# Group Key Agreement From Signcryption

**Xixiang Lv and Hui Li**
State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, Peoples R China
[e-mail: xxlv@mail.xidian.edu.cn; lihui@mail.xidian.edu.cn]
* Corresponding author: Xixiang Lv

*Received August 12, 2012; revised October 4, 2012; accepted October 30, 2012; published December 27, 2012*

## Abstract

There is an intuitive connection between signcryption and key agreement. Such a connector may lead to a novel way to construct authenticated and efficient group key agreement protocols. In this paper, we present a primary approach for constructing an authenticated group key agreement protocol from signcryption. This approach introduces desired properties to group key agreement. What this means is that the signcryption gives assurance to a sender that the key is available only to the recipient, and assurance to the recipient that the key indeed comes from the sender. Following the generic construction, we instantiate a distributed two-round group key agreement protocol based on signcryption scheme given by Dent [8]. We also show that this concrete protocol is secure in the outsider unforgeability notion and the outsider confidentiality notion assuming hardness of the Gap Diffie-Hellman problem.

***Keywords:*** Secure group communication, Group key agreement, Signcryption

This work was sponsored by the National Natural Science Foundation of China (Grant No. 61202388), the National Science & Technology Pillar Program (2008BAH22B03, 2007BAH08B01) and the Fundamental Research Funds for the Central Universities (K5051201019).
**http://dx.doi.org/10.3837/tiis.2012.12.017**

## 1. Introduction

Secure group communication is of great importance for many collaborative and distributed applications, such as video conferencing, collaborative computation, file sharing via internet, secure group chat, group purchase of encrypted content and so on. In order to achieve secure group communications in such scenarios, a group key for fast encryption and decryption must be shared only by group members. Group communication messages are encrypted with the group key, and thus a group member can collaborate with other members without worrying about information leakage. An authenticated key agreement protocol aims at ensuring that no entity other than group members can possibly compute the agreed session key. This kind of protocols is secure against active adversaries. We know that active adversaries are more powerful since they are assumed to have a complete control of the communication channel. Hence, it is essential for a group key agreement (GKA) protocol to be resistant against attacks from active adversaries. This calls for authenticated key agreement protocols.

The main features we would like to find in a GKA scheme are security and efficiency, in the presence of an active adversary. The typical approach to the problem [1-4] requires some data to go through the complete set of parties, which by sequentially adding some private contribution, "build" the actual key in a linear number of rounds of communication. The main problem with this approach is, of course, that it may lead to very slow protocols. To improve on communication complexity, the natural solution is to try to design a scheme that allows for simultaneous sending of contributions. The signcryption based Authenticated GKA protocol in this paper is designed with the above concept.

Signcryption, introduced by Zheng [5], is an asymmetric cryptographic primitive that provides both privacy and authenticity at greater efficiency than the generic composition of signature and encryption schemes. Zheng [6] later observed that a signcryption scheme can be used as a key transport protocol by simply choosing a new key and sending it in a signcrypted message. Choudary [7] defined a security model for this construction. However, they did not present how to extend these notions to the group setting. Actually, signcryption scheme can also be used to construct group key agreement protocols with two rounds. This approach also introduces desired properties to group key agreement. What this means is that the signcryption gives assurance to the sender that the key is available only to the intended recipient, and assurance to the recipient that the key came from the intended sender. The main purpose of this paper is to define a primary approach for constructing an authenticated group key agreement protocol from signcryption. We also give a concrete authenticated group key agreement scheme based on signcryption.

## 2. Paper Organization

The rest of the paper is organized as follows: We outline our main contributions in Section 3 and survey the related works in Section 4. Next, a primary generic construction idea of an unforgeable GKA protocol from signcryption is presented in Section 5, followed by a concrete protocol from signcryption scheme of Dent [8] in Section 6. Then, the security analysis is given in Section 7. This paper is finally concluded in Section 8.

## 3. Our Contribution

In this paper, we consider a different approach to the existing solutions of group key agreement, namely to incorporate the signcryption into the design of group key agreement. As we will show in this paper, the adoption of signcryption key establishment to the group scenario is not very straightforward. Nonetheless, by constructing a recursive chain from signcryption and then extracting group key from this recursive chain, we obtain an unforgeable group key agreement protocol. In fact, an unforgeable GKA protocol means an authenticated GKA protocol that is secure with unforgeability. The unforgeability makes the recipient be assured that the key indeed comes from the intended sender and does not undergo any modification. To support our idea, we prove that our scheme is secure in the outsider unforgeability notion and the outsider confidentiality notion assuming hardness of the Gap Diffie-Hellman (GDH) problem.

## 4. Related Work

Group key agreement (GKA) is the core issue of secure group communication. Up to now, it is still an elusive open problem to construct an efficient GKA protocol. Since the publication of two-party Diffie-Hellman (DH) key exchange in 1976 [9], various solutions have been proposed to extend Diffie-Hellman key exchange to multi-party key agreement, such as [10][11][12][13][14][15]. Wu et al. [16] proposed the first one-round asymmetric GKA protocol. However, none of the prior arts is given a property of unforgeability. Katz, J. and Yung, M. [17] utilized a compiler to transform the well-known BD protocol [12] to an authenticated protocol that is secure against an active adversary. Gorantla, M. et al. [18] leveraged an mKEM (multi key encapsulation mechanism [19]) to achieve an authenticated GKA protocol, in which by using mKEM each group member encapsulates a random nonce and broadcast it as its contribution towards the session key. Instead of these traditional solutions, we will present how to construct a mutual authenticated group key agreement protocol from signcryption schemes by constructing an ordered key chain and a recursive relation. Our concept grew out of a search for mutual authenticated GKA schemes that is secure with the unforgeability and a research on key establishment from signcryption schemes that are reviewed as follows.

