On the Vulnerability of a Group Key Transfer Protocol based on Secret Sharing

Ruxandra F. Olimid[*][†]
[*]Department of Computer Science,
University of Bucharest, Romania
E-mail: ruxandra.olimid@fmi.unibuc.ro

***Abstract*—Group Key Transfer (GKT) protocols allow multiple parties to share a common secret key: a trusted entity selects a private key and securely distributes it to the qualified participants. Hsu et al. introduced a GKT protocol based on secret sharing, which they claimed to be secure. Unlike their affirmation, we report a vulnerability: an insider can cancel key consistency such that at the end of the protocol distinct users own different keys. This leads to the futility of the protocol. Even more, the attacker is able to choose the values of the injected keys on his own wish. Finally, we propose a simple and efficient countermeasure that stands against the revealed attack.**

I. Introduction

A general trend of today's society is to benefit of group applications that allow multiple users to share the same resources or perform collaborative tasks. Applications like audio-video conferences, data sharing or collaborative computing are used daily in offices and homes all over the world. In this context, group security becomes an important duty: it guarantees that only authorized users are allowed to access the resources or perform different tasks within the group.

To achieve security, most protocols require a key establishment phase, which grant all users a common secret session key. The key is subsequently used to achieve the main goals of secure communication (confidentiality, authentication, integrity).

A *Group (or conference) Key Establishment (GKE)* protocol is a key establishment protocol run by more than two parties. GKE divide into two classes: *Group Key Transfer (GKT)* protocols and *Group Key Agreement (GKA)* protocols. GKT protocols rely on a trusted entity called *Key Generation Center (KGC)* to select a private key, which is then secretly distributed to all qualified users; sometimes, one of the users (the initiator of the protocol) may play the role of the KGC. In contrast, GKA protocols do not assume the existence of a trusted entity; the common key is agreed by the cooperation of all communication users.

Secret sharing schemes are widely used as building blocks of GKE protocols, due to the advantages they provide: a convenient way to differentiate between principals power within the group, delegation of duties by passing shares to other participants, group authentication instead of entity authentication, cheating detection, simple management of group sizing using the accepted threshold [1]. Secret sharing represents a method to split a secret into multiple shares, which are then distributed to the participants via secure channels. The secret can be recovered only when the members of an authorized subset of participants combine their shares together.

A. Motivation and personal contributions

Many papers published in the last years introduce GKT protocols based on secret sharing that lack security proofs. The omission of a solid security analysis leads to a high probability to discover vulnerabilities. The following examples support our claim: Nam et al. [2] disclosed a replay attack against Harn and Lin's protocol [3], Olimid [4] and Kim [5] revealed several attacks against Sun et al.'s construction [6] and Olimid [7] mounted an insider attack against Yuan et al.'s proposal [8].

We enrich the list of weak GKT protocols by showing the insecurity of Hsu et al.'s construction [9].

First, we prove that it fails against an inside attack. More precisely, we show that an active insider can always ruin the protocol in the sense that distinct users accept with different group keys. In addition, the values of these keys are not random, but chosen by the adversary. The vulnerability is caused by an authenticity flaw, which permits the attacker to impersonate the initiator.

Second, we suggest a simple and efficient countermeasure against the exposed vulnerability. We highlight that our aim is to give a solution that stands against the proposed attack; we do not affirm that the improved version is secure against other attacks.

B. Outline

The paper is organized as follows. The next section contains the preliminaries. Section III describes Hsu et al.'s protocol. Section IV reveals the proposed attack. Section V analyzes possible countermeasures. Last section concludes.

II. Preliminaries

A. Informal Security Goals

A group key establishment protocol should satisfy a set of properties, which we informally recall next.

The key must remain hidden for any party except the authorized participants, even if the protocol runs for multiple times, called sessions (*key confidentiality*). More, even if a session key is compromised, this should have no impact on other session keys (*known key security*). Ideally, no information about the key should be disclosed regardless the adversary's behavior in future or past runs of the protocol (*forward* and *backward secrecy*).

