Stronger Security Model of Group Key Agreement

Jianjie Zhao[1], Dawu Gu[2], and M. Choudary Gorantla[3]

[1] School of Information Security Engineering, Shanghai Jiao Tong University. Shanghai 200240, China.
[2] Department of Computer Science and Engineering, Shanghai Jiao Tong University. Shanghai 200240, China.
[3] Society for Electronic Transactions and Security, CIT Campus, Taramani, Chennai 600113, India.
jjzhao81@gmail.com,dwgu@sjtu.edu.cn,mc.gorantla@gmail.com

Abstract. In PKC 2009, Gorantla, Boyd and González Nieto presented a nice result on modelling security for group key agreement (GKA) protocols. They proposed a novel security model (GBG model) that better supports the adversaries' queries than previous models for GKA protocols by considering KCI resilience. However, ephemeral key leakage attack resistance has been left outside the scope of the GBG model. In this paper, we demonstrate an ephemeral key leakage on an existing GKA protocol which has been shown secure in the GBG model. We then extend the GBG model by allowing the adversary greater attack powers of leaking ephemeral keys in GKA protocol session. We also apply the well known NAXOS trick to propose an improvement to an existing GKA protocol, which can resist the ephemeral key leakage attack. The security of the improved protocol has been argued under the our new model.

Keywords: Group key agreement; Ephemeral key leakage attack; Security model; Provable security

1 Introduction

Recent rapid development in networking and related mass communication media and digital technology opens the extensive possibilities for group applications. Group applications represent a special case of multi-party applications where the participants have some unifying relationship concerning their rights, responsibility and application goals [1]. There are many well-known forms of group applications such as digital conferences, text-based group communication, file and data sharing etc. On the other hand, security is increasingly important for these applications where group members want to prevent outsiders from obtaining communication content which sometimes are confidential or sensitive. The detailed information on the security issue for group applications can be obtained in [1].

Group key establishment is an information security process that enables users establish a session key known only to them. This session key can be applied for subsequent cryptographic usage. Group key establishment can be divided in two different classes: group key distribution (GKD) and group key agreement (GKA). The main characteristic of a GKD protocol is that it needs some trusted party (either a third party or a group member) and some trusted transport channel. In contrast, GKA protocol allows all participants to establish the session key where no party is needed to choose as the key generator and distributor and no secure channel is required. In this paper, we restrict our research to GKA protocols.

Although there are many security models [2–8] for two or three-party key agreement protocols, the formal model proposed by Bresson, Chevassut, Pointcheval and Quisquater [9], which we refer to as the BCPQ model, is the first security model for GKA protocols. Following the work of Bresson, Chevassut and Pointcheval [10] (BCP model) that extended the BCPQ model to the dynamic membership case, they [11] revised the BCP model to meet the internal state information reveal resistance (BCP+) model. However, these models do not consider any malicious behaviour by the participants themselves. In CCS 2005, Katz and Shin [12] proposed a security model for GKA protocol, which provide the first formal treatment of security of GKA in the presence of malicious participants. Bohli, González-Vasco and Steinwandt [13] (BGS model) and Bresson and Manulis [14] (BM model) later proposed two extensions of the KS model under different corruption models. Additionally, the BGS and BM model deal with the issue of contributiveness in the presence of malicious insiders.

All the models above provide definitions that consider the requirements on indistinguishability of computed group keys and forward secrecy. However, as discussed by Gorantla et al. [15] the above security notions do not provide the key compromise impersonation (KCI) resilience. To demonstrate the importance of considering resistance to KCI attacks for GKA protocols, the authors presented KCI attacks on the protocols of Boyd and González Nieto [16], Al-Riyami and Paterson [17] and Bresson et al. [18]. They then proposed a new model (GBG model) by taking KCI resilience into account and also showed that the protocol of Bohli et al. [13] (BGS protocol) to be secure in the GBG model.

Inspired by Manulis et al.'s recent work [19] where the authors pointed out that the GKA security models have so far not considered leakage of ephemeral keys, we augment the GBG model by considering ephemeral key leakage. In our new model, we allow the adversary to reveal any long-term key and ephemeral key of participants involved except for both long-term key and ephemeral key of one of the participants of the test session. We then improve the BGS protocol and show that it is secure in the new model. More specially, we have the following results:

1. *A stronger security model for GKA protocols.* By updating the definitions of adversary queries and notion of freshness of the GBG model, we propose a new model for GKA protocols which provides the adversary more attack capabilities than those in the earlier models.

2. *An attack on the BGS protocol and its improvement.* We first show that the BGS protocol is not secure in our new security model. To avoid this drawback, a solution where the ephemeral secret result will be generated by the hash function of the long-term key and the ephemeral key is put forward. This technique is known as the NAXOS trick in the literature [20].

3. *A straightforward security proof in the new model.* Under some well-studied assumptions, we prove the improved BGS protocol is secure in our new model.

We begin with a brief review of the GBG model in Section 2.1 and discuss the attack not covered by their definition in Section 2.2, then we give a simple ephemeral key leakage attack on the BGS protocol using this security flaw in the remainder of this section. Section 3 introduces our extended GBG model. In Section 4, we present an improved version of the BGS protocol which can resist the attack discussed in Section 2.2. The formal security argument is given in Section 5.

2 The GBG model and the BGS protocol

In this section, we overview the GBG model briefly and point out that the ephemeral key leakage attack is not covered by it. Then we present this attack on the BGS protocol which is provably secure in the GBG model.

2.1 Overview of GBG model

The GBG model [15] provides a formal security assurance to GKA protocols. We give a high-level overview of the GBG model in this subsection.

