A Secure Fault-Tolerant Conference-Key Agreement Protocol

Wen-Guey Tzeng

Abstract—When a group of people want to communicate securely over an open network, they run a conference-key protocol to establish a common conference key K such that all their communications thereafter are encrypted with the key K. In this paper, we propose a provably secure fault-tolerant conference-key agreement protocol under the authenticated broadcast channel model. We show that a passive adversary gets zero knowledge about the conference key established by the honest participants under the assumption of a variant Diffie-Hellman decision problem. We also show that the honest participants can agree on a common conference key no matter how many participants are malicious. Furthermore, we show that even if the broadcast channel is not authenticated, our protocol is secure against impersonators under the random oracle model.

Index Terms—Conference key, provable security, fault tolerance.

1 Introduction

When a group of people want to communicate securely over an open network, they run a conference-key protocol to establish a common conference key K such that all their communications thereafter are encrypted with key K. The first type of conference-key protocol, called conference-key distribution, is that a chairman selects a conference key and distributes the key to the participants. The second type of conference-key protocol, called conference-key agreement, is that all participants together compute a common key without a chairman. The latter one is suitable for distributed environments. Conference-key protocols are also designed for various types of network connection, such as the ring connection, the star connection, the broadcast connection, etc. The conference keys of a conference-key protocol are either predistributed or dynamic. The conference key is fixed for a particular group of participants in a predistributed conference-key protocol, while it is different for each session in a dynamic conference-key protocol. The predistributed conference-key protocol often lacks of flexibility.

In this paper, we propose a provably secure fault-tolerant conference-key agreement protocol under the authenticated broadcast channel model. The adversary that attacks our protocol can be either active or passive. An active adversary (malicious participant) tries to disrupt establishment of a common conference key among the honest participants, while a passive adversary tries to learn the conference key by listening to the communication of participants. We tolerate the case that a malicious participant gets the conference key since the malicious participant can simply behave properly to get the key. We consider fault tolerance because, sometimes, the conference is emergent and delay or destruction of the conference can cause serious damage. We show that a passive adversary gets no information (zero knowledge) about the common conference key established by the honest participants under the assumption of a variant Diffie-Hellman decision problem. We also show that the honest participants can agree on a common conference key no matter how many participants are malicious.

We can relax the requirement of the broadcast channel being authenticated. An attacker may try to impersonate a participant if the broadcast channel is not authenticated. Following a general practice in provable security, we show that our protocol is secure against impersonators in the random oracle model [1]. The basic technique is to use a signature scheme that is "existentially unforgeable" [24].

Computing a conference key is a special case of secure multiparty computation in which a group of people evaluate a function $f(k_1, k_2, \cdots)$ securely, with each person possessing a private input k_i. Therefore, it is possible to have a secure conference-key agreement protocol by the generic construction for secure multiparty computation. However, there are some distinct features in the conference-key agreement protocol. First, there are no private channels between participants, which is a general assumption in secure multiparty computation. Second, a cheater's goal in a conference-key agreement protocol is to disrupt conference-key establishment among the honest participants. This is quite different from the goal of cheaters in secure multiparty computation. Third, in multiparty computation, when a cheater is found, the cheater's secret x_i, which is shared into others, is recovered by honest participants so that evaluation can proceed. In conference-key agreement, since a cheater's (session) secret is not a necessity in computing a conference key, the cheater is simply excluded from participating when found.

There has been intensive research on conference-key protocols. For example, conference-key distribution protocols (with a chairman) have been studied in [4], [10], [11],

• *The author is with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan 30050. E-mail: tzeng@cis.nctu.edu.tw.*

Manuscript received 27 Jan. 2000; revised 25 July 2001; accepted 26 July 2001.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 111322.

[18], predistributed conference-key protocols have been studied in [5], [6], [21], and conference-key agreement protocols have been studied in [16], [18], [19], [30], [31]. Most proposed protocols focus on privacy of conference keys and message efficiency for various types of network connection. Nevertheless, they do not have the capability of fault tolerance so that a malicious participant can easily mislead other participants to compute different keys. On the other hand, Klein et al. [17] proposed a fault-tolerant conference-key agreement protocol. However, the protocol is quite inefficient and its security is not rigidly proven. Burmester and Desmedt [8] proposed an efficient (two-round) protocol (Protocol 3) for the broadcast channel with $f(k_1, k_2, \ldots, k_n) = g^{k_1k_2+k_2k_3+\cdots+k_nk_1} \bmod p$. They showed that an eavesdropper cannot compute the common conference key if the Diffie-Hellman decision problem is intractable. In contrast, an eavesdropper gets zero knowledge about the common conference key in our protocol if a variant Diffie-Hellman decision problem is hard. In the modified Protocol 7 (authenticated key distribution), they used an interactive proof for authenticating sent messages to show that the protocol is secure against impersonators. Nevertheless, the number of rounds in the protocol is proportional to the number of participants. Furthermore, both protocols cannot withstand the attack of malicious participants.

