Computer Standards & Interfaces 31 (2009) 401–405

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

A conference key agreement protocol with fault-tolerant capability

Kuo-Hsuan Huang [a,*], Yu-Fang Chung [b,c], Hsiu-Hui Lee [d], Feipei Lai [a,d,e], Tzer-Shyong Chen [c]

[a] Department of Electrical Engineering, National Taiwan University, Taiwan
[b] Department of Electrical Engineering, Tunghai University, Taiwan
[c] Department of Information Management, Tunghai University, Taiwan
[d] Department of Computer Science and Information Engineering, National Taiwan University, Taiwan
[e] Graduate Institute of Biomedical Electronics and Bioinformatics, National Taiwan University, Taiwan

ARTICLE INFO

Article history:
Received 24 July 2007
Received in revised form 5 March 2008
Accepted 4 May 2008
Available online 18 May 2008

Keywords:
Conference key
Key agreement
Fault tolerance
Discrete logarithm problem

ABSTRACT

Secure communication is of utmost importance to participants of Internet conferences. Secure communication thwarts eavesdropping. In an Internet conference, all conference participants together establish a common conference key to enable multi-party and secure exchange of messages. However, malicious conference participants may try to obtain the conference key through unfair means, and this could result in the generation of different conference keys. This paper is intended as a proposal of a new form of conference key agreement protocol. It emphasizes the filtering of malicious participants at the beginning of the conference to ensure that all participants obtain the same conference key. The proposed method also has fault-tolerant capability. Efficiency and security of a protocol is important in practice. The security of the proposed protocol is based on discrete logarithm problem assumption. The protocol is executed in computationally secure environment. The secret information of a user cannot be determined from its corresponding public information and therefore ensures privacy. Since efficiency of a protocol depends on low computation cost, the protocol attempts to achieve lower computation cost without compromising on security.

1. Introduction

Owing to the convenience of the Internet, using the Internet for communications has become the trend of the present and the future. Using the Internet, a group of people can, without assembling together, achieve the object of holding a conference. Since members communicate in open Internet environment, in order to prevent outsiders from obtaining the content of the conference, the members should encrypt conversations with a secret key cryptosystem, such as Advanced Encryption Standard (AES) [1]. According to the secure communication system model proposed by Shannon [2], all members must be sharing a common secret key before the running conference encryption. Therefore, secure and efficient construction of the shared key is a subject of great importance and interest to a large number of researchers. The first commonly known key agreement protocol was proposed by Diffie and Hellman [3] in 1976. Its security is based on the difficulty of solving Discrete Logarithm Problem (DLP). Later, it was developed into International Standards, for example, PKCS #3 [4] and ANSI X9.42 [5].

Considering an example of network conference based on key agreement protocol, before the start of the conference, the members must establish a shared key to encrypt the details of the conference; this key is called conference key. The conference key is established jointly by all members of the conference, and not by any single member; this method is known as the conference key agreement protocols [6–14]. Thus, a situation where one member might have greater rights than other members can be avoided. Most current conference key agreement protocols emphasizes the privacy of conference key and message efficiency under the framework of different network connections; therefore they do not have fault-tolerant ability to detect malicious participant deliberately upsetting the key generation process causing other participants to obtain different conference keys. Tzeng [15] proposed a conference key agreement protocol with fault tolerance in recent years. The protocol can still correctly establish a conference key even if there are several malicious participants among the conference participants. However, the method requires each participant to create n n-power polynomials, where n is the number of participants; this is a serious encumbrance to efficiency.

In order to let all network conference participants to exchange information via secret communication, there must be a key agreement protocol to allow members to jointly construct a conference

* Corresponding author.
E-mail address: d92921013@ntu.edu.tw (K.-H. Huang).

