ISSN 1342-2812

Research Reports on Mathematical and Computing Sciences

Key-Substitution Attacks on Group Signature

Koichi Sakumoto and Keisuke Tanaka

January 2007, C–237

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES C: Computer Science

Key-Substitution Attacks on Group Signature

Koichi Sakumoto and Keisuke Tanaka *

Dept. of Mathematical and Computing Sciences
Tokyo Institute of Technology
W8-55, 2-12-1 Ookayama Meguro-ku, Tokyo 152-8552, Japan
{sakumot3, keisuke}@is.titech.ac.jp

January 4, 2007

Abstract

Group signatures were introduced by Chaum and Van Heyst [12], and many security requirements for group signatures have been proposed. Bellare, Micciancio, and Warinschi [4] showed that satisfying full-anonymity and full-traceability is sufficient, in the sence that all the above-mentioned requirements are implied by them. Wilson and Menezes [5] introduced a considerable attack against standard signatures, key substitution attack. In this paper, we propose security conditions of group signatures against this attack and show that the security requirements are not sufficient regarding the attack. We also propose a group signature scheme that is secure against the key substitution attack, fully-anonymous, and fully-traceable.

Keywords: group signature scheme, key substitution attack

1 Introduction

Group signatures, introduced by Chaum and Van Heyst [12], allow any member of a group to sign a message on behalf of the group. Moreover, no one can find out which group member provided it. Preserving the anonymity of the signer can be important in many applications where the signer does not want to be directly identified with the message that he signed. However, there exist a special party, called the group manager, who has the ability to trace the signer of any given signature in the case of a later dispute.

The core requirements of group signatures are that the group manager has a secret key *gmsk* based on which it can, given a signature σ, extract the identity of the group member who created σ (traceability) and on the other hand an entity not holding *gmsk* should not be able, given a signature σ, to extract the identity of the group member who created σ (anonymity).

Major requirements have been introduced as follows.

unforgeability It is computationally unfeasible to produce a message and signature pair (m, σ) that are accepted by the verification algorithm, without knowledge of the secret key(s).

exculpability No member of the group and not even the group manager can produce signatures on behalf of other users [3].

*Supported in part by NTT Information Sharing Platform Laboratories and Grant-in-Aid for Scientific Research, Ministry of Education, Culture, Sports, Science, and Technology, 16092206.

traceability Originally [12], "traceability" was used to denote the functional property that if a message is signed with a i-th user's key and the opening algorithm is applied to the resulting signature, the output of the opening algorithm should be i. Later, the term has been overloaded to include an actual security requirement, namely that it is not possible to produce signatures which can not be traced to one of the group that has produced the signature.

coalition resistance No colluding subset of group members (even if comprised of the entire group) can generate a valid signature that the group manager can not be traced to any of them [1].

framing resistance No set of group members can combine their keys to produce a valid signatures in such a way that the opening algorithm will attribute the signature to a different member of the group [13].

anonymity It is infeasible to ascertain the group member who signed a message without knowing the group maneges secret key.

unlinkability It is not feasible to decide whether two signatures have been issued by the same group member or not.

Bellare, Micciancio, and Warinschi [4] formulated strong versions of the anonymity and traceability, which they called full-anonymity and full-traceability respectively. Moreover, they showed satisfying the two requirements is sufficient, in the sense that all the above-mentioned requirements are implied by full-anonymity plus full-traceability.

Wilson and Menezes [5] introduced a property which they called duplicate-signature key selection property on a signature scheme. In this scenario, an attacker is given a message m along with a signature s for a verification key y and tries to find another verification key $\overline{y}$ (along with the matching secret key) such that s is valid on m with respect to $\overline{y}$, too.

Menezes and Smart [16] formalized this property as the key substitution attack and proved that several established signature schemes, including DSA, EC-DSA, and Schnorr signatures, are secure in this setting, provided that the parameters are restricted accordingly. In this formulation, an attacker is provided some message along with a signature s for a verification key y and attempts to produce a valid pair of a public key and a private key on which s is valid for the provided message.

Bohli, Rohrich, and Steinwandt [6] considered other two attack models, which were derived on the assumptions that there exists a "corrupted signer" and a "malicious signer", respectively. In the former model, an attacker is presented both a public key and a secret key. In the later model, given a domain parameter, an attacker himself produces two public (and secret) keys $y, \overline{y}$, a message m, and a signature σ such that σ is valid signature for m on both y and $\overline{y}$. They also classified the goals of an attacker, a strong model and a weak model. They call the strong key substitution attack if an attacker needs to output both a public key and a corresponding secret key. In the weak key substitution attack, an attacker does not output a secret key.

In 2004, Boneh and Boyen [7] proposed a signature scheme based on a bilinear map in the standard model, which is existential unforgeable against adaptive chosen message attack. Tan showed that Boneh and Boyen's scheme succumbs to the key substitution attack [20], and proposed a signature scheme based on a bilinear map that is secure against the key substitution attack and existential unforgeable in the standard model [21].

For several signature schemes, including the proposals that are provably secure in the sense of existential unforgeability, it has been demonstrated that key substitution attacks are possible [5, 16, 14, 19, 6, 20].

In this paper, we formulate the security requirements against the key substitution attack [16, 14, 19, 20, 21, 6] on group signature schemes. Bellare, Micciancio, and Warinschi [4] formulate

the full-anonymity and the full-traceability and sais that two requirements are enough, in the sense that all the other requirements are implied by them. Nevertheless, we show that Boneh, Boyen and Shacham's short group signatures [8] which are full-anonymous and full-traceable are not secure against key substitution attack. This means that satisfying the full-anonymity and the full-traceability is not sufficient to be secure against the attack.

We propose the security against key substitution attacks as the additional requirement for group signature schemes, and formalize two security requirements against this attack. In one of the scenarios, an attacker is provided some message along with a signature s for a group public key gpk and attempts to produce another group public key $\overline{gpk}$ for which s is valid for the provided message. In the other scenario, an attacker is provided a group public key gpk, a group manager's secret key $gmsk$, and users' secret keys **gsk** and attempts to produce another group public key $\overline{gpk}$, a message m, and a signature σ such that σ is valid signature for m on both gpk and $\overline{gpk}$. We call the former attack "key substitution attack 1" and the latter "key substitution attack 2". We prove that the security requirement against key substitution attack 2 is truly stronger than that against key substitution attack 1. We also construct a group signature scheme that is fully-traceable, CPA-fully-anonymous, and secure against key substitution attack 2.

In the next section, we review some cryptographic tools that our main construction based on. In section 3, we review the model of group signature schemes and their proposed security requirement. We also formalize two security definition against key substitution attacks. One is inspired by Menezes and Smart [16] and Tan [21], the other is stronger attack. In section 4, we show the insecurity of a fully-traceable and CPA-fully-anonymous group signature scheme proposed by Boneh, Boyan and Shacham [8] against the weaker key substitution attack. We improve the group signatue scheme and show that the modified scheme is fully-traceable, CPA-fully-anonymous, and secure against the stronger key substitution attack. We prove that there exists a separation between the weaker and stronger key substitution attacks in section 5.

2 Primitive

If x is a string, then $|x|$ denotes its length, while if S is a set then $|S|$ denotes its size. If $k \in \mathbb{N}$ then 1^k denotes the string of k ones. For any integer n, $[n] = \{1, \cdots n\}$. For a randomized algorithm $\mathtt{A}$, $[\mathtt{A}(x, y, \cdots)]$ denotes the set of all points having positive probability of being output by $\mathtt{A}$ on inputs $x, y, \cdots$. For a randomized algorithm $\mathtt{A}$, $z \xleftarrow{\$} \mathtt{A}(x, y, \cdots)$ denotes the operation where on the input $x, y, \cdots$, $\mathtt{A}$ outputs z.

We say that a function is polynomially bounded if there exists a polynomial $p(\cdot)$ such that $f(k) \leq p(k)$ for all sufficiently large $k \in \mathbb{N}$. We also say f is nice if it is polynomially bounded and computable in polynomial-time regarding its input. In this paper, we need a notion of the negligibility of a two-argument function $\mu : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. We say such a μ is negligible if for every nice function $n : \mathbb{N} \to \mathbb{N}$, the function $\mu_n : \mathbb{N} \to \mathbb{N}$ where $\mu_n(k) = \mu(k, n(k))$ is negligible.

2.1 Trapdoor commitment scheme

We review a few concepts related to trapdoor commitment schemes [11, 10]. A trapdoor commitment scheme (also known as a chameleon hash function [15, 2]) is a function which is associated with a pair of a public key and a private key. The main property of a function which we want is the collision-resistance: it is infeasible to find two inputs that are mapped to the same value without the trapdoor. On the other hand, the knowledge of the trapdoor suffices to find collisions easily.

We follow the notation [11]. A trapdoor commitment scheme is defined by a triplet $\mathcal{TC} = (\mathcal{K}, \mathcal{C}, \mathcal{D})$ of polynomial-time algorithms:

- The key generation algorithm $\mathcal{K}$ is a randomized algorithm which takes as input 1^k, where $k \in \mathbb{N}$ is the security parameter and outputs a pair of a public key and a private key (pk, sk).
- The commitment function $\mathcal{C}$ is a deterministic algorithm which takes as input pk, m, and r, where pk is a public key, m is a message and r is an auxiliary parameter, and outputs a committed value c.
- The collision-finding function $\mathcal{D}$ is a deterministic algorithm which takes as input sk, m, r, and m', where pk is a secret key, m, m' are messages and r is an auxiliary parameter, and outputs an auxiliary parameter r' such that $\mathcal{C}(pk, m, r) = \mathcal{C}(pk, m', r')$.

There are various security properties on a trapdoor commitment scheme, however, we only require the collision-resistance. We define formally the advantage of an adversary $\mathtt{A}$ in defeating it by:

$$\begin{aligned}\mathbf{Adv}^{col}_{\mathcal{TC},\mathtt{A}}(k) \quad = \quad & \Pr[(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k); ((m, r), (m', r')) \xleftarrow{\$} \mathtt{A}(1^k, pk) : \\ & \mathcal{C}(pk, m, r) = \mathcal{C}(pk, m', r') \wedge (m, r) \neq (m', r')].\end{aligned}$$

Definition 1 ((t, ϵ)-collision resistance of trapdoor commitment schemes). *We say that a trapdoor commitment scheme $\mathcal{TC}$ is (t, ϵ)-collision resistant if for any polynomial-time adversary $\mathtt{A}$ that runs in time t, the function $\mathbf{Adv}^{col}_{\mathcal{TC},\mathtt{A}}(\cdot)$ is at most $\epsilon(\cdot)$.*

2.2 Hash functions

In this paper, we treat a hash function $\mathcal{H}$ as a function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$, where $\mathcal{H}$ is given at random from a family of functions. We would like to deal with the difficulty with which an adversary is able to find two distinct points in the domain of a hash function that hashes to the same range point. Following [17], we define the advantage of an adversary $\mathtt{A}$ in defeating collision-resistance of the hash function $\mathcal{H}$ by:

$$\mathbf{Adv}^{col}_{\mathcal{H},\mathtt{A}}(k) = \Pr[(x, y) \xleftarrow{\$} \mathtt{A}(1^k) : \mathcal{H}(x) = \mathcal{H}(y) \wedge x \neq y].$$

Definition 2 ((t, ϵ)-collision resistance of hash functions). *We say that a hash function $\mathcal{H}$ is (t, ϵ)-collision resistant if for any polynomial-time adversary $\mathtt{A}$ that runs in time t, the function $\mathbf{Adv}^{col}_{\mathcal{H},\mathtt{A}}(\cdot)$ is at most $\epsilon(\cdot)$.*

2.3 Bilinear groups

We review a few concepts related to bilinear maps. We follow the notation of [9]:

1. G_1 and G_2 are two (multiplicative) cyclic groups of prime order p;
2. g_1 is a generator of G_1 and g_2 is a generator of G_2;
3. ϕ is a computable isomorphism from G_2 to G_1, with $\phi(g_2) = g_1$; and
4. e is a computable bilinear map $e : G_1 \times G_2 \to G_T$ with the following properties:
   - Bilinearity: for all $u \in G_1, v \in G_2$ and $a, b \in \mathbb{Z}, e(u^a, v^b) = e(u, v)^{ab}$
   - Non-degeneracy: $e(g_1, g_2) \neq 1$

Throughout this paper, we consider a bilinear map $e : G_1 \times G_2 \to G_T$ where all groups G_1, G_2, G_T are multiplicative and of prime order p. One could set $G_1 = G_2$. However, we allow for the more general case where $G_1 \neq G_2$ in the same way as [8].

