ISSN 1342-2812

Research Reports on Mathematical and Computing Sciences

Key-Substitutable Signature and its Application to Certified Signature

Koichi Sakumoto and Keisuke Tanaka

January 2008, C–250

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES C: Computer Science

Key-Substitutable Signature and its Application to Certified 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 7, 2008

Abstract

The key substitution property is introduced by Blake-Wilson and Menezes [1] and formalized by Menezes and Smart [8] as attacks. The key substitution property is as follow: another person other than true signer can produce another public (and secret) key such that a message and signature pair created by the signer is valid under the public key. The research of the key substitution attacks [8, 6, 10, 2, 11, 12] is only to attack a certain signature scheme or only to detect the attacks so far. In this paper, we introduce key-substitutable signature scheme. In the key-substitutable signature scheme, it is basically infeasible to produce a substitute public key, however, an user can create a substituted key pair by interaction with the original signer. We propose the formal model of the key-substitutable signature scheme and formalize the security requirements, *unforgeability* and *non-substitutability.* We also propose a construction of key-substitutable signature scheme based on ElGamal signature scheme and prove that the construction satisfies the all security requirements. Furthermore, we construct a new certified-signature scheme achieving higher security based on key-substitutable signature schemes. We also show that the "traditional" certified-signature scheme in [3] does not satisfy this higher security.

Keywords: signature scheme, key substitution attack, certified-signature scheme.

1 Introduction

Background. Digital signatures should be "unforgeable", which means that it is infeasible to produce signatures of other users on documents that they did not sign. This property provides a clear statement of the essential requirements of handwritten signatures. However, in contrast to handwritten signatures, we cannot recognize that a digital signature produced by Alice is Alice's signature at a glance. It is not ascertained that the signature is Alice's until the signature is verified with Alice's public key. Therefore, it may be ascertained that the signature is Bob's when it verified with Bob's public key. It is an interesting property that a signature may be regarded as both Alice's and Bob's.

Blake-Wilson and Menezes [1] introduced this property as the duplicate-signature key selection property on signature schemes. Menezes and Smart [8] argued that the duplicate-signature key selection property seems to be a considerable attack, called it the key substitution attack, in the "multi-user setting". In this setting, there may be more than two signers that use the same system and different public keys. The research of the key substitution attacks [8, 6, 10, 2, 11, 12] consider the key substitution mainly as the negative property.

Contribution. Let (m_A, σ_A) be a message and signature pair produced by a user A with a public and secret key pair (pk_A, sk_A). We call A the original signer of (m_A, σ_A). We say that a public key pk_B is a substitute public key for pk_A on (m_A, σ_A) if (m_A, σ_A) is valid under the public key pk_B and $pk_B \neq pk_A$. We say that a public and secret key pair (pk_B, sk_B) is a substitute public and secret key pair for (pk_A, sk_A) on (m_A, σ_A) if pk_B is a substitute public key for pk_A on (m_A, σ_A) and sk_B is the corresponding secret key.

In this paper, we consider the key substitution as the positive property and formalize it as *key-substitutable signature*. The key substitution attack on standard signature schemes was introduced by Blake-Wilson and Menezes [1] and studied in [8, 10, 11, 12]. In the key substitution attack, users other than the original signer can generate substitute public keys. In our model of key-substitutable signature schemes, it is infeasible for any user to independently perform the key substitution attack without the assistance of the original signer. By running an interactive protocol with the original signer A, another user B $(\neq A)$ can generate a substitute public and secret key pair for (pk_A, sk_A). We call such an interactive protocol as a *key substitution protocol* and call the signature schemes with key substitution protocols as *key-substitutable signature schemes*. Furthermore, we require that, after running the key substitution protocol, the original signer A cannot forge the signatures signed by B who owns a substitute public key (and its corresponding secret key), and that B also cannot forge A's signatures. We formalize these security requirements as *non-substitutability* and *unforgeability*. In our model, we propose a construction based on the ElGamal signature scheme. Furthermore, we construct a new certified-signature scheme achieving higher security based on key-substitutable signature schemes. We also show that the "traditional" certified-signature scheme in [3] does not satisfy this higher security.

Certified-Signature Schemes. Public-key cryptosystem implicitly relies on the existence of public-key infrastructure (PKI). Each user associated with some PKI has a public and secret key pair for the cryptosystem where the public key is authenticated and publicly available. The designers of public-key cryptosystems always define how the public and the secret keys are generated and used, but almost never carefully specify how to bind keys with user identities. Recently, Boldyreva, Fischlin, Palacio, and Warinschi [3] studied PKIs with respect to security of encryption and digital signature. They consider the security of joint constructions of the algorithms of an encryption or a signature scheme and a key-registration protocol between the certification authority (CA) and the users. Boldyreva, Fischlin, Palacio, and Warinschi [3] call the model of signature schemes with PKI as *certified-signature scheme*.

The adversary in their model attempts the forgery by outputting a user identity $\mathtt{ID}$, a public key pk, a message m and, a signature σ. Roughly, the adversary wins if the signature σ is valid on m under the chosen public key pk, and either (1) the honest user $\mathtt{ID}$ has registered the public key pk and has not signed the message m, (2) the public key pk has not been registered with an honest CA, or (3) the same public key pk has been registered by another honest user $\mathtt{ID}'$ with an honest CA. The condition (1) corresponds to the existential unforgeability for standard digital signature schemes, and we require that it holds even if the CA is corrupted. The condition (2) guarantees that signatures on the keys that are not bound to the identities of the users (i.e., "outside of the PKI") are not valid on the assumption that CA is honest. The condition (3) prevents attacks where, for example, a malicious user claims the authorship of a message signed by a different user on the assumption that CA is honest.

Weakening the assumption that CA is honest is one of important issues in PKI. The security definition of the paper [3] guarantees the condition (3) on the assumption that the CA is honest. In this paper, our proposed scheme of certified signature based on key substitutable signature guarantee the security with respect to the condition (3) even if the CA is corrupted.

Organization. In the next section, we review some cryptographic tools on which our main construction is based. We describe the model of key-substitutable signature in Section 3 and formally define the security requirements in Section 4. In Section 5, we propose two concrete key-substitutable signature schemes based on the ElGamal signature scheme [5, 9], and prove that the scheme satisfies the security requirements in the random oracle model. Furthermore, in Section 6, we suggest that key-substitutable signature schemes can be applied to certified signature.

2 Preliminaries

Let X be a finite set. We denote by $x \xleftarrow{R} X$ that an element of $x \in X$ is randomly chosen. If A is a randomized algorithm, then $[A(x, y, \cdots)]$ denotes the set of all elements output by A on inputs $x, y, \cdots$ with positive probability. If G is a group, $\langle g \rangle$ denotes a group generated by $g \in G$. We call a function $\texttt{neg} : \mathbb{N} \to \mathbb{R}$ negligible, if for every positive polynomial $p(\cdot)$ there exists an N such that for all $n > N$, $\texttt{neg}(n) < \frac{1}{p(n)}$.

The ElGamal signature scheme. We review the ElGamal signature scheme [5, 9] which we employ in our proposed scheme. In order to apply the ElGamal scheme into our scheme, we require that the subgroup of $\mathbb{Z}_p^*$ is restricted to that with the prime p of the form $p = 2q + 1$ and $q \equiv 3 \pmod 4$ where q is also prime.

- *The setup algorithm*: it chooses a random large prime p of size λ and a random element g whose order is prime q where $p = 2q + 1$ and $q \equiv 3 \pmod 4$. The common parameter is (p, g).
- *The key generation algorithm*: it chooses $x \xleftarrow{R} \mathbb{Z}_q$ and computes $y = g^x \bmod p$. The secret key is x. The public key is y.
- *The signing algorithm*: in order to sign a message m, one generates $k \xleftarrow{R} \mathbb{Z}_q^*$ and computes $r = g^k \bmod p$ and $s = k^{-1}(H(m, r) - xr) \bmod q$. The signature is $\sigma = (s, r)$.
- *The verification algorithm*: it checks the equation: $g^{H(m,r)} = y^r r^s \pmod p$.

The scheme employs a hash function H over $\mathbb{Z}_q$ modeled as the random oracle in the security proof. In this paper, we use the following fact [9]: the ElGamal signature scheme is existential unforgeable against the key-only attack under the assumption that the discrete logarithm problem is hard in the random oracle model. In precise, we assume that the ElGamal signature scheme is (t, ϵ, q_H)-EU-KOA, that is, there is no adversary $\mathtt{A}$ who takes as input a public key generated by the key generation algorithm, accesses to the random oracle at most q_H times, and forges a signature with advantage at least ϵ in time t.

The property of primes. In our construction, we use primes p and q such that $p = 2q + 1$ and $q \equiv 3 \pmod 4$. We use the following property of primes: if g is an element whose order q in $\mathbb{Z}_p^*$ then $(g^x \bmod p) \bmod q \neq 0$ for all $x \in \mathbb{Z}_q$. Thus, every element in $\mathbb{Z}_p^*$ has always its inverse in $\mathbb{Z}_q^*$. We briefly explain this fact. It is easy to see that $p \equiv -1 \pmod 8$. From Theorem A.48 of [4] and the fact that $p \equiv -1 \pmod 8$, we obtain $(\frac{2}{p}) = 1$, where $(\frac{x}{p})$ is the Legendre symbol of $x \bmod p$. From Proposition A.47 of [4] and $(\frac{2}{p}) = 1$, we obtain $2^{\frac{p-1}{2}} \equiv 1 \pmod p$. Therefore, $2^q \equiv 2^{\frac{p-1}{2}} \equiv 1 \pmod p$. Since $2q \equiv -1 \pmod p$, we have $(2q)^q \equiv -1 \pmod p$. Hence, we can see $q^q \equiv -1 \pmod p$, that is, $q \notin \langle g \rangle$. This implies $(g^x \bmod p) \bmod q \neq 0$ for all $x \in \mathbb{Z}_q$.

3 The model

In the standard signature schemes, a user generates a public and secret key pair (pk, sk). The user produces signatures using the secret key sk for messages. The signatures are verified using the public key pk.

The key substitution attacks on standard signature schemes were introduced by Blake-Wilson and Menezes [1] and studied in [8, 10, 11, 12]. In the key substitution attack, users other than the original signer can generate substitute public keys. Let (m_A, σ_A) be a message and signature pair produced by a user A with a public and secrete key pair (pk_A, sk_A). We call A the original signer of (m_A, σ_A). We say that a public key pk_B is a substitute public key for pk_A on (m_A, σ_A) if (m_A, σ_A) is valid under the public key pk_B and $pk_B \neq pk_A$. We say that a public and secret key pair (pk_B, sk_B) is a substitute public and secret key pair for (pk_A, sk_A) on (m_A, σ_A) if pk_B is a substitute public key for pk_A on (m_A, σ_A) and sk_B is the corresponding secret key. We also say that B is a substitute signer for A on (m_A, σ_A) if B has a substitute public and secret key pair for (pk_A, sk_A) on (m_A, σ_A).

In our model of key-substitutable signature scheme, basically, it is infeasible for any user to independently perform the key substitution attack. However, by running an interactive protocol with the original signer A, another user B ($\neq A$) can generate a substitute public and secret key pair for (pk_A, sk_A). We call such an interactive protocol as a *key substitution protocol* and call a signature scheme with a key substitution protocol as a *key-substitutable signature scheme*. Furthermore, we require the original signer A cannot forge another signer B's signatures and B also cannot forge A's signatures, after running the key substitution protocol.

Algorithms. A key-substitutable signature scheme $\mathcal{KS} = (\texttt{Setup}, \texttt{Gen}, \texttt{Sig}, \texttt{Ver}, (\texttt{KSA1}, \texttt{KSA2}))$ is specified by a setup algorithm `Setup`, a key generation algorithm `Gen`, a signing algorithm `Sig`, a verification algorithm `Ver`, and a key substitution protocol (`KSA1`, `KSA2`).

The setup algorithm `Setup` is a randomized algorithm which takes as input 1^λ, where $\lambda \in \mathbb{N}$ is the security parameter, and outputs a common parameter `cp` shared among signers. We denote this as $\texttt{cp} \leftarrow \texttt{Setup}(1^\lambda)$.

The key generation algorithm `Gen` is a randomized algorithm which takes as input `cp`, and outputs a pair (pk, sk), where pk is a public key and sk is a secret key. We denote this as $(pk, sk) \leftarrow \texttt{Gen}(\texttt{cp})$.

The signing algorithm `Sig` is a randomized algorithm which takes as input $(\texttt{cp}, pk, sk, m)$, where m is a message, and outputs a signature σ. We denote this as $\sigma \leftarrow \texttt{Sig}(\texttt{cp}, pk, sk, m)$.

The verification algorithm `Ver` is a deterministic algorithm which takes as input $(\texttt{cp}, pk, m, \sigma)$, and outputs 0 or 1. We denote this as $0/1 \leftarrow \texttt{Ver}(\texttt{cp}, pk, m, \sigma)$.

The key substitution protocol `KSA` is a pair of interactive randomized algorithms (`KSA1`, `KSA2`) which takes $(\texttt{cp}, pk, m, \sigma)$ as common input. `KSA1` takes as auxiliary input sk. `KSA1` outputs a public key $\overline{pk}$ and `KSA2` outputs a pair $(\overline{pk}, \overline{sk})$ after the interaction. We denote this by $(\overline{pk}; \overline{pk}, \overline{sk}) \leftarrow \langle \texttt{KSA1}(sk), \texttt{KSA2} \rangle(\texttt{cp}, pk, m, \sigma)$.

