On the Security of an Authenticated Group Key Transfer Protocol Based on Secret Sharing

Ruxandra F. Olimid

Department of Computer Science, University of Bucharest, Romania
ruxandra.olimid@fmi.unibuc.ro

Abstract. Group key transfer protocols allow multiple parties to share a common secret key. They rely on a mutually trusted key generation center (KGC) that selects the key and securely distributes it to the authorized participants. Recently, Sun et al. proposed an authenticated group key transfer protocol based on secret sharing that they claim to be secure. We show that this is false: the protocol is susceptible to insider attacks and violates known key security. Finally, we propose a countermeasure that maintains the benefits of the original protocol.

Keywords: group key transfer, secret sharing, attack, cryptanalysis.

1 Introduction

Confidentiality represents one of the main goals of secure communication. It assures that the data are only accessible to authorized parties and it is achieved by encryption. In case of symmetric cryptography, the plaintext is encrypted using a secret key that the sender shares with the qualified receiver(s). Under the assumption that the system is secure, an entity that does not own the private key is unable to decrypt and thus the data remain hidden to unauthorized parties.

The necessity of a (session) key establishment phase before the encrypted communication starts is immediate: it allows the authorized parties to share a common secret key that will be used for encryption.

Key establishment protocols divide into key transfer protocols - a mutually trusted key generation center (KGC) selects a key and securely distributes it to the authorized parties - and key agreement protocols - all qualified parties are involved in the establishment of the secret key. The first key transfer protocol was published by Needham and Schroeder in 1978 [11], two years after Diffie and Hellman had invented the public key cryptography and the notion of key agreement [4].

The previous mentioned protocols restrict to the case of two users. As a natural evolution, *group (or conference) key establishment protocols* appeared a few years latter. Ingemarsson et al. introduced the first key transfer protocol that permits to establish a private key between multiple parties [8]. Their protocol generalizes the Diffie-Hellman key exchange.

A general construction for a key transfer protocol when KGC shares a long-term secret with each participant is straightforward: KGC generates a fresh key and sends its encryption (under the long-term secret) to each authorized party. The qualified users decrypt their corresponding ciphertext and find the session secret key, while the unqualified users cannot decrypt and disclose the session key. KGC performs n encryptions and sends n messages, where n is the number of participants. Therefore, the method becomes inefficient for large groups.

Secret sharing schemes are used in group key transfer protocols to avoid such disadvantages. A secret sharing scheme splits a secret into multiple shares so that only authorized sets of shares may reconstruct the secret. Blakley [1] and Shamir [14] independently introduce secret sharing schemes as key management systems. The particular case when all shares are required for reconstruction is called *all-or-nothing secret sharing scheme*; the particular case when at least k out of n shares are required for reconstruction is called *(k,n)-threshold secret sharing scheme*.

Various group key establishment protocols based on secret sharing schemes exist in the literature. Blom proposed an efficient key transfer protocol in which every two users share a common private key that remains hidden when less than k users cooperate [2]. Blundo et al. generalized Blom's protocol by allowing any t users to share a private key, while it remains secure for a coalition of up to k users [3]. Fiat and Naor improved the construction even more by permitting any subset of users to share a common key in the same conditions [5]. Pieprzyk and Li gave a couple of group key agreement protocols based on Shamir's secret scheme [9, 12]. Recently, Harn and Lin also used Shamir's scheme to construct a key transfer protocol [6], which was proved to be insecure and further adjusted [10]. Some other examples from literature include Sáez's protocol [13] (based on a family of vector space secret sharing schemes), Hsu et al.'s protocol [7] (based on linear secret sharing schemes) and Sun et al's group key transfer protocol [15], which we will refer for the rest of this paper.

Sun et al. claim that their construction is secure and provides several advantages: each participant stores a single long-term secret for multiple sessions, computes the session key by a simple operation and the protocol works within dynamic groups (i.e. members may leave or join the group). We demonstrate that they are wrong. First, we show that the protocol is susceptible to insider attacks: any qualified group member may recover a session key that he is unauthorized to know. Second, we prove that the protocol violates known key security: any attacker who gains access to one session key may recover any other key. We propose an improved version of Sun et al's group key transfer protocol that stands against both attacks and achieves the benefits claimed in the original work.