0920-5489/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2008.05.015

key that will be used to perform encryptions. This protocol must have a process for detection and elimination of malicious participants so as to prevent legitimate members from obtaining an erroneous conference key. Besides, considering practicality, key agreement protocol must have low computation load. Hence, this study proposes a more efficient and fault-tolerant conference key agreement protocol. The rest of this paper is organized as follows: Section 2 describes a new conference key agreement protocol. Section 3 carries out security and performance analysis. Section 4 furnishes the conclusions.

2. The proposed protocol

The general objective of the conference key agreement protocol is to allow a group of people to convene a conference on the Internet. The conventions usually require time effects. Hence, besides fault tolerance capabilities, the efficiency of the protocol too must be considered during the designing of the distributed conference key agreement protocol. A new conference key agreement protocol is proposed to meet the above requirements.

The proposed method has four phases, including parameter generation phase, secret distribution and commitment phase, subkey computation and verification phase, fault detection phase, and conference key computation phase. These phases are explained below.

2.1. Parameter generation phase

The system authority selects the following parameters and function, and declares them publicly:

(1) p: a large prime number comprised of $2q+1$, where q is also a large prime;
(2) g: a q-order generator over $GF(p)$;
(3) $H(\cdot)$: a one-way hash function.

Each user U_i is provided with the following pair of two corresponding keys:

(1) Private key denoted as $x_i \in Z_q^*$;
(2) Public key denoted as $y_i = g^{x_i} \bmod p$.

The protocol starts up the initiator who calls for a conference by initializing a set of participants U. First, let $U=\{U_1, U_2, \ldots, U_n\}$ be the initial participant set. Each participant $U_i (1 \le i \le n)$ is a part of U. In addition, the function of time stamp T is required, and it will be updated to a new one in each conference section.

2.2. Secret distribution and commitment phase

All participants U_i of set U execute the following steps to distribute his subkey to other participants:

Step 1: Randomly select an integer $a_i \in Z_q^*$, and calculate the common session key k_{ij} shared with all other participants U_j using the public key y_j of U_j:

$$k_{ij} = y_j^{a_i} \bmod p \bmod q, \ 1 \le j \le n \tag{2.1}$$

Step 2: Randomly select a line $L(x)$:

$$L(x) = c_i x + CK_i \bmod q \tag{2.2}$$

where $c_i = g^{a_i} \bmod p$ and CK_i is the subkey that U_i offers to share with the other participants;

Step 3: Calculate the values d_{ij} and d_{ij}' using the session key k_{ij} and the polynomial $L(x)$:

$$d_{ij} = L(k_{ij}) \bmod q, \ 1 \le j \le n \tag{2.3}$$

$$d_{ij}' = k_{ij} \oplus d_{ij}, \ 1 \le j \le n \tag{2.4}$$

Step 4: Randomly select an integer $r_i \in Z_q^*$, *and generate the individual digital signature* (R_i, S_i) *on the subkey* CK_i *using the individual private key* x_i*:*

$$R_i = g^{r_i} \bmod p \tag{2.5}$$

$$S_i = x_i H(CK_i || T) + r_i R_i \bmod q \tag{2.6}$$

Step 5: Broadcast the message $M_i=\{T, R_i, S_i, c_i, d_{i1}', d_{i2}', \ldots, d_{in}'\}$.

2.3. Subkey computation and verification phase

Each participant $U_i \in U$ recovers the subkey CK_j using the received message $M_j=\{T, R_j, S_j, c_j, d_{j1}', d_{j2}', \ldots, d_{jn}'\}$ according to the following steps:

Step 1: Check the time stamp T in advance, if it is invalid, terminate the subkey computation and verification phase;
Step 2: Calculate the common session key k_{ji} shared with all other participants U_j using the individual private key x_i and the value c_j:

$$k_{ji} = c_j^{x_i} \bmod p \bmod q, \ 1 \le j \le n \tag{2.7}$$

Step 3: Calculate the subkey CK_j using the session key k_{ji} , the values d_{ji}' and c_j:

$$d_{ji} = d_{ji}' \oplus k_{ji}, \ 1 \le j \le n \tag{2.8}$$

$$CK_j = d_{ji} - c_j k_{ji} \bmod q, \ 1 \le j \le n \tag{2.9}$$

Step 4: Check whether the data (R_j, S_j) on CK_j is the right signature made by U_j by using the following signature verification equation:

$$g^{S_j} \stackrel{?}{=} y_j^{H(CK_j||T)} R_j^{R_j} (\bmod p), \ 1 \le j \le n \tag{2.10}$$

If the Eq. (2.10) is satisfied, broadcast v_{ij}="success"; otherwise, broadcast v_{ij}="failure".

2.4. Fault detection phase

Each participant $U_i \in U$ executes the following procedure to detect faults:

Step 1: On receiving v_{ji}="failure" for U_j, U_j claims that U_i is faulty. Meanwhile, U_i secretly exposes the self-retained value a_i and his subkey CK_i to all other participants;
Step 2: On receiving v_{jm}="failure", U_j claims that U_m $(m \ne i)$ is faulty, and the following procedure is executed:
Step 2.1: Wait for the fault detection messages a_m and CK_m from U_m;
Step 2.2: If no fault detection messages are received from U_m , then set U_m as a malicious participant;

Table 1
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

Step 2.3: On receiving a_m and CK_m, check whether R_m, S_m, c_m, and d_{mj}' are correct:
Step 2.3.1: Check whether a_m satisfies $c_m = g^{am} \bmod p$;
Step 2.3.2: Input a_m and CK_m into Eqs. (2.1)~(2.4) to check whether the resulting value is the same as $d_{mj}^{'}$;
Step 2.3.3: Check whether the data (R_m, S_m) on CK_m is the right signature made by U_m. If satisfied, set U_j as a malicious participant; otherwise, set U_m as the malicious one.
Step 3: The malicious participant is removed from the set of participants U by the other honest participants and the protocol is restarted.

2.5. Conference key computation phase

When the previous phase is executed until no more faults are detected, each participant U_i in the set of $U' = \{U_1', U_2', \ldots, U_m'\}$ calculates the conference key CK, as follows:

$$CK = (CK'_1 + CK'_2 + \ldots + CK'_m) \bmod q \tag{2.11}$$

3. Analyses of security and efficiency

This section analyzes possible attacks on the protocol and fault tolerance capability of the conference key agreement protocol. The analysis proves the protocol to be fault-tolerant and secure against attacks. Furthermore, a good conference key agreement protocol should be also efficient in performance, so costs for computation and transmission have to be taken into consideration in demonstrating efficiency.

3.1. Security analysis

The security of the conference key protocol discussed in the paper is based on the following assumed problem:

[Discrete logarithm problem assumption]

Let p be a large prime number and q a large prime factor of $p-1$, g is a q-order generator over $GF(p)$. Given an integer y that satisfies $y = g^x \bmod p$. It is computationally secure to obtain the unknown x from the disclosed y.

The following proves the protocol capable of fault tolerance:

(1) Malicious participant U_i attempting to use different subkeys CK_is to dupe honest participants are removed from the set of participants.
(2) No honest participants shall be excluded from the set of participants.

Lemma 3.1.1. *Malicious participants U_i attempting to use different CK_is to dupe honest participants are removed from the set of participants.*

Proof. There are two methods of attack. One, the malicious participant U_i sends a "wrong" message M_i, so the two honest participants U_j and U_m will obtain two different subkeys CK_i due to the "wrong" message M_i. Take one of the honest participants U_j for example, he sends such a message v_{ji}="failure" since one signature (R_i, S_i) cannot belong to two different subkeys at the same time. On receiving the message, U_i broadcasts the value a_i and subkey CK_i for validation. All honest participants apply a_i in the equation $c_i = g^{a_i} \bmod p$ to validate a_i; next, they apply a_i and c_i in Eqs. (2.1)~(2.4) to verify whether the resulting value is the same as d_{ij}', and then check whether the signature (R_i, S_i) is actually generated for the subkey CK_i. Since U_j declared CK_i to be faulty, at least one of the three verification steps shown above cannot hold. Thus, U_i will be removed from the set of participants. The other method of attack is, U_i sends out the message v_{ij}="failure" and declares U_j who is actually honest as a malicious one. In response to the situation, U_j broadcasts the value a_j and his subkey CK_j for validation. U_i will be proved to be the malicious one when U_j passes verification, and he is then removed from the set of participants.