Let (m_A, σ_A) be a message and signature pair produced by a user A with a public and secret key pair (pk_A, sk_A). In the key substitution protocol which takes as common input $(\texttt{cp}, pk_A, m_A, \sigma_A)$ and auxiliary input sk_A, `KSA1` is an algorithm of the original signer A and `KSA2` is an algorithm of another user who wants a substitute public and secret key pair on (m_A, σ_A). That is, the output $(\overline{pk}; \overline{pk}, \overline{sk}) \leftarrow \langle \texttt{KSA1}(sk), \texttt{KSA2} \rangle(\texttt{cp}, pk_A, m_A, \sigma_A)$ is supposed to satisfy $\texttt{Ver}(\texttt{cp}, \overline{pk}, m_A, \sigma_A) = 1$.

For simplifying the model, we assume that the key substitution protocol runs on the secure channel. That is, we do not consider the attacker eavesdropping the interaction with `KSA1` and `KSA2`.

Correctness. A key-substitutable signature scheme must satisfy the following correctness requirements: a signature created correctly by the signing algorithm is always valid. More precisely, for any common parameter $\mathtt{cp}$ generated by the setup algorithm, any public and secret key pair (pk, sk) produced by the key generation algorithm or the key substitution protocol which takes as input the common parameter $\mathtt{cp}$, any message m, and any signature σ created by the signing algorithm under (pk, sk), the verification algorithm must accept the message and signature pair (m, σ) under the public key pk. We formalize this as follows.

Definition 1 (correctness)**.** *We say that a key-substitutable signature scheme $\mathcal{KS}$ is correct if $\mathcal{KS}$ satisfies the following condition:* $\forall\lambda \in \mathbb{N}$, $\mathtt{cp} \in [\mathtt{Setup}(1^\lambda)]$, $(pk, sk) \in [\mathtt{Gen}(\mathtt{cp})] \cup [\langle \mathtt{KSA1}(\cdot), \mathtt{KSA2}\rangle(\mathtt{cp}, \cdot, \cdot, \cdot)]$, $m \in M$, $\sigma \in [\mathtt{Sig}(\mathtt{cp}, pk, sk, m)]$, $\mathtt{Ver}(\mathtt{cp}, pk, m, \sigma) = 1$.

Key-substitutability. A key-substitutable signature scheme should also satisfy the following the key-substitutability requirement: the original signer A of a message and signature pair (m, σ) with a public and secret key pair (pk_A, sk_A) can allow another user B to generate a substitute public and secret key pair (pk_B, sk_B) for (pk_A, sk_A) on (m, σ) by running the key substitution protocol $\langle \mathtt{KSA1}(sk_A), \mathtt{KSA2}\rangle(\mathtt{cp}, pk_A, m, \sigma)$.

Definition 2 (key-substitutability)**.** *We say that a key-substitutable signature scheme $\mathcal{KS}$ has key-substitutability if $\mathcal{KS}$ satisfies the following condition:* $\forall\lambda \in \mathbb{N}$, $\mathtt{cp} \in [\mathtt{Setup}(1^\lambda)]$, $(pk, sk) \in [\mathtt{Gen}(\mathtt{cp})] \cup [\langle \mathtt{KSA1}(\cdot), \mathtt{KSA2}\rangle(\mathtt{cp}, \cdot, \cdot, \cdot)]$, $m \in M$, $\sigma \in [\mathtt{Sig}(\mathtt{cp}, pk, sk, m)]$,

$$1 - \Pr\left[\begin{array}{c}(pk'; pk', sk') \leftarrow \langle \mathtt{KSA1}(sk), \mathtt{KSA2}\rangle(\mathtt{cp}, pk, m, \sigma) \\ : \mathtt{Ver}(\mathtt{cp}, pk', m, \sigma) = 1 \textit{ and } pk \neq pk'\end{array}\right] = \mathtt{neg}(\lambda).$$

4 The Security notions

The key-substitutable signature scheme should fulfill two security requirements, *unforgeability* and *non-substitutability*. The unforgeability means that no adversary can forge a message and signature pair which is valid under another user's public key. The non-substitutability means that no adversary A can produce a substitute public key pk for another user's public key pk' on (m, σ) without the running key substitution protocol on $(\mathtt{cp}, pk', m, \sigma)$. We note that the key-substitutability and the non-substitutability do *not* conflict. The key-substitutability assures any user, only if they run the key substitution protocol, to produce a substitute public and secret key pair.

In our model, there exist two types of signers, which we call the original signer and a substitute signer. The original signer has a public and secret key pair generated by the key generation algorithm. A substitute signer has a public and secret key pair generated by the key substitution protocol. Therefore, we consider the unforgeability and the non-substitutability with respect to the two type of signers.

4.1 Security requirements for the original signer

We formalize the security requirements, the unforgeability and the non-substitutability for the original signer U_1, whose public and secret key pair (pk, sk) is produced by the key generation algorithm.

We suppose the adversary who attempts to break the unforgeability or the non-substitutability with respect to U_1. That is, the adversary wants to forge a message and signature pair which is valid under U_1's public key, or, to produce a substitute public key $\overline{pk}$ on (m, σ) without running the key substitution protocol with respect to (m, σ) produced by U_1.

We consider the adversary who, given public key pk produced by the key generation algorithm, not only accesses the signing oracle but also runs the key substitution protocol as the role of $\mathtt{KSA2}$

with the original signer U_1. We allow the adversary to run the key substitution protocol with U_1 once per each message and signature pair (m, σ). We require that such no adversary can forge a signature on any message which is valid under the public key of U_1 (unforgeability); and that such no adversary can produce even a substitute public key $\overline{pk}$ on (m, σ) without running key substitution protocol where (m, σ) is one of the message and signature pair generated by user U_1 (non-substitutability).

For a key-substitutable signature scheme $\mathcal{KS} = (\mathtt{Setup}, \mathtt{Gen}, \mathtt{Sig}, \mathtt{Ver}, (\mathtt{KSA1}, \mathtt{KSA2}))$, we associate the adversary $\mathtt{A}$ with the following experiment. $\mathtt{A}$ takes as input the common parameter $\mathtt{cp}$ and the original signer U_1's public key pk generated by the key generation algorithm. $\mathtt{A}$ can access the signing oracle $\mathtt{Sig}(\mathtt{cp}, pk, sk, \cdot)$. Let L_s be the list of message and signature pairs (m, σ) where m is a query to the signing oracle and σ is the corresponding answer. $\mathtt{A}$ can also run the key substitution protocol as the role of $\mathtt{KSA2}$ with the original signer U_1 with respect to each $(m_i, \sigma_i) \in L_s \backslash L_k$. Let $L_k \subset L_s$ be the list of message and signature pairs (m_i, σ_i) such that the adversary $\mathtt{A}$ ran the key substitution protocol with respect to (m_i, σ_i). Let $L_{pk} = \{pk\} \cup \{pk_i\}$ where pk is the input public key pk and pk_i is the output of $\mathtt{KSA1}$ on the common input $(\mathtt{cp}, pk, m_i, \sigma_i)$ $((m_i, \sigma_i) \in L_k)$ with the adversary $\mathtt{A}$. The experiment returns 1 if the adversary defeats the unforgeability, that is, produces a forgery (m, σ) such that (1) (m, σ) is valid under the public key pk; and (2) m is never queried to the signing oracle. The experiment also returns 1 if the adversary breaks the non-substitutability, that is, produce $(\overline{pk}, m, \sigma)$ such that (1) (m, σ) is valid under the output public key $\overline{pk}$; (2) m is queried to the signing oracle and σ is the corresponding answer; (3) $\overline{pk}$ is *not* one of the public keys under which (m, σ) is valid due to the correctness or the key-substitutability. Otherwise the experiment returns 0. Note that, it is easy to generate for the adversary such a public key $\overline{pk}$ if the adversary runs key substitution protocol with respect to (m, σ) because of the key-substitutability.

```
Experiment $\mathbf{Exp}^{\mathtt{atk\text{-}1}}_{\mathcal{KS},\mathtt{A}}(\lambda)$
  $\mathtt{cp} \leftarrow \mathtt{Setup}(1^\lambda)$;  $(pk, sk) \leftarrow \mathtt{Gen}(\mathtt{cp})$;
  $(\overline{pk}, m, \sigma) \leftarrow \mathtt{A}^{\langle \mathtt{KSA1}(sk), \mathtt{A} \rangle(\mathtt{cp}, pk, \cdot, \cdot), \mathtt{Sig}(\mathtt{cp}, pk, sk, \cdot)}(\mathtt{cp}, pk)$;
  If $\mathtt{atk} = \mathtt{uf}$ and $\mathtt{Ver}(\mathtt{cp}, pk, m, \sigma) = 1$ and $(m, \cdot) \notin L_s$ then return 1;
  Else if $\mathtt{atk} = \mathtt{ns}$ and $\mathtt{Ver}(\mathtt{cp}, \overline{pk}, m, \sigma) = 1$ and $(m, \sigma) \in L_s$
    and $\overline{pk} \notin L_{pk}$ then return 1;
  Else return 0;
```

Concerning attacks against the original signer, we define the advantage of the adversary $\mathtt{A}$ on the key-substitutable signature scheme $\mathcal{KS}$ by the probability:

$$\mathbf{Adv}^{\mathtt{atk\text{-}1}}_{\mathcal{KS},\mathtt{A}}(\lambda) = \Pr[\mathbf{Exp}^{\mathtt{atk\text{-}1}}_{\mathcal{KS},\mathtt{A}}(\lambda) = 1].$$

Definition 3 (unforgeability for the original signer)**.** *We say that a key-substitutable signature scheme $\mathcal{KS}$ has (t, ϵ, q_s, q_k)-unforgeability for the original signer if there is no adversary $\mathtt{A}$ such that $\mathbf{Adv}^{\mathtt{uf\text{-}1}}_{\mathcal{KS},\mathtt{A}}(\cdot)$ is at least $\epsilon(\cdot)$; the number of $\mathtt{A}$'s signing queries is bounded by q_s; $\mathtt{A}$ can run the key substitution protocol with respect to q_k message and signature pairs in L_s; and $\mathtt{A}$'s running time is bounded by t.*

Definition 4 (non-substitutability for the original signer)**.** *We say that a key-substitutable signature scheme $\mathcal{KS}$ has (t, ϵ, q_s, q_k)-non-substitutability for the original signer if there is no adversary $\mathtt{A}$ such that $\mathbf{Adv}^{\mathtt{ns\text{-}1}}_{\mathcal{KS},\mathtt{A}}(\cdot)$ is at least $\epsilon(\cdot)$; the number of $\mathtt{A}$'s signing queries is bounded by q_s; $\mathtt{A}$ can run the key substitution protocol with respect to q_k message and signature pairs in L_s; and $\mathtt{A}$'s running time is bounded by t.*

4.2 Security requirements for a substitute signer

We formalize the security requirements, the unforgeability and the non-substitutability for a substitute signer U_2, whose public and secret key pair $(\overline{pk}, \overline{sk})$ is produced by the key substitution protocol. In this protocol, U_2 plays the role of KSA2 and the original signer plays the role of KSA1. We should consider the case that, concerning the security of the substitute signer U_2, the original signer who runs KSA1 may manipulate U_2 to produce fragile keys. Therefore, concerning the key substitution protocol to generate the substitute signer U_2's public and secret key pair, we allow the adversary to choose the common input (pk, m, σ) and to participate the key substitution protocol as the role of KSA1. Furthermore, the adversary can not only access the signing oracle but also run the key substitution protocol as the role of KSA2 with the substitute signer U_2. We allow the adversary to run the key substitution protocol with U_2 once per each pair (m, σ) of message and signature.

We require that such no adversary can forge a signature on any message which is valid under the public key of the substitute signer U_2 (unforgeability); and that such no adversary can produce a substitute public key $\overline{pk}$ (and the corresponding secret key $\overline{sk}$) on (m, σ) without running the key substitution protocol where (m, σ) is one of the message and signature pair (m, σ) generated by the substitute signer U_2 (non-substitutability). In our security definition, the unforgeability and the non-substitutability are required even if the adversary is allowed to participate the substitute signer's key generation as the role of KSA1 by running the key substitution protocol with the substitute signer U_2.