The paper is organized as follows. The next section contains the preliminaries. Section 3 describes Sun et al.'s authenticated group key transfer protocol. Section 4 introduces the proposed attacks. In Section 5 we analyze possible countermeasures. Section 6 concludes.

2 Preliminaries

2.1 Security Goals

Group key transfer protocols permit multiple users to share a common private key by using pre-established secure communication channels with a trusted KGC, which is responsible to generate and distribute the key. Each user registers to KGC for subscribing to the key distribution service and receives a long-term secret, which he will later use to recover the session keys.

We will briefly describe next the main security goals that a group key transfer protocol must achieve: *key freshness*, *key confidentiality*, *key authentication*, *entity authentication*, *known key security* and *forward secrecy.*

Key freshness ensures the parties that KGC generates a random key that has not been used before. Unlike key agreement protocols, the users are not involved in the key generation phase, so the trust assumption is mandatory.

Key confidentiality means that a session key is available to authorized parties only. Adversaries are categorized into two types: insiders - that are qualified to recover the session key - and outsiders - that are unqualified to determine the session key. A protocol is susceptible to insider attacks if an insider is able to compute secret keys for sessions he is unauthorized for. Similarly, it is vulnerable to outsider attacks if an outsider is capable to reveal any session key.

Key authentication assures the group members that the key is distributed by the trusted KGC and not by an attacker. It may also stand against a replay attack: no adversary can use a previous message originated from KGC to impose an already compromised session key.

Entity authentication confirms the identity of the users involved in the protocol, so that an attacker cannot impersonate a qualified principal to the KGC.

Known key security imposes that a compromised session key has no impact on the confidentiality of other session keys: even if an adversary somehow manages to obtain a session key, all the other past and future session keys remain hidden.

Forward secrecy guarantees that even if a long-term secret is compromised, this has no impact on the secrecy of the previous session keys.

2.2 Secret Sharing

A secret sharing scheme is a method to split a secret into multiple shares, which are then securely distributed to the participants. The secret can be recovered only when the members of an authorized subset of participants combine their shares together. The set of all authorized subsets is called the *access structure.* The access structure of a (k, n) *threshold secret sharing scheme* consists of all sets whose cardinality is at least k. The access structure of an *all-or-nothing secret sharing scheme* contains only one element: the set of all participants.

Generally, a secret sharing scheme has 3 phases: *sharing* (a dealer splits the secret into multiple parts, called *shares*), *distribution* (the dealer securely transmits the shares to the parties) and *reconstruction* (an authorized group of parties put their shares together to recover the secret).

Group key establishment protocols use secret sharing schemes due to the benefits they introduce: decrease computational and transmission costs, represent a convenient way to differentiate between principals and their power within the group, permits delegation of shares, accepts cheating detection, permits the sizing of the group, etc. [12]. For more information, the reader may refer to [12].

2.3 Discrete Logarithm Assumption

Let G be a cyclic multiplicative group of order p with $g \in G$ a generator.

The Discrete Logarithm Assumption holds in G if given g^a, any probabilistic polynomial-time adversary $\mathcal{A}$ has a negligible probability in computing a:

$$Adv_{\mathcal{A}} = Pr[\mathcal{A}(p, g, g^a) = a] \leq negl(k) \tag{1}$$

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

3 Sun et al.'s Group Key Transfer Protocol

Let n be the size of the group of participants, $\mathcal{U} = \{U_1, U_2, ..., U_n\}$ the set of users, G a multiplicative cyclic group of prime order p with $g \in G$ as a generator, H a secure hash function.

Sun et al. base their protocol on a *derivative secret sharing*, which we describe next and briefly analyze in Section 5:

Derivative Secret Sharing [15].

Phase 1: Secret Sharing

The dealer splits the secret $S \in G$ into two parts n times:

$$S = s_1 + s_1' = s_2 + s_2' = ... = s_n + s_n' \tag{2}$$

Phase 2: Distribution

The dealer sends the share $s_i' \in G$ to $U_i \in \mathcal{U}$ via a secure channel.

