Wireless Netw
DOI 10.1007/s11276-016-1223-1

Computation-efficient key establishment in wireless group communications

Ching-Fang Hsu[1] · Lein Harn[2] · Yi Mu[3] · Maoyuan Zhang[1] · Xuan Zhu[1]

Abstract Efficient key establishment is an important problem for secure group communications. The communication and storage complexity of group key establishment problem has been studied extensively. In this paper, we propose a new group key establishment protocol whose computation complexity is significantly reduced. Instead of using classic secret sharing, the protocol only employs a linear secret sharing scheme, using Vandermonde Matrix, to distribute group key efficiently. This protocol drastically reduces the computation load of each group member and maintains at least the same security degree compared to existing schemes employing traditional secret sharing. The security strength of this scheme is evaluated in detail. Such a protocol is desirable for many wireless applications where portable devices or sensors need to reduce their computation as much as possible due to battery power limitations. This protocol provides much lower computation complexity while maintaining low and balanced communication complexity and storage complexity for secure group key establishment.

Keywords Wireless group key transfer · Vandermonde matrix · Linear secret sharing · Computation-efficient

✉ Ching-Fang Hsu
cherryjingfang@gmail.com

[1] Computer School, Central China Normal University, Wuhan, China

[2] Department of Computer Science Electrical Engineering, University of Missouri-Kansas City, Kansas City, MO, USA

[3] School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia

1 Introduction

In many applications related to wireless networks and distributed computing, group communications [28–30] is an efficient means of a many-to-many communication style in a *group*, this goes beyond both one-to-one communication (i.e., unicast) and one-to-many communication (i.e., multicast). The privacy of a group communication session is usually ensured using (symmetric) encryption. All the members in a group share a session (group) key. However, the group membership changes dynamically. Thus, group keys shall change dynamically to ensure both forward secrecy and backward secrecy of group sessions. The forward secrecy is maintained if an old member who has been excluded from the current and future group sessions cannot access the communication of the current and future group sessions, and the backward secrecy is guaranteed if a new member of the current group session cannot recover the communication data of past sessions. Each group session thus needs a new group key that is only known to the current group members. At the same time, in group key transfer, outsider attacker can try to recover the secret group key that the outsider of a particular group is unauthorized to know, and insider attacker who is authorized to know the secret group key can attempt to obtain other group member's secret that is used to recover the group key. Each group key transfer thus needs to resist the outsider and the insider attackers.

In this paper, we study how a group key can efficiently be transmitted in computation. We adopt a common model where group keys are issued and transmitted by a key generation center (KGC), as it has much less communication complexity, as compared to distributed key exchange protocols, which is a very desired property in most wireless applications [1–6]. The resources needed for

Published online: 12 February 2016

the KGC to transmit group keys to group members include communication, storage, and computation resources. The communication complexity is usually measured by the number of data bits that need to be transmitted from the KGC to group members to convey information of group keys, whereas the storage complexity is measured by the number of data bits that the KGC and group members need to store to obtain group keys. Another similarly important but usually undernoticed, if not ignored, factor is the computation complexity, which can be measured by the number of computation operations (or the computation time on a given computing platform) that the KGC and group members need to distribute and recover group keys. Hereafter, the problem of how resources can effectively be used to distribute group keys is referred to as the group key transfer problem.

The group key transfer problem has been studied extensively in the larger context of key management for secure group communications [7, 8], mainly on balancing the storage complexity and the communication complexity. Research efforts have been made to achieve low communication and storage complexity for group key transfer. In addition to using (symmetric) encryption, static secret sharing via broadcast channel was studied in [9, 10]. However, this threshold-based scheme can only distribute a group key to a designated group of members for one-time use. Thus, the scheme does not provide forward or backward secrecy. A secure lock method based on the Chinese remainder theorem was proposed in [11]. However, its prohibitively high communication complexity and computation complexity make it only practical for a very small group with limited number of members. Various theoretical measures and schemes for group key distribution were introduced in [12]. Along the same line, many research efforts have been made on balancing communication complexity and storage complexity of the group key distribution problems, for example, [1, 2, 13–16].

Although most research on group key transfer has been on balancing communication complexity and storage complexity, very few efforts have been made to reduce computation complexity [17]. It has been long assumed that expensive encryption and decryption operations are necessary to distribute group keys. When group communication is becoming increasingly practical over general Internet, it especially gains most on true broadcast communication media such as wireless networks. In such wireless systems, group members are often of various lightweight mobile devices or sensors. Since it is becoming increasingly affordable to embed considerable computing power (and storage capacity) into these devices, their battery power will remain to be limited for a long time ahead. Computation complexity is thus more important than storage complexity for these devices in many applications. Hence, it becomes at least equally, if not more important, to reduce the computation complexity of the group key transfer problem, which has been understudied so far.

