International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:8, No:12, 2014

An Enhanced Fault-Tolerant Conference Key Agreement Protocol

Cheng-Chi Lee, Chun-Ta Li, Chia-Ying Wu, Shiow-Yuan Huang

***Abstract*—**Establishing a secure communication of Internet conferences for participants is very important. Before starting the conference, all the participants establish a common conference key to encrypt/decrypt communicated messages. It enables participants to exchange the secure messages. Nevertheless, in the conference, if there are any malicious participants who may try to upset the key generation process causing other legal participants to obtain a different conference key. In this article, we propose an improved conference key agreement with fault-tolerant capability. The proposed scheme can filter malicious participants at the beginning of the conference to ensure that all participants obtain the same conference key. Compare with other schemes, our scheme is more secure and efficient than others.

***Keywords*—**Conference key, Diffie-Hellman protocol, key agreement, fault tolerance.

I. Introduction

DUE to the convenience of the Internet, a lot of people use the Internet to organize a conference or to communicate with each other even though they located in different places in the world. However, the network is a shared medium so that the weakness security attacks such as eavesdropping, replay attack and modification attack. Thus, we have to establish a common conference key for encrypting/decrypting our communications over an insecure network. The first conference key establishment was proposed by [9] in 1982. In general, there are two types of conference key establishment schemes: conference key distribution is proposed in [4], [7], [8], [16], and conference key agreement is proposed in [1]-[3], [10], [12]. The conference key distribution protocol needs a trusted third party (TTP) to establish a key and then securely distribute it to each participant. A conference key agreement protocol allows all participants to establish a common session key without a trusted third party (TTP). The advantage of conference key distribution protocol is low communication and computational costs, and simplicity. The conference key agreement protocol does not require TTP to establish the session key, but it needs higher communication and computational costs than conference key distribution protocol.

In 1976, Diffie and Hellman [5] proposed the first commonly known key agreement protocol that allows two participants to establish a common secret key that is used for encrypting/decrypting communications. Its security is based on the difficulty of Discrete Logarithm Problem (DLP). Up to now, most conference key agreements focus on communication efficiency and privacy of conference key. Considering an example, before starting the network conference, all the members need establish a common key to encrypt/decrypt the message of the conference. But there is one or more malicious conferees who attempt to destroy the conference lead to other conferees who obtain a different conference key. To solve this problem, fault tolerant conference key agreement [6], [11], [13]-[15] was proposed by many scholars. Among these protocols, [14] proposed a fault tolerant conference key agreement. This protocol can correctly acquire a conference key even if there are several malicious conferees. However, Tzeng's protocol requires creating n n-power polynomials for each participant and the result of heavy computation is inefficient, where n is the number of participants. To solve this problem, [6] proposed a conference key agreement protocol in 2009. Compared with Tzeng's scheme their scheme is more efficient. However, we found Huang et al.'s protocol [6] is vulnerable to modification attack, and we also show a simple improvement to eliminate the security weakness of Huang et al.'s protocol [6].

This paper is organized as follows: in Section II, we review Huang et al.'s conference key agreement protocol, and in section III, we introduce the security flaws of the Huang et al.'s protocol [6]. In section IV we show a simple improvement to the proposed vulnerability. We discuss the security and performance of our improved scheme in the Section V. Finally, our conclusion is given in Section VI.

II. Review of Huang et al.'s Protocol [6]

In this section, we review Huang et al.'s [6] conference key agreement protocol. Their proposed method comprises five phases: parameter generation phase, secret distribution and commitment phase, sub key computation and verification phase, fault detection phase, and conference key computation phase. The protocol starts with an initiator who convenes a conference for a set U of participants. Let $U= \{U_1, U_2,\ldots,U_n\}$ be the initial set of participant that want to generate a conference key. Each $U_i(1\leq i\leq n)$ is part of U.

A. Parameter Generation Phase

The following system parameters and function are used throughout the paper.

q A large prime.

C. C. Lee is with the Dep. of Library and Information Science, Fu Jen Catholic University, Taipei, Taiwan and Asia University, Taichung, Taiwan (e-mail: cclee@mail.fju.edu.tw).