We say that a truple of two groups (G_1, G_2) is a bilinear group pair if the group action in G_1 and G_2, the map ϕ, and the bilinear map e are all efficiently computable.

3 Group signature schemes

3.1 The model and proposed requirements

In the group signature setting introduced by Chaum and Van Heyst [12], there is a group having numerous members and a single manager. Associated to the group is a single signature-verification key *gpk* called the group public key. Each group member i has its own secret signing key based on which it can produce a signature relative to *gpk*. The group manager has a secret key which can revoke the anonymity of the signed message.

Following the notation [4, 16], we define a group signature scheme $\mathcal{GS} = (\texttt{GDg}, \texttt{GDv}, \texttt{GKg}, \texttt{GKv}, \texttt{GSg}, \texttt{GSv}, \texttt{Open})$ of seven polynomial-time algorithms:

- The group domain parameter generation algorithm `GDg` is a randomized algorithm which takes as input 1^k and 1^n, where $k \in \mathbb{N}$ is the security parameter and $n \in \mathbb{N}$ is the group size (i.e. the number of members of the group), and outputs a group domain parameter D to be shared between one or more groups and possibly some state information I required to verify that these parameters satisfy some security requirements. It may be the case that `GDg` simply returns its input, namely 1^k and 1^n.

- The group domain parameter validation algorithm `GDv` is a deterministic algorithm which on input a security parameter k, a group size n, a group domain parameter D, and possibly some state information I, outputs 1 if the group domain parameter conforms to the specified security requirements. Otherwise, it outputs 0.

- The randomized group key generation algorithm `GKg` takes as input a group domain parameter D and returns a tuple $(gpk, gmsk, \mathbf{gsk})$, where *gpk* is a group public key, *gmsk* is a group manager's secret key, and $\mathbf{gsk}$ is an n-vector of keys with $\mathbf{gsk}[i]$ being a secret signing key for the player $i \in [n]$.

- The group public key validation algorithm `GKv` is a deterministic algorithm which on input a group domain parameter D and a group public key *gpk*, outputs 1 if these parameters conform to the specified requirements. Otherwise, it outputs 0.

- The randomized group signing algorithm `GSg` takes as input a group public key *gpk*, a group secret signing key $\mathbf{gsk}[i]$ associated with the group domain parameter D and a message m and returns a signature of m under $\mathbf{gsk}[i]$ ($i \in [n]$).

- The deterministic group signature verification algorithm `GSv` takes as input a group public key *gpk*, a group domain parameter D, a message m, and a candidate signature σ for m and returns 1 if the signature conforms to the specified security requirements. Otherwise, it returns 0.

- The deterministic opening algorithm `Open` takes as input a group manager secret key *gmsk*, a group domain parameter D, a group public key *gpk*, a message m, and a signature σ of m and returns an identity i or the symbol $\perp$ to indicate failure.

Bellare et al. [4] provided three properties that a group signature scheme must satisfy; the correctness, the full-anonymity, and the full-traceability, and showed that all existing security requirements are implied by satisfying the full-anonymity and the full-traceability. We review the properties along with these definitions of a group signature scheme as follows.

Correctness A group signature scheme must satisfy the following correctness requirement: a signature created correctly by the signing algorithm is always valid and recovered the identity of the signer by the opening algorithm.

Definition 3 (correctness). *We say that $\mathcal{GS}$ is correct if for all $k, n \in \mathbb{N}$, $D \in [\mathtt{GDg}(1^k, 1^n)]$, $(gpk, gmsk, \mathbf{gsk}) \in [\mathtt{GKg}(D)]$, $i \in [n]$, and $m \in \{0,1\}^*$,*

$$\mathtt{GSv}(D, gpk, m, \mathtt{GSg}(D, gpk, \mathbf{gsk}[i], m)) = 1 \quad \textit{and} \quad \mathtt{Open}(D, gpk, gmsk, m, \mathtt{GSg}(D, gpk, \mathbf{gsk}[i], m)) = i.$$

Full-anonymity. Informally, the anonymity requires that an adversary not in possession of the group manager's secret key cannot identify the signer of a signature. An adversary $\mathtt{A}$ is given the secret keys of all group members. There are two settings of the adversary, that is, the CCA model [4] and the CPA model [8]. In the CCA model [4], $\mathtt{A}$ is given access to the opening oracle $\mathtt{Open}(D, gpk, gmsk, \cdot, \cdot)$, which when queried with a message m and signature σ, answers with $\mathtt{Open}(D, gpk, gmsk, m, \sigma)$. On the other hand, in the CPA model [8], $\mathtt{A}$ is not allowed to access the oracle. Bellare et al. [4] associated $\mathtt{A}$ with the following experiment.

Experiment $\mathbf{Exp}^{atk\text{-}anon\text{-}b}_{\mathcal{GS},\mathtt{A}_{atk}}(k, n)$
$D \xleftarrow{\$} \mathtt{GDg}(1^k, 1^n)$; $(gpk, gmsk, \mathbf{gsk}) \xleftarrow{\$} \mathtt{GKg}(D)$
$(\mathrm{St}, i_0, i_1, m) \xleftarrow{\$} \mathtt{A}_{atk}(\mathtt{choose}, D, gpk, \mathbf{gsk})$
$\sigma \xleftarrow{\$} \mathtt{GSg}(D, gpk, \mathbf{gsk}[i_b], m)$
$d \xleftarrow{\$} \mathtt{A}_{atk}(\mathtt{guess}, \mathrm{St}, \sigma)$;
If $\mathtt{A}_{atk}$ did not query its oracle with (m, σ) in the guess stage then return d EndIf
Return 0.

Let us now proceed the formalization. For any group signature scheme $\mathcal{GS} = (\mathtt{GDg}, \mathtt{GDv}, \mathtt{GKg}, \mathtt{GKv}, \mathtt{GSg}, \mathtt{GSv}, \mathtt{Op}$ we associate an adversary $\mathtt{A}_{atk}$ ($atk \in \{cpa, cca\}$) and a bit b with the experiment. Here, $\mathtt{A}_{atk}$ is an adversary that functions in two stages, the choosing stage and the guessing stage. In the choosing stage, $\mathtt{A}_{atk}$ takes as input the group members' secret keys $\mathbf{gsk}$, the group public key gpk, and the group domain parameter D. During this stage, $\mathtt{A}_{cca}$ can also query the opening oracle $\mathtt{Open}(D, gpk, gmsk, \cdot, \cdot)$ on group signatures of his choice. On the other hand, $\mathtt{A}_{cpa}$ can not query the opening oracle. At the end of this stage, $\mathtt{A}_{atk}$ is required to output two valid identities $1 \le i_0, i_1 \le n$, and a message m. The adversary also outputs some state information to be used in the second stage of the attack. In the second stage, the adversary is given the state information and a signature σ on m produced using the secret key of the user i_b. $\mathtt{A}_{cca}$ can still query the opening oracle except for the challenge signature σ. The goal of $\mathtt{A}_{atk}$ is to distinguish between the two secret keys on which the signature was created. We denote the advantage of the adversary $\mathtt{A}_{atk}$ in breaking the atk-full-anonymity of $\mathcal{GS}$ by

$$\mathbf{Adv}^{atk\text{-}anon}_{\mathcal{GS},\mathtt{A}_{atk}}(k, n) = |\Pr[\mathbf{Exp}^{atk\text{-}anon\text{-}1}_{\mathcal{GS},\mathtt{A}_{atk}}(k, n) = 1] - \Pr[\mathbf{Exp}^{atk\text{-}anon\text{-}0}_{\mathcal{GS},\mathtt{A}_{atk}}(k, n) = 1]|.$$

Definition 4 (Full-anonymity). *For $atk \in \{CPA, CCA\}$, we say that a group signature scheme is atk-fully-anonymous if for any polynomial-time adversary $\mathtt{A}$, the two-argument function $\mathbf{Adv}^{atk\text{-}anon}_{\mathcal{GS},\mathtt{A}_{atk}}(\cdot, \cdot)$ is negligible.*

Although the full-anonymity in the CPA setting is weaker than that in the CCA setting, Boneh, Boyen and Shacham [8] mentioned that it is sufficient that a group signature scheme satisfies the fully-traceability and the CPA-fully-anonymity in order to achieve the traditional security requirements, which are the unforgeability, the exculpability, the traceability, the coalition resistance, the framing resistance, the anonymity and the unlinkability.

Full-traceability. In the case of misuse, the signer anonymity can be revoked by the group manager. We require that no colluding set S of group members (even consisting of the entire group, and even being in possession of the secret key to open signatures) can create signatures that cannot be either opened or traced back to any coalition of the members. We remark that

giving the opening key to the adversary models the compromise of the group manager's key by the adversary, not the corruption of the group manager. We call this requirement the full-traceability. Bellare et al. [4] associated $\mathtt{A}$ with the following experiment.

Experiment $\mathbf{Exp}^{trace}_{\mathcal{GS},\mathtt{A}}(k,n)$
$D \xleftarrow{\$} \mathtt{GDg}(1^k,1^n)$; $(gpk,gmsk,\mathbf{gsk}) \xleftarrow{\$} \mathtt{GKg}(D)$
$St \leftarrow (D,gpk,gmsk)$; $C \leftarrow \emptyset$; $K \leftarrow \epsilon$; Cont $\leftarrow$ true;
while (Cont = true) do
$(\mathrm{Cont},\mathrm{St},j) \xleftarrow{\$} \mathtt{A}^{\mathtt{GSg}(D,gpk,\mathbf{gsk}[\cdot],\cdot)}(\mathtt{choose},\mathrm{St},K)$;
If Cont = true then $C \leftarrow C \cup \{j\}$; $K \leftarrow \mathbf{gsk}[j]$ EndIf
Endwhile
$(m,\sigma) \xleftarrow{\$} \mathtt{A}^{\mathtt{GSg}(D,gpk,\mathbf{gsk}[\cdot],\cdot)}(\mathtt{guess},\mathrm{St})$;
If $\mathtt{GSv}(D,gpk,m,\sigma) = 0$ then return 0
If $\mathtt{Open}(D,gpk,gmsk,m,\sigma) = \perp$ then return 1
If there exists $i \in [n]$ such that all the following conditions are true then return 1 else return 0 :
1. $\mathtt{Open}(D,gpk,gmsk,m,\sigma) = i$
2. $i \notin C$
3. (i,m) was not queried by $\mathtt{A}$ to $\mathtt{GSg}(D,gpk,\mathbf{gsk}[\cdot],\cdot)$

Here, an adversary $\mathtt{A}$ runs in two stages, the choosing stage and the guessing stage. On input the group domain parameter D, the group public key gpk, and the secret key of the group manager $gmsk$, the adversary starts its attack by adaptively corrupting a set C of group members and querying the signing oracle $\mathtt{GSg}(D,gpk,\mathbf{gsk}[\cdot],\cdot)$. At the end of the choosing stage, the set C contains the identities of the corrupted members. In the guessing stage, the adversary attempts to produce a forgery (m,σ), and the experiment returns 1 if σ is a valid group signature on m and the opening algorithm returns $\perp$ or some valid user identity i such that $i \notin C$. Otherwise, the experiment returns 0. We say $\mathtt{A}$ wins if the experiment returns 1. We define the advantage of the adversary $\mathtt{A}$ in defeating full-traceability of a group signature scheme $\mathcal{GS}$ by:

$$\mathbf{Adv}^{trace}_{\mathcal{GS},\mathtt{A}}(k,n) = \Pr[\mathbf{Exp}^{trace}_{\mathcal{GS},\mathtt{A}}(k,n) = 1].$$

Definition 5 (Full-traceability). *We say that $\mathcal{GS}$ is fully-traceable if for any polynomial-time adversary $\mathtt{A}$, the two-argument function $\mathbf{Adv}^{trace}_{\mathcal{GS},\mathtt{A}}(\cdot,\cdot)$ is negligible.*

3.2 Key substitution attacks

We define two security conditions of group signature schemes against two key substitution attack models. We call one "key substitution attack 1" and the other "key substitution attack 2". It is easy to prove that if a group signature scheme $\mathcal{GS}$ is secure against key substitution attack 2, then $\mathcal{GS}$ is secure against key substitution attack 1. At the end of this section, we prove it. In the section 5, we also prove that the security requirement against key substitution attack 2 is truly stronger than that against key substitution attack 1.