Participants. A GKA protocol runs in a network of multiple interconnected participants where each participant is activated to run sessions for itself and its peers. As a result, participants in such a session establish a key called a session key. We denote the participant set by $\mathcal{U} = \{U_1, U_2, \cdots, U_n\}$ and the protocol may be run among any subset of these parties. In a GKA protocol, each participant may execute a polynomial number of protocol instances in parallel. We refer to the i-th instance of $U \in \mathcal{U}$ as $\prod_U^i$.

Adversary model. The communication network is assumed to be fully controlled by an adversary M, that is, it may eavesdrop, delay, alter and insert messages during the process of the protocol at will. We define the security of a GKA protocol by a series of games between a challenger and the adversary M (an outsider adversary or a malicious insider) in which the adversary must solve a challenge on a test session. In this games, M is allowed to select the identities of all the honest participants and ask the following queries in any sequence.

– **Execute**$(\prod_U^i)$. The adversary makes this query to obtain the protocol messages that were exchanged during the honest execution $(\prod_U^i)$ of the protocol. This query models the passive attacks.

- **Send**($\prod_U^i, m$). The adversary makes this query to obtain the message that the instance $\prod_U^i$ would generate on receipt of the message m.
- **RevealKey**($\prod_U^i$). The adversary makes this query to obtain the session key established at the accepted instance $\prod_U^i$.
- **Corrup**(U). The adversary makes this query to obtain the long-term key of the participant U.
- **RevealState**($\prod_U^i$). The adversary makes this query to obtain the internal state of ($\prod_U^i$). We assume that the internal state is erased once $\prod_U^i$ has accepted. Hence, a **RevealState** query to an accepted instance returns nothing.
- **Test**($\prod_U^i$). Only one query of this type is allowed for the adversary. Provided that $\prod_U^i$ is accepted, the adversary M can make this query at any time. To respond to this query, a random bit $b \in \{0,1\}$ is secretly selected. If $b = 1$, then the real session key established at ($\prod_U^i$) is returned. Otherwise, a uniformly chosen random value from the session key distribution is returned.

We now define the **AKE security**, the **MA security** and the **Contributiveness** based on the definitions of partner and freshness.

Definition 1 (Partner). Informally, an instance $\prod_U^i$ is said to be *accepted* if it goes into an accept state after receiving the last expected protocol message, that is, when an instance accepts, it holds a session key, a session identifier *sid* and a partner identifier *pid*. The information of whether an instance has terminated with acceptance or not is assumed to be known to the adversary. Two instances $\prod_U^i$ and $\prod_{U'}^j$ at two different parties U and U' respectively are considered partnered iff

1. Both the instances have accepted.
2. The session identifiers of $\prod_U^i$ and $\prod_{U'}^j$ are the same.
3. The partner identifiers of $\prod_U^i$ and $\prod_{U'}^j$ are the same.

Definition 2 (Freshness). An instance $\prod_U^i$ is fresh if the following conditions hold:

1. The instance $\prod_U^i$ or any its partner has not been asked a **RevealKey** query after their acceptance.
2. The instance $\prod_U^i$ or any its partner has not been asked a **RevealState** query before their acceptance.
3. If $\prod_{U'}^j$ is a partner of $\prod_U^i$ and M asked **Corrupt** (U'), then any message that M sends to $\prod_U^i$ on behalf of $\prod_{U'}^j$ must come from $\prod_{U'}^j$ intended to $\prod_U^i$.

Definition 3 (AKE security). In Stage 1, an adversary M_{AKE} against the AKE security notion is allowed to make all the queries defined above. M_{AKE} makes **Test** query to an instance $\prod_U^i$ at the end of Stage 1 and is given a challenge key K_b. It can continue asking queries in Stage 2. Finally, M_{AKE} outputs a bit

b' and wins the AKE security game if (1) $b = b'$; (2) the instance $\prod_U^i$ that was asked **Test** query remained fresh till the end of M_{AKE}'s execution. We define the advantage of M_{AKE} in winning the AKE security game as $Adv_{M_{AKE}} = |2 \cdot Pr[b' = b] - 1|$. A protocol is called AKE secure if $Adv_{M_{AKE}}$ is negligible in the security parameter k.

Definition 4 (MA security). An adversary M_{MA} against the MA security notion is allowed to make all the queries defined above. M_{MA} violates the mutual authentication property of the GKA protocol if at some point during the protocol run, there exists an uncorrupted instance $\prod_U^i$ that has accepted with a session key sk_U^i and another party $U' \in pid_U^i$ that is uncorrupted at the time $\prod_U^i$ accepts such that

1. There is no instance $\prod_{U'}^j$ with $(pid_{U'}^j, sid_{U'}^j) = (pid_U^i, sid_U^i)$ **or**
2. There is an instance $\prod_{U'}^j$ with $(pid_{U'}^j, sid_{U'}^j) = (pid_U^i, sid_U^i)$ that has accepted with $sk_{U'}^j \neq sk_U^i$.

We define the probability of M_{MA} in winning the MA security game as $Adv_{M_{MA}}$. A protocol is said to provide mutual authentication in the presence of insiders if $Adv_{M_{MA}}$ is negligible in k.

Definition 5 (Contributiveness [14]). An adversary M_{Con} against the contributiveness notion is allowed to make all the queries defined above. It operates in two stages **prepare** and **attack** as follows:

prepare. M_{Con} queries the instance $\prod$ and outputs some state information ζ along with a key $\tilde{k}$.

At the end of **prepare** stage, a set $\sum$ is built such that $\sum$ consist of uncorrupted instances which have been asked either **Execute** or **Send** queries.

attack. On input $(\zeta, \sum)$, M_{Con} interacts with the instances of $\prod$ as in the **prepare** stage.