2 Model

A user in the system is a probabilistic polynomial-time Turing machine. Each user U_i has a secret information x_i and a corresponding public information y_i. The system has a public directory that records the system's public parameters and each user's public information that can be accessed by everyone. All users are connected by an authenticated broadcast network such that the messages sent on the network can be identified and cannot be altered, blocked, or delayed. Therefore, everyone can send and receive the message on the network without interruption. No private channel exists between users. A group of users who want to establish a conference key is called the *set of participants*. A participant may be malicious in any way.

There are two types of adversaries that are both probabilistic polynomial-time Turing machines. A *passive adversary* who is not a participant listens to the broadcast channel and tries to learn the conference key established by the honest participants. An *active adversary* who is a participant tries to disrupt establishment of a common conference key among the honest participants. An active adversary mainly sends "malicious" messages into the broadcast channel to fool an honest participant into believing that he has computed the same conference key as that of other honest participants, while he has not indeed. We don't care about the possibility that two or more cheating participants collaborate and result in one of them or other malicious participants not being able to compute the key. This includes the following case: A malicious participant U_i sends "malicious" messages, but all honest participants compute the same key. Another malicious participant, U_j, though receiving an incorrect key, still claims that he has received the correct key. We tolerate this case since this type of collaboration between U_i and U_j does no harm to the honest participants.

A conference-key agreement protocol is secure if it can withstand attacks of passive and active adversaries. For security against a passive adversary, we mean that the adversary alone can construct a view that is computationally indistinguishable from the real conversation occurred among the participants. For security against an active adversary, we mean that if the adversary does not follow the protocol in *any* way, the probability that he can disrupt establishment of a common conference key among honest participants is negligible.

3 Design Principles

Our protocol is component-based, that is, our protocol uses cryptographic modules as building blocks. Component-based design has many merits. First, because of using modular design, it is easy to upgrade the components of the protocol in case better components, in either efficiency, cost, or security, are available. Also, in case security flaws are found in a component, we can replace the component only and need not abandon the whole established system. Second, it is easier to apply strong security analysis on the protocol. Since each component has a single security goal, we can analyze each component in the focused security features. Third, it is flexible and suitable for use in a large system. A conference may be called among the participants all over the world. Flexibility of component-based design allows each user to choose adequate components for each conference session. Therefore, component-based design is suitable for large and heterogeneous systems.

The idea of our design is to decompose the target function f of the secure multiparty computation into n subfunctions f_i, $1 \le i \le n$. Each participant, U_i, handles one subfunction, f_i, independently. If participant U_i sends out messages such that any other participant cannot evaluate f_i, U_i is a cheater. The other participants exclude U_i from participation and restart the protocol. This process continues until all cheaters are found.

Suppose that each participant U_i holds a secret, x_i, $1 \le i \le n$. They need to evaluate a function f to get the conference key $K = f(k_1, k_2, \ldots, k_n)$, where k_i is a randomly selected secret (subkey) of U_i for the conference session. In our protocol, we let each participant U_i handle a function f_i and the conference-key function is

$$f(k_1, k_2, \ldots, k_n) = \Sigma f_i(k_1, k_2, \ldots, k_n),$$

where $f_i(k_1, k_2, \ldots, k_n) = k_i$. Since the result k_i of f_i is independent of other parameters $k_j, j \neq i$, participant U_i can broadcast messages so that other participants can evaluate f_i in a secure computation way.

As mentioned previously, our protocol is component-based. It contains the following components:

1. Component of secure multiparty computation for f_i,
2. Component of k_i commitment and verification.

Our conference key agreement protocol has the following four stages:

1. **Secret distribution and commitment:** Using the paradigm of secure multiparty computation, each participant U_i broadcasts $\vec{w}_i$ so that any participant U_j can compute $f_i(\ldots, k_i, \ldots) = k_i$ from $\vec{w}_i$ and his secret x_j. Since the computation is secure, no passive adversary shall get any information about k_i. Also, U_i broadcasts $\vec{c}_i$ that commits to k_i so that other participants can verify the correctness of k_i.
2. **Subkey computation and verification:** U_i computes k'_j of all other participants $U_j, j \neq i$. When U_i gets k'_j, he can use $\vec{c}_j$ to check whether k'_j is correct.
3. **Fault detection:** If the above verification is not correct, U_i asks U_j to reveal information about commitment $\vec{c}_i$ and messages $\vec{w}_i$ so that all participants can determine whether U_i is cheating. If U_i detects a cheater, he deletes the cheater from his participant set and restarts the protocol.
4. **Conference-key computation:** When no faults are detected, add all subkeys together to get the conference key.

