Attacks on Bresson-Chevassut-Essiari-Pointcheval's Group Key Agreement Scheme for Low-Power Mobile Devices ⋆

Junghyun Nam, Seungjoo Kim, and Dongho Won

School of Information and Communication Engineering,
Sungkyunkwan University, 300 Chunchun-dong, Jangan-gu, Suwon-si,
Gyeonggi-do 440-746, Korea
{jhnam, skim}@ece.skku.ac.kr, dhwon@dosan.skku.ac.kr

Abstract. In this paper, we show that Bresson-Chevassut-Essiari-Pointcheval's group key agreement scheme does not meet the main security properties: implicit key authentication, forward secrecy, and known key security. Also, we propose an improved version which fixes the security flaws found in the scheme.

Keywords: Group key agreement, key authentication, forward secrecy, known key security.

1 Introduction

When one considers the broad range of wirelessly connected mobile devices used today, it is clear that integrating such network-enabled devices into secure communication systems is of essential importance. Recently, Bresson *et al.* [1] proposed a very efficient group key agreement scheme well suited for a wireless network environment. The scheme consists of three protocols: the setup protocol $\mathsf{GKE.Setup}$, the remove protocol $\mathsf{GKE.Remove}$, and the join protocol $\mathsf{GKE.Join}$. The main $\mathsf{GKE.Setup}$ protocol allows a set of mobile users (also called *clients*) and a wireless gateway (also called *server*) to agree on a session key. The other protocols of the scheme aim to efficiently handle dynamic membership changes of clients in one wireless domain. In this paper, we demonstrate the insecurity of the $\mathsf{GKE.Setup}$ protocol by presenting some active attacks against implicit key authentication, forward secrecy, and known key security. Furthermore, we modify the scheme so that all these security goals can be attained.

2 Brief Review of Bresson *et al.*'s Scheme

Arithmetic is in a finite cyclic group $\mathbb{G} = \langle g \rangle$ of ℓ-bit prime order q, where ℓ is a security parameter. Both g and q are publicly known. There are three hash functions $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\ell$, $\mathcal{H}_0 : \{0,1\}^* \rightarrow \{0,1\}^{\ell_0}$, where ℓ_0 needs not be equal to ℓ, and $\mathcal{H}_1 : \{0,1\}^{\ell_1} \times \mathbb{G} \rightarrow \{0,1\}^{\ell_0}$, where ℓ_1 is the maximal bit-length of a counter c used in the scheme.

2.1 Long-Term Keys

Let $\mathcal{C}$ denote the set of all potential clients, and let S denote the server. Before the protocol is run for the first time, an initialization phase occurs during which:

⋆ A revised version of this paper will appear in IEEE Communications Letters, under the title "A Weakness in the Bresson-Chevassut-Essiari-Pointcheval's Group Key Agreement Scheme for Low-Power Mobile Devices". This work was supported by Korea Research Foundation Grant (KRF-2004-003-D00392).

1. Each client $U_i \in \mathcal{C}$ generates a pair of signing private/public keys (SK_i, PK_i) by running the key generation algorithm of a signature scheme.
2. The server S sets its private/public keys to be $(SK_S, PK_S) = (x, y)$, where $x \in_R \mathbb{Z}_q^*$ and $y = g^x$.

2.2 The GKE.Setup Protocol

Let $\mathcal{G}_c \subseteq \mathcal{C}$ be a set of *clients* who wish to share a session key with the server S. Let $\mathcal{I}_c$ be the set of indices of the clients in $\mathcal{G}_c$. The protocol executes in two rounds. In the first round, S collects contributions from individual clients and then, in the second round, it sends the group keying material to the clients. The actual protocol proceeds as follows:

1. Each client $U_i \in \mathcal{G}_c$ chooses a random $x_i \in \mathbb{Z}_q^*$ and computes
$$y_i = g^{x_i}$$
$$\alpha_i = y^{x_i}.$$
Client U_i then signs y_i to obtain signature σ_i and sends (y_i, σ_i) to the server S.
2. For all $i \in \mathcal{I}_c$, the server S verifies the signature σ_i and computes
$$\alpha_i = y_i^x.$$
S then initializes the counter c to 0, and computes the shared secret value
$$K = \mathcal{H}_0(\mathsf{c} \| \{\alpha_i\}_{i \in \mathcal{I}_c})$$
and for all $i \in \mathcal{I}_c$,
$$K_i = K \oplus \mathcal{H}_1(\mathsf{c} \| \alpha_i).$$
The server S sends to each client U_i the values (c, K_i).
3. Upon receiving c and K_i, client U_i recovers the shared secret value K as
$$K = K_i \oplus \mathcal{H}_1(\mathsf{c} \| \alpha_i).$$
Finally, both the server and the clients compute the same session key as:
$$sk = \mathcal{H}(K \| \mathcal{G}_c \| S).$$

3 Security Analysis

3.1 Implicit Key Authentication

The fundamental security requirement for a group key agreement protocol to achieve is the property referred to as implicit key authentication. In protocols providing implicit key authentication, each participant is assured that no one other than the intended parties can learn the value of the session key.

To show that the GKE.Setup protocol does not provide implicit key authentication, we consider two runs of the protocol which are executed in an either concurrent or non-concurrent manner. We denote by $\mathcal{G}_c$ and $\mathcal{G}_c'$ the sets of clients with respect to the first and second runs, respectively. Assume that the adversary A participates as a client in the first run of the protocol (i.e., $A \in \mathcal{G}_c$), but is intended to be excluded from the second run (i.e., $A \notin \mathcal{G}'_c \cup \{S\}$). Also assume that two client sets $\mathcal{G}_c$ and $\mathcal{G}'_c$ are non-disjoint. The goal of adversary A is to share the same key with the participants of the second run. To do so, the adversary A gathers some information during the first run and uses that information to impersonate some client in the second run. The detailed attack scenario is as follows:

1. In the first run of the protocol, the adversary A computes the shared secret value K participating as a client. A then obtains $\mathcal{H}_1(\mathsf{c}\|\alpha_i)$ for all $i \in \mathcal{I}_c$ by computing
$$\mathcal{H}_1(\mathsf{c}\|\alpha_i) = K \oplus K \oplus \mathcal{H}_1(\mathsf{c}\|\alpha_i) = K \oplus K_i,$$
which can be done without knowing α_i. The adversary A records $\mathcal{H}_1(\mathsf{c}\|\alpha_i)$ and (y_i, σ_i) for all $i \in \mathcal{I}_c$.
2. In the first round of the second run, the adversary A (pretending to be U_j for some $U_j \in \mathcal{G}_c \cap \mathcal{G}'_c$) replaces the message (y'_j, σ'_j) sent by U_j with (y_j, σ_j) stored in the previous step of this scenario. Because the server S thinks that (y_j, σ_j) is from U_j, it will compute the shared secret value K' as per protocol specification and will send to client U_j the values $\mathsf{c} = 0$ and
$$K'_j = K' \oplus \mathcal{H}_1(\mathsf{c}\|\alpha'_j),$$
with α'_j computed as
$$\alpha'_j = y_j^x.$$
3. In the second round of the second run, the adversary A records K'_j which is transmitted through an open channel. Now, from the values K'_j and $\mathcal{H}_1(\mathsf{c}\|\alpha_j)$ (obtained in the first step of this scenario), the adversary A can recover the shared secret value K' as follows:
$$K' = K' \oplus \mathcal{H}_1(\mathsf{c}\|\alpha'_j) \oplus \mathcal{H}_1(\mathsf{c}\|\alpha_j) = K'_j \oplus \mathcal{H}_1(\mathsf{c}\|\alpha_j).$$
This equation holds, since $\alpha'_j = \alpha_j$ and thus
$$\mathcal{H}_1(\mathsf{c}\|\alpha'_j) = \mathcal{H}_1(\mathsf{c}\|\alpha_j).$$
Finally, the adversary A can share the same session key $sk' = \mathcal{H}(K'\|\mathcal{G}'_c\|S)$ with all the participants of the second run except U_j.