At the end of this stage M_{Con} outputs (U, i) and wins the games if an instance $\prod_U^i$ at an uncorrupted party U has terminated accepting $\tilde{k}$ with $\prod_U^i \notin \sum$.

We define the probability of M_{Con} in winning the above game as $Adv_{M_{Con}}$. A protocol is said to provide contributiveness property in the presence of insiders if $Adv_{M_{Con}}$ is negligible in k.

2.2 An attack not covered by the GBG model

We focus on Condition 2 in the definition of "Freshness" (Definition 2) and point out an attack which may compromise the security of GKA protocols proven secure in the GBG model. According to this condition, the GBG model does not allow the adversary to make **RevealState** query against the test session which it wants to attack. In other words, the GBG model does not provide any security guarantees for a session if the ephemeral key of either party has been leaked. To avoid such an attack (we call this attack the ephemeral key leakage (EKL) attack), GBG model assumes that the internal state should be erased after $\prod_U^i$ has accepted. However, this restriction prevents the adversary from revealing the ephemeral keys of participants during the protocol execution process, that is, the GBG model restricts the adversary's ability to attack the objective GKA protocol using the revealed ephemeral key during the protocol execution process. Next, we will present a concrete EKL attack on the BGS protocol which has been shown secure in the GBG model.

2.3 EKL attack on BGS protocol

The BGS protocol [13] proposed by Bohli et al. has been shown to satisfy their definitions of outsider and insider security. We briefly review the protocol here.

Round 1:

Computation

1. Each U_i chooses $k_i \overset{R}{\leftarrow} \{0,1\}^k$, $x_i \overset{R}{\leftarrow} \mathbb{Z}_q$ and computes $y_i = g^{x_i}$. U_n additionally computes $H(k_n)$.
2. Each U_i except U_n sets $M_i^I = k_i \| y_i$, while U_n sets $M_n^I = H(k_n) \| y_n$.
3. Each U_i computes a signature σ_i^I on $M_i^I \| pid_i$.

Broadcast

Each U_i broadcasts $M_i^I \| \sigma_i^I$.

Check

Each U_i checks all signatures σ_j^I of incoming messages $M_j^I \| \sigma_j^I$ for $j \neq i$.

Round 2:

Computation

1. Each U_i computes $t_i^L = H(y_{i-1}^{x_i})$, $t_i^R = H(y_{i+1}^{x_i})$, $T_i = t_i^L \oplus t_i^R$ and $sid = H(pid \| k_1 \| \cdots \| k_{n-1} \| H(k_n))$. U_n additionally computes $mask_n = k_n \oplus t_n^R$.
2. Each U_i except U_n sets $M_i^{II} = T_i \| sid_i$ while U_n sets $M_n^{II} = mask_n \| T_n \| sid_n$.
3. Each U_i computes a signature σ_i^{II} on M_i^{II}.

Broadcast

Each U_i broadcasts $M_i^{II} \| \sigma_i^{II}$.

Check

1. Each U_i verifies the incoming the signatures σ_j^{II} on the corresponding message M_j^{II} for each $j \in [1, n]$ and $j \neq i$ also checks that $T_1 \oplus \cdots \oplus T_n \overset{?}{=} 0$.
2. Each U_i for $i < n$, extracts $k_n = mask_n \oplus T_1 \oplus \cdots \oplus T_{i-1} \oplus t_i^L$ and checks the commitment $H(k_n)$ sent in Round 1 for the k_n extracted.

Key Computation

Each U_i computes the session key $sk = H(pid \| k_1 \| \cdots \| k_n)$.

Fig. 1. BGS protocol

Let $\mathcal{U} = \{U_1, U_2, \cdots, U_n\}$ be the set of the parties who want to establish the session key. Suppose that the group members are ordered in a logical ring with U_{i-1} and U_{i+1} being the left and right neighbors of U_i for $1 \leq i \leq n$, $U_0 = U_n$ and $U_{n+1} = U_1$. During the initialization phase, a cyclic group $\mathbb{G}$ of prime order q, an arbitrary generator g of $\mathbb{G}$ and the description of a hash function H that maps to $\{0,1\}^k$ are chosen. Each party is assumed to have a long-term key pair for public signature. Figure 1 shows the execution process after initialization phase.

We show that BGS protocol is vulnerable to the EKL attack. Without loss of generality, we assume that the ephemeral key of U_i is leaked, an adversary M can compute $t_i^L = H(y_{i-1}^{x_i})$ and $t_i^R = H(y_{i+1}^{x_i})$ in Round 2. Using the public $mask_n$, $T_1, T_2, \cdots, T_{i-1}$ and its own t_i^L, M will extract $k_n = mask_n \oplus T_1 \oplus T_2 \oplus \cdots \oplus T_{i-1} \oplus t_i^L$ and $sk = H(pid \| k_1 \| \cdots \| k_n)$. In this way, M easily win the AKE game by selecting the test session at U_i.

A straightforward solution is to delete x_i of participant U_i during the protocol execution process, but it is infeasible. In more specific terms, it is impossible for U_i to delete x_i after Round 1, since U_i has to use x_i in Round 2 to compute t_i^L and t_i^R; if U_i deletes x_i after Round 2, the unavoidable leakage of x_i during the first round could destroy the security of BGS protocol.

Note: It is worthy of being mentioned that BGS protocol is secure in GBG model since according to the principle of GBG model, all the ephemeral x_is will be erased after the corresponding session completed. Here we would like to reiterate that the attack not covered in the GBG model.

3 Extended GBG model

We now present our extended GBG (eGBG) model.

