*Regular Paper*

# Conference Key Agreement Protocol Using Oblivious Transfer

Ari Moesriami Barmawi,† Shingo Takada† and Norihisa Doi†

Conference key is one method for establishing secure communication among members of a group. Recently, much work has been done on the establishment of the conference key. However, when confirming the key, if each member signs the conference key which he/she has calculated, each member would need to verify that all members signed the conference key in order to verify that all members are holding the same conference key. In this case, using the conventional methods, each member needs to perform several decryptions, which contribute to a high computational burden to the group. The basic idea of our protocol is to establish a conference key based on oblivious transfer which can be used in both asymmetric and symmetric cryptography. Our protocol can reduce the number of decryptions for the key confirmation without sacrificing the level of security. In our proposed method, we break the conference key into several individual secret keys in accordance with the number of members within the group. This individual key will be used by each member to sign or encrypt (asymmetrically) the established conference key. To perform the key confirmation, each member multiplies all signed conference keys and decrypts (asymmetrically) the multiplied signed conference key using the multiplicative inverse of his locally calculated conference key. Thus, by using our proposed protocol, each member only needs to perform one decryption for the key confirmation. Furthermore, by using the individual secret key, each member can directly communicate with each other with the support of the leader, while the leader does not gain any knowledge of messages which are exchanged between the communicating members. This last feature can not be found in previous methods except for Li-Pieprzyk's. The difference between Li-Pieprzyk's and ours is that for the key generation, we need only a few modular exponentiations while the former needs much more.

## 1. Introduction

There are many key agreement protocols which have been proposed for establishing session keys between more than two users, such as those proposed by Burmester-Desmedt [5], Steiner-Tsudik-Waidner[15),16)], Ateniese-Steiner-Tsudik [2], Just-Vaudenay [11] which are based on Diffie-Hellman key exchange, those of Boyd [3),4)], which are based on a one way function, and Li-Pieprzyk [12] which is based on secret sharing. Suppose for the key confirmation, each member signs the conference key which he/she calculated. To verify whether all members are holding the same conference key, each member of the group needs to verify that all members have signed the keys. In this case, the above protocols need to conduct several decryptions which increase the computational burden on the group. Furthermore, the previously proposed protocols did not provide any facilities for two members in the group to communicate securely with each other (i.e., other members of the group outside the communicating members learn nothing about messages that have been exchanged between the two communicating members), except Li-Pieprzyk's. However, our proposed protocol is more efficient than Li-Pieprzyk's, since for the key generation, Li-Pieprzyk's protocol needs several modular exponentiations, while ours needs only a few.

We propose a protocol for generating the conference key and the individual secret keys using oblivious transfer. Using our method, we can establish a conference key which can be used in both symmetric and asymmetric cryptography.

In our proposed protocol, we have assumed that the group is predetermined. At the end of our proposed protocol, each member of the group can calculate the conference and individual keys. The conference key is used to encrypt (symmetrically and asymmetrically) messages which can be decrypted by all members of the group. The individual key of member $U_i$ is used to encrypt (symmetrically) messages which can be decrypted (symmetrically) only by member $U_i$ itself and the leader of the group. In the key confirmation, this individual key is used for signing the established conference key. Then, each member can verify that all members have the same conference key by multiply-

† Department of Computer Science, Graduate School of Science and Technology, Keio University

ing all signed keys and decrypting (asymmetrically) the result using the multiplicative inverse of his locally calculated conference key. Hence, for the key confirmation, each member has to perform only one decryption. Thus, instead of being used for encrypting/decrypting messages symmetrically, the conference key can be used for encrypting messages asymmetrically as well.

This individual key can also be used for signing a contract. Suppose there is a contract which has to be signed by all members. The leader broadcasts a message, then each member signs this message using his individual key and sends it back to the leader. Furthermore, the leader can verify whether the message he sent has been signed by all members only by multiplying all signed messages and decrypting (asymmetrically) it with the multiplicative inverse of the conference key, instead of decrypting each message (one by one). Thus, in this case the computational burden of the leader will be decreased.

This paper first describes the features of the supporting protocols. Section 3 describes the details of our proposed protocol. Section 4 analyzes security issues. Section 5 compares our proposed protocol with previous protocols. Section 6 makes concluding remarks.

## 2. Features of Supporting Protocols

This section describes two methods which our proposed protocol uses: oblivious transfer and multiplicative-additive share converter proposed by Gilboa [8].

### 2.1 Oblivious Transfer

Oblivious transfer was first introduced by Rabin [13] with versions such as "one out of two oblivious transfer" being proposed by Even-Goldreich-Lempel [7]. The latest version of oblivious transfer was discussed by Goldreich [9]. In this section, we discuss the "one out of $k$ oblivious transfer", which is usually called $\binom{k}{1}$-OT (for $k \in \{2, 3, \ldots\}$).

Suppose a sender holds input $b(1), \ldots, b(k)$, and the receiver holds $i \in \{1, 2, \ldots, k\}$. Oblivious transfer is a protocol to transfer the $i$-th bit to the receiver without letting the receiver obtain knowledge of any other bit and without letting the sender obtain knowledge of the identity of the bit required by the receiver.