Zheng who introduced the notion of signcryption later observed that a signcryption scheme can be used as a key transport protocol by simply choosing a new key and sending it in a signcrypted message [6]. This intuitively gives the desired properties for key establishment since the signcryption gives assurance to the sender that the key is available only to the recipient, and vice versa. This merit is also desired in GKA. In this paper, we will use signcryption to construct a mutual authenticated group key agreement protocol. What follows are the prior arts that are more germane to our work. Dent [8][20] discussed how a signcryption KEM can be used as a one-pass key establishment protocol. Bjørstad and Dent [21] proposed the concept of signcryption tag-KEM and claimed that better key establishment mechanisms can be built with signcryption tag-KEM. M. Choudary Gorantla et al. [7] define security notions and show how signcryption and one-pass key establishment can be related under those notions. Dent [22] discussed some aspects about key transport protocols and key agreement protocols from signcryption techniques, including security models, entity authentication and key compromise impersonation attacks. However, none of these papers mentioned about any relations between group key agreement and signcryption. Actually,

group key agreement protocols can also be built from signcryption schemes, which is the main purpose of this paper.

## 5. Group Key Agreement from Signcryption

We now consider the generic construction for an authenticated two-round group key agreement protocol by using signcryption algorithms. What follows is the main idea. All group participants $U_1,U_2,\cdots,U_n$ are organized in an ordered chain and $U_{i+1}$ is the successor of $U_i$. The symmetric session key computed in a signcryption algorithm is used as the shared secret between the participant $U_i$ and its successor $U_{i+1}$, $i=1,\cdots,n$. Specifically, the outgoing message of the signcryption algorithm becomes the key encapsulation, and the session key computation process at the receiver $U_{i+1}$ in the signcryption algorithm can be used as the decapsulation algorithm to retrieve the symmetric key. The signcryption gives confidentiality protection and origin authentication to the group key agreement protocol which includes the following two rounds.

**Round 1.** Let $U=\{U_1,U_2,\cdots,U_n\}$ be the set of protocol participants. All the participants $U_1,U_2,\cdots,U_n$ run the following process. The participant $U_i$ sends signcryption message $\psi_i$ to its successor $U_{i+1}$, $i=1,\cdots,(n-1)$. The participant $U_n$ sends signcryption message $\psi_n$ to $U_1$. This process is presented in the following **Fig. 1**.

$$U_1 \xrightarrow{\psi_1} U_2 \xrightarrow{\psi_2} U_3 \xrightarrow{\psi_3} U_4 \cdots\cdots \longrightarrow U_{n-1} \xrightarrow{\psi_{n-1}} U_n \xrightarrow{\psi_n} U_1$$

**Fig. 1**: Signcryption message sending process

The above process can be simultaneous and parallel.

**Round 2.** From the signcryption messages, the participants perform the following steps:

(1) The successor $U_{i+1}$, $i=1,\cdots,(n-1)$, verify the validity of the signcryption message $\psi_i$ from $U_i$ ($U_1$ verifies $\psi_n$);

(2) The participants $U_i$, $i=2,\cdots,n$, compute the secret $k_{i-1,i}$ ($U_1$ computes $k_{n,1}$), which is the session secret computed by a sender $U_{i-1}$ and its successor $U_i$ in the signcryption algorithm. We can present this via the following **Fig. 2**.

$$U_1 \xleftrightarrow{k_{1,2}} U_2 \xleftrightarrow{k_{2,3}} U_3 \xleftrightarrow{k_{3,4}} U_4 \cdots\cdots \longleftrightarrow U_{n-1} \xleftrightarrow{k_{n-1,n}} U_n \xleftrightarrow{k_{n,1}} U_1$$

**Fig. 2**: Shared secret between a participant and its successor

(3) The participants $U_i$, $i=2,\cdots,n$, compute and broadcast $X_i$, where $X_i=H_1(k_{i-1,i},ID_{\text{session}})\oplus H_1(k_{i,i+1},ID_{\text{session}})$. Note that the first participant $U_1$ computes and broadcasts $X_1=H_1(k_{n,1},ID_{\text{session}})\oplus H_1(k_{1,2},ID_{\text{session}})$. Here, $H_1(\cdot)$ is a one-way hash function and $ID_{\text{session}}$ is the public ephemeral information that consists of participants' identities and a nonce, aiming to make the protocol secure against known-key attacks.

Finally, with secret $k_{i,i+1}$, the participant $U_i$ $(i=1,\cdots,n)$ compute $B_i$ and further get all $B_j$ $(j=1,\cdots,n)$ as the following **Fig. 3**:

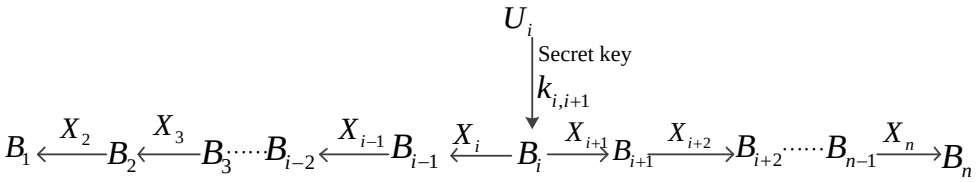


**Fig. 3**: Participant $U_i$ obtains all secrets by using its secret key

Then, the participant $U_i$ $(i=1,\cdots,n)$ compares $B_{i-1}$ and $H_1(k_{i-1,i}, ID_{\text{session}})$. If $B_{i-1} = H_1(k_{i-1,i}, ID_{\text{session}})$, each user computes the group session key $K_i = H_1(B_1 \| B_2 \| \cdots \| B_n)$. Obviously, $K_1 = K_2 = \cdots = K_n$. What this means is that all the participants agree on a common group session key.

