FRAMING HONEST PARTICIPANTS IN A FAULT TOLERANT GROUP KEY AGREEMENT PROTOCOL

[1]ZIBA ESLAMI, [2]SAEIDEH KABIRI-RAD

[1,2]Department of Computer Science, Shahid Beheshti University, G.C.,Tehran, Iran
Email: [1]z_eslami@sbu.ac.ir, [2]s_kabirirad@sbu.ac.ir

Abstract: Recently, Zhao et al. proposed an efficient fault tolerant group key agreement protocol.In this paper, we show that the fault detection technique of their protocol has a security flaw. We prove that a malicious member of the group can easily frame an honest participant such that when the fault detection is executed, then the honest member will be marked malicious and get excluded while the malicious participant can remain in the group. We then propose an improvement to overcome this attack.

Keywords: Cryptography, Group key agreement, Fault-tolerance, Efficiency, Attack.

I. INTRODUCTION

Due to technological advances in computer networks, group communications have become increasingly popular. However, in transmitting information over open channels such as the Internet, messages should be encrypted to safeguard against security threats such as eavesdropping, message forgery or impersonation of participants. Therefore, a cryptographic protocol must be employed to establish a common key among the group members such that malicious behavior can be detected.

The cryptographic primitive dealing with this problem is called a group key establishment protocol. There are two kinds of group key establishment protocols: group key distribution and group key agreement. In group key distribution protocols, there is a trusted entity who is responsible for generating and distributing the group key while a group key agreement protocol involves all participants cooperatively establishing the key. The security in group key distribution protocols might be circumvented by a malicious chairperson, for example by sending afaulty key to some participants but group key agreement protocols do not have this drawback.The first key agreement protocol that enabled two participants to establish a common key using their own secret information and some other publicly exchanged information was proposed in 1976 by Diffie and Hellman [1]. The security of the scheme was based on the discrete logarithm problem and it was not suitable for groups of users. In 1982, Ingemaresson et al. [2] proposed the first conference-key establishment protocol that enabled a group of participants to establish a common key.

A group key agreement may suffer from malicious attacks for example a malicious participant attempts to disrupt the establishment of the conferencekey, other honest participants will be unable to compute the same conference key, and thus will be unable to identify the malicious participant. To overcome this drawback, fault-tolerant GKA protocols [3-10]Weredevelopedto assure that all malicious participants are excluded from the set of participants and that no honest participants are excluded. Tzeng[5] proposes a polynomial-based method to realize fault-tolerance which requires that each participant create nn-power polynomials, where n is the number of participants, this process results in massive computational cost. Also in 2004, Yi [11] pointed out that Tzeng's protocol might suffer from a different subkey attack. Tseng [8] proposes a communication-efficient GKA protocol that enjoys the fault-tolerance and forward secrecy with a constant message size and round efficiency. The recent result is given by Huang et al.[4], their protocol can filter out malicious participants at the beginning of the key agreement process and with lower computational cost than Tseng.Zhao et al. [12] proposed a fault-tolerant GKA protocol named efficient GKA (EGKA) with lower computational cost and average communication cost than all three fault-tolerant GKA protocols mentioned in [5,8,4]. However, (EGKA) suffers from a weakness such that it is possible for an honest group member to be marked as malicious and get excluded. In this paper we prove that if a malicious participant U_m publishes the value of only one honest participant U_h dishonestly, U_h is detected as malicious and while U_m can remain in the set of honest participants. Then we improve the scheme such that remove this fault.
The rest of this paper is organized as follows. In Section 2, a brief review of (EGKA) scheme is presented. The details of the weakness will be described in Section 3. Then, in section 4 we improve (EGKA) and finally, conclusions are given in the next section.

II. REVIEW OF ZHAO ET AL.'S SCHEME

In this section, we briefly review the (EGKA) protocol [12] which is composed of five phases.It is assumed that (EGKA) will run in a static network where authorized group members are determined in advance.We first introduce notations that will be used throughout the paper.

TS	The trusted server in registration phase,
$U = \{U_1, U_2, \dots, U_n\}$	The set of participants,
ID_i	Identity of U_i,
T_i	System clock time of U_i,
$(p_i, q_i, N_i, f_i, s_i)$	A set of RSA parameters forU_i, where p_i and q_i are two large primes, $N_i = p_i q_i$ and $f_i s_i = 1\ (mod\ (p_i - 1)(q_i - 1))$. (p_i, q_i, s_i)areprivate, while (N_i, f_i) are publicsuch that $m^{s_i \cdot f_i} = m^{f_i \cdot s_i} = m\ (mod\ N_i)$, for all m.
v_{ji}	The result of judging honesty of U_i by U_j, i.e.,$v_{ji} = failure$ if and only if U_j detects that U_i is malicious,
SK	Session key of the protocol,
$h(M)$	A collision-free one-way hash function h applied to message M,
$\|\|$	The concatenation operator.

