# An Analysis of Group Key Agreement Schemes based on the Bellare-Rogaway Model in Multi-party Setting

**Meng-Hui Lim[1], Bok-Min Goi[2] and Sang Gon Lee[3]**
[1] School of Engineering, Yonsei University, Seoul, South Korea
[e-mail: menghui.lim@gmail.com]
[2] Faculty of Engineering & Science, Tunku Abdul Rahman University, Kuala Lumpur, Malaysia
[e-mail: goi.bokmin@gmail.com]
[3] Division of Computer and Information Engineering, Dongseo University, Busan, South Korea
[e-mail: nok60@gdsu.dongseo.ac.kr]
*Corresponding author: Sang Gon Lee

*Received December 20, 2010; revised February 8, 2011; accepted March 5, 2011; published April 29, 2011*

## Abstract

Group key agreement protocols derive a shared secret key for a group of users to ensure data confidentiality or/and integrity among the users in the subsequent communications. In this paper, we inspect two group key agreement schemes which have been proposed by Shi et al. and Zheng et al. in 2005 and 2007 respectively. Although both schemes were claimed to be secure in a heuristic way, we reveal several flaws using the Bellare-Rogaway security model extended to multi-party setting by Bresson et al. These flaws are found to be originated from inappropriate selection of key derivation function, inadvertent exclusion of partners' identities from the protocol specification and insufficient consideration in preserving known temporary information security and key freshness properties. Furthermore, we suggest and discuss proper countermeasures to address such flaws.

***Keywords:*** Group Key Agreement Protocol, Cryptanalysis

**DOI: 10.3837/tiis.2011.04.011**

## 1. Introduction

Group key agreement protocols allow $n \geq 2$ principals to agree upon a common secret key (session key) by having each principal to contribute an equal share of information via an insecure network (e.g. the Internet). This is a popular way of establishing group secrets since it is appropriate for dynamic peer groups. Unlike centralized and distributed group key management methods which relies on a single entity in generating and distributing the secret, group key agreement approach avoids single point(s) of trust problems and failures which could be difficult to handle.

The main security goal of group key agreement schemes is to prevent any unauthorized parties from gaining access to the secret session key although they may have full control over the public network. Despite being fundamental in many secure electronic applications nowadays such as video conferencing and database replications, designing a secure group key agreement protocol can be notoriously hard. The difficulty of obtaining high assurance in the security of existing protocols can be seen from the examples of errors found in many such protocols years after they were published.

In 1998, Blake-Wilson and Menezes [6][7] defined a number of vital security attributes which can effectively be used in scrutinizing the security of a key agreement protocol. Such security attributes include *known session key security*, *perfect forward secrecy*, *key compromise impersonation resilience*, *unknown key share resilience* and *key control resilience*.

- **Known session key security.** A protocol is considered to be *known session key secure* if it remains achieving its goal in the face of an adversary who has learned some previous session keys.
- **(Perfect) forward secrecy.** A protocol enjoys *forward secrecy* if the secrecy of the previous session keys is not affected when the long term private keys of one or more entities are compromised. *Perfect forward secrecy* refers to the scenario when the long term private keys of all the participating entities are compromised.
- **Key-Compromise Impersonation Resilience.** Suppose that a party, say *C*'s long term private key has been disclosed. Obviously an adversary who knows this value can now impersonate *C* since it is precisely the value which identifies *C*. We say that a protocol is *key-compromise impersonation resilient* if this loss will not enable an adversary to masquerade as other legitimate entities to *C* as well or obtain other entities secret key.
- **Unknown Key-Share Resilience.** In an unknown key-share attack, an adversary convinces a group of entities that they share a key with the adversary whereas in fact, the key is shared between the group and another party. This situation can be exploited in a number of ways by the adversary when the key is subsequently used to provide encryption or integrity.
- **Key Control Resilience.** It should not be possible for any of the participants (or an adversary) to compel the session key to a preselected value or predict the value of the session key.

Group key establishment has a long history in the literature. The first known proposal goes back to 1982 where Ingemarsson et al. [21] presented a conference key distribution system which admits any group of stations to share the same encryption and decryption keys. In 1998, Ateniese et al. [1] further identified several additional desirable security goals of authenticated

group Diffie-Hellman key exchange and subsequently put forward a series of protocols, namely GDH.1, GDH.2, and GDH.3, in which the latter two served as the basis of CLIQUES family. Unfortunately, they have later been proven flawed by Pereira and Quisquater [26]. Another group key agreement protocol proposed by Bresson et al. [10] for low-power mobile devices has been illustrated insecure by Nam et al. [25] due to the absence of implicit key authentication, forward secrecy and known key security. Boyd and Gonzalez Nieto [9] have put forward a conference key agreement protocol (where the number of participants involved is exactly three) with a claimed proof of security. However, the proof model as well as the security of such a protocol has both been invalidated by Choo et al. in [19].

Generally, formal analysis of a protocol is essential in assuring the practitioners the security of a protocol. Group key agreement protocols received the first formal treatment in 2002 where Bresson et al. [11][12] presented a rigorous framework for modeling group key establishments based on [2][3][4][5][14]. Recently, Bohli et al. [8] further extended Bresson et al.'s work to cover attacks from malicious protocol participants and formally defined the concept of integrity and strong entity authentication.

In this work, we advocate the importance of formal analysis of group key agreement protocol and we hope that this would enable protocol designers to adopt a formal approach and to avoid similar attacks in the future proposals. Particularly, we examine two heuristically-analyzed group key establishment protocols proposed by Shi et al. and Zheng et al. respectively in [27] and [28] using the Bresson et al.'s security framework, and point out several attacks (unknown key share attack, key replicating attack and triangle attack) as well as some essential security properties which both schemes do not satisfy, although a minor flaw of Shi et al.'s scheme has been heuristically illustrated by Zhou et al. in [29].

Our main contributions lie in the three following aspects:

- demonstrate the unpublished attacks on two protocols based on formal analysis,
- suggest appropriate countermeasures to address the flaws, and
- promote formal approach when designing protocols for cryptographic purpose.

The structure of this paper is organized as follows. In the next section, we will illustrate some basic properties of bilinear pairings and several underlying assumptions which the discussed schemes are based on. In section 3, we will provide an overview of Bresson et al.'s security model. In section 4 and section 5, we will revisit Zheng et al.'s and Shi et al.'s group key agreement protocols accordingly, highlight their demerits and discuss appropriate countermeasures. Section 6 concludes this paper.

## 2. Preliminaries

Let $X_1$ be an additive group and $X_2$ be a multiplicative group of a large prime order $q$. With $P \in X_1$, we define $\hat{e}: X_1 \times X_1 \to X_2$ as a bilinear pairing with the following properties:

- **Bilinearity**: $\hat{e}(aP, bP) = \hat{e}(P, P)^{ab} = \hat{e}(abP, P)$ for any $a, b \xleftarrow{R} Z_q^*$.
- **Non-degeneracy**: $\hat{e}(P, P) \neq 1$.
- **Computability**: There exists an efficient algorithm to compute $\hat{e}(P, P)$.

Since most of the cryptographic protocols based their security on conjectures (i.e. Bilinear Diffie-Hellman (BDH) conjecture, Computation Diffie-Hellman (CDH) conjecture, etc), the

group key agreement protocols presented in this paper will not be exceptional as they also rest upon the following conjectures before their security can be claimed.

Let $G$ be a parameter generator. Given system parameters $1^k$ as input, $G$ generates two cyclic groups $(X_1, X_2)$ of prime order $q$ and a generator $P$ of $X_1$.