**A member revocation:** Assume that a participant $U_x$ $(1 < x < n)$ leaves the group. Then group members change the group size into $(n-1)$. The participants $U_{x-1}$ and $U_{x+1}$ respectively remove the shared values $k_{x-1,x}$ and $k_{x,x+1}$ with $U_x$. The participant $U_{x+1}$ becomes the new successor of participant $U_{x-1}$. Aiming to update group key, the participant $U_{x-1}$ needs to send new signcryption message $\psi'_{x-1}$ to its new successor $U_{x+1}$. Then, the participant $U_{x+1}$ verifies the validity of the signcryption message $\psi'_{x-1}$ and computes the secret $k_{x-1,x+1}$ which is a new shared secret between $U_{x-1}$ and its new successor $U_{x+1}$. Each entity $U_j$ that follows $U_x$ changes their index to $(j-1)$. Then, from Step (3) of Round 2, all the $(n-1)$ participants implement the above protocol to get a new group session key.

**A new member join:** Assume that a new entity joins the group of which size is $n$. Then, the new participant $U_{n+1}$, becomes the successor of participant $U_n$ and the participant $U_1$ becomes the successor of participant $U_{n+1}$. The participant $U_n$ sends signcryption message $\psi_n'$ to its new successor $U_{n+1}$ while $U_{n+1}$ sends signcryption message $\psi_{n+1}$ to $U_1$. From the signcryption message $\psi_n'$, the new participant $U_{n+1}$ verifies the validity of the signcryption message and computes the secret $k_{n,n+1}$ which is the new shared secret between $U_n$ and its new successor $U_{n+1}$. The first participant $U_1$ updates its secret with $k_{n+1,1}$. Then, from step (3) of round 2, the participants in the group implement the above protocol to get a new group session key.

## 6. A Concrete Construction from Signcryption Scheme of Dent

This section will present a concrete GKA protocol from the signcryption scheme given by Dent [8]. First, let $(G, P, q)$ be the system parameters, where $G$ is a cyclic additive group with a prime order $q$ and $P$ is an arbitrary generator of $G$. In reality, $G$ is cyclic additive group $E(\text{GF}(q))$ on elliptic curve $E$. We assume that $E$ is a supersingular elliptic curve. The discrete logarithm problems (DLP) in $G$ are intractable. The protocol includes the following two rounds.

**Round 1.** Let $U = \{U_1, U_2, \cdots, U_n\}$ be the set of protocol participants. We assume that $x_i \in [1, \cdots, q-1]$ is the secret key of the participant $U_i$ with identity $ID_i$ $(i = 1, \cdots, n)$, and $Y_i = x_i P \in G$ is the corresponding public key. The participants $U_i$ $(i = 1, \cdots, n)$ perform the

following steps:

(1) Randomly choose $C_i \in G$;

(2) Compute $k_{i,i+1} = H_2(ID_i, ID_{i+1}, x_i Y_{i+1} + C_i)$;

(3) Send signcryption message $\psi_i = (ID_i, C_i)$ to its successor $U_{i+1}$, $i = 1, \cdots, (n-1)$. The participant $U_n$ sends signcryption message $\psi_n = (ID_n, C_n)$ to its successor $U_1$. This process is synchronous and parallel.

**Round 2.** Using the signcryption message $\psi_{i-1}$ from $U_{i-1}$, The participants $U_i$, $i = 2, \cdots, n$, performs the following steps:

(1) Compute the secret $k_{i-1,i} = H_2(ID_{i-1}, ID_i, x_i Y_{i-1} + C_{i-1})$ which is a shared secret computed by a sender $U_{i-1}$ and its successor $U_i$ from signcryption algorithm. The first participant $U_1$ computes $k_{n,1} = H_2(ID_n, ID_1, x_1 Y_n + C_n)$.

(2) Compute and broadcast $X_i = H_1(k_{i-1,i}, ID_{\text{session}}) \oplus H_1(k_{i,i+1}, ID_{\text{session}})$. The first participant $U_1$ computes $X_1 = H_1(k_{n,1}, ID_{\text{session}}) \oplus H_1(k_{1,2}, ID_{\text{session}})$.

Finally, each participant $U_i$ $(i = 1, \cdots, n)$ uses the shared secret $k_{i,i+1}$ ($U_n$ uses $k_{n,1}$) to get all $B_j$ $(j = 1, \cdots, n)$ as the following algorithms:

When $j \geq i$,

$$B_i = H_1(k_{i.i+1}, ID_{\text{session}});$$
$$B_{i+1} = X_{i+1} \oplus B_i;$$
$$B_{i+2} = X_{i+2} \oplus B_{i+1};$$
$$\cdots\cdots;$$
$$B_n = X_n \oplus B_{n-1}.$$

When $j < i$,

$$B_{i-1} = X_i \oplus B_i;$$
$$B_{i-2} = X_{i-1} \oplus B_{i-1};$$
$$\cdots\cdots;$$
$$B_1 = X_2 \oplus B_2.$$

Here, $ID_{\text{session}}$ is the public ephemeral information that consists of participants' identities and a nonce, aiming to make the protocol secure against known-key attacks.