The protocol is as follows:

- The sender has input $(b(1), \ldots, b(k)) \in \{0, 1\}$ (where $b(i)$ means the $i$-th bit of $b$), and the receiver has input $i \in \{1, 2, \ldots, k\}$.
- The sender selects a trapdoor index $\alpha$ and forms the trapdoor permutation $(f_\alpha : D_\alpha \mapsto D_\alpha)$ which is asymmetric, and sends $\alpha$ to the receiver.
- The receiver uniformly and independently selects $e(1), \ldots, e(k) \in D_\alpha$, sets $y(i) = f_\alpha(e(i))$ and $y(j) = e(j)$ for every $j \neq i$, and sends $(y(1), y(2), \ldots, y(k))$ to the sender.
- Upon receiving $(y(1), y(2), \ldots, y(k))$, using the inverting-with-trapdoor algorithm and the trapdoor $t$, the sender computes $x(j) = f_\alpha^{-1}(y(j))$, for every $j \in \{1, \ldots, k\}$. It sends $(b(1) \oplus b(x(1)), \ldots, b(k) \oplus b(x(k)))$ to the receiver.
- Finally, after receiving $(b(1) \oplus b(x(1)), \ldots, b(k) \oplus b(x(k)))$, the receiver calculates $(b(i) \oplus b(x(i))) \oplus b(e(i))$. At this point the receiver obtains $b(i)$, since $x(i) = f_\alpha^{-1}(f_\alpha(e(i)))$.

### 2.2 Converting Multiplicative Shares into Additive Shares

Gilboa has proposed a protocol for converting multiplicative shares into additive shares [8]. Suppose the sender holds multiplicative shares $a$ and the receiver holds $b$. Then:

(1) The receiver selects at random and independently $\theta$ ring elements denoted by $s_0, \ldots, s_{\theta-1} \in \boldsymbol{R}$. The ring $\boldsymbol{R}$ is a set whose elements can be encoded using $\theta$ bits (where $\theta = \log|\boldsymbol{R}|$ and $|\boldsymbol{R}|$ is the cardinality of $\boldsymbol{R}$). Receiver chooses $\theta$ pairs of elements in $\boldsymbol{R}$: $(t_0^0, t_0^1), \ldots, (t_{\theta-1}^0, t_{\theta-1}^1)$. For every $i$ $(0 < i < \theta - 1)$, the receiver defines $t_i^0 = s_i$ and $t_i^1 = 2^i b + s_i$.

(2) Let the binary representation of $a$ be $a_0, \ldots, a_{\theta-1}$. The sender and receiver execute $\theta$ oblivious transfers ($\binom{2}{1}$-OTs) [7),9)]. In the $i$-th execution, the sender chooses $t_i^{a_i}$ from the pair $(t_i^0, t_i^1)$.

(3) The sender sets $(x = \sum_{i=0}^{\theta-1} t_i^{a_i})$ and the receiver sets $(y = -\sum_{i=0}^{\theta-1} s_i)$.

At the end of this protocol, the sender and receiver hold the additive shares $x$ and $y$, such that $(x + y = ab)$.

## 3. The Proposed Protocol

Our proposed protocol assumes that one of the users is the leader whose commands must be authenticated and has public and secret keys. Since his command should be authenticated and be known only to group members, he has

to sign the parameters chosen by himself which will determine the conference and individual keys. Thus, the leader's public key should be publicly known.

The goal of our proposed protocol is to generate a conference key together with the individual key (i.e., a key which has to be kept secret between the leader and a certain member of the group). Using those keys, we obtain some advantages which are not provided by previous methods. The advantages are as follows:

- Each member can communicate securely with the leader (using symmetric cryptosystem), where other members learn nothing from their exchanged messages.
- The members can communicate with each other securely (using asymmetric cryptosystem), where both the leader and other members learn nothing.
- The computational burden in the key confirmation is decreased.

### 3.1 Details of Proposed Protocol

Suppose a group has $r$ members (including the group leader) who are honest. Furthermore, the conference key is actually the product of $r$ individual keys. The individual key of each member is determined by each member's randomly chosen number and the leader's number. To obtain the conference and individual keys, the group has to execute the following protocol:

(1) All members including the leader have to agree on the size of their randomly chosen numbers $n_i$ that will determine the conference and their individual keys and a large strong prime $\theta$. Suppose the size of each $n_i$ is $l$ bits. Let $\beta = \phi(\theta)$ (where $\phi(\theta)$ is the Euler Totient Function of $\theta$). Furthermore, they choose a number $\alpha$ where the order of ($\alpha \bmod \beta$) should be the largest possible order of an integer modulo $\beta$. This means that the order of ($\alpha \bmod \beta$) should be equal to $\lambda(\beta)$ (where $\lambda(\beta)$ is the Carmichael Function of $\beta$ [1)]) because the largest possible order of an integer modulo $\beta$ is equal to $\lambda(\beta)$ [1)].

(2) Let $U_1$ be the group leader, and $U_i$ ($i = 2, \ldots, r$) be group members. Suppose each member as well as the leader has their own public and secret keys, where the public key of each member is known only by the leader, and the leader's public key is publicly known.

(3) Each member $U_i$ chooses any integer $dum_i$, encrypts it with the leader's public key, and sends it to the leader.

(4) Each member privately chooses a number $n_i$.