**Conjecture 1. Discrete Logarithm Conjecture (DL).** The advantage of an algorithm $A$ in solving the DL problem (given $\langle P, aP \rangle$, compute $a$) is defined by:

$$Adv_{G,A}(k) = \Pr\left[ A(q, X_1, P, aP) = a \mid \langle q, X_1, P \rangle \leftarrow G(1^k), a \xleftarrow{R} Z_q^* \right]$$

For any randomized polynomial time (in $k$) algorithm $A$, the advantage $Adv_{G,A}(k)$ is negligible.

**Conjecture 2. Computational Diffie-Hellman Conjecture (CDH).** The advantage of an algorithm $A$ in solving the CDH problem (given $\langle P, aP, bP \rangle$, compute $abP$) is defined by:

$$Adv_{G,A}(k) = \Pr\left[ A(q, X_1, P, aP, bP) = abP \mid \langle q, X_1, P \rangle \leftarrow G(1^k), a, b \xleftarrow{R} Z_q^* \right]$$

For any randomized polynomial time (in $k$) algorithm $A$, the advantage $Adv_{G,A}(k)$ is negligible.

**Conjecture 3. Bilinear Diffie-Hellman Conjecture (BDH).** The advantage of an algorithm $A$ in solving the BDH problem (given $\langle P, aP, bP, cP \rangle$, compute $\hat{e}(P,P)^{abc}$) is defined by:

$$Adv_{G,A}(k) = \Pr\left[ \begin{array}{l} A(q, X_1, X_2, P, aP, bP, cP) = \hat{e}(P,P)^{abc} \mid \\ \langle q, X_1, X_2, P \rangle \leftarrow G(1^k), a, b, c \xleftarrow{R} Z_q^* \end{array} \right]$$

For any randomized polynomial time (in $k$) algorithm $A$, the advantage $Adv_{G,A}(k)$ is negligible.

## 3. Informal Overview of the Bellare-Rogaway Security Model in Multi-Party Settings [4][12][16]

Let $T = \{U_1, U_2, ..., U_z\}$ be a non-empty set of protocol participants, where $z$ is a polynomial bound on the number of participants. Each participant $U \in \{U_1, U_1, ..., U_n\} \subset T$ involved in a protocol session is modelled as an oracle in which an oracle $\prod_U^s$ denotes the $s$-th instance of a protocol principal $U$ in a particular protocol run. Every oracle at any time exists in one of several possible states: Accept, Reject or $\perp$ (indicates no decision reached yet), holds a public session identifier $sid_U^s$ (usually formed by concatenation of exchanged messages), a set of partner identities $pid_U^s$ (including $U$ itself) and a session key $sk_U^s$ (if it has accepted). Besides, there also exists an adversary $A \not\subset T$ which is granted enormous power to control over the network by interacting with oracles through oracle queries. The queries effectively model $A$'s capabilities and capture several attractive security attributes of a key agreement protocol. These predefined queries are described as follows:

**Send** $\left(\prod_U^s, m\right)$ **query** allows *A* to send a message *m* of his choice to oracle $\prod_U^s$. The response is computed according to the protocol specification and returned to *A* subsequently. After the query is made, *A* would also be informed about the decision of the oracle whether to accept or to terminate. *A* can induce $\prod_U^s$ to initiate a new protocol run with participants of *A*'s choosing by sending $m = \{U_1, U_2, ..., U_n\}$ with $\{U_1, U_2, ..., U_n\} \in T$. As a result, the protocol run is indexed with a new *s* and $pid_U^s$ is initialized to $\{U_1, U_2, ..., U_n\} \cup \{U\}$. This means that *U* will be aiming at establishing a common session key with principals specified in *m*.

**Reveal** $(\prod_U^s)$ **query** that captures known key security of the protocol reveals $sk_U^s$ (if any) held by $\prod_U^s$ to *A*. An oracle is considered *opened* if it has been subject to a **Reveal** query.

**Corrupt(*U*) query** reveals *U*'s long term key to *A*. This query basically models insider attacks by *A* where *A* may control the behaviour of the corrupted *U* at will by using **Send** queries.

**Coin** $(\prod_U^s)$ **query** models known session-specific temporary information security of the protocol. This query is adopted from [16] and it was not originally included as one of the queries of Bellare-Rogaway model. This query allows *A* to reveal the random coin flips $r_U^s$ of the oracle $\prod_U^s$ and it can be issued at anytime in the middle of session *s* or even after the completion of session *s*. An oracle is said to be *controlled* if it has been subject to a **Coin** query.

**Test** $(\prod_U^s)$ **query** is the only oracle query that does not correspond to any of *A*'s abilities. When a **Test** query is made, for a random bit *b*, the session key held by the oracle is returned if *b* = 0 and otherwise if *b* = 1, a random *k*-bit string is returned, provided that the oracle must be *fresh*: It must have accepted, be uncorrupted before it accepts, not have been subject to a **Reveal** query, not have been issued both **Coin** and **Corrupt** queries, not have participated in a session with any revealed partner oracle (see Definition 1 below), not have engaged in a protocol execution with any corrupted partner oracle before it accepts, not have participated in a session with any controlled and corrupted partner oracle, and if it or any partner oracle is corrupted after it accepts, it must not have been opened. In order to capture the notion of security, *A* will try its best to distinguish $sk_U^s$ from the random string by guessing at *b*. If we define *GoodGuess* to be the event where *A* guesses correctly, then *A*'s advantage, denoted by $advantage^A(k)$ is defined as

$$advantage^A(k) = 2 \cdot \Pr[GoodGuess] - 1$$

**Definition 1. (Definition of Partnership)** Two oracles $\prod_{U_1}^{s_1}$ and $\prod_{U_2}^{s_2}$ are referred to as *partners* if they have both accepted, $sid_{U_1}^{s_1} = sid_{U_2}^{s_2}$ and $pid_{U_1}^{s_1} = pid_{U_2}^{s_2}$.

**Definition 2. (Definition of Security)** A key establishment protocol is considered to be secure if:

1. All uncorrupted oracles which have completed a matching session, always hold the same session key which is distributed randomly and uniformly on $\{0, 1\}^k$ in the presence of the benign adversary.
2. $advantage^A(k)$ is negligible.

The first condition ensures that a secure protocol does indeed distribute a key correctly under the existence of the adversary, provided that the involved entities behave correctly and the transmission between them is not tampered with. By fulfilling this completeness criterion, it means the protocol will complete as expected. Whereas for the second condition, which is also the ultimate goal of a secure protocol, the adversary should be impeded in obtaining any useful information about the session key held by a fresh oracle under any circumstances.

## 4. Zheng et al.'s Group Key Agreement Protocol

Zheng et al. [28] put forward a two-round group key agreement protocol in 2007 which was claimed to be secure against active adversaries in the random oracle model.

**System Setup.** $\{p, g, Z_p^*, H\}$ are selected and advertised as public system parameters where $p$ denotes a $l$-bit large prime, $g$ denotes a generator of the multiplicative group $Z_p^*$ and $H: \{0,1\}^* \to Z_p^*$ denotes a one-way collision-resistant hash function. Each member $U_i \in T$ obtains its long-term public key pair $\{x_i, y_i, CA(y_i)\}$ from the certification authority, where $x_i \in_R [1, p-2]$ and $y_i = g^{x_i} \bmod p$.

**Key Agreement.** Suppose that there exist $n$ members, $\{U_1, U_2, ..., U_n\} \subset T$, involving in an execution of initial key agreement protocol. A detailed overview of the exchanged messages is given in **Fig. 1**.