C. T. Li is with the Dep. of Information Management, Tainan University of Technology, Tainan, Taiwan (Corresponding e-mail: th0040@mail.tut.edu.tw).

C. Y. Wu and S. Y. Huang is with the Dep. of Photonics & Communication Engineering, Asia University, Taichung, Taiwan.

p	A large prime number comprised of $2q+1$.
g	A q-order generator over $GF(p)$.
$H(\cdot)$	A secure one-way hash function.
T	Time stamp to detect a delay and it will be updated to a new one in each conference section.
k_{ij}	The common section key that U_i shared with all other participants U_j.
k_{ji}	The common section key that U_i receive from U_j.
CK_i	The subkey that U_i shared with all other participants.
$\|$	String concatenation operation which combines two values into one.
$\oplus$	The bitwise XOR operation.

Meanwhile, each U_i is provided with the following pair of two corresponding keys:

1. Private key is denoted by $x_i \in Z^*_q$;
2. Public key is denoted by $y_i = g^{xi} \bmod p$.

B. Secret Distribution and Commitment Phase

All participants U_i execute the following steps to share their subkey CK_i to other participants:

Step1. Each U_i ($1 \le i \le n$) randomly choose an integer $a_i \in Z^*_q$, and computes the session key k_{ij} using other participants U_j's public key y_j:

$$k_{ij} = y_j^{ai} \bmod p \bmod q,\ 1 \le j \le n$$

Step2. Select a line $L(x)$:

$$L(x) = c_i x + CK_i \bmod q$$
$$c_i = g^{ai} \bmod p$$

Step3. Computes d_{ij} and d_{ij}':

$$d_{ij} = L(k_{ij}) \bmod q,\ 1 \le j \le n$$
$$d_{ij}' = k_{ij} \oplus d_{ij},\ 1 \le j \le n$$

Step4. Randomly select an integer $r_i = Z^*_q$ and computes the digital signature (R_i, S_i):

$$R_i = g^{r_i} \bmod p$$
$$S_i = x_i H(CK_i \| T) + r_i R_i \bmod q$$

Step5. Broadcast the message $M_i = \{T, R_i, S_i, c_i, d_{i1}', d_{i2}', \ldots, d_{in}'\}$.

C. Subkey Computation and Verification Phase

After receiving message $M_j = \{T, R_j, S_j, c_j, d_{j1}', d_{j2}', \ldots, d_{jn}'\}$ from U_j ($1 \le j \le n$), each U_i recovers the subkey CK_j by using M_j ($i \ne j$) and U_j execute the following steps. The steps of the subkey computation and verification phase are as follows:

Step1. Check the time stamp T.

Step2. Compute the session key k_{ji} using private key x_i and c_j:

$$k_{ji} = c_j^{xi} \bmod p \bmod q,\ 1 \le j \le n$$

Step3. Compute the subkey CK_j using k_{ji}, d_{ji}' and c_j:

$$d_{ji} = d_{ji}' \oplus k_{ji},\ 1 \le j \le n$$

$$CK_j = d_{ji} - c_j k_{ji} \bmod q,\ 1 \le j \le n$$

Step4. Check the digital signature (R_j, S_j):

$$g^{sj} \stackrel{?}{=} y_j^{H(CKj \| T)} R_j^{Rj} \pmod p$$

If the steps 1~4 are assured, broadcast v_{ij}="success"; otherwise, broadcast v_{ij}="failure".

D. Fault Detection Phase

In this phase, each participant U_i executes the following procedure if they receive v_{jm}="failure":

1. On receiving v_{jm}="failure", U_j claims that U_m ($m \ne i$) is faulty, each participant wait for the fault detection message a_m and CK_m from U_m. If no one receives the fault detection message, then set U_m as a malicious participant;
2. On receiving message a_m and CK_m from U_m. Each participant executed the following steps to detect fault.

a. Compute $c_m' = g^{am}$, check whether $c_m = c_m'$.
b. Check d_{mj}' by input a_m and CK_m into secret distribution and commitment phase steps 1-3.
c. Check whether signature (R_m, S_m) is correct made by U_m. If all the steps are satisfied, set U_j as a malicious participant; otherwise, set U_m as a malicious one.