Phase 3: Reconstruction

1. The dealer broadcasts the shares $s_1, s_2, ..., s_n$ at once, when the users want to recover the secret S.

2. Any user $U_i \in \mathcal{U}$ reconstructs the secret as:

$$S = s_i' + s_i \tag{3}$$

Next, we review Sun et. al's group key transfer protocol, which we prove vulnerable in Section 4:

Sun et al.'s Group Key Transfer Protocol [15].

Phase 1: User Registration

During registration, KGC shares a long-term secret $s'_i \in G$ with each user $U_i \in \mathcal{U}$.

Phase 2: Group Key Generation and Distribution

1. A user, called the initiator, sends a group key distribution request that contains the identity of the qualified participants for the current session $\{U_1, U_2, ..., U_t\}$ to KGC.[1]
2. KGC broadcasts the received list as a response.
3. Each member $U_i, i = 1, \ldots, t$ that identifies himself in the list sends a random challenge $r_i \in \mathbb{Z}_p^*$ to KGC.
4. KGC randomly selects $S \in G$ and invokes the derivative secret sharing scheme to split S into two parts t times such that $S = s_1 + s'_1 = s_2 + s'_2 = ... = s_t + s'_t$. He computes the session private key as $K = g^S$, t messages $M_i = (g^{s_i+r_i}, U_i, H(U_i, g^{s_i+r_i}, s'_i, r_i)), i = 1, \ldots, t$ and $Auth = H(K, g^{s_1+r_1}, ..., g^{s_t+r_t}, U_1, ..., U_t, r_1, ..., r_t)$. At last, KGC broadcasts $(M_1, ..., M_t, Auth)$ as a single message.
5. After receiving M_i and $Auth$, each user U_i, $i = 1, \ldots, t$ computes $h = H(U_i, g^{s_i+r_i}, s'_i, r_i)$ using $g^{s_i+r_i}$ from M_i, s'_i the long-term secret and r_i as chosen in step 3. If h differs from the corresponding value in M_i, the user aborts; otherwise, he computes $K' = g^{s'_i} \cdot g^{s_i+r_i}/g^{r_i}$ and checks if $Auth = H(K', g^{s_1+r_1}, ..., g^{s_t+r_t}, U_1, ..., U_t, r_1, ..., r_t)$. If not, he aborts; otherwise, he consider K' to be the session key originated from KGC and returns a value $h_i = H(s'_i, K', U_1, ..., U_t, r_1, ...r_t)$ to KGC.
6. KGC computes $h'_i = H(s'_i, K, U_1, ..., U_t, r_1, ...r_t)$ using his own knowledge on s'_i and checks if h'_i equals h_i, certifying that all users possess the same key.

The authors claim that their construction is secure under the discrete logarithm assumption and has multiple advantages: each participant needs to store only one secret share for multiple sessions (the long-term secret $s'_i, i = 1, \ldots, n$), the dynamic of the group preserves the validity of the shares (if a user leaves or joins the group there is no need to update the long-term secrets), each authorized user recovers the session key by a simple computation. Unlike their claim, in the next section we prove that the protocol is insecure.

4 The Proposed Attacks

We show that Sun et al.'s protocol is insecure against insider attacks and violates known key security.

4.1 Insider Attack

Let $U_a \in \mathcal{U}$ be an authorized user for a session (k_1), s'_a his long-term secret, $\mathcal{U}_{(k_1)} \subseteq \mathcal{U}$ the qualified set of participants of the session, $(g^{s_{i(k_1)}+r_{i(k_1)}})_{U_i \in \mathcal{U}_{(k_1)}}$

[1] Without loss of generality, any subset of $\mathcal{U}$ can be expressed as $\{U_1, U_2, ..., U_t\}$ by reordering.

the values that were broadcast as part of $(M_i)_{U_i \in \mathcal{U}_{(k_1)}}$ in step 4 and $K_{(k_1)} = g^{S_{(k_1)}}$ the session key.

