Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2008, Article ID 271989, 9 pages
doi:10.1155/2008/271989

Research Article

Constructing UC Secure and Constant-Round Group Key Exchange Protocols via Secret Sharing

Chunjie Cao,[1,2] Chao Yang,[1] Jianfeng Ma,[1] and Sangjae Moon[3]

[1] *Key Laboratory of Computer Networks and Information Security, Xidian University, Xi'an 710071, China*
[2] *No.36 Research Institute of China Electronic Technology Corporation (CETC), Jiaxing, Zhejiang 314033, China*
[3] *Mobile Network Security Technology Research Center, Kyungpook National University, Daeyu 702701, South Korea*

Correspondence should be addressed to Chunjie Cao, chunjie.cao@gmail.com

Received 10 June 2007; Revised 30 October 2007; Accepted 30 April 2008

Recommended by Mohamed Hossam Ahmed

Group key exchange (GKE) is one of the basic building blocks in securing group communication. A number of solutions to GKE problem have been proposed, but most of them are not scalable and require a number of rounds linear with the number of group members. We present a method of constructing constant-round and identity-based protocol via secret sharing for GKE within universally composability (UC) framework. The resultant protocol focuses on round efficiency and three rounds of communication are required. The protocol allows the batch verification of messages signed by all other group participants. Moreover, compared with other identity-based protocols, the key generation center (KGC) in our protocol is not always online.

1. INTRODUCTION

In many modern group-oriented and distributed applications, such as distributed simulation, multiuser games, and collaborative tools, scalable and reliable group communication is one of the critical problems. Regardless of the concrete applications environment, security services are necessarily required to provide communication privacy and integrity, which are impossible without secure and efficient key distribution. A group key exchange (GKE) protocol allows a group of participants to establish a common session key which is used to protect the sensible information.

Among the existing authentication systems, asymmetric technologies such as public key infrastructure (PKI) and identity-based (ID-based) system are commonly adopted. The concept of ID-based cryptosystem was firstly proposed by Shamir [1]. Such a scheme has a unique property that a user's public key can be easily calculated from his identity, while the private key can be calculated by a trusted authority called key generation center (KGC). In a typical PKI system, a user should apply for his public key certificate from a certificate authority (CA) and other partners can use this certificate to authenticate the user. In an ID-based system, however, the partner only needs the public identity of the user, such as email address. Thus, compared with certificate-based PKI system, an ID-based system greatly simplifies the procedure of key management.

Communication security is a very important issue when we design a group key exchange protocol in peer group. Only recently have Bresson, Chevassut, Pointcheval, and Quisquater (BCPQ) given the first provably secure model and protocol [2–4] for group key exchange setting. Their protocol is based on the protocol of Steiner et al. [5], and requires n rounds to establish a key among a group of n users. The BCPQ model is an important step and very helpful in analyzing and designing group key exchange protocols.

1.1. Related works

1.1.1. Group key exchange

A number of studies [5–21] have considered the problem of extending the two-party Diffie-Hellman (DH) protocol [22] to the multiparty setting. A class of generic n-party DH protocols is defined in [5] and extended to provide implicit key authentication in [17], and one practical protocol of which is A-GDH.2. A tree-based DH group key exchange protocol has been proposed by Kim et al. [16, 23] which is shown to be secure against passive adversaries. Also several papers have attempted to establish ID-based authenticated key exchange

protocol. Joux presented a one-round tripartite key exchange protocol [10] using pairings. But it is vulnerable to "man-in-the-middle" attack. Zhang et al. [14] proposed a new ID-based authenticated three-party key exchange protocol, in which the authenticity is assured by a special signature scheme from pairing. Recently, an ID-based group key exchange protocol which uses the one-way function trees and a pairing is proposed by Reddy and Nalla [7] with informal security analysis. Barua et al. [6] introduced an ID-based multiparty key exchange scheme which uses ternary trees. But the protocols of Reddy and Barua have $\lceil \log_2 n \rceil$ and $\lceil \log_3 n \rceil$ communications rounds, respectively, and are not scalable. Wang and Wu [21] proposed an efficient ID-based multicast scheme which needs a group controller. However, in this paper, we will focus on the peer groups and contributory key exchange.

There are two kinds of famous constant-round group key agreement protocols, one is based on the BD scheme which was proposed by Burmester and Desmedt [13], and the other is based on secret sharing scheme. In PKC'04, Choi et al. [8] presented an efficient ID-based group key exchange schemes from bilinear pairings which is an authenticated bilinear variant of BD scheme, but soon found to be flawed by Zhang and Chen [24]. Tzeng and Pieprzyk and Li [25, 26] have shown how secret sharing scheme can be exploited as a building block in group key establishment. Bresson and Catalano proposed a practical and simple group key exchange scheme which combines the ElGamal encryption scheme and the secret sharing technique [19]. Nevertheless, in the protocol of Pieprzyk and Li [26], confidence in fresh of the key depends on a random value supplied by a trusted third party, and this protocol does not provide forward secrecy. Also the scheme [25] of Tzeng lacks forward secrecy.