3. Removed all the malicious participants from U and restarted the protocol.

E. Conference Key Computation Phase

When malicious participants are excluded from U, each honest participants U_i in the set of $U' = \{U_1', U_2', \ldots, U_n'\}$ calculates the conference key CK.

$$CK = (CK'_1 + CK'_2 + \ldots + CK'_n) \bmod q$$

III. Weakness of Huang et al.'s [6] Protocol

In this section, we point out that Huang et al.'s [6] protocol is vulnerable to a modification attack. This can lead to two different situations. One of the situations is all participants may generate different conference keys, and the other is that honest participants will be excluded from the conference. We assume that E is the attacker who alters the exchanged messages in the subkey computation and verification phase.

Situation 1. Honest participants will be excluded from the set of participants.

First we assume U_i and U_j both honest participants in the set of $U = \{U_1, U_2, \ldots, U_n\}$. In subkey computation and verification phase, U_i executes the steps 1~4 after U_i receiving message M_j.

Step1. U_i verifies the steps 1~4.

Step2. If satisfied, U_i broadcasts v_{ij}="success"

Step3. The attacker E intercepts the message send from U_i. Then E modifies the message v_{ij}="success" to v_{ij}="failure" and broadcasts the message.

Step4. After receiving v_{ij}="failure", all participants start fault detection phase. U_j broadcasts fault detection message a_i, CK_i.

Step5. Other participants verify a_i, CK_i.

Step6. Because U_j is an honest participant, so the fault detection message will pass the verification. Finally, all the participants set U_i as a malicious participant and remove U_i from the set of participants. In fact, U_i is not a malicious participant.

U_i	$U_j(i \neq j)$
1. $ai \in_R Z^*_q$ $k_{ij}=y_j^{ai} \bmod p \bmod q$; $1 \le j \le n$	
2. choose a line $L(x)$ $L(x)=c_i x + CK_i \bmod q$ $c_i=g^{ai} \bmod p$	
3. $d_{ij}=L(k_{ij}) \bmod q$; $1 \le j \le n$ $d_{ij}=c_i k_{ij}+CK_i \bmod q$ $d_{ij}'=k_{ij} \oplus d_{ij}$	
4. $r_i \in_R Z^*_q$ $R_i=g^{ri} \bmod p$ $S_i=x_i H(CK_i \| T)+r_i R_i \bmod q$	
5. $M_i=\{T, R_i, S_i, c_i, d_{i1}', d_{i2}', \ldots, d_{in}'\}$ →	

Fig. 1 Secret distribution and commitment phase

Situation 2. All conference participants obtain different conference keys.

Set U_i is a malicious participant who sends a wrong message M_i. One of the honest participants U_j broadcasts message v_{ji}="failure" after verifying the individual digital signature (R_i, S_i).

Step1. U_j verifies U_i's individual digital signature (R_i, S_i).

Step2. We assume U_i is a malicious participant, then $g^{sj} \neq y_j^{H(CKj \| T)} R_j^{Rj} \pmod p$,

U_j broadcasts v_{ji}="failure".

Step3. The attacker E intercepts the message sent from U_j. E Modifies the message v_{ji}="failure" to v_{ji}="success" and broadcasts the message.

Step4. All participants compute $CK=(CK'_1+CK'_2+...+CK'_n) \bmod q$ after no more faults are detected. Actually, U_i sends a wrong message M_i causing all conference participants to obtain different conference keys.

The protocol is capable of fault tolerance, and this means that it can achieve the following two lemmas.

Lemma 1. Malicious participant U_i attempting to use different subkeys CK_is to cheat honest participants into accepting a different conference key shall be removed from the set of participants.

Lemma 2. No honest participants shall be excluded from the set of participants.

We prove the Huang et al.'s [6] protocol not only leads to all participants obtaining different conference keys but also fails to achieve the **Lemma 2**.

