# Interleaving and Collusion Attacks on a Dynamic Group Key Agreement Scheme for Low-Power Mobile Devices*

Junghyun Nam[1], Juryon Paik[2], Jeeyeon Kim[2],
Youngsook Lee[3], Woongryul Jeon[2], and Dongho Won[2]**

[1] Department of Computer Engineering, Konkuk University, Korea
jhnam@kku.ac.kr
[2] Department of Computer Engineering, Sungkyunkwan University, Korea
wise96@ece.skku.ac.kr,jeeyeonkim@paran.com,{wrjeon,dhwon}@security.re.kr
[3] Department of Cyber Investigation Police, Howon University, Korea
ysooklee@howon.ac.kr

**Abstract.** This paper investigates the security of Bresson et al.'s dynamic group key agreement scheme for low-power mobile devices. Bresson et al.'s scheme consists of three protocols — the setup protocol, the join protocol, and the remove protocol — which are designed to minimize the cost of the rekeying operations associated with group updates. The protocols of the scheme were claimed to be provably secure against active adversaries. In this paper, we show that this claim is not necessarily true but in fact, none of the protocols are secure in the presence of an active adversary. We demonstrate this by mounting interleaving attacks on the setup and join protocols and mounting a collusion attack on the remove protocol.

**Key words:** Key agreement, dynamic group, wireless communication, interleaving attack, collusion attack.

## 1 Introduction

In this paper, we revisit the dynamic group key agreement scheme proposed by Bresson et al. [1]. This scheme is not only simple and efficient but also well suited for unbalanced networks consisting of devices with strict power consumption restrictions and wireless gateways with less stringent restrictions. The proposed scheme consists of three protocols: the setup protocol `GKE.Setup`, the remove protocol `GKE.Remove`, and the join protocol `GKE.Join`. The main `GKE.Setup` protocol allows a set of mobile devices (also called *clients*) and a wireless gateway (also called *server*) to agree on a common secret key called a session key. To meet

* This work was supported by Priority Research Centers Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0018397).
** The corresponding author: Dongho Won (e-mail: dhwon@security.re.kr).

the efficiency needs of clients, the protocol shifts most of computational burden to the gateway and provides mobile devices with the ability to perform public-key cryptography operations off-line. The other protocols of the scheme allow the server to efficiently handle dynamic membership changes of clients within a wireless domain.

Although Bresson et al.'s scheme was claimed to be secure under certain intractability assumptions, it turned out that none of the protocols of the scheme are secure in the presence of an active adversary. Due to space limitation, we here present only an interleaving attack against the `GKE.Join` protocol. Attacks against the `GKE.Setup` and `GKE.Remove` protocols will be given in the full version of this paper.

## 2 The `GKE.Join` Protocol

Let $\mathbb{G}$ be a finite cyclic group of $\ell$-bit prime order $q$, where $\ell$ is a security parameter, and let $g$ be an arbitrary generator of $\mathbb{G}$. The protocl uses three hash functions $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^{\ell}$, $\mathcal{H}_0 : \{0,1\}^* \rightarrow \{0,1\}^{\ell_0}$, and $\mathcal{H}_1 : \{0,1\}^{\ell_1} \times \mathbb{G} \rightarrow \{0,1\}^{\ell_0}$. Long-term keys are generated as follows:

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

Let $\mathcal{J}$ be a set of new clients who want to join an existing client group $\mathcal{G}_c$. Then, the client group $\mathcal{G}_c$ is updated to be $\mathcal{G}_c \cup \mathcal{J}$ and the `GKE.Join` protocol is performed to provide $S$ and each client $U_i \in \mathcal{G}_c$ with a new session key $sk$.

**Round 1.** Each new client $U_j \in \mathcal{J}$ chooses a random $x_j \in \mathbb{Z}_q$, and precomputes $y_j = g^{x_j}$, $\alpha_j = y^{x_j}$ and a signature $\sigma_j$ of $y_j$ under the signing key $SK_j$. Each client $U_j \in \mathcal{J}$ then sends $(y_j, \sigma_j)$ to the server $S$.

**Round 2.** The server $S$ verifies the incoming signatures. If they are all correct, $S$ increases the counter $\mathtt{c}$, computes the common secret value $K = \mathcal{H}_0(\mathtt{c} \| \{\alpha_i\}_{i \in \mathcal{I}_c})$, and sends to each client $U_i \in \mathcal{G}_c$ the values $\mathtt{c}$ and $K_i = K \oplus \mathcal{H}_1(\mathtt{c} \| \alpha_i)$.