Users should be confident that the group key has been randomly chosen (*key randomness*) and has never been used before (*key freshness*). In case of GKT protocols, this introduces a mandatory trust assumption: all parties must trust the KGC (or the initiator) to generate a key that fulfills these properties; in case of GKA protocols, this is achieved by construction when the inputs of all users have the same impact on the key computation and no party is able to force the key to a chosen value (*key control*).

Authorized users must be aware of the identity of the other parties involved in the key establishment (*key* and *entity authentication*). This implies that it should be impossible for the adversary to *impersonate* other parties or for a user to be tricked to believe that he shares the key with someone else (*unknown key share*).

Although confidentiality and authentication guarantee that only qualified members may obtain the group key, a participant has no assurance that the others have actually computed the key. Such an extra requirement is called *key confirmation* and certifies each user that the others really obtained the same group key. We introduce a Key Confirmation phase inspired by the work of Boyd [10] in Section V.

Regarding the appartenance to the group, a key establishment protocol must stand against two types of adversaries: *outsiders* and *insiders*. An outsider is a non-registered group member who can never take part to protocol executions, while an insider is a registered group member who may initiate or take part to protocol sessions. Clearly, an insider is more powerful than an outsider and therefore should be considered while analyzing the security of a protocol.

In the present paper we consider an active insider (i.e. has full control over the communication channel, being able to modify, delete or inject messages) that aims to ruin *key consistency*, such that at the end of the protocol distinct users own different keys; the attacker can choose the values of the keys he injects to the participants. Even if the attack can be easily detected at application runtime, it leads to the futility of the protocol. We exemplify such an attack against Hsu et al.'s GKT protocol in Section IV.

B. Security Assumptions

We review two main security assumption Hsu et al.'s protocol relies on.

1) Computational Diffie-Hellman (CDH) assumption: Let G be a multiplicative cyclic group of prime order p, with g as generator. The Computational Diffie-Hellman (CDH) assumption holds if given g, g^a and g^b, any probabilistic polynomial-time adversary $\mathcal{A}$ has a negligible probability in computing g^{ab}:

$$Adv_{\mathcal{A}}^{CDH} = Pr[\mathcal{A}(p, g, g^a, g^b) = g^{ab}] \leq negl(\mathcal{K}) \quad (1)$$

where $a, b \in \mathbb{Z}_p^*$ are random and $\mathcal{K}$ is the security parameter.

2) Discrete Logarithm Problem (DLP): Let G be a multiplicative cyclic group of prime order p, with g as generator. The Discrete Logarithm (DLP) is hard in G if given g^a, any probabilistic polynomial-time adversary $\mathcal{A}$ has a negligible probability in computing a:

$$Adv_{\mathcal{A}}^{DLP} = Pr[\mathcal{A}(p, g, g^a) = a] \leq negl(\mathcal{K}) \quad (2)$$

where $a \in \mathbb{Z}_p^*$ is random and $\mathcal{K}$ is the security parameter.

C. Notations

We introduce next the notations we will use for the rest of the paper.

Let $\mathcal{U} = \{U_1, \ldots, U_m\}$ be the set of all possible users, $\{U_1, \ldots, U_t\}$, $t \leq m$ the subset of authorized participants to a given session with U_1 as initiator (after a possible reordering), (pk_i, sk_i) the public-private key pair of U_i such that $pk_i = g^{sk_i}$ in a multiplicative cyclic group G, $Sig_{U_i}(M)$ the signature of U_i on a message M, Ver_{U_i} the corresponding public verification and h a collision-resistant hash function. We consider $U_a \in \{U_2, \ldots, U_m\}$ to be an active attacker, other than the initiator, whose goal is to break key consistency (the initiator has no interest in ruin the protocol; besides, for $U_a = U_1$ the attack is trivial).

We denote by $\leftarrow^R X$ a uniformly random choice from a specified set of values X, $\|$ string concatenation and $A \rightarrow^* $ a broadcast message originating from A.

III. Hsu et al.'s Protocol

In 2012, Hsu et al. introduced a GKT protocol based on a secret sharing [9]. Figure 1 describes the protocol in detail.