Secret sharing was first introduced by Blakley [18] and Shamir [10] in 1979. Since avoiding the use of encryption one by one can introduces less computation complexity and it maintains at least the same security degree of using encryption algorithms, secret sharing has been used to design group key transfer protocols. Laih et al. [19] 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 [20–22] following the same concept to propose ways to distribute group messages to multiple users. Recently, [23] proposed a novel group key transfer protocol using (t, n) secret sharing that provided confidentiality and authentication, where KGC and each group member need to compute a t-degree interpolating polynomial to distribute and recover the secret group key.

In this paper, we propose a new group key transfer scheme that drastically reduces computation complexity and yet maintains at least the same security degree of using classic threshold secret sharing without increasing communication or storage complexity. In our scheme, information related to group keys is hidden using Vandermonde Matrix rather than interpolating polynomial. In general, linear secret sharing scheme (LSSS) based on Vandermonde Matrix has much lower computation complexity than classic threshold secret sharing, which has been verified by our comparisons described later in Sect. 5. Thus, the computation complexity of group key transfer can be significantly reduced. The similar idea of using LSSS based on Vandermonde Matrix to achieve privacy was employed in [24, 28]. The major difference between these two schemes and ours is that our scheme allows lower computation complexity only by Vandermonde Matrix and one-way hash function, whereas other two schemes have more extra computation complexity by adding ElGamal encryption algorithm or CDH assumption. The security strength of our scheme will be evaluated in detail, as well as its communication, storage, and computation complexity. Aside from its low computation complexity, this scheme also has low storage complexity, i.e., $O(1)$ for an individual group member and $O(t)$ for the KGC, where t is the number of group members. Comparisons are conducted to show great reduction of our scheme in computation complexity than using traditional threshold secret sharing. Such a protocol is desirable for many wireless applications where portable devices or sensors need to reduce their computation as much as possible due to battery power limitations. This protocol provides much lower computation complexity while maintaining low and balanced communication complexity and storage complexity for secure group key establishment.

The rest of this paper is organized as follows: In the next section, we provide some preliminaries. In Sect. 3, we propose our group key transfer protocol. We analyze the security of our proposed protocol in Sect. 4. Performance evaluation of the proposed scheme is discussed in Sect. 5. We conclude in Sect. 6.

2 Preliminaries

In this section we introduce some fundamental backgrounds.

In a secret sharing scheme, a secret s is divided into n *shares* and shared among a set of n shareholders by a mutually trusted *dealer* in such a way that authorized subsets of shareholders can reconstruct the secret but unauthorized subsets of shareholders cannot determine the secret. If any unauthorized subset of shareholders can not obtain any information about the secret, then the scheme is called perfect. The set of authorized subsets of shareholders is called access structure and the set of unauthorized subsets of shareholders is called prohibited structure.

Formally, let $\mathcal{P} = \{1, \ldots, n\}$ be a set of shareholders, by using Shannon's entropy function we say that a scheme is a *secret sharing scheme* with respect to access structure Γ, if the following holds [25].

1. *Correctness* Any subset of shareholders supposed to get s can compute s. Namely, for any $A \in \Gamma$, it holds $H(S|A) = 0$.
2. *Security* Any subset of shareholders not supposed to get s can not reconstruct s, even if they pool all their shares together. Namely, for any $A \notin \Gamma$, it holds $0 < H(S|A) \leq H(S)$. In the case that $H(S|A) = H(S)$, shareholders in A pool their shares together obtain no information on s. This is the case we are interested in, and we say that this scheme is *perfect*.

For a perfect secret sharing scheme, we say that it is *ideal*, if the shares of shareholders are taken from the same domain as the secret (as proved in [3], this is the minimal size of the shares); we say that it is *linear*, if the reconstruction operations are linear [26].

Monotone span programs (MSP) is introduced as linear models computing monotone Boolean functions by Karchmer and Wigderson [27]. We denote an MSP by $\mathcal{M}(\mathcal{K}, M, \psi)$, where M is a $d \times l$ matrix over a finite field $\mathcal{K}$ and ψ:{1, …, d} → {1, …, n} is a surjective labeling map which actually distributes to each participant some rows of M. We call d the size of the MSP. For any subset $A \subseteq \mathcal{P}$, there is a corresponding characteristic vector $\vec{\delta}_A = (\delta_1, \ldots, \delta_n) \in \{0,1\}^n$, where $\delta_i = 1$ if and only if $i \in A$. Consider a monotone Boolean function f:{0, 1}^{n} → {0, 1} such that for any $A \subseteq \mathcal{P}$ and any $B \subseteq A, f(\vec{\delta}_B) = 1$ implies $f(\vec{\delta}_A) = 1$. We say that an MSP $\mathcal{M}(\mathcal{K}, M, \psi)$ computes the monotone Boolean function f with respect to a target vector $\vec{v} \in \mathcal{K}^l \backslash \{(0, \ldots 0)\}$, if it holds that $\vec{v} \in span\{M_A\}$ if and only if $f(\vec{\delta}_A) = 1$, where M_A consists of the rows r of M with $\psi(r) \in A$ and $\vec{v} \in span\{M_A\}$ means that there exists a vector $\vec{w}$ such that $\vec{v} = \vec{w} M_A$.