**Key computation.** Each client $U_i \in \mathcal{G}_c$ already holds the value $\alpha_i = y^{x_i}$ and the old counter value (set to zero for the new ones). So it first checks that the new counter is greater than the old one, and simply recovers the common secret value $K = K_i \oplus \mathcal{H}_1(\mathtt{c} \| \alpha_i)$ and the session key $sk = \mathcal{H}(K \| \mathcal{G}_c \| S)$.

## 3 An Interleaving Attack on `GKE.Join`

Let's assume that a set of new clients, $\mathcal{J}$, wants to join two existing sessions of the `GKE.Setup` protocol with the client groups $\mathcal{G}_c$ ($\mathcal{A} \in \mathcal{G}_c$) and $\mathcal{G}'_c$ ($\mathcal{A} \notin \mathcal{G}'_c$), respectively. Assume further that the clients in $\mathcal{J}$ are permitted to join, and

thus two concurrent runs of the `GKE.Join` protocol are started with the new client groups $\mathcal{G}_c = \mathcal{G}_c \cup \mathcal{J}$ and $\mathcal{G}'_c = \mathcal{G}'_c \cup \mathcal{J}$, respectively. Then, an interleaving attack given below can be mounted against the clients in $\mathcal{J} \subset \mathcal{G}'_c$. Increasing the counter (in the second round of the `GKE.Join` protocol) does not play any role in preventing the following attack.

1. In the first round of the *second run*, the adversary intercepts all the messages $(y'_j, \sigma'_j)$ sent to $S$ by the clients in $\mathcal{J} \subset \mathcal{G}'_c$.
2. In the first round of the *first run*, the adversary $\mathcal{A}$ replaces the message $(y_j, \sigma_j)$ sent to $S$ by each client $U_j$ in $\mathcal{J} \subset \mathcal{G}_c$ with $(y'_j, \sigma'_j)$ obtained in the previous step of this scenario.
3. In the second round of the *first run*, the server $S$ operates as specified in the protocol since the received signatures are all valid; $S$ computes $\alpha_j = {y'_j}^x$ for each new client $U_j$ in $\mathcal{J} \subset \mathcal{G}_c$, increases the counter $\mathtt{c}$, and computes the common secret value $K$. The server $S$ then sends to each client $U_i$ in $\mathcal{G}_c$ the values $\mathtt{c}$ and $K_i = K \oplus \mathcal{H}_1(\mathtt{c} \| \alpha_i)$, and to the adversary $\mathcal{A}$ the values $\mathtt{c}$ and $K_{\mathcal{A}} = K \oplus \mathcal{H}_1(\mathtt{c} \| \alpha_{\mathcal{A}})$. Now, the adversary $\mathcal{A}$ intercepts all the messages sent by $S$ to the other clients, while recovering the common secret value $K$ from $K_{\mathcal{A}}$.
4. In the second round of the *second run*, the adversary $\mathcal{A}$ (pretending to be the server $S$) sends to each client $U_j$ in $\mathcal{J} \subset \mathcal{G}'_c$ the message $(\mathtt{c}, K_j)$ intercepted in the second round of the first run. After receiving this message from $\mathcal{A}$, each client in $\mathcal{J} \subset \mathcal{G}'_c$ first checks that the newly received counter is greater than the old one; this verification will succeed since the server $S$ increased the counter in the second round of the first run. Then, each client in $\mathcal{J} \subset \mathcal{G}'_c$ recovers $K$ from $K_j$ and compute the session key as:

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

Consequently, all the clients in $\mathcal{J} \subset \mathcal{G}'_c$ share with the adversary the same key $sk'$. Hence, the adversary $\mathcal{A}$ can decrypt all the encrypted messages sent to $S$ from the clients in $\mathcal{J} \subset \mathcal{G}'_c$. But, the clients in $\mathcal{J} \subset \mathcal{G}'_c$ believe that they have established a session key with $S$ while in fact they have shared it with $\mathcal{A}$.

## References


1. E. Bresson, O. Chevassut, A. Essiari, D. Pointcheval.: Mutual Authentication and Group Key Agreement for Low-Power Mobile Devices. Computer Communications, vol. 27, no. 17, pp. 1730–1737, 2004.