Actually, each participant U_i can use a different method for securely computing f_i and committing to k_i as long as the methods are known by other participants.

4 A Concrete Protocol

The system has public parameters:

- p: a large prime number that is $2q + 1$, where q is also a large prime.
- H: a one-way permutation from Z_q to Z_q.
- g: a generator for the subgroup $G_q = \{i^2 | i \in Z_p^*\}$ of quadratic residues of Z_p^*.

Each user U_i has two parameters:

- Private parameter x_i: a number in Z_q^*.
- Public parameter $y_i = g^{x_i} \bmod p$. Since q is a prime number, y_i is a generator for G_q.

The protocol starts with an initiator calling for a conference for a set U of participants. Without loss of generality, let $U = \{U_1, U_2, \ldots, U_n\}$ be the initial participant set. Each U_i, $1 \leq i \leq n$, knows U.

1. **Secret distribution and commitment:** Each participant U_i does the following:
   a. Randomly select $R_i, K_i \in Z_q, S_i \in Z_q^*$.
   b. Compute a polynomial $h_i(x)$ (over Z_q) of degree n that passes points $(j, y_j^{R_i} \bmod p \bmod q)$, $1 \leq j \leq n$, and $(0, K_i)$.
   c. Compute and broadcast

$$w_{ij} = h_i(n + j) \bmod q, 1 \leq j \leq n,$$
$$\alpha_i = g^{R_i} \bmod p,$$
$$\gamma_i = g^{S_i} \bmod p,$$
$$\delta_i = S_i^{-1}(H(K_i) - \gamma_i x_i) \bmod q.$$

2. **Subkey computation and verification:** Each participant U_i does the following for $j \neq i$:
   a. On receiving w_{jl}, $1 \leq l \leq n$, and α_j, compute polynomial $h'_j(x)$ (over Z_q) of degree n that passes $(n + l, w_{jl}), 1 \leq l \leq n$ and $(i, \alpha_j^{x_i} \bmod p \bmod q)$.
   b. Let $K'_j = h'_j(0) \bmod q$.
   c. Check whether (γ_j, δ_j) is the ElGamal signature of $H(K'_j)$ by U_j, i.e., check whether $g^{H(K'_j)} \bmod p = y_j^{\gamma_j} \gamma_j^{\delta_j} \bmod p$. If so, broadcast V_{ij} = "success." Otherwise, broadcast V_{ij} = "failure."
3. **Fault detection:** Each participant U_i does the following for $j \neq i$:
   a. On receiving V_{ji} = "failure" for some U_j: U_j claims that U_i itself is faulty.
      i. Output R_i, K_i, S_i.
   b. On receiving V_{jm} = "failure": U_j claims that U_m, $m \neq i$, is faulty.
      i. Wait for U_m's fault detection messages R_m, K_m, S_m.
      ii. If U_m's fault detection messages are not received, set U_m as a malicious participant.
      iii. On receiving R_m, K_m, S_m, check whether $w_{ml}, 1 \leq m \leq n$, α_m, γ_m, and δ_m are correct, i.e., check whether $\alpha_m = g^{R_m} \bmod p$, whether there is an n-degree polynomial over Z_q passing points $(0, K_m)$, $(l, y_l^{R_m} \bmod p \bmod q)$, and $(n + l, w_{ml})$, $1 \leq l \leq n$, and whether (γ_m, δ_m) is the ElGamal signature of U_m on $H(K_m)$. If so, set U_j as a malicious participant. Otherwise, set U_m as a malicious participant.
   c. Restart the protocol by deleting malicious participants from his participant set U.
4. **Conference-key computation:** If no faults are detected in the fault detection stage, each participant U_i computes the conference

$$K = (K'_{i_1} + K'_{i_2} + \cdots + K'_{i_m}) \bmod q,$$

where the current participant set is

$$U' = \{U_{i_1}, U_{i_2}, \ldots, U_{i_m}\}.$$

5 Security Analysis

We show security of the above protocol in correctness, fault tolerance, and withstanding the attack of passive adversaries.

5.1 Correctness and Fault Tolerance

For correctness (completeness) of our protocol, we show that if all participants follow the protocol, they compute a common conference key.