For a key-substitutable signature scheme $\mathcal{KS} = (\mathtt{Setup}, \mathtt{Gen}, \mathtt{Sig}, \mathtt{Ver}, (\mathtt{KSA1}, \mathtt{KSA2}))$, we associate the adversary $\mathtt{A} = (\mathtt{A}_1, \mathtt{A}_2, \mathtt{A}_3)$ with the following experiment. $\mathtt{A}_1$ takes as input the common parameter $\mathtt{cp}$ and chooses a public key pk_0 and a message and signature pair (m^*, σ^*) as he likes. Then, with respect to the common input $(\mathtt{cp}, pk_0, m^*, \sigma^*)$, the adversary $\mathtt{A}_2$ and the substitute signer U_2 run the key substitution protocol as the role of KSA1 and KSA2, respectively. $\mathtt{A}_2$ uses KSA2 to produce U_2's substitute public and secret key pair (pk, sk) for pk_0 on (m^*, σ^*). Then, $\mathtt{A}_3$ receives the target public key pk output by KSA2. $\mathtt{A}$ can access the signing oracle $\mathtt{Sig}(\mathtt{cp}, pk, sk, \cdot)$. Let L_s be the list of message and signature pairs (m, σ) where m is a query to the signing oracle and σ is the corresponding answer. $\mathtt{A}_3$ can also run the key substitution protocol with the substitute signer U_2 with respect to $(m_i, \sigma_i) \in L_s \setminus L_k$. Let $L_k \subset L_s$ be the list of message and signature pairs (m_i, σ_i) such that the adversary $\mathtt{A}$ run the key substitution protocol with respect to (m_i, σ_i). Let $L_{pk} = \{pk\} \cup \{pk_i\}$ where pk is the input public key pk and pk_i is the output of KSA1 on the common input $(\mathtt{cp}, pk, m_i, \sigma_i)$ ($(m_i, \sigma_i) \in L_k$) with the adversary $\mathtt{A}_3$. The experiment returns 1 if the adversary defeats the unforgeability, that is, produces a forgery (m, σ) such that: (1) (m, σ) is valid under the public key pk; and (2) m is never queried to the signing oracle. The experiment also returns 1 if the adversary breaks the non-substitutability, that is, produces $(\overline{pk}, m, \sigma)$ such that: (1) (m, σ) is valid under the output public key $\overline{pk}$; (2) m is queried to the signing oracle and σ is the corresponding answer; (3) $\overline{pk}$ is *not* one of the public keys under which (m, σ) is valid due to the correctness or the key-substitutability. Otherwise the experiment returns 0. Note that, it is easy to generate for the adversary such a public key $\overline{pk}$ if the adversary runs the key substitution protocol with respect to (m, σ) because of the key-substitutability.

```
Experiment $\mathbf{Exp}^{\mathtt{atk\text{-}2}}_{\mathcal{KS},\mathtt{A}}(\lambda)$
  $\mathtt{cp} \leftarrow \mathtt{Setup}(1^\lambda)$;  $(pk_0, m^*, \sigma^*, \mathrm{St}) \leftarrow \mathtt{A}_1(\mathtt{cp})$;
  $(\mathrm{St}; pk, sk) \leftarrow \langle \mathtt{A}_2(\mathrm{St}), \mathtt{KSA2} \rangle(\mathtt{cp}, pk_0, m^*, \sigma^*)$;
  If $\mathtt{Ver}(\mathtt{cp}, pk, m^*, \sigma^*) = 0$ then return 0;
  $(m, \sigma) \leftarrow \mathtt{A}_3{}^{\langle \mathtt{KSA1}(sk), \mathtt{A}_3 \rangle(\mathtt{cp}, pk, \cdot, \cdot), \mathtt{Sig}(\mathtt{cp}, pk, sk, \cdot)}(\mathrm{St}, pk)$;
  If $\mathtt{atk} = \mathtt{uf}$ and $\mathtt{Ver}(\mathtt{cp}, pk, m, \sigma) = 1$ and $(m, \cdot) \notin L_s$ then return 1;
  Else if $\mathtt{atk} = \mathtt{ns}$ and $\mathtt{Ver}(\mathtt{cp}, \overline{pk}, m, \sigma) = 1$ and $(m, \sigma) \in L_s$
    and $\overline{pk} \notin L_{pk}$ then return 1;
  Else return 0;
```

Concerning the attack against a substitute signer, we define the advantage of the adversary $\mathtt{A}$

on the signature scheme $\mathcal{KS}$ by the probability:

$$\mathbf{Adv}^{\texttt{atk-2}}_{\mathcal{KS},\mathtt{A}}(\lambda) = \Pr[\mathbf{Exp}^{\texttt{atk-2}}_{\mathcal{KS},\mathtt{A}}(\lambda) = 1].$$

Definition 5 (unforgeability for a substitute signer)**.** *We say that a key-substitutable signature scheme $\mathcal{KS}$ has (t, ϵ, q_s, q_k)-unforgeability for a substitute signer if there is no adversary* A *such that $\mathbf{Adv}^{\texttt{uf-2}}_{\mathcal{KS},\mathtt{A}}(\cdot)$ is at least $\epsilon(\cdot)$; the number of* A*'s signing queries is bounded by q_s;* A *can run the key substitution protocol with respect to q_k message and signature pairs in L_s; and* A*'s running time is bounded by t.*

Definition 6 (non-substitutability for a substitute signer)**.** *we say that a key-substitutable signature scheme $\mathcal{KS}$ has $(t, \epsilon, q_s, q_k, q_k)$-non-substitutability for a substitute signer if there is no adversary* A *such that $\mathbf{Adv}^{\texttt{ns-2}}_{\mathcal{KS},\mathtt{A}}(\cdot)$ is at least $\epsilon(\cdot)$; the number of* A*'s signing queries is bounded by q_s;* A *can run the key substitution protocol with respect to q_k message and signature pairs in L_s; and* A*'s running time is bounded by t.*

4.3 Additional security requirement

Let A be the original signer of (m_A, σ_A) and B a substitute signer of (m_A, σ_A). In some cases, for example in key registration, the indistinguishability that B's public (and secret) key pair is generated by the key substitution protocol or not is required. Of cause, a person who knows (m_A, σ_A) can see that the public keys of A and B have some relation, that is, (m_A, σ_A) is valid under both the public keys of A and B. However, a person who does not know (m_A, σ_A) may distinguish whether there exists (m, σ) such that (m, σ) is valid under both public keys of A and B. Furthermore, even if a person knows (m_A, σ_A), he may not be able to judge which of A and B is the original signer. Our construction described in Section 5 have both properties.

5 The Construction

In this section, we construct a key-substitutable signature scheme $\mathcal{KS}_1$ based on the ElGamal signature scheme [5, 9].

First, we give the idea of our construction $\mathcal{KS}_1$. Concerning (m, σ) produced by a certain user A, we want to prevent the key substitution attacks, where users other than the original signer can freely produce a substitute public key on (m, σ). Note that we might have some trouble on the construction of the key substitution protocol, when, for each message and signature pair (m, σ), there exist only one public key pk such that (m, σ) is valid under pk. One such example is the ElGamal signature scheme described in Section 2.

Roughly, in the key substitution attack, an attacker produces a substitute public key corresponding to a signature. On the other hand, in the forgery of signatures, an attacker produces a signature corresponding to a public key. From this point of view, the public key and the signature can be considered as a kind of symmetric elements. In our scheme, we change the verification equation of the ElGamal signature scheme into the one such that a public key and a signature are symmetric. The verification equation is as follows: $g^{H(m,r,y)} \equiv y^{r\alpha} r^{ys} \pmod p$ where a public key is (y, α) and a signature is (r, s).

This is not enough for our construction. We see the equation's exponent $H(m, r, y) \equiv xr\alpha + kys \pmod q$. Let (m, r, s) be a message and signature pair produced by the original signer A who has the public and secret key (y, x). We assume that A allows a substitute signer B to possess a public and secret key $(\overline{y}, \overline{\alpha}, \overline{x})$ such that (m, r, s) is valid under $(\overline{y}, \overline{\alpha})$, that is, $H(m, r, \overline{y}) \equiv \overline{x}r\overline{\alpha} + k\overline{y}s \pmod q$. Unfortunately, the substitute signer B can compute k and also the original signer A's secret key x. Therefore, we divide the secret key x into two part x_1 and x_2, and the signature is produced by using the "temporary" secret key $z = x_1 + cx_2$ where c is changed for each time and each user. We require that it is infeasible for a substitute signer B to forge the original signer A's signature even if the substitute signer B knows the temporary secret key $z = x_1 + cx_2$. Since B cannot compute a different value $z' = x_1 + c'x_2$ to sign a different message unless B knows x_1 and x_2.

We further modify this form for our construction. We assume that the original signer A allows a substitute signer B to possess two substitute public and secret key pair $(x_1', x_2', y_1', y_2', \alpha')$ on (m', r', s') and $(x_1'', x_2'', y_1'', y_2'', \alpha'')$ on (m'', r'', s'') where (m', r', s') and (m'', r'', s'') are produced by the original signer A. Then the substitute signer B can compute k' and k'' where $r' = g^{k'} \bmod p$ and $r'' = g^{k''} \bmod p$. The substitute signer B can compute also the original signer A's secret key x_1 and x_2. Therefore, we divide r's exponent k into two part k_1 and k_2 and use $k_1 + dk_2$ where d is changed for each time and each user. We require that it is infeasible for the substitute signer B to produce another public key $(\overline{y_1}, \overline{y_2}, \overline{\alpha})$ such that (m, r_1, r_2, s) is valid under $(\overline{y_1}, \overline{y_2}, \overline{\alpha})$ even if B knows $k_1 + dk_2$. Since B needs a different value $k_1 + d'k_2$ to produce such a public key. In our construction, we append the random bit string w to the public key.

With the idea described above, we construct the key-substitutable signature scheme $\mathcal{KS}_1$. The scheme employs hash functions H mapping to $\mathbb{Z}_q$ and h_1, h_2, h_3 mapping to $\mathbb{Z}_q^*$, which are modeled as the random oracles in the security proofs.

$\texttt{Setup}(1^\lambda)$ This randomized algorithm takes as input a security parameter 1^λ, and proceeds as follows. It selects λ-bit prime p such that $p = 2q + 1$, q is prime, and $q \equiv 3 \pmod 4$. It selects an element $g \in \mathbb{Z}_p^*$ whose order is q. The common parameter is $\texttt{cp} = (g, p)$.

$\texttt{Gen}(\texttt{cp})$ This randomized algorithm takes as input a common parameter $\texttt{cp} = (g, p)$, and proceeds as follows. It selects $x_1 \overset{R}{\longleftarrow} \mathbb{Z}_q$, $x_2 \overset{R}{\longleftarrow} \mathbb{Z}_q$, $\alpha \overset{R}{\longleftarrow} \mathbb{Z}_q^*$, and $w \overset{R}{\longleftarrow} \{0,1\}^\lambda$ and computes $y_1 \leftarrow g^{x_1} \bmod p$ and $y_2 \leftarrow g^{x_2} \bmod p$. The public key is $pk = (y_1, y_2, \alpha, w)$. The secret key is $sk = (x_1, x_2)$.

$\texttt{Sig}(\texttt{cp}, pk, sk, m)$ This randomized algorithm takes as input a common parameter $\texttt{cp} = (g, p)$, a public key $pk = (y_1, y_2, \alpha, w)$, a secret key is $sk = (x_1, x_2)$, and a message m, and proceeds as follows. It selects $k_1 \overset{R}{\longleftarrow} \mathbb{Z}_q^*$ and computes $r_1 \leftarrow g^{k_1} \bmod p$, $k_2 \leftarrow h_3(r_1, x_1, x_2)$, $r_2 \leftarrow g^{k_2} \bmod p$, $c \leftarrow h_1(m, r_1, r_2, y_1, y_2, w)$, $d \leftarrow h_2(m, r_1, r_2, y_1, y_2, w)$, $y \leftarrow y_1 y_2^c \bmod p$, $r \leftarrow r_1 r_2^d \bmod p$, and $s \leftarrow (H(c, r, y) - (x_1 + cx_2) r\alpha)(y(k_1 + dk_2))^{-1} \bmod q$. The signature is $\sigma = (r_1, r_2, s)$.

$\texttt{Ver}(\texttt{cp}, pk, m, \sigma)$ This deterministic algorithm takes as input a common parameter $\texttt{cp} = (g, p)$, a public key $pk = (y_1, y_2, \alpha, w)$, a message m, and a signature $\sigma = (r_1, r_2, s)$, and proceeds as follows. It computes $c \leftarrow h_1(m, r_1, r_2, y_1, y_2, w)$, $d \leftarrow h_2(m, r_1, r_2, y_1, y_2, w)$, $y \leftarrow y_1 y_2^c \bmod p$, and $r \leftarrow r_1 r_2^d \bmod p$. It checks $g^{H(c,r,y)} = y^{r\alpha} r^{ys} \pmod p$ (and $y_1^q \equiv y_2^q \equiv r_1^q \equiv r_2^q \equiv 1 \pmod p$). It returns 1 if it is true, otherwise returns 0.

$\langle \texttt{KSA1}(sk), \texttt{KSA2} \rangle (\texttt{cp}, pk, m, \sigma)$ This protocol takes as common input $(\texttt{cp}, pk, m, \sigma) = ((g, p), (y_1, y_2, \alpha, w), m, (r_1, r_2, s))$ and $\texttt{KSA1}$ takes as auxiliary input $sk = (x_1, x_2)$.

$\texttt{KSA2}$*'s first step.* $\texttt{KSA2}$ selects $\overline{x_1} \overset{R}{\longleftarrow} \mathbb{Z}_q$ and $\overline{x_2} \overset{R}{\longleftarrow} \mathbb{Z}_q$ and computes $\overline{y_1} \leftarrow g^{\overline{x_1}} \bmod p$ and $\overline{y_2} \leftarrow g^{\overline{x_2}} \bmod p$. $\texttt{KSA2}$ then sends $(\overline{y_1}, \overline{y_2})$ to $\texttt{KSA1}$.

$\texttt{KSA1}$*'s first step.* $\texttt{KSA1}$ selects $\overline{w} \overset{R}{\longleftarrow} \{0,1\}^\lambda$ and sends $\overline{w}$ to $\texttt{KSA2}$.

$\texttt{KSA2}$*'s second step.* $\texttt{KSA2}$ computes $\overline{c} \leftarrow h_1(m, r_1, r_2, \overline{y_1}, \overline{y_2}, \overline{w})$, and $z \leftarrow \overline{x_1} + \overline{cx_2} \bmod q$. If $z \bmod q = 0$ then $\texttt{KSA2}$ outputs $\perp$, otherwise, $\texttt{KSA2}$ sends z to $\texttt{KSA1}$.