Beimel [26] proved that devising a linear secret sharing scheme (LSSS) for an access structure Γ is equivalent to constructing an MSP computing the monotone Boolean function f_Γ such that $f_\Gamma(\vec{\delta}_A) = 1$ if and only if $A \in \Gamma$. It is known that an MSP $\mathcal{M}(\mathcal{K}, M, \psi)$ can compute f_Γ if and only if there exists a vector $\vec{v}$ which lies in the space $\bigcap_{A \in \Gamma_{\min}} \sum_{i \in A} V_i - \bigcup_{B \in \mathcal{A}_{\max}} \sum_{i \in B} V_i$, where V_i is the space spanned by the row vectors of M distributed to participant i according to ψ, and the vector $\vec{v}$ can be seen as the target vector described above. Hence, finding the linear spaces V_i such that $\bigcap_{A \in \Gamma_{\min}} \sum_{i \in A} V_i - \bigcup_{B \in \mathcal{A}_{\max}} \sum_{i \in B} V_i \neq \emptyset$ is a key step to build an LSSS with respect to Γ.

LSSS based on Vandermonde Matrix is introduced by Hsu et al. in [25]. Suppose that $\bar{V} = \mathcal{K}^{n+1}$ is the $(n + 1)$ dimensional linear space over a finite field $\mathcal{K}$. The characteristic $char(\mathcal{K}) = p$ and p is a safe large prime. Given a basis $\{e_1, \ldots, e_{n+1}\}$ of $\bar{V}$ with $\vec{e}_i = (0, \ldots, 0, \overset{i}{1}, 0, \ldots, 0) \in \mathcal{K}^{n+1}$ for $1 \leq i \leq n + 1$, the mapping $\mathbf{v} : \mathcal{K} \ominus \rightarrow \bar{V}$ defined by $\mathbf{v}(x) = \sum_{i=1}^{n+1} x^{i-1} e_i = (1, x, x^2, \ldots, x^n)$ is determined. For $x_i \in \mathcal{K} (i \in \{1, \ldots, n + 1\})$, the Vandermonde Matrix V_{n+1} can be represented as follows

$$V_{n+1} = \begin{pmatrix} \mathbf{v}(x_1) \\ \mathbf{v}(x_2) \\ \cdots \\ \mathbf{v}(x_{n+1}) \end{pmatrix} = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_1^n \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{n+1} & x_{n+1}^2 & \cdots & x_{n+1}^n \end{pmatrix}$$

In LSSS based on Vandermonde Matrix, there are $(n + 1)$ shareholders $P = \{P_0, P_1, \ldots, P_n\}$ and a mutually trusted dealer D, and the scheme consists of two algorithms:

1. *Share generation algorithm* the dealer D first picks a Vandermonde Matrix V_{n+1} and a random vector $\mathbf{r} = (r_0, r_1, r_2, \ldots, r_n) \in \bar{V}$ and let $\mathbf{r}$ be public, in which the secret $S = s_0 + s_1 + \cdots + s_n$ and all computations are performed in the finite field $\mathcal{K}$, and D computes:

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_1^n \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{n+1} & x_{n+1}^2 & \cdots & x_{n+1}^n \end{pmatrix} \begin{pmatrix} r_0 \\ r_1 \\ \cdots \\ r_n \end{pmatrix} = \begin{pmatrix} s_0 \\ s_1 \\ \cdots \\ s_n \end{pmatrix}$$

Then, the algorithm outputs a list of $(n + 1)$ shares $(x_0, x_1, \ldots, x_n)$ and distributes each share x_i to corresponding shareholder P_i secretly.

2. *Secret reconstruction algorithm* this algorithm takes all $(n + 1)$ shares $(x_0, x_1, \ldots, x_n)$ and the public vector $\mathbf{r}$ as inputs, and outputs the secret $S = s_0 + s_1 + \cdots + s_n$ by computing each inner product $(\mathbf{v}(x_i), \mathbf{r}) = s_i$.

We note that because every set of at most $(t + 1)$ vectors of the form $\mathbf{v}(x)$ is linearly independent, the above scheme satisfies the basic requirements of secret sharing scheme as follows: (1) With knowledge of all $(n + 1)$ shares, it can reconstruct the secret S easily; (2) With knowledge of fewer than $(n + 1)$ shares, it cannot get *any* information about the secret S. LSSS based on Vandermonde Matrix 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 [25].