Participants. A GKA protocol runs in a network of multiple interconnected participants where each participant is activated to run sessions for itself and its peers. As a result, participants in such a session establish a session key. We denote the participant set by $\mathcal{U} = \{U_1, U_2, \cdots, U_n\}$ and the protocol may be run among any subset of these parties. In a GKA protocol, each participant may execute a polynomial number of protocol instances in parallel. We refer to the i-th protocol instance at $U \in \mathcal{U}$ as $\prod_U^i$.

Adversary model. The communication network is assumed to be fully controlled by an adversary M, that is, it may eavesdrop, delay, alter and insert messages during the process of the protocol at will. We define the security of a GKA protocol by a series of games between a challenger and the adversary M (an outsider adversary or a malicious insider) in which the adversary must solve a challenge on a test session. In this games, M is allowed to select the identities of all honest participants and issue the following queries in any sequence.

- **Execute**($\prod_U^i$)**.** The adversary makes this query to obtain the message that were exchanged during the honest execution ($\prod_U^i$) of the protocol. This query models the passive attacks.
- **Send**($\prod_U^i, m$). The adversary makes this query to obtain the message that the instance $\prod_U^i$ would generate on receipt of the message m.

- **RevealKey**$(\prod_U^i)$. The adversary makes this query to obtain the session key established at the accepted $\prod_U^i$.
- **Long-termKeyReveal**(U). The adversary makes this query to obtain the long-term key of the participant U.
- **EphemeralKeyReveal**$(\prod_U^i)$. The adversary makes this query to obtain the ephemeral key of U in the i-th run.
- **Test**$(\prod_U^i)$. Only one query of this form is allowed for the adversary. Provided that $\prod_U^i$ is accepted, the adversary M can make this query at any time. To respond to this query, a random bit $b \in \{0,1\}$ is selected. If $b = 1$, then the real session key is returned. Otherwise, a random value uniformly chosen from the session key distribution is returned.

We now define the notion of freshness according to the same definition of partnership in the GBG model.

Definition 6 (Freshness). An instance $\prod_U^i$ is fresh if the following conditions hold:

1. The instance $\prod_U^i$ or any its partner has not been asked a **RevealKey** query after their acceptance.
2. The instance $\prod_U^i$ or any its partner has not been asked a **Long-termKeyR eveal** query and a **EphemeralKeyReveal** query simultaneously before their acceptance. That is, the adversary is allowed to reveal any subset of four pieces of secret information (the long-term keys and the ephemeral keys of the participant and its partner) which does not contain both the long-term and ephemeral keys of one party.
3. If $\prod_{U'}^j$ is a partner of $\prod_U^i$ and M asked **Long-termKeyReveal** (U'), then any message that M sends to $\prod_U^i$ on behalf of $\prod_{U'}^j$,must come from $\prod_{U'}^j$ intended to $\prod_U^i$.

The description of the **AKE security**, **MA security** and **Contributiveness** are the same as that in the GBG model in Section 2.1.

Compared to the GBG model, we

1. Replace the **RevealState** query with the **EphemeralKeyReveal** by which the adversary can get the ephemeral key of the instance $\prod_U^i$. At the same time, we remove the limitation of the original GBG model where once $(\prod_U^i)$ has accepted the internal state is erased.
2. Replace the **Corrupt** query with the **Long-termKeyReveal** to give the adversary the power to reveal the long-term keys.
3. Modify the Condition 2 in the notion of "freshness" to expand the attack scope of the adversary.

Remark 1. In many security models, when the adversary corrupts an honest party, it takes full control over this party and reveals all its secret information. In the GBG model, since the **Corrupt** query just simulates the long-term key leakage attack, we change the tag of this query to capture the adversarial ability more accurately. The role of **Long-termKeyReveal** in our model is the same as that of **Corrupt** in the GBG model.

Overall, eGBG model extends the adversarial capabilities to the following extent: we allow the adversary to reveal any subset of the long-term keys and ephemeral keys of $\prod_U^i$ and its partner which does not contain both the long-term and ephemeral keys of one party.

4 BGS+ protocol

To construct a GKA protocol which is secure in eGBG model, we modify some details of the original BGS protocol and get BGS+ protocol.

4.1 Description of BGS+ protocol

We first introduce some notation that will be used in the rest of the paper.

$\mathcal{U} = \{U_1, U_2, \cdots, U_n\}$: The set of the parties who want to establish the session key. We also suppose that the group members are ordered in a logical ring.

p, q, g: Two large primes p and q with $q|(p-1)$, and a generator g of group $\mathbb{G}$ with order q.

k: The security parameter in this protocol.

(pk_i, sk_i): The long-term key pair of the participant U_i which is used for the signature.

k_i, x_i: The ephemeral keys pair of the participant U_i.

$H : \{0,1\}^* \to \{0,1\}^\lambda$: Hash function modelled as a random oracle, where λ is another security parameter.

$H_1/H_2 : \{0,1\}^* \to Z_q/Z_p$: Hash functions modelled as random oracles.

SK: Session key established by the participants.

This protocol aims to establish a confidential communication. The detailed description can be seen in Fig.2. Note that we drop the operator "mod$p$" for clarity.

4.2 Some design principles

The most significant difference between BGS protocol and ours is that we use $\widetilde{x_i} = H_1(sk_i, x_i)$ instead of x_i as the ephemeral intermediate result to generate y_i, t_i^L and t_i^R. This benefit of doing so is obvious: since we will destroy $\widetilde{x_i}$ after getting the needed values, this method avoid the possibility that the ephemeral secret state will be obtained by the adversary during the protocol execution process. On the other hand, even if the adversary gets the ephemeral key x_i, it can not obtain the $\widetilde{x_i}$ yet since it has to get the long-term key sk_i at the same time, but it is not allowed to reveal the long-term key and the ephemeral key simultaneously in eGBG model.