1.1.2. Provable security for protocols

The basic idea of proving the security of a protocol in a model in which the parties have a random oracle and then instantiating that oracle with an appropriate cryptographic primitive originates in [27, 28]. In 1993, Bellare and Rogaway [29] proposed a formal model for proving security of protocols in a two-party setting. A modular approach is presented by Bellare et al. [30] to design and analyze key exchange protocols. The modularity is achieved by applying an authenticator to protocols which have been proven secure in a much simplified adversarial setting where authentication of communication links is not required. Based on these works, Bresson et al. (BCPQ) defined a sound formalization [3] for the authenticated group DH key exchange and provided provably secure protocols within this model. We refer to protocols secure in BCPQ model as AKE-secure. But AKE-secure does not take into account any notion of protection against "insider attack," and AKE-secure protocols may be completely insecure against attacks by malicious insiders. Katz and Shin [31] proposed a solution within the universally composability (UC) framework [32, 33] which can guarantee the security of a protocol when it runs concurrently with other protocols.

1.2. Our contribution

The purpose of this paper is to present a method of constructing UC-secure constant-round ID-based group key exchange protocols. The resultant protocol is round efficient and only needs three rounds. It allows the batch verification of messages signed by all other group participants, which greatly improves computational efficiency. In addition, the protocol is a contributory key exchange, hence it does not impose a heavy computational burden on a particular party. The most important is that the new protocol is UC-secure and most secret sharing schemes could be adopted to construct our protocol.

2. PRELIMINARIES

2.1. Admissible bilinear map [34]

Let G_1 be a cyclic additive group of prime order q and G_2 be a cyclic multiplicative group of same order q. Let P be an arbitrary generator of G_1. We assume that discrete logarithm problem in both G_1 and G_2 are intractable. A map e: $G_1 \times G_1 \to G_2$ satisfying the following properties is called an admissible bilinear map:

(i) bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ $\forall P, Q \in G_1$ and $a, b \in Z_q^*$,

(ii) nondegeneracy: if P is a generator of G_1, then $e(P, P)$ is a generator of G_2, that is, $e(P, P) \neq 1$,

(iii) computability: there exists an efficient algorithm to compute $e(P, Q)$ $\forall P, Q \in G_1$.

2.2. Computational DH (CDH) problem in G_1

Input: (P, aP, bP) for some $a, b \in Z_q^*$.

Output: abP.

The success probability of any probabilistic polynomial time adversary A in solving CDH problem in G_1 is defined to be:

$$\mathrm{Succ}_{\mathcal{A},G_1}^{\mathrm{CDH}} = \Pr ob\left[\mathcal{A}(P, aP, bP, abP) = 1 : a, b \in Z_q^*\right]. \tag{1}$$

CDH assumption

There exists no algorithm running in expected polynomial time, which can solve the CDH problem with nonnegligible probability. Namely, for any probabilistic polynomial time (PPT) adversary A, $\mathrm{Succ}_{\mathcal{A},G_1}^{\mathrm{CDH}}$ is negligible.

2.3. Aggregate signature

In the construction of our authenticated protocol, we use the bilinear aggregate signature scheme firstly introduced by Boneh et al. [35]. But the base signature scheme is the ID-based bilinear signature scheme proposed by Hess [36].

An aggregate signature scheme is a digital signature that supports aggregation. Concretely, given n signatures on n distinct messages from n distinct participants, it is possible

to aggregate all these signatures into a single short signature. This single signature and the n original messages will convince the verifier that participant u_i indeed signed message m_i. The aggregate signature scheme is formally denoted as $\Lambda = \{G, K, \text{Sig}, \text{Ver}, A\text{Sig}, A\text{Ver}\}$, where $\{G, K, \text{Sig}, \text{Ver}\}$ is a standard digital signature scheme, which is called the base signature scheme. Here G is a randomized system parameters generator algorithm, K is a randomized key generation algorithm, Sig is a randomized signing algorithm, and Ver is a deterministic algorithm. The aggregation signature algorithm and the aggregation verification algorithm are, respectively, ASig and AVer. The aggregate signature is generated as follows:

$$\delta = A\text{Sig}(\delta_1, \delta_2, \ldots, \delta_n), \tag{2}$$

where δ_i is the signature of message m_i relative to public key PK_i and δ is the single aggregate signature. The verification is done by checking whether

$$\begin{aligned} &A\text{ver}(\text{PK}_1, \text{PK}_2, \ldots, \text{PK}_i; m_1, m_2, \ldots, m_n; \\ &\quad A\text{Sig}(\delta_1, \delta_2, \ldots, \delta_n)) = 1. \end{aligned} \tag{3}$$

Note that we set the co-GDH gap groups are equivalent, so the computational co-DH and decisional co-DH problems [37] reduce to the standard CDH and DDH problems [35].

3. THE MODEL

The model described in this section is a *static* group security model extended from one of Bresson et al. [4] which follows the approach of Bellare and Rogaway [29, 38, 39].

3.1. Adversarial model

Let $U = \{U_1, U_2, \ldots, U_n\}$ and $\text{ID} = \{\text{ID}_1, \text{ID}_2, \ldots, \text{ID}_n\}$ be a set of n users and their identities, respectively. Each user U_i has a unique identity ID_i, which is known to all the other users, and all these identities are distinct. Each user can execute the protocol multiple times with different partners: this is modeled by allowing each user an unlimited number of *instances* with which to execute the protocol. We denote instance t of U_i, called an oracle, as $\prod_i^t$ for an integer $t \in N$.

3.1.1. Initialization

In this phase, each user $U_i \in U$ gets his long-term public and private keys. ID-based protocol requires the following initialization phase:

(1) the KGC randomly chooses a secret key $s \in Z_q$ as master key, then computes, and publishes $P_{\text{pub}} = sP$,

(2) when each user with identity ID wants to obtain public/private key pair, the KGC uses its master secret key s to compute the corresponding private key S_{ID} and transmits it to the user through a secure channel.