Consequently, implicit key authentication is not guaranteed in the protocol, as soon as the adversary participates as a client in a protocol execution and is intended to be excluded from another protocol execution with a non-disjoint set of clients.

3.2 Forward Secrecy

The perfect forward secrecy property says that earlier session keys are protected against loss of some underlying information at the present time. As noted by the authors themselves, the $\mathsf{GKE.Setup}$ protocol does not provide perfect forward secrecy; as soon as the long-term private key x of the server is leaked, all the past session keys can be recovered since every α_i can easily be computed from y_i and x.

However, it is claimed by the authors that the protocol achieves partial forward secrecy; disclosure of the private signing keys of clients does not reveal anything about previous session keys. In support of this claim, they argue that the long-term keys of the clients are used for implicit authentication only, and not for hiding the session key. But, this claim is flawed. The attack below shows that if some client's signing key is ever revealed, then any previous session key can be computed by an active adversary. As a simple scenario, we consider two runs of the protocol with the first one being completed before the second one begins. Similarly as before, we denote the client groups with respect to the first and second runs of the protocol by $\mathcal{G}_c$ and $\mathcal{G}'_c$ respectively. Assume that the adversary A wants to recover the session key established in the first run of the protocol in which she has not participated. The attack is launched as follows:

1. In the first run of the protocol, the adversary A eavesdrops on the session recording the transmitted messages (y_i, σ_i) and
$$K_i = K \oplus \mathcal{H}_1(\mathsf{c}\|\alpha_i)$$
for some $i \in \mathcal{I}_c$.
2. Now, the adversary A participates as a client in the second run of the protocol. Because we consider forward secrecy, we will assume that the private signing key SK_j of some other client $U_j \in \mathcal{G}'_c$ is exposed to A.
3. In the first round of the second run, the adversary A proceeds much like a normal client by sending the message (y'_A, σ'_A) to the server. In the same time period, the adversary A (pretending to be the client U_j) replaces the message (y'_j, σ'_j) sent by U_j with
$$(y_i, \sigma''_j),$$
where σ''_j is the signature of y_i (stored in the first step) under the private key SK_j. Note that the adversary A can sign any message of its choice on behalf of U_j.
4. Because the verification of signature σ''_j will succeed, the server S will compute α'_j as
$$\alpha'_j = y_i^x$$
and the shared secret value K' as per protocol specification. Then S will send to client U_j the values $\mathsf{c} = 0$ and
$$K'_j = K' \oplus \mathcal{H}_1(\mathsf{c}\|\alpha'_j).$$
5. Now, in the second round of the second run, the adversary A should be able to compute the shared secret value K' since it participates as a group member. From K' and K'_j, the adversary A can recover $\mathcal{H}_1(\mathsf{c}\|\alpha'_j)$ as follows:
$$\mathcal{H}_1(\mathsf{c}\|\alpha'_j) = K' \oplus K' \oplus \mathcal{H}_1(\mathsf{c}\|\alpha'_j) = K' \oplus K'_j.$$
6. With this information $\mathcal{H}_1(\mathsf{c}\|\alpha'_j)$, the adversary A can recover the shared secret value K of the first run of the protocol as follows:
$$K = K \oplus \mathcal{H}_1(\mathsf{c}\|\alpha_i) \oplus \mathcal{H}_1(\mathsf{c}\|\alpha'_j) = K_i \oplus \mathcal{H}_1(\mathsf{c}\|\alpha'_j),$$
since $\alpha_i = \alpha'_j$ and thus $\mathcal{H}_1(\mathsf{c}\|\alpha_i) = \mathcal{H}_1(\mathsf{c}\|\alpha'_j)$. Finally, A can compute the session key $sk = \mathcal{H}(K\|\mathcal{G}_c\|S)$.

Therefore, once an underlying key is exposed, there is nothing to prevent an adversary with the key from accessing privileged information communicated in earlier sessions.