(5) Suppose $U_1$ chooses at random $l$ ring elements denoted by $s_{1,1}, s_{1,2}, \ldots, s_{1,l} \in \boldsymbol{R}$ and $n_1$ as his multiplicative share (we set $s_{1,\gamma}$ as the $\gamma^{th}$ bit of $s$ for $\gamma = 1, \ldots, l$). $\boldsymbol{R}$ is the ring whose elements can be encoded using $l$ bits. He then sets $l$ pairs of $(t^0_{1,\gamma}, t^1_{1,\gamma})$ where $(t^0_{1,\gamma} = 2^\gamma n_1 + s_{1,\gamma})$ and $(t^1_{1,\gamma} = s_{1,\gamma})$. Each member $U_i$ has his own multiplicative share $n_i$ which consists of $l$ bits $(n_{i,1}, \ldots, n_{i,l})$. $U_1$ and all members invoke $l$ $\binom{2}{1}$-OTs, such that $(n_1 n_i = x_1 + x_i)$ (where $x_i$ and $x_1$ are the secret additive shares of user $U_i$ ($i = 2, \ldots, r$) and $U_1$ respectively) using Gilboa's method. In this case, $x_1 = -\sum_{\gamma=1}^{\gamma=l} s_{1,\gamma}$ and $x_i = \sum_{\gamma=1}^{\gamma=l} t^{n_{i,\gamma}}_{i,\gamma}$.

(6) Each member $U_i$ broadcasts ($y_i \equiv \alpha^{x_i + dum_i} \bmod \beta$) encrypted with his secret key and $y_i$ itself.

(7) The leader verifies whether $y_i$ comes from $U_i$ by decrypting the encrypted message using the member's public key and comparing the value of $y_i$ which is obtained from the signed message and the plaintext one.

If they are equal (i.e., the leader verifies that this message was signed by the legitimate sender) then he executes the following procedure:

- Calculate

$$\alpha^{x_i} \bmod \beta \equiv (y_i \alpha^{-dum_i}) \bmod \beta$$

- Calculate

$$P \equiv \alpha^{(r-1)x_1 + \sum_{i=2}^{i=r} x_i} \bmod \beta$$

The leader chooses a number $dum_1$ such that $P'$ (where $P' = [P(\alpha^{dum_1})]^t \bmod \beta$) is not congruent to 1 mod $\beta$. The individual key of the leader is defined as follows:

$$K_1 \equiv B^t \bmod \beta$$

where

$$B \equiv \alpha^{dum_1} \bmod \beta$$

Then the conference key $K$ is

$$K = P' \bmod \beta \qquad (1)$$

where $t$ should be agreed upon in advance by all members of the group

and $t$ is not congruent to 0 mod $\lambda(\beta)$ (where $\lambda$ is the Charmichael Function [1]).

- Broadcast the size of $(\alpha^{((r-1)x_1+dum_1)-\sum_{i=2}^{i=r} dum_i} \bmod \beta)$ denoted as $u$ along with $(\alpha^{n_1} \bmod \beta)$ and the signed $H(\alpha^{((r-1)x_1+dum_1)-\sum_{i=2}^{i=r} dum_i} \bmod \beta)$ (where $H$ is agreed upon in advance by all members and the leader).

If they are not equal he will interrupt the protocol execution, and then repeat the protocol from the beginning.

(8) $U_1$ sends $[\alpha^{((r-1)x_1+dum_1)-\sum_{i=2}^{i=r} dum_i} \bmod \beta]$ by executing $u$ $\binom{2}{1}$-OTs.

(9) Each member $U_i$ verifies whether the leader is legitimate by comparing the values of $H(\alpha^{((r-1)x_1+dum_1)-\sum_{i=2}^{i=r} dum_i} \bmod \beta)$ which is sent by the leader and the one which is calculated by $U_i$. Since $H$ is known only to the members and the leader of the group, adversaries can not calculate $H$.

Each member $U_i$ can calculate the conference key $K$ by using the following equation:

$$K = \left[\alpha^{(r-1)x_1+dum_1-\sum_{i=2}^{i=r} dum_i} \left[\prod_{i=2}^{i=r} y_i\right]\right]^t \bmod \beta \quad (2)$$

(10) Each member $U_i$ can calculate its individual key as follows:

$$K_i = \alpha^{n_1 n_i t} \bmod \beta \quad (3)$$

and the leader can calculate $U_i$'s individual key by using the following equation:

$$K_i = \alpha^{(x_i+x_1)t} \bmod \beta \quad (4)$$

(11) To verify whether all members (including the leader) hold the same conference key, each party broadcasts $\{K\}_{K_i}$ whose value is equal to $SK_i = K^{K_i \bmod \beta} \bmod \theta$. Then each member calculates

$$K \equiv \left(\prod_{i=1}^{i=r} SK_i\right)^{(K^{-1}) \bmod \beta} \bmod \theta \quad (5)$$

If the value of Eq. (5) and the value of $K$ which is locally calculated by a member are equal then the member can verify that all members hold an equal value of $K$.

Message 1. $U_1 \xrightarrow{\text{OT}} U_i : (t^0_{i,\gamma}, t^1_{i,\gamma})$

Message 2. $U_i \longrightarrow U_1 : \{dum_i\}_{Kp_1}$

Message 3. $U_i \longrightarrow * : y_i, \{y_i\}_{Ks_i}$

Message 4. $U_1 \longrightarrow U_i : u,\ \alpha^{n_1} \bmod \beta,$

$$\left\{H\left(\alpha^{(r-1)x_1+dum_1-\sum_{i=2}^{i=r} dum_i} \bmod \beta\right)\right\}_{Ks_1}$$

Message 5. $U_1 \xrightarrow{\text{OT}} U_i :$

$$\alpha^{(r-1)x_1+dum_1-\sum_{i=2}^{i=r} dum_i} \bmod \beta$$

Message 6. $U_i \longrightarrow * : SK_i$

**Fig. 1** Conference key agreement protocol using oblivious transfer.

By using this procedure, each group member can calculate his individual and conference keys by using Eqs. (3) and (2) respectively, and the leader can calculate each member's individual and conference keys by using Eqs. (4) and (1) respectively.