3 The proposed protocol

We assume that there are t members in a group $\{1, \ldots, t\}$. In order to achieve secure group communication, the group's session keys are needed to be securely distributed among all group members prior to exchanging communication messages. Typically, the deputy of KGC is to select fresh session keys and securely distribute them to group members, in a way that only group members can derive the session key upon receiving the broadcasted messages. First of all, each member is required to register at KGC. Then, KGC makes records of all registered members, and removes any unsubscribed members.

Our group key transfer protocol consists of pre-distributing phase and group key distributing phase. The detailed description is as follows:

3.1 Pre-distributing phase

KGC selects a finite field $\mathcal{K}$ with the characteristic $char(\mathcal{K}) = p$, where p is a safe large prime, constructs two secure one-way hash functions $h_1(\cdot)$ and $h_2(\cdot)$ whose codomains are both $\mathcal{K}$. Then, each user i registers at KGC and shares his long-term secret $x_i \in \mathcal{K}$ with KGC in a secure manner. KGC publishes p, $h_1(\cdot)$ and $h_2(\cdot)$.

3.2 Group key distributing phase

Upon receiving a group key generation request from any user, KGC needs to randomly select a group key. KGC will distribute this group key to all group members in a secure and authenticated manner. *All communications in distributing phase are in an open broadcast channel.* For example, we assume that a group consists of t members, $\{1, \ldots, t\}$, and the shared secrets are $x_1, x_2, \ldots, x_t$. 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 $\{1, \ldots, t\}$.
- *Step* 2 KGC broadcasts the list of all group members, $\{1, \ldots, t\}$, as a response.
- *Step* 3 Each participating group member needs to send a random challenge, $r_i \in \mathcal{K}$, to KGC.
- *Step* 4 KGC randomly selects a group key $S \in \mathcal{K}$ $(S \neq \sum_{1 \le i \le t} s_i)$ and a random value $r_0 \in \mathcal{K}$. KGC also computes t additional values, u_i for $i = 1, \ldots, t$, and the value of $Auth = h_2(S, 1, \ldots, t, r_0, r_1, \ldots, r_t, u_1, \ldots, u_t)$, in which the vector $\mathbf{r} = (r_0, r_i, \ldots, r_t) \in \mathcal{K}^{t+1}$ and the inner product $(\mathbf{v}(x_i \oplus h_1(x_i, r_i, r_0)), \mathbf{r}) = s_i$ and $u_i = (S - s_i) \bmod p$. We define that $x \oplus y$ denotes $x + y \,(\mathrm{mod} p)$. KGC broadcasts $\{Auth, r_0, u_i\}$, for $i = 1, \ldots, t$, to all group members. All computations are performed in Z_p^*.
- *Step* 5 For each group member, $i\,(i \in \{1, \ldots, t\})$, knowing the public value, u_i, is able to compute the inner product $(\mathbf{v}(x_i \oplus h_1(x_i, r_i, r_0)), \mathbf{r}) = s_i$ and recover the group key $S = (u_i + s_i) \bmod p$. Then, i computes $h_2(S, 1, \ldots, t, r_0, r_1, \ldots, r_t, u_1, \ldots, u_t)$ and checks whether this hash value is identical to $Auth$. If these two values are identical, i authenticates the group key is sent from KGC.

Remark 1 The conventional group key transfer protocols based on threshold secret sharing schemes need to compute a t-degree interpolating polynomial to distribute and recover group keys. Actually, this approach can cause an increase of computational complexity. To overcome these drawbacks, in this paper, we construct a group key transfer protocol by a linear secret sharing based on Vandermonde Matrix, which is no need to compute a t-degree interpolating polynomial and only require that each member compute an inner product of two vectors to recover the group key.

4 Security analysis

In this section, we analyze that the proposed protocol has the following security advantages:

1. This protocol achieves security feathers with key freshness, key confidentiality and key authentication.
2. This protocol can resist the attacks in both synchronous and asynchronous networks.
3. Both the backward secrecy and the forward secrecy of group communication are maintained. i.e., the newly joined members cannot recover the communications of

the old sessions, and those old members who left the group cannot access the current session.

4. Both the outside attacker and the inside attacker are prevented. i.e., the outside attacker cannot recover the group key, and the inside attacker can recover the secret group key but cannot recover other member's secret shared with KGC.

Theorem 1 *The proposed protocol achieves the security feathers with key freshness, key confidentiality and key authentication.*

Proof *Key freshness* is ensured by KGC since a random group key is selected by KGC for each service request. In addition, the equation $S = (u_i + s_i) \bmod p$ used to recover the group key, where $s_i = (\mathbf{v}(x_i \oplus h_1(x_i, r_i, r_0)), \mathbf{r})$ is a function of random challenge r_i selected by each group member and random value r_0 selected by KGC.