3.3 Known Key Security

A protocol is said to provide known key security if compromising of session keys does not allow a passive adversary to compromise keys of other sessions, nor an active adversary to impersonate one of the protocol parties. In this subsection, we will extend our security analysis of the protocol by presenting an active known key attack. We will assume two sessions of the protocol with the same participants. Then the following attack is possible:

1. In the first run of the protocol, the adversary A eavesdrops on the session recording the transmitted messages (y_i, σ_i) and (c, K_i) for all $i \in \mathcal{I}_c$. Since we consider known key attack, we will assume that the session key K of this run is revealed to A.
2. In the first round of the second run, the adversary A replaces the message (y_i', σ_i') sent by each U_i with (y_i, σ_i) obtained in the previous step.
3. Since all the y_i's are replayed, the server S will compute the same session key as computed in the first run of the protocol.

Therefore, at the end of this scenario, the server S will share with the adversary A the key that has been shared by all participants of the first session of the protocol.

4 Improvement

To overcome the attacks and to provide perfect forward secrecy, the scheme is modified as follows.

4.1 Long-Term Keys

During the initialization phase, each potential participant (including both the server and the clients) generates the signing private/public keys (SK, PK) by running the key generation algorithm of a signature scheme.

4.2 The Modified Setup Protocol

1. Each client $U_i \in \mathcal{G}_c$ chooses a random $x_i \in \mathbb{Z}_q^*$ and computes
$$y_i = g^{x_i}.$$
Client U_i then signs y_i to obtain signature σ_i and sends (y_i, σ_i) to the server S, who chooses a random $x_s \in \mathbb{Z}_q^*$ and computes $y_s = g^{x_s}$.
2. For all $i \in \mathcal{I}_c$, the server S verifies the signature σ_i and computes
$$\alpha_i = y_i^{x_s}.$$
Then S initializes the counter c to 0, and computes the shared secret value
$$K = \mathcal{H}_0(\mathsf{c}\|\{\alpha_i\}_{i\in\mathcal{I}_c})$$
and for all $i \in \mathcal{I}_c$,
$$K_i = K \oplus \mathcal{H}_1(\mathsf{c}\|\alpha_i).$$
The server S signs the message $\mathsf{c}\|\{K_i\}_{i\in\mathcal{I}_c}\|y_s$ to obtain signature σ_s and broadcasts $(\mathsf{c}, \{K_i\}_{i\in\mathcal{I}_c}, y_s, \sigma_s)$ to the clients.

3. Upon receiving $(\mathsf{c}, \{K_i\}_{i \in \mathcal{I}_c}, y_s, \sigma_s)$, each client U_i verifies the signature σ_s, computes

$$\alpha_i = y_s^{x_i},$$

and then recovers the shared secret value K as

$$K = K_i \oplus \mathcal{H}_1(\mathsf{c} \| \alpha_i).$$

Finally, both the server and the clients compute the same session key as:

$$sk = \mathcal{H}(K \| \mathcal{G}_c \| S).$$

The clients stores c and y_s for future use in the other protocols for member removal and addition.

The modification to the remove and join protocols is straightforward.

In the modified version, each α_i is derived as a function of the ephemeral random values x_s and x_i contributed by both the server S and the client U_i. Thus, each run of the improved protocol computes a unique α_i and $\mathcal{H}_1(\mathsf{c} \| \alpha_i)$ even if y_i is replayed by the adversary. This prevents the adversary from attacking our modified protocol against the main security properties. As for computational costs, the modified version is as efficient as the original protocol except that it additionally requires one signature generation for the server and one signature verification for a client.

5 Conclusion

We have presented a security analysis of Bresson *et al.*'s group key agreement scheme [1]. The analysis has led us to pinpoint critical security flaws in the main protocol of the scheme. We have also presented a modified version of the scheme which satisfies all the security properties: key authentication, forward secrecy, and known key security. The attacks given in this paper demonstrate again the necessity that active adversaries are to be considered carefully in designing a protocol for key agreement, especially in a group setting.

References

1. E. Bresson, O. Chevassut, A. Essiari, and D. Pointcheval, "Mutual authentication and group key agreement for low-power mobile devices," *Computer Communications*, vol. 27(17), 2004, pp. 1730–1737. A preliminary version appeared in Proc. of the 5th IFIP-TC6/IEEE International Conference on Mobile and Wireless Communications Networks (MWCN '03).