**Round 1:**

**Broadcast.** Each $U_i$ selects a random $k_i \in [1, p-1]$, computes $r_i = g^{k_i} \bmod p$ and broadcasts $(y_i, r_i)$.

**Computation.** Each $U_i$ waits until messages $(y_i, r_i)$ for all $U_j$ arrived, selects a subsession key $K_i \in_R Z_p^*$, computes $s_{ij} = r_j^{k_i} = g^{k_i k_j} \bmod p$ and $t_{ij} = s_{ij} K_i \bmod p$.

**Round 2:**

**Computation.** Each $U_i$ selects a random $\alpha_i \in_R [1, p-2]$ and computes $\gamma_i = g^{\alpha_i} \bmod p$, , $\delta_i = \alpha_i^{-1}(\beta_i - x_i \gamma_i) \bmod (p-1)$ where $T_i$ denotes timestamp.

**Broadcast.** Each $U_i$ broadcasts $(y_i, CA(y_i), r_i, t_{i1}, t_{i2}, ..., t_{ii-1}, t_{ii+1}, ..., t_{in}, \sigma_i, T_i)$ where $\sigma_i = \{\gamma_i, \delta_i\}$.

**Check.** Each $U_i$ waits for all incoming messages, checks the validity of $CA(y_j)$, computes

$s_{ji} = s_{ij} = r_j^{k_i} = g^{k_i k_j} \bmod p$ and recovers the respective subsession key $K_j = t_{ji} s_{ji}^{-1} \bmod p$. Each $U_i$ then reconstructs $\beta_j = H\left(g \mid y_j \mid K_j \mid r_j \mid \gamma_j \mid T_j\right)$, $\Gamma_j = y_j^{\gamma_j} \gamma_j^{\delta_j} \bmod p$ verifies and accepts only if $\Gamma_j = g^{\beta_j} \bmod p$ holds.

**Key Computation.** Each $U_i$ computes the session key $sk = (K_1 + K_2 + \ldots + K_n) \bmod p$.

**Fig. 1**. Zheng et al.'s Group Key Agreement Protocol [28]

In order for the scheme to be dynamically applicable, Zheng et al. proposed two other key agreement algorithms, namely the member-joining algorithm and the member-removing algorithm. However, we omit full description of them as they basically consist of similar operations and possess similar weaknesses which we will describe in the next subsection.

### 4.1 Unknown Key Share Attack

This protocol is claimed to be secure against active adversaries in the random oracle model but in this subsection, we rebut their claim by invalidating the security of their scheme using the security model outlined in Section 3.

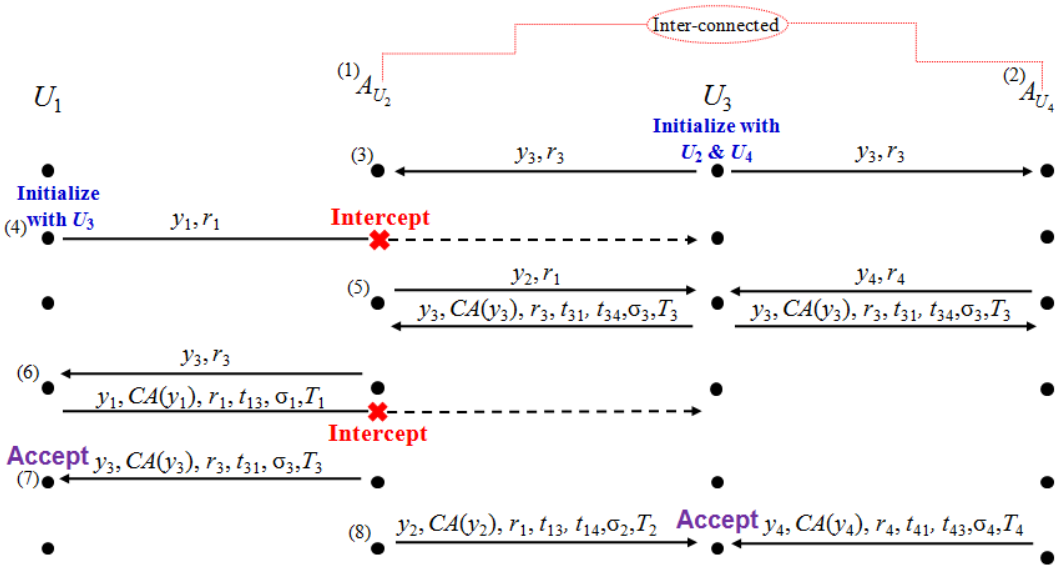


**Fig. 2**. Unknown Key Share Attack.

**Fig. 2** depicts an execution of Zheng et al.'s group key agreement protocol in the presence of two malicious participants $A_{U_2}$ and $A_{U_4}$ (with $A_{U_i}$ denoted as an adversary-controlled party $U_i$). We find that the conspiration of merely two corrupted participants is sufficient to carry out an unknown key share attack completely. Specifically, one of the corrupted users, $A_{U_4}$ needs to be responsible for decrypting $K_3$ (by performing the message exchange

honestly) and sharing this key with $A_{U_2}$ who acts as a malicious man-in-the-middle modifying and conveying messages between both honest parties $U_1$ and $U_3$. At the end of the attack, $U_3$ believes that the session key $sk_{U_3} = (K_1 + K_3 + K_4) \bmod p$ is being shared with $A_{U_2}$ and $A_{U_4}$, but in fact, a portion of this key $sk'_{U_3} = (K_1 + K_3) \bmod p$ is actually shared among $U_1$ and $U_3$ without the knowledge of $U_3$.

During the attack, *A* may issue the following queries accordingly in order to win the game.

1. **Corrupt** ($U_2$), where *A* is responded with $x_2$.
2. **Corrupt** ($U_4$), where *A* is responded with $x_4$.
3. **Send** $\left(\prod_{U_3}^{s_3}, \{U_2, U_4\}\right)$ where *A* is responded with $\{y_3, r_3\}$.
4. **Send** $\left(\prod_{U_1}^{s_1}, \{U_3\}\right)$, where *A* is responded with $\{y_1, r_1\}$.
5. **Send** $\left(\prod_{U_3}^{s_3}, \{(y_2, r_1), (y_4, r_4)\}\right)$, where *A* is responded with $\{y_3, CA(y_3), r_3, t_{31}, t_{34}, \sigma_3, T_3\}$. *A* then reconstructs $K_3 = t_{34} s_{43}^{-1} \bmod p$ and $s_{13} = t_{31} K_3^{-1} \bmod p$.
6. **Send** $\left(\prod_{U_1}^{s_1}, \{y_3, r_3\}\right)$, where *A* is responded with $\{y_1, CA(y_1), r_1, t_{13}, \sigma_1, T_1\}$. *A* recovers $K_1 = t_{13} s_{13}^{-1} \bmod p$ and produces $t_{14} = s_{41} K_1 \bmod p$, $\sigma_2 = \{\beta_2, \gamma_2\}$ with the knowledge of $K_1$.
7. **Send** $\left(\prod_{U_1}^{s_1}, \{y_3, CA(y_3), r_3, t_{31}, \sigma_3, T_3\}\right)$, and $\prod_{U_1}^{s_1}$ changes its state to Accept.
8. **Send** $\left(\prod_{U_3}^{s_3}, \{y_2, CA(y_2), r_1, t_{13}, t_{14}, \sigma_2, T_2\}, \{y_4, CA(y_4), r_4, t_{41}, t_{43}, \sigma_4, T_4\}\right)$ and $\prod_{U_3}^{s_3}$ changes its state to Accept.
9. **Test** $\left(\prod_{U_1}^{s_1}\right)$, where *A* is responded with a value $sk_{U_1}^{s_1 *}$. *A* computes $sk_{U_1}^{s_1} = (K_1 + K_3) \bmod p$ and terminates the game by outputting *b*' = 0 if $sk_{U_1}^{s_1 *} = sk_{U_1}^{s_1}$ and *b*' = 1 otherwise.