Theorem 5.1 (Correctness). *If all participants follow the protocol, they compute a common conference key.*

Proof. From the broadcast messages of participant U_j, participant U_i can compute the polynomial $h_j(x) \bmod q$ passing points $(n + l, w_{jl})$, $1 \leq l \leq n$, and

$$(i, \alpha_j^{x_i} \bmod p \bmod q).$$

U_i then computes $K_j = h_j(0) \bmod q$. By verification messages γ_j and δ_j, U_i can check whether K_j is correct. Since, for fixed γ_j and δ_j, the signed text $H(K_j) \in Z_q$ is unique, all participants compute the same K_j. Thus, they compute the same conference key $K = (K_1 + K_2 + \cdots + K_n) \bmod q$. □

For fault tolerance (robustness), we show two things:

1. Any malicious participant U_i who tries to cheat honest participants into accepting different K_i will be excluded from the participant sets of all honest participants.
2. An honest participant will not be excluded from the participant set of any other honest participant.

Note that it does not matter that a malicious U_i causes other malicious participants to compute different K_i.

Lemma 5.2. *Any malicious participant U_i who tries to cheat honest participants into accepting different K_i shall be excluded from the participant sets of all honest participants.*

Proof. Malicious participants can deviate from the protocol in two ways. First, a malicious participant U_i sends "wrong" $w_{il}, 1 \le l \le n$, α_i, γ_i, and δ_i so that two honest participants U_j and U_m compute different K_i. In this case, one of them, say U_j, shall send V_{ji} = "failure" since γ_i and δ_i cannot be the ElGamal signature of two different K_is. Then, U_i has to broadcast R_i, K_i, and S_i for verification. Every honest participant verifies whether $\alpha_i = g^{R_i} \bmod p$, (γ_i, δ_i) is the signature of $H(K_i)$ and the polynomial passing $(n+l, w_{il}), 1 \le l \le n$, and $(0, K_i)$ also passes points $(j, y_j^{R_i} \bmod p \bmod q), 1 \le j \le n$. Since the honest U_j claims that K_i is wrong, for all participants, at least one of the above check cannot hold. Therefore, all honest participants exclude U_i from their participant sets.

Second, U_i sends V_{ij} = "failure" to claim that U_j is malicious, while U_j is indeed honest. In this case, U_j broadcasts R_j, K_j, and S_j to prove his honesty. Since U_j is honest, all honest participants decide that U_i is malicious. Therefore, the malicious U_i is excluded by all honest participants. □

Lemma 5.3. *No honest participant excludes any other honest participant from his participant set.*

Proof. Since an honest participant U_i follows the protocol, his broadcast messages make all participants compute the same K_i. Even if some malicious participant U_j claims that he is faulty, he can send R_i, K_i, and S_i to prove his honesty. Therefore, no honest participant shall exclude U_i from his participant set. □

By the above two lemmas, we can show that all honest participants compute the same conference key even if the majority of the participants are malicious.

Theorem 5.4 (Fault tolerance). *All honest participants have the same participant set and thus compute the same conference key no matter how many participants are malicious.*

Proof. By the above two lemmas, each honest participant's participant set consists of two types of participants: honest participants and those participants U_i who, though deviating from the protocol, make all honest participants compute the same K_i. Therefore, all honest participants compute the same conference key. □

5.2 Security against Passive Attackers

A passive attacker (eavesdropper) tries to learn information about the conference key by listening to the broadcast channel. We show that an eavesdropper cannot get any information about K_i of U_i by demonstrating that the attacker's view of the messages broadcast by U_i on the broadcast channel can be simulated without knowing secrets x_i and K_i.

We use tuples of random variables to model the attacker's view and the simulated one. We say that the attacker's view and the simulated view (transcript) are computationally indistinguishable if no probabilistic polynomial-time algorithm can tell the sampling source by observing given samples. Since the simulated view is constructed without knowing U_i's secrets and is computationally indistinguishable from the real view, the attacker gets no information about U_i's secrets.

We need an assumption to show that the simulated transcript is computationally indistinguishable from the real one. This assumption is a little stronger than that about the regular Diffie-Hellman decision problem that is discussed in some papers [7], [23], [29]. The assumption about the regular Diffie-Hellman decision problem is that, for any given $y_1, y_2 \in G_q - \{1\}$, the following two tuples of random variables,

$$(y_1, y_2, y_1^R \bmod p, y_2^R \bmod p)$$

and

$$(y_1, y_2, u_1, u_2),$$

are computationally indistinguishable, where R is randomly chosen from Z_q and u_1 and u_2 are randomly chosen from G_q. Therefore,

$$(y_1, y_2, y_1^R \bmod p \bmod q, y_2^R \bmod p \bmod q)$$

and

$$(y_1, y_2, u_1 \bmod q, u_2 \bmod q)$$

are computationally indistinguishable. Note that y_1 and y_2 must be quadratic residues of Z_p^*; otherwise, one can tell the above probability distributions apart. We note that $u_1 \bmod q$ and $u_2 \bmod q$ do not range all over Z_q. Therefore, we need an assumption about a variation of the Diffie-Hellman decision problem.