Appendix A.1 shows that the resulting $K$ is the product of all members' individual keys.

Since in a group there is always a possibility for member addition and removal during the conference, we will discuss the alteration of group membership in the next section.

### 3.2 Alteration of Group Membership

In our proposed scheme, we have assumed that the number of members in the group is determined prior to the execution of conference key agreement protocol. However, it is often necessary to either add a new member, or remove an existing group member after the initial group creation. Naturally, it is desirable to do so without having to rerun the entire protocol anew, unless the leader has changed. In this subsection, we will briefly sketch out the member addition and member removal schemes.

#### 3.2.1 Member Addition

The main security requirement of member addition is that the previous group keys are kept secret from both outsiders and the group's new member. In our proposed scheme, this can be

achieved as follows:

(1) Suppose that there is a new member $U_{r+1}$ who will enter the group. First, $U_{r+1}$ sends his public key to the leader and chooses a randomly chosen number $n_{r+1}$ whose size is $l$.

(2) The leader chooses a number $z$ (where $z$ is not congruent to (0 mod $\lambda(\beta)$)), calculates ($\alpha^{n_1 z}$ mod $\beta$) and ($\alpha^{z}$ mod $\beta$), signs them with his secret key and broadcasts the signed message.

(3) The leader and the new member execute Gilboa's method for converting their multiplicative share into an additive one, such that:

$$x'_1 + x_{r+1} = (n_1 z n_{r+1})$$

(4) Member $U_{r+1}$ signs his share ($\alpha^{(x_{r+1})}$ mod $\beta$) with his secret key, encrypts the signed message with the leader's public key, and sends it to the leader.

(5) The leader calculates:

$V \equiv \alpha^{(((r-1)x_1+dum_i)z+(\sum_{i=2}^{i=r+1} x_i)z)t} \bmod \beta$

If $V$ is congruent to (1 mod $\beta$) then the leader chooses a new number

$B' \equiv \alpha^{dum'_1} \bmod \beta$

to replace $B$ such that $V'$ is not congruent to (1 mod $\beta$), and $V' = V(B^{-zt}(B')^{zt} \bmod \beta)$. Otherwise, ($V' = V$ mod $\beta$) and ($B' = B$). Thus, the new conference key is

$$K' = V' \bmod \beta \tag{6}$$

and the individual key of the leader is $K_1 \equiv (B')^{tz} \bmod \beta$.

(6) The leader encrypts

$Z \equiv (\alpha^{(((r-1)x_1+dum'_1)z+(n_1 z n_{r+1}))} \bmod \beta)$

with the previous conference key $K$ and broadcasts it. He calculates

$$X \equiv (K^{z/t}\alpha^{(-dum_1+dum'_1)z+x'_1}) \bmod \beta \tag{7}$$

He encrypts $X$, $t$ with $U_{r+1}$'s public key and sends them to $U_{r+1}$.

(7) Member $U_{r+1}$ calculates the new conference key using the following equation:

$$K' = [X\alpha^{x_{r+1}}]^t \bmod \beta \tag{8}$$

and he calculates his individual key using the following equation:

$$K_{r+1} = (\alpha^{n_1 z})^{n_{r+1}t} \bmod \beta \tag{9}$$

(8) The leader can calculate the new conference key using Eq. (6) and the individual key of each member as follows:

$$K'_i = (\alpha^{(x_i+x_1)zt}) \bmod \beta \tag{10}$$

(9) Each member $U_i$ (for $i = 2, \ldots, r$) broadcasts

$$\alpha^{x_i z+dum_i} \bmod \beta \tag{11}$$

(10) Each member $U_i$ can calculate the new conference key $K'$ as follows:

$$K' = \left[ Z \left( \prod_{i=2}^{i=r} \alpha^{x_i z+dum_i} \right) \left( \alpha^{-\sum_{i=2}^{i=r} dum_i} \right) \right]^t \bmod \beta \tag{12}$$

and his new individual key is:

$$K'_i = \alpha^{z n_1 n_i t} \bmod \beta \tag{13}$$

### 3.2.2 Member Removal

The main security requirement of member removal is that subsequent (future) group keys are kept secret from both outsiders and former group members.

The basic idea of member removal in our proposed procedure is to change the number $t$ into $mt$ and delete all removed member's share. Hence, the remaining members do not have to change their share. Since the removed member does not know the value of $m$, he can not calculate the new conference key, although the remaining members' shares are not changed.

The following procedure describes the member deletion process.

(1) Let $U_d$ be the member slated for removal from the group, where $d \in \{2, \ldots, r\}$.

(2) The leader calculates

$Newdum \equiv \alpha^{-((\sum_{i=2}^{i=r} dum_i)-dum_d)} \bmod \beta$

Furthermore, he broadcasts *Newdum*.