Table 1 describes the internal states of $\prod_{U_1}^{s_1}, \prod_{A_{U_2}}^{s_2}, \prod_{U_3}^{s_3}, \prod_{A_{U_4}}^{s_4}$ at the end of the protocol execution. We observe that the test oracle $\prod_{U_1}^{s_1}$ is neither a partner oracle of $\prod_{A_{U_2}}^{s_2}, \prod_{U_3}^{s_3}$ nor $\prod_{A_{U_4}}^{s_4}$ according to Definition 1 since $sid_{U_1}^{s_1} \neq \left(sid_{A_{U_2}}^{s_2} = sid_{U_3}^{s_3} = sid_{A_{U_4}}^{s_4}\right)$ and $pid_{U_1}^{s_1} \neq \left(pid_{A_{U_2}}^{s_2} = pid_{U_3}^{s_3} = pid_{A_{U_4}}^{s_4}\right)$. Since the session key held by $\prod_{U_1}^{s_1}$ can be recovered by $A_{U_2}$ with the help of $A_{U_4}$ without affecting the freshness of the test oracle, *A* is able to respond with the correct bit *b*' in its **Test** query with probability 1.

**Table 1**. Internal States of $\prod_{U_1}^{s_1}, \prod_{A_{U_2}}^{s_2}, \prod_{U_3}^{s_3}$ & $\prod_{A_{U_4}}^{s_4}$

| $\prod_U^s$ | $sid_U^s$ | $pid_U^s$ |
|---|---|---|
| $\prod_{U_1}^{s1}$ | $y_1, r_1, y_3, r_3, \{y_1, CA(y_1), r_1, t_{13}, \sigma_1, T_1\}, \{y_3, CA(y_3), r_3, t_{31}, \sigma_3, T_3\}$ | $\{U_1, U_3\}$ |
| $\prod_{\mathcal{A}_{U_2}}^{s2}$ | $y_2, r_1, y_3, r_3, y_4, r_4, \{y_2, CA(y_2), r_1, t_{13}, t_{14}, \sigma_2, T_2\},$ $\{y_3, CA(y_3), r_3, t_{31}, t_{34}, \sigma_3, T_3\}, \{y_4, CA(y_4), r_4, t_{41}, t_{43}, \sigma_4, T_4\}$ | $\{\mathcal{A}_{U_2}, U_3, \mathcal{A}_{U_4}\}$ |
| $\prod_{U_3}^{s3}$ | $y_2, r_1, y_3, r_3, y_4, r_4, \{y_2, CA(y_2), r_1, t_{13}, t_{14}, \sigma_2, T_2\},$ $\{y_3, CA(y_3), r_3, t_{31}, t_{34}, \sigma_3, T_3\}, \{y_4, CA(y_4), r_4, t_{41}, t_{43}, \sigma_4, T_4\}$ | $\{\mathcal{A}_{U_2}, U_3, \mathcal{A}_{U_4}\}$ |
| $\prod_{\mathcal{A}_{U_4}}^{s4}$ | $y_2, r_1, y_3, r_3, y_4, r_4, \{y_2, CA(y_2), r_1, t_{13}, t_{14}, \sigma_2, T_2\},$ $\{y_3, CA(y_3), r_3, t_{31}, t_{34}, \sigma_3, T_3\}, \{y_4, CA(y_4), r_4, t_{41}, t_{43}, \sigma_4, T_4\}$ | $\{\mathcal{A}_{U_2}, U_3, \mathcal{A}_{U_4}\}$ |

## 4.2 Other Issues

**Potential Ephemeral Leakage via Coin Query.** Accidental leakage of session-specific internal state could possibly render a key establishment insecure if no extra precaution was taken during a key establishment protocol design. In fact, this leakage is not trivial since practically, *A* could have gained full control over the (external) random source of which the participants are depending on in generating their random ephemeral secrets, or *A* could have hacked into/hijacked a participant's machine during/after a key establishment session and if the ephemeral secrets were stored in an insecure memory or were not erased properly after employment, *A* would always be able to recover them in no time.

Canetti-Krawczyk [14] first formally treated potential ephemeral leakage issue by introducing a **Session State Reveal** query to grant the adversary an additional capability in revealing the internal state of a specific session in their security model. As indicated by Canetti-Krawzyck, the amount of information included in the local state of a session should be specified explicitly by the protocol itself so that any revelation of such information would not trivially affect the secrecy of session key. Motivated by this pioneered work, more discussions have been opened up on this issue after then. Cheng et al. [16] introduced a **Coin** query (similar to **Session State Reveal** query) and precisely addressed how the query should be responded, particularly on what information should be disclosed. Recently, LaMacchia et al. [23] extended Canetti-Krawczyk's work by further defining a much stronger adversary in their model where the capability of issuing an **Ephemeral Key Reveal** query is granted. Specifically, they mandate revelation of all session-specific ephemeral information held by a party when such a query is being issued. This in fact facilitates the differentiation between protocols which are breakable and those that are still secure when all session-specific random flips of a party happen to be leaked.

In Zheng et al.'s protocol, the ephemeral secrets involved in a session basically consist of $\{k_1, k_2, ..., k_n, \alpha_1, \alpha_2, ..., \alpha_n\}$. However, we find that none of these could afford to be revealed (by having *A* to issue a **Coin** query) without affecting the secrecy of the session key *sk* or a party's long term secret key $x_i$. For instance, if an ephemeral secret $k_i$ happens to be exposed, *A* would then be able to compute $s_{ij} = r_j^{k_i}$ and reconstruct *sk* by obtaining $K_j = t_{ij} s_{ij}^{-1} \bmod p$ from the public information $\{r_j, t_{ji}\}$ of every other user $U_j$ for $1 \le j \le n, j \ne i$. With this, *A* can distinguish the session key from a random string and therefore able to win the random-bit guessing game with overwhelming probability by issuing a **Test** query to any fresh oracle. On

the other hand, if $\alpha_i$ happens to be leaked by some means, a malicious participant $A_I$, who is able to obtain the subsession key $K_i$ can derive $U_i$'s secret key by computing $x_i = (\beta_i - \alpha_i \delta_i) / \gamma_i \bmod (p-1)$.

**Imperfect Key Control.** Moreover, it is not difficult to see that in the above scheme, a malicious insider $A_I$ can predetermine the session key to a fixed value $sk'$ by executing the protocol with $U_1$ and $U_2$ in the following fashion: $A_I$ honestly follows all the protocol specification except in the second round, $A_I$ does not compute the second message independently but on the contrary, $A_I$ waits until $U_1$ and $U_2$'s messages for the second round arrived and recovers the subsession keys $K_1$ and $K_2$. In the sequel, instead of randomly selecting a subsession key $K_{A_I}$, $A_I$ sets $K_{A_I} = (sk' - K_1 - K_2) \bmod p$ before performing subsequent computations and sending $A_I$'s second message to the other users. At the end of the protocol execution, both $U_1$ and $U_2$ would agree on the session key $sk = \left(K_1 + K_2 + K_{A_I}\right) \bmod p = sk'$. In fact, we discover that the imperfect key control could further lead to an extended unknown key share attack which was termed as *bilateral unknown key share attack* as discussed in detail by Chen and Tang in [15].