Key substitution attack 1 Informally, if a signature scheme satisfies this property, an adversary can not create another group public key on which a message and a signature pair can be valid which is also valid on the original group group public key. Following the descriptions of the key substitution attacks [16, 21] and a group signature scheme [4], we define an experiment which performs the key substitution attack 1 against a group signature scheme.

Experiment $\mathbf{Exp}^{ksa1}_{\mathcal{GS},\mathtt{A}}(k,n)$
$D \xleftarrow{\$} \mathtt{GDg}(1^k,1^n)$; $(gpk,gmsk,\mathbf{gsk}) \xleftarrow{\$} \mathtt{GKg}(D)$;

$\mathrm{St} \leftarrow (D, gpk)$; $L \leftarrow \emptyset$; $\sigma \leftarrow \epsilon$; $\mathrm{Cont} \leftarrow \mathrm{true}$;
while $(\mathrm{Cont} = \mathrm{true})$ do
$(\mathrm{Cont}, \mathrm{St}, m, j) \xleftarrow{\$} \mathtt{A}(\mathtt{choose}, \mathrm{St}, \sigma)$;
If $\mathrm{Cont} = \mathrm{true}$ then $\sigma \xleftarrow{\$} \mathtt{GSg}(D, gpk, \mathbf{gsk}[j], m)$; $L \leftarrow L \cup \{(m, \sigma)\}$ EndIf
Endwhile
$\overline{gpk} \xleftarrow{\$} \mathtt{A}(\mathtt{guess}, \mathrm{St})$;
If all the following conditions are true then return 1 else return 0 :
1. there exists $(m, \sigma) \in L$ such that $\mathtt{GSv}(D, \overline{gpk}, m, \sigma) = 1$
2. $\mathtt{GKv}(D, \overline{gpk}) = 1$
3. $gpk \neq \overline{gpk}$

We formally define the security against this attack. Here, an adversary $\mathtt{A}$ runs in two stages, the choosing stage and the guessing stage. On the input of the group public key gpk, the adversary adaptively queries a group signature generated by an arbitrary group member i for any message in the choosing stage. If Cont is true then we regard an output $(\mathrm{Cont}, \mathrm{St}, m, j)$ in this stage as a query (m, j) to the signing oracle $\mathtt{GSg}(D, gpk, \mathbf{gsk}[\cdot], \cdot)$. The list L records pairs of messages and signatures, where the messages are queried from $\mathtt{GSg}(D, gpk, \mathbf{gsk}[\cdot], \cdot)$ by $\mathtt{A}$ during the attack and the signatures are the corresponding answers. In the guessing stage, $\mathtt{A}$ attempts to produce a forged key $\overline{gpk}$, and the experiment returns 1 if $\mathtt{A}$'s output satisfies all the following winning conditions: there exists $(m, \sigma) \in L$ such that σ is valid for m on $\overline{gpk}$, $\overline{gpk}$ is valid group public key regarding the group domain parameter D, and gpk and $\overline{gpk}$ are different group public keys each other. We say $\mathtt{A}$ wins if the experiment returns 1. We denote by

$$\mathbf{Adv}^{ksa1}_{\mathcal{GS},\mathtt{A}}(k, n) = \Pr[\mathbf{Exp}^{ksa1}_{\mathcal{GS},\mathtt{A}}(k, n) = 1],$$

the advantage of $\mathtt{A}$ in breaking the security against key substitution attack 1 of $\mathcal{GS}$. This probability is taken over $\mathtt{GDg}$, $\mathtt{GKg}$, $\mathtt{GSg}$, and $\mathtt{A}$.

Definition 6 (Security against key substitution attack 1). *We say that a group signature scheme is secure against the key substitution attack 1 if for any polynomial-time adversary* $\mathtt{A}$*, the two-argument function* $\mathbf{Adv}^{ksa1}_{\mathcal{GS},\mathtt{A}}(\cdot, \cdot)$ *is negligible.*

Definition 7 ((t, q_s, ϵ)-secure against key substitution attack 1). *We say that a group signature scheme $\mathcal{GS}$ is (t, q_s, ϵ)-secure against key substitution attack 1 if for any polynomial-time adversary* $\mathtt{A}$ *which queries to the signing oracle at most q_s times and runs in time t, the two-argument function* $\mathbf{Adv}^{ksa1}_{\mathcal{GS},\mathtt{A}}(\cdot, \cdot)$ *is at most* $\epsilon(\cdot, \cdot)$.

Key substitution attack 2 In the group signature scheme model, the following scenario can be considerable; a signature σ on a message m was created by a member of the group 1. Another member of group 1 may want σ not to be published. Then he creates another group 2 where (m, σ) is valid for the group 2 and insists that σ is created by a member of the group 2. To avoid this scenario, we define the stronger security property as follows. To capture the possibility of an adversary $\mathtt{A}$ colluding with group members and a group manager, we give $\mathtt{A}$ all group members' secret keys and a group manager's secret key. We remark that giving the opening key to the adversary models the compromise of the group manager's key by the adversary, not the corruption of the group manager, since group signature scheme models do not consider it. We associate $\mathtt{A}$ with the following experiment.

Experiment $\mathbf{Exp}^{ksa2}_{\mathcal{GS},\mathtt{A}}(k, n)$
$D \xleftarrow{\$} \mathtt{GDg}(1^k, 1^n)$;
$(gpk, gmsk, \mathbf{gsk}) \xleftarrow{\$} \mathtt{GKg}(D)$;

$(\overline{gpk}, m, \sigma) \xleftarrow{\$} \mathtt{A}(D, gpk, gmsk, \mathbf{gsk})$;
If the following conditions are true then return 1 else return 0 :

1. $\mathtt{GSv}(D, gpk, m, \sigma) = 1$
2. $\mathtt{GSv}(D, \overline{gpk}, m, \sigma) = 1$
3. $\mathtt{GKv}(D, \overline{gpk}) = 1$
4. $gpk \neq \overline{gpk}$

In the same way to the former security, we formally define the security condition against the key substitution attack 2 using the experiment. A takes input of a group public key gpk, a group manager's secret key $gmsk$, and users' secret keys **gsk** , and produces a pair $(\overline{gpk}, m, \sigma)$ where $\overline{gpk}$ is another group public key, m is a message, and σ is a group signature. The experiment returns 1 if A's output satisfies all the following winning conditions: σ is valid for m on both gpk and $\overline{gpk}$, $\overline{gpk}$ is valid group public key regarding the group domain parameter D, and gpk and $\overline{gpk}$ are different group public keys each other. We say A wins if the experiment returns 1. We define the advantage of A in defeating the security against the key substitution attack 2 of the group signature scheme $\mathcal{GS}$ by:

$$\mathbf{Adv}^{ksa2}_{\mathcal{GS},\mathtt{A}}(k, n) = \Pr[\mathbf{Exp}^{ksa2}_{\mathcal{GS},\mathtt{A}}(k, n) = 1].$$

Definition 8 (Security against key substitution attack 2). *We say that a group signature scheme is secure against the key substitution attack 2 if for any polynomial-time adversary* A*, the two-argument function* $\mathbf{Adv}^{ksa2}_{\mathcal{GS},\mathtt{A}}(\cdot,\cdot)$ *is negligible.*

Definition 9 ((t,ϵ)-secure against key substitution attack 2). *We say that a group signature scheme $\mathcal{GS}$ is (t,ϵ)-secure against key substitution attack 2 if for any polynomial-time adversary* A *such that runs in time t, the two-argument function* $\mathbf{Adv}^{ksa2}_{\mathcal{GS},\mathtt{A}}(\cdot,\cdot)$ *is at most $\epsilon(\cdot,\cdot)$.*

Lemma 10. *Let $\mathcal{GS}$ be a group signature scheme. If $\mathcal{GS}$ is (t',ϵ')-secure against key substitution attack 2, then $\mathcal{GS}$ is (t, q_s, ϵ)-secure against key substitution attack 1 where $t' = t + O(q_s)$, $\epsilon' = \frac{\epsilon}{q_s}$, and q_s is an arbitrary number.*

Proof. Given an algorithm A that breaks the (t, q_s, ϵ)-security of $\mathcal{GS}$ against key substitution attack 1, we construct an algorithm B which breaks the (t', ϵ')-security of $\mathcal{GS}$ against key substitution attack 2.

B is given a group domain parameter D, a group public key gpk, a group manager's secret key $gmsk$, and users' secret key **gsk**. B then provides A the group domain parameter D and the group public key gpk.

When A queries (i, M) to A's signing oracle $\mathtt{GSg}(D, gpk, \mathbf{gsk}[\cdot], \cdot)$, B computes correctly a signature σ for m using $\mathbf{gsk}[i]$, responds σ as the response from A's signing oracle, and records a pair (M, σ) to the list L.

In the random oracle model, A can query the random oracle. When A queries $(M, \mathbf{T}, \mathbf{R})$ to A's random oracle $\mathcal{H}_A$, B queries $(M, \mathbf{T}, \mathbf{R})$ to B's random oracle $\mathcal{H}_B$, receives the response h from $\mathcal{H}_B$, and responds h as the response from $\mathcal{H}_A$.

A outputs a group public key $\overline{gpk}$. B takes (M, σ) at random from the list L, and outputs a pair $(\overline{gpk}, M, \sigma)$. If B responds correctly A's signing query, then $(\overline{gpk}, M, \sigma)$ satisfies the first winning condition of $\mathbf{Exp}^{ksa2}_{\mathcal{GS},\mathtt{A}}$: $\mathtt{GSv}(D, gpk, M, \sigma) = 1$. If $\overline{gpk}$ satisfies the first winning condition of $\mathbf{Exp}^{ksa1}_{\mathcal{GS},\mathtt{A}}$: $\exists (m, \sigma) \in L$ s.t. $\mathtt{GSv}(D, \overline{gpk}, m, \sigma) = 1$ with some probability ϵ_0, then $(\overline{gpk}, M, \sigma)$ satisfies the second winning condition of $\mathbf{Exp}^{ksa2}_{\mathcal{GS},\mathtt{A}}$: $\mathtt{GSv}(D, \overline{gpk}, m, \sigma) = 1$ with some probability $\frac{\epsilon_0}{q_s}$. If $\overline{gpk}$ satisfies the second and third winning conditions of $\mathbf{Exp}^{ksa1}_{\mathcal{GS},\mathtt{A}}$, then $(\overline{gpk}, M, \sigma)$ satisfies the third and forth winning conditions of $\mathbf{Exp}^{ksa2}_{\mathcal{GS},\mathtt{A}}$, respectively. That is, if A's output $\overline{gpk}$ satisfies the winning conditions of $\mathbf{Exp}^{ksa1}_{\mathcal{GS},\mathtt{A}}$ with some probability ϵ_0 then B's output $(\overline{gpk}, M, \sigma)$ satisfies the winning conditions of $\mathbf{Exp}^{ksa2}_{\mathcal{GS},\mathtt{A}}$ with some probability $\frac{\epsilon_0}{q_s}$.