Next, each participant $U_i$ $(i = 1, \cdots, n)$ verifies if $X_1 \oplus X_2 \oplus X_3 \oplus \cdots \oplus X_{n-1} \oplus X_n$ equals to zero. If not, output an error symbol $\perp$ and abort. Thus, each participant can find false broadcast messages by this verifying. If it does, the participant $U_i$ computes the session key $K_i = H_1(B_1 \| B_2 \| \cdots \| B_n)$. This will be the common group session key agreed by all participants.

## 7. Security Consideration

### 7.1 Security Notions

A signcryption group key agreement SGKA is specified by four polynomial-time algorithms: common-key-gen, participant-key-gen, encapsulation and decapsulation.

**Common-key-gen:** It is a probabilistic polynomial time (PPT) algorithm that takes the security parameter as input and outputs the common/public parameters used in the scheme.

These parameters include description of the underlying groups and hash functions used.

**Participant-key-gen:** For the security notions defined in this paper we distinguish the algorithm participant-key-gen as sender-key-gen and receiver-key-gen respectively. Here a sender is $U_i$ and the corresponding receiver is its successor $U_{i+1}$, $i=1,\cdots,n$. Sender-key-gen is a PPT algorithm that takes the common/public parameters as input and outputs the private-public key pair $(sk_s, pk_s)$ of the sender. Receiver-key-gen is a PPT algorithm that takes the common/public parameters as input and outputs the private-public key pair $(sk_r, pk_r)$ of the receiver.

**Encapsulation:** This is a PPT algorithm that takes the common/public parameters, a sender's private key $sk_s$ and a receiver's $pk_r$ as input. It returns a session key $\Re$ and its encapsulation $\psi$.

**Decapsulation:** It is a deterministic polynomial-time algorithm that takes the common/public parameters, a sender's public key $pk_s$, a receiver's private key $sk_r$ and an encapsulation $\psi$ as input. It outputs either a session key $\Re$ or an error symbol $\perp$.

In the security model for a signcryption key agreement SKA, the adversary is given the power to obtain encapsulations of a sender created for a receiver through a flexible encapsulation oracle (FEO) [23]. The adversary is also given access to a flexible decapsulation oracle (FDO) [23] that decapsulates a given encapsulation created for a receiver by a sender. The set of participants is $\{U_1,U_2,\cdots,U_n\}$ and the participant $U_{i+1}$ is the successor of the participant $U_i$. The participant $U_i$ is the sender and its successor $U_{i+1}$ is the corresponding receiver. Let $(x_i, Y_i)$ be the private-public key pair used by a sender $U_i$ for encapsulation and $(x_{i+1}, Y_{i+1})$ be the private-public key pair used by its successor $U_{i+1}$ for decapsulation. Here, $i=1,\cdots,n$.

The challenger initially fixes the set of participants $\{U_1,U_2,\cdots,U_n\}$ and their key pairs. The adversary is given all the public keys of the participants initially so that it can choose the public keys from the given group when accessing the oracles. The behavior of a sender's FEO and a receiver's FDO is described as follows.

**FEO**: On receiving $pk_r$, FEO returns a pair $(\Re, \psi)$, where $\psi$ is an encapsulation of $\Re$ generated using $sk_s$ and $pk_r$. The adversary may choose $Y_{i+1}$ as the receiver's public key, i.e., $pk_r = Y_{i+1}$, and then $\Re = k_{i,i+1}$, $\psi = \psi_i$.

**FDO**: On receiving $\psi$, FDO returns a session key $\Re$ or an error symbol $\perp$ after performing decapsulation on $\psi$ using $sk_r$ and $pk_s$. The adversary may choose $Y_i$ as the sender's public key, i.e., $pk_s = Y_i$, and then $\psi = \psi_i$, $\Re = k_{i,i+1}$.

Here the participant $U_{i+1}$ is the successor of the participant $U_i$ $(i=1,\cdots,n)$.

**Insider confidentiality**. An insider adversary $A^{CCA}$ against confidentiality of SKA is assumed to have knowledge of all key pairs except the private key of the receiver $U_{i+1}$ used for decapsulation. The goal of $A^{CCA}$ is to break the confidentiality of encapsulations created for $U_{i+1}$ by $U_i$. It is given access only to FDO as the oracle FEO can be simulated with the knowledge of $U_i$'s private key used for encapsulation. We call this notion as security FDO-IND-CCA2.

Challenge Phase: After adaptively asking the FDO queries, $A^{CCA}$ outputs a public key $pk_s'$. The challenger generates a valid session key, encapsulation pair $(\Re_0, \psi^*)$ using the private key $sk_s'$ corresponding to $pk_s'$ and $U_{i+1}$'s public key $Y_{i+1}$. It selects a key $\Re_1$ randomly from the session key distribution. It then chooses $b \in_R \{0,1\}$ and gives $(\Re_b, \psi^*)$ as the challenge.

$A^{CCA}$ can continue its execution except asking the FDO$(pk_s', \psi^*)$ query that trivially decides the guess. However, an FDO query on $\psi^*$ using a public key $pk_s \neq pk_s'$ is still allowed.