### 4.3 Countermeasures

During the unknown key share attack, each El Gamal signature $\sigma_i = \{\gamma_i, \delta_i\}$ is relayed to a different session involving different set of users due to lack of inclusion of the participants' identities and the session identifiers (first round messages) in the signature scheme. Note also that the adversary $A$ is unable to fabricate or modify the signature due to the sealing secrets $\alpha_i$ and $x_i$. Thus, an effective way to prevent such an unknown key share attack is to alter the $\beta_i$ component in the signature scheme such that

$$\beta_i = H\left(g \mid K_i \mid \gamma_i \mid T_i \mid y_1 \mid \ldots \mid y_n \mid r_1 \mid \ldots \mid r_n\right)$$

includes all the session-specific participants' public keys $\{y_1, y_2, \ldots, y_n\}$ and the first round messages $\{r_1, r_2, \ldots, r_n\}$ in each $\beta_i$ for $1 \le i \le n$. In this sense, users' identities of a particular session and the messages received in the first round of the protocol can be explicitly identified by the partners of $U_i$ during the second round of the protocol through verification of partners' messages. Considering the attack scenario shown in **Fig. 2** against our improvement, $A$ will not anymore be able to claim the ownership of the message $(r_3, t_{31})$ and hence unable to deceive $U_1$ into believing that the partners whom $U_3$ is establishing a session key with is exactly $U_1$ himself since the verification of $\Gamma_3$ by $U_1$ in the second round of the protocol would yield a negative result due to different partner identities used in $\beta_i$. A similar prevention can also be seen from the unsuccessful deception of $U_3$ with the misleading ownership of message $(r_1, t_{13})$. In other words, if $A$ carries out the step-by-step attack listed in Section 4.1, the simulation would halt at steps 7 and 8 as the oracles $\prod_{U_1}^{s_1}$ and $\prod_{U_3}^{s_3}$ would change their state into Reject.

To effectively address the issue of potential ephemeral leakage, enhancements are required in such a way that if $A$ issues a **Coin** $(\prod_{U_i}^{s_i})$ query to an accepted oracle to reveal the

ephemeral secrets, the secrecy of the session key *sk* as well as the long term private key $x_i$ should remain in effect. To recall, the two ephemeral secrets that a user $U_i$ possesses are $k_i$ (for binding the subsession key $K_i$ in round 1 of the protocol) and $\alpha_i$ (for use in the signature scheme). For the former case, instead of protecting $K_i$ using ephemeral secret solely, it is suggested that embedding the use of long term secrets together with the ephemeral secrets in binding the subsession key would completely overcome the leakage threat [23]. Thus, a straightforward way of achieving that is to redefine $r_i = g^{k_i x_i}$ , $r_j = g^{k_j x_j}$ and $s_{ij} = r_j^{k_i x_i} = r_i^{k_j x_j}$ for $1 \le i, j \le n, j \ne i$ and $K_i$ can then be retrieved by $K_i = t_{ij} s_{ij}^{-1} \bmod p$ where $t_{ij}$ is public. With this, we can ensure that $s_{ij}$ can be recovered only if a party has full knowledge of both long term and ephemeral secrets of $U_i$: $\{k_i, x_i\}$ or $U_j$: $\{k_j, x_j\}$, in which the party could be nobody else other than $U_i$ or $U_j$ himself. In other words, provided that the session-specific public information is made available to *A*, *A* will not be able to recover the respective subsession key by issuing exposing either $\{k_i, k_j\}$, $\{x_i, k_j\}$ or $\{k_i, x_j\}$ during the key establishment. To succeed, *A* must issue both **Coin**$(\prod_{U_i}^{s_i})$ and **Corrupt($U_i$)** queries, or **Coin**$(\prod_{U_j}^{s_j})$ and **Corrupt($U_j$)** queries to obtain $\{x_i, k_i\}$ or $\{x_j, k_j\}$ in order to derive the respective subsession key $K_i$. However, he is not allowed to issue a **Test** query to any of the target or partner oracles anymore, as both **Coin** and **Corrupt** queries issued to the same entity have violated the freshness of such oracles. Thus, the advantage of adversary in winning the bit guessing game is expected to remain negligible. As for the leakage of $\alpha_i$, the ElGamal signature scheme employed in the protocol should be replaced with a more secure and unforgeable scheme in order to prevent any undesired long term secret key recovery by any insider adversary $A_I$.

The possible threat of having a malicious insider $A_I$ to control over the value of the session key can be withstood by forcing the selection of the subsession key $K_i$ of every participant $U_i$ to be performed in the first round of the protocol in order to ensure the independency of $K_i$ with respect to the other $K_j$ in a specific session. As an example, an extra parameter $H(K_i \mid y_i \mid r_i)$ an be broadcasted along with $\{y_i, r_i\}$ by every participant $U_i$ in the first protocol round and upon receiving it, such a parameter should be verified by every partner $U_j$ after deriving the corresponding subsession key $K_i$ from $t_{ij}$ in the second protocol round. Since $K_i$ is originally included in $\beta_i$ in which its authenticity can be assured, the adversary hence would not be able to modify $H(K_i \mid y_i \mid r_i)$ of an honest user $U_i$ before reaching the intended parties in the first protocol round since such malicious act will be detected during $U_i$'s signature verification in the second protocol round.

In addition, it makes sense to use a key deriving function such as a one-way collision-resistant hash function for deriving the final session key. For instance, $sk_{U_i}^{s_i} = kdf\left(pid_{U_i}^{s_i} \mid sid_{U_i}^{s_i} \mid (K_1 + K_2 + ... + K_n) \bmod p\right)$. It is worth noting that the inclusion of the involved participants' identities $pid_{U_i}^{s_i}$ and the session identifiers $sid_{U_i}^{s_i}$ in the key

derivation function is necessary to prevent any adversary *A* from forcing two different sessions with different participated entities or/and different session identifers to share a same session key. As put forward and proven by Choo et al.[18], this construction is of high importance in withstanding any malicious key replicating and unknown key share attacks.

## 5. Shi et al.'s Group Key Agreement Protocol

Shi et al. [27] proposed a one-round identity based group key agreement protocol in 2005, whose security claims were heuristically substantiated. Their proposed algorithm can be described as follows:

**System Setup.** Let $\langle X_1, X_2, q, P, \hat{e} \rangle$ be as specified. A Key Generation Center (KGC) chooses $s_1, s_2 \in_R Z_q^*$ as its private keys, computes $P_{pub} = s_1 P$ and $P_{pub}' = s_2 P$ as its public keys, selects a one-way collision-free hash function $H_1 : \{0,1\}^* \to Z_q^*$ and advertises $\langle q, X_1, X_2, \hat{e}, P, P_{pub}, P_{pub}', H_1 \rangle$ as the public system parameters.

**Key Extraction.** For every user $U_i$ with its identity information $ID_i$, the KGC computes $I_i = H(ID_i)$ as its digital identifier. The user's public key is computed as $Q_i = (I_i s_1 + s_2)P = I_i P_{pub} + P_{pub}'$ . The KGC computes each user's private key as $S_i = (I_i s_1 + s_2)^{-1} P$ and sends $S_i$ to the respective $U_i$ through a secure channel.

**Key Agreement.** Let $U_1, U_2, ..., U_n$ be the users with $ID_i$, $1 \le i \le n$. Suppose that the users wish to establish a communication session by agreeing on a session key. The protocol can be carried out as in **Fig. 3**.

**Broadcast.** Each $U_i$ picks a random $a_i \in_R Z_p^*$, computes $T_{i,j} = a_i Q_j$ for every $1 \le j \le n, j \ne i$ and broadcasts $T_{i,j}$ to the respective user $U_j$.