$\texttt{KSA1}$*'s second step.* If $\texttt{KSA1}$ receives z, computes $\overline{c} \leftarrow h_1(m, r_1, r_2, \overline{y_1}, \overline{y_2}, \overline{w})$ and $\overline{y} \leftarrow \overline{y_1 y_2}^{\overline{c}} \bmod p$. (If $\overline{y} \neq g^z \bmod p$ then $\texttt{KSA1}$ sends $\perp$ to $\texttt{KSA2}$.) $\texttt{KSA1}$ computes $c \leftarrow h_1(m, r_1, r_2, y_1, y_2, w)$, $y \leftarrow y_1 y_2^c \bmod p$, $\nu \leftarrow (H(c, r, y) - (x_1 + cx_2) r\alpha) y^{-1} \bmod q$, $d \leftarrow h_2(m, r_1, r_2, y_1, y_2, w)$, $\overline{d} \leftarrow h_2(m, r_1, r_2, \overline{y_1}, \overline{y_2}, \overline{w})$, $k_2 \leftarrow h_3(r_1, x_1, x_2)$, $\overline{\nu} \leftarrow \nu + (\overline{d} - d) k_2 s$, $\overline{r} = r_1 r_2^{\overline{d}} \bmod p$, and $\overline{\alpha} \leftarrow (H(\overline{c}, \overline{r}, \overline{y}) - \overline{\nu y})(\overline{r} z)^{-1} \bmod q$. $\texttt{KSA1}$ sends $\overline{\alpha}$ to $\texttt{KSA2}$.

`KSA2`*'s third step.* Upon receiving $\overline{\alpha}$, if $\overline{\alpha} = 0$ then `KSA2` outputs $\perp$, otherwise, `KSA2` outputs the public key $\overline{pk} = (\overline{y_1}, \overline{y_2}, \overline{\alpha}, \overline{w})$ and the secret key $\overline{sk} = (\overline{x_1}, \overline{x_2})$.

It is easy to see that the key-substitutable signature scheme $\mathcal{KS}_1$ is correct and has the key-substitutability. We show that the key-substitutable signature scheme $\mathcal{KS}_1$ satisfies all of the security requirements as shown in Theorem 7, 8, 9, and 10. The proofs of Theorem 7, 8, 9, and 10 are in the Appendices of the full version of the paper.

Theorem 7. *If the ElGamal signature scheme is (t, ϵ, q_H)-EU-KOA, then the key-substitutable signature scheme $\mathcal{KS}_1$ has $(t', \epsilon', q_s, q_k, q_H, q_{h_1}, q_{h_2}, q_{h_3})$-unforgeability 1, where $\epsilon \geq \epsilon' - \delta$, $\delta = (\frac{2(q_{h_1}+q_{h_2}+2q_k+2q_s)}{q^2} + \frac{q_H+q_s}{q}) \cdot q_s + \frac{q_{h_1}+q_{h_2}+2q_k+2q_s}{2^\lambda} \cdot q_k$, and $t = (q_{h_1} + q_{h_2} + q_{h_3})O(\lambda) + (q_H + q_s + q_k + 2)O(\lambda^3) + t'$. Here q_H, q_{h_1}, q_{h_2}, and q_{h_3} are the number of the queries to the random oracle H, h_1, h_2, and h_3 respectively.*

Theorem 8. *If the ElGamal signature scheme is (t, ϵ, q_H)-EU-KOA, then the key-substitutable signature scheme $\mathcal{KS}_1$ has $(t', \epsilon', q_s, q_k, q_H, q_{h_1}, q_{h_2}, q_{h_3})$-non-substitutability 1, where $\epsilon \geq \frac{\epsilon'-\delta}{q_s q_{h_2}}$, $\delta = (\frac{2(q_{h_1}+q_{h_2}+2q_k+2q_s)}{q^2} + \frac{q_H+q_s}{q}) \cdot q_s + \frac{q_{h_1}+q_{h_2}+2q_k+2q_s}{2^\lambda} \cdot q_k$, and $t = (q_{h_1}+q_{h_2}+q_{h_3})O(\lambda)+q_H O(\lambda^2)+(q_s + q_k + 2)O(\lambda^3) + t'$. Here q_H, q_{h_1}, q_{h_2}, and q_{h_3} are the number of the queries to the random oracle H, h_1, h_2, and h_3 respectively.*

Theorem 9. *If the ElGamal signature scheme is (t, ϵ, q_H)-EU-KOA, then the key-substitutable signature scheme $\mathcal{KS}_1$ is $(t', \epsilon', q_s, q_k, q_H, q_{h_1}, q_{h_2}, q_{h_3})$-unforgeability 2, where $\epsilon \geq \frac{\epsilon'}{q_{h_1}} - \delta$, $\delta = (\frac{2(q_{h_1}+q_{h_2}+2q_k+2q_s)}{q^2} + \frac{q_H+2q_s+q_{h_1}+q_k}{q}) \cdot q_s + \frac{2q_{h_1}+q_{h_2}+3q_k+3q_s}{2^\lambda} \cdot q_k + \frac{q_{h_1}+q_s+q_k}{q} \cdot q_{h_1}$, and $t = (q_{h_1}+q_{h_2}+q_{h_3})O(\lambda) + (q_H + q_s + q_k + 2)O(\lambda^3) + t'$. Here q_H, q_{h_1}, q_{h_2}, and q_{h_3} are the number of the queries to the random oracle H, h_1, h_2, and h_3 respectively.*

Theorem 10. *If the ElGamal signature scheme is (t, ϵ, q_H)-EU-KOA, then the key-substitutable signature scheme $\mathcal{KS}_1$ is $(t', \epsilon', q_s, q_k, q_H, q_{h_1}, q_{h_2}, q_{h_3})$-non-substitutability 2, where $\epsilon \geq (\frac{\epsilon'}{q_{h_1}} - \delta)\frac{1}{q_s q_{h_2}}$, $\delta = (\frac{2(q_{h_1}+q_{h_2}+2q_k+2q_s)}{q^2} + \frac{q_H+q_s}{q}) \cdot q_s + \frac{q_{h_1}+q_{h_2}+2q_k+2q_s}{2^\lambda} \cdot q_k$, and $t = (q_{h_1} + q_{h_2} + q_{h_3})O(\lambda) + (q_H + q_s + q_k + 2)O(\lambda^3) + t'$. Here q_H, q_{h_1}, q_{h_2}, and q_{h_3} are the number of the queries to random oracle H, h_1, h_2, and h_3 respectively.*

Extensions. Recall that the verification equation's exponent is $H(m, r, y) = (x_1+cx_2)r\alpha+(k_1+dk_2)ys \bmod q$ where $y_1 = g^{x_1} \bmod p$, $y_2 = g^{x_2} \bmod p$, $r_1 = g^{k_1} \bmod p$, and $r_2 = g^{k_2} \bmod p$. If the adversary obtains only one public and secret key pair (pk, sk) per each message and signature pair (m, σ), the adversary can know only $k_1+d'k_2$ and cannot know any of k_1, k_2, x_1, and x_2. Therefore, if the adversary obtains only one key pair (pk, sk) per each message and signature pair, the key-substitutable signature scheme $\mathcal{KS}_1$ has the unforgeability and the non-substitutability. Now, we extends the form of signature to $(r_1, r_2, \cdots, r_N, s)$ and the verification equation's exponent to $H(m, r, y) = (x_1 + cx_2)r\alpha + (k_1 + d_1k_2 + \cdots + d_{N-1}k_N)ys \bmod q$. In this case, even if the adversary obtains at most $N - 1$ key pairs per each pair (m, σ) of message and signature, the adversary cannot know any of $k_1, \cdots, k_N$, x_1, and x_2. Therefore, the extended $\mathcal{KS}_1$ satisfies the unforgeability and the non-substitutability. Note that this extension is *not* necessary for our application to certified-signature in Section 6.

6 An Application to Certified Signature

Public key cryptosystem implicitly relies on the existence of a public-key infrastructure (PKI), where each user has a public and secret key pair for the cryptosystem, and an associated user with a public key is publicly available. The designers of public-key cryptosystems always define how the public and the secret keys are generated and used, but almost never carefully specify how to bind keys with user identities. In particular, the security of the combination of a key-registration protocol with existing encryption or signature schemes does not immediately follow from the security of the individual components.

Boldyreva, Fischlin, Palacio, and Warinschi [3] studied PKIs with respect to security of encryption and digital signature schemes. Their security notions are against an adversary with broad capabilities that take into account threats arising from the key-registration protocol, the presence of several parties, including the users and the (possibly corrupt) certification authority. The models directly capture settings where users have multiple public keys, and where keys have additional attributes, such as an expiration date.

Model of the PKI. Users register public-keys by interacting with a certification authority (CA) on public, authenticated channels. After registration, parties can sign messages using the secret keys associated to the public key the have registered. Signatures can then be verified, the verification process involves both the public key of the CA and that of the user. Boldyreva, Fischlin, Palacio, and Warinschi [3] call the model of signature schemes with PKI as *certified-signature scheme*.

The adversary attempts a forgery by outputting a user identity $\mathtt{ID}$, a public key pk, a message m and a signature σ. Roughly, the adversary wins if the signature σ is valid on m under the chosen public key pk, and either (1) the honest user $\mathtt{ID}$ has registered the public key pk and has not signed the message m, (2) the public key pk has not been registered with an honest CA, or (3) the same public key pk has been registered by another honest user $\mathtt{ID}'$ with an honest CA. The condition (1) corresponds to the existential unforgeability for standard digital signature schemes, and we require that it holds even if the CA is corrupt. The condition (2) guarantees that signatures for keys that are not bound to identities of the users (i.e., "outside of the PKI") are not valid on the assumption that CA is honest. The condition (3) prevents attacks where for example a malicious user claims authorship of a message signed by another user on the assumption that CA is honest.

Weakening the assumption of CA's trust. Weakening the assumption of CA's trust is one of important issues on PKI. The security definition in [3] guarantees the condition (3) on the assumption that the CA is honest. With respect to the condition (3), our certified signature schemes based on key-substitutable signatures guarantee the security even if the CA is corrupted. In this section, we consider the stronger security definition of certified-signature schemes. We review the model and security notions of certified-signature schemes, propose our certified-signature schemes based on key-substitutable signature scheme, and show that the traditional certified-signature schemes is *not* secure if CA is corrupted, with respect to the condition (3).

Overview of certified-signature schemes We first review the model of certified-signature schemes [3]. A certified-signature scheme $\mathcal{CS} = (\mathcal{G}, \mathcal{K}, (\mathcal{C}, \mathcal{U}), \mathcal{S}, \mathcal{V})$, is specified by a parameter-generation algorithm $\mathcal{G}$, a key generation algorithm $\mathcal{K}$, a public-key registration protocol $(\mathcal{C}, \mathcal{U})$, a signing algorithm $\mathcal{S}$, and a verification algorithm $\mathcal{V}$.

The parameter-generation algorithm $\mathcal{G}$ is a randomized algorithm which takes input 1^λ, where λ is the security parameter, and outputs a global parameter $\mathtt{cp}$ shared among all parties.

The key-generation algorithm $\mathcal{K}$ is a randomized algorithm which takes input $\mathtt{cp}$, and outputs a public and secret key pair (pk_{CA}, sk_{CA}) of the certificate authority (CA).

The public-key registration protocol $(\mathcal{C}, \mathcal{U})$ is a pair of interactive randomized algorithms. $\mathcal{C}$ takes input CA's secret key sk_{CA}. $\mathcal{U}$ takes input the identity $\mathtt{ID}$ of a user and the CA's public key pk_{CA} corresponding sk_{CA}. As result of the interaction, $\mathcal{C}$ outputs $(\texttt{ID}, pk, \texttt{cert})$, where pk is a public key and $\texttt{cert}$ is an issued certificate. The local output of $\mathcal{U}$ is $(\texttt{ID}, pk, sk, \texttt{cert})$, where sk is a secret key that user $\texttt{ID}$ uses to produce a signature. We denote this as $((\texttt{ID}, pk, \texttt{cert}), (\texttt{ID}, pk, sk, \texttt{cert})) \leftarrow (\mathcal{C}(sk_{CA}), \mathcal{U}(\texttt{ID}, pk_{CA}))$. Either party can quit the execution prematurely, in which case the output of the party is set to $\perp$.

The signing algorithm $\mathcal{S}$ is a randomized algorithm which takes as input $(\texttt{ID}, sk, \texttt{cert}, pk_{CA}, m)$, where m is a message, and outputs a signature σ. We denote this as $\sigma \leftarrow \mathcal{S}(\texttt{ID}, sk, \texttt{cert}, pk_{CA}, , m)$.

The verification algorithm $\mathcal{V}$ is a deterministic verification algorithm which takes as input $(\texttt{ID}, pk, \texttt{cert}, pk_{CA}, m, \sigma)$, and outputs 0 or 1. We denote this as $0/1 \leftarrow \mathcal{V}(\texttt{ID}, pk, \texttt{cert}, pk_{CA}, m, \sigma)$.