The participant U_a is qualified to determine (k_1) session key as:

$$K_{(k_1)} = \frac{g^{s'_a} \cdot g^{s_{a(k_1)} + r_{a(k_1)}}}{g^{r_{a(k_1)}}} \tag{4}$$

Since $g^{s_{i(k_1)} + r_{i(k_1)}}$ and $r_{i_{(k_1)}}$ are public, he is able to compute $g^{s'_i}$, for all $U_i \in \mathcal{U}_{(k_1)}$:

$$g^{s'_i} = \frac{K_{(k_1)} \cdot g^{r_{i(k_1)}}}{g^{s_{i(k_1)} + r_{i(k_1)}}} \tag{5}$$

Suppose that U_a is unauthorized to recover (k_2) session key, $(k_2) \neq (k_1)$. However, he can eavesdrop the exchanged messages. Therefore, he is capable to determine

$$g^{s_{j(k_2)}} = \frac{g^{s_{j(k_2)} + r_{j(k_2)}}}{g^{r_{j(k_2)}}} \tag{6}$$

for all $U_j \in \mathcal{U}_{(k_2)}$, where $\mathcal{U}_{(k_2)} \subseteq \mathcal{U}$ is the qualified set of parties of the session (k_2).

We assume that there exists a participant $U_b \in \mathcal{U}_{(k_1)} \bigcap \mathcal{U}_{(k_2)}$ that is qualified for both sessions (k_1) and (k_2). The inside attacker U_a can find the key $K_{(k_2)}$ of the session (k_2) as:

$$K_{(k_2)} = g^{s'_b} \cdot g^{s_{b(k_2)}} = g^{s'_b + s_{b(k_2)}} \tag{7}$$

In conclusion, an insider can determine any session key under the assumption that at least one mutual authorized participant for both sessions exists, which is very likely to happen.

The attack also stands if there is no common qualified user for the two sessions, but there exists a third one (k_3) that has a mutual authorized party with each of the former sessions. The extension is straightforward: let $U_{1,3}$, $U_{2,3}$ be the common qualified parties for sessions (k_1) and (k_3), respectively (k_2) and (k_3). U_a computes the key $K_{(k_3)}$ as in the proposed attack due to the common authorized participant $U_{1,3}$. Once he obtains the key $K_{(k_3)}$, he mounts the attack again for sessions (k_3) and (k_2) based on the common party $U_{2,3}$ and gets $K_{(k_2)}$.

The attack extends in chain: the insider U_a reveals a session key $K_{(k_x)}$ if he is able to build a chain of sessions $(k_1)...(k_x)$, where (k_i) and (k_{i+1}) have at least one common qualified member $U_{i,i+1}$, $i = 1, \ldots, x-1$ and U_a is authorized to recover the key $K_{(k_1)}$.

4.2 Known Key Attack

Suppose an attacker (insider or outsider) owns a session key $K_{(k_1)}$. We also assume that he had previously eavesdropped values $r_{i_{(k_1)}}$ in step 3 and $g^{s_{i(k_1)} + r_{i(k_1)}}$ in step 4 of session (k_1) so that for all $U_i \in \mathcal{U}_{(k_1)}$ he determines:

$$g^{s_{i_{(k_1)}}} = \frac{g^{s_{i_{(k_1)}} + r_{i_{(k_1)}}}}{g^{r_{i_{(k_1)}}}} \tag{8}$$

Because the session key $K_{(k_1)}$ is exposed, he can also compute $g^{s'_i}$, for all $U_i \in \mathcal{U}_{(k_1)}$:

$$g^{s'_i} = \frac{K_{(k_1)}}{g^{s_{i_{(k_1)}}}} \tag{9}$$

Let (k_2) be any previous or future session that has at least one common qualified participant U_b with (k_1), i.e. $U_b \in \mathcal{U}_{(k_1)} \bigcap \mathcal{U}_{(k_2)}$. As before, the attacker eavesdrops $r_{b_{(k_2)}}$ and $g^{s_{b_{(k_2)}} + r_{b_{(k_2)}}}$ and computes

$$g^{s_{b_{(k_2)}}} = \frac{g^{s_{b_{(k_2)}} + r_{b_{(k_2)}}}}{g^{r_{b_{(k_2)}}}} \tag{10}$$

The attacker can now recover the key $K_{(k_2)}$:

$$K_{(k_2)} = g^{s'_b} \cdot g^{s_{b_{(k_2)}}} = g^{s'_b + s_{b_{(k_2)}}} \tag{11}$$

The attack may also be mount in chain, similar to the insider attack. We omit the details in order to avoid repetition.

The first attack permits an insider to reveal any session key he was unauthorized for, while the second permits an attacker (insider or outsider) to disclose any session key under the assumption that a single one has been compromised.