Assumption 1 (Variant Diffie-Hellman decision problem). *Let $p = 2q + 1$ and G_q be the quadratic-residue subgroup of Z_p^*. Given any generators $y_1, y_2 \in G_q - \{1\}$, the following two random-variable tuples are computationally indistinguishable:*

$$(y_1, y_2, y_1^R \bmod p \bmod q, y_2^R \bmod p \bmod q)$$

and

$$(y_1, y_2, u_1, u_2),$$

where $R, u_1, u_2 \in Z_q$.

The simulator of the adversary's view on broadcast messages of U_i does the following:

1. Randomly select $w'_{ij} \in Z_q, 1 \le j \le n$, $R'_i \in Z_q$, $S'_i \in Z_q^*$, $\delta'_i \in Z_q$,
2. Output the simulated transcript:

$$w'_{i1}, w'_{i2}, \ldots, w'_{in}, \alpha'_i, \gamma'_i, \delta'_i,$$

where $\alpha'_i = g^{R'_i} \bmod p$ and $\gamma'_i = g^{S'_i} \bmod p$.

We now show, on random variables $K_i, R_i \in Z_q, S_i \in Z_q^*$, the real view

$$(w_{i1}, w_{i2}, \ldots, w_{in}, \alpha_i, \gamma_i, \delta_i)$$

and, on random variables

$$w'_{ij} \in Z_q, 1 \le j \le n, R'_i \in Z_q, S'_i \in Z_q^*, \delta'_i \in Z_q,$$

the simulated view

$$w'_{i1}, w'_{i2}, \ldots, w'_{in}, \alpha'_i, \gamma'_i, \delta'_i$$

are computationally indistinguishable, where

$$\begin{aligned}
\alpha_i &= g^{R_i} \bmod p,\\
\gamma_i &= g^{S_i} \bmod p,\\
\delta_i &= S_i^{-1}(H(K_i) - \gamma_i x_i) \bmod q,\\
\alpha'_i &= g^{R'_i} \bmod p,\\
\gamma'_i &= g^{S'_i} \bmod q,
\end{aligned}$$

and $w_{ij} = h_i(n+j), 1 \le j \le n$, is described in our protocol. Since, for any $\gamma_0 \in G_q - \{1\}$ and $\delta_0 \in Z_q$,

$$\Pr[\gamma_i = \gamma_0, \delta_i = \delta_0] = \Pr[\gamma'_i = \gamma_0, \delta'_i = \delta_0] = \frac{1}{q(q-1)},$$

we only have to consider the probability distributions for any $\vec{\beta}$,

$$\Pr[(w_{i1}, w_{i2}, \ldots, w_{in}, \alpha_i) = \vec{\beta} | \gamma_i = \gamma_0, \delta_i = \delta_0]$$

and

$$\Pr[(w'_{i1}, w'_{i2}, \ldots, w'_{in}, \alpha'_i) = \vec{\beta} | \gamma'_i = \gamma_0, \delta'_i = \delta_0].$$

For any fixed γ_0 and δ_0, the random variable K_i is fixed, say k_0. We have

$$\begin{aligned}
&\Pr[(w_{i1}, w_{i2}, \ldots, w_{in}, \alpha_i) = \vec{\beta} | \gamma_i = \gamma_0, \delta_i = \delta_0]\\
&= \Pr[(w_{i1}, w_{i2}, \ldots, w_{in}, \alpha_i) = \vec{\beta} | K_i = k_0]
\end{aligned}$$

and

$$\begin{aligned}
&\Pr[(w'_{i1}, w'_{i2}, \ldots, w'_{in}, \alpha'_i) = \vec{\beta} | \gamma'_i = \gamma_0, \delta'_i = \delta_0]\\
&= \Pr[(w'_{i1}, w'_{i2}, \ldots, w'_{in}, \alpha'_i) = \vec{\beta}].
\end{aligned}$$

We show that they are computationally indistinguishable.