IV. Our Improvement

To resist the modification attack, we propose an improvement on the Huang et al.'s [6] scheme in this section. Our improvement contains four phases: secret distribution and commitment phase, sub key computation and verification phase, fault detection phase, and conference key computation phase. The parameters used here are the same as the Huang et al.'s scheme [6]. Detailed steps of the proposed scheme are described as follows.

A. Secret Distribution and Commitment Phase

This phase is the same as that in the Huang et al.'s [6] protocol, each U_i broadcast the message $M_i=\{T, R_i, S_i, c_i, d_{i1}', d_{i2}', \ldots, d_{in}'\}$ to other participants U_j. This phase is shown in Fig. 1.

B. Subkey Computation and Verification Phase

After receiving M_j, each participant U_i executes the following steps and the detailed steps are described in Fig. 2.

Step1. Check whether or not time stamp T is valid.

Step2. Compute the section key $k_{ji}=c_j^{xi} \bmod p \bmod q$ ($1 \le j \le n$) using x_i and c_j.

Step3. Calculate the subkey CK_j:

$$d_{ji}=d_{ji}' \oplus k_{ji}$$

$$CK_j=d_{ji}-c_j k_{ji} \bmod q$$

Step4. Check the signature (R_j, S_j) by the following signature verification equation:

$$g^{sj} \stackrel{?}{=} y_j^{H(CKj \| T)} R_j^{Rj} \pmod p$$

Step5. If steps 1-4 is satisfied, then v_{ij}="success" otherwise v_{ij}="failure" and then compute $h(k_{ij} \| v_{ij})$.

Step6. Broadcast v_{ij} and $h(k_{ij} \| v_{ij})$. As compared with Huang et al.'s [6] protocol, our improvement provides message authentication.

U_i	$U_j(i \neq j)$
	$M_j=\{T, R_j, S_j, c_j, d_{j1}', d_{j2}', \ldots, d_{jn}'\}$ ←
1. Varity T	
2. $k_{ji}=c_j^{xi} \bmod p \bmod q$; $1 \le j \le n$	
3. $d_{ji}=d_{ji}' \oplus k_{ji}$ $CK_j=d_{ji}-c_j k_{ji} \bmod q$	
4. $g^{sj} \stackrel{?}{=} y_j^{H(CKj \| T)} R_j^{Rj} \pmod p$	
5. If Eq.4 is satisfied, v_{ij}="success" otherwise v_{ij}="failure", compute $h(k_{ij} \| v_{ij})$.	
6. Broadcast v_{ij} and $h(k_{ij} \| v_{ij})$	
	$h(k_{ji} \| v_{ij})$, v_{ij} →

Fig. 2 Subkey computation and verification phase

C. Fault Detection Phase

As receiving the message v_{ij} and $h(k_{ji} \| v_{ij})$, each participant first computes $h(k_{ji} \| v_{ij})'$ and verifies the equation $h(k_{ji} \| v_{ij})' \stackrel{?}{=}$

$h(k_{ji}\|v_{ij})$. If the equation is satisfied, then start to execute the fault detection phase, or else, set U_i as a malicious participant. Later on, each participant waits for the fault detection message a_j and CK_j from U_j. If no fault detection is received from U_j, then set U_j as a malicious participant. This phase works as follows.

1. On receiving message a_j and CK_j from U_j. Each participant executes the following steps to detect fault.

a. Compute $c_m'=g^{am}$, check whether $c_m= c_m'$;

b. Check d_{ji}' by input a_j and CK_j into secret distribution and commitment phase steps 1-3.

c. Check whether signature (R_j, S_j) is correct made by U_j. If all the steps are satisfied, set U_i as a malicious participant; otherwise, set U_j as a malicious one.

2. Remove all the malicious participants from U and restarts the protocol.

D. Conference Key Computation Phase

When the aforementioned process is executed until no more faults are detected, each honest participant of the set $U'= \{U_1', U_2', ..., U_n'\}$ computes the conference key CK, as follows:

$$CK= (CK'_1+CK'_2+...+CK'_n) \bmod q$$

V. Analysis of Our Improvement

In this section, we describe the security analysis of our improvement and compare performance with Huang et al.'s protocol. At first, we show an improved method against the modification attack as:

A. Modification Attack

Our improvement not only can resist impersonated attacks, replay attacks but also can resist the modification attack that we find in Huang et al.'s protocol.