(3) The leader chooses a number $m$ (where $m$ is not congruent to (0 mod $\lambda(\beta)$) such that

$$\alpha^{[((r-2)x_1+dum_1)+\sum_{\substack{i=2;\\ i\neq d}}^{i=r} x_i]mt} \bmod \beta$$

is not congruent to (1 mod $\beta$). Finally he encrypts $m$ with each member's individual key, and then sends it to each member $U_i$ ($i = 2, \ldots, r$ and $i \neq d$).

(4) Each member $U_i$ calculates the new conference key as follows:

$$K' = \left[ \left( \alpha^{(((r-2)x_1+dum_1)+\sum_{\substack{i=2;\\ i\neq d}}^{i=r} y_i)} \right) Newdum \right]^{mt} \bmod \beta \tag{14}$$

and his individual key is:

$$K'_i = (K_i)^{mt} \bmod \beta \tag{15}$$

By using the above procedure each member can easily calculate the new conference and individual keys, while still keeping future keys secret from both outsiders and former group members, because the former members do not know the value of $m$.

### 3.3 Communication among the Members

Our proposed protocol provides the facility for each member to be able to communicate with other member securely via the leader using the individual key. In this case, the leader helps the members to communicate with each other, but he does not know messages which have been exchanged between the two members. The procedure to communicate securely is as follows:

(1) First of all we assume that the leader is honest. During the communication, the two communicating members and the leader use asymmetric cryptosystem.

(2) Suppose member $U_i$ will send a message $M$ (where $M \in \boldsymbol{Z}_\theta$) to member $U_j$. $U_i$ sends $U_j$ and the leader $(M^{K_i R_i \bmod \beta} \bmod \theta)$. $R_i$ (and $R_j$ which will appear in the next step) is an integer chosen by $U_i$ (or $U_j$). $gcd(R_i, \beta) = 1$ and $gcd(R_j, \beta) = 1$ must hold. Furthermore, $K_i R_i$ and $K_j R_j$ should not be congruent to $(0 \bmod \lambda(\beta))$.

(3) The leader sends $U_j$ message $(M^{R_i K_j \bmod \beta} \bmod \theta)$

(4) Member $U_j$ calculates $(M^{R_i K_j K_j^{-1} \bmod \beta} \bmod \theta)$ for obtaining $(M^{R_i \bmod \beta} \bmod \theta)$ and calculates

$$G = (M^{(K_i R_i K_j R_j) \bmod \beta} \bmod \theta)$$

Furthermore, member $U_j$ sends $G$ to $U_i$.

(5) $U_i$ then calculates:

$$E = G^{(R_i K_i)^{-1} \bmod \beta} \bmod \theta$$

and sends it to $U_j$ along with $(M^{R_i \bmod \beta} \bmod \theta)$.

(6) $U_j$ obtains $M$ by calculating:

$$\begin{aligned} D &\equiv E^{(K_j R_j)^{-1} \bmod \beta} \bmod \theta \\ &\equiv M \bmod \theta \end{aligned}$$

To verify whether $U_i$ is a legitimate member, $U_j$ compares the values of $(M^{R_i \bmod \beta} \bmod \theta)$ sent by the leader and $U_i$. If they are equal, $U_i$ is a legitimate member. Otherwise he is an adversary.

Since the leader does not know the values of $R_i$ and $R_j$, it is not possible for him to obtain $M$. Thus, using this method, two members can communicate securely with the help of the leader. However, the leader does not gain any knowledge about the messages that are exchanged between the two members.

## 4. Security Analysis

In this section, we will discuss the security of our proposed scheme against passive and active intruders. We also specify security goals which are achievable by the proposed protocol.

### 4.1 Passive Attacks

We first analyze the resistance of our proposed scheme against passive attacks. Passive attacks whose aim is key recovery for a given session involves eavesdropping on messages passed between participants for that session.

**Lemma 1** By assuming that all members of the group (including the leader) are honest, then the probability for breaking the protocol using the passive attack depends on the value of $\beta$'s prime factors.

*Proof.* From Section 3.1 of this paper, the conference key recovery will be successful if the adversary can guess $t$ and $(\alpha^{(r-1)x_1 - \sum_{i=2}^{i=r} dum_i} \bmod \beta)$ after eavesdropping $y_i$ and $u$. If we assume that all members are honest and the size of $t$ is $|t|$ bits, then there are $2^{|t|}$ possible values of $t$. Thus, the probability for obtaining a certain value of $t$ is $1/2^{|t|}$. The probability for obtaining a certain value of $(\alpha^{(r-1)x_1 + dum_1 - \sum_{i=2}^{i=r} dum_i} \bmod \beta)$ is $(1/(\lambda(\beta) - 1))$, since the largest possible order of an integer modulo $\beta$ is equal to $\lambda(\beta)$, and this means that there are $(\lambda(\beta) - 1)$ possible values for $(\alpha^{(r-1)x_1 + dum_1 - \sum_{i=2}^{i=r} dum_i} \bmod \beta)$. From the above discussion, the probability for obtaining the value of $K$ is equal to $(1/[(\lambda(\beta) - 1)(|2^t|)])$. But since the largest possible order of an integer modulo $\beta$ is $(\lambda(\beta) - 1)$ while $((2^{(|t|)}(\lambda(\beta) - 1)) > \lambda(\beta) - 1)$, the probability for obtaining $K$ is equal to $1/(\lambda(\beta) - 1)$. Finally, the protocol will be secure if we choose a large strong prime of $\beta$ such that $\lambda(\beta)$ is large as well.