In both cases, the attacker computes the session key as the product of two values: g^{s_b} (disclosed only by eavesdropping) and $g^{s'_b}$ (revealed by eavesdropping when a session key is known). We remark that the attacker is unable to determine the long-term secret s'_b if the discrete logarithm assumption holds, but we emphasize this does not imply that the protocol is secure, since the adversary's main objective is to find the session key that can be disclosed without knowing s'_b.

5 Countermeasures

Sun et al.'s group key agreement protocol fails because values $g^{s'_i}$, $i = 1, \ldots, n$ are maintained during multiple sessions. We highlight that the derivative secret sharing scheme suffers from a similar limitation caused by the usage of the long-term secrets s'_i, $i = 1, \ldots, n$ during multiple sessions: any entity that discloses a secret S determines the values $s'_i = S - s_i$ by eavesdropping s_i, $i = 1, \ldots, n$ and uses them to reveal other shared secrets.

A trivial modification prevents the proposed attacks: KGC replaces the values s'_i at the beginning of each session. This way, even if the attacker determines $g^{s'_i}$ in one round, the value becomes unusable. However, this introduces a major drawback: KGC must share a secure channel with each user for any round. If this were the case, KGC could have just sent the secret session key via the secure channel and no other protocol would have been necessary. In conclusion, the group key transfer protocol would have become useless.

Another similar solution exists: during the user registration phase an ordered set of secrets is shared between KGC and each user. For each session, the corresponding secret in the set is used in the derivative secret sharing. Although this solution has the benefit that it only requires the existence of secure channels at registration, it introduces other disadvantages: each user must store a considerable larger quantity of secret information, the protocol can run for at most a number of times equal to the set cardinality, KGC must broadcast the round number so that participants remain synchronized.

We propose next a countermeasure inspired by the work of Pieprzyk and Li [12]. The main idea consist of using a different public value $\alpha \in G$ to compute the session key $K = \alpha^S$ for each round.

The Improved Version of the Group Key Transfer Protocol .

Phase 1: User Registration

During registration, KGC shares a long-term secret $s'_i \in G$ with each user $U_i \in \mathcal{U}$.

Phase 2: Group Key Generation and Distribution

1. A user, called the initiator, sends a group key distribution request that contains the identity of the qualified participants for the current session $\{U_1, U_2, ..., U_t\}$ to KGC.
2. KGC broadcasts the received list as a response.
3. Each member $U_i, i = 1, \ldots, t$ that identifies himself in the list sends a random challenge $r_i \in \mathbb{Z}_p^*$ to KGC.
4. KGC randomly selects $S \in G$ and invokes the derivative secret sharing scheme to split S into two parts t times such that $S = s_1 + s'_1 = s_2 + s'_2 = ... = s_t + s'_t$. He chooses $\alpha \in G$ at random, computes the session private key as $K = \alpha^S$, t messages $M_i = (\alpha^{s_i+r_i}, U_i, H(U_i, \alpha^{s_i+r_i}, s'_i, r_i, \alpha)), i = 1, \ldots, t$ and $Auth = H(K, \alpha^{s_1+r_1}, ..., \alpha^{s_t+r_t}, U_1, ..., U_t, r_1, ..., r_t, \alpha)$. At last, KGC broadcasts $(M_1, ..., M_t, Auth, \alpha)$ as a single message.
5. After receiving M_i, $Auth$ and α, each user U_i, $i = 1, \ldots, t$ computes $h = H(U_i, \alpha^{s_i+r_i}, s'_i, r_i, \alpha)$ using $\alpha^{s_i+r_i}$ from M_i, s'_i the long-term secret and r_i as chosen in step 3. If h differs from the corresponding value in M_i, the user aborts; otherwise, he computes $K' = \alpha^{s'_i} \cdot \alpha^{s_i+r_i}/\alpha^{r_i}$ and checks if $Auth = H(K', \alpha^{s_1+r_1}, ..., \alpha^{s_t+r_t}, U_1, ..., U_t, r_1, ..., r_t, \alpha)$. If not, he aborts; otherwise, he consider K' to be the session key originated from KGC and returns a value $h_i = H(s'_i, K', U_1, ..., U_t, r_1, ...r_t, \alpha)$ to KGC.
6. KGC computes $h'_i = H(s'_i, K, U_1, ..., U_t, r_1, ...r_t, \alpha)$ using his own knowledge on s'_i and checks if h'_i equals h_i, certifying that all users possess the same key.