Guess Phase: Finally, $A^{CCA}$ outputs a bit $b'$ and wins the game if $b' = b$. The advantage of $A^{CCA}$ in winning the FDO-IND-CCA2 game is

$$\mathrm{Adv}_{A^{CCA},SKA} = 2 \cdot \Pr[b' = b] - 1.$$

**Outsider confidentiality**. For outsider confidentiality the adversary is assumed to know all the private keys except $U_i$'s private key used for encapsulation and $U_{i+1}$'s private key used for decapsulation. The goal of the adversary in this notion is to break the confidentiality of encapsulations created by $U_i$ for $U_{i+1}$. The adversary must be given access to both FEO and FDO. We call this notion FEO/FDO-IND-CCA2. After adaptively asking the FEO and FDO queries, the challenge and guess phases are carried on as described above. The advantage of an adversary in winning the FEO/FDO-IND-CCA2 game is also defined in the same way as above.

**Outsider unforgeability.** An outsider adversary $A^{CMA}$ against unforgeability of SKA is assumed to know all private keys except $U_i$'s private key used for encapsulation and $U_{i+1}$'s private used for decapsulation. The goal of $A^{CMA}$ is to forge a valid session key and encapsulation pair $(\Re^*, \psi^*)$ such that $\psi^*$ is an encapsulation of $\Re^*$ created by $U_i$ for $U_{i+1}$. It is given access to both FEO and FDO.

After querying FEO and FDO adaptively, $A^{CMA}$ produces a forgery $(\Re^*, \psi^*)$. It wins the game if decapsulation$(y_i, x_{i+1}, \psi^*) = \Re^* \neq \perp$. We call this notion FEO/FDO-sUF-CMA for strong unforgeability against outsider attacks. The advantage of $A^{CMA}$ in winning the FEO/FDO-sUF-CMA game is the probability of $A^{CMA}$ outputting a valid trivial restriction for $(\Re^*, \psi^*)$. What this means is that $(\Re^*, \psi^*)$ was never an output of FEO.

**Insider unforgeability.** An insider adversary against unforgeability of SKA is assumed to know all the private keys except $U_i$'s private key used for encapsulation. The goal of the adversary in this notion is to forge a valid encapsulation created by $U_i$ for $U_{i+1}$. It is given access only to FEO as the key for decapsulation are known to the adversary. We call this notion FEO-sUF-CMA for strong unforgeability against insider attacks. After adaptively querying the FEO, the adversary outputs a forgery $(\Re^*, \psi^*, pk_r^*)$. It wins FEO-sUF-CMA game if decapsulation$(y_i, sk_r^*, \psi^*) = \Re^* \neq \perp$. The advantage of the adversary in winning the FEO-sUF-CMA game is the probability of outputting a valid trivial restriction for $(\Re^*, \psi^*, pk_r^*)$. What this means is that $(\Re^*, \psi^*)$ was never an output of FEO.

## 7.2 Security Analysis

The proposed group key agreement scheme from signcryption includes two processes: to construct a recursive key chain from signcryption and to extract group key from the recursive chain. Actually, computing group key from recursive chain is only a series of additions modulo 2. Accordingly, the security proof is focused on the process of constructing a recursive chain from signcryption, and its security follows the following three lemmas. Here, Lemma 2 gives the authentication of the proposed scheme, while Lemma 3 combined with Claim 1 presents the confidentiality of it. Both are outsider security. Lemma 2 gives assurance to the recipient $U_{i+1}$ that the key indeed comes from the intended sender $U_i$, and Lemma 3 gives assurance to a group member $U_i$ that the key is available only to the intended recipient, i.e., its successor $U_{i+1}$. The proof of the Lemmas is provided in the appendix.

**Claim 1:** Given $B_i = H_1(k_{i,i+1}, ID_{session})$, it is difficult to find $k_{i,i+1}$, the session secret between a sender $U_i$ and its successor $U_{i+1}$, where $i = 1, \cdots, n$. This is derived from the assumption that $H_1(\cdot)$ is a one-way hash function, i.e., given a hash value $h$, it is difficult to find any message $m$ such that $h = H_1(m)$.

**Lemma 2**: The signcryption key agreement SKA is secure in the outsider unforgeability notion in the random oracle model assuming hardness of the Gap Diffie Hellman (GDH) problem in the group *G*.

**Lemma 3**: The signcryption key agreement SKA is secure in the outsider confidentiality notion in the random oracle model assuming hardness of the GDH problem in the group *G*.

## 8. Conclusion

In this paper, we suggest a primary approach for construction of group key agreement protocol from signcryption. This approach introduces the desirable property of mutual authentication. We have proved that our scheme is secure in the outsider unforgeability notion and the outsider confidentiality notion assuming hardness of the GDH problem. The following Table 1 depicts a comparison with some typical schemes in terms of security and efficiency.

Here, "Rounds" is the total number of rounds and a round means that each party sends one message and can broadcast simultaneously. The entries "Au.", "FS." and "MA." respectively indicate the desirable properties of authentication, forward secrecy and mutual authentication. The terms "Ucasts" and "Bcasts" respectively denote the total number of unicast messages and broadcast messages of all members. Similarly, "U. Size" and "B. Size" respectively represent the cumulative unicast message size and the cumulative broadcast message size of each member. The abbreviation "Comp. costs" means computational costs of each member measured by "Exp.", "ModM." or "SM." respectively representing modular exponentiations, modular multiplications and scalar multiplications of ECC (Elliptic Curve Cryptography). In addition, we use "Sig." and "Ver." to denote signing and verifying operations of digital signatures.