Key confidentiality is provided due to the security features of LSSS based on Vandermonde Matrix. KGC randomly selects a group key S and makes t values, $u_i = (S - s_i) \bmod p$ for $i = 1, \ldots, t$, publicly known. For each authorized group member, including the secret x_i shared with KGC, he/she knows the inner product $(\mathbf{v}(x_i \oplus h_1(x_i, r_i, r_0)), \mathbf{r}) = s_i$. Thus, any authorized group member is able to recover the secret group key $S = (u_i + s_i) \bmod p$. However, for any unauthorized member (or outsider), there are only t values $u_i = (S - s_i) \bmod p$ for $i = 1, \ldots, t$ available and he obtains no information on s_i, $\sum_{1 \le i \le t} s_i$ and S since $S \neq \sum_{1 \le i \le t} s_i$. Thus, unauthorized member knows nothing about the group key. This property is unconditionally secure since there has no other computational assumption based upon.

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 each member's long-term secret x_i. In addition, any replay of $\{Auth, r_0, u_i\}$, for $i = 1, \ldots, t$, of KGC in step 4 can be detected since the group key is a function of random challenge r_i selected by each group member and random value r_0 selected by KGC.

Theorem 2 *The proposed protocol can resist the attacks in both synchronous and asynchronous networks.*

Proof In our proposed solution, each user needs not to release a value based on his long-term secret x_i. Group key reconstruction is based on all released values from KGC. There are only t values $u_i = (S - s_i) \bmod p$ for $i = 1, \ldots, t$ and *Auth*, r_0 available. Even if values are released asynchronously, attackers can not obtain any "good" information last after knowing all released values from KGC, which will be analyzed in Theorems 4 and 5.

Theorem 3 *The proposed protocol achieves the backward secrecy and the forward secrecy. i.e., the newly joined members cannot recover the old group keys, and those old members who left the group cannot access the current group key.*

Proof For every session, whenever some new members join or some old members leave a group, the KGC needs to distribute a new group key to all the current group members. In each session, the group key is a function of each current group member's long-term secret x_i and the fresh random vector $\mathbf{r} = (r_0, r_i, \ldots, r_t)$ determined both by each current group member and KGC. Namely, the newly joined members can recover the current group key but cannot recover the previous group keys, and those old members who have left the group cannot recover the current group key. Hence, our protocol achieves both the backward secrecy and the forward secrecy of group communication.

In group key transfer, adversaries can be categorized into two types. The first type of adversaries is 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 member 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 is 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.

Theorem 4 (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 unconditionally secure, which is ensured by the LSSS based on Vandermonde Matrix.

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 random vector $\mathbf{r} = (r_0, r_i, \ldots, r_t)$ determined both by each member and KGC. Namely, a compromised group key cannot be reused since the random vector $\mathbf{r} = (r_0, r_i, \ldots, r_t)$ is different in each session.

Theorem 5 (insider attack) *Assume that the protocol runs successfully many times, then the secret $x_i \in \mathcal{K}$ of each group member shared with KGC remains unknown to all other group members* (*and outsiders*).

Proof For a group key service request, KGC randomly selects a group key S and makes t values, $u_i = (S - s_i) \bmod p$ for $i = 1, \ldots, t$, publicly known. For each authorized group member, with knowledge of the secret shared with KGC and t public information, he/she knows u_i and is able to compute the inner product $(\mathbf{v}(x_i \oplus h_1(x_i, r_i, r_0)), \mathbf{r}) = s_i$. Thus, any authorized group member is able to reconstruct the group key $S = (u_i + s_i) \bmod p$, where the vector $\mathbf{r} = (r_0, r_i, \ldots, r_t)$. However, the secret $x_i \in \mathcal{K}$ of each group member shared with KGC remains unknown to outsiders.

In our proposed protocol, group key service requests 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 a service request for a group containing the adversary and the target group member. The adversary also forges the challenge of the target group member for this service. The KGC generates the group key S. Then, the adversary is able to reconstruct the group key. At the same time, the adversary can obtain the inner product $(\mathbf{v}(x_{\text{target}} \oplus h_1(x_{\text{target}}, r_{\text{target}}, r_0)), \mathbf{r}) = s_{\text{target}}$ from $s_{\text{target}} = (S - u_{\text{target}}) \bmod p$. However, the adversary has no idea about $h_1(x_{\text{target}}, r_{\text{target}}, r_0)$ even if r_{target} has been intercepted, since x_{target} is only known by the target group member and KGC. Alternatively, the adversary has no chance of getting any valid information about x_{target} with the equation $(\mathbf{v}(x_{\text{target}} \oplus h_1(x_{\text{target}}, r_{\text{target}}, r_0)), \mathbf{r}) = s_{\text{target}}$. Therefore, the proposed protocol resists against insider attack, i.e., although an authorized member can derive the group key S, he can not obtain other member's long-term secret.