3.1.2. Queries

Normally, the security of a protocol is related to the adversary's ability, which is formally modeled by queries issued by the adversary. We assume that a probabilistic polynomial time adversary A can completely control the communications and make queries to any instance. We now explain the capability that each kind of query captures.

(i) *Extract* (ID_i): this query allows the adversary to get the long-term private key corresponding to ID_i, where $\text{ID}_i \notin \text{ID}$.

(ii) *Send* ($\prod_i^t$, M): this query allows the adversary to make the user ID_i run the protocol normally and send message M to instance $\prod_i^t$ which will return a reply.

(iii) *Reveal* ($\prod_i^t$): this query models the adversary's ability to find session group keys. If an oracle has accepted, holding a session key K, then K is returned to the adversary. Note that we say that an oracle accepts when it has enough information to compute a session key. At any time, an oracle can accept and it accepts at most once in executing an operation. As soon as an oracle accepts in executing an operation, the session key is defined.

(iv) *Corrupt* (ID_i): this query models the attacks revealing the long-term private key S_i. This does not output any internal data of ID_i.

(v) *Test* ($\prod_i^t$): this query models the semantic security of a session key. This query is allowed only once by the adversary. A random bit b is chosen, if $b = 1$, then the session key is returned, otherwise a random value is returned.

In this model, we consider two types of adversaries according to their attack types. The attack types are simulated by the queries issued by the adversaries. A passive adversary is allowed to issue "Reveal," "Corrupt," and "Test" queries, while an active adversary is additionally allowed to issue "Send" and "Extract" queries.

3.2. Security notions

Definition 1 (partner IDS). Partner identities for instance $\prod_i^t$ which consist of the users (including ID_i himself) with whom $\prod_i^t$ intends to establish a session key. Partner identities of instance $\prod_i^t$ are denoted by pid ($\prod_i^t$).

Definition 2 (session IDS). The *session* ID is the unique identity of a session, which is denoted by sid ($\prod_i^t$). To achieve the goal of UC-security, we follow [30, 33] in assuming that sid is provided by some higher-level protocol when the GKE protocol is first initiated.

Definition 3 (freshness). An oracle is called fresh (or holds a fresh key) if the following two conditions are satisfied. First, nobody in U has ever been asked for a "Corrupt" query from the beginning of the game. Second, in the current operation execution, $\prod_i^t$ has accepted and neither U_i nor his partners have been asked for a "Reveal" query.

$F_{\text{SS-GKE}}$ proceeds as follows, running on security parameter k, with players $U_1, \ldots, U_n$, and an ideal adversary S.

Initialization: upon receiving (sid, pid, new-session) from player U_i for the first time (where pid is a set of at least two distinct user identities containing U_i), record (sid, pid, U_i), and send this to S. In addition, if there are already $|\text{pid}| - 1$ recorded tuples (sid, pid, U_j) for $U_j \in \text{pid} \setminus \{U_i\}$, then store (sid, pid, Initialized) and send this to S.

Secret Distribution: upon receiving a message (Share, sid, pid, U_i) from U_i, where there is a recorded tuple (sid, pid, Initialized), do the following:

(i) if all $U \in$ pid are uncorrupted, choose $k_i \leftarrow \{0,1\}^k$ and compute $P_i = g(k_i)$, where g is a function that can generate $|\text{pid}| - 1$ secret shares. Afterward record (shared, sid, pid, U_i, P_i) and send it to all players and S. If there are already $|\text{pid}| - 1$ recorded tuples (shared, sid, pid, U_i, P_i,) then store (sid, pid, SecretDistributed) and send this to S,

(ii) if U_i is corrupted, wait for S to send k_i', P_i' and then record (shared, sid, pid, U_i, P_i').

Key Generation: upon receiving a message (*KeyGeneration*) from S where there is a recorded tuple (sid, pid, SecretDistributed), do the following:

(i) if all $U \in$ pid are uncorrupted, compute key $= f(k_1, k_2, \ldots, k_{|\text{pid}|-1})$, finally, store (sid, pid, key),

(ii) if any of the $U \in$ pid are corrupted, send the corresponding secret k_i to S, wait for S to send a message (SecretKey, key$'$) and then store (sid, pid, key$'$).

Key Delivery: if S sends a message (deliverKey, U_i), where there is a recorded tuple (sid, pid, key) and $U_i \in$ pid, then send (sid, pid, key) to player U_i.

Player Corruption: if S corrupts $U_i \in$ pid where there is a recorded tuple (sid, pid, key) and message (sid, pid, key) has not yet been sent to U_i, then the adversary is given key. Otherwise, S is given nothing.

Algorithm 1: Secret sharing-based GKE ideal functionality $F_{\text{SS-GKE}}$.

Let F be a collision-resistant pseudorandom function, and assume that v_0, $v_1 \leftarrow \{0,1\}^k$ are publicly known and $v_0 \neq v_1$.

Initialization Phase: each player U_i generates long-term verification/signing keys $(\text{PK}_i; \text{SK}_i)$ (in addition to any keys needed for π).