**Computation.** Each $U_i$ waits until messages $T_{j,i}$ for all $U_j$ arrived and computes the session key $sk = \hat{e}\left(T_{1,i} + T_{2,i} + ... + T_{i-1,i} + a_i Q_i + T_{i+1,i} + ... + T_{n,i}, S_i\right)$.

**Fig. 3**. Shi et al.'s Group Key Agreement Protocol [27]

### 5.1 Weaknesses

**Key Replicating Attack.** **Fig. 4** depicts an example execution of Shi et al.'s protocol in the presence of a malicious adversary *A*. For simplicity, we set *n* = 2 where the involved protocol principals in our example are $U_1$ and $U_2$. In fact, this attack can be expanded to any number of parties in the similar fashion. At the end of the protocol execution, both $U_1$ and $U_2$ have terminated with the same session key $sk = \hat{e}\left(P, P\right)^{(a_1 + a_2 + x)}$. However, according to Definition 1, both $U_1$ and $U_2$ are non-partners since they do not have matching conversations. $U_1$'s transcript is recorded as $(a_1 Q_2 \,|\, (a_2 + x) Q_1)$ whilst $U_2$'s transcripts is recorded as

$(a_2Q_1 \mid (a_1 + x)Q_2)$. If $\prod_{U_1}^{s_1}$ is chosen as the test oracle, *A* could then trivially reveal the non-partner oracle $\prod_{U_2}^{s_2}$ to acquire the (same) session key while preserving the freshness of session $s_1$. Subsequently *A* will be able to win the game by guessing the correct bit *b'* with non-negligible probability, in violation with Condition 2 of Definition 2. Hence, the key establishment goal is apparently violated.

| $U_1$ | | $\mathcal{A}$ | | $U_2$ |
|---|---|---|---|---|
| $\{Q_1 = (\mathcal{H}_1(ID_1)s_1 + s_2)P\}$ | | | | $\{Q_2 = (\mathcal{H}_1(ID_2)s_1 + s_2)P\}$ |
| $\{S_1 = (\mathcal{H}_1(ID_1)s_1 + s_2)^{-1}P\}$ | | | | $\{S_2 = (\mathcal{H}_1(ID_2)s_1 + s_2)^{-1}P\}$ |
| $a_1 \in_R \mathbb{Z}_p^*$ | | | | $a_2 \in_R \mathbb{Z}_p^*$ |
| $T_{1,2} = a_1Q_2$ | $\xrightarrow{T_{1,2}}$ | Intercept | $\xleftarrow{T_{2,1}}$ | $T_{2,1} = a_2Q_1$ |
| | | $x \in_R \mathbb{Z}_p^*$ | | |
| | | $T_{1,2}^\star = a_1Q_2 + xQ_2$ | | |
| $K_1 = \hat{e}(a_1Q_1 + T_{2,1}^\star, S_1)$ | $\xleftarrow{T_{2,1}^\star}$ | $T_{2,1}^\star = a_2Q_1 + xQ_1$ | $\xrightarrow{T_{1,2}^\star}$ | $K_2 = \hat{e}(T_{1,2}^\star + a_2Q_2, S_2)$ |
| $= \hat{e}(a_1Q_1 + a_2Q_1 + xQ_1, S_1)$ | | | | $= \hat{e}(a_1Q_2 + xQ_2 + a_2Q_2, S_2)$ |
| $= \hat{e}((a_1 + a_2 + x)P, P)$ | | | | $= \hat{e}((a_1 + a_2 + x)P, P)$ |
| $sk_{U_1}^{s1} = sk_{U_2}^{s2} = \hat{e}(P, P)^{(a_1+a_2+x)}$ | | | | |

**Fig. 4**. Key Replicating Attack against Shi et al.'s Protocol

**Triangle Attack.** Since key derivation function is not employed in constructing the session key, we spot another valid insider known key attack against Shi et al.'s group key agreement scheme. Motivated by Burmester [13], our triangle attack is basically two-pronged, consisting of a passive eavesdropping attack, followed by an active known key attack. In the passive session (Session (a)), the adversary *A* eavesdrops on the exchanged messages and replay them in the subsequent active sessions (Session (b) and (c)). By revealing the session keys of the active sessions, *A* is then able to perform a series of computation to recover the session key of the eavesdropped session. In the sequel, we present a series of queries that an adversary may ask during its attack in order to win the game with non-negligible probability.

**Phase 1 (Session (a))**:

1. **Send** $\left(\prod_{U_1}^{s_1(a)}, \{U_2\}\right)$, where *A* is responded with $\left\{T_{1,2}^{(a)}\right\}$.
2. **Send** $\left(\prod_{U_2}^{s_2(a)}, \{U_1\}\right)$, where *A* is responded with $\left\{T_{2,1}^{(a)}\right\}$.
3. **Send** $\left(\prod_{U_1}^{s_1(a)}, \left\{T_{2,1}^{(a)}\right\}\right)$, where $\prod_{U_1}^{s_1(a)}$ changes its state to Accept.
4. **Send** $\left(\prod_{U_2}^{s_2(a)}, \left\{T_{1,2}^{(a)}\right\}\right)$, where $\prod_{U_2}^{s_2(a)}$ changes its state to Accept.

**Phase 2 (Session (b)):**

1. **Send** $\left(\prod_{U_1}^{s_1(b)}, \{U_3\}\right)$, where *A* is responded with $\left\{T_{1,3}^{(b)}\right\}$.

2. **Send** $\left(\prod_{U_1}^{s_1(b)}, \{T_{2,1}^{(a)}\}\right)$, where $\prod_{U_1}^{s_1(b)}$ changes its state to Accept.

**Phase 3 (Session (c)):**

1. **Send** $\left(\prod_{U_2}^{s_2(c)}, \{U_3\}\right)$, where $A$ is responded with $\{T_{2,3}^{(c)}\}$.
2. **Send** $\left(\prod_{U_2}^{s_2(c)}, \{T_{1,2}^{(a)}\}\right)$, where $\prod_{U_2}^{s_2(c)}$ changes its state to Accept.

**Attacking Events:**

1. **Corrupt** $(U_3)$, where $A$ is responded with $S_3$.
2. **Reveal** $(\prod_{U_1}^{s_1(b)})$, where $A$ is responded with $sk_{U_1}^{s_1(b)} = \hat{e}(P,P)^{(a_1^{(b)}+a_2^{(a)})}$.
3. **Reveal** $(\prod_{U_2}^{s_2(c)})$, where $A$ is responded with $sk_{U_2}^{s_2(c)} = \hat{e}(P,P)^{(a_2^{(c)}+a_1^{(a)})}$.
4. $A$ computes accordingly $z_1 = \hat{e}\left(T_{1,3}^{(b)}, S_3\right) = \hat{e}(P,P)^{\left(a_1^{(b)}\right)}$, $z_2 = \hat{e}\left(T_{2,3}^{(c)}, S_3\right) = \hat{e}(P,P)^{\left(a_2^{(c)}\right)}$ and $sk^{(a)} = \dfrac{sk_{U_1}^{s_1(b)} \cdot sk_{U_2}^{s_2(c)}}{z_1 \cdot z_2} = \hat{e}(P,P)^{\left(a_1^{(a)}+a_2^{(a)}\right)}$.
5. **Test** $(\prod_{U_1}^{s_1(a)})$, where $A$ is responded with a value $sk_{U_1}^{s_1(a)*}$. $A$ terminates the game by outputting $b' = 0$ if $sk_{U_1}^{s_1(a)*} = sk^{(a)}$ and $b' = 1$ otherwise.