If the attacker tries to impersonate the target group member by reusing previously recorded information, i.e., the inner product $(\mathbf{v}(x_{\text{target}} \oplus h_1(x_{\text{target}}, r_{\text{target}}, r_0)), \mathbf{r}) = s_{\text{target}}$, this attack cannot succeed since s_{target} is a function of random values r_{target} and r_0. Even if the adversary can replay the challenge r_{target} of the target group member, he cannot reuse r_0 since r_0 randomly selected by KGC is different in each session.

5 Performance evaluation

In this section, we will firstly compare our scheme with public-key-based key distribution protocols. Then, we compare ours with a threshold secret-sharing-based key distribution protocol [23] proposed recently, in terms of storage requirement, communication and computational costs will be discussed in detail.

5.1 Comparison 1

Compared with the public-key-based key distribution protocols, our scheme has the following advantages:

(a) Instead of using public key encryptions in which the security is based on computation assumptions, we use the secret sharing as a tool of broadcast encryption in which the security is unconditionally secure. In addition, instead of doing encryption one at a time, our scheme can perform encryption all at once to reduce computational complexity.

(b) The public-key-based key distribution protocol requires larger rekeying overheads when any membership of user has changed since broadcasting keys need to be updated. But our scheme uses secret sharing and KGC can manage any membership change efficiently. There is no rekeying issue.

(c) Since computation in our scheme uses a smaller modulus (say 160 bits only) as compared with public-key computations using a larger modulus (say 1024 bits at least in RSA), computations in our scheme are much faster than public-key computations.

5.2 Comparison 2

Suppose that a set of users is $\{1, \ldots, n\}$ and the group is $\{1, \ldots, t\}$, where $n \geq t$ and $t \geq 2$. The performance evaluation is as follow.

5.2.1 Time complexities

Let TM, TI and TH be execution time for performing a modular multiplication, a modular inverse and a one-way hash function, respectively. As compared to TM or TI, the time for performing modular addition or subtraction required in the proposed scheme can be ignored.

The proposed protocol Given the shared secrets $x_1, x_2, \ldots, x_t$, a group key $S \in \mathcal{K}$ and the vector $\mathbf{r} = (r_0, r_i, \ldots, r_t) \in \mathcal{K}^{t+1}$, the time complexity for distributing the group key by KGC is $t \times 2t \times \text{TM} + 2\text{TH}$. The time complexity for recovering the group key by each group member is $2t \times \text{TM} + 2\text{TH}$, where computing the

Table 1 Computational comparison of the proposed scheme and Harn's scheme in each group key transfer

Scheme	Distributing the group key	Recovering the group key
The proposed scheme	$t \times 2t \times \mathrm{TM} + 2\mathrm{TH}$	$2t \times \mathrm{TM} + 2\mathrm{TH}$
Harn's scheme	$t \times (t+1) \times t \times (\mathrm{TM} + \mathrm{TI}) + \mathrm{TH}$	$(t+1) \times t \times (\mathrm{TM} + \mathrm{TI}) + \mathrm{TH}$

Table 2 Comparison of communication costs between the proposed scheme and Harn's scheme

Scheme	User registration	Group key generation and distribution	Total
The proposed scheme	$n\|p\|$	$t\|p\| + (t+1)\|p\| + \|H\|$	$(n+2t+1)\|p\| + \|H\|$
Harn's scheme	$2n\|pq\|$	$t\|pq\| + 2t\|pq\| + \|H\|$	$(2n+3t)\|pq\| + \|H\|$

inner product $(\mathbf{v}(x_i \oplus h_1(x_i, r_i, r_0)), \mathbf{r}) = s_i$ needs $2t \times \mathrm{TM} + \mathrm{TH}$.

Harn's protocol Given the shared secrets (x_i, y_i), a group key k and the random challenge R_i for $1 \le i \le t$, the time complexity for distributing the group key by KGC is $t \times (t+1) \times t \times (\mathrm{TM} + \mathrm{TI}) + \mathrm{TH}$. The time complexity for recovering the group key by each group member is $(t+1) \times t \times (\mathrm{TM} + \mathrm{TI}) + \mathrm{TH}$.

Computational comparison of the proposed protocol and Harn's protocol is shown in Table 1. From Table 1, as compared with Harn's protocol, the proposed protocol LSSS based on Vandermonde Matrix causes a significant decrease of the computational complexity.

5.2.2 Communication costs

The communication costs required in the proposed protocol are measured by the total volume of data transmission during pre-distributing phase and group key distributing phase, respectively. In our protocol, $|p|$ is the size of the adopted finite field Z_p^* and $|H|$ is the output size of one-way hash function. Obviously, the communication cost for user registration is $n|p|$. The communication cost for group key generation and distribution is $t|p| + (t+1)|p| + |H|$.