Security notions. We next review the definition of certified-signature schemes [3]. We associate a certified-signature scheme $\mathcal{CS} = (\mathcal{G}, \mathcal{K}, (\mathcal{C}, \mathcal{U}), \mathcal{S}, \mathcal{V})$, an adversary $\texttt{A}$, and security parameter λ with the following experiment. The experiment maintains two virtual lists *RegListPub* and *RegListSec*. $\texttt{A}$ knows the elements of *RegListPub* but not those of *RegListSec*. In the beginning of the experiment, the adversary $\texttt{A}$ is given as input $\texttt{cp} \leftarrow \mathcal{G}(1^\lambda)$ and then make the following request or queries:

- Corruption of certification authority: First, $\texttt{A}$ decides whether to corrupt the CA or not. If $\texttt{A}$ corrupts then $\texttt{A}$ chooses the CA's public key pk_{CA}, else $\texttt{A}$ is given a CA's public key pk_{CA} where $(pk_{CA}, sk_{CA}) \leftarrow \mathcal{K}(\texttt{cp})$.
- Registering keys of users: During the experiment, $\texttt{A}$ can specify a user $\texttt{ID}$ from the set of identities, to initiate a run of the public-key registration protocol with the honest or corrupt certification authority. If this is the first time the user $\texttt{ID}$ is activated then $\texttt{A}$ first decides whether to corrupt this user or not. In the execution with the CA, we assume that at least one party is honest. At the end of the execution, $\mathcal{C}$ outputs values $(\texttt{ID}, pk, \texttt{cert})$ and $\mathcal{U}$ outputs values $(\texttt{ID}', pk', sk', \texttt{cert}')$. If $\mathcal{U}$ is honest, we store $(\texttt{ID}, pk, \texttt{cert})$ in *RegListPub* and $(\texttt{ID}', pk', sk', \texttt{cert}')$ in *RegListSec*. If only $\mathcal{C}$ is honest, we store $(\texttt{ID}, pk, \texttt{cert})$ in *RegListPub*. If one of the parties is dishonest or stops prematurely then $\perp$ is stored in the corresponding list. Notice that all steps in the experiment, including steps of this interactive protocol may be arbitrarily interleaved.
- Signature queries: $\texttt{A}$ can make signature requests to a universal signing oracle $\mathcal{US}^{pk_{CA}}$: on a query $(ID, pk, \texttt{cert}, m)$ the oracle, if user $\texttt{ID}$ is honest, it looks up the corresponding entry $(ID, pk, sk, \texttt{cert})$ in *RegListSec* and returns to $\texttt{A}$ a signature $\mathcal{S}(\texttt{ID}, sk, \texttt{cert}, pk_{CA}, m)$. Otherwise, the answer of the oracle is $\perp$.

Eventually, $\texttt{A}$ stops and outputs an attempted forgery $(\texttt{ID}, pk, \texttt{cert}, m, \sigma)$. The experiment returns 1 if $\mathcal{V}(pk_{CA}, \texttt{ID}, pk, \texttt{cert}, m, \sigma)$ and the following conditions are satisfied:

1. $\texttt{ID}$ is honest, and no valid signing query $(\texttt{ID}, pk, \texttt{cert}', m)$ was made for any $\texttt{cert}'$, or
2. CA is honest and $(\texttt{ID}, pk, \texttt{cert}') \notin$ *RegListPub*, for any $\texttt{cert}'$, or
3. CA is honest and $(\texttt{ID}', pk, \texttt{cert}') \in$ *RegListPub*, for some honest user $\texttt{ID}' \neq \texttt{ID}$,

otherwise, it returns 0.

In this paper, we reduce the limitation "CA is honest" on the third winning condition. That is, we relax the third condition as follows:

3'. $(\texttt{ID}', pk, \texttt{cert}') \in$ *RegListPub*, for some honest user $\texttt{ID}' \neq \texttt{ID}$.

We first show that the traditional certified-signature scheme is not secure on the stronger security definition. Second, we show that our certified-signature scheme based on key-substitutable signature guarantee the security definition.

Traditional certified-signature schemes. Here we review the traditional approach to certified signature, where the users' public keys are certified by the CA using a signature scheme. Users produce signatures by using the secret keys associated with their certificated public keys. Signature verification consists of verifying the signature of the user and the validity of the certificates for the users' public-keys.

Let $\mathcal{DS} = (\mathcal{G}_0, \mathcal{K}_0, \mathcal{S}_0, \mathcal{V}_0)$ be a standard signature scheme. The traditional certified-signature scheme $\mathcal{CS}_1 = (\mathcal{G}_1, \mathcal{K}_1, (\mathcal{C}_1, \mathcal{U}_1), \mathcal{S}_1, \mathcal{V}_1)$ is defined as follows.

- The parameter generation algorithm $\mathcal{G}_1$ on input 1^λ outputs $\texttt{cp} \leftarrow \mathcal{G}_0(1^\lambda)$.

- The key generation algorithm $\mathcal{K}_1$ on input $\texttt{cp}$ outputs $(pk_{CA}, sk_{CA}) \leftarrow \mathcal{K}_0(\texttt{cp})$.

- The public-key registration protocol $(\mathcal{C}_1(sk_{CA}),\mathcal{U}_1(\texttt{ID}, pk_{CA}))$ proceeds as follows. $\mathcal{U}_1$ generates $(pk, sk) \leftarrow \mathcal{K}_0(\texttt{cp})$ and sends to $\mathcal{C}_1$. $\mathcal{C}_1$ sends a random "challenge" message $M' \leftarrow \{0,1\}^\lambda$ to $\mathcal{U}_1$. $\mathcal{U}_1$ computes $\sigma' \leftarrow \mathcal{S}_0(sk, 0||M')$ and sends it to $\mathcal{C}_1$. If $\mathcal{V}_0(pk, 0||M', \sigma') = 1$ then $\mathcal{C}_1$ computes $\texttt{cert} \leftarrow \mathcal{S}_0(sk_{CA}, (\texttt{ID}, pk))$, sends $\texttt{cert}$ to $\mathcal{U}_1$ and outputs $(\texttt{ID}, pk, \texttt{cert})$. The user outputs $(\texttt{ID}, sk, pk, \texttt{cert})$.

- The signing algorithm $\mathcal{S}_1$ on input $(\texttt{ID}, sk, \texttt{cert}, pk_{CA}, m)$ outputs $\sigma \leftarrow \mathcal{S}_0(sk, 1||m)$.

- The verification algorithm $\mathcal{V}_1$ on input $(\texttt{ID}, pk, \texttt{cert}, pk_{CA}, m)$ outputs 1 if $\mathcal{V}_0(pk_{CA}, (\texttt{ID}, pk), \texttt{cert}) = 1$ and $\mathcal{V}_0(pk, 1||m, \sigma) = 1$. Otherwise, it outputs 0.

Boldyreva, Fischlin, Palacio, and Warinschi [3] show that the traditional certified-signature scheme $\mathcal{CS}_1$ is secure if a standard signature scheme $\mathcal{DS}$ is existentially unforgeable under chosen message attack [7]. However, we conclude that $\mathcal{CS}_1$ is not secure on our stronger security definition. Since CA can easily produce, for any honest user who has certified $(\texttt{ID}, pk)$, a certificate $\texttt{cert}'$ for another $\texttt{ID}'$ such that $\mathcal{V}_1(pk_{CA}, (\texttt{ID}', pk), \texttt{cert}') = 1$.

A certified-signature scheme based on key-substitutable signature. In contrast to the traditional certified-signature scheme, our certified-signature scheme based on key-substitutable signature satisfies the stronger security definition of certified signature. We describe our certified-signature scheme $\mathcal{CS}_2 = (\mathcal{G}_2, \mathcal{K}_2, (\mathcal{C}_2, \mathcal{U}_2), \mathcal{S}_2, \mathcal{V}_2)$. based on a key-substitutable signature scheme $\mathcal{KS} = (\texttt{Setup}, \texttt{Gen}, \texttt{Sig}, \texttt{Ver}, (\texttt{KSA1}, \texttt{KSA2}))$.

- The parameter generation algorithm $\mathcal{G}_2$ on input 1^λ outputs $\texttt{cp} \leftarrow \texttt{Setup}(1^\lambda)$.

- The key generation algorithm $\mathcal{K}_2$ on input $\texttt{cp}$ outputs $(pk_{CA}, sk_{CA}) \leftarrow \texttt{Gen}(\texttt{cp})$.

- The public-key registration protocol $(\mathcal{C}_2(sk_{CA}),\mathcal{U}_2(\texttt{ID}, pk_{CA}))$ proceeds as follows. $\mathcal{U}_2$ sends the identity $\texttt{ID}$ to $\mathcal{C}_2$. $\mathcal{C}_2$ responds a signature $\texttt{cert} \leftarrow \texttt{Sig}(\texttt{cp}, pk_{CA}, sk_{CA}, 0||\texttt{ID})$ to $\mathcal{U}_2$. If $\texttt{Ver}(\texttt{cp}, pk_{CA}, 0||\texttt{ID}, \texttt{cert}) = 1$, then $\mathcal{C}_2$ and $\mathcal{U}_2$ run the key substitution protocol with respect to $(0||\texttt{ID}, \texttt{cert})$ where $\mathcal{C}_2$ plays the role of $\texttt{KSA1}$ and $\mathcal{U}_2$ plays the role of $\texttt{KSA2}$. That is, $(pk; pk, sk) \leftarrow \langle \texttt{KSA1}(sk_{CA}), \texttt{KSA2}\rangle(\texttt{cp}, pk, 0||\texttt{ID}, \texttt{cert})$. Finally, $\mathcal{C}_2$ outputs $(\texttt{ID}, pk, \texttt{cert})$. $\mathcal{U}_2$ outputs $(\texttt{ID}, pk, sk, \texttt{cert})$.

- The signing algorithm $\mathcal{S}_2$ on input $(\texttt{ID}, sk, \texttt{cert}, pk_{CA}, m)$ outputs $\sigma \leftarrow \texttt{Sig}(\texttt{cp}, pk, sk, 1||m)$.

- The verification algorithm $\mathcal{V}_2$ on input $\mathcal{C}_2(\texttt{ID}, pk, \texttt{cert}, pk_{CA}, m, \sigma)$ outputs 1 if $\texttt{Ver}(\texttt{cp}, pk_{CA}, 0||\texttt{ID}, \texttt{cert}) = 1$ and $\texttt{Ver}(\texttt{cp}, pk, 0||\texttt{ID}, \texttt{cert}) = 1$ and $\texttt{Ver}(\texttt{cp}, pk, 1||m, \sigma) = 1$. Otherwise, it outputs 0.

The proof idea is to transform an attacker against the certified-signature scheme $\mathcal{CS}_2$ into one against the key-substitutable signature scheme $\mathcal{KS}$. An adversary who breaks the winning condition (1) and (3') immediately yields a forgery for a substitute signer of the key-substitutable signature scheme. It is due to the impossibility for CA to produce $(1||M', \sigma')$ or $(0||\mathtt{ID}', \mathtt{cert}')$ which is valid under honest user's public key pk. An adversary who breaks the winning condition (2) immediately yields a forgery for the original signer of the key-substitutable signature scheme. It is due to the impossibility for a user to produce $(0||\mathtt{ID}', \mathtt{cert}')$ which is valid under CA's public key pk_{CA}.

7 Conclusion

We have introduced the key-substitutable signature scheme, and formalized its security requirement. Furthermore, we have proposed a construction based on the ElGamal signature scheme, and proved that this construction satisfies the security requirements. Finally, we have suggested that key-substitutable signature can be applied to certified signature with higher security requirements.

References

[1] 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.

[2] 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.

[3] Boldyreva, A., Fischlin, M., Palacio, A., and Warinschi, B. A closer look at pki: Security and efficiency. In *Public Key Cryptography* (2007), T. Okamoto and X. Wang, Eds., vol. 4450 of *Lecture Notes in Computer Science*, Springer, pp. 458–475.

[4] Delfs, H., and Knebl, H. *Introduction to Cryptography: Principles and Applications.* Springer, 2002.

[5] Gamal, T. E. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO* (1984), pp. 10–18.

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

[7] Goldwasser, S., Micali, S., and Rivest, R. L. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput. 17*, 2 (1988), 281–308.

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

[9] Pointcheval, D., and Stern, J. Security proofs for signature schemes. In *EUROCRYPT* (1996), pp. 387–398.

[10] 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.

[11] 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.

[12] 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 Proof of Theorems

A.1 Proof of Theorem 7

Proof. Given an algorithm A that breaks the $(t', \epsilon', q_s, q_k, q_H, q_{h_1}, q_{h_2}, q_{h_3})$-unforgeability 1 of $\mathcal{KS}_1$, we construct an algorithm B that breaks the (t, ϵ, q_H)-EU-KOA of the ElGamal signature scheme.

B is given a public key (g, y_2), where $g \in \mathbb{Z}_p^*$ is an element whose order is q and $y_2 \in \langle g \rangle$, and can access to the random oracle H'.

Initialization of A. B chooses $x_1, \alpha \overset{R}{\longleftarrow} \mathbb{Z}_q^*$ and $w \overset{R}{\longleftarrow} \{0,1\}^\lambda$ and sets $y_1 = g^{x_1} \bmod p$. B then provides A the common parameter (p, g) and the public key (y_1, y_2, α, w).

Simulation of random oracle H. This random oracle takes as input a query (c, r, y) and proceeds as follows. If $H(c, r, y)$ has been determined, B answers the value. Otherwise, if $y = y_1 y_2^c \bmod p$ then B answers $(H'(c, r)c + x_1 r)\alpha \bmod q$, else B answers a random value in $\mathbb{Z}_q$.

Simulation of random oracle h_1. This random oracle takes as input a query $(m, r_1, r_2, y_1, y_2, w)$ and proceeds as follows. If $h_1(m, r_1, r_2, y_1, y_2, w)$ has been determined, B answers the value. Otherwise, B answers a random value $c \in \mathbb{Z}_q^*$.

Simulation of random oracle h_2. This random oracle takes as input a query $(m, r_1, r_2, y_1, y_2, w)$ and proceeds as follows. If $h_2(m, r_1, r_2, y_1, y_2, w)$ has been determined, B answers the value. Otherwise, B answers a random value $d \in \mathbb{Z}_q^*$.