5 Security Analysis

Now, we prove that BGS+ protocol achieves the **AKE security** goal, the **MA security** goal and the **Contributiveness** under reasonable and well-defined intractability assumptions.

Round 1:

Computation

1. Each U_i chooses $x_i \overset{R}{\leftarrow} \mathbb{Z}_q$ and computes $\widetilde{x_i} = H_1(sk_i, x_i)$ and $y_i = g^{\widetilde{x_i}}$. After obtaining y_i, U_i destroys $\widetilde{x_i}$.

2. Each U_i sets $M_i^I = y_i \| pid$.

3. Each U_i computes a signature σ_i^I on M_i^I.

Broadcast

Each U_i broadcasts $M_i^I \| \sigma_i^I$.

Round 2:

Check

Each U_i checks all signatures σ_j^I of incoming messages $M_j^I \| \sigma_j^I$ for $j \neq i$.

Computation

1. Each U_i computes $\widetilde{x_i} = H_1(sk_i, x_i)$, $t_i^L = H_2(y_{i-1}^{\widetilde{x_i}})$, $t_i^R = H_2(y_{i+1}^{\widetilde{x_i}})$ and $T_i = t_i^L \oplus t_i^R$.

2.Each U_i chooses $k_i \overset{R}{\leftarrow} \{0,1\}^k$. Then U_i except U_n computes $M_i^{II} = T_i \| k_i$. U_n computes $mask_n = k_n \oplus t_n^R$ and $H_2(k_n)$ and sets $M_n^{II} = mask_n \| T_n \| H_2(k_n)$. After this step, each U_i destroys $\widetilde{x_i}$, t_i^L and t_i^R, U_n additionally destroys k_n.

3. Each U_i computes a signature σ_i^{II} on M_i^{II}.

Broadcast

Each U_i broadcasts $M_i^{II} \| \sigma_i^{II}$.

After Round 1 and Round 2

Check

1. Each U_i verifies the incoming the signatures σ_j^{II} on the corresponding message M_j^{II} for each $j \in [1, n]$ and $j \neq i$ also checks that $T_1 \oplus \cdots \oplus T_n \overset{?}{=} 0$.

2. Each U_i for $i < n$, computes $\widetilde{x_i} = H_1(sk_i, x_i)$ and $t_i^L = H_2(y_{i-1}^{\widetilde{x_i}})$, extracts $k_n = mask_n \oplus T_1 \oplus \cdots \oplus T_{i-1} \oplus t_i^L$ and checks the commitment $H_2(k_n)$ sent in Round 2 for the k_n extracted. If so, each U_i computes $sid = H_2(pid \| k_1 \| \cdots \| k_{n-1} \| H_2(k_n))$.

Key Computation

Each U_i computes the session key $SK = H(sid \| k_1 \| \cdots \| k_n)$.

Fig. 2. BGS+ protocol

Theorem 1 (AKE security). Let H_1, H_2 and H be three random oracles and the signature scheme used in GBS+ is UF-CMA secure, M_{AKE} denotes an adversary that has the ability to perform the oracle queries mentioned in Section 3. Then BGS+ is AKE secure if the CDH assumption holds in G. Furthermore, suppose M_{AKE} has an advantage $Adv_{BGS+}^{AKE}(M_{AKE})$ to break AKE security of BGS+ by activating at most l sessions for each party, then there is a CDH problem solver W that can solve the CDH problem in polynomial time with an advantage $Adv_{BGS+}^{CDH}(W)$ and

$Adv_{BGS+}^{AKE}(M_{AKE}) \leq \frac{q_s^2+q_{H_2}}{2^{k-1}} + \frac{q_{H_1}^2}{q} + \frac{q_{H_2}^2}{p} + \frac{q_H^2}{2^\lambda} + 2n^2 \cdot Adv_{BGS+}^{Sig} + 2nl \cdot Adv_{BGS+}^{CDH}(W)$, where n is the number of parties; Adv_{BGS+}^{Sig} denotes the advantage of a polynomial adversary against the UF-CMA security of the signature; q_{H_1}, q_{H_2} and q_H are the maximum number of times of hash queries H_1, H_2 and H, respectively; q_s represents the maximum number of queries to the **Send** oracles asked by the adversary.

Proof. In this proof, we incrementally define a sequence of games starting at the real execution $G0$ and ending up at game $G5$ where the adversary has no advantage. Each game addresses a different security aspect. For each game G_i, we define $Succ_i$ $(i = 0, 1, \cdots, 5)$ as the event that M_{AKE} wins the game.

We use the game hopping technique based on the failure events and bridging steps suggested by Shoup [21]. Consider an event E that may occur during the adversary M_{AKE}'s execution such that E is detectable by the simulator, E is independent of $Succ_i$, G_i and G_{i+1} are identical unless E occurs, then we have

$$|Pr[Succ_{i+1}] - Pr[Succ_i]| \leq Pr[\mathsf{E}]. \quad (1)$$

Game $G0$. This game corresponds to the real execution. In the game, we don't modify the simulation of the oracles, all the instances modeled as the real execution in the random oracle model. By the definition, we have

$$Adv_{BGS+}^{AKE}(M_{AKE}) = |2 \cdot Pr[Succ_0] - 1|. \quad (2)$$

Game $G1$. This game is as the same as the game $G0$ except that the simulation fails if the event Forge happens.

The event Forge occurs when the adversary M_{AKE} successfully forgery of one of the honest parties' signatures. According to the analysis in Gorantla et al.'s work [15], if the case happens, it will contradict against the assumption that the signature scheme in BGS+ protocol is UF-CMA secure.