Another possibility to attack the proposed protocol is through the eavesdropping of $(\alpha^{n_1} \bmod \beta)$, because from $(\alpha^{n_1} \bmod \beta)$, the attacker can try to guess the value of $(\alpha^{n_1 \sum_{i=2}^{i=r} n_i} \bmod \beta)$ and obtain the value of $K$.

We will now observe the possibility of obtaining $K$ through eavesdropping ($\alpha^{n_1} \bmod \beta$). Since the largest possible order of an integer modulo $\beta$ is $\lambda(\beta)$, then the probability for obtaining a certain value of ($\alpha^{n_1 \sum_{i=2}^{i=r} n_i} \bmod \beta$) using brute force attack is $1/(\lambda(\beta)-1)$.

Finally, since the value of $\lambda(\beta)$ depends on the values of $\beta$'s prime factors [1], the probability of breaking the protocol using passive attack depends on the value of $\beta$'s prime factors. Thus, we conclude that the greater the prime factors of $\beta$ become, the smaller the probability for finding $K$ becomes. □

### 4.2 Active Attack

In this section we will discuss the security of our proposed scheme against active attacks.

An active attack is usually defined as an impersonation attack which involves an attacker, who is given access to all publicly available information and attempts to successfully complete a protocol with the other members by impersonating a party. Recall that a key agreement protocol is successful if each of the members accept the identity of the other and terminate with the same key.

**Lemma 2** The probability for adversaries to successfully impersonate a legitimate member depends on the probability for finding or guessing $t$ and the secret key of the legitimate member.

*Proof.* In our proposed protocol, the adversary who will impersonate a member of the group (for example member $U_i$) may choose his/her random number $n_i'$, performing Gilboa's method with the leader for obtaining his/her additive share $x_i'$ and calculating $y_i'$. However, since in step (6) of the protocol (see Section 3) he has to sign $y_i$ with the secret key without any knowledge of $U_i$'s secret key, he can not pass this step easily. The adversary also can not calculate the value of $K$ and his individual key $K_i$ because he/she does not have the value of $t$. Hence, he can not pass the last step of the protocol. This means that to pass the last step of the protocol, he has to guess $t$. If the guesses are successful, the adversary can pass the protocol, and it means that the impersonation has been performed successfully. Otherwise, the adversary will be excluded from the group, and the other members will execute the protocol from the beginning. Thus, the probability that the intruder can impersonate the legitimate member depends on the difficulty of finding $t$ as well.

We can conclude that the probability for impersonating a legitimate member $U_i$ depends on the probability of finding $t$ and $U_i$'s secret key. Suppose the size of $t$ is $|t|$ and the probability for finding $U_i$'s secret key is $\delta$, then the probability for breaking the protocol is $[\delta/(2^{|t|})]$. □

### 4.3 Security Goals Achievable by the Protocol

In this section we discuss the security goals which are achievable by our proposed protocol.

**Theorem 1** Assume that all members of the group (including the leader) are honest. The protocol attains the following security goals:

(1) Key Freshness:
All members possess $K$ which they can verify is fresh/new.

(2) Key Confidentiality:
It is infeasible to find $K$ by eavesdropping on the protocol, even if the protocol is repeated many times.

(3) Group Authentication:
The leader decides who will obtain his share, while the other members receive an authenticated message regarding which other members have obtained it.

(4) Key Confirmation:
To make one member sure that the other members possess the same common key.

*Proof.*

(1) Key Freshness:
It is clear that $K$ can be derived from the value of $y_i$, $x_1$, $B$ and $dum_i$ (see Eq. (2)). Thus, $K$ is fresh/new as long as the four variables which determine $K$ are new. The freshness is probabilistic.

(2) Key Confidentiality:
After broadcasting $y_i$, all outsiders know $y_2, \ldots, y_n$. Since for calculating $K$ the outsiders need the value of $(\alpha^{((r-1)x_1+dum_1)-\sum_{i=2}^{i=r} dum_i} \bmod \beta)$, then the outsiders can not calculate $K$ without guessing $(\alpha^{((r-1)x_1+dum_1)-\sum_{i=2}^{i=r} dum_i} \bmod \beta)$. As has been mentioned in Section 4.1, the probability of successfully guessing in passive attacks is $1/(\lambda(\beta)-1)$. Thus, we conclude that to preserve the key confidentiality, $t$ and $\theta$ should be changed after $(\lambda(\beta)-2)$ repetitions. To strengthen the protocol, we can use $\theta$

which is a product of two or three strong large prime numbers. Thus, the key confidentiality is preserved as long as the requirement in Section 4.1 is fulfilled.

(3) Group Authentication:
Since before the execution of the protocol all members sent their public keys to the leader, the leader can decide that only members who sent him their public keys will obtain his share $(\alpha^{((r-1)x_1+dum_1)-\sum_{i=2}^{i=r} dum_i} \bmod \beta)$ and each member receives the authenticated share of the leader. This means that group authentication holds.