The protocol assumes that the DLP is hard in G (i.e. given $pk_i = g^{sk_i}$ it is computationally infeasible to compute sk_i, $i = 1, \ldots, m$) and that CDH holds (i.e. given pk_i and pk_j it is computationally infeasible to compute $g^{sk_i sk_j}$, $i, j = 1, \ldots, m$, $i \neq j$). Although the computations are performed within the group G, we do not explicit specify that during protocol description, since it is clear from the context.

We highlight next some advantages of the protocol.

First, unlike the majority of GKT protocols, the construction does not inquire an online KGC - the initiator U_1 performs this role.

Second, the key transfer is performed over public channels only. The existence of secure channels over public networks is a strong assumption, which usually represents a limitation of GKT, imposed by a previously shared secret between the KGC and each group member (during the Users Registration phase). In Hsu et al.'s proposal, each participant U_i, $i = 1, \ldots, m$, must own a public-secret key pair (pk_i, sk_i) authenticated by a trusted authority with a certificate. Then, the initiator U_1 establishes a common secret key with each other participant at runtime, which has the advantage to be fresh for each session.

Initialization. Let G be a multiplicative cyclic group of order p, with g as generator, where p is a large safe prime (i.e. $p' = \frac{p-1}{2}$ is also prime);

Users Registration. Each user U_i, $i = 1, \ldots, m$ owns a public-private key pair (pk_i, sk_i) s.t. $pk_i = g^{sk_i}$ in G;

Round 1. User U_1:

1.1. chooses $r_1 \leftarrow^R \mathbb{Z}_p^*$;

1.2. sends a key generation request:

$$U_1 \rightarrow^*: (\{U_1, \ldots, U_t\}, r_1, pk_1)$$

Round 2. Each user $U_i, i = 2, \ldots, t$:

2.1. chooses $r_i \leftarrow^R \mathbb{Z}_p^*$;

2.2. computes $S_i = pk_1^{sk_i r_i r_1}$ a shared secret with U_1 and $Auth_i = h(S_i, r_1)$;

2.3. broadcasts:

$$U_i \rightarrow^*: (r_i, pk_i, Auth_i)$$

Round 3. User U_1:

3.1. computes $S_i = pk_i^{sk_1 r_i r_1}$, $i = 2, \ldots, t$;

3.2. checks if $Auth_i = h(S_i, r_1)$, $i = 2, \ldots, t$; If at least one equality does not hold, he quits;

3.3. chooses the group key $k \leftarrow^R \mathbb{Z}_p^*$, splits each secret S_i into two parts $S_i = x_i || y_i$ and computes $t - 1$ values $K_i = k - T_i$, where $T_i = (y_i v(x_i), r)$ is the inner product of the vectors $y_i v(x_i) = y_i \sum_{j=1}^{t} x_i^{j-1} e_j$ ($e_j = (0, \ldots, 1, \ldots, 0)$ with 1 on position j), $i = 2, \ldots, t$ and $r = (r_1, \ldots, r_t)$;

3.4. computes $Auth = h(k, U_1, \ldots, U_t, r_1, \ldots, r_t, K_2, \ldots, K_t)$;

3.5. broadcasts:

$$U_1 \rightarrow^*: (Auth, K_2, \ldots, K_t)$$

Key Computation. Each user U_i, $i = 2, \ldots, t$:

4.1. computes the inner product $T_i = (y_i v(x_i), r)$, recovers the group key $k = T_i + K_i$;

4.2. checks if $Auth = h(k, U_1, \ldots, U_t, r_1, \ldots, r_t, K_2, \ldots, K_t)$. If the equality does not hold, he quits.

Fig. 1. Original Version of Hsu et al.'s Group Key Transfer Protocol [9]

Third, the protocol allows each user U_i to recover the group key by only using his two corresponding shares: T_i (computed by the user in the Key Computation phase) and K_i (received from the initiator in Round 3). No additional interaction between participants is required for key computation.

IV. Insider Attack

Hsu et al.'s protocol is vulnerable to an active attack mounted from inside. The adversary succeeds to make different authorized users accept distinct keys and therefore break key consistency: any qualified user $U_i \in \mathcal{U} \setminus \{U_a, U_1\}$ (except the initiator) computes a key k_i that has been chosen in advance by the adversary. The attack assumes that U_a is able to: (1) intercept exchanged messages; (2) prevent messages from reaching their destination and (3) inject messages on his own choice.