Lemma 3.1.2 . No honest participant shall be excluded from the set of participants.

Proof. An honest participant U_i broadcasts genuine messages that produce the same subkeys CK_i. When a malicious participant U_j sends out the failure message to frame the honest participant U_i, U_i can clarify the matter for himself by broadcasting the value a_i and his subkey CK_i. Hence, no honest participant shall be excluded from the set of participants.

These two lemmas above prove that malicious participants cannot interfere with the generation of the conference key even if they occupy the majority of the set. Besides attacks from malicious participants, the conference key agreement protocol may also be susceptible to eavesdropping on the broadcasting channel and the impersonator's attack. In the following, the analyses on possible attacks are given and the attacks are proven to be infeasible.

Attack 1. The attacker impersonate an authorized participant U_i to resend the previous key agreement message M.
[Analysis of situation 1] Since the generated key agreement messages M for all conferences have a time stamp T, the participants can verify the validity of the time stamp according to the signature verification equation. Hence message replay attacks are avoided.
Attack 2. The attacker attempts to obtain the private keys of all users from individual public keys or individual signatures.
[Analysis of situation 2] To determine the private key x_i of user U_i using the individual public key $y_i = g^{x_i} \bmod p$ of a user, an attacker will have to first solve the discrete logarithm problem. As to obtaining the private key x_i of a user from the individual signature equation $S_i = x_i H(CK_i \| T) + r_i R_i \bmod q$ where x_i and r_i, an attacker does not have sufficient information to derive x_i since there are two unknown values in the equation. This method of attack will not be feasible.
Attack 3. An attacker attempts to obtain the subkey CK_i of user U_i through transmitted message $M_i = \{T, R_i, S_i, c_i, d_{i1}', d_{i2}', \ldots, d_{in}'\}$.
[Analysis of situation 3] An attacker first computes the session key k_{ij}, and uses the obtained information—the message $M_i = \{T,$

Table 2
Analysis of computation cost

Items of comparison	Protocol by Tzeng	The proposed protocol
Secret distribution and commitment phase	$1T_{L(n)} + nT_{p(n)} + (n+2)T_{EXP} + 2T_{MUL} + 1T_H + 1T_{INV}$	$(n+2)T_{EXP} + (n+2)T_{MUL} + 1T_H$
Subkey computation and verification phase	$nT_{L(n)} + 4nT_{EXP} + nT_{MUL} + nT_H$	$4nT_{EXP} + nT_{MUL} + nT_H$
Fault detection phase	$1T_{L(n)} + (2n+1)T_{p(n)} + 5T_{EXP} + 1T_{MUL} + 1T_H$	$5T_{EXP} + 2T_{MUL} + 1T_H$

Table 3
Analysis of transmission cost

Protocol by Tzeng	The proposed protocol
$(n+1) \mid q \mid +2 \mid p \mid$	$(n+1) \mid q \mid +2 \mid p \mid +\mid T \mid$

R_i, S_i, c_i, d_{i1}', d_{i2}',..., d_{in}'} to obtain the subkey CK_i. However, the operation of session key k_{ij} requires using the private key x_i of user U_i. As shown in the analysis of situation 2, the private key of user cannot be obtained through unfair means. Hence the attack is infeasible.

