptanalysis of a group key transfer protocol based on secret sharing. In: *International Conference on Future Generation Information Technology*, pp. 309–315. Springer, Berlin (2011).
24. Naoui S., Elhdhili M.E., Saidane L.A.: Lightweight enhanced collaborative key management scheme for smart home application. In: *2017 International Conference on High Performance Computing & Simulation (HPCS)*, pp. 777–784. IEEE (2017).
25. Olimid R.F.: On the security of an authenticated group key transfer protocol based on secret sharing. In: *Information and Communication Technology-EurAsia Conference*, pp. 399–408. Springer, Berlin (2013).
26. Shamir A.: How to share a secret. *Commun. ACM* **22**(11), 612–613 (1979).
27. Shamir A., Tauman Y.: Improved online/offline signature schemes. In: *Annual International Cryptology Conference*, pp. 355–367. Springer, Berlin (2001).
28. Steiner M., Tsudik G., Waidner M.: Diffie–Hellman key distribution extended to group communication. In: *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, pp. 31–37. ACM, New York (1996).
29. Sun Y., Wen Q., Sun H.: An authenticated group key transfer protocol based on secret sharing. *Procedia Eng.* **29**, 403–408 (2012).
30. Tanaka H., Nakajima K., Ishigaki K., et al.: Hybrid pen-input character recognition system based on integration of online–offline recognition. In: *Proceedings of the Fifth International Conference on Document Analysis and Recognition, 1999 (ICDAR'99)*, pp. 209–212. IEEE (1999).
31. Ustaoglu B.: Obtaining a secure and efficient key agreement protocol from (H) MQV and NAXOS. *Des. Codes Cryptogr.* **46**(3), 329–342 (2008).
32. Wang X., Yu H.: How to break MD5 and other hash functions. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 19–35. Springer, Berlin (2005).
33. Wang X., Yin Y.L., Yu H.: Finding collisions in the full SHA-1. In: *Annual International Cryptology Conference*, pp. 17–36. Springer, Berlin (2005).
34. Xu S., Mu Y., Susilo W.: Online/Offline signatures and multisignatures for AODV and DSR routing security. In: *Australasian Conference on Information Security and Privacy*, pp. 99–110. Springer, Berlin (2006).
35. Yuan W., Hu L., Li H.: Security and improvement of an authenticated group key transfer protocol based on secret sharing. *Appl. Math. Inf. Sci.* **7**(5), 1943 (2013).
36. Zhang L., Wu Q., Domingo-Ferrer J.: Round-efficient and sender-unrestricted dynamic group key agreement protocol for secure group communications. *IEEE Trans. Inf. Forensics Secur.* **10**(11), 2352–2364 (2015).