The Protocol: players run protocol π. If U_i would terminate without accepting in π, then it terminates without accepting in π'. Otherwise, if U_i would accept in protocol π with output $(\text{sid}_i, \text{pid}_i, \text{key}_i)$, this player performs the following additional steps:

(i) U_i computes $\text{ack}_i = F(\text{key}_i, v_0)$ and $\text{sk}_i = F(\text{key}_i, v_1)$. Next, U_i erases all its local state except for ack_i; sk_i, sid_i, and pid_i. Then, U_i computes a signature $\sigma_i = \text{Sign}(\text{SK}_i, (U_i; \text{sid}_i, \text{pid}_i, \text{ack}_i))$ and sends the message $(U_i; \sigma_i)$ to all players in pid_i,

(ii) upon receipt of $|\text{pid}_i| - 1$ messages $(U_j; \sigma_j)$ from all other players $U_j \in \text{pid}_i \setminus \{U_i\}$, U_i checks that Verify $(\text{PK}_j, (U_j; \text{sid}_j, \text{pid}_j, \text{ack}_j), \sigma_j) = 1 \ \forall U_j \in \text{pid}_i$. If all verifications are successful, then U_i accepts and erases its internal state, and outputs $(\text{sid}_i, \text{pid}_i, \text{sk}_i)$. Otherwise, U_i terminates without accepting.

Algorithm 2: AKE → UC compiler.

Definition 4 (authenticated key exchange security (AKE security)). We say that event Succ occurs if the adversary issues "Test" query to a fresh oracle and correctly guesses the bit b (distinguishing the key from a random string). The advantage of an adversary A in attacking protocol P is defined as $\text{Adv}_A^P(k) = |2\Pr[\text{Succ}] - 1|$.

A protocol P is AKE secure, if the following two properties are satisfied:

(i) *consistency*: in the presence of an adversary, all partner oracles accept the same key,

(ii) *Secrecy*: for any PPT adversary A, $\text{Adv}_A^P(k)$ is negligible.

Definition 5 (perfect forward secrecy). A protocol provides perfect forward secrecy if an adversary does not get nonnegligible knowledge information about session keys previously established when making "Corrupt" queries to all group members. We define $\text{Adv}_A^P(t, q_s, q_h)$ to be the maximal advantage of any active adversary attacking protocol P, running in time t and making q_s "Send" queries and q_h "Hash" queries.

Note that we do not define any notion of explicit authentication or, equivalently, confirmation that the other members of the group have computed the common key. However, explicit authentication in our protocol can be achieved at little additional cost. Previous work [4] shows how to achieve explicit authentication for any group authenticated key exchange protocol using one additional round and minimal extra computation.

4. UNIVERSAL COMPOSABLE GKE PROTOCOLS VIA SECRET SHARING

In this section, we introduce the ideal functionality for group key exchange protocol via secret sharing within the UC

Round 1. **Initialization**: each participant U_i picks randomly r_i, $r_i' \in Z_q^*$, computes, and broadcasts $(\text{sid}, O_i = r_iP, O_i' = r_i'P)$.

Round 2. **Secret Distribution**: on receiving O_j with the correct sid, each participant U_i picks randomly $K_i \in Z_q$ and computes a polynomial $f_i(x) = K_i + a_{i_1}x + a_{i_2}x^2 + \cdots + a_{i_{n-1}}x^{n-1}$ passing points $(j, H_3(r_iO_j))$, $1 \le j \le n$, $j \ne i$ and $(0, K_i)$. Then computes

$$P_{ij} = f_i(n+j), \quad 1 \le j \le n,\ j \ne i;$$

$$P_i = P_{i1}\|P_{i2}\|\cdots\|P_{in}; \qquad O = O_1\|O_2\|\cdots\|O_n,$$

$$O' = O_1'\|O_2'\|\cdots\|O_n';$$

$$h_i = H_2(P_i\|O\|O'\|K_i\|\text{sid}\|\text{pid});$$

$$\delta_i = r_iP_{\text{pub}} + h_iS_i$$

and broadcasts $(\text{sid}, P_i, \delta_i)$.

Round 3. **Key Confirmation**

On receiving (P_{jl}, δ_l) with correct sid, $1 \le l \le n$, $l \ne j \ne i$, each participant U_i computes polynomial $f_j'(x)$ of degree n that passes $(n+l, P_{jl})$ and $(i, H_3(r_iO_j))$. Then U_i computes $K_{ij} = f_j'(0)$ and checks

$$e\left(\sum_{j=1}^{n}\delta_j, P\right) = e\left(\sum_{j=1}^{n}(O_j + h_jQ_j), P_{\text{pub}}\right).$$

If the above aggregate signature is verified successfully, U_i computes

$$\text{key}_i == H_4(K_1 + K_2 + \cdots + K_n); \qquad \text{ack}_i = H_4(\text{key}_i, v_0); \qquad \text{sk}_i = H_4(\text{key}_i, v_1),$$

$$h_i' = H_2(\text{ID}_i\|\text{ack}_i\|\text{sid}\|\text{pid}); \qquad \delta_i' = r_i'P_{\text{pub}} + h_i'S_i$$

and broadcasts $(\text{sid}, \text{ID}_i, \delta_i')$.

When receiving $|\text{pid}| - 1$ messages from other participants, U_i verifies the aggregate signature as above. If the verification is successful, U_i accepts with $(\text{sid}, \text{pid}, \text{sk}_i)$.