3.3. Performance analysis

The method of Tzeng [15] was declared efficient. A comparison of the proposed method with Tzeng's method is made to analyze efficiency. The analysis of performance is divided into analyses of computation costs and transmission costs. Computation costs include cost of calculating the conference key message and cost of executing fault detection using other participants' subkeys. Transmission costs include transmission load of messages broadcasted by each participant. Modular addition, modular subtraction and exclusive OR operations have lower computation costs in opposed to modular multiplication or modular exponential operations; hence their computation costs are ignored to make efficiency estimation easier. The mathematical notation is as follows (Table 1).

Table 2 analyzes computation costs. The table shows that the proposed method requires more modular multiplication operation than Tzeng's protocol in the secret distribution and commitment phase and fault detection Phase. However, it does not require the establishment and calculation of an n-power polynomial which eliminates computation costs of $T_{L(n)}$ and $T_{p(n)}$. Hence the proposed method has an overall lower computation load.

Table 3 compares transmission load where $|T|$ represents the bit length of time stamp T. Although the proposed method has a heavier cost in terms of the length of $|T|$; however, it can protect against replay attack. Hence it is more secure than the compared. Besides, the general requirement of p is 512 bits, and 160 bits for q. Not considering $|T|$, the compared method requires $(160n+1184)$ bits of transmission load, and so do ours.

4. Conclusion

There proposes a distributed conference key agreement protocol with efficient fault detection in the paper. Broadcasting a computationally indistinguishable message on the network helps to detect both the malicious participants and the passive attackers before starting a conference. The proposed method enabled quick generation of conference key and the efficient detection and elimination of malicious participants ensured that all participants obtain the same and true conference key.

The security of the proposed protocol is based on the assumed problem—discrete logarithm problem assumption. The protocol is settled under computationally security which ensures that secret information of users cannot be derived from its corresponding public information and hence all secret information is confidential. Besides, the proposed protocol efficiently lowers computation load without compromising on security. Hence the conference key agreement protocol works efficiently in a distributed Internet environment.

References

[1] FIPS PUB 197, Advanced Encryption Standard, Federal Information Processing Standards Publications, US Dept. of Commerce/N.I.S.T., Nov. 2001.
[2] C.E. Shannon, Communication theory of secret systems, Bell System Technical Journal 28 (4) (1949) 656–715.
[3] W. Diffie, M. Hellman, New directions in cryptography, IEEE Transactions on Information Theory 22 (6) (1976) 644–654.
[4] PKCS #3, Diffie-Hellman Key-agreement Standard, RSA Laboratories, Redwood City, California, November 1993.
[5] American National Standards Institute, Accredited Standards Committee X9 Working Draft, Ansi X9.42-1993, Public Key Cryptography for the Financial Services Industry: Management of Symmetric Algorithm Keys Using Diffie-Hellman, American Bankers Association, 1994.
[6] C. Boyd, J.M.G. Nieto, Round-Optimal Contributory Conference Key Agreement, Proceedings of the 2003 International Workshop on Practice and Theory in Public-Key Cryptography, 2003, pp. 161–174.
[7] C. Popescu, A Secure Authenticated Key Agreement Protocol, Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference, vol. 2, 2004, pp. 783–786.
[8] G. Ateniese, M. Steiner, G. Tsudik, New multiparty authentication services and key agreement protocols, IEEE Journal on Selected Areas in Communications 18 (4) (2000) 628–639.
[9] Y. Kim, A. Perrig, G. Tsudik, Simple and Fault-tolerant Key Agreement for Dynamic Collaborative Groups, Proceedings of the Seventh ACM Conference Computer and Communication Security, 2000, pp. 235–244.
[10] B.E. Jung, An efficient group key agreement protocol, IEEE Communications Letters 10 (2) (2006) 106–107.
[11] P.P.C. Lee, J.C.S. Lui, D.K.Y. Yau, Distributed collaborative key agreement and authentication protocols for dynamic peer groups, IEEE/ACM Transactions on Networking 14 (2) (2006) 263–276.
[12] W.H. Kim, E.K. Ryu, J.Y. Im, K.Y. Yoo, New conference key agreement protocol with user anonymity, Computer Standards & Interfaces 27 (2005) 185–190.
[13] Y. Kim, A. Perrig, G. Tsudik, Group key agreement efficient in communication, IEEE Transactions on Computers 53 (7) (2004) 905–921.
[14] M. Steiner, G. Tsudik, M. Waidner, Key agreement in dynamic peer groups, IEEE Transactions on Parallel and Distributed Systems 11 (8) (2000) 769–780.
[15] W.G. Tzeng, A secure fault-tolerant conference key agreement protocol, IEEE Transactions on Computers 51 (4) (2002) 373–379.