Step 1. U_a is qualified to participate to a protocol session and hence he finds the key:

$$k = T_a + K_a$$

Step 2. U_a intercepts the broadcast message sent in Round 3 of the protocol and prevents it from reaching U_i, $U_i \in \mathcal{U} \setminus \{U_a, U_1\}$;

Step 3. U_a eavesdrops on K_i and therefore computes:

$$T_i = k - K_i$$

Step 4. U_a chooses a key k_i, computes the corresponding value

$$K_i' = k_i - T_i$$

and authenticates his selection as:

$$Auth_i = h(k_i, U_1, \ldots, U_t, r_1, \ldots, r_t, K_2, \ldots, K_i', \ldots K_t)$$

Step 5. U_a sends:

$$U_a \rightarrow U_i : (Auth_i, K_2, \ldots, K_i', \ldots, K_t)$$

Step 6. U_i recovers the correct value T_i, computes the group key

$$k_i = T_i + K_i'$$

and checks that

$$Auth = h(k_i, U_1, \ldots, U_t, r_1, \ldots, r_t, K_2, \ldots, K_i', \ldots, , K_t)$$

holds. U_i accepts k_i as the correct group key.

Fig. 2. Insider Attack against the Original Version of Hsu et al.'s Protocol

Figure 2 reveals the attack. It is self-contained and hence we omit other comments. We only emphasize that the user U_i is unable to discover the attack during the protocol execution: $U_i \in \mathcal{U} \setminus \{U_a, U_1\}$ recovers the correct value T_i, but then computes the group key as $k_i = T_i + K_i'$. The verification holds in the Key Computation phase, because the attacker was able to compute $Auth_i$ based on his own choice of k_i. Therefore, U_i considers k_i to be the correct group key.

We remark that U_a can mount the same attack simultaneously against multiple users to induce each one a different key. More, the adversary's identity remains hidden, which allows him to attack several times, without being detected and excluded from the group.

V. Countermeasures

The previous attack is caused by an authentication flaw: the group key k is not properly authenticated as originating from the initiator U_1. This allows the adversary to impersonate U_i and send a modified but valid authentication message that helps him to achieve his goal.

A trivial way to detect the attack is immediate: a Key Confirmation phase assures that all users posses the correct key [10]. Figure 3 describes this enhancement: each user signs the group key he obtained and broadcasts it to the other members.

Initialization. Let G be a multiplicative cyclic group of order p, with g as generator, where p is a large safe prime (i.e. $p' = \frac{p-1}{2}$ is also prime);

Users Registration. Each user U_i, $i = 1, \ldots, m$ owns a public-private key pair (pk_i, sk_i) s.t. $pk_i = g^{sk_i}$ in G;

Round 1. User U_1:
1.1. chooses $r_1 \xleftarrow{R} \mathbb{Z}_p^*$;
1.2. sends a key generation request:
$$U_1 \rightarrow^*: (\{U_1, \ldots, U_t\}, r_1, pk_1)$$

Round 2. Each user $U_i, i = 2, \ldots, t$:
2.1. chooses $r_i \xleftarrow{R} \mathbb{Z}_p^*$;
2.2. computes $S_i = pk_1^{sk_i r_i r_1}$ a shared secret with U_1 and $Auth_i = h(S_i, r_1)$;
2.3. broadcasts:
$$U_i \rightarrow^*: (r_i, pk_i, Auth_i)$$

Round 3. User U_1:
3.1. computes $S_i = pk_i^{sk_1 r_i r_1}$, $i = 2, \ldots, t$;
3.2. checks if $Auth_i = h(S_i, r_1)$, $i = 2, \ldots, t$; If at least one equality does not hold, he quits;
3.3. chooses the group key $k \xleftarrow{R} \mathbb{Z}_p^*$, splits each secret S_i into two parts $S_i = x_i || y_i$ and computes $t-1$ values $K_i = k - T_i$, where $T_i = (y_i v(x_i), r)$ is the inner product of the vectors $y_i v(x_i) = y_i \sum_{j=1}^{t} x_i^{j-1} e_j$ ($e_j = (0, \ldots, 1, \ldots, 0)$ with 1 on position j), $i = 2, \ldots, t$ and $r = (r_1, \ldots, r_t)$;
3.4. computes $Auth = h(k, U_1, \ldots, U_t, r_1, \ldots, r_t, K_2, \ldots, K_t)$;
3.5. broadcasts:
$$U_1 \rightarrow^*: (Auth, K_2, \ldots, K_t)$$