**Table 1.** Comparison with some typical schemes

| | Y. Kim et al.[14] | M. Steiner [15] IKA.2 | Katz, J. et al. [17] | Gorantla, M. et al. [18] | Ours |
|---|---|---|---|---|---|
| Rounds | 2 | *n*+1 | 3 | 1 | 2 |
| Au. | No | No | Yes | Yes | Yes |
| FS. | No | No | Yes | No | No |
| MA. | No | No | No | No | Yes |
| Ucasts. | 0 | 2*n*-3 | 0 | 0 | *n* |

| | | | | | |
|---|---|---|---|---|---|
| Bcasts. | $n$+1 | 2 | 2$n$ | $n$ | $n$ |
| U. size | 0 | 2$n$-3 | 0 | 0 | 1 |
| B. size | (2$n$-1) for $i$=1; ($n$-$i$+1) for others | $n$+1 | 2 | $n$+1 | 1 |
| Comp. costs | (3$n$-2) Exp. | (2$n$+1) Exp. | 3Exp.+($n\log_2 n$)ModM.+ 2 Sig.+$n$ Ver. | 2$n$ Exp. | 2 SM. |

It can observed from the table that our protocol is the most efficient GKA scheme with authentication, since it has the smallest message sizes as well as the lowest computational costs for each member. Although the protocol of Gorantla, M. et al. [19] needs only one round, it is not mutual authentication, that is, the recipient is not assured that the contributions come from the intended sender. Our protocol needs two rounds of communication, whereas the application of signcryption enables it to give assurance to the sender that the key is available only to the intended recipient, and vice versa.

It still remains an open problem to derive a one-round group key agreement protocol from signcryption. After all, it is inconvenient to require all the parties to stay online concurrently to implement a two-round protocol.

## References


[1] G. Ateniese, M. Steiner and G. Tsudik, "New multi-party authentication services and key agreement protocols", *IEEE Selected Areas in Communications*, vol. 18, no. 4, pp. 628-639, 2000. Article (CrossRef Link)

[2] M. Bellare and P. Rogaway, "Entity authentication and key distribution", in *Proc. of Crypto '93*, LNCS 773, pp. 232-249, Springer, 1993. Article (CrossRef Link)

[3] S. Blake-Wilson and A. Menezes, "Authenticated Diffie-Hellman key agreement protocols", in *Proc. of SAC '98*, LNCS 1556, pp. 339-361, Springer, 1998. Article (CrossRef Link)

[4] E. Bresson, O. Chevassut and D. Pointcheval, "Provably authenticated group Diffie-Hellman key exchange – the dynamic case", in *Proc. of Asiacrypt '01*, pp. 290-309, Springer-Verlag, 2001. Article (CrossRef Link)

[5] Y. Zheng, "Digital Signcryption or How to Achieve Cost (Signature & Encryption) << Cost (Signature) + Cost (Encryption)", in *Proc. of CRYPTO 1997,* LNCS 1294, pp. 165-179, Springer, Heidelberg, 1997. Article (CrossRef Link)

[6] Y. Zheng, "Shortened Digital Signature, Signcryption and Compact and Unforgeable Key Agreement Schemes", *Technical report, A submission to IEEE P1363 Standard Specifications for Public Key Cryptography,* 1998. Article (CrossRef Link)

[7] M. Choudary Gorantla, Colin Boyd, Juan Manuel Gonzalez Nieto, "On the Connection Between Signcryption and One-pass Key Establishment", In Galbraith, Steven (Ed.) *Cryptography and Coding: 11th IMA International Conference*, 18-20th, 2007. Article (CrossRef Link)

[8] A. Dent, "Hybrid Signcryption Schemes with Outsider Security", in *Proc. of ISC 2005,* LNCS 3650, pp. 203-217, Springer, Heidelberg, 2005. Article (CrossRef Link)

[9] W. Diffie and M. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory,* vol. 22, no. 6, pp. 644-654, 1976. Article (CrossRef Link)

[10] A. Joux, "A One Round Protocol for Tripartite Diffie-Hellman", *Journal of Cryptology*, vol. 17, no. 4, pp. 263-276, 2004. Article (CrossRef Link)

[11] D. Boneh, and A. Silverberg, "Applications of Multilinear Forms to Cryptography", *Contemporary Mathematics 324*, pp.71-90, 2003. Article (CrossRef Link)

[12] M. Burmester and Y.G. Desmedt, "A Secure and Efficient Conference Key Distribution System", in *Proc. of EUROCRYPT 1994,* LNCS 950, pp. 275-286, Springer, Heidelberg, 1995. Article (CrossRef Link)