Algorithm 3: The UC-secure GKE protocol ID-SS.

framework [32, 33]. Then, we show that an AKE-secure GKE protocol based on secret sharing can be compiled to be a UC-secure protocol by applying the compiler (depicted in Algorithm 2 proposed by Katz and Shin [31]. Our secret sharing-based GKE ideal functionality $F_{\text{SS-GKE}}$ is depicted in Algorithm 1. In the following, we assume that (1) the underlying group communication system is resistant to fail-stop failures, which means that the system should provide a consistent membership view to all group members and reliable and causally ordered multicasts; (2) unicast and multicast are reliable. We assume that any user can broadcast messages to others in the broadcast network.

In ACM CCS 2005, Katz and Shin proposed a compiler and [31] for GKE protocol, where an AKE-secure GKE protocol π can be compiled to be a UC-secure protocol π'. To construct our UC-secure and constant-round GKE protocol via secret sharing, this compiler is involved in our protocol and is a key component. The compiler is depicted in Algorithm 2.

Theorem 1 (see [31]). *If π is an AKE-secure GKE protocol, then applying the AKE → UC compiler to π results in a UC-secure protocol π'.*

From Theorem 1, we can get the following corollary.

Corollary 1. *If π is an AKE-secure GKE protocol based on secret sharing, then applying the AKE → UC compiler to π results in a UC-secure protocol π'.*

5. THE PROTOCOL ID-SS

To construct the UC-secure ID-based GKE protocol via secret sharing, we are motivated by the scheme of Shamir [1]. The resultant protocol is denoted as ID-SS. We assume that there exists an authenticated secure channel between the user and KGC for the distribution of the long-term private key.

System setup

Given the security parameter q, the KGC chooses groups G_1 and G_2 of prime order q, a generator P of G_1, and a bilinear map e: $G_1 \times G_2 \to G_2$. Let H_1: $\{0,1\}^* \to G_1$ be a map-to-point hash function, H_2: $G_1 \times Z_q \times \{0,1\}^* \times \{0,1\}^* \to Z_q$, H_3: $G_1 \to Z_q$ be other two hash functions, and H_4 be a key derivation function. H_1, H_2, H_3, and H_4 are considered as random oracles. Also the KGC randomly selects v_0, $v_1 \leftarrow \{0,1\}^q$, the master secret key $s \in Z_q$ and computes $P_{\text{pub}} = sP \in G_1$ that is made public. Then KGC publishes the following system parameters:

$$\{e, G_1, G_2, q, P, P_{\text{pub}}, H_1, H_2, H_3, H_4, v_0, v_1\}. \tag{4}$$

Extract

Given a public identity $\text{ID} \in \{0,1\}^*$, the KGC computes $Q_{\text{ID}} = H_1(\text{ID}) \in G_1$ and associated private key $S_{\text{ID}} = sQ_{\text{ID}} \in G_1$ that is transmitted to the user.

Let $U = \{U_1, U_2, \ldots, U_n\}$ be a set of users who want to establish a common session key and ID_i be the identity of U_i. Then the public and private key pair of U_i is $(\mathrm{ID}_i, S_i = sQ_i)$. Now we describe the protocol in Algorithm 3.

6. SECURITY ANALYSIS OF THE PROTOCOL ID-SS

Theorem 2. *Suppose that the hash functions H_1, H_2, H_3, and H_4 are random oracles. Then the protocol ID-SS is an AKE-secure protocol providing perfect forward secrecy under the CDH assumption. Concretely,*

$$\mathrm{Adv}_A^{\text{ID-SS}}(t, q_s, q_h) \le 2 \cdot n \cdot \mathrm{Succ}_{\Gamma}^{\mathrm{Forgery}_{\Lambda}}(t) + 2 \cdot l \cdot q_h \cdot \mathrm{Succ}_{\Psi}^{\mathrm{CDH}}(t). \tag{5}$$

Proof. Firstly, we prove the correctness of the protocol. In other words, if all users follow the process of the protocol, they can compute a common group key. Because of $r_i O_j = r_i r_j P = r_j O_i$, user U_i can compute the polynomial $f_j(x)$ passing $(n + l, P_{jl})$, $1 \le l \le n$ and $(i, r_i O_j)$ according to the messages related to user U_j. Then U_i computes $K_j = f_j(0)$. By verifying aggregate signature δ, U_i can check whether K_j is correct or not. So all participants can derive the same group key $K = H_4(K_1 + K_2 + \cdots + K_n)$.

Secondly, we prove that the protocol is a GK secure protocol in the presence of an adversary A,

(1) assuming that A modifies the flows, then we build a forger Γ,

(2) assuming that A does not modify the flows, then we build a CDH-solver Ψ. □

Forger Γ

Assume that A breaks the protocol ID-SS by forging a signature at least with the probability φ. We can construct a forger Γ that generates a valid message pair (ID, m, δ) from A. Γ receives ID as the input and accesses a (public) signing oracle. Γ randomly picks $i \in [1, n]$ and honestly generates all other public and private keys for the system. However, for user U_i, Γ sets ID as U_i's public key. Then Γ starts running A as a subroutine and answers the oracle queries made by A as follows:

(i) when A makes "Send $(*, m)$" queries, Γ responds in a straightforward way. When A makes "Send $(*, m, \delta)$" queries, Γ responds in a straightforward way using long-term keys to sing the flows except if A makes the query of the form "Send $(\prod_j^t, m, \delta)$." If this occurs, Γ goes through the signing oracle and stores the response in a variable α,

(ii) when A makes a "Reveal" query, Γ gives the session key to A,

(iii) when A makes a "Corrupt" query, Γ answers in a straightforward way except if A makes the query of "Corrupt (ID)". If this occurs, Γ stops and outputs "Fail,"

(iv) when A makes a "Hash" query, Γ answers as a random oracle in a straightforward way,

(v) when A makes a "Test" query, since all the accepted session keys are known from "Reveal" queries, the query can be answered with the correct session key.