Lemma 5.5. *Under Assumption 1, for any fixed $K_i = k_0 \in Z_q$, on random variables $R_i, R'_i, w'_{i1}, w'_{i2}, \ldots, w'_{in} \in Z_q$,*

$$(w_{i1}, w_{i2}, \ldots, w_{in}, \alpha_i)$$

and

$$(w'_{i1}, w'_{i2}, \ldots, w'_{in}, \alpha'_i)$$

are computationally indistinguishable.

Proof. By the assumption

$$(y_1, y_2, \ldots, y_n, y_1^{R_i} \bmod p \bmod q, y_2^{R_i} \bmod p \bmod q, \ldots, y_n^{R_i} \bmod p \bmod q, g^{R_i} \bmod p)$$

and

$$(y_1, y_2, \ldots, y_n, u_1, u_2, \ldots, u_n, g^{R'_i} \bmod p)$$

are computationally indistinguishable, where $R_i, R'_i \in Z_q$ and $u_j \in Z_q, 1 \le j \le n$. Let h_i (over Z_q) be the n-degree polynomial passing points $(0, k_0)$ and (j, u_j), $1 \le j \le n$. By applying a polynomial interpolation on them, we have that

$$(w_{i1}, w_{i2}, \ldots, w_{in}, g^{R_i} \bmod p)$$

and

$$(\bar{w}_{i1}, \bar{w}_{i2}, \ldots, \bar{w}_{in}, g^{R'_i} \bmod p)$$

are computationally indistinguishable, where $\bar{w}_{ij} = \bar{h}_i(n+j) \bmod q, 1 \le j \le n$. Since, for any $\bar{w}^0_{ij} \in Z_q$, $1 \le j \le n$, and $\bar{\alpha}_0 \in G_q$,

$$\begin{aligned}
&\Pr[(\bar{w}_{i1}, \bar{w}_{i2}, \ldots, \bar{w}_{in}, g^{R'_i} \bmod p) = (\bar{w}^0_{i1}, \bar{w}^0_{i2}, \ldots, \bar{w}^0_{in}, \bar{\alpha}_0)]\\
&= \Pr[(\bar{w}'_{i1}, \bar{w}'_{i2}, \ldots, \bar{w}'_{in}, g^{R'_i} \bmod p) = (\bar{w}^0_{i1}, \bar{w}^0_{i2}, \ldots, \bar{w}^0_{in}, \bar{\alpha}_0)]\\
&= \frac{1}{q^{n+1}},
\end{aligned}$$

thus $(w_{i1}, w_{i2}, \ldots, w_{in}, \alpha_i)$ and $(w'_{i1}, w'_{i2}, \ldots, w'_{in}, \alpha'_i)$ are computationally indistinguishable. □

Therefore, the simulator outputs a transcript that is computationally indistinguishable from the real one.

Theorem 5.6 (Privacy). *Under Assumption 1, for any $i, 1 \le i \le n$, the real communication transcript of U_i*

$$(w_{i1}, w_{i2}, \ldots, w_{in}, \alpha_i, \gamma_i, \delta_i)$$

and the simulated one

$$(w'_{i1}, w'_{i2}, \ldots, w'_{in}, \alpha'_i, \gamma'_i, \delta'_i)$$

are computationally indistinguishable, where random variables $R_i, K_i \in Z_q$, $S_i \in Z_q^*$, and $w'_{i1}, w'_{i2}, \ldots, w'_{in} \in Z_q$, $S'_i \in Z_q^*, \delta'_i \in Z_q$.

Proof. This is obvious by Lemma 5.5. □

6 Security against Impersonators

Another type of attackers is impersonators (outsiders) who want to impersonate participants. The "authentication" assumption for the broadcast channel is to deter these impersonators. In our protocol, we require the participant U_i to sign $H(K_i)$, instead of K_i. We note that an outsider, without knowing U_i's secret x_i, can sign a random message $m = -\gamma_i a b^{-1} \bmod p$ by choosing $\gamma_i = g^a y_i^b \bmod p$ and $\delta_i = -\gamma_i b^{-1} \bmod q$ first for $a \in Z_q$ and $b \in Z_q^*$ [22]. If we only require U_i to sign $K_i = m$, the impersonator can share K_i with other participants even though he cannot compute other participants' K_js.

We don't have a rigid proof for our protocol's strength against impersonators if the channel is not authenticated. Nevertheless, we give some explanation for the protocol's strength in this case. First, if the impersonator chooses K_i first and then signs $H(K_i)$, he has to sign a chosen message $H(K_i)$, which is not known to be possible in the ElGamal signature scheme. Second, if the impersonator chooses $m = H(K_i)$ first, he has to compute $K_i = H^{-1}(m)$ in order to share K_i with other participants. This occurs with only a negligible probability under H being a one-way permutation. Strong evidence shows that this approach (full-domain-hash-then-sign) is secure against signature forgery, thus impersonators [2].