(4) Key Confirmation:
At the end of the protocol each member (including the leader) can check whether other members of the group are holding the same $K$ by comparing $K$ which is calculated from the broadcasted messages of all members and the value which is calculated by himself. This proves that the key confirmation is satisfied. □

## 5. Comparison with Previous Key Agreement Protocols

Suppose that for the key confirmation each member of the group (including the leader) has to broadcast the conference key signed (asymmetrically) with each member's secret key. Thus, for verifying that all members of the group (including the leader) hold the same conference key, each member has to decrypt all encrypted messages which are sent by the other members. Using the previous protocols proposed by Boyd [3),4)], Burmester-Desmedt [5)], Just-Vaudenay [11)], Steiner-Tsudik-Waidner [15),16)], Ateniese-Steiner-Tsudik [2)], Li-Pieprzyk [12)], this will contribute a high computational burden for all members of the group in the key confirmation if they use a signature such as RSA signature to sign the conference key. This is because each member has to decrypt several signed messages, i.e., each member has to perform several (asymmetric) decryptions.

In our proposed protocol each member signs/encrypts (asymmetrically) the established conference key using his individual key and broadcasts it. Furthermore, each member multiplies all encrypted messages and decrypts (asymmetrically) the result using the multiplicative inverse of the conference key. A member can verify whether all members have a conference key with the same value by comparing the conference key which is signed by all members and the one he has calculated. This will reduce the computational burden of each member for the key confirmation, since using our proposed protocol each member needs to perform only one (asymmetric) decryption.

This method can also be used for signing a contract/agreement. Suppose that a member of a group $U_i$ is going to propose a project, and he asks other members for approval of the proposal. Using the previous schemes, $U_i$ has to do $(r-1)$ verifications (for a group with $r$ members) to check whether all members of the group agree or not. It means that $U_i$ has to do $(r-1)$ (asymmetric) decryptions. This will increase the computational burden for $U_i$.

In our proposed scheme, each member of the group has his/her own individual secret key which can be used to sign his/her agreement. So, all members will sign the agreement and $U_i$ will verify all members agreement by doing **one** verification which means that he/she needs to perform only one (asymmetric) decryption for verifying all members' agreements.

Using our proposed protocol, the individual key can be used by a member to communicate with another member securely via the leader while the leader can not gain any knowledge about the messages sent among his members. The protocol of Li-Pieprzyk also has this capability, but our proposed protocol is more efficient than Li-Pieprzyk's. The conference key is established in Li-Pieprzyk's protocol with each member performing about $(2r^2 + 3r + 1)$ exponentiations. On the other hand, in our proposed method, each member needs to perform no more than about 10 exponentiations and $(l + u)$ oblivious transfers.

## 6. Conclusion

We proposed a new key agreement protocol which is based on oblivious transfer. Our proposed scheme introduced individual keys which can be used by each user to sign a common message, which is not included in previously proposed protocols. By using individual keys, we can reduce the number of verifications and also reduce the computational burden for the verifier. Using this individual key, a member can communicate securely with other members via the leader, while the leader can not gain any knowledge about the message sent among his members.

Our proposed scheme will be secure as long as the value of $t$ and the secret number chosen by each member are kept secret.

The protocol can be strengthened by choosing $\theta$ to be a product of two or three strong large prime numbers.

## References


1) Adler, A. and Coury, J.E.: *The Theory of Numbers*, Jones and Barlett (1995).
2) Ateniese, G., Steiner, M. and Tsudik G.: Authenticated Group Key Agreement and Friends, *Proc. 5th ACM Conference on Computer and Communication Security*, pp.17–26 (1998).
3) Boyd, C.: On Key Agreement and Conference Key Agreement, *Proc. Information Security and Privacy Australasian Conference* (*ACISP*), LNCS, Vol.1270, pp.294–302, Springer-Verlag (1997).
4) Boyd, C.: Towards a Classification of Key Agreement Protocols, *Proc. Computer Security Foundation Workshop*, pp.38–43 (1995).
5) Burmester, M. and Desmedt, Y.G.: Efficient and Secure Conference Key Distribution, *Proc. Security Protocols International Workshop*, Cambridge, United Kingdom, LNCS, Vol.1189, pp.119–129, Springer-Verlag (1996).
6) Davies, D. and Price, W.: *Security for Computer Networks*, John Wiley and Sons (1989).
7) Even, S., Goldreich, O. and Lempel, A.: A Randomized Protocol for Signing Contract, *Comm. ACM*, Vol.28, No.6, pp.637–647 (1985).
8) Gilboa, N.: Two RSA Key Generation, *Crypto 99*, LNCS, Vol.1666, pp.116–129, Springer-Verlag (1999).
9) Goldreich, O.: Secure Multy-Party Computation, *Working Draft*, http://www.wisdom.eizmann.ac.il/õded/pp.html/prot.ps (1998).
10) Ingemarsson, I. and Tang. D.T.: A Conference Key Distribution System, *IEEE Trans. Information Theory*, Vol.28, No.5, pp.714–720 (1982).
11) Just, M. and Vaudenay. S.: Authenticated Multy-Party Key Agreement, *Advances in Cryptology ASIACRYPT '96*, LNCS, Vol.1163, pp.36–49, Springer-Verlag (1996).
12) Li, C. and Pieprzyk, J.: Conference Key Agreement from Secret Sharing, *Proc. ACISP 1999*, Springer-Verlag, pp.64–76 (1999).
13) Rabin, M.: How to Exchange Secrets by Oblivious Transfer, Tech. Memo TR-81, Aiken Computation Laboratory, Harvard University (1981).
14) Rivest, R.L., Shamir, A. and Adleman L.: Method for Obtaining Signatures and Public-Key Cryptosystems, *Comm. ACM*, Vol.21, No.2, pp.120–126 (1978).
15) Steiner, M., Tsudik, G. and Waidner, M.: Diffie-Hellman Key Distribution Extended to Group Communication, *Proc. Third ACM Conference on Computer Communications*, pp.31–37 (1996).
16) Steiner, M., Tsudik, G. and Waidner, M.: CLIQUES: A New Approach to Group Key Agreement, *Proc. 18th International Conference on Distributed Computing Systems* (*ICDCS 98*), pp.380–387 (1998).
17) Stinson, D.: *Cryptography Theory and Practice*, CRC (1996).