If A has already issued the query of "Send $(\prod_j^t, m, \delta)$," where δ is a valid signature on m with respect to ID and $(m, \delta) \notin \alpha$, then Γ stops and outputs (m, δ) as a forgery. Otherwise, Γ simply aborts. So the probability $\mathrm{Succ}_{\Gamma}^{\mathrm{Forgery}_{\Lambda}}(t)$ of Γ outputting a forgery is the product of the probability that A generates a valid signature and the probability that A correctly guesses the value of i:

$$\mathrm{Succ}_{\Gamma}^{\mathrm{Forgery}_{\Lambda}}(t) \ge \frac{\varphi}{n}. \tag{6}$$

CDH-attacker Ψ

Next, we assume that A breaks the protocol ID-SS without generating a forgery of signature. Thus from A, we can construct a CDH-attacker Ψ that breaks the protocol by solving an instance of the CDH problem.

Let l be an upper bound on the number of sessions invoked by A, then Ψ randomly chooses and $\gamma \in [1, l]$ representing a guess that as to which query of A activates the instance for which A will ask its "Test" query.

Ψ receives an instance $(P, aP$, and $bP)$ of the CDH problem as input and randomly selects $i, j \in [1, n]$.

Then Ψ starts running A as a subroutine and answers the oracle queries made by A. We now describe the simulation of the oracle queries of A in detail.

(i) When A makes a "Send $(*, m)$" query, Ψ proceeds as in protocol ID-SS using a random value except if the query is "Send (Π_i, m)" or "Send (Π_j, m)" query in the γth session. If this occurs, Ψ sets $O_i = aP$, $O_j = bp$. When A makes "Send $(*, m, \delta)$" queries, Ψ responds in a straightforward way using long-term keys to sing the flows except if the query is "Send (Π_i, m, δ)" or "Send (Π_j, m, δ)" query in the γth session. If this occurs, Ψ responds using a random value and long-term keys to sing the flows,

(ii) when A makes a "Corrupt query," Ψ answers with the corresponding long-term private key in a straightforward way,

(iii) when A makes a "Reveal query," Ψ answers in a straightforward way except if the session key is of the γth session. In the latter case, Ψ stops and outputs "Fail,"

(iv) when A makes a "Hash query," Ψ answers as a random oracle in a straightforward way,

(v) when A makes a "Test query," Ψ answers with a random string.

Since Ψ knows all the keys except for one execution of ID-SS, this simulation is perfectly indistinguishable from an execution of the real protocol ID-SS.

At some stage, A completes and returns a value b'. The probability that Ψ correctly guesses on which session key A

will make the "Test query" is the probability that Ψ correctly guesses the value γ. That is $\mu = 1/l$.

Let ask H be the event that A makes a "Hash query" on $(K_1 + K_2 + \cdots + K_n)$ and Forge be the event that A forges a signature with regard to some participant's long-term public key. We emphasize that, in the random oracle model, A cannot get any advantage on a random value without asking for it. The success probability of Ψ is the probability that A asks the correct value to the hash oracle multiplied by the probability that Ψ correctly chooses the "Hash query" and multiplied by the probability that Ψ correctly guesses the value γ. That is:

$$\mathrm{Succ}_{\Psi}^{\mathrm{CDH}}(t) \geq \frac{\Pr[\text{ask } H]}{q_{H \cdot l}}. \tag{7}$$

Finally, we have:

$$\begin{aligned}
&\Pr[b = b'] \\
&\quad = \Pr[b = b' \mid \text{Forge}] \Pr[\text{Forge}] \\
&\qquad + \Pr[b = b' \mid \neg\text{Forge}] \Pr[\neg\text{Forge}] \\
&\quad \leq \Pr[b = b' \mid \text{Forge}] \\
&\qquad + \Pr[b = b' \mid \neg\text{Forge}] \Pr[\neg\text{Forge}] \\
&\quad \leq \varphi + \Pr[b = b' \mid \neg\text{Forge}] \Pr[\neg\text{Forge}] \\
&\quad \leq \varphi + \Pr[\neg\text{Forge} \wedge \text{ask } H] \Pr[b = b' \mid \neg\text{Forge} \wedge \text{ask } H] \\
&\qquad + \Pr[\neg\text{Forge} \wedge \neg\text{ask } H] \Pr[b = b' \mid \neg\text{Forge} \wedge \neg\text{ask } H] \\
&\quad = \varphi + \Pr[b = b' \mid \neg\text{Forge} \wedge \text{ask } H] \Pr[\neg\text{Forge} \wedge \text{ask } H] + \frac{1}{2} \\
&\quad \leq \varphi + \Pr[\neg\text{Forge} \wedge \text{ask } H] + \frac{1}{2} \\
&\quad \leq \varphi \Pr[\text{ask } H] + \frac{1}{2}.
\end{aligned} \tag{8}$$

Then from the definition $\mathrm{Adv}_A^P(k) = |2\Pr[\mathrm{Succ}] - 1|$ and above three equations, we can get the result as follows:

$$\mathrm{Adv}_A^{\text{ID-SS}}(t, q_s, q_h) \leq 2 \cdot n \cdot \mathrm{Succ}_{\Gamma}^{\mathrm{Forgery}_{\Lambda}}(t) + 2 \cdot l \cdot q_h \cdot \mathrm{Succ}_{\Psi}^{\mathrm{CDH}}(t). \tag{9}$$

We next show that the authentication scheme Λ is secure against existential forgery on adaptively chosen ID attack.