Key Computation. Each user U_i, $i = 2, \ldots, t$:
4.1. computes the inner product $T_i = (y_i v(x_i), r)$, recovers the group key $k = T_i + K_i$;
4.2. checks if $Auth = h(k, U_1, \ldots, U_t, r_1, \ldots, r_t, K_2, \ldots, K_t)$. If the equality does not hold, he quits.

Key Confirmation. Each user U_i, $i = 1, \ldots, t$:
5.1. computes $V_i = Sig_{U_i}(h(k, U_1, \ldots, U_t, r_1, \ldots, r_t, K_2, \ldots, K_t))$;
5.2. broadcasts:
$$U_i \rightarrow^*: V_i$$
5.3. checks if $Ver_{U_j}(V_j, h(k, U_1, \ldots, U_t, r_1, \ldots, r_t, K_2, \ldots, K_t))$ holds for all $j = 1, \ldots, t$, $j \neq i$. If at least one equality does not hold, he quits.

Fig. 3. First Improved Version of Hsu et al.'s Group Key Transfer Protocol

To maintain the confidentiality of its value, the key is first hashed, along with some of the public values used during the protocol execution.

Initialization. Let G be a multiplicative cyclic group of order p, with g as generator, where p is a large safe prime (i.e. $p' = \frac{p-1}{2}$ is also prime);

Users Registration. Each user U_i, $i = 1, \ldots, m$ owns a public-private key pair (pk_i, sk_i) s.t. $pk_i = g^{sk_i}$ in G;

Round 1. User U_1:
1.1. chooses $r_1 \xleftarrow{R} \mathbb{Z}_p^*$;
1.2. sends a key generation request:
$$U_1 \rightarrow^*: (\{U_1, \ldots, U_t\}, r_1, pk_1)$$

Round 2. Each user $U_i, i = 2, \ldots, t$:
2.1. chooses $r_i \xleftarrow{R} \mathbb{Z}_p^*$;
2.2. computes $S_i = pk_1^{sk_i r_i r_1}$ a shared secret with U_1 and $Auth_i = h(S_i, r_1)$;
2.3. broadcasts:
$$U_i \rightarrow^*: (r_i, pk_i, Auth_i)$$

Round 3. User U_1:
3.1. computes $S_i = pk_i^{sk_1 r_i r_1}$, $i = 2, \ldots, t$;
3.2. checks if $Auth_i = h(S_i, r_1)$, $i = 2, \ldots, t$. If at least one equality does not hold, he quits;
3.3. chooses the group key $k \xleftarrow{R} \mathbb{Z}_p^*$, splits each secret S_i into two parts $S_i = x_i || y_i$ and computes $t-1$ values $K_i = k - T_i$, where $T_i = (y_i v(x_i), r)$ is the inner product of the vectors $y_i v(x_i) = y_i \sum_{j=1}^{t} x_i^{j-1} e_j$ ($e_j = (0, \ldots, 1, \ldots, 0)$ with 1 on position j), $i = 2, \ldots, t$ and $r = (r_1, \ldots, r_t)$;
3.4. computes $Auth = Sig_{U_1}(h(k, U_1, \ldots, U_t, r_1, \ldots, r_t, K_2, \ldots, K_t))$;
3.5. broadcasts:
$$U_1 \rightarrow^*: (Auth, K_2, \ldots, K_t)$$

Key Computation. Each user U_i, $i = 2, \ldots, t$:
4.1. computes the inner product $T_i = (y_i v(x_i), r)$, recovers the group key $k = T_i + K_i$;
4.2. checks if $Ver_{U_1}(Auth, h(k, U_1, \ldots, U_t, r_1, \ldots, r_t, K_2, \ldots, K_t))$ holds. If the equality does not hold, he quits;