Since the keys given to $\mathtt{A}$ and the answers of $\mathtt{A}$'s queries are all valid, $\mathtt{B}$ simulates $\mathtt{A}$'s enviroment with proper distributions. Therefore, $\mathtt{A}$ outputs gpk that satisfies the winning conditions of $\mathbf{Exp}^{ksa1}_{\mathcal{GS},\mathtt{A}}$ with advantage ϵ and $\mathtt{B}$ breaks the security against key substitution attack 2 of $\mathcal{GS}$ with at least $\frac{\epsilon}{q_s}$.

$\mathtt{B}$'s running time is the sum of $\mathtt{A}$'s running time and the time to answer $\mathtt{A}$'s queries to the signing oracle and to take (M, σ) from the list L. It takes constant time to answer each query and to take (M, σ), and the numbers of queries to the signing oracle is at most q_s. If $\mathtt{A}$ runs in time t, $\mathtt{B}$ runs in time $t' = t + O(q_s)$. □

4 Boneh, Boyen and Shacham's short group signatures

In this section, we analyze the security of a short group signature scheme proposed by Boneh, Boyen and Shacham [8] against key substitution attacks. The full-anonymity and the full-traceability of this scheme are based on the Strong Diffe-Hellman Assumption (SDH) in groups with a bilinear map and the Decision Linear Assumption (LA), where the LA holds on generic bilinear groups in the sense of Shorp [18]. In their proof, the hash function $\mathcal{H}$ is treated as a random oracle. We denote their short group signature scheme as $\mathcal{GS}_0 = (\mathtt{GDg_0}, \mathtt{GDv_0}, \mathtt{GKg_0}, \mathtt{GKv_0}, \mathtt{GSg_0}, \mathtt{GSv_0}, \mathtt{Open_0})$. First, we show the insecurity of $\mathcal{GS}_0$. This means that satisfying the full-anonymity and the full-traceability is not sufficient to be secure against the attack. We also propose the modified scheme and prove that the modified scheme is correct, fully-anonymous, full-traceable, and secure against key substitution attack 2.

4.1 Insecurity of Boneh, Boyen and Shacham's scheme

As mentioned above, the short group signature scheme $\mathcal{GS}_0$ is fully-traceable and fully-anonymous on the Strong Diffe-Hellman assumption and the Decision Linear Assumption in the random oracle model [4]. Nevertheless, we can construct an algorithm $\mathtt{A}$ which breaks this scheme's security against the key substitution attack 1 as follows. $\mathtt{A}$ takes as input a group domain parameter D and a group public key $gpk = (g_1, g_2, h, u, v, w)$ and computes another group public key $\overline{gpk}$ as follows:

1. It queries a message (i, m) to the signing oracle $\mathtt{GSg_0}(\mathtt{D}, gpk, \mathbf{gsk}[\cdot], \cdot)$ and receives a signature $\sigma = (\mathbf{T}, c, \mathbf{s})$, where $\mathbf{T} = (T_1, T_2, T_3)$ and $\mathbf{s} = (s_\alpha, s_\beta, s_x, s_{\delta 1}, s_{\delta 2})$.
2. It sets $e_1 = s_x$, $e_2 = -s_\alpha - s_\beta$, and $e_3 = -s_{\delta_1} - s_{\delta_2}$.
3. It chooses $X \in \mathbb{Z}_p^*$ and sets $Y = c(X-1)e_2^{-1}$ and $Z = e_3(1-X)X^{-1}e_2^{-1}$, where $X \neq 1$.
4. It sets $\overline{h} = h^X T_3^Y$ and $\overline{w} = g_2^Z w^{X^{-1}}$.
5. It outputs $\overline{gpk} = (g_1, g_2, \overline{h}, u, v, \overline{w})$.

We show $\mathtt{A}$'s output $\overline{gpk}$ satisfies the winning conditions of the experiment $\mathbf{Exp}^{ksa1}_{\mathcal{GS}_0,\mathtt{A}}$. Using the definition of $\mathtt{GKv_0}$ in $\mathcal{GS}_0$ and the construction of $\mathtt{A}$, we can see that $\mathtt{A}$'s output $\overline{gpk}$ satisfies the second and third winning conditions, respectively. Let $\tilde{\mathbf{R}} = (\tilde{R_1}, \tilde{R_2}, \tilde{R_3}, \tilde{R_4}, \tilde{R_5})$ be computed using gpk and σ and $\overline{\mathbf{R}} = (\overline{R_1}, \overline{R_2}, \overline{R_3}, \overline{R_4}, \overline{R_5})$ computed using $\overline{gpk}$ and σ. If $\overline{\mathbf{R}} = \tilde{\mathbf{R}}$ then $\mathcal{H}(m, \mathbf{T}, \overline{\mathbf{R}}) = \mathcal{H}(m, \mathbf{T}, \tilde{\mathbf{R}})$, that is, $\mathtt{GSv_0}(D, gpk, m, \sigma) = \mathtt{GSv_0}(D, \overline{gpk}, m, \sigma)$. Therefore, to prove that it satisfies the first condition $\mathtt{GSv_0}(D, \overline{gpk}, m, \sigma) = 1$, it is sufficient to show that $\overline{\mathbf{R}} = \tilde{\mathbf{R}}$. gpk and $\overline{gpk}$ are identical except for h and x and R_1, R_2, R_4, R_5 are computed without h and w.

Therefore, we can see $\tilde{R_1} = \overline{R_1}$, $\tilde{R_2} = \overline{R_2}$, $\tilde{R_4} = \overline{R_4}$, and $\tilde{R_5} = \overline{R_5}$. We can see $\tilde{R_3} = \overline{R_3}$ as follows:

$$\begin{aligned}
\overline{R_3} &= e(T_3, g_2)^{s_x} \cdot e(\overline{h}, \overline{w})^{-s_\alpha - s_\beta} \cdot e(\overline{h}, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot (\frac{e(T_3, \overline{w})}{e(g_1, g_2)})^c \\
&= e(T_3, g_2)^{e_1} \cdot e(\overline{h}, \overline{w})^{e_2} \cdot e(\overline{h}, g_2)^{e_3} \cdot e(T_3, \overline{w})^c \cdot e(g_1, g_2)^{-c} \\
&= e(T_3, g_2)^{e_1} \cdot e(h^X, g_2^Z)^{e_2} \cdot e(T_3^Y, g_2^Z)^{e_2} \cdot e(h^X, w^{X^{-1}})^{e_2} \cdot e(T_3^Y, w^{X^{-1}})^{e_2} \cdot e(h^X, g_2)^{e_3} \cdot e(T_3^Y, g_2)^{e_3} \\
&\quad \cdot e(T_3, g_2^Z)^c \cdot e(T_3, w^{X^{-1}})^c \cdot e(g_1, g_2)^{-c} \\
&= e(T_3, g_2)^{e_1 + YZe_2 + Ye_3 + Zc} \cdot e(h, w)^{XX^{-1}e_2} \cdot e(h, g_2)^{XZe_2 + Xe_3} \cdot e(T_3, w)^{YX^{-1}e_2 + X^{-1}c} \cdot e(g_1, g_2)^{-c} \\
&= e(T_3, g_2)^{e_1} \cdot e(h, w)^{e_2} \cdot e(h, g_2)^{e_3} \cdot (\frac{e(T_3, w)}{e(g_1, g_2)})^c \\
&= \tilde{R_3}.
\end{aligned}$$

Note that, since Y and Z are computed as $Y = c(X-1)e_2^{-1}$ and $Z = e_3(1-X)X^{-1}e_2^{-1}$ by $\mathtt{A}$, respectively,

$$\begin{aligned}
&e_1 + YZe_2 + Ye_3 + Zc \\
&\quad = e_1 + \{c(X-1)e_2^{-1} \cdot e_3(1-X)X^{-1}e_2^{-1} \cdot e_2\} + \{c(X-1)e_2^{-1} \cdot e_3\} + e_3(1-X)X^{-1}e_2^{-1} \cdot c \\
&\quad = e_1 + ce_2^{-1}e_3\{X^{-1}(X-1)(1-X) + (X-1) + (1-X)X^{-1}\} \\
&\quad = e_1, \\
&XZe_2 + Xe_3 \\
&\quad = X \cdot e_3(1-X)X^{-1}e_2^{-1} \cdot e_2 + Xe_3 \\
&\quad = e_3, \\
&YX^{-1}e_2 + X^{-1}c \\
&\quad = c(X-1)e_2^{-1} \cdot X^{-1}e_2 + X^{-1}c \\
&\quad = c.
\end{aligned}$$

Hence, $\mathtt{A}$ breaks the security against the key substitution attack 1.

Lemma 11. *The Short Signature Scheme $\mathcal{GS}_0$ is not secure against the key substitution attack 1.*

4.2 Modified short signature scheme

The original short signature scheme $\mathcal{GS}_0$ is not secure against the key substitution attack 1. To achieve this security, we modify the scheme.

In this paper, we describe the following modified group signature scheme as $\mathcal{GS}_1 = (\mathtt{GDg_1}, \mathtt{GDv_1}, \mathtt{GKg_1}, \mathtt{GKv_1}, \mathtt{GS}$ We construct $\mathcal{GS}_1$ using $\mathcal{GS}_0$ as follows.

- $\mathtt{GDg_1}$, $\mathtt{GDv_1}$, $\mathtt{GKg_1}$, and $\mathtt{GKv_1}$ are identical to those in $\mathcal{GS}_0$.
- $\mathtt{GSg_1}(D, gpk, \mathbf{gsk}[i], M) = \mathtt{GSg_0}(D, gpk, \mathbf{gsk}[i], M||gpk)$.
- $\mathtt{GSv_1}(D, gpk, M, \sigma) = \mathtt{GSv_0}(D, gpk, M||gpk, \sigma)$.
- $\mathtt{Open_1}(D, gpk, gmsk, M, \sigma) = \mathtt{Open_0}(D, gpk, gmsk, M||gpk, \sigma)$.

We show $\mathcal{GS}_1$ is correct, fully-traceable, and fully-anonymous if $\mathcal{GS}_0$ are satisfies these security properties (Lemma 12, Lemma 14, Lemma 16).