The countermeasure eliminates both attacks. Under the discrete logarithm assumption, a value ${\alpha_{(k_1)}}^{s'_i}$ from a session (k_1) can no longer be used to compute a session key $K_{(k_2)} = \alpha_{(k_2)}^{s'_i+s_{i_{(k_2)}}}$ with $(k_2) \neq (k_1)$. The values α are authenticated to originate from KGC so that an attacker cannot impersonate the KGC and use a suitable value (for example $\alpha_{(k_2)} = \alpha^a_{(k_1)}$ for a known a).

We remark that the modified version of the protocol maintains all the benefits of the original construction and preserves the computational cost, while the transmission cost increases negligible. However, we admit that it conserves a weakness of the original protocol: cannot achieve forward secrecy. Any attacker that obtains a long-term secret becomes able to compute previous keys of sessions he had eavesdropped before. The limitation is introduced by construction because the long-term secret is directly used to compute the session key.

6 Conclusions

Sun et al. recently introduced an authenticated group key transfer protocol based on secret sharing [15], which they claimed to be efficient and secure. We proved that they are wrong: the protocol is vulnerable to insider attacks and violates known key security. We improved their protocol by performing a slight modification that eliminates the proposed attacks and maintains the benefits of the original work.

Acknowledgments. This paper is supported by the Sectorial Operational Program Human Resources Development (SOP HRD), financed from the European Social Fund and by the Romanian Government under the contract number SOP HDR/107/1.5/S/82514.

References

1. Blakley, G.: Safeguarding cryptographic keys. In: Proceedings of the 1979 AFIPS National Computer Conference. pp. 313–317. AFIPS Press (1979)
2. Blom, R.: An optimal class of symmetric key generation systems. In: Proc. of the EUROCRYPT 84 workshop on Advances in cryptology: theory and application of cryptographic techniques. pp. 335–338. Springer-Verlag, New York, USA (1985)
3. Blundo, C., Santis, A.D., Herzberg, A., Kutten, S., Vaccaro, U., Yung, M.: Perfectly-secure key distribution for dynamic conferences. In: CRYPTO. pp. 471–486 (1992)
4. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
5. Fiat, A., Naor, M.: Broadcast encryption. In: CRYPTO. pp. 480–491 (1993)
6. Harn, L., Lin, C.: Authenticated group key transfer protocol based on secret sharing. IEEE Trans. Comput. 59(6), 842–846 (2010)
7. Hsu, C., Zeng, B., Cheng, Q., Cui, G.: A novel group key transfer protocol. Cryptology ePrint Archive, Report 2012/043 (2012)
8. Ingemarsson, I., Tang, D.T., Wong, C.K.: A conference key distribution system. IEEE Transactions on Information Theory 28(5), 714–719 (1982)
9. Li, C.H., Pieprzyk, J.: Conference key agreement from secret sharing. In: Proceedings of the 4th Australasian Conference on Information Security and Privacy. pp. 64–76. ACISP '99, Springer-Verlag, London, UK (1999)

10. Nam, J., Kim, M., Paik, J., Jeon, W., Lee, B., Won, D.: Cryptanalysis of a group key transfer protocol based on secret sharing. In: Proceedings of the Third international conference on Future Generation Information Technology. pp. 309–315. FGIT'11, Springer-Verlag, Berlin, Heidelberg (2011)
11. Needham, R.M., Schroeder, M.D.: Using encryption for authentication in large networks of computers. Commun. ACM 21(12), 993–999 (1978)
12. Pieprzyk, J., Li, C.H.: Multiparty key agreement protocols. In: IEEE Proceedings - Computers and Digital Techniques. pp. 229–236 (2000)
13. Sáez, G.: Generation of key predistribution schemes using secret sharing schemes. Discrete Applied Mathematics 128(1), 239–249 (2003)
14. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)
15. Sun, Y., Wen, Q., Sun, H., Li, W., Jin, Z., Zhang, H.: An authenticated group key transfer protocol based on secret sharing. Procedia Engineering 29(0), 403 – 408 (2012)