We denote Adv_{BGS+}^{Sig} be the advantage of a polynomial adversary against the UF-CMA security of the signature, since the probability that M_{AKE} doesn't get the long-term key of the target party is $\geq \frac{1}{n}$ and the probability of M_{AKE} outputting a valid forgery on behalf of the target party is also $\geq \frac{1}{n}$, we have $Adv_{BGS+}^{Sig} \geq \frac{1}{n^2}$, then

$$Pr[Succ_1] - Pr[Succ_0] \leq Pr[\mathsf{Forge}] \leq n^2 \cdot Adv_{BGS+}^{Sig}. \quad (3)$$

Game $G2$. In this game, we simulate all oracles in $G1$, except that we halt all executions in which the event $\mathsf{Collision}$ occurs.

The event Collision occurs when the random oracles H_1, H_2 and H produce collisions for any of their own inputs. According to the birthday paradox, the probability of collisions in output of H_1 and H_2 oracles are at most $q_{H_1}^2/(2q)$ and $q_{H_2}^2/(2p)$ respectively where q_{H_i} for $i = 1, 2$ denotes the maximum number of queries to H_i. Similarly, the probability of collision in output of H oracle is at most $q_H^2/2^{\lambda+1}$, where q_H is the maximum number of times of hash queries H. Consequently,

$$|Pr[Succ_2] - Pr[Succ_1]| \leq Pr[\mathsf{Collision}] \leq \frac{q_{H_1}^2}{2q} + \frac{q_{H_2}^2}{2p} + \frac{q_H^2}{2^{\lambda+1}}. \tag{4}$$

Game $G3$. This game is as the same as the previous game except that the simulation fails if the event Repeat occurs.

The event Repeat occurs when different parties choose the same random value k_i as their own shares. Let q_s be the maximum number of queries to the **Send** oracles asked by the adversary, then $Pr[\mathsf{Repeat}] \leq \frac{q_s^2}{2^k}$. Therefore, we have

$$|Pr[Succ_3] - Pr[Succ_2]| \leq Pr[\mathsf{Repeat}] \leq \frac{q_s^2}{2^k}. \tag{5}$$

Game $G4$. In this game, we change the simulation of queries to the **Send** query. This time, we change the way we compute the value y_{i-1} in some manner (we will describe these ways subsequently) to replace y_{i-1} when the adversary makes the corresponding **Send** query. Then we will have

$$|Pr[Succ_4] - Pr[Succ_3]| \leq nlq_{H_2} \cdot Adv_{BGS+}^{CDH}(W). \tag{6}$$

Proof. By assuming a successful adversary against this game, we can construct a CDH solver W.

There is only difference between $G3$ and $G4$ in the way to generate y_{i-1} and y_i for a session, we will show the difference between the current game and the previous one is negligible as long as the CDH assumption holds.

First, the CDH solver W obtains CDH tuple (g, g^a, g^b). As the real execution, W randomly select a party U_i $(i = 1, 2, \cdots, n)$ and a session executed by it as the test session. When the corresponding **Send** query is asked, the CDH solver sets $A = y_{i-1}$ and $B = y_i$ and returns A and B to U_i. For other queries, W returns as $G3$.

Let $Pr_1(M_{AKE})$ be the probability of the event that M_{AKE} will select U_i and the corresponding chosen by W as the test session, since there are n parties and each party can active at most l sessions, we have $Pr_1(M_{AKE}) \geq \frac{1}{ln}$.

By the definition of freshness, the adversary is allowed to reveal any subset of long-term keys and ephemeral keys of the parties, but it is not allowed to get both the long-term keys and ephemeral keys of one party. Special for the party U_i, eGBG model allows the adversary to reveal sk_i or x_i, but doesn't allow to reveal both sk_i and x_i. Therefore, the CDH solver simulates all oracle queries without knowing a and b ($a = \widetilde{x_i}$ and $b = \widetilde{x_{i-1}}$).

Following the simulation principle above, if M_{AKE} successfully violates the AKE security of BGS+ protocol, it must have obtained $k_n = mask_n \oplus T_1 \oplus \cdots \oplus$

$T_{i-1} \oplus t_i^L$ and know the value $t_i^L = H_2(y_{i-1}^{\widetilde{x_i}}) = H_2(y_i^{\widetilde{x_{i-1}}})$. Since we assume H_2 is modeled as a random oracle, to get t_i^L, the only way for M_{AKE} is to own $y_{i-1}^{\widetilde{x_i}}$ ($y_i^{\widetilde{x_{i-1}}}$) and ask **Hash** query (H_2). Once it asks **Hash** query (H_2), it has to leave the correct $y_{i-1}^{\widetilde{x_i}}$ ($y_i^{\widetilde{x_{i-1}}}$).

With the probability $Pr_2(M_{AKE}) \geq \frac{1}{q_{H_2}}$, W can choose the appropriate $y_{i-1}^{\widetilde{x_i}}$ ($y_i^{\widetilde{x_{i-1}}}$) and solve the CDH problem with the advantage $Adv_{BGS+}^{CDH}(W)$ in polynomial time: $CDH(A, B) = CDH(g^a, g^b) = y_{i-1}^{\widetilde{x_i}} = y_i^{\widetilde{x_{i-1}}} = g^{ab}$.

As we can see form the above analysis, we have $Adv_{BGS+}^{CDH}(W) \geq \frac{1}{nlq_{H_2}}|Pr[Succ_4] - Pr[Succ_3]|$, that is, $|Pr[Succ_4] - Pr[Succ_3]| \leq nlq_{H_2} \cdot Adv_{BGS+}^{CDH}(W)$.