Throughout the game, note that $U_1$ or $U_2$ has not been issued any **Corrupt** query and $\prod_{U_1}^{s_1(a)}$ or its partner oracle $\prod_{U_2}^{s_2(a)}$ has not been subject to any **Reveal** query. Since at the end of the game, $A$ could distinguish the session key of session (a), $sk^{(a)}$ from a random string without violating the freshness of the test oracle, $A$ is therefore able to output the guess correctly with probability 1.

### 5.2 Countermeasures

It would appear that by changing the construction of the session key to $sk = H_2\left(pid \mid sid \mid \hat{e}(P,P)^{(a_1+a_2)}\right)$ with $H_2 : \{0,1\}^* \rightarrow X_2$, $pid = \{ID_{U_1} \mid ID_{U_2}\}$ and $sid = \{T_{1,2} \mid T_{2,1}\}$, the depicted flaws due to the key replicating attack and the triangle attack can both be effectively defeated.

**Tackling the Key Replicating Attack.** The inclusion of $pid$ and $sid$ in the key derivation function is necessary as it effectively binds the session key to the involved players $U_1$ and $U_2$, and all messages communicated by them. Note that this session key construction strategy was first proposed by Choo et al. [18]. If $A$ happens to modify any of the messages during transmission as shown in **Fig. 4**, each party will be ending up with a different session key since $sk_{U_1}^{s_1} = H_2\left(ID_{U_1} \mid ID_{U_2} \mid T_{1,2} \mid T_{2,1}^* \mid \hat{e}(a_1Q_1 + T_{2,1}^*, S_1)\right) = H_2(ID_{U_1} \mid ID_{U_2} \mid a_1Q_2 \mid (a_2+x)Q_1 \mid \hat{e}(P,P)^{(a_1+a_2+x)})$ while

$sk_{U_2}^{S_2} = H_2\left(ID_{U_1} \mid ID_{U_2} \mid T_{1,2}^* \mid T_{2,1} \mid \hat{e}(T_{1,2}^* + a_2 Q_2, S_2)\right) = H_2(ID_{U_1} \mid ID_{U_2} \mid (a_1 + x)Q_2 \mid a_2 Q_1 \mid \hat{e}(P,P)^{(a_1+a_2+x)})$ .

Therefore, the key replicating attack as shown in **Fig. 4** will no longer be applicable against Shi et al.'s protocol.

**Tackling the Triangle Attack.** Since a one-way collision-free hash function $H_2$ is used as a key derivation function, revealing the session keys of active sessions will not assist the adversary *A* in recovering the partial keys of the eavesdropped session [13]. As a result, the adversarial computation in step 4 of the attacking events in the previous subsection will not be feasible. Hence, the triangle attack will no longer be valid against Shi et al.'s protocol.

## 5. Conclusions

The long term intent of our research is to emphasize the importance of taking the provable security approach to designing and analyzing protocols as highlighted time and again by researchers. While numerous notable researchers have contributed significantly to this field in the provable sense, many protocols remain and which are subsequently proposed in parallel, that either come:

- without security treatments,
- with heuristic security arguments, or
- with provable security claims but without any proofs.

None of the above three is sufficient considering the maturity of provable security for key establishment protocols.

To this end, we have highlighted the security weaknesses of Zheng et al.'s and Shi et al's group key agreement protocol, particularly due to:

- inappropriate key derivation function,
- receiving users' identities not explicitly identified,
- known temporary information security property not adequately considered,
- potential key freshness violation by malicious insider.

Furthermore, to draw some lessons learnt, we discussed appropriate countermeasures to address the flaws, which significantly support the findings of Burmester's [13] and Choo et al.'s [18] work. The extension of this work would be to equip these rectified protocols with full formal proofs. In fact, this can be done in a straight forward manner by following typical approaches in the literature [17][18][20].

## Acknowledgements

We are very grateful to Dr. Raphael Phan for offering invaluable discussion and constructive comments on our work. The first author would like to acknowledge the support by the Ministry of Knowledge Economy of South Korea under IT/SW creative research program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2010-(C1810 -1002-0016)). The third author would also like to acknowledge 2010 Research Fund of Dongseo University.

## References

[1] G. Ateniese, M. Steiner and G. Tsudik, "New Multiparty Authentication Services and Key Agreement Protocols," *Journal of Selected Areas in Communications*, vol. 18, no. 4, pp. 1-13, 2009. Article (CrossRef Link).

[2] M. Bellare, R. Canetti and H. Krawczyk, "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols," in *J. Vitter (Ed.), Proc. 30th Annual ACM Symposium on the Theory of Computing*, ACM STOC 1998, ACM Press, Dallas, Texas, USA, pp. 419-428, 1998. Article (CrossRef Link).

[3] M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated Key Exchange Secure against Dictionary Attacks," in *B. Preneel (Ed.), Proc. International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology - EUROCRYPT 2000, Lecture Notes in Computer Science*, vol. 1807, Springer-Verlag, Berlin/Heidelberg, Bruges, Belgium, pp. 139-155, 2000. Article (CrossRef Link).

[4] M. Bellare and P. Rogaway, "Entity Authentication and Key Distribution," in *D.R. Stinson (Ed.), Proc. 13th Annual International Cryptology Conference, Advances in Cryptology - CRYPTO 1993, Lecture Notes in Computer Science*, vol. 773, Springer-Verlag, Santa Barbara, California, USA, pp. 232-249, 1993. Article (CrossRef Link).

[5] M. Bellare and P. Rogaway, "Provably Secure Session Key Distribution - The Three Party Case," in *F.T. Leighton, A. Borodin (Eds.), Proc. 27th ACM Symposium on the Theory of Computing, ACM STOC 1995*, ACM Press, Las Vegas, Nevada, USA, pp. 57-66, 1995. Article (CrossRef Link).

[6] S. Blake-Wilson, D. Johnson and A. Menezes, "Key Agreement Protocols and their Security Analysis," in *M. Darnell (Ed.), Proc. 6th IMA International Conference on Cryptography and Coding, Lecture Notes in Computer Science*, vol. 1355, Springer-Verlag, Cirencester, UK, pp. 30-45, 1997. Article (CrossRef Link).

[7] S. Blake-Wilson and A. Menezes, "Authenticated Diffie-Hellman Key Agreement Protocols," in *S.E. Tavares, H. Meijer (Eds.), Proc. 5th Annual Workshop on Selected Areas in Cryptography, SAC 1998, Lecture Notes in Computer Science*, vol. 1556, Springer-Verlag, Atlanta, Georgia, USA, pp. 339-361, 1998. Article (CrossRef Link).

[8] J.-M. Bohli, M.I.G. Vasco and R. Steinwandt, "Secure Group Key Establishment Revisited," *International Journal of Information Security*, vol. 6, no. 4, pp. 243-254, 2007. Article (CrossRef Link).

[9] C. Boyd and J. M. González Nieto, "Round-optimal Contributory Conference Key Agreement," *Public Key Cryptography (PKC 2003), Lecture Notes in Computer Science*, Springer-Verlag, vol. 2567, pp. 161-174, 2003. Article (CrossRef Link).

[10] E. Bresson, O. Chevassut, A. Essiari, and D. Pointcheval, "Mutual Authentication and Group Key Agreement for Low-Power Mobile Devices," *Computer Communications*, vol. 27, no. 17, pp. 1730-1737, 2004. Article (CrossRef Link).

[11] E. Bresson, O. Chevassut and D. Pointcheval, "Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case," in *C. Boyd (Ed.), Proc. 7th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology - ASIACRYPT 2001, Lecture Notes in Computer Science*, vol. 2248, Springer-Verlag, Gold Coast, Australia, pp. 290-309, 2001. Article (CrossRef Link).