In Harn's protocol, for the same number n of users, the communication costs required in user registration is $2n|pq|$ and the communication cost for group key generation and distribution is $t|pq| + 2t|pq| + |H|$.

Comparison of communication costs between the proposed protocol and Harn's protocol is shown in Table 2. It can be seen that in the proposed protocol, the communication costs required can be significantly reduced.

5.2.3 Storage complexities

In the proposed protocol, each user i registers at KGC and shares his long-term secret $x_i \in \mathcal{K}$ with KGC in a secure manner. Storage complexities are $O(1)$ for an individual group member and $O(t)$ for the KGC, where t is the number of group members. In Harn's protocol, during registration KGC shares a secret (x_i, y_i) with each user i, where $x_i, y_i \in Z_{pq}^*$. As compared with Harn's protocol, the proposed protocol LSSS based on Vandermonde Matrix causes a half decrease of the storage complexity.

6 Conclusions

We have proposed a computation-efficient group key transfer protocol by LSSS based on Vandermonde Matrix. This protocol drastically reduces the computation load of each group member compared to existing schemes employing traditional secret sharing. Such a protocol is desirable for many wireless applications. The security strength of this scheme is evaluated in detail, as well as its communication, storage, and computation complexity.

Acknowledgments This work was supported by the self-determined research funds of CCNU from the colleges' basic research and operation of MOE, under Grant CCNU15ZD003 and CCNU15A02018, and the major Project of national social science fund, under Grant 12&2D223.

References

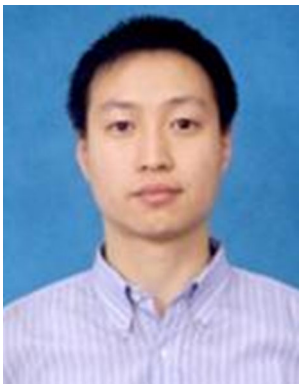
1. Stinson, D. R. (1997). On some methods for unconditionally secure key distribution and broadcast encryption. *Designs, Codes and Cryptography, 12*, 215–243.
2. Stinson, D. R., & van Trung, T. (1998). Some new results on key distribution patterns and broadcast encryption. *Designs, Codes and Cryptography, 14*, 261–279.
3. Waldvogel, M., Caronni, G., Sun, D., Weiler, N., & Plattner, B. (1999). The VersaKey framework: Versatile group key management. *IEEE Journal on Selected Areas in Communications, 7*(8), 1614–1631.
4. Wallner, D., Harder, E., & Agee, R. (1999). Key management for multicast: Issues and architectures. RFC 2627.
5. Wong, C. K., Gouda, M., & Lam, S. S. (1998). Secure group communications using key graphs. In *Proceedings of ACM SIGCOMM'98*.
6. Mittra, S. (1997). Iolus: A framework for scalable secure multicasting. In *Proceedings of ACM SIGCOMM'97* (pp. 277–288).