Lemma 12. *If $\mathcal{GS}_0$ is correct, then $\mathcal{GS}_1$ is correct.*

Proof. We assume that $\mathcal{GS}_0$ is correct. Then, for all $k, n \in \mathbb{N}$, $D \in [\mathtt{GDg}_0(1^k, 1^n)]$, $(gpk, gmsk, \mathbf{gsk}) \in [\mathtt{GKg}_0(D)]$, $i \in [n]$, and $m \in \{0,1\}^*$,

$$\mathtt{GSv}_0(D, gpk, m, \mathtt{GSg}_0(D, gpk, \mathbf{gsk}[i], m)) = 1 \quad \textit{and} \quad \mathtt{Open}_0(D, gpk, gmsk, m, \mathtt{GSg}_0(D, gpk, \mathbf{gsk}[i], m)) = i.$$

Hence, for all $k, n \in \mathbb{N}$, $D \in [\mathtt{GDg}_1(1^k, 1^n)]$, $(gpk, gmsk, \mathbf{gsk}) \in [\mathtt{GKg}_1(D)]$, $i \in [n]$, and $m \in \{0,1\}^*$,

$$\begin{aligned} &\mathtt{GSv}_1(D, gpk, m, \mathtt{GSg}_1(D, gpk, \mathbf{gsk}[i], m) \\ &= \mathtt{GSv}_0(D, gpk, m||gpk, \mathtt{GSg}_0(D, gpk, \mathbf{gsk}[i], m||gpk)) = 1 \\ &\mathtt{Open}_1(D, gpk, gmsk, m, \mathtt{GSg}_1(D, gpk, \mathbf{gsk}[i], m)) \\ &= \mathtt{Open}_0(D, gpk, gmsk, m||gpk, \mathtt{GSg}_0(D, gpk, \mathbf{gsk}[i], m||gpk)) = i. \end{aligned}$$

This means that the modified group signature scheme $\mathcal{GS}_1$ is correct. □

Definition 13 ((t, q_H, q_s, ϵ)-full-traceability). *In the random oracle model, we say that a group signature scheme $\mathcal{GS}$ is (t, q_H, q_s, ϵ)-fully-traceable if for any polynomial-time adversary $\mathtt{A}$ which queries to the random oracle at most q_H times and to the signing oracle at most q_s times in running time t, the two-argument function $\mathbf{Adv}^{trace}_{\mathcal{GS},\mathtt{A}}(\cdot,\cdot)$ is at most $\epsilon(\cdot,\cdot)$.*

Lemma 14. *If $\mathcal{GS}_0$ is $(t', q'_H, q'_s, \epsilon')$-fully-traceable, then $\mathcal{GS}_1$ is (t, q_H, q_s, ϵ)-fully-traceable, where $t' = t + O(q_s) + O(q_H)$, $q'_H = q_H$, $q'_s = q_s$, and $\epsilon' = \epsilon$ in the random oracle model.*

Proof. Given an algorithm $\mathtt{A}$ that breaks the (t, q_H, q_s, ϵ)-fully-traceability of $\mathcal{GS}_1$, we construct an algorithm $\mathtt{B}$ that breaks the $(t', q'_H, q'_s, \epsilon')$-fully-traceability of the original group signature scheme.

$\mathtt{B}$ is given a group domain parameter D, a group public key gpk, and a group manager's secret key $gmsk$. $\mathtt{B}$ then provides $\mathtt{A}$ the group domain parameter D, the group public key gpk, and the group manager's secret keys $gmsk$.

At any time, $\mathtt{A}$ can query the random oracle $\mathcal{H}$. When $\mathtt{A}$ queries $(M, \mathbf{T}, \mathbf{R})$ to $\mathtt{A}$'s random oracle $\mathcal{H}_A$, $\mathtt{B}$ queries $(M, \mathbf{T}, \mathbf{R})$ to $\mathtt{B}$'s random oracle $\mathcal{H}_B$, receives the response h from $\mathcal{H}_B$, and responds h as the response from $\mathcal{H}_A$.

When $\mathtt{A}$ queries (i, M) to $\mathtt{A}$'s signing oracle $\mathtt{GSg}_1(D, gpk, \mathbf{gsk}[\cdot], \cdot)$, $\mathtt{B}$ queries $(i, M||gpk)$ to $\mathtt{B}$'s signing oracle $\mathtt{GSg}_0(D, gpk, \mathbf{gsk}[\cdot], \cdot)$. $\mathtt{B}$ then receives the response σ from $\mathtt{GSg}_0(D, gpk, \mathbf{gsk}[\cdot], \cdot)$, and responds with σ as the response from $\mathtt{GSg}_1(D, gpk, \mathbf{gsk}[\cdot], \cdot)$.

When $\mathtt{A}$ asks for the private key of a user indexed i, $\mathtt{B}$ asks for the private key $\mathbf{gsk}[i]$ of the user indexed i and responds $\mathbf{gsk}[i]$ to $\mathtt{A}$.

Finally, $\mathtt{A}$ outputs a pair (m, σ). $\mathtt{B}$ returns $(m||gpk, \sigma)$ as the answer of its own challenge. Since $(m||gpk, \sigma)$ satisfies the winning conditions in $\mathbf{Exp}^{trace}_{\mathcal{GS}_0,\mathtt{B}}$ whenever (m, σ) satisfies ones in $\mathbf{Exp}^{trace}_{\mathcal{GS}_1,\mathtt{A}}$, $\mathtt{B}$ wins whenever $\mathtt{A}$ does.

Since the keys given to $\mathtt{A}$ and the answers of $\mathtt{A}$'s queries are all valid, $\mathtt{B}$ simulates $\mathtt{A}$'s enviroment with proper distributions. Therefore, $\mathtt{A}$ breaks the traceability of the group signature σ with the advantage ϵ, and $\mathtt{B}$ can break the traceability of the group signature σ with the same advantage.

$\mathtt{B}$'s running time is the sum of $\mathtt{A}$'s running time and the time to answer $\mathtt{A}$'s queries to the random oracle and the signing oracle. Each query is answered in constant time, and the numbers of queries to the random oracle and the signature oracle are at most q_s, q_H respectively. If $\mathtt{A}$ runs in time t, $\mathtt{B}$ runs in time $t' = t + O(q_s) + O(q_H)$. □

Definition 15 ((t, q_H, ϵ)-atk-full-anonymity). *In the random oracle model, for $atk \in \{CPA, CCA\}$, we say that $\mathcal{GS}$ is (t, q_H, ϵ)-atk-fully-anonymous if for any polynomial-time adversary $\mathtt{A}$ which queries to the random oracle at most q_H times and runs in time t, the two-argument function $\mathbf{Adv}^{atk\text{-}anon}_{\mathcal{GS},\mathtt{A}_{atk}}(\cdot,\cdot)$ is at most $\epsilon(\cdot,\cdot)$.*

Lemma 16. *If $\mathcal{GS}_0$ is (t', q'_H, ϵ')-CPA-fully-anonymous, then $\mathcal{GS}_1$ is (t, q_H, ϵ)-CPA-fully-anonymous, where $t' = t + O(q_H)$, $q'_H = q_H$, and $\epsilon' = \epsilon$.*

Proof. Given an algorithm $\mathtt{A}$ that breaks the (t, q_H, ϵ)-CPA-fully-anonymity of $\mathcal{GS}_1$, we construct an algorithm $\mathtt{B}$ that breaks the (t', q'_H, ϵ')-CPA-fully-anonymity of $\mathcal{GS}_0$.

$\mathtt{B}$ is given a group domain parameter D, a group public key gpk, and users' secret keys $\mathbf{gsk}$. It then provides $\mathtt{A}$ the group domain parameter D, the group public key gpk, and the users' secret keys $\mathbf{gsk}$.

At any time, $\mathtt{A}$ can query the random oracle $\mathcal{H}_A$. When $\mathtt{A}$ queries $(M, \mathbf{T}, \mathbf{R})$ to $\mathtt{A}$'s random oracle $\mathcal{H}_A$, $\mathtt{B}$ queries $(M, \mathbf{T}, \mathbf{R})$ to $\mathtt{B}$'s random oracle $\mathcal{H}_B$, receives the response h from $\mathcal{H}_B$, and responds with h as the response from $\mathcal{H}_A$.

$\mathtt{A}$ requests its full-anonymity challenge by providing two indices i_0 and i_1 and a message M. $\mathtt{B}$ then requests its full-anonymity challenge by providing the two indices i_0 and i_1 and a message $M||gpk$. It is given a group signature σ of user i_b where the bit b is chosen by $\mathtt{B}$'s group signature challenger and responds σ as response from $\mathtt{A}$'s group signature challenger.

Finally, $\mathtt{A}$ outputs a bit d. $\mathtt{B}$ returns this d as the answer to its own challenge. If σ is returned by $\mathtt{B}$ as a group signature by the user i_b, $\mathtt{B}$ also responds σ as a group signature by the user i_b. Therefore, $\mathtt{B}$ answers its challenge correctly whenever $\mathtt{A}$ does.

Since the keys given to $\mathtt{A}$ and the answers to $\mathtt{A}$'s queries are all valid, $\mathtt{B}$ simulates $\mathtt{A}$'s enviroment with proper distributions. Therefore, $\mathtt{A}$ breaks the anonymity of the group signature σ with advantage ϵ, and $\mathtt{B}$ can break the anonymity of the group signature σ with the same advantage.

$\mathtt{B}$'s running time is the sum of $\mathtt{A}$'s running time and the time to answer $\mathtt{A}$'s queries to the random oracle. Each query is answered in constant time, and the numbers of queries to the random oracle is at most q_H. If $\mathtt{A}$ runs in time t, $\mathtt{B}$ runs in time $t' = t + O(q_H)$. □

In Lemma 14 and 16, note that ϵ is negligible whenever ϵ' is, and that $\mathtt{B}$ is polynomial-time adversary whenever $\mathtt{A}$ is since $\mathtt{A}$ and $\mathtt{B}$ take input with the same length. Also note that $\mathtt{B}$ is polynomial-time adversary whenever $\mathtt{A}$ is.

Finally, we prove that the modified group signature scheme is secure against key substitution attack 2, assuming that the hash function is collision resistant.

Lemma 17. *If the hash function $\mathcal{H}$ in $\mathcal{GS}_1$ is (t', ϵ')-collision resistant, then the modified group signature scheme $\mathcal{GS}_1$ is (t, ϵ)-secure against key substitution attack 2, where $t' = t + O(1)$, $\epsilon' = \epsilon$.*

Proof. Given an algorithm $\mathtt{A}$ that breaks the (t, ϵ)-security against key substitution attack 2 of $\mathcal{GS}_1$, we construct an algorithm $\mathtt{B}$ that breaks the (t', ϵ')-collision resistance of the hash function $\mathcal{H}$ in $\mathcal{GS}_1$.

$\mathtt{B}$ is given a hash function's security parameter $1^k = 1^{|p|}$. $\mathtt{B}$ chooses an arbitrary nice function $n(k)$ as a number n of group members and generates a group domain parameter $D \xleftarrow{\$} \mathtt{GDg_1}(1^k, 1^n)$ and a pair $(gpk, gmsk, \mathbf{gsk}) \xleftarrow{\$} \mathtt{GKg_1}(D)$. $\mathtt{B}$ then provides $\mathtt{A}$ the group domain parameter D, the group public key gpk, the group manager's secret key $gmsk$, and the users' secret keys $\mathbf{gsk}$.