Lemma 1. *Let G_1 be an additive group with order q and the map-to-point hash function H_1 be a random oracle. We assume that the PPT forger A breaks the bilinear aggregate signature scheme Λ for an adaptively chosen ID with advantage ε_0 and running time t_0. Suppose that A makes at most q_{H_1} queries to the hash function H_1. Then from A, we can construct a PPT forger B for a given ID with advantage $\varepsilon_0 \leq \varepsilon_1(1 - 1/q)/q_{H_1}$ and running time $t_1 \leq t_0$.*

Lemma 2. *Let the hash function H_1, H_2 be random oracles. Suppose that B is a PPT forger for a given ID with advantage $\varepsilon_1 \geq 10q_{H_1}(q_s + q_{H_2})/(q-1)$ and running time t_1. Suppose that B makes at most q_{H_1}, q_{H_2}, q_s, and q_{ex} queries to the H_1, H_2, "Send" and "Extract" oracles, respectively. Then from B, we can construct a PPT attacker C that can solve the CDH problem within time $t_2 \leq 120686q_{H_2}t_1/\varepsilon_1$.*

The security analysis of Lemmas 1 and 2 is similar to that of [8], for space limitation, we omit the proof of them. Then, we can directly obtain the following theorem from the above two lemmas.

Theorem 3. *Let H_1, H_2 be random oracles. Then the bilinear aggregate signature scheme Λ on G_1 is secure against existential forgery on adaptively chosen ID attack under the CDH assumption.*

Then from Theorem 2 and Corollary 1, we deduce the following theorem.

Theorem 4. *Suppose that the hash functions H_1, H_2, H_3, and H_4 are random oracles. Then the protocol ID-SS is UC-secure.*

7. CONCLUSION

In this paper, a method of constructing UC secure and constant-round GKE protocol was presented. It allows modular design and analysis of the GKE protocol and the resultant protocol only needs three communication rounds to compute a common group key. Moreover, most secret sharing schemes could be adopted to construct UC secure and constant-round GKE protocol according to our method.

The efficiency of protocols with UC security is usually low for their high security rank. As future work, we plan to formally examine the possibility of extending this security model to improve the performance of protocols by analyzing the relation between security and efficiency under UC framework.

ACKNOWLEDGMENTS

This work was supported in part by the National High Technology Research and Development Program of China (2007AA01Z429, 2007AA01Z405), and the National Natural Science Foundation of China (60573036, 60503012, 60702059).

REFERENCES

[1] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of the 4th Annual International Cryptology Conference (CRYPTO '84)*, vol. 196 of *Lecture Notes in Computer Science*, pp. 47–53, Santa Barbara, Calif, USA, August 1984.

[2] E. Bresson, O. Chevassut, and D. Pointcheval, "Provably authenticated group Diffie-Hellman key exchange—the dynamic case," in *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT '01)*, vol. 2248 of *Lecture Notes in Computer Science*, pp. 290–309, Gold Coast, Australia, December 2001.

[3] E. Bresson, O. Chevassut, and D. Pointcheval, "Dynamic group Diffie-Hellman key exchange under standard assumptions," in *Proceedings of the 21st Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt '02)*, vol. 2332 of *Lecture Notes in Computer Science*, pp. 321–336, Amsterdam, The Netherlands, April-May 2002.

[4] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater, "Provably authenticated group Diffie-Hellman key exchange," in *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS '01)*, pp. 255–264, Philadelphia, Pa, USA, November 2001.

[5] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman key distribution extended to group communication," in *Proceedings of the 3rd ACM Conference on Computer and Communications Security (CCS '96)*, pp. 31–37, New Delhi, India, March 1996.

[6] R. Barua, R. Dutta, and P. Sarkar, "Extending Joux's protocol to multi party key agreement," in *Proceedings of the 4th International Conference on Cryptology in India (INDOCRYPT '03)*, vol. 2904 of *Lecture Notes in Computer Science*, pp. 205–217, New Delhi, India, December 2003.

[7] K. C. Reddy and D. Nalla, "Identity based authenticated group key agreement protocol," in *Proceedings of the 3rd International Conference on Cryptology in India (INDOCRYPT '02)*, vol. 2551 of *Lecture Notes in Computer Science*, pp. 215–233, Hyderabad, India, December 2002.

[8] K. Y. Choi, J. Y. Hwang, and D. H. Lee, "Efficient ID-based group key agreement with bilinear maps," in *Proceedings of the 7th International Workshop on Theory and Practice in Public Key Cryptography (PKC '04)*, vol. 2947 of *Lecture Notes in Computer Science*, pp. 130–144, Singapore, March 2004.

[9] C. Cao, J. Ma, and S. Moon, "Provable efficient certificateless group key exchange protocol," *Wuhan University Journal of Natural Sciences*, vol. 12, no. 1, pp. 41–45, 2007.

[10] A. Joux, "A one round protocol for tripartite Diffie-Hellman," in *Proceedings of the 4th International Symposium on Algorithmic Number Theory (ANTS '00)*, vol. 1838 of *Lecture Notes in Computer Science*, pp. 385–394, Leiden, The Netherlands, July 2000.