Simulation of random oracle h_3. This random oracle takes as input a query (r_1, x_1, x_2) and proceeds as follows. If $h_3(r_1, x_1, x_2)$ has been determined, B answers the value. Otherwise, B answers a random value in $\mathbb{Z}_q^*$.

Simulation of signing oracle. In the j-th signing query from A, B proceeds as follows. In particular, we denote the j-th signing query m as m_j.

B chooses $c_j, v_j, d_j, d'_j (d_j \neq d'_j) \overset{R}{\longleftarrow} \mathbb{Z}_q^*$ and $u_j, e_j \overset{R}{\longleftarrow} \mathbb{Z}_q$. B sets $r_{2j} = g^{e_j}(y_1 y_2^{c_j})^{v_j} \bmod p$, $r_{1j} = g^{u_j} r_{2j}^{-d'_j} \bmod p$, $r_j = r_{1j} r_{2j}^{d_j} \bmod p$, $y_j = y_1 y_2^{c_j} \bmod p$, and $s_j = -r_j \alpha v_j^{-1} y_j^{-1} (d_j - d'_j)^{-1} \bmod q$. If $h_1(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, $h_2(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, or $H(c_j, r_j, y_j)$ have been determined then B outputs $\perp$ and stops, else B sets $H(c_j, r_j, y_j) = (u_j + e_j(d_j - d'_j)) y_j s_j \bmod q$, $h_1(m_j, r_{1j}, r_{2j}, y_1, y_2, w) = c_j$, and $h_2(m_j, r_{1j}, r_{2j}, y_1, y_2, w) = d_j$. Finally, B answers (r_{1j}, r_{2j}, s_j). [1]

[1] The correctness of this signing oracle:

$$\begin{aligned} y_j^{r_j \alpha} r_j^{y_j s_j} &= y_j^{r_j \alpha}((g^{u_j} r_{2j}^{-d'_j}) r_{2j}^{d_j})^{y_j s_j} \\ &= y_j^{r_j \alpha}(g^{u_j}(g^{e_j} y_j^{v_j})^{-d'_j + d_j})^{y_j(-r_j \alpha v_j^{-1} y_j^{-1}(d_j - d'_j)^{-1})} \\ &= g^{(u_j + e_j(d_j - d'_j)) y_j s_j} \\ &= g^{H(c_j, r_j, y_j)}, \end{aligned}$$

where $c_j = h(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, $d_j = h_2(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$ $r_j = r_{1j} r_{2j}^{d_j} \bmod p$, and $y_j = y_1 y_2^{c_j} \bmod p$.

Simulation of `KSA1` with respect to $(m_j, r_{1j}, r_{2j}, s_j)$ in the list L_s. In the first step of `KSA1`, `B` receives $(\overline{y_1}, \overline{y_2})$ and selects $c'_j \xleftarrow{R} \mathbb{Z}_q^*$ and $\overline{w} \xleftarrow{R} \{0,1\}^\lambda$. If $h_1(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ or $h_2(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ has been determined then `B` outputs $\perp$ and stops, else `B` sets $h_1(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w}) = c'_j$ and $h_2(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w}) = d'_j$. `B` then sends $\overline{w}$ to `B`.

In the second step of `KSA1`, `B` receives z, computes $\overline{y_j} = \overline{y_1 y_2}^{c'_j} \bmod p$, and checks $\overline{y_j} \neq g^z \bmod p$. `B` then computes $\overline{r_j} = r_{1j} r_{2j}^{d'_j} \bmod p$ and $\overline{\alpha} \leftarrow (H(c'_j, \overline{r_j}, \overline{y_j}) - u_j \overline{y_j} s_j)(\overline{r_j} z)^{-1} \bmod q$ and sends $\overline{\alpha}$ to `A`. [2]

Output. `B` obtains `A`'s output (m, r_1, r_2, s). `B` sets $r = r_1 r_2^d \bmod p$ and $s' = (y_1 y_2^c \bmod p) c^{-1} \alpha^{-1} s \bmod q$ where $c = h_1(m, r_1, r_2, y_1, y_2, w)$ and $d = h_2(m, r_1, r_2, y_1, y_2, w)$. `B` then outputs (c, r, s').

Analysis of success probability. We first show that if `A`'s output (m, r_1, r_2, s) satisfies the winning conditions then `B`'s output (c, r, s') satisfies the winning conditions. We assume the follows.

- $g^{H(c,r,y)} = y^{r\alpha} r^{ys}$ where $c = h_1(m, r_1, r_2, y_1, y_2, w)$, $d = h_2(m, r_1, r_2, y_1, y_2, w)$, $y = y_1 y_2^c \bmod p$, and $r = r_1 r_2^d \bmod p$.
- m is not queried to the signing oracle $\mathcal{S}$.

Then, we can see the following equations.

$$\begin{aligned} y_2^r r^{s'} &= y_2^r (r^{ys})^{c^{-1}\alpha^{-1}} \\ &= y_2^r (g^{H(c,r,y)} y^{-r\alpha})^{c^{-1}\alpha^{-1}} \\ &= y_2^r (g^{H(c,r,y)} (g^{x_1} y_2^c)^{-r\alpha})^{c^{-1}\alpha^{-1}} \\ &= g^{(H(c,r,y)\alpha^{-1} - x_1 r)c^{-1}} \\ &= g^{H'(c,r)}, \end{aligned}$$

where $c = h_1(m, r_1, r_2, y_1, y_2, w)$, $d = h_2(m, r_1, r_2, y_1, y_2, w)$, $y = y_1 y_2^c \bmod p$, and $r = r_1 r_2^d \bmod p$.

We next estimate the probability that `B` failures the simulation of the signing oracle, `KSA1`, and the random oracles.

In the simulation of the signing oracle, only if $h_1(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, $h_2(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, or $H(c_j, r_j, y_j)$ have been determined then `B` fails. The probability that this happens is at most $\frac{2(q_{h_1}+q_k+q_s)}{q^2} + \frac{2(q_{h_2}+q_k+q_s)}{q^2} + \frac{q_H+q_s}{q} = \frac{2(q_{h_1}+q_{h_2}+2q_k+2q_s)}{q^2} + \frac{q_H+q_s}{q}$.

In the simulation of `KSA1`, only if $h_1(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ or $h_2(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ has been determined then `B` fails. The probability that this happens is at most $\frac{q_{h_1}+q_k+q_s}{2^\lambda} + \frac{q_{h_2}+q_k+q_s)}{2^\lambda} = \frac{q_{h_1}+q_{h_2}+2q_k+2q_s}{2^\lambda}$.

In the mentioned above, we can see that `B` failures the simulation of `A`'s environment with probability at most $(\frac{2(q_{h_1}+q_{h_2}+2q_k+2q_s)}{q^2} + \frac{q_H+q_s}{q}) \cdot q_s + \frac{q_{h_1}+q_{h_2}+2q_k+2q_s}{2^\lambda} \cdot q_k$. We denote the probability as δ. And the successful simulation is properly distributed. Therefore, we can see $\epsilon \geq \epsilon' - \delta$.

Analysis of running time. `B` takes the time $O(\lambda^3)$ to the simulation of initialization, $O(\lambda^3)$ to the random oracle oracle H, $O(\lambda)$ to each random oracle oracle h_1, h_2, and h_3, $O(\lambda^3)$ to the signing oracle, $O(\lambda^3)$ to the key substitution algorithm `KSA1`, $O(\lambda^3)$ to produce output, and t' to the running time of `A`. The total is $(q_{h_1} + q_{h_2} + q_{h_3})O(\lambda) + (q_H + q_s + q_k + 2)O(\lambda^3) + t'$.

□

[2] The correctness of this simulation of `KSA1`:

$$\begin{aligned} \overline{y_j}^{\overline{r_j}\alpha} \overline{r_j}^{\overline{y_j} s_j} &= (g^z)^{\overline{r_j}((H(c'_j, \overline{r_j}, \overline{y_j}) - u_j \overline{y_j} s_j)(\overline{r_j} z)^{-1})} (g^{u_j})^{\overline{y_j} s_j} \\ &= g^{H(c'_j, \overline{r_j}, \overline{y_j})}, \end{aligned}$$

where $c'_j = h(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$, $d'_j = h_2(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ $\overline{r_j} = r_{1j} r_{2j}^{d'_j} (= g^{u_j}) \bmod p$, and $\overline{y_j} = \overline{y_1 y_2}^{c'_j} \bmod p$.

A.2 proof of Theorem 8

Proof. Given an algorithm A that breaks the $(t', \epsilon', q_s, q_k, q_H, q_{h_1}, q_{h_2}, q_{h_3})$-non-substitutability 1 of $\mathcal{KS}_1$, we construct an algorithm B that breaks the (t, ϵ, q_H)-EU-KOA of the ElGamal signature scheme.

B is given a public key (g, r^*), where $g \in \mathbb{Z}_p^*$ is an element whose order is q and $r^* \in \langle g \rangle$, and can access to the random oracle H'.

Initialization of A. If $r^* = 1$ then B finds the corresponding secret key $k^*(= 0)$, therefore, B stops. Otherwise, B proceeds as follows. B chooses $\beta \overset{R}{\longleftarrow} \{1, \cdots, q_s\}$, $\zeta \overset{R}{\longleftarrow} \{1, \cdots, q_{h_2}\}$, $b, t, x_2 \overset{R}{\longleftarrow} \mathbb{Z}_q$, $a, d^*, d_\beta, d'_\beta, c_\beta, \alpha \overset{R}{\longleftarrow} \mathbb{Z}_q^*$, and $w \overset{R}{\longleftarrow} \{0,1\}^\lambda$ and sets $y_2 = g^{x_2} \bmod p$, $y_1 = r^{*a^{-1}(d^*-d'_\beta)^{-1}} y_2^{-c_\beta} \bmod p$, $r_{2\beta} = r^{*(d^*-d'_\beta)^{-1}} g^b \bmod p$, $r_{1\beta} = r^* r_{2\beta}^{-d^*} g^t$, $y_\beta = y_1 y_2^{c_\beta} \bmod p$, $r_\beta = r_1 r_2^{d_\beta} \bmod p$, $s_\beta = -a^{-1}(d^* - d'_\beta)^{-1} r_\beta \alpha (1 + (d^* - d'_\beta)^{-1}(d_\beta - d^*))^{-1} y_\beta^{-1} \bmod q$, and $H(c_\beta, r_\beta, y_\beta) = (t + b(d_\beta - d^*)) y_\beta s_\beta$. Finally B provides A the common parameter (p, g) and the public key (y_1, y_2, α, w).

Simulation of random oracle H. This random oracle takes as input a query (c, r, y) and proceeds as follows. If $H(c, r, y)$ has been determined, B answers the value. Otherwise, if $(c, r, y) = (c, r^* g^t, y)$ then B answers $H'(c, y) s_\beta - ty \bmod q$, else answers a random value in $\mathbb{Z}_q$.

Simulation of random oracle h_1. This random oracle takes as input a query $(m, r_1, r_2, y_1, y_2, w)$ and proceeds as follows. If $h_1(m, r_1, r_2, y_1, y_2, w)$ has been determined, B answers the value. Otherwise, B answers a random value $c \in \mathbb{Z}_q^*$.

Simulation of random oracle h_2. In the l-th query from A, B receives a query $(m, r_1, r_2, y_1, y_2, w)$ and proceeds as follows.

If $h_2(m, r_1, r_2, y_1, y_2, w)$ has been determined, B answers the value. Otherwise, if $l = \zeta$ then B answers d^*, else B answers a random value $d \in \mathbb{Z}_q^*$.

Simulation of random oracle h_3. This random oracle takes as input a query (r_1, x_1, x_2) and proceeds as follows. If $h_3(r_1, x_1, x_2)$ has been determined, B answers the value. Otherwise, B answers a random value in $\mathbb{Z}_q^*$.

Simulation of signing oracle. In the j-th signing query from A, B proceeds as follows. In particular, we denote the j-th signing query m as m_j.

If $j \neq \beta$, B simulates as the same way to the proof of Theorem 7. B chooses $c_j, v_j, d_j, d'_j (d_j \neq d'_j) \overset{R}{\longleftarrow} \mathbb{Z}_q^*$ and $u_j, e_j \overset{R}{\longleftarrow} \mathbb{Z}_q$. B sets $r_{2j} = g^{e_j}(y_1 y_2^{c_j})^{v_j} \bmod p$, $r_{1j} = g^{u_j} r_{2j}^{-d'_j} \bmod p$, $r_j = r_{1j} r_{2j}^{d_j} \bmod p$, $y_j = y_1 y_2^{c_j} \bmod p$, and $s_j = -r_j \alpha v_j^{-1} y_j^{-1} (d_j - d'_j)^{-1} \bmod q$. If $h_1(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, $h_2(m_j, r_{1j}, r_{2j}, y_1, y_2, w$ or $H(c_j, r_j, y_j)$ have been determined then B outputs $\perp$ and stops, else B sets $H(c_j, r_j, y_j) = (u_j + e_j(d_j - d'_j)) y_j s_j \bmod q$, $h_1(m_j, r_{1j}, r_{2j}, y_1, y_2, w) = c_j$, and $h_2(m_j, r_{1j}, r_{2j}, y_1, y_2, w) = d_j$. Finally, B answers (r_{1j}, r_{2j}, s_j). [3]

If $j = \beta$, B answers $(r_{1\beta}, r_{2\beta}, s_\beta)$. If $h_1(m_\beta, r_{1\beta}, r_{2\beta}, y_1, y_2, w)$ has been determined then B

[3] The correctness of this $i \neq \beta$-th signing oracle is the same as Theorem 7.

outputs $\perp$ and stops, otherwise, B sets $h_1(m_\beta, r_{1\beta}, r_{2\beta}, y_1, y_2, w) = c_\beta$. [4]

Simulation of KSA1 with respect to $(m_j, r_{1j}, r_{2j}, s_j)$ in the list L_s. If $j \neq \beta$, B simulates as the same way to the proof of Theorem 7. In the first step of KSA1, B receives $(\overline{y_1}, \overline{y_2})$ and selects $c'_j \xleftarrow{R} \mathbb{Z}_q^*$ and $\overline{w} \xleftarrow{R} \{0,1\}^\lambda$. If $h_1(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ or $h_2(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ has been determined then B outputs $\perp$ and stops, else B sets $h_1(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w}) = c'_j$ and $h_2(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w}) = d'_j$. B then sends $\overline{w}$ to B.