Finally, $\mathtt{A}$ outputs a pair $(m, \sigma, \overline{gpk})$. $\mathtt{B}$ parses σ as $(\mathbf{T}, c, \mathbf{s})$, gpk as (g_1, g_2, h, u, v, w), and $\overline{gpk}$ as $(\overline{g_1}, \overline{g_2}, \overline{h}, \overline{u}, \overline{v}, \overline{w})$. $\mathtt{B}$ computes $\tilde{\mathbf{R}} = (\tilde{R_1}, \tilde{R_2}, \tilde{R_3}, \tilde{R_4}, \tilde{R_5})$ and $\overline{\mathbf{R}} = (\overline{R_1}, \overline{R_2}, \overline{R_3}, \overline{R_4}, \overline{R_5})$ as follows:

$$
\begin{aligned}
\tilde{R_1} &= u^{s_\alpha} \cdot T_1^{-c}, \quad & \tilde{R_2} &= v^{s_\beta} \cdot T_2^{-c}, \\
\tilde{R_3} &= e(T_3, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha - s_\beta} \cdot e(h, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot \left(\tfrac{e(T_3, w)}{e(g_1, g_2)}\right)^c, & & \\
\tilde{R_4} &= u^{-s_{\delta_1}} \cdot T_1^{s_x}, \quad & \tilde{R_5} &= v^{-s_{\delta_2}} \cdot T_2^{s_x}, \\
\overline{R_1} &= \overline{u}^{s_\alpha} \cdot T_1^{-c}, \quad & \overline{R_2} &= \overline{v}^{s_\beta} \cdot T_2^{-c}, \\
\overline{R_3} &= e(T_3, g_2)^{s_x} \cdot e(\overline{h}, \overline{w})^{-s_\alpha - s_\beta} \cdot e(\overline{h}, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot \left(\tfrac{e(T_3, \overline{w})}{e(g_1, g_2)}\right)^c, & & \\
\overline{R_4} &= \overline{u}^{-s_{\delta_1}} \cdot T_1^{s_x}, \quad & \overline{R_5} &= \overline{v}^{-s_{\delta_2}} \cdot T_2^{s_x}.
\end{aligned}
\tag{1}
$$

$\mathtt{B}$ then outputs a pair $(x, y) = ((m||gpk, \mathbf{T}, \tilde{\mathbf{R}}), (m||\overline{gpk}, \mathbf{T}, \overline{\mathbf{R}}))$. If $(m, \sigma, \overline{gpk})$ satisfies the first,

second, and forth winning conditions of $\mathbf{Exp}^{ksa2}_{\mathcal{GS}_1,\mathtt{A}}$, respectively, we can see that

$$c = \mathcal{H}(m||gpk, \mathbf{T}, \tilde{\mathbf{R}}) = \mathcal{H}(x),$$
$$c = \mathcal{H}(m||\overline{gpk}, \mathbf{T}, \overline{\mathbf{R}}) = \mathcal{H}(y),$$
$$x = (m||gpk, \mathbf{T}, \tilde{\mathbf{R}}) \neq (m||\overline{gpk}, \mathbf{T}, \overline{\mathbf{R}}) = y,$$

where $\sigma = (\mathbf{T}, c, \mathbf{s})$. Therefore, $\mathtt{B}$ breaks the collision resistance of the hash function whenever $\mathtt{A}$ breaks the security of the key substitution attack 2.

Since the keys given to $\mathtt{A}$ and the answers of $\mathtt{A}$'s queries are all valid, $\mathtt{B}$ simulates $\mathtt{A}$'s enviroment with proper distributions. Therefore, $\mathtt{A}$ breaks the security against the key substitution attack 2 of $\mathcal{GS}_1$ with advantage ϵ, and $\mathtt{B}$ breaks the collision resistance of $\mathcal{H}$ with the same advantage.

$\mathtt{B}$'s running time is the sum of $\mathtt{A}$'s running time and the time to compute $\mathtt{A}$'s input that can be created in constant time. If $\mathtt{A}$ runs in time t, $\mathtt{B}$ runs in time $t + O(1)$. □

Note that ϵ is negligible whenever ϵ' is since n is a nice function on k, and that $\mathtt{B}$ is polynomial-time adversary whenever $\mathtt{A}$ is since $\mathtt{A}$'s input length is polynomial bounded on $\mathtt{B}$'s one.

5 Separation between our proposed security notions

In Section 3.2, we proved that if a group signature scheme $\mathcal{GS}$ is secure the against key substitution attack 2, then $\mathcal{GS}$ is secure against key substitution attack 1. At this point, it is not still clear whether or not there exists a separation between sequrity requirements against key substitution attack 1 and 2. In this section, We show that there exists the separation, that is, there exists a group signature scheme $\mathcal{GS}_2$ using a collision resistant trapdoor commitment, which secure against key substitution attack 1, and not secure against key substitution attack 2.

Let $\mathcal{TC} = (\mathcal{K}, \mathcal{C}, \mathcal{D})$ be a collision resistant trapdoor commitment. Using Boneh, Boyen and Shacham's group signature scheme $\mathcal{GS}_0$ and $\mathcal{TC}$, we construct a group signature scheme $\mathcal{GS}_2 = (\mathtt{GDg}_2, \mathtt{GDv}_2, \mathtt{GKg}_2, \mathtt{GKv}_2, \mathtt{GSg}_2, \mathtt{GSv}_2, \mathtt{Open}_2)$ as follows.

- $\mathtt{GDg}_2(1^k, 1^n)$ performs as follows. Using $\mathtt{GDg}_0$, it generates $D_0 \xleftarrow{\$} \mathtt{GDg}_0(1^k, 1^n)$. It chooses a reasonable security parameter l for the trapdoor commitment such that there exists a constant $\alpha : l = \alpha k$. The group domain parameter is $D = (D_0, 1^l)$.
- $\mathtt{GDv}_2$ is identical to $\mathtt{GDv}_0$.
- $\mathtt{GKg}_2(D)$ performs as follows. Using $\mathtt{GKg}_0$ and $\mathcal{K}$, it parses D as $(D_0, 1^l)$ and generates key pair $(gpk_0, gmsk_0, \mathbf{gsk}_0) \xleftarrow{\$} \mathtt{GKg}_0(D_0)$ and $(pk_{\mathcal{TC}}, sk_{\mathcal{TC}}) \xleftarrow{\$} \mathcal{K}(1^l)$. It chooses $m_{\mathcal{TC}} \in M$ and $r_{\mathcal{TC}} \in R$ at random and computes $c_{\mathcal{TC}} \leftarrow \mathcal{C}(pk_{\mathcal{TC}}, m_{\mathcal{TC}}, r_{\mathcal{TC}})$. The group public key is $gpk = (gpk_0, pk_{\mathcal{TC}}, c_{\mathcal{TC}}, m_{\mathcal{TC}}, r_{\mathcal{TC}})$. The private key of the group manager is $gmsk = (gmsk_0, sk_{\mathcal{TC}})$. The i-th user's secret key is $\mathbf{gsk}[i] = \mathbf{gsk}_0[i]$.
- $\mathtt{GKv}_2(D, gpk)$ performs as follows. It parses gpk as $(gpk_0, pk_{\mathcal{TC}}, c_{\mathcal{TC}}, m_{\mathcal{TC}}, r_{\mathcal{TC}})$ and returns 1 if $c_{\mathcal{TC}} = \mathcal{C}(pk_{\mathcal{TC}}, m_{\mathcal{TC}}, r_{\mathcal{TC}})$. Otherwise, it returns 0.
- $\mathtt{GSg}_2(D, gpk, \mathbf{gsk}[i], M)$ performs as follows. It parses gpk as $(gpk_0, pk_{\mathcal{TC}}, c_{\mathcal{TC}}, m_{\mathcal{TC}}, r_{\mathcal{TC}})$ and D as $(D_0, 1^l)$ and runs $\mathtt{GSg}_0(D_0, gpk, \mathbf{gsk}[i], M||gpk_0||pk_{\mathcal{TC}}||c_{\mathcal{TC}})$.
- $\mathtt{GSv}_2(D, gpk, M, \sigma)$ performs as follows. It parses gpk as $(gpk_0, pk_{\mathcal{TC}}, c_{\mathcal{TC}}, m_{\mathcal{TC}}, r_{\mathcal{TC}})$ and D as $(D_0, 1^l)$ and runs $\mathtt{GSv}_0(D_0, gpk, M||gpk_0||pk_{\mathcal{TC}}||c_{\mathcal{TC}}, \sigma)$.
- $\mathtt{Open}_2(D, gpk, gmsk, M, \sigma)$ performs as follows. It parses gpk as $(gpk_0, pk_{\mathcal{TC}}, c_{\mathcal{TC}}, m_{\mathcal{TC}}, r_{\mathcal{TC}})$ and D as $(D_0, 1^l)$ and runs $\mathtt{Open}_0(D_0, gpk, gmsk, M||gpk_0||pk_{\mathcal{TC}}||c_{\mathcal{TC}}, \sigma)$.

In the same way as Lemma 12, 14, 16, we can say that if $\mathcal{GS}_0$ is correct, fully-traceable and CPA-fully-anonymous, then $\mathcal{GS}_2$ is correct, fully-traceable and CPA-fully-anonymous.

Now, we show that $\mathcal{GS}_2$ is secure against key substitution attack 1 (Lemma 18), and not secure against key substitution attack 2 (Lemma 19).

Lemma 18. *If the hash function $\mathcal{H}$ in $\mathcal{GS}_2$ is (t', ϵ')-collision resistant and the trapdoor commitment scheme $\mathcal{TC}$ in $\mathcal{GS}_2$ is (t'', ϵ'')-collision resistant, then $\mathcal{GS}_2$ is (t, q_s, ϵ)-secure against key substitution attack 1 where $t' = t + O(q_s)$, $t'' = t + O(q_s)$, $\epsilon' = \frac{\epsilon}{2q_s}$, $\epsilon'' = \frac{\epsilon}{2}$, and q_s is an arbitrary number.*

Proof. Given an algorithm $\mathtt{A}$ that breaks the (t, q_s, ϵ)-security against key substitution attack 1 of $\mathcal{GS}_2$, we construct either an algorithm $\mathtt{B}_1$ which breaks the (t', ϵ')-collision resistance of $\mathcal{H}$ in $\mathcal{GS}_2$ or an algorithm $\mathtt{B}_2$ which breaks the (t'', ϵ'')-collision resistance of $\mathcal{TC}$ in $\mathcal{GS}_2$, where $t' = t + O(q_s)$, $t'' = t + O(q_s)$, $\epsilon' = \frac{\epsilon}{2q_s}$, $\epsilon'' = \frac{\epsilon}{2}$, and q_s is an arbitrary number.

On the point of the output, we classify $\mathtt{A}$ into either Type 1 Forger or Type 2 Forger as follows:

- On the input of $gpk = (gpk_0, pk_{\mathcal{TC}}, c_{\mathcal{TC}}, m_{\mathcal{TC}}, r_{\mathcal{TC}})$, $\mathtt{A}$ which classified in this group outputs $\overline{gpk} = (\overline{gpk_0}, \overline{pk_{\mathcal{TC}}}, \overline{c_{\mathcal{TC}}}, \overline{m_{\mathcal{TC}}}, \overline{r_{\mathcal{TC}}})$ such that $(m_{\mathcal{TC}}, r_{\mathcal{TC}}) = (\overline{m_{\mathcal{TC}}}, \overline{r_{\mathcal{TC}}})$. If $\mathtt{A}$ classified into this group, $\mathtt{A}$ is called (t, q_s, ϵ)-Type 1 Forger.
- On the input of $gpk = (gpk_0, pk_{\mathcal{TC}}, c_{\mathcal{TC}}, m_{\mathcal{TC}}, r_{\mathcal{TC}})$, $\mathtt{A}$ which classified in this group outputs $\overline{gpk} = (\overline{gpk_0}, \overline{pk_{\mathcal{TC}}}, \overline{c_{\mathcal{TC}}}, \overline{m_{\mathcal{TC}}}, \overline{r_{\mathcal{TC}}})$ such that $(m_{\mathcal{TC}}, r_{\mathcal{TC}}) \neq (\overline{m_{\mathcal{TC}}}, \overline{r_{\mathcal{TC}}})$. If $\mathtt{A}$ classified into this group, $\mathtt{A}$ is called (t, q_s, ϵ)-Type 2 Forger.

It is easy to see that $\mathtt{A}$ is certainly classified into either Type 1 or Type 2. Therefore, if there exists an adversary $\mathtt{A}$ that breaks the (t, q_s, ϵ)-security against key substitution attack 1 of $\mathcal{GS}_2$, there exists either $(t, q_s, \frac{\epsilon}{2})$-Type 1 Forger $\mathtt{A}_1$ or $(t, q_s, \frac{\epsilon}{2})$-Type 2 Forger $\mathtt{A}_2$. We construct $\mathtt{B}_1$ that breaks the collision resistance of $\mathcal{H}$ using $\mathtt{A}_1$ and $\mathtt{B}_2$ that breaks the collision resistance of $\mathcal{TC}$ using $\mathtt{A}_2$.