Game $G5$. This game is as the same as $G4$ except that in the test session the game halts if M_{AKE} asks **Hash** query (H) with the corresponding input $(sid\|k_1\| \cdots \|k_n)$. Since there is no useful information about k_n in Round 2, the only way for the adversary to getting k_n is to guess it with the probability $\frac{1}{2^k}$ and to check it using $H_2(k_n)$. Therefore,

$$|Pr[Succ_5] - Pr[Succ_4]| \leq \frac{q_{H_2}^2}{2^k}. \tag{7}$$

On the other hand, if the adversary doesn't make any **Hash** query for H_2 with the correct input, it will not have any advantage in distinguishing the real session key from a random one and so $Pr[Succ_5] = \frac{1}{2}$.

Together with (2)-(7), Theorem 1 is proven.

Theorem 2 (MA security). Let H_1, H_2 and H be three random oracles and the signature scheme used in GBS+ is UF-CMA secure, M_{MA} denotes an adversary that has the ability to perform the oracle queries mentioned in Section 3. Then BGS+ is MA secure. Furthermore, the MA advantage of M_{MA} $Adv_{BGS+}^{MA}(M_{MA})$ is upper bounded by

$Adv_{BGS+}^{MA}(M_{MA}) \leq |n^2 \cdot Adv_{BGS+}^{Sig} + \frac{q_s^2}{2^{k-1}} + \frac{q_{H_1}^2}{q} + \frac{q_{H_2}^2}{p} + \frac{q_H^2}{2^\lambda} - 1|$, where n is the number of parties; Adv_{BGS+}^{Sig} denotes the advantage of a polynomial adversary against the UF-CMA security of the signature; q_{H_1}, q_{H_2} and q_H are the maximum number of times of hash queries H_1, H_2 and H; q_s represents the maximum number of queries to the **Send** oracles asked by the adversary.

Proof. We follow the hypothesis of the proof of Theorem 1 and define $Succ_i$ ($i = 0, 1, 2, 3$) as the event that the adversary M_{MA} wins the game Gi.

Game $G0$. Like $G0$ in Theorem 1, this game represents the real execution. In the game, all the instances modeled as the real execution in the random oracle model. By the definition, we have

$$Adv_{BGS+}^{MA}(M_{MA}) = |2 \cdot Pr[Succ_0] - 1|. \tag{8}$$

Game $G1$. In this game, we simulate all oracles in $G0$, except that we halt all executions in which the event Forge defined in Theorem 1 occurs. Then we have

$$Pr[Succ_1] - Pr[Succ_0] \leq Pr[\mathsf{Forge}] \leq n^2 \cdot Adv_{BGS+}^{Sig}. \tag{9}$$

Game $G2$. In this game, we simulate all oracles in $G1$, except that we halt all executions in which the event $\mathsf{Collision}$ defined in Theorem 1 occurs. Consequently,

$$|Pr[Succ_2] - Pr[Succ_1]| \leq Pr[\mathsf{Collision}] \leq \frac{q_{H_1}^2}{2q} + \frac{q_{H_2}^2}{2p} + \frac{q_H^2}{2^{\lambda+1}}. \quad (10)$$

Game $G3$. This game is as the same as the previous game except that the simulation fails if the event Repeat defined in Theorem 1 occurs. Then we have

$$|Pr[Succ_3] - Pr[Succ_2]| \leq Pr[\mathsf{Repeat}] \leq \frac{q_s^2}{2^k}. \quad (11)$$

If $G3$ doesn't abort, all the honest parties compute the same key. Hence $Pr[Succ_3] = 0$.

Together with (8)-(11), Theorem 2 is proven.

Theorem 3 (Contributiveness). Let H_1, H_2 and H be three random oracles and the signature scheme used in GBS+ is UF-CMA secure, M_{Con} denotes an adversary that has the ability to perform the oracle queries mentioned in Section 3 with the advantage $Adv_{BGS+}^{Con}(M_{Con})$. Then BGS+ is contributive and $Adv_{BGS+}^{Con}(M_{Con}) \leq |\frac{q_s^2}{2^{k-1}} + \frac{q_{H_2}^2}{p} + \frac{q_H^2}{2^\lambda} - 1|$, where q_{H_2} and q_H are the maximum number of times of hash queries H_2 and H; q_s represents the maximum number of queries to the **Send** oracles asked by the adversary.

Proof. Following the proofs on Theorem 1 and Theorem 2, we construct four games from $G0$ to $G3$.

Game $G0$. This game represents the real execution in the random oracle model. By the definition, we have

$$Adv_{BGS+}^{Con}(M_{Con}) = |2 \cdot Pr[Succ_0] - 1|. \quad (12)$$

Game $G1$. In this game, we simulate all oracles as in $G0$, except that we halt all executions in which the event Repeat defined in Theorem 1 occurs. Then we have

$$|Pr[Succ_1] - Pr[Succ_0]| \leq Pr[\mathsf{Repeat}] \leq \frac{q_s^2}{2^k}. \quad (13)$$

Game $G2$. In this game, we simulate all oracles in $G1$, except that we halt all executions in which the event $\mathsf{Collision1}$ for input k_n occurs. Consequently,

$$|Pr[Succ_2] - Pr[Succ_1]| \leq Pr[\mathsf{Collision1}] \leq \frac{q_{H_2}^2}{2p}. \quad (14)$$

Game $G3$. In this game, we simulate all oracles in $G2$, except that we halt all executions in which the event $\mathsf{Collision2}$ for input $(sid\|k_1\|\cdots\|k_n)$ occurs Consequently,

$$|Pr[Succ_3] - Pr[Succ_2]| \leq Pr[\mathsf{Collision2}] \leq \frac{q_H^2}{2^{\lambda+1}}. \quad (15)$$

If $G3$ doesn't abort, the output of the random oracle H is uniformly distributed. Hence, $Pr[Succ_3] = 0$.

Together with (12)-(15), Theorem 3 is proven.