In the second step of KSA1, B receives z, computes $\overline{y_j} = \overline{y_1 y_2}^{c'_j} \bmod p$, and checks $\overline{y_j} \neq g^z \bmod p$. B then computes $\overline{r_j} = r_{1j} r_{2j}^{d'_j} \bmod p$ and $\overline{\alpha} \leftarrow (H(c'_j, \overline{r_j}, \overline{y_j}) - u_j \overline{y_j} s_j)(\overline{r_j} z)^{-1} \bmod q$ and sends $\overline{\alpha}$ to A. [5]

If $j = \beta$, B proceeds as follows. In the first step of KSA1, B receives (y'_1, y'_2) and selects $c'_\beta \xleftarrow{R} \mathbb{Z}_q^*$ and $w' \xleftarrow{R} \{0,1\}^\lambda$. If $h_1(m_\beta, r_{1\beta}, r_{2\beta}, y'_1, y'_2, w')$ or $h_2(m_\beta, r_{1\beta}, r_{2\beta}, y'_1, y'_2, w')$ has been determined then B outputs $\perp$ and stops, else B sets $h_1(m_\beta, r_{1\beta}, r_{2\beta}, y'_1, y'_2, w') = c'_\beta$ and $h_2(m_\beta, r_{1\beta}, r_{2\beta}, y'_1, y'_2, w') = d'_\beta$. B then sends w' to B.

In the second step of KSA1, B receives z, computes $y'_\beta = y'_1 y_2'^{c'_\beta} \bmod p$, and checks $y'_\beta \neq g^z \bmod p$. B then computes $r'_\beta = r_{1\beta} r_{2\beta}^{d'_\beta} \bmod p$ and $\alpha' \leftarrow (H(c'_\beta, r'_\beta, y'_\beta) - (t + b(d'_\beta - d^*)) y'_\beta s_\beta)(r'_\beta z)^{-1} \bmod q$ and sends α' to A. [6]

Output. B obtains A's output $(m, r_1, r_2, s, \overline{y_1}, \overline{y_2}, \overline{\alpha}, \overline{w})$. If $(m, r_1, r_2, s) \neq (m_\beta, r_{1\beta}, r_{2\beta}, s_\beta)$, B outputs $\perp$ and stops. If $h_2(m, r_1, r_2, \overline{y_1}, \overline{y_2}, \overline{w})$ has never been determined then B sets $h_2(m, r_1, r_2, \overline{y_1}, \overline{y_2}, \overline{w}) = d^*$. Else if $h_2(m, r_1, r_2, \overline{y_1}, \overline{y_2}, \overline{w}) \neq d^*$ then B outputs $\perp$ and stops.

B sets $\overline{y} = \overline{y_1 y_2}^c \bmod p$, $\overline{r} = r_1 r_2^{d^*} \bmod p$, and $\overline{\alpha'} = \overline{r\alpha} s_\beta^{-1} \bmod q$ where $c = h_1(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$. B then outputs $(c, \overline{y}, \overline{\alpha'})$.

Analysis of success probability. We first show that if A's output $(m, r_1, r_2, s, \overline{y_1}, \overline{y_2}, \overline{\alpha}, \overline{w})$ satisfies (1) the winning conditions, (2) $(m, r_1, r_2, s) = (m_\beta, r_{1\beta}, r_{2\beta}, s_\beta)$, and (3) $d^* = h_2(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$ then B's output the winning conditions. We assume the follows.

- $g^{H(c, \overline{r}, \overline{y})} = \overline{y}^{\overline{r\alpha}} r^{*\overline{y} s_\beta}$ where $c = h_1(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$, $d^* = h_2(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$, $\overline{y} = \overline{y_1 y_2}^c \bmod p$, and $\overline{r} = r_{1\beta} r_{2\beta}^{d^*} \bmod p$.
- $(m, r_1, r_2, s) \in L_s$.

[4] The correctness of this β-th signing oracle:

$$\begin{aligned}
y_\beta^{r_\beta \alpha} r_\beta^{y_\beta s_\beta} &= (r^{*a^{-1}(d^* - d'_\beta)^{-1}})^{r_\beta \alpha} (g^t r^* r_{2\beta}^{d_\beta - d^*})^{y_\beta s_\beta} \\
&= (r^{*a^{-1}(d^* - d'_\beta)^{-1}})^{r_\beta \alpha} (g^t r^* (r^{*(d^* - d'_\beta)^{-1}} g^b)^{d_\beta - d^*})^{y_\beta s_\beta} \\
&= r^{*a^{-1}(d^* - d'_\beta)^{-1} r_\beta \alpha + (1 + (d^* - d'_\beta)^{-1}(d_\beta - d^*)) y_\beta s_\beta} g^{(t + b(d_\beta - d^*)) y_\beta s_\beta} \\
&= g^{(t + b(d_\beta - d^*)) y_\beta s_\beta} \\
&= g^{H(c_\beta, r_\beta, y_\beta)},
\end{aligned}$$

where $c_\beta = h(m_\beta, r_{1\beta}, r_{2\beta}, y_1, y_2, w)$, $d_\beta = h_2(m_\beta, r_{1\beta}, r_{2\beta}, y_1, y_2, w)$ $r_\beta = r_{1\beta} r_{2\beta}^{d_\beta} \bmod p$, and $y_\beta = y_1 y_2^{c_\beta} \bmod p$.

[5] The correctness of this simulation is the same as Theorem 7.

[6] The correctness of this simulation:

$$\begin{aligned}
y_\beta'^{r'_\beta \alpha'} r_\beta'^{y'_\beta s_\beta} &= (g^z)^{r'_\beta (H(c'_\beta, r'_\beta, y'_\beta) - (t + b(d'_\beta - d^*)) y'_\beta s_\beta)(r'_\beta z)^{-1}} (g^{t + b(d'_\beta - d^*)})^{y'_\beta s_\beta} \\
&= g^{H(c'_\beta, r'_\beta, y'_\beta)},
\end{aligned}$$

where $c'_\beta = h(m_\beta, r_{1\beta}, r_{2\beta}, y'_1, y'_2, w')$, $d'_\beta = h_2(m_\beta, r_{1\beta}, r_{2\beta}, y'_1, y'_2, w')$ $r'_\beta = r_{1\beta} r_{2\beta}^{d'_\beta} (= g^{t + b(d'_\beta - d^*)}) \bmod p$, and $y'_\beta = y'_1 y_2'^{c'_\beta} \bmod p$.

- $(m, r_1, r_2, s) \notin L_k$.
- $(\overline{y_1}, \overline{y_2}, \overline{\alpha}, \overline{w}) \notin \{(y'_1, y'_2, \alpha', w')\}$. [7]
- $(y_1, y_2, \alpha, w) \neq (\overline{y_1}, \overline{y_2}, \overline{\alpha}, \overline{w})$ In our case, if $(y_1, y_2, w) = (\overline{y_1}, \overline{y_2}, \overline{w})$ then $\alpha = \overline{\alpha}$. Therefore, we obtain $(y_1, y_2, w) \neq (\overline{y_1}, \overline{y_2}, \overline{w})$.

Then, we can see the following equations.

$$\begin{array}{rcl} r^{*\overline{y}}\overline{y}^{\overline{\alpha'}} & = & (\overline{r}g^{-t})^{\overline{y}}\overline{y}^{\overline{\alpha r}s_\beta^{-1}} \\ & = & (g^{H(c,\overline{r},\overline{y})}\overline{y}^{-\overline{r\alpha}})^{s_\beta^{-1}}\overline{y}^{\overline{\alpha r}s_\beta^{-1}}g^{-t\overline{y}} \\ & = & g^{H'(c,\overline{y})} \end{array}$$

where $c = h_1(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$, $d^* = h_2(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$, $\overline{y} = \overline{y_1}\overline{y_2}^c \bmod p$, and $\overline{r} = r_{1\beta}r_{2\beta}^{d^*}(= r^*g^t) \bmod p$.

We next estimate the probability that B failures the simulation of the initialization, the signing oracle, KSA1, and the random oracles.

In the simulation of the signing oracle, only if $h_1(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, $h_2(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, or $H(c_j, r_j, y_j)$ have been determined then B fails. The probability that this happens is at most $\frac{2(q_{h_1}+q_k+q_s)}{q^2} + \frac{2(q_{h_2}+q_k+q_s)}{q^2} + \frac{q_H+q_s}{q} = \frac{2(q_{h_1}+q_{h_2}+2q_k+2q_s)}{q^2} + \frac{q_H+q_s}{q}$.

In the simulation of KSA1, only if $h_1(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ or $h_2(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ has been determined then B fails. The probability that this happens is at most $\frac{q_{h_1}+q_k+q_s}{2^\lambda} + \frac{q_{h_2}+q_k+q_s)}{2^\lambda} = \frac{q_{h_1}+q_{h_2}+2q_k+2q_s}{2^\lambda}$.

In the mentioned above, we can see that B failures the simulation of A's environment with at most $(\frac{2(q_{h_1}+q_{h_2}+2q_k+2q_s)}{q^2} + \frac{q_H+q_s}{q}) \cdot q_s + \frac{q_{h_1}+q_{h_2}+2q_k+2q_s}{2^\lambda} \cdot q_k$. We denote the probability as δ. And the successful simulation is properly distributed. Therefore, we can see that A outputs $(m, r_1, r_2, s, \overline{y_1}, \overline{y_2}, \overline{\alpha}, \overline{w})$ such that $(m, r_1, r_2, s) = (m_\beta, r_{1\beta}, r_{2\beta}, s_\beta)$ and $d^* = h_2(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$ with the advantage $\frac{\epsilon'-\delta}{q_s q_{h_2}}$. As the mentioned above, $\epsilon \geq \frac{\epsilon'-\delta}{q_s q_{h_2}}$.

Analysis of running time. B takes the time $O(\lambda^3)$ to the simulation of initialization, $O(\lambda^2)q_H$ to the random oracle oracle H, $O(\lambda)$ to each random oracle oracle h_1, h_2, and h_3, $O(\lambda^3)$ to the signing oracle, $O(\lambda^3)$ to the key substitution algorithm KSA1, $O(\lambda^3)$ to produce the output, and t' to the running time of A. The total is $(q_{h_1} + q_{h_2} + q_{h_3})O(\lambda) + q_H O(\lambda^2) + (q_s + q_k + 2)O(|q||p|^2) + t'$. □

A.3 proof of Theorem 9

Proof. Given an algorithm A that breaks the $(t', \epsilon', q_s, q_k, q_H, q_{h_1}, q_{h_2}, q_{h_3})$-unforgeability 2 of $\mathcal{KS}_1$, we construct an algorithm B that breaks the (t, ϵ, q_H)-EU-KOA of the ElGamal signature scheme.

B is given a public key (p, g, y_2), where $g \in \mathbb{Z}_p^*$ is an element whose order is q and $y_2 \in \langle g \rangle$, and can access to the random oracle H'.

Initialization of A. B provides $\mathtt{A}_1$ the common parameter $\mathtt{cp} = (p, g)$. For this simulation, B selects $z \xleftarrow{R} \mathbb{Z}_q$ and $c^* \xleftarrow{R} \mathbb{Z}_q^*$ and computes $y_1 \leftarrow g^z y_2^{-c^*} \bmod p$. B then sets $L_c = \{c^*\}$ and chooses $\beta \xleftarrow{R} \{1, \cdots, q_{h_1}\}$.

[7] This means that $h_2(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$ has never setted as $\{d_1, \cdots, d_{q_s}, d'_1, \cdots, d'_{q_s}\}$. That is, it setted as d^* with probability $\frac{1}{q_{h_2}}$.

Simulation of random oracle H. This random oracle takes as input a query (c, r, y) and proceeds as follows. If $H(c, r, y)$ has been determined, B answers the value. Otherwise, if $y = y_1 y_2^c \bmod p$ and $c \not\equiv c^* \pmod q$ then B answers $(H'(c, r)(c^* - c) + zr)\alpha \bmod q$, else B answers a random value in $\mathbb{Z}_q$.

Simulation of random oracle h_1. In the i-th query from A, B receives a query $(m, r_1, r_2, y_1, y_2, w)$ and proceeds as follows.

If $h_1(m, r_1, r_2, y_1, y_2, w)$ has been determined, B answers the value. Otherwise, if $i = \beta$ then B answers c^*, else B answers a random value $c \in \mathbb{Z}_q^*$. However, if $c \in L_c$ then B outputs $\perp$ and stops, else B adds $L_c \leftarrow L_c \cup \{c\}$.