First, we construct an algorithm $\mathtt{B}_1$ that breaks the collision resistance of the hash function $\mathcal{H}$. $\mathtt{B}_1$ is given a hash function $\mathcal{H}$'s security parameter $1^k = 1^{|p|}$. $\mathtt{B}_1$ chooses an arbitrary nice function $n(k)$ as a number n of group members, and generates a group domain parameter $D \xleftarrow{\$} \mathtt{GDg}_2(1^k, 1^n)$ and a pair $(gpk, gmsk, \mathbf{gsk}) \xleftarrow{\$} \mathtt{GKg}_2(D)$. $\mathtt{B}_1$ then provides $\mathtt{A}_1$ the group domain parameter D and the group public key gpk.

When $\mathtt{A}_1$ queries (i, M) to $\mathtt{A}_1$'s signing oracle $\mathtt{GSg}_2(D, gpk, \mathbf{gsk}[\cdot], \cdot)$, $\mathtt{B}_1$ computes σ correctly using $\mathbf{gsk}$, responds with σ as the response from $\mathtt{GSg}_2(D, gpk, \mathbf{gsk}[\cdot], \cdot)$, and records the pair (M, σ) to the list L.

$\mathtt{A}_1$ outputs a group public key $\overline{gpk}$. $\mathtt{B}_1$ takes at random (m, σ) from the list L and obtains $\tilde{\mathbf{R}}$ and $\overline{\mathbf{R}}$ in the same way to the Equations (1). $\mathtt{B}_1$ also parses σ as $(\mathbf{T}, c, \mathbf{s})$ and $\overline{gpk}$ as $(\overline{gpk_0}, \overline{pk}, \overline{c_{\mathcal{TC}}}, \overline{m_{\mathcal{TC}}}, \overline{r_{\mathcal{TC}}})$. It then outputs a pair $(x, y) = ((m||gpk_0||pk||c_{\mathcal{TC}}, \mathbf{T}, \tilde{\mathbf{R}}), (m||\overline{gpk_0}||\overline{pk}||\overline{c_{\mathcal{TC}}}, \mathbf{T}, \overline{\mathbf{R}}))$. If $(m, \sigma, \overline{gpk})$ satisfies $\mathtt{GSv}_2(D, gpk, m, \sigma) = 1$ and $\mathtt{GSv}_2(D, \overline{gpk}, m, \sigma) = 1$, we can see that

$$c = \mathcal{H}(m||gpk_0||pk||c_{\mathcal{TC}}, \mathbf{T}, \tilde{\mathbf{R}}) = \mathcal{H}(x),$$
$$c = \mathcal{H}(m||\overline{gpk_0}||\overline{pk}||\overline{c_{\mathcal{TC}}}, \mathbf{T}, \tilde{\mathbf{R}}) = \mathcal{H}(x),$$

where $\sigma = (\mathbf{T}, c, \mathbf{s})$. If $(m_{\mathcal{TC}}, r_{\mathcal{TC}}) = (\overline{m_{\mathcal{TC}}}, \overline{r_{\mathcal{TC}}})$ and $gpk \neq \overline{gpk}$, we obtain $(gpk_0, pk, c_{\mathcal{TC}}) \neq (\overline{gpk_0}, \overline{pk}, \overline{c_{\mathcal{TC}}})$. Therefore, if $\mathtt{B}$ takes a pair (m, σ) that satisfies the winning conditions of $\mathbf{Exp}^{ksa1}_{\mathcal{GS}_2, \mathtt{A}_1}$ from the list L and $(m_{\mathcal{TC}}, r_{\mathcal{TC}}) = (\overline{m_{\mathcal{TC}}}, \overline{r_{\mathcal{TC}}})$, $\mathtt{B}_1$ breaks the collision resistance of $\mathcal{H}$.

Since the keys given to $\mathtt{A}_1$ and the answers of $\mathtt{A}_1$'s queries are all valid, $\mathtt{B}_1$ simulates $\mathtt{A}_1$'s enviroment with proper distributions. Therefore, $\mathtt{A}_1$ outputs gpk that satisfies the winning conditions in $\mathbf{Exp}^{ksa1}_{\mathcal{GS}_2, \mathtt{A}_1}$ and $(m_{\mathcal{TC}}, r_{\mathcal{TC}}) = (\overline{m_{\mathcal{TC}}}, \overline{r_{\mathcal{TC}}})$ with advantage $\frac{\epsilon}{2}$. If $\mathtt{A}_1$'s output satisfies the winning conditions of $\mathbf{Exp}^{ksa1}_{\mathcal{GS}_2, \mathtt{A}_1}$ with some advantage ϵ_0, (M, σ) that is taken by $\mathtt{B}_1$ from the list L satisfies

$\mathtt{GSv}(D, \overline{gpk}, M, \sigma)$ with advantage at least $\frac{\epsilon_0}{q_s}$. Hence, $\mathtt{B_1}$ can break the collision resistance of $\mathcal{H}$ with at least $\frac{\epsilon}{2q_s}$. Note that, since n is an nice function on k and q_s is polynomial bounded on k, ϵ is negligible whenever ϵ' is.

$\mathtt{B_1}$'s running time is the sum of $\mathtt{A_1}$'s running time and the time to answer $\mathtt{A_1}$'s queries to the signing oracle, to compute $\mathtt{A_1}$'s input, and to take (m, σ) from the list L. It takes constant time to answer each query, to compute $\mathtt{A_1}$'s input, and to take (m, σ). The numbers of queries to the signing oracle is at most q_s. If $\mathtt{A_1}$ runs in time t, $\mathtt{B_1}$ runs in time $t' = t + O(q_s)$.

Secondly, we construct an algorithm $\mathtt{B_2}$ that breaks the collision resistance of a trapdoor commitment scheme $\mathcal{TC}$. $\mathtt{B_2}$ is given $\mathcal{TC}$'s security parameter 1^l and public key pk. $\mathtt{B_2}$ chooses $\mathcal{GS}_0$'s reasonable security parameter k and an arbitrary nice function $n(k)$ as a number n of group members, and generates a group domain parameter $D_0 \xleftarrow{\$} \mathtt{GDg_0}(1^k, 1^n)$ and a pair $(gpk_0, gmsk_0, \mathbf{gsk}_0) \xleftarrow{\$} \mathtt{GKg_0}(D_0)$. $\mathtt{B_2}$ also chooses $m_{\mathcal{TC}}, r_{\mathcal{TC}}$ at random and computes $c_{\mathcal{TC}} = \mathcal{C}(pk, m_{\mathcal{TC}}, r_{\mathcal{TC}})$. $\mathtt{B_2}$ then provides $\mathtt{A_2}$ the group domain parameter $D = (D_0, 1^l)$ and the group public key $gpk = (gpk_0, pk, c_{\mathcal{TC}}, m_{\mathcal{TC}}, r_{\mathcal{TC}})$.

$\mathtt{B_2}$ simulates $\mathtt{A_2}$'s signing oracle is the same way to the one in the $\mathtt{A_1}$.

$\mathtt{A_2}$ outputs a group public key $\overline{gpk}$. $\mathtt{B_2}$ parses $\overline{gpk}$ as $\overline{gpk} = (\overline{gpk_0}, \overline{pk}, \overline{c_{\mathcal{TC}}}, \overline{m_{\mathcal{TC}}}, \overline{r_{\mathcal{TC}}})$ and then outputs a pair $(x, y) = ((m_{\mathcal{TC}}, r_{\mathcal{TC}}), (\overline{m_{\mathcal{TC}}}, \overline{r_{\mathcal{TC}}})$. If $\mathtt{A_2}$ outputs $\overline{gpk}$ that satisfies $(m_{\mathcal{TC}}, r_{\mathcal{TC}}) \neq (\overline{m_{\mathcal{TC}}}, \overline{r_{\mathcal{TC}}})$ and the second winning condition of $\mathbf{Exp}^{ksa1}_{\mathcal{GS}_2, \mathtt{A_2}}$ then $\mathtt{B_2}$ breaks the collision resistance of $\mathcal{TC}$.

Since the keys given to $\mathtt{A_2}$ and the answers of $\mathtt{A_2}$'s queries are all valid, $\mathtt{B_2}$ simulates $\mathtt{A_2}$'s enviroment with proper distributions. Therefore, $\mathtt{A_2}$ outputs gpk such that satisfies the winning conditions in $\mathbf{Exp}^{ksa1}_{\mathcal{GS}_2, \mathtt{A_2}}$ and $(m_{\mathcal{TC}}, r_{\mathcal{TC}}) \neq (\overline{m_{\mathcal{TC}}}, \overline{r_{\mathcal{TC}}})$ with advantage $\frac{\epsilon}{2}$, and $\mathtt{B_2}$ can break the collision resistance of the trapdoor commitment $\mathcal{TC}$ with the same advantage. Note that, since n is a nice function on k, ϵ is negligible whenever ϵ'' is.

$\mathtt{B_2}$'s running time is the sum of $\mathtt{A_2}$'s running time, the time to answer $\mathtt{A_2}$'s queries to the signing oracle, and to compute $\mathtt{A_2}$'s input that is created in constant time. Each query is answered in constant time and the numbers of queries to the signing oracle is at most q_s. If $\mathtt{A_2}$ runs in time t, $\mathtt{B_2}$ runs in time $t'' = t + O(q_s)$. □

Note that ϵ is negligible whenever both ϵ' and ϵ'' are (cf. proof of the Lemma 18), and that $\mathtt{B_i}$ is polynomial adversary whenever $\mathtt{A_i}$ is since $\mathtt{A_i}$'s input length is polynomial bounded on $\mathtt{B_i}$'s one.

Lemma 19. *$\mathcal{GS}_2$ is not secure against key substitution attack 2.*

Proof. We construct an algorithm $\mathtt{A}$ which breaks the security of $\mathcal{GS}_2$ against the key substitution attack 2 as follows. $\mathtt{A}$ takes as input a group domain parameter D, a group public key $gpk = (gpk_0, pk_{\mathcal{TC}}, c_{\mathcal{TC}}, m_{\mathcal{TC}}, r_{\mathcal{TC}})$, a manager's secret key $gmsk = (\xi_1, \xi_2, sk_{\mathcal{TC}})$, and users' secret keys $\mathbf{gsk}$, and computes another group public key $\overline{gpk}$, a message m, and a group signature σ as follows:

1. $\mathtt{A}$ chooses a message m, user's index $i \in [n]$, and $\overline{m_{\mathcal{TC}}}$ $(\neq m_{\mathcal{TC}}) \in M$ at random.
2. $\mathtt{A}$ computes $\sigma \xleftarrow{\$} \mathtt{GSg_2}(D, gpk, \mathbf{gsk}[i], m)$ and $\overline{r_{\mathcal{TC}}} \leftarrow \mathcal{D}(sk_{\mathcal{TC}}, m_{\mathcal{TC}}, r_{\mathcal{TC}}, c_{\mathcal{TC}}, \overline{m_{\mathcal{TC}}})$.
3. $\mathtt{A}$ sets $\overline{gpk} = (gpk_0, pk_{\mathcal{TC}}, c_{\mathcal{TC}}, \overline{m_{\mathcal{TC}}}, \overline{r_{\mathcal{TC}}})$.
4. $\mathtt{A}$ returns $(\overline{gpk}, m, \sigma)$.