[13] E. Bresson, O. Chevassut, and D. Pointcheval, "Dynamic Group Diffie-Hellman Key Exchange

under Standard Assumptions", in *Proc. of Eurocrypt 2002*, LNCS 2332, pp.321-336, 2002. Article (CrossRef Link)
[14] Y. Kim, A. Perrig and G. Tsudik, "Communication-Efficient Group Key Agreement. Information Systems Security", in Proc. of the 17th International Information Security Conference IFIP SEC'01, 2001. Article (CrossRef Link)
[15] M. Steiner, "Secure group key agreement", Ph.D. Thesis, Naturwissenschaftlich-Technischen Fakultät I der Universität des Saarlandes, March 2002. Article (CrossRef Link)
[16] Qianhong Wu, Yi Mu, Willy Susilo, Bo Qin, and Josep Domingo-Ferrer, "Asymmetric Group Key Agreement", in *Proc. of EUROCRYPT 2009*, LNCS 5479, pp. 153-170, 2009. Article (CrossRef Link)
[17] Katz, J., Yung, M., "Scalable Protocols for Authenticated Group Key Exchange", in Proc. of CRYPTO 2003, LNCS 2729, pp. 110-125, Springer, Heidelberg, 2003. Article (CrossRef Link)
[18] Gorantla, M., Boyd, C., González Nieto, J. and Manulis, M, "Generic one round group key exchange in the standard model", in Proc. of Information, Security and Cryptology-ICISC 2009, LNCS 5984, pp. 1-15, Springer-Verlag Berlin Heidelberg, 2010. Article (CrossRef Link)
[19] Smart, N.P., "Efficient Key Encapsulation to Multiple Parties", in Proc. of the Fourth Conference on Security in Communication Networks (SCN'04), LNCS 3352, pp. 208-219, Springer, Heidelberg, 2005. Article (CrossRef Link)
[20] A. Dent, "Hybrid Signcryption Schemes with Insider Security", in *Proc. of ACISP 2005*, LNCS 3574, pp. 253-266, Springer, Heidelberg, 2005. Article (CrossRef Link)
[21] Tor E. Bjørstad and Alexander W. Dent, "Building Better Signcryption schemes with Tag-KEMs", in *Proc. of Public Key Cryptography (PKC 2006),* LNCS 3958, pp. 491-507, Springer-Verlag, 2006. Article (CrossRef Link)
[22] Alexander W. Dent, "Key establishment using signcryption techniques", In: Dent AW, Zheng Y editors, *Practical Signcryption,* pp. 217-240, Springer-Verlag, 2010. Article (CrossRef Link)
[23] Yoshida M., Fujiwara T., "On the Security of Tag-KEM for Signcryption", *Electronic Notes in Theoretical Computer Science*, vol. 171, no. 1, pp. 83-91, Elsevier, 2007. Article (CrossRef Link)

## Appendix: Proof of Lemmas.

***Proof of Lemma 2*:** In what follows we will show that if there exists a polynomial time adversary $A^{CMA}$ against the unforgeability of the SKA with non-negligible advantage, then we can construct a polynomial time algorithm $A^{GDH}$ which solves the Gap Diffie-Hellman (GDH) problem with the same advantage as $A^{CMA}$. This action can fully prove Lemma 2. Here we make the GDH problem solve the Computational Diffie-Hellman (CDH) with the assistance of a decisional Diffie-Hellman oracle $O_{DDH}$.

Let $A=aP$, $B=bP$. The problem instance $A, B$ is given to $A^{GDH}$ aiming to find the value $abP$. The process shown below is that $A^{GDH}$ runs $A^{CMA}$ and simulates the answers to the queries made by $A^{CMA}$.

- **Hash:** For Hash queries, Initially $A^{GDH}$ has an empty list $L_{H_2}$ that stores the past returned hash values. On input $(ID_i, ID_{i+1}, X_i)$, $A^{GDH}$ first checks if there is an existing entry $(ID_i, ID_{i+1}, X_i, k_{i,i+1})$ for some $k_{i,i+1}$ in $L_{H_2}$. If so, it returns this $k_{i,i+1}$; otherwise it accesses the global encapsulation list $L_\varepsilon$ and acts as the following:

**if** $(ID_i, ID_{i+1}, C_i, k_{i,i+1}) \in L_\varepsilon$ for some $k_{i,i+1}$ and $C_i$ values **then**
compute $T_i = X_i - C_i$
**if** $O_{DDH}(Y_i, Y_{i+1}, T_i) = \text{True}$ **then**
**if** $Y_i = A$ and $Y_{i+1} = B$ **then**

return $T_i$ as solution to the GDH challenger and exit
**else**
return $k_{i,i+1}$ to $\mathrm{A}^{\mathrm{CMA}}$
update $L_{\mathrm{H}_2} = L_{\mathrm{H}_2} \| (ID_i, ID_{i+1}, X_i, k_{i,i+1})$
**end**
**else**
select $k_{i,i+1}$ randomly from the key distribution and return it to $\mathrm{A}^{\mathrm{CMA}}$
update $L_{\mathrm{H}_2} = L_{\mathrm{H}_2} \| (ID_i, ID_{i+1}, X_i, k_{i,i+1})$
**end**
**else**
select $k_{i,i+1}$ randomly from the key distribution and return it to $\mathrm{A}^{\mathrm{CMA}}$
update $L_{\mathrm{H}_2} = L_{\mathrm{H}_2} \| (ID_i, ID_{i+1}, X_i, k_{i,i+1})$
**end**