According to the attack method in Section III.*A*, we presume that E is the attacker who eavesdrops on the broadcast channel and tries to alter the exchanged messages in the sub key computation and verification phase. Assume E replaces v_{ij} with v_{ij}' and resend to U_j. On receive v_{ij}', $h(k_{ij}\|v_{ij})$, U_j first computes $h(k_{ij}\|v_{ij}')$ and check $h(k_{ij}\|v_{ij}')\stackrel{?}{=} h(k_{ij}\|v_{ij})$. Thus attacker need to compute the session key k_{ij}. However, to find $k_{ji}= c_j^{xi} \bmod p \bmod q$ is hard. First, he needs to compute the private key x_i of user U_i. To obtain x_i from $y_i=g^{xi}$ the attacker will have to solve the discrete logarithm problem. Therefore the modification attack cannot work in our improvement.

B. Replay Attack

The adversary might try to impersonate valid participants U_i to resend the message M_i in secret distribution and commitment phase. Since generated the message M for all conferences has a time stamp T. As receiving the message M_i all the participants can verify the validity of the timestamp T from M_i and the signature verification equation. So replay attack can be avoided in our improvement.

C. Performance

In addition to security of the system, the efficiency of the protocol has been an important issue. In this section we compare with Huang et al.'s protocol [6]. The protocol of Huang et al. has been in comparison with the method of Tzeng that was declared efficient. The analysis of performance is partition into analysis of computation costs and transmission costs. Computation costs contain cost of secret distribution and commitment phase, subkey computation and verification phase and cost of executing fault detection. Transmission costs include the load of each participants broadcasted transmission message. The notation is as follows Table I.

Table II illustrates the comparison of the improvement and Huang et al. proposed protocol. The table shows that our improvement increased the one way hash function in subkey computation and verification phase and fault detection phase. However, our improvement is more secure than Huang et al.'s protocol. Since the proposed protocol suffers from modification attack, our improvement used message authentication code to resist modification attack.

Table III analyzes communication cost. Owing to used message authentication code, our improvement has a heavier cost of q. Nevertheless, it can remedy Huang et al.'s protocol.

TABLE I
Definitions of Mathematical Notations

Notation	*Definition*
$T_{L(n)}$	The time for establishing an n-power Lagrange polynomial interpolation
$T_{P(n)}$	The time for calculating the output of an n-power polynomial
T_{EXP}	The time for the modular exponential operation
T_{MUL}	The time for the modular multiplicative operation
T_H	The time for executing the adopted one-way hash function H
T_{INV}	The time for modular inverse operation
$\|x\|$	The bit length of x
n	The total number of participants

TABLE II
Comparison of Computational Cost

	Tzeng [14]	Huang et al. [6]	Our improvement
Secret distribution and commitment phase	$1T_L(n)+nT_p(n)+(n+2)T_{EXP}+2T_{MUL}+1T_H+1T_{INV}$	$(n+2)T_{EXP}+(n+2)T_{MUL}+1T_H$	$(n+2)T_{EXP}+(n+2)T_{MUL}+1T_H$
Subkey computation and verification phase	$nT_L(n)+4nT_{EXP}+nT_{MUL}+nT_H$	$4nT_{EXP}+nT_{MUL}+nT_H$	$4nT_{EXP}+nT_{MUL}+2nT_H$
Fault detection phase	$1T_L(n)+(2n+1)T_p(n)+5T_{EXP}+1T_{MUL}+1T_H$	$5T_{EXP}+2T_{MUL}+1T_H$	$5T_{EXP}+2T_{MUL}+(n+1)T_H$

TABLE III
Comparison of Communication Cost

Tzeng [14]	Huang et al. [6]	Our improvement
$(n+1) \| q \| +2 \| p\|$	$(n+1) \| q \| +2 \| p \| +\| T \|$	$(n+2) \| q \| +2 \| p \| +\| T \|$

VI. Conclusions

In this paper, the security of a conference key agreement protocol with fault-tolerant capability by Huang et al. [6] is analyzed. The main idea of the protocol is the filtering of malicious participants at the beginning of the conference to ensure that all participants can obtain the same conference key.