2.1. Registration phase

TSperforms the following steps:

1. Generates for each $\{U_i\}_{i=1}^n$ a set of RSA parameters $(p_i, q_i, N_i, f_i, s_i)$,
2. Sends s_i to U_i, $1 \le i \le n$ via a secure channel,
3. Publishes $\{(f_1, N_1), \dots, (f_n, N_n)\}$.

2.2. Sub-key distribution and commitment phase

Each $\{U_i\}_{i=1}^n$ performs the following steps:

1. Chooses a $K_i \in Z_{N_i}$ as his own sub-key,
2. Computes $R_i = (K_i)^{s_i}\ (mod\ N_i)$,
3. Computes $I_{ij} = (R_i||ID_i)^{f_j}\ (mod\ N_j)forU_j (i \neq j)$,
4. Computes $h_i = h(K_i||T_i)(mod\ N_i)$,
5. Publishes$M_i = \{T_i, h_i, I_{i1}, \dots, I_{i(i-1)}, I_{i(i+1)}, \dots, I_{in}\}$.

2.3. Sub-key recovery and verification phase

Each $\{U_j\}_{j=1}^n$ performs the following steps:

1. Receives M_i from $U_i, i \neq j$,
2. If $T_i$is not reasonable, claims that U_i is faulty,
   else runs the following steps:
3. Computes $(I_{ij})^{s_j} = R_i'||ID_i\ (mod\ N_j)$ to obtain R_i' and ID_i,
4. Computes $K_i' = (R_i')^{f_i}(mod\ N_i)$,
5. Computes $h_i' = h(K_i'||T_i)(mod\ N_i)$,
6. If $h_i' = h_i\ (mod\ N_i)$ then broadcasts $v_{ji} = success$, else $v_{ji} = failure$ is broadcast,

Now, at the end of this phase, if we have $v_{ji} = failure$, then the group executes the fault detection phase. However, if all group members are honest, i.e. we have $v_{ji} = success$ for all $i$and j, the participants directly execute the key computation phase.

2.2. Fault detection phase

This phase is run if there are i and j such that $v_{ji} = failure$. In this case there are two possibilities:

1. $v_{ji} = failure$ and U_i is the real malicious participant,
2. $v_{ji} = failure$andU_j is the malicious participant. In this case, U_j tries to cheat honest participants into excluding the honest participant U_i.

In order to detect the real adversary, the following procedure is proposed in [12]:

Participants wait for U_i to send them the fault detection messages $R_i$and K_i. If no one can receive this information from U_i in a valid period, then U_i is detected as a malicious participant, otherwise the group proceeds as follows:

Each participant U_k $(k \neq i)$ performs the following steps:

1. Receives R_i and K_i from U_i,
2. Computes $(R_i||ID_i)^{f_k}\ (mod\ N_k)$,
3. If $(R_i||ID_i)^{f_k} \neq I_{ik}\ (mod\ N_k)$, then U_i must be the malicious participant,
4. Else
   4.1. Computes $h"_i = h(K_i||T_i)(mod\ N_i)$,
   4.2. If $h_i'' \equiv h_i\ (mod\ N_i)$, then U_j is malicious, else U_i is malicious.

The malicious participants are then excluded from the group and the protocol is restarted.

2.5. Session key computation phase

Suppose that the set $U' = \{U_1', U_2', \dots, U_{n'}'\}$ is the set of honest members. Now, the conference key SK is computed as $SK = K_1' + K_2' + \cdots + K_{n'}'$ by each U_i and the sub-keys $\{K_i\}_{i=1}^{n'}$ are destroyed.

III. HOW TO FRAME UP HONEST PARTICIPANTS IN (EGKA)

In this section, we show that the fault detection phase as proposed in [12] has a security flaw such that it is possible for an honest group member to be marked as malicious and get excluded from the group. We prove that a malicious participant U_m can easily cause an honest participant U_h to broadcast $v_{hm} = failure$. However, when the fault detection is executed, then (honest) U_h will be marked malicious while (malicious) U_m can remain in the set of honest participants.

Theorem 1. Let $\mathcal{U} = \{U_1, \dots, U_n\}$ be a set of n users who run the protocol (EGKA). Suppose that U_h is an honest participant while U_m acts maliciously and uses two different sub-keys K_m and K_m''.

Let the values published by U_m in sub-key distribution and commitment phase, be $M_m = \{T_m, h_m, I_{m1}, \dots, I_{m(h-1)}, I''_{m(h)}, I_{m(h+1)}, \dots, I_{mn}\}$, where $I''_{m(h)} = (R''_m||ID_m)^{f_h}(mod\ N_h)$, $R''_m = (K''_m)^{s_m}(mod\ N_m)$ and $K''_m \neq K_m(mod\ N_m)$. Then the following holds:

A) At the end of sub-key recovery and verification phase, U_h broadcasts $v_{hm} = failure$.