- **FEO:** $\mathrm{A}^{\mathrm{GDH}}$ initially starts with an empty encapsulation list $L_\varepsilon$. On input $(Y_i, Y_{i+1})$ $\mathrm{A}^{\mathrm{GDH}}$ first selects $C_i \in G$. It then checks each entry $(ID_i, ID_{i+1}, X_i, k_{i,i+1})$ in $L_{\mathrm{H}_2}$ to see if $O_{\mathrm{DDH}}(Y_i, Y_{i+1}, T_i) = \mathrm{True}$ for the same $(Y_i, Y_{i+1})$ as the input to FEO. If so, it fetches the corresponding $k_{i,i+1}$ from $L_{\mathrm{H}_2}$, otherwise it selects $k_{i,i+1}$ randomly from the key distribution. It returns $(k_{i,i+1}, C_i)$ to $\mathrm{A}^{\mathrm{CMA}}$. Finally, $L_\varepsilon$ is updated to $L_\varepsilon = L_\varepsilon \| (ID_i, ID_{i+1}, C_i, k_{i,i+1})$.
- **FDO:** On input $(Y_i, Y_{i+1}, C_i)$, $\mathrm{A}^{\mathrm{GDH}}$ first checks if there is an entry $(ID_i, ID_{i+1}, C_i, k_{i,i+1}) \in L_\varepsilon$. If so, it returns the corresponding key $k_{i,i+1}$. Otherwise, it acts as the following:

**if** $(ID_i, ID_{i+1}, X_i, k_{i,i+1}) \in L_{H_2}$ for some $X_i$ **then**
compute $T_i = X_i - C_i$
**if** $O_{\mathrm{DDH}}(Y_i, Y_{i+1}, T_i) = \mathrm{True}$ **then**
**if** $Y_i = P_A$ and $Y_{i+1} = P_B$ **then**
return $T_i$ as solution to the GDH challenger and exit
**else**
take corresponding $k_{i,i+1}$ from $L_{\mathrm{H}_2}$ and return it to $\mathrm{A}^{\mathrm{CMA}}$
update $L_\varepsilon = L_\varepsilon \| (ID_i, ID_{i+1}, C_i, k_{i,i+1})$
**end**
**else**
select $k_{i,i+1}$ randomly from the key distribution and return it to $\mathrm{A}^{\mathrm{CMA}}$
update $L_\varepsilon = L_\varepsilon \| (ID_i, ID_{i+1}, C_i, k_{i,i+1})$
**end**
**else**
select $k_{i,i+1}$ randomly from the key distribution and return it to $\mathrm{A}^{\mathrm{CMA}}$
update $L_\varepsilon = L_\varepsilon \| (ID_i, ID_{i+1}, C_i, k_{i,i+1})$
**end**

**Answering the GDH challenger:** Eventually, $A^{CMA}$ outputs a forgery $(k'_{i,i+1}, C'_i)$ as an encapsulation created by $U_i$ for $U_{i+1}$. For the forgery to be valid under the outsider unforgeabilioty notion FEO/FDO-sUF-CMA, $C'_i$ must be a valid encapsulation of $k'_{i,i+1}$. If $C'_i$ is a valid encapsulation of $k'_{i,i+1}$ then $A^{CMA}$ must have queried the Hash with corresponding keying material, in which case $A^{GDH}$ would have answered the GDH challenger already. Hence, the advantage of $A^{GDH}$ to solve the GDH problem is the same as the advantage of $A^{CMA}$.

***Proof of Lemma 3:*** As defined in Section 7.1, for outsider confidentiality, the adversary $A^{CCA}$ is given all the private keys except $U_i$'s private key used for encapsulation and $U_{i+1}$'s private key used for decapsulation. The goal of $A^{CCA}$ is to break the confidentiality of encapsulations created by $U_i$ for $U_{i+1}$. In what follows we will show that if there exists a polynomial time adversary $A^{CCA}$ against the confidentiality of the SKA with non-negligible advantage, then we can construct a polynomial time algorithm $A^{GDH}$ which solves the Gap Diffie-Hellman (GDH) problem with the same advantage as $A^{CCA}$. This action can fully prove Lemma 3.

Let $A = aP$, $B = bP$. The problem instance $A, B$ is given to $A^{GDH}$ aiming to find the value $abP$. The process shown below is that $A^{GDH}$ runs $A^{CCA}$ and simulates the answers to the queries defined in the same way as above.

**Answering the GDH challenger:** After adaptively asking the FEO and FDO queries, $A^{CCA}$ outputs a public key $Y'_i$. The challenger gives $(k^b_{i,i+1}, C_i')$ as the challenge, in which a $b \in_R \{0,1\}$. When $b = 0$, $k^0_{i,i+1}$ is a valid session key, i.e., encapsulation pair $(k^0_{i,i+1}, C_i')$ using the private key $x'_i$ and $U_{i+1}$'s public key $Y_{i+1}$; when $b = 1$, $k^1_{i,i+1}$ is randomly selected from the session key distribution.

Finally, $A^{CCA}$ outputs a bit $b'$ as its guess. Suppose $b' = 0$, then $A^{CCA}$ must have queried the Hash with corresponding keying material, in which case $A^{CCA}$ would have already answered the GDH challenger. Hence, the advantage of $A^{GDH}$ to solve the GDH problem is the same as the advantage of $A^{CCA}$.

**Dr. Xixiang Lv** studied in Xidian University, Peoples R China, from 1997 to 2007, and received her respective M.S. and Ph.D. in cryptography in 2004 and 2007. She is now an associate professor of Xidian University, Peoples R China. Her research interests lie in information security and wireless network security.

**Dr. Hui Li** received his BE from Fu Dan University, Peoples R China, in 1990, and his Ph.D. in Communication and Electronic Engineering from Xidian University, Peoples R China, in 1998. Prof. Li has published around 50 academic papers in the areas of information security and coding theory. His research interests include information security, coding theory and wireless network security.