7. Rafaeli, S., & Hutchison, D. (2003). A survey of key management for secure group communication. *ACM Computing Surveys, 35*(3), 309–329.
8. Rodeh, O., Birman, K., & Dolev, D. (2001). The architecture and performance of security protocols in the ensemble group communication system. *ACM Transactions on Information and System Security, 4*(3), 289–319.
9. McEliece, R. J., & Sarwate, D. V. (1981). On sharing secrets and Reed–Solomon codes. *Communications of the ACM, 26*(9), 583–584.
10. Shamir, A. (1979). How to share a secret. *Communications of the ACM, 24*(11), 612–613.
11. Chou, G. H., & Chen, W. T. (1989). Secure broadcasting using the secure lock. *IEEE Transactions on Software Engineering, 15*(8), 929–934.
12. Fiat, A., & Naor, M. (1994). Broadcast encryption. In *Advances in Cryptology—Proceedings of 13th Annual International Cryptology Conference (CRYPTO'94)* (pp. 480–491).
13. Blundo, C., & Cresti, A. (1995). Space requirement for broadcast encryption. In *Advances in Cryptology—Proceedings of Workshop Theory and Application of Cryptographic Techniques (EUROCRYPT'95)* (pp. 287–298).
14. Blundo, C., De Santis, A., Herzberg, A., Kutten, S., Vaccaro, U., & Yung, M. (1993). Perfectly secure key distribution in dynamic conferences. In *Advances in Cryptology—Proceedings of Workshop Theory and Application of Cryptographic Techniques (EUROCRYPT'93)* (pp. 471–486).
15. Blundo, C., Frota Mattos, L. A., & Stinson, D. R. (1996). Trade-offs between communication and storage in unconditionally secure schemes for broadcast encryption and interactive key distribution. In *Advances in Cryptology—Proceedings of 16th Annual International Cryptology Conference (CRYPTO'96)* (pp. 387–400).
16. Luby, M., & Staddon, J. (1998). Combinatorial bounds for broadcast encryption. In *Advances in Cryptology—Proceedings of International Conference Theory and Application of Cryptographic Techniques (EUROCRYPT'98)* (pp. 512–526).
17. Sherman, A. T., & McGrew, D. A. (2003). Key establishment in large dynamic groups using one-way function trees. *IEEE Transactions on Software Engineering, 29*(5), 444–458.
18. Blakley, G. R. (1979). Safeguarding cryptographic keys. In *Proceedings of American Federation of Information Processing Societies. (AFIPS'79) National Computer Conference* (Vol. 48, pp. 313–317).
19. Laih, C., Lee, J., & Harn, L. (1989). A new threshold scheme and its application in designing the conference key distribution cryptosystem. *Information Processing Letters, 32*, 95–99.
20. Berkovits, S. (1991). How to broadcast a secret. In *Proceedings of Eurocrypt'91 Workshop Advances in Cryptology* (pp. 536–541).
21. Li, C. H., & Pieprzyk, J. (1999). Conference key agreement from secret sharing. In *Proceedings of Fourth Australasian Conference Information Security and Privacy (ACISP'99)* (pp. 64–76).
22. Saze, G. (2003). Generation of key predistribution schemes using secret sharing schemes. *Discrete Applied Mathematics, 128*, 239–249.
23. Harn, L., & Lin, C. (2010). Authenticated group key transfer protocol based on secret sharing. *IEEE Transactions on Computers, 59*(6), 842–846.
24. Hsu, Chingfang, Zeng, Bing, Cui, Guohua, & Chen, Liang. (2013). A new secure authenticated group key transfer protocol. *Wireless Personal Communications*. doi:10.1007/s11277-013-1298-2.
25. Hsu, Chingfang, Cheng, Qi, Tang, Xueming, & Zeng, Bing. (2011). An ideal multi-secret sharing scheme based on MSP. *Information Sciences, 181*(7), 1403–1409.
26. Beimel, A. (1996). Secure schemes for secret sharing and key distribution. *Ph.D. Dissertation*, Technion—Israel Institute of Technology Haifa, Israel.
27. Karchmer M., & Wigderson, A. (1993). On span programs. In *Proceedings of 8th Annual Conference Structure in Complexity*, San Diego, CA (pp. 102–111).
28. Hsu, C., Zeng, B., & Zhang, M. (2014). A novel group key transfer for big data security. *Applied Mathematics and Computation, 249*, 436–443.
29. Cho, J.-H., Chen, I.-R., & Eltoweissy, M. (2008). On optimal batch rekeying for secure group communications in wireless networks. *Wireless Networks, 14*(6), 915–927.
30. Choi, D., Choi, H.-K., & Lee, S.-Y. (2015). A group-based security protocol for machine-type communications in LTE-advanced. *Wireless Networks, 21*(2), 405–419.

Ching-Fang Hsu received the M.Eng. and the Ph.D. degrees in information security from the Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2010 respectively. From Sep. 2010 to Mar. 2013, she was a Research Fellow at the Huazhong University of Science and Technology. She is currently an Associate Professor at Central China Normal University, Wuhan, China. Her research interests are in cryptography and network security, especially in secret sharing and its applications.

Lein Harn received the B.S. degree in electrical engineering from the National Taiwan University in 1977, the M.S. degree in electrical engineering from the State University of New York-Stony Brook in 1980, and the Ph.D. degree in electrical engineering from the University of Minnesota in 1984. He is currently a full Professor at the Department of Electrical and Computer Engineering, University of Missouri, Kansas City (UMKC). He is currently investigating new ways of using secret sharing in various applications.

Yi Mu received his Ph.D. from the Australian National University in 1994. He is a full professor in the School of Computing and Information Technology, University of Wollongong. Prior to joining University of Wollongong in 2003, he was a senior lecturer in the Department of Computing, Macquarie University. He also worked in Department of Computing and IT, University of Western Sydney as a lecturer. His current research interest includes cryptography, network security, access control, and computer security.



Maoyuan Zhang received the Ph.D. degrees from the Huazhong University of Science and Technology, Wuhan, China, in 2005. From 2005 to 2007, he was a Research Fellow at the Huazhong University of Science and Technology. He is currently a full Professor at Central China Normal University, Wuhan, China. His research interests are in computer networks and network security.

Xuan Zhu received the M.S. and the Ph.D. degrees in computer science from the University of Paris XI, Paris, France, in 2003 and 2007 respectively. She is currently a lecturer at Central China Normal University, Wuhan, China. Her research interests include Internet of Things, eHealthcare and speech processing.