[11] M. Steiner, G. Tsudik, and M. Waidner, "Key agreement in dynamic peer groups," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 8, pp. 769–780, 2000.

[12] I. Ingemarsson, D. T. Tang, and C. K. Wong, "A conference key distribution system," *IEEE Transactions on Information Theory*, vol. 28, no. 5, pp. 714–720, 1982.

[13] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," in *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques (Eurocrypt '94)*, vol. 950 of *Lecture Notes in Computer Science*, pp. 275–286, Perugia, Italy, May 1994.

[14] F. Zhang, S. Liu, and K. Kim, "ID-based one round authenticated tripartite key exchange protocol with pairings," *Cryptology ePring Archive*, Report 2002/122, 2002.

[15] K. Becker and U. Wille, "Communication complexity of group key distribution," in *Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS '98)*, pp. 1–6, San Francisco, Calif, USA, November 1998.

[16] Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," in *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS '00)*, pp. 235–244, Athens, Greece, November 2000.

[17] G. Ateniese, M. Steiner, and G. Tsudik, "New multiparty authentication services and key agreement protocols," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 628–639, 2000.

[18] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Transactions on Networking*, vol. 8, no. 1, pp. 16–30, 2000.

[19] E. Bresson and D. Catalano, "Constant round authenticated group key agreement via distributed computation," in *Proceedings of the 7th International Workshop on Theory and Practice in Public Key Cryptography (PKC '04)*, vol. 2947 of *Lecture Notes in Computer Science*, pp. 115–129, Singapore, March 2004.

[20] M. Abdalla, E. Bresson, O. Chevassut, and D. Pointcheval, "Password-based group key exchange in a constant number of rounds," in *Proceedings of the 9th International Conference on Theory and Practice of Public-Key Cryptography (PKC '06)*, vol. 3958 of *Lecture Notes in Computer Science*, pp. 427–442, New York, NY, USA, April 2006.

[21] L. Wang and C.-K. Wu, "Efficient identity-based multicast scheme from bilinear pairing," *IEE Proceedings: Communications*, vol. 152, no. 6, pp. 877–882, 2005.

[22] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[23] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Transactions on Information and System Security*, vol. 7, no. 1, pp. 60–96, 2004.

[24] F. Zhang and X. Chen, "Attack on an ID-based authenticated group key agreement scheme from PKC 2004," *Information Processing Letters*, vol. 91, no. 4, pp. 191–193, 2004.

[25] W.-G. Tzeng, "A secure fault-tolerant conference-key agreement protocol," *IEEE Transactions on Computers*, vol. 51, no. 4, pp. 373–379, 2002.

[26] J. Pieprzyk and C.-H. Li, "Multiparty key agreement protocols," *IEE Proceedings: Computers and Digital Techniques*, vol. 147, no. 4, pp. 229–236, 2000.

[27] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *Journal of the ACM*, vol. 33, no. 4, pp. 792–807, 1986.

[28] O. Goldreich, S. Goldwasser, and S. Micali, "On the cryptographic applications of random functions," in *Proceedings of the 4th Annual International Cryptology Conference (CRYPTO '84)*, vol. 196 of *Lecture Notes in Computer Science*, pp. 276–288, Santa Barbara, Calif, USA, August 1984.

[29] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS '93)*, pp. 62–73, Fairfax, Va, USA, November 1993.

[30] M. Bellare, R. Canetti, and H. Krawczyk, "A modular approach to the design and analysis of authentication and key exchange protocols," in *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC '98)*, pp. 419–428, Dallas, Tex, USA, May 1998.

[31] J. Katz and J. S. Shin, "Modeling insider attacks on group key-exchange protocols," in *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS '05)*, pp. 180–189, Alexandria, Va, USA, November 2005.

[32] R. Canetti, "Universally composable security: a new paradigm for cryptographic protocols," in *Proceedings of the 42nd Foundations of Computer Science Symposium (FOCS '01)*, pp. 136–145, Las Vegas, Nev, USA, October 2001.

[33] R. Canetti and H. Krawczyk, "Universally composable notions of key exchange and secure channels," in *Proceedings of the 21st International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt '02)*, vol. 2332 of *Lecture Notes in Computer Science*, pp. 337–351, Amsterdam, The Netherlands, April-May 2002.

[34] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.

[35] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proceedings of the 22nd Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt '03)*, vol. 2656 of *Lecture Notes in Computer Science*, pp. 416–432, Warsaw, Poland, May 2003.

[36] F. Hess, "Efficient identity based signature schemes based on pairings," in *Proceedings of the 17th Annual ACM Symposium on Applied Computing (SAC '02)*, pp. 310–324, Madrid, Spain, March 2002.

[37] R. Dutta, R. Barua, and P. Sarkar, "Pairing-based cryptography: a survey," *Cryptology ePrint Archive*, Report 2004/064, 2004.

[38] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Proceedings of the 13th Annual International Cryptology Conference (CRYPTO '93)*, vol. 773 of *Lecture Notes in Computer Science*, pp. 232–249, Santa Barbara, Calif, USA, August 1993.

[39] M. Bellare and P. Rogaway, "Provably secure session key distribution: the three party case," in *Proceedings of the 27th Annual ACM Symposium on Theory of Computing (STOC '95)*, pp. 57–66, Las Vegas, Nev, USA, May-June 1995.