In provable security, the random oracle model is sometimes adopted to demonstrate security of a protocol by assuming that the hash function is actually a random function. Under the random oracle model, we can relax the requirement of the "authenticated" broadcast channel since the signature part of our protocol is "existentially unforgeable" against the adaptively chosen message attack. Therefore, we show that, even if the broadcast channel is not authenticated, our protocol is secure against impersonators under the random oracle model.

Theorem 6.1. *Assume that the broadcast channel is not authenticated. If computing discrete logarithm modulo a prime is hard, our protocol is secure against an impersonator's adaptively chosen message attack under the random oracle model.*

Proof. The proof follows from that in [24] directly. Under the random oracle model, we assume that the one-way permutation H is a true random function, that is, $H(K_i)$ is an independent random variable from K_i. Note that, since the hash result is random now, we have to put it as a part of the signature. Also, since the ElGamal signature is "existentially forgeable," the chosen message query on K_i can be answered as follows: We first produce an existential forgery (h', γ', δ') and let h' be the hash result $H(K_i)$.

Assume that an impersonator can impersonate U_i. Then, he can sign K_i with a nonnegligible probability ϵ. By a probability argument, there exists a set Ω of (K_i, S_i) such that, for any particular (K_i, S_i) in Ω, the impersonator can sign K_i with S_i with a probability at least $\epsilon/2$. We use the impersonator to sign K_i into two different results $(K_i, \gamma_i, h_1, \delta_i^{(1)})$ and $(K_i, \gamma_i, h_2, \delta_i^{(2)})$ with a nonnegligible probability, where $\gamma_i = g^{S_i} \bmod p$, $\delta_i^{(1)} = S_i^{-1}(h_1 - \gamma_i x_i) \bmod q$, $\delta_i^{(2)} = S_i^{-1}(h_2 - \gamma_i x_i) \bmod q$, and h_1 and h_2 are the hash results of $H(K_i)$ under the random oracle model. Since the random factor γ_i is used twice, one can compute U_i's secret x_i, which is a contradiction. □

7 Conclusion

Assuming an authenticated broadcast channel, we have presented a conference-key agreement protocol that is provably secure against passive and active adversaries under the assumption of a variant Diffie-Hellman decision problem. We argue that our protocol is secure against impersonators if the full-domain-hash-then-sign paradigm for ElGamal signature is secure. Furthermore, we show that, even if the broadcast channel is not authenticated, our protocol is secure against impersonators under the random oracle model.

Our protocol is efficient. It uses only two rounds to compute a common conference key after all malicious participants are detected. Nevertheless, the size of messages that each participant sends is proportional to the number of participants. It is interesting to design a provably secure conference-key agreement protocol with both round and message-efficiency.

Acknowledgments

Research supported in part by the National Science Council grant NSC-88-2213-E-009-058 and by the Ministry of Education grant Excellence Project 90-E-FA04-1-4, Taiwan, Republic of China.

References

[1] M. Bellare and P. Rogaway, "Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols," *Proc. First ACM Conf. Computer and Comm. Security,* pp. 62-73, 1993.

[2] M. Bellare and P. Rogaway, "The Exact Security of Digital Signatures, How to Sign with RSA and Rabin," *Proc. Advances in Cryptology—Eurocrypt '96,* pp. 399-416, 1996.

[3] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation," *Proc. 20th ACM Symp. Theory of Computing,* pp. 1-10, 1988.

[4] S. Berkovits, "How to Broadcast a Secret," *Proc. Advances in Cryptology—Eurocrypt '91,* pp. 535-541, 1991.

[5] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," *Proc. Advances in Cryptology—Eurocrypt '84,* pp. 335-338, 1985.

[6] C. Blundo, A.D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-Secure Key Distribution for Dynamic Conferences," *Proc. Advances in Cryptology—Crypto '92,* pp. 471-486, 1993.

[7] D. Boneh and R. Venkatesan, "Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Problems," *Proc. Advances in Cryptology—Crypto '96,* pp. 129-142, 1996.

[8] M. Burmester and Y. Desmedt, "A Secure and Efficient Conference Key Distribution System," *Proc. Advances in Cryptology —Eurocrypt '94,* pp. 275-286, 1995.

[9] R. Canetti and A. Herzberg, "Maintaining Security in the Presence of Transient Faults," *Proc. Advances in Cryptology—Crypto '94,* pp. 425-438, 1994.