## Appendix

### A.1 Calculation of $K$

This appendix shows why the conference key $K$ calculated by each member using the proposed protocol is correct, i.e., $K = \prod_{i=1}^{i=r} K_i$.

We first review the messages sent by each party in the proposed protocol:

- After the first message, all members obtain their additive share with the leader $x_i$, such that $x_1 + x_i = n_1 n_i$.
- After Message 2, the leader obtains all member's chosen numbers $dum_i$.
- After Message 3, the leader obtains all member's share

  $$\alpha^{x_i} \bmod \beta \equiv y_i \alpha^{dum_i} \bmod \beta$$

  and calculates the conference key

  $$K \equiv \alpha^{((r-1)x_1 + dum_1 + \sum_{i=2}^{i=r} x_i)t} \bmod \beta$$

  At this point, the conference key $K$ is the product of all members' individual keys:

  $$\begin{aligned} K &\equiv [\alpha^{((r-1)x_1 + dum_1 + \sum_{i=2}^{i=r} x_i)} \bmod \beta]^t \\ &\equiv [\alpha^{[\sum_{i=2}^{i=r}(x_i + x_1)] + dum_1} \bmod \beta]^t \\ &\equiv [\alpha^{[\sum_{i=2}^{i=r}(n_i n_1)] + dum_1} \bmod \beta]^t \\ &\equiv [\alpha^{[\sum_{i=2}^{i=r}(n_i n_1)t] + dum_1 t} \bmod \beta] \\ &\equiv [\alpha^{[\sum_{i=2}^{i=r}(n_i n_1)t] + K_1} \bmod \beta] \end{aligned}$$

  Since the individual key of member $U_i$ is :

  $$K_i \equiv \alpha^{n_1 n_i t} \bmod \beta$$

  then the conference key $K$ is exactly :

  $$K \equiv \prod_{i=1}^{i=r} K_i \bmod \beta$$

  which is the product of all members' (including the leader) individual keys.
- After Messages 4 and 5, each member can calculate the conference key $K$ using Eq. (2).

- Finally, each member can verify whether all members hold the same conference key by multiplying all signed messages ($K^{K_i \bmod \beta} \bmod \theta$) as follows:

$$PRODUCT \equiv \prod_{i=1}^{i=r} K^{K_i \bmod \beta} \bmod \theta$$
$$\equiv K^{K \bmod \beta} \bmod \theta$$

and decrypts $PRODUCT$ using the multiplicative inverse of the locally calculated $K$:

$$K \equiv (PRODUCT)^{K^{-1} \bmod \beta} \bmod \theta$$
$$\equiv K^{(K)(K^{-1} \bmod \beta)} \bmod \theta$$
$$\equiv K^{1 \bmod \beta} \bmod \theta$$

(Received May 10, 2000)
(Accepted November 2, 2000)

**Ari Moesriami Barmawi** received the B.E. in Electronic Engineering from Bandung Institute of Technology (Indonesia), in 1985. She was a system engineer in Computer Centre of Indonesian Aircraft Industry during 1986–1992. Since 1993 she has been a lecturer of Bandung Institute of Technology. She received M.E. in Computer Science from Keio University in 1997. Her interest is in computer security.

**Shingo Takada** received the B.E. in electrical engineering, and M.E. and Ph.D. in Computer Science from Keio University, Yokohama, Japan in 1990, 1992, 1995, respectively. From 1995 to 1999, he was a research associate at Nara Institute of Science and Technology. He is currently an assistant professor at Keio University (Faculty of Science and Technology). His interests include software engineering, information exploration, and cognitive science. He is a member of the IPSJ, JSSST, IEICE, JSAI, ACM, and IEEE-CS.

**Norihisa Doi** has been a professor in the Department of Computer Science at Keio University since 1986. He received a B.E., M.E., and Ph.D. in computer science from Keio University in 1964, 1966, and 1975, respectively. During 1975–76, he was a visiting faculty member in the Department of Computer Science at Carnegie-Mellon University. In 1976, he was a visiting professor in the Computer Communication Networks Group at the University of Waterloo. He was a member of the executive boards of the Japan Society for Software Science and Technology (JSSST), the Information Processing Society of Japan (IPSJ), the Japanese Society for Artificial Intelligence (JSAI), and the Japan Society for Security Management, the chief editor of the transactions of the JSSST, the chief editor of the transactions of the IPSJ, the head of the Japanese delegation of ISO/IEC JTC1/SC22. Currently, he is a member of the Science Council of Japan, the chairman of the National Committee of Information and Computer Science, a chairman of OMG (Object Management Group) Japan-SIG, a councilor for the Japan Society for Software Science and Technology, the chairman of the Technical Committee of the Information Technology Promotion Agency (IPA), the chairman of the Committee for Investigating the Standardization of Agent-Oriented Computing of the Japanese Standards Association, and the chairman of the Business Object Consortium.