[12] E. Bresson, O. Chevassut, D. Pointcheval and J.-J. Quisquater, "Provably Authenticated Group Diffie-Hellman Key Exchange," in *P. Samarati (Ed.), Proc. 8th ACM Conference on Computer and Communications Security, ACM CCS 2001*, ACM Press, Philadelphia, Pennsylvania, USA, pp. 255-264, 2001. Article (CrossRef Link).

[13] M. Burmester, "On the Risk of Opening Distributed Keys," in *Y.G. Desmedt (Ed.), Proc. 14th Annual International Cryptology Conference, Advances in Cryptology - CRYPTO 1994, Lecture Notes in Computer Science*, vol. 839, Springer-Verlag, Santa Barbara, California, USA, pp. 308-317, 1994. Article (CrossRef Link).

[14] R. Canetti and H. Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels," in *B. Pfitzmann (Ed.), Proc. International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology - EUROCRYPT 2001, Lecture*

*Notes in Computer Science*, vol. 2045, Springer-Verlag, Innsbruck, Austria, pp. 453-474, 2001. Article (CrossRef Link).

[15] L. Chen, Q., Tang, "Bilateral Unknown Key-Share Attacks in Key Agreement Protocols," *Journal of Universal Computer Science*, vol. 14, no. 3, pp. 416-440, 2008. Article (CrossRef Link).

[16] Z. Cheng, M. Nistazakis, R. Comley and L. Vasiu, "On the Indistinguishability-based Security Model of Key Agreement Protocols-Simple Cases," *Cryptology ePrint Archive: Report 2005/129*, 2005. Article (CrossRef Link).

[17] K.-K.R. Choo, "Key Establishment : Proofs and Refutations," *PhD thesis, Queensland University of Technology*, 2006. Article (CrossRef Link).

[18] K.-K.R. Choo, C. Boyd and Y. Hitchcock, "On Session Key Construction in Provably-Secure Key Establishment Protocols," in *E. Dawson, S. Vaudenay (Eds.), Proc. 1st International Conference on Cryptology in Malaysia, Progress in Cryptology - MYCRYPT 2005, Lecture Notes in Computer Science*, vol. 3715, Springer-Verlag, Kuala Lumpur, Malaysia, pp. 116-131, 2005. Article (CrossRef Link).

[19] K.-K.R. Choo, C. Boyd and Y. Hitchcock, "Errors in Computational Complexity Proofs for Protocols," in *B. Roy (Ed.), Proc. 11th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology - ASIACRYPT 2005, Lecture Notes in Computer Science*, vol. 3788, Springer-Verlag, Chennai, India, pp. 624-643, 2005. Article (CrossRef Link).

[20] R. Dutta and R. Barua, "Provably Secure Constant Round Contributory Group Key Agreement in Dynamic Setting," *IEEE Trans. on Information Theory*, vol. 54, no. 5, pp. 2007-2025, 2008. Article (CrossRef Link).

[21] I. Ingemarsson,T.D. Tang and C.K. Wong, "A Conference Key Distribution System," *IEEE Transactions of Information Theory*, vol. 28, no. 5, pp. 714-719, 1982. Article (CrossRef Link).

[22] M. Just and S. Vaudenay, "Authenticated Multi-Party Key Agreement," in *K. Kim, T. Matsumoto (Eds.), Proc. 2nd International Conference on the Theory and Applications of Cryptology and Information Security, Advances in Cryptology - ASIACRYPT 1996, Lecture Notes in Computer Science*, vol. 1163, Springer Berlin/Heidelberg, Gyeongju, Korea, pp. 36-49, 1996. Article (CrossRef Link).

[23] B. LaMacchia, K. Lauter and A. Mityagin, "Stronger Security of Authenticated key Exchange," in *W. Susilo, J.K. Liu, Y. Mu (Eds.), Proc. 1st International Conference, PROVSEC 2007, Lecture Notes in Computer Science*, vol. 4784, Springer Berlin/Heidelberg, Wollongong, Australia, pp. 1-16, 2007. Article (CrossRef Link).

[24] M.-H. Lim, C.-M. Yeoh, S. Lee, H. Lim and H. Lee, "A Secure and Efficient Three-Pass Authenticated Key Agreement Protocol Based on Elliptic Curves," in *A. Das, H.K. Pung, F.B.S. Lee, L.W.C. Wong (Eds.), Proc. 7th International IFIP-TC6 Networking Conference, NETWORKING 2008, Lecture Notes in Computer Science*, vol. 4982, Springer Berlin/Heidelberg, Singapore, pp. 170-182, 2008. Article (CrossRef Link).

[25] J. Nam, S. Kim, and D. Won, "Attacks on Bresson-Chevassut-Essiari-Pointcheval's Group Key Agreement Scheme for Low-Power Mobile Devices," *Cryptology ePrint Archive: Report 2004/251*, 2004. Article (CrossRef Link).

[26] O. Pereira and J.-J. Quisquater, "A Security Analysis of the Cliques Protocols Suites," in *Proc. 14th IEEE Computer Security Foundations Workshop, CSFW 2001, IEEE Computer Society Press*, Cape Breton, Nova Scotia, Canada, pp. 73-81, 2001. Article (CrossRef Link).

[27] Y. Shi, G. Chen and J. Li, "ID-Based One Round Authenticated Group Key Agreement Protocol with Bilinear Pairings," in *Proc. International Conference on Information Technology: Coding and Computing, ITCC 2005, vol. 1, IEEE Computer Society Press*, Las Vegas, Nevada, USA, pp. 757-761, 2005. Article (CrossRef Link).

[28] S. Zheng, S. Wang and G. Zhang, "A Dynamic, Secure and Efficient Group Key Agreement Protocol," *Frontiers of Electrical and Electronic Engineering in China*, vol. 2, no. 2, Higher Education Press co-published with Springer-Verlag GmbH, 182-185, 2007. Article (CrossRef Link).

[29] L. Zhou, W. Susilo, and Y. Mu, "Efficient ID-Based Authenticated Group Key Agreement from

Bilinear Pairings," *MSN 2006, Lecture Notes in Computer Science*, vol. 4325, Springer Berlin, pp. 521-532, 2006. Article (CrossRef Link).

**Meng-Hui Lim** obtained his B.Eng in 2006 from Multimedia University in Malaysia and M.Eng degree in 2009 from Dongseo University in South Korea. He is currently working towards his Ph.D. degree in Electronics Engineering Department at Yonsei University in South Korea. His research interests include cryptography, authentication protocols, biometric security, pattern recognition and discretization. He has more than 20 international publications in his area of research.

**Bok-Min Goi** received his B.Eng degree from University of Malaya (UM) in 1998, and the M.Eng.Sc and Ph.D degrees from Multimedia University (MMU), Malaysia in 2002 and 2006, respectively. He is now the Deputy Dean (R&D and Postgraduate Programmes) and an associate professor in the Faculty of Engineering and Science, University Tunku Abdul Rahman (UTAR), Malaysia. He was the General Chair for ProvSec 2010 and CANS 2010, and the PC members for many crypto / security conferences. His research interests include cryptology, security protocols, information security, digital watermarking, and embedded systems design.

**Sang-Gon Lee** received the BEng, MEng, and PhD degree in electronics engineering from Kyungpook National University, Korea, in 1986, 1988, and 1993, respectively. He is a professor at the Division of Computer & Information Engineering, Dongseo University. He was an assistant/associate professor at Chang-shin College from 1991 to 1993 and a visiting scholar at QUT, Australia from 1993 to 1994. His research areas include information security, network security, wireless sensor network and wireless mesh network.