6 Conclusion

We analyze GBG model and find that this model doesn't consider the ephemeral key leakage resistance property. Considering this drawback, we extend GBG model to the following extent: the only corruption powers we don't give an adversary in the game are that it is not allowed to reveal both the long-term key and the ephemeral key of a legitimate party simultaneously. We then introduce a new GKA protocol called BGS+ protocol based on BGS protocol which is provably secure in GBG model but cannot resist ephemeral key leakage attack and prove that it is secure in the new model. As far as we know, this is the first provably secure GKA protocol in the strongest security model.

As for limitations, we didn't take the efficiency into account. A natural direction for further research is the design of provably secure protocols with lower computational cost and communication cost. Additionally, the design of GKA protocols in standard models where the security of the protocols isn't dependent on the security of random oracles is another imperative future research consideration.

Acknowledgement

This work is partially supported by the Specialized Research Fund for the Doctoral Program of Higher Education (No. 200802480019) and the National Natural Science Foundation of China (No.61073150).

References

1. Manulis, M.: Provably Secure Group Key Exchange. Volume 5 of IT Security. Europäischer Universitätsverlag, Berlin, Bochum, Dülmen, London, Paris (August 2007) Available at `http://www.manulis.eu/phd.html`. Last accessed on 29 May 2008.
2. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Advances in Cryptology–CRYPTO'93. Volume 773 of LNCS., Springer (1993) 232–249
3. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Advances in Cryptology - EUROCRYPT'01. Volume 2045 of LNCS., Springer (2001) 453–474
4. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger Security of Authenticated Key Exchange. In Susilo, W., Liu, J.K., Mu, Y., eds.: Provable Security, First International Conference, ProvSec 2007. Volume 4784 of LNCS., Springer (2007) 1–16
5. Abdalla, M., Fouque, P.A., Pointcheval, D.: Password-Based Authenticated Key Exchange in the Three-Party Setting. In Vaudenay, S., ed.: Public Key Cryptography–PKC'05. Volume 3386 of LNCS., Springer (2005) 65–84
6. Wang, W., Hu, L.: Efficient and Provably Secure Generic Construction of Three-Party Password-Based Authenticated Key Exchange Protocols. In Barua, R., Lange, T., eds.: Progress in Cryptology–INDOCRYPT'06. Volume 4329 of LNCS., Springer (2006) 118–132

7. Yoneyama, K.: Efficient and Strongly Secure Password-Based Server Aided Key Exchange (Extended Abstract). In Chowdhury, D.R., Rijmen, V., Das, A., eds.: Progress in Cryptology–INDOCRYPT'08. Volume 5365 of LNCS., Springer (2008) 172–184
8. Blake-Wilson, S., Johnson, D., Menezes, A.: Key Agreement Protocols and Their Security Analysis. In: Cryptography and Coding, 6th IMA International Conference. Volume 1355 of LNCS., Springer (1997) 30–45
9. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.J.: Provably authenticated group Diffie-Hellman key exchange. In: CCS'01: Proceedings of the 8th ACM conference on Computer and Communications Security, ACM (2001) 255–264
10. Bresson, E., Chevassut, O., Pointcheval, D.: Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case. In: Advances in Cryptology–ASIACRYPT'01. Volume 2248 of LNCS., Springer (2001) 290–309
11. Bresson, E., Chevassut, O., Pointcheval, D.: Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In: Advances in Cryptology–EUROCRYPT'02. Volume 2332 of LNCS., Springer (2002) 321–336
12. Katz, J., Shin, J.S.: Modeling insider attacks on group key-exchange protocols. In: Proceedings of the 12th ACM Conference on Computer and Communications Security–CCS'05, ACM (2005) 180–189
13. Bohli, J.M., Gonzalez Vasco, M.I., Steinwandt, R.: Secure group key establishment revisited. Int. J. Inf. Sec. **6**(4) (2007) 243–254
14. Bresson, E., Manulis, M.: Securing Group Key Exchange against Strong Corruptions. In: Proceedings of ACM Symposium on Information, Computer and Communications Security (ASIACCS'08), ACM Press (2008) 249–260
15. Gorantla, M.C., Boyd, C., González Nieto, J.M.: Modeling Key Compromise Impersonation Attacks on Group Key Exchange Protocols. In Jarecki, S., Tsudik, G., eds.: Public Key Cryptography–PKC'09. Volume 5443 of LNCS., Springer (2009) 105–123
16. Boyd, C., Nieto, J.M.G.: Round-Optimal Contributory Conference Key Agreement. In: Public Key Cryptography–PKC'03. Volume 2567 of LNCS., Springer (2003) 161–174
17. Al-Riyami, S.S., Paterson, K.G.: Tripartite Authenticated Key Agreement Protocols from Pairings. In: Cryptography and Coding, 9th IMA International Conference. Volume 2898 of LNCS., Springer (2003) 332–359
18. Bresson, E., Chevassut, O., Essiari, A., Pointcheval, D.: Mutual Authentication and Group Key Agreement for Low-Power Mobile Devices. In: Proc. of MWCN 03, World Scientific Publishing (October 2003) 5962
19. Manulis, M., Suzuki, K., Ustaoglu, B.: Modeling Leakage of Ephemeral Secrets in Tripartite/Group Key Exchange. In Lee, D., Hong, S., eds.: Information, Security and Cryptology–ICISC'09, Revised Selected Papers. Volume 5984 of LNCS., Springer (2010) 16–33
20. Lauter, K., Mityagin, A.: Security Analysis of KEA Authenticated Key Exchange Protocol. In Yung, M., Dodis, Y., Kiayias, A., Malkin, T., eds.: Public Key Cryptography–PKC'06. Volume 3958 of LNCS., Springer (2006) 378–394
21. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004) `http://eprint.iacr.org/`.