Fig. 4. Second Improved Version of Hsu et al.'s Group Key Transfer Protocol

The drawback of this solution is clear - the computational and transmission costs significantly increase: in addition to the original protocol, each user generates one signature and verifies $t-1$ others, respectively an additional round of communication is necessary and t more broadcast messages circulate over the network. More, this approach does not eliminate the attack, but only reveals it during the key generation process, in advance to the execution of the application. We remark that such attacks are usually disclosed at runtime, since the users realize that they cannot properly achieve their tasks (for example they are not able to communicate between

TABLE I
COMPARISON OF COMPUTATIONAL AND TRANSMISSION OVERHEAD

	No. of Sig	No. of Ver	No. of rounds	No. of broadcast messages
Trivial countermeasure	t	$t(t-1)$	1	t
Proposed countermeasure	1	$t-1$	0	0

themselves).

We propose a different improvement: the initiator signs the value $Auth$ so that the attacker cannot forge it. Figure 4 describes the countermeasure in detail. Unlike the first solution, the second discards the authentication flaw (U_a cannot impersonate U_1 anymore) and decreases the overall cost (in addition to the original protocol, the initiator generates one signature and the rest of participants verify it).

Table I gives an overhead comparison between the trivial countermeasure and the proposed improvement, plotting the additional costs introduced by the two solutions.

VI. CONCLUSION

Hsu et al. recently introduced a GKT protocol based on secret sharing, which they claimed to be secure and interesting for practical applications [9]. However, the authors only provide informal and incomplete security arguments to support their affirmation. We prove that they are wrong by revealing an authentication flaw, which permits an attacker to break the key consistency of the protocol: the adversary makes different users compute distinct keys, which leads to the futility of the protocol.

First, we mention the trivial solution of adding a Key Confirmation phase; however, this does not eliminate the authentication flaw and significantly decreases the efficiency. Finally, we propose a simple countermeasure that stand against the proposed attack, introduces only a small overload and maintains the same number of rounds as the original protocol.

We do not claim that our improvement leads to a secure version of the protocol, but only affirm that it stands against the reveal attack. A detailed security analysis could be considered for future work.

REFERENCES

[1] J. Pieprzyk and C.-H. Li, "Multiparty key agreement protocols," *IEE Proceedings-Computers and Digital Techniques*, vol. 147, no. 4, pp. 229–236, 2000.

[2] J. Nam, M. Kim, J. Paik, W. Jeon, B. Lee, and D. Won, "Cryptanalysis of a group key transfer protocol based on secret sharing," in *FGIT*, 2011, pp. 309–315.

[3] L. Harn and C. Lin, "Authenticated group key transfer protocol based on secret sharing," *IEEE Transactions on Computers*, vol. 59, no. 6, pp. 842–846, 2010.

[4] R. F. Olimid, "On the security of an authenticated group key transfer protocol based on secret sharing," in *ICT-EurAsia*, 2013, pp. 399–408.

[5] M. Kim, N. Park, and D. Won, "Cryptanalysis of an authenticated group key transfer protocol based on secret sharing," in *GPC*, 2013, pp. 761–766.

[6] Y. Sun, Q. Wen, H. Sun, W. Li, Z. Jin, and H. Zhang, "An authenticated group key transfer protocol based on secret sharing," *Procedia Engineering*, vol. 29, no. 0, pp. 403 – 408, 2012.

[7] R. F. Olimid, "Cryptanalysis of a password-based group key exchange protocol using secret sharing," *Applied Mathematics and Information Sciences*, vol. 7, no. 4, pp. 1585–1590, 2013.

[8] W. Yuan, L. Hu, H. Li, and J. Chu, "Cryptanalysis of a password-based group key exchange protocol using secret sharing," *Applied Mathematics and Information Sciences*, vol. 7, no. 1, pp. 145–150, 2013.

[9] C. Hsu, B. Zeng, Q. Cheng, and G. Cui, "A novel group key transfer protocol," Cryptology ePrint Archive, Report 2012/043, 2012, http://eprint.iacr.org/.

[10] C. Boyd, "On key agreement and conference key agreement," in *ACISP*, 1997, pp. 294–302.