However, we find that Huang et al.'s conference key agreement protocol is vulnerable to modification attack. It may result in other participants obtaining different conference keys and honest participants will be excluded from the set of participants. Thus, we propose an improvement to remedy Huang et al.'s weaknesses. The proposed improvement only requires one additional hash function, as we prove; our protocol can work against the proposed attack method.

ACKNOWLEDGMENT

This research was partially supported by the Ministry of Science and Technology, Taiwan, R.O.C., under contract no.: MOST 103-2221-E-030-016.

REFERENCES

[1] E. Bresson, O. Chevassut and D. Pointcheval, "Provably authenticated group Diffie-Hellman key exchange-the dynamic case", *Lecture Notes in Computer Science*, vol. 2248, 2001, pp. 290–309.
[2] E. Bresson, O. Chevassut, D. Pointcheval and J. Quisquater, "Provably authenticated group Diffie–Hellman key exchange", *Eighth ACM Conference Computer and Communications Security*, 2001, pp. 255–264.
[3] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system", *Lecture Notes in Computer Science*, Springer, Berlin, vol. 950, 1994, pp. 275–286.
[4] I. Chung, W. Choi, Y. Kim and M. Lee, "The design of conference key distribution system employing a symmetric balanced incomplete block", *Information Processing Letters*, vol. 81, no. 6, 2002, pp. 313–318.
[5] W. Diffie and M. E. Hellman, "New Direction in Cryptography", *IEEE Transaction on Information Theory*, vol. IT–22, no. 6, 1976, pp. 644-654.
[6] K. Huang, Y. Chung, H. Lee, F. Lai and T. Chen, "A conference key agreement protocol with fault–tolerant capability", *Computer Standards and Interfaces*, vol. 31, no. 2, 2009, pp. 401–405.
[7] S. Hirose and K. Ikeda, "A conference distribution system for the start configuration based on the discrete logarithm problem", *Information Processing Letters*, vol. 62, no. 4, 1997, pp. 189–192.
[8] M. S. Hwang and W. P. Yang, "Conference key distribution schemes for secure digital mobile communications", *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 2, 1995, pp. 416–420.
[9] I. Ingermarsson and C. Wong, "A conference key distribution system", *IEEE Transactions on Information Theory*, vol. 28, no. 5, 1982, pp. 714–720.
[10] Y. Kim, A. Perrig and G. Tsudik, "Group key agreement efficient in communication", *IEEE Transactions on Computers*, vol. 53, no. 7, 2004, pp. 905–921.
[11] Y. Kim, A. Perrig and G. Tsudik, "Simple and fault–tolerant key agreement for dynamic collaborative groups", *in Proceedings of ACM Conference on Computer and Communications Security*, 2000, pp. 235–244.
[12] W. H. Kim, E. K. Ryu, J. Y. Im and K. Y. Yoo, "New conference key agreement protocol with user anonymity", *Computer Standards & Interfaces*, vol. 27, no. 2, 2005, pp. 185–190.
[13] S. Lee, J. Kim and S. Hong, "Security weakness of Tseng's fault–tolerant conference key agreement protocol", *Journal of Systems and Software*, vol. 82, no. 7, 2009, pp. 1163–1167.
[14] W. G. Tzeng, "A secure fault–tolerant conference key agreement protocol", IEEE Transactions on Computers, vol. 51, no. 4, 2002, pp. 373–379.
[15] Y. M. Tseng, "A communication–efficient and fault–tolerant conference–key agreement protocol with forward secrecy", *Journal of Systems and Software*, vol. 80, no. 7, 2007, pp. 1091–1101.
[16] Y. M. Tseng, "Cryptanalysis and improvement of key distribution system for VSAT satellite communication", *International Journal of Informatica*, vol. 13, no. 3, 2002, pp. 369–376.