*B)*In fault detection phase, if U_m sends his real sub-key K_m then U_h is detected malicious and will be excluded from the group.

Proof 1.

(A) Note that in sub-key distribution and commitment phase, U_m uses the fake sub-key K''_m to produce values corresponding to U_h (i.e. R''_m, I''_{mh}). Therefore, U_h obtains the following during sub-key recovery and verification phase:

Step 3: $(I''_{mh})^{s_h} = R'_m||ID_m = R''_m||ID_m\ (mod\ N_h)$,
Step 4: $K'_m = (R'_m)^{f_m} = (R''_m)^{f_m}(mod N_m) = K''_m$,

Therefore $(K'_m = K''_m) \neq K_m$. Now, in step 5, he computes $h'_m = h(K'_m||T_m) = h(K''_m||T_m)$ which is different from $h_m(= h(k_m||T_m))$ and hence U_h broadcasts $v_{hm} = failure$. However, since U_m publishes all other values using his real sub-key K_m, we have $v_{km} = success$, for all $k \neq h$.

*(B)*Consider a participants $U_k, k \neq h$. During fault detection, he receives K_m and R_m from U_m, then computes $(R_m||ID_m)\hat{}(f_k)(mod N_k)$. Since I_{mk} has been computed with U_m's real sub-key, then $(R_m||ID_m)^{f_k} = I_{mk}$.

IV. IMPROVED FAULT DETECTION FOR (EGKA)

In this section, we improve fault detection phase of (EGKA) so that the framing attack would not take place. The idea is that when $v_{ji} = failure$, then in fault detection phase, we should check values of both U_i and U_j to detect the malicious participant. The details are as follows.

Each participant $U_k(k \neq i)$ performs the following steps:

1. Receives R_i and K_i from U_i,
2. Computes $(R_i||ID_i)^{f_k}(mod\ N_k)$,
3. If $(R_i||ID_i)^{f_k} \neq I_{ik}(mod\ N_k)$, then U_i must be the malicious participant,
4. Else
   - 4.1. Computes $h'_i{}' = h(K_i||T_i)(mod\ N_i)$,
   - 4.2. If $h'_i{}' \neq h_i(mod\ N_i)$, then U_i is malicious,
   - 4.3. Else
     - 4.3.1. Computes $(R_i||ID_i)^{f_j}(mod\ N_j)$
     - 4.3.2. If $(R_i||ID_i)^{f_j} = I_{ij}(mod\ N_j)$, then U_j is malicious, else U_i is malicious.

.

Now, it is easy to see that the framing attack described in Section 3 does not work with the improved version.

CONCLUSIONS

In this paper, we show that the (EGKA) protocol from Zhao et al. is such that it is possible for an honest group member to be marked as malicious and get excluded from the group. We then propose an improvement to overcome this security weakness.

REFERENCES

[1] W. Diffie and M. Hellman, New directions in cryptography, IEEE Transactions on InformationTheory 22(6) (1976) 644-654.

[2] I. Ingemaresson, T. D. Tang and C. K. Wong, A conference key distribution system, IEEETrans. Info. Theory 28(5) (1982) 714-720.

[3] J. C. Cheng and C. S. Laih, Conference key agreement protocol with non-interactive faulttoleranceover broadcast network, International Journal of Information Security, SpringerVerlag8 (2009) 37-48.

[4] K. H. Huang, Y. F. Chung, H. H. Lee, F. Lai and T. S. Chen, A conference key agreement protocolwith fault tolerant capability, Computer Standards and Interfaces 31 (2009) 401-405.

[5] W. Tzeng, A secure fault-tolerant conference key agreement protocol, IEEE Transactions onComputers 51 (4) (2002) 373-379.

[6] Y. M. Tseng, Multi-party key agreement protocols with cheater identification, AppliedMathematics and Computation 145 (2003) 551-559.

[7] Y. Tseng, An improved conference-key agreement protocol with forward secrecy, Informatica16 (2) (2005) 275-284.

[8] Y. Tseng, A communication-efficient and fault-tolerant conference-key agreement protocolwith forward secrecy, Journal of Systems and Software 80 (7) (2007) 1091-1101.

[9] S. Lee, J. Kim and S. Hong, Security weakness of Tseng's fault-tolerant conference-key agreementprotocol, Journal of Systems and Software 82 (7) (2009) 1163-1167.

[10] M. H. Zheng, H. H. Zhou, G. H. Cui and L. S. Han, A provable secure group key agreementprotocol with fault-tolerant, Tien Tzu Hsueh Pao/Acta Electronica Sinica 37 (11) (2009)2396-2402.

[11] X. Yi, Identity-based fault-tolerant conference key agreement, IEEE Trans. Depend. SecurityComputation 1 (3) (2004) 170-178.

[12] J. Zhao, D. Gu and Y. Li, An efficient fault-tolerant group key agreement protocol, Computer Communications 33 890-895.

★★★