Kuo-Hsuan Huang received the B.S. and the M.S. degrees from Dayeh University in 2001 and 2003 respectively, both in Computer Science and Information Engineering, Taiwan. He is currently a Ph.D. candidate in Computer Science of the Electrical Engineering Department in National Taiwan University, doing research, i.e., Information Security, Cryptography, and Medical System Security.

Yu-Fang Chung received the B.A. degree in English Language, Literature and Linguistics from Providence University in 1994, the M.S. degree from Dayeh University in 2003, and the Ph.D. degree from National Taiwan University in 2007, both in Computer Science, Taiwan. She is currently an assistant professor in the Departments of Electronic Engineering and Information Management at Tunghai University, doing research, i.e., Information Security and Cryptography.

Hsiu-Hui Lee received the B.S. and the M.S. degrees in Electrical Engineering in 1980 and 1982, respectively, and the Ph.D. degree in Computer Science and Information Engineering in 1992, all from National Taiwan University (NTU), Taipei, Taiwan. In 1982, she was an instructor at the Department of Computer Science and Information Engineering of Tamkang University. Since 1983, she has been on the faculty and currently is an associate professor of the Department of Computer Science and Information Engineering at NTU. From August 2003 to July 2005, she also served as the Chief of the Information Management Division of the Computer & Information Networking Center at NTU and the Vice Director of the Information Systems Office of National Taiwan University Hospital. Her research interests include distributed object system, temporal database, and medical information system.

Feipei Lai received a B.S.E.E. degree from National Taiwan University in 1980, and M.S. and Ph.D. degrees in Computer Science from the University of Illinois at Urbana-Champaign in 1984 and 1987, respectively. He is a professor in the Graduate Institute of Biomedical Electronics and Bioinformatics, the Department of Computer Science & Information Engineering and the Department of Electrical Engineering at National Taiwan University. He is the vice superintendent of the National Taiwan University Hospital. He is the chairman of Taiwan Network Information Center. He was a visiting professor in the Department of Computer Science and Engineering at the University of Minnesota, Minneapolis, USA. He was also a guest Professor at the University of Dortmund, Germany and a visiting senior computer system engineer in the Center for Supercomputing Research and Development at the University of Illinois at Urbana-Champaign. Dr. Lai holds 6 Taiwan patents and 3 USA patents currently. His current research interests are SOC low power computing and medical information system. Prof. Lai is one of the founders of the Institute of Information & Computing Machinery. He is also a member of Phi Kappa Phi, Phi Tau Phi, ACM, and the Chinese Institute of Engineers. Dr. Lai is the chairman of Taiwan Internet Content Rating Foundation. He received the Taiwan Fuji Xerox Research award in 1991. Dr. Lai is a senior member of IEEE and included in "Who's Who in Science and Engineering" and "Who's Who in the World."

Tzer-Shyong Chen received the B.S. degree from Tunghai University in 1989, the M.S. degree from National Chiao Tung University in 1991, and the Ph.D. degree from National Taiwan University in 1996, all in Computer Science, Taiwan. He is currently a professor in the Department of Information Management at Tunghai University, doing research, i.e., Information Security, Cryptography, and Network Security.