$(\overline{gpk}, m, \sigma)$ satisfies the first and third winning conditions of $\mathbf{Exp}^{ksa2}_{\mathcal{GS}_2, \mathtt{A}}$, since σ is created correctly using $\mathtt{GSg_2}$ and $\overline{r_{\mathcal{TC}}}$ is produced correctly using $\mathcal{D}$, respectively. It is trivial that it satisfies the forth winning condition: $gpk \neq \overline{gpk}$. Considering the second winning condition, since a part of $\overline{gpk}$ that is related to σ is not different from gpk's one, it holds $\mathtt{GSv_2}(D, \overline{gpk}, m, \sigma) = 1$.

Hence, $\mathtt{A}$ breaks the security against the key substitution attack 2. □

6 Conclusions

We formalized two key substitution attacks against group signature schemes, and showed that the security requirements proposed by now (full-anonymity and full-traceability) is not sufficient regarding even the key substitution attack 1. We proposed the group signature scheme that is secure against key substitution attack 2, fully-anonymous, and fully-traceable. We also showed that there is a separation between the attack 1 and attack 2.

References

[1] Ateniese, G., Camenisch, J., Joye, M., and Tsudik, G. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO* (Santa Barbara, California, USA, August 2000), M. Bellare, Ed., vol. 1880 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 255–270.

[2] Ateniese, G., and de Medeiros, B. On the key exposure problem in chameleon hashes. In *Security in Communication Networks, 4th International Conference, SCN 2004, Amalfi, Italy, September 8-10, 2004, Revised Selected Papers* (2004), C. Blundo and S. Cimato, Eds., vol. 3352 of *Lecture Notes in Computer Science*, Springer, pp. 165–179.

[3] Ateniese, G., and Tsudik, G. Some open issues and new directions in group signatures. In *FC '99: Proceedings of the Third International Conference on Financial Cryptography* (London, UK, February 1999), M. K. Franklin, Ed., vol. 1648 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 196–211.

[4] Bellare, M., Micciancio, D., and Warinschi, B. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT* (Warsaw, Poland, May 2003), E. Biham, Ed., vol. 2656 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 614–629.

[5] Blake-Wilson, S., and Menezes, A. Unknown key-share attacks on the station-to-station (STS) protocol. In *Public Key Cryptography* (Kamakura, Japan, March 1999), H. Imai and Y. Zheng, Eds., vol. 1560 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 154–170.

[6] Bohli, J.-M., Röhrich, S., and Steinwandt, R. Key substitution attacks revisited: Taking into account malicious signers. *International Journal of Information Security 5*, 1 (2006), 30–36.

[7] Boneh, D., and Boyen, X. Short signatures without random oracles. In *EUROCRYPT* (Interlaken, Switzerland, May 2004), C. Cachin and J. Camenisch, Eds., vol. 3027 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 56–73.

[8] Boneh, D., Boyen, X., and Shacham, H. Short group signatures. In *CRYPTO* (Santa Barbara, California, USA, August 2004), M. Franklin, Ed., vol. 3152 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 41–55.

[9] Boneh, D., Lynn, B., and Shacham, H. Short signatures from the Weil pairing. In *ASIACRYPT* (Gold Coast, Australia, December 2001), C. Boyd, Ed., vol. 2248 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 514–532.

[10] Brassard, G., Chaum, D., and Crépeau, C. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences 37*, 2 (1988), 156–189.

[11] Bresson, E., Catalano, D., and Pointcheval, D. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In *ASIACRYPT* (Taipei, Taiwan, November 2003), C. S. Laih, Ed., vol. 2894 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 37–54.

[12] Chaum, D., and van Heyst, E. Group signatures. In *EUROCRYPT* (Brighton, UK, April 1991), D. Davies, Ed., vol. 547 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 257–265.

[13] Chen, L., and Pedersen, T. P. New group signature schemes (extended abstract). In *EUROCRYPT* (Perugia, Italy, May 1994), A. De Santis, Ed., vol. 950 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 171–181.

[14] Geiselmann, W., and Steinwandt, R. A key substitution attack on sflash^{v3}. Cryptology ePrint Archive, Report 2003/245, 2003.

[15] Krawczyk, H., and Rabin, T. Chameleon signatures. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2000, San Diego, California, USA* (2000), The Internet Society.

[16] Menezes, A., and Smart, N. P. Security of signature schemes in a multi-user setting. *Designs, Codes and Cryptography 33*, 3 (2004), 261–274.

[17] Rogaway, P., and Shrimpton, T. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *FSE* (2004), B. K. Roy and W. Meier, Eds., vol. 3017 of *Lecture Notes in Computer Science*, Springer, pp. 371–388.

[18] Shoup, V. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT* (Konstanz, Germany, May 1997), W. Fumy, Ed., vol. 1233 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 256–266.

[19] Tan, C. H. Key substitution attacks on some provably secure signature schemes. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 87-A*, 1 (2004), 226–227.

[20] Tan, C. H. Key substitution attacks on provably secure short signature schemes. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 88-A*, 2 (2005), 611–612.

[21] Tan, C.-H. Signature scheme in multi-user setting. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E89-A*, 5 (2006), 1339–1345.

A Description of Boneh, Boyen and Shacham's Short Group Signatures

We describe the short group signature scheme $\mathcal{GS}_0 = (\mathtt{GDg}_0, \mathtt{GDv}_0, \mathtt{GKg}_0, \mathtt{GKv}_0, \mathtt{GSg}_0, \mathtt{GSv}_0, \mathtt{Open}_0)$.

$\mathtt{GDg}_0(1^k, 1^n)$ This randomized algorithm takes as input parameters 1^k and 1^n, the security parameter and the number of members of the group, respectively. Note that n is a nice function on k. It proceeds as follows. It chooses a bilinear group pair (G_1, G_2) with a computable isomorphism ϕ and a computable map e as in Section 2.3 and a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$. Suppose that the SDH assumption holds on (G_1, G_2), the Linear assumption holds on G_1. The group domain parameter is $D = (1^k, 1^n, G_1, G_2, \phi, e, \mathcal{H})$.

$\texttt{GDv}_0(1^k, 1^n, D)$ This deterministic algorithm takes as input parameters 1^k, 1^n and D, the security parameter, the number of members of the group, and the group domain parameter, respectively, and outputs 1 if D has the correct format. Otherwise, it returns 0.

$\texttt{GKg}_0(D)$ This randomized algorithm takes as input a group domain parameter D and proceeds as follows. It selects a generator g_2 in G_2 uniformly at random and sets $g_1 = \phi(g_2)$. It also selects $h \xleftarrow{R} G_1 \setminus \{1_{G_1}\}$ and $\xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p^*$, and sets $u, v \in G_1$ such that $u^{\xi_1} = v^{\xi_2} = h$. It then selects $\gamma \xleftarrow{R} \mathbb{Z}_p^*$ and sets $w = g_2^{\gamma}$. Using γ, it generates for each user i, $1 \leq i \leq n$, an SDH tuple (A_i, x_i) by selecting $x_i \xleftarrow{R} \mathbb{Z}_p^*$ and computing $A_i = g_1^{\frac{1}{\gamma + x_i}} \in G_1$. The group public key is $gpk = (g_1, g_2, h, u, v, w)$. The private key of the group manager that is able to trace signatures is $gmsk = (\xi_1, \xi_2)$. Each user's private key is the tuple $\mathbf{gsk}[i] = (A_i, x_i)$. No party except for the private-key issuer is allowed to possess γ.

$\texttt{GKv}_0(D, gpk)$ This deterministic algorithm takes as input a group domain parameter D and a group public key gpk and outputs 1 if gpk is in a correct format. Otherwise, it returns 0.

$\texttt{GSg}_0(D, gpk, \mathbf{gsk}[i], M)$ This deterministic algorithm takes as input a group domain parameter D, a group public key $gpk = (g_1, g_2, h, u, v, w)$, a user's key $\mathbf{gsk}[i] = (A_i, x_i)$, and a message $M \in \{0,1\}^*$ and computes the signature as follows:

1. It selects exponents $\alpha, \beta \xleftarrow{R} \mathbb{Z}_p$ and computes the value $\mathbf{T} = (T_1, T_2, T_3)$ such that:
$$T_1 = u^{\alpha}, \quad T_2 = v^{\beta}, \quad T_3 = A_i h^{\alpha+\beta}.$$
2. It computes two helper values $\delta_1 = x_i\alpha$ and $\delta_2 = x_i\beta \in \mathbb{Z}_p$, and picks blinding values $r_\alpha, r_\beta, r_x, r_{\delta_1}$, and r_{δ_2} randomly from $\mathbb{Z}_p$.
3. It computes $\mathbf{R} = (R_1, R_2, R_3, R_4, R_5)$ as follows:
$$\begin{gathered} R_1 = u^{r_\alpha}, \quad R_2 = v^{r_\beta}, \\ R_3 = e(T_3, g_2)^{r_x} \cdot e(h, w)^{-r_\alpha - r_\beta} \cdot e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}}, \\ R_4 = T_1^{r_x} \cdot u^{-r_{\delta_1}}, \quad R_5 = T_2^{r_x} \cdot v^{-r_{\delta_2}}. \end{gathered}$$
4. It computes $c = \mathcal{H}(M, \mathbf{T}, \mathbf{R})$.
5. Using c, it constructs the value $\mathbf{s} = (s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ as follows:
$$s_\alpha = r_\alpha + c\alpha, \quad s_\beta = r_\beta + c\beta, \quad s_x = r_x + cx_i, \quad s_{\delta_1} = r_{\delta_1} + c\delta_1, \quad s_{\delta_2} = r_{\delta_2} + c\delta_2.$$
6. it outputs the signature $\sigma = (\mathbf{T}, c, \mathbf{s})$.

$\texttt{GSv}_0(D, gpk, M, \sigma)$ This deterministic algorithm takes as input a group domain parameter D, a group public key $gpk = (g_1, g_2, h, u, v, w)$, a message M, and a group signature $\sigma = (\mathbf{T}, c, \mathbf{s})$ and verifies as follows:

1. It computes $\tilde{\mathbf{R}} = (\tilde{R_1}, \tilde{R_2}, \tilde{R_3}, \tilde{R_4}, \tilde{R_5})$ as follows:
$$\begin{gathered} \tilde{R_1} = u^{s_\alpha} \cdot T_1^{-c}, \quad \tilde{R_2} = v^{s_\beta} \cdot T_2^{-c}, \\ \tilde{R_3} = e(T_3, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha - s_\beta} \cdot e(h, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot (\tfrac{e(T_3, w)}{e(g_1, g_2)})^c, \\ \tilde{R_4} = u^{-s_{\delta_1}} \cdot T_1^{s_x}, \quad \tilde{R_5} = v^{-s_{\delta_2}} \cdot T_2^{s_x}. \end{gathered}$$
2. It checks that $c = \mathcal{H}(M, \mathbf{T}, \tilde{\mathbf{R}})$. It returns 1 if this check succeeds. Otherwise, it returns 0.

$\texttt{Open}_0(D, gpk, gmsk, M, \sigma)$ This deterministic algorithm takes as input a group domain parameter D, a group public key $gpk = (g_1, g_2, h, u, v, w)$, the corresponding group manager's private key $gmsk = (\xi_1, \xi_2)$, a message M, and a signature $\sigma = (\mathbf{T}, c, \mathbf{s})$, and proceeds as follows.

1. It verifies that σ is a valid signature on M using $\texttt{GSv}_0(D, gpk, M, \sigma)$.
2. It recovers the user's A as $A = T_1^{-\xi_1} \cdot T_2^{-\xi_2} \cdot T_3$ from $\mathbf{T} = (T_1, T_2, T_3)$.
3. Using the elements $\{A_i\}$ of the users' private keys, if there exists a user indexed i corresponding to the identity A recovered from the signature, it outputs i, else outputs $\perp$.