[10] C.C. Chang and C.H. Lin, "How to Converse Securely in a Conference," *Proc. IEEE 30th Ann. Int'l Carnahan Conf.,* pp. 42-45, 1996.

[11] C.C. Chang, T.C. Wu, and C.P. Chen, "The Design of a Conference Key Distribution System," *Proc. Advances in Cryptology—Auscrypt '92,* pp. 459-466, 1992.

[12] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory,* vol. 22, pp. 644-654, 1976.

[13] W. Diffie, P.C. van Oorschot, and M.J. Weiner, "Authentication and Authenticated Key Exchanges," *Design, Codes and Cryptography,* vol. 2, pp. 107-125, 1992.

[14] M. Fitzi, M. Hirt, and U. Maurer, "Trading Correctness for Privacy in Unconditional Multi-Party Compution," *Proc. Advances in Cryptology—Crypto '98,* pp. 121-136, 1998.

[15] T.L. Hwang and J.L. Chen, "Identity-Based Conference Key Broadcast Systems," *IEE Proc.: Computers and Digital Techniques,* vol. 141, no. 1, pp. 57-60, 1994.

[16] I. Ingemarsson, D.T. Tang, and C.K. Wong, "A Conference Key Distribution System," *IEEE Trans. Information Theory,* vol. 28, no. 5, pp. 714-720, 1982.

[17] B. Klein, M. Otten, and T. Beth, "Conference Key Distribution Protocols in Distributed Systems," *Proc. Codes and Ciphers—Cryptography and Coding IV,* pp. 225-242, 1995.

[18] K. Koyama, "Secure Conference Key Distribution Schemes for Conspiracy Attack," *Proc. Advances in Cryptology—Eurocrypt '92,* pp. 449-453, 1993.
[19] K. Koyama and K. Ohta, "Identity-Based Conference Key Distribution Systems," *Proc. Advances in Cryptology—Crypto '87,* pp. 175-184, 1988.
[20] K. Koyama and K. Ohta, "Security of Improved Identity-Based Conference Key Distributioin Systems," *Proc. Advances in Cryptology—Eurocrypt '88,* pp. 11-19, 1988.
[21] T. Matsumoto and H. Imai, "On the Key Predistribution System: A Practical Solution to the Key Distribution Problem," *Proc. Advances in Cryptology—Crypto '87,* pp. 185-193, 1988.
[22] C. Mitchell, F. Piper, and P. Wild, "Digital Signature," *Contempary Cryptography, The Science of Information Integrity,* pp. 325-378, 1992.
[23] M. Naor and O. Reingold, "Number-Theoretic Constructions of Efficient Pseudorandom Functions," *Proc. 38th IEEE Symp. Foundations of Computer Science,* 1997.
[24] D. Pointcheval and J. Stern, "Security Proofs for Signatue Schemes," *Proc. Advances in Cryptology—Eurocrypt '96,* pp. 387-398, 1996.
[25] T. Rabin and M. Ben-Or, "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority," *Proc. 26th ACM Symp. Theory of Computing,* pp. 73-85, 1989.
[26] R. Rueppel and P. Van Oorschot, "Modern Key Agreement Techniques," *Computer Comm.,* 1994.
[27] A. Shamir, "How to Share a Secret," *Comm. ACM,* vol. 22, pp. 612-613, 1979.
[28] A. Shimbo and S. Kawamura, "Cryptanalysis of Several Conference Key Distribution Schemes," *Proc. Advances in Cryptology —Asiacrypt '91,* pp. 265-276, 1993.
[29] V. Shoup, "Lower Bounds for Discrete Logarithms and Related Problems," *Proc. Advances in Cryptology—Eurocrypt '97,* pp. 256-266, 1997.
[30] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener, "A Secure Audio Teleconference System," *Proc. Advances in Cryptology—Crypto '88,* pp. 520-528, 1990.
[31] T.C. Wu, "Conference Key Distribution System with User Anonymity Based on Algebraic Approach," *IEE Proc.: Computers and Digital Techniques,* vol. 144, no. 2, pp. 145-148, 1997.
[32] Y. Yacobi, "Attack on the Koyama-Ohta Identity Based Key Distribution Scheme," *Proc. Advances in Cryptology—Crypto '87,* pp. 429-433, 1988.

Wen-Guey Tzeng received the BS degree in computer science and information engineering from National Taiwan University, Taiwan, in 1985 and the MS and PhD degrees in computer science from the State University of New York at Stony Brook in 1987 and 1991, respectively. He joined the Department of Computer and Information Science, National Chiao Tung University, Taiwan, in 1991. Dr. Tzeng's current research interests include cryptology, information security, and network security.