Simulation of random oracle h_2. This random oracle takes as input a query $(m, r_1, r_2, y_1, y_2, w)$ and proceeds as follows. If $h_2(m, r_1, r_2, y_1, y_2, w)$ has been determined, B answers the value. Otherwise, B answers a random value $d \in \mathbb{Z}_q^*$.

Simulation of random oracle h_3. This random oracle takes as input a query (r_1, x_1, x_2) and proceeds as follows. If $h_3(r_1, x_1, x_2)$ has been determined, B answers the value. Otherwise, B answers a random value in $\mathbb{Z}_q^*$.

The simulation of KSA2 with respect to (m^*, r_1^*, r_2^*, s^*). In the first step, B sends (y_1, y_2).

In the second step, B receives w and proceeds as follows. If $h_1(m^*, r_1^*, r_2^*, y_1, y_2, w)$ has never been determined then B sets $h_1(m^*, r_1^*, r_2^*, y_1, y_2, w) = c^*$ and resets β as 0. Else if $h_1(m^*, r_1^*, r_2^*, y_1, y_2, w) \neq c^*$ then B outputs $\perp$ and stops. B then sends z. [8]

In the third step, B receives α and provides A$_2$ the public key $pk = (y_1, y_2, \alpha, w)$. Note that α should satisfy $g^{H(c^*, r^*, y)} \equiv y^{r^* \alpha} r^{*ys^*} \pmod p$, where $y = y_1 y_2^{c^*} \bmod p$ $(= g^z \bmod p)$, $r^* = r_1^* r_2^{*d^*} \bmod p$, and $d^* = h_2(m^*, r_1^*, r_2^*, y_1, y_2, w)$.

Simulation of signing oracle. In the j-th signing query from A, B proceeds as follows. In particular, we denote the j-th signing query m as m_j.

B chooses $c_j, v_j, d_j, d'_j (d_j \neq d'_j) \xleftarrow{R} \mathbb{Z}_q^*$ and $u_j, e_j \xleftarrow{R} \mathbb{Z}_q$. B sets $r_{2j} = g^{e_j}(y_1 y_2^{c_j})^{v_j} \bmod p$, $r_{1j} = g^{u_j} r_{2j}^{-d'_j} \bmod p$, $r_j = r_{1j} r_{2j}^{d_j} \bmod p$, $y_j = y_1 y_2^{c_j} \bmod p$, and $s_j = -r_j \alpha v_j^{-1} y_j^{-1} (d_j - d'_j)^{-1} \bmod q$. If $h_1(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, $h_2(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, or $H(c_j, r_j, y_j)$ have been determined then B outputs $\perp$ and stops, else B sets $H(c_j, r_j, y_j) = (u_j + e_j(d_j - d'_j)) y_j s_j \bmod q$, $h_1(m_j, r_{1j}, r_{2j}, y_1, y_2, w) = c_j$, and $h_2(m_j, r_{1j}, r_{2j}, y_1, y_2, w) = d_j$. If $c_j \in L_c$, B outputs $\perp$ and stops. Otherwise, B sets $L_c \leftarrow L_c \cup \{c_j\}$ and answers (r_{1j}, r_{2j}, s_j). [9]

Simulation of KSA1 with respect to $(m_j, r_{1j}, r_{2j}, s_j)$ in the list L_s. In the first step of KSA1, B receives $(\overline{y_1}, \overline{y_2})$ and selects $c'_j \xleftarrow{R} \mathbb{Z}_q^*$ and $\overline{w} \xleftarrow{R} \{0, 1\}^\lambda$. If $h_1(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ or $h_2(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ has been determined then B outputs $\perp$ and stops, else B sets $h_1(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w}) = c'_j$ and $h_2(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w}) = d'_j$. If $c'_j \in L_c$, B outputs $\perp$ and stops. Otherwise, B sets $L_c \leftarrow L_c \cup \{c'_j\}$ and sends $\overline{w}$ to A$_3$.

In the second step of KSA1, B receives z, computes $\overline{y_j} = \overline{y_1 y_2}^{c'_j} \bmod p$, and checks $\overline{y_j} \neq g^z \bmod p$. B then computes $\overline{r_j} = r_{1j} r_{2j}^{d'_j} \bmod p$ and $\overline{\alpha} \leftarrow (H(c'_j, \overline{r_j}, \overline{y_j}) - u_j \overline{y_j} s_j)(\overline{r_j} z)^{-1} \bmod q$ and sends $\overline{\alpha}$ to A$_3$. [10]

[8] The correctness of this simulation: $g^z \equiv y_1 y_2^{c^*}$ where $c^* = h_1(m^*, r_1^*, r_2^*, y_1, y_2, w)$.

[9] The correctness of this simulation is the same as Theorem 7.

[10] The correctness of this simulation is the same as Theorem 7.

Output. B obtains $\mathtt{A}_3$'s output (m, r_1, r_2, s). B sets $r = r_1 r_2^d \bmod p$ and $s' = (y_1 y_2^c \bmod p)(c^* - c)^{-1}\alpha^{-1}s \bmod q$ where $c = h_1(m, r_1, r_2, y_1, y_2, w)$ and $d = h_2(m, r_1, r_2, y_1, y_2, w)$. B then outputs (c, r, s').

Analysis of success probability. We first show that if A's output (m, r_1, r_2, s) satisfies the winning conditions then B's output the winning conditions. We assume the follows.

- $g^{H(c,r,y)} = y^{r\alpha} r^{ys}$ where $c = h_1(m, r_1, r_2, y_1, y_2, w)$, $d = h_2(m, r_1, r_2, y_1, y_2, w)$, $y = y_1 y_2^c \bmod p$, and $r = r_1 r_2^d \bmod p$.
- m is not queried to the signing oracle $\mathcal{S}$.

Note that $c \neq c^*$. To explain this, we assume that $c = c^*$. Since c is the answer of the random oracle h_1 for the query $(m, r_1, r_2, y_1, y_2, w)$, c^* is the answer for $(m^*, r_1^*, r_2^*, y_1, y_2, w)$, and the answers of random oracle h_1 for distinct queries are always distinct, we obtain $(m, r_1, r_2) = (m^*, r_1^*, r_2^*)$. Therefore, we can also see $s = s^*$, that is, $(m, r_1, r_2, s) = (m^*, r_1^*, r_2^*, s^*)$. This is contradict to $(m, r_1, r_2, s) \neq (m^*, r_1^*, r_2^*, s^*)$.

Then, we can see the following equations.

$$\begin{aligned} y_2^r r^{s'} &= y_2^r (r^{(y_1 y_2^c \bmod p)s})^{(c^*-c)^{-1}\alpha^{-1}} \\ &= y_2^r (g^{H(c,r,y_1 y_2^c \bmod p)} (y_1 y_2^c)^{-r\alpha})^{(c^*-c)^{-1}\alpha^{-1}} \\ &= y_2^r (g^{H(c,r,y_1 y_2^c \bmod p)} ((g^z y_2^{-c^*}) y_2^c)^{-r\alpha})^{(c^*-c)^{-1}\alpha^{-1}} \\ &= g^{(H(c,r,y_1 y_2^c \bmod p)\alpha - zr)(c^*-c)^{-1}} \\ &= g^{H'(c,r)}, \end{aligned}$$

where $c = h_1(m, r_1, r_2, y_1, y_2, w)$.

We next estimate the probability that B failures the simulation of the initialization, KSA2, the signing oracle, KSA1, and the random oracles.

In the simulation of KSA2, $\mathtt{A}_1$ outputs (m^*, r_1^*, r_2^*, s^*) such that $h_1(m^*, r_1^*, r_2^*, s^*, y_1, y_2, w)$ has never been determined or $h_1(m^*, r_1^*, r_2^*, s^*, y_1, y_2, w)$ has been determined as c^* with advantage $\frac{\epsilon'}{q_{h_1}}$.

In the simulation of the random oracle h_1, only if $c \in L_c$ happens, B fails. The probability that this happens is at most $\frac{q_{h_1}+q_s+q_k}{q}$.

In the simulation of the signing oracle, only if $h_1(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, $h_2(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, $H(c_j, r_j, y_j)$, or $c_j \in L_c$ have been determined then B fails. The probability that this happens is at most $\frac{2(q_{h_1}+q_k+q_s)}{q^2} + \frac{2(q_{h_2}+q_k+q_s)}{q^2} + \frac{q_H+q_s}{q} + \frac{q_{h_1}+q_s+q_k}{q} = \frac{2(q_{h_1}+q_{h_2}+2q_k+2q_s)}{q^2} + \frac{q_H+2q_s+q_{h_1}+q_k}{q}$.

In the simulation of KSA1, only if $h_1(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$, $h_2(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$, or $c_j' \in L_c$ has been determined then B fails. The probability that this happens is at most $\frac{q_{h_1}+q_k+q_s}{2^\lambda} + \frac{q_{h_2}+q_k+q_s)}{2^\lambda} + \frac{q_{h_1}+q_s+q_k}{q} = \frac{2q_{h_1}+q_{h_2}+3q_k+3q_s}{2^\lambda}$.

In the mentioned above, we can see that B failures the simulation of A's environment with at most $(\frac{2(q_{h_1}+q_{h_2}+2q_k+2q_s)}{q^2} + \frac{q_H+2q_s+q_{h_1}+q_k}{q}) \cdot q_s + \frac{2q_{h_1}+q_{h_2}+3q_k+3q_s}{2^\lambda} \cdot q_k + \frac{q_{h_1}+q_s+q_k}{q} \cdot q_{h_1}$. We denote the probability as δ. And the successful simulation is properly distributed. Therefore, we can see $\epsilon \geq \frac{\epsilon'}{q_{h_1}} - \delta$.

Analysis of running time. B takes the time $O(\lambda^3)$ to the simulation of initialization, $O(\lambda^3)$ to the random oracle oracle H, $O(\lambda)$ to each random oracle oracle h_1, h_2, and h_3, $O(\lambda^3)$ to the key substitution algorithm KSA2, $O(\lambda^3)$ to the signing oracle, $O(\lambda^3)$ to the key substitution algorithm KSA1, $O(\lambda^3)$ to produce the output, and t' to the running time of A. The total is $(q_{h_1} + q_{h_2} + q_{h_3})O(\lambda) + (q_H + q_s + q_k + 2)O(\lambda^3) + t'$.

□

A.4 proof of Theorem 10

Proof. Given an algorithm A that breaks the $(t', \epsilon', q_s, q_k, q_H, q_{h_1}, q_{h_2}, q_{h_3})$-non-substitutability 2 of $\mathcal{S}$, we construct an algorithm B that breaks the (t, ϵ, q_H)-EU-KOA of the ElGamal signature scheme.

B is given a public key (p, g, r_0), where $g \in \mathbb{Z}_p^*$ is an element whose order is q and $r_0 \in \langle g \rangle$, and can access to the random oracle H'.

Initialization of A. If $r_0 = 1$ then B finds the corresponding secret key $k_0 (= 0)$, therefore, B stops. Otherwise, B proceeds as follows. B chooses $\beta \overset{R}{\longleftarrow} \{1, \cdots, q_s\}$, $\gamma \overset{R}{\longleftarrow} \{1, \cdots, q_{h_1}\}$, $\zeta \overset{R}{\longleftarrow} \{1, \cdots, q_{h_2}\}$, $a, d^*, d'_\beta, d_\beta, c^*, c_\beta (c^* \neq c_\beta) \overset{R}{\longleftarrow} \mathbb{Z}_q^*$, and $b, t, x_2 \overset{R}{\longleftarrow} \mathbb{Z}_q$, computes $y_2 = g^{x_2} r_0^{a^{-1}(d^* - d'_\beta)^{-1}(c_\beta - c^*)^{-1}} \bmod p$, $y_1 = r_0^{a^{-1}(d^* - d'_\beta)^{-1}} y_2^{-c_\beta} \bmod p$, $r_{2\beta} = r_0^{(d^* - d'_\beta)^{-1}} g^b \bmod p$, $r_{1\beta} = r_0 r_{2\beta}^{-d^*} g^t$, $y_\beta = y_1 y_2^{c_\beta} \bmod p$, $r_\beta = r_1 r_2^{d_\beta} \bmod p$, and sets $s_\beta = -a^{-1}(d^* - d'_\beta)^{-1} r_\beta \alpha (1 + (d^* - d'_\beta)^{-1}(d_\beta - d^*))^{-1} y_\beta^{-1} \bmod q$, and $H(c_\beta, r_\beta, y_\beta) = (t + b(d_\beta - d^*)) y_\beta s_\beta$. Finally, B provides $\mathtt{A}_1$ the common parameter $\mathtt{cp} = (p, g)$.

Simulation of random oracle H. This random oracle takes as input a query (c, r, y) and proceeds as follows. If $H(c, r, y)$ has been determined, B answers the value. Otherwise, if $(c, r, y) = (c, r_0 g^t, y)$ then B answers $H'(c, y) s_\beta - ty \bmod q$, else answers a random value in $\mathbb{Z}_q$.

Simulation of random oracle h_1. In the i-th query from A, B receives a query $(m, r_1, r_2, y_1, y_2, w)$ and proceeds as follows.

If $h_1(m, r_1, r_2, y_1, y_2, w)$ has been determined, B answers the value. Otherwise, if $i = \gamma$ then B answers c^*, else B answers a random value $c \in \mathbb{Z}_q^*$.

Simulation of random oracle h_2. In the l-th query from A, B receives a query $(m, r_1, r_2, y_1, y_2, w)$ and proceeds as follows.

If $h_2(m, r_1, r_2, y_1, y_2, w)$ has been determined, B answers the value. Otherwise, if $l = \zeta$ then B answers d^*, else B answers a random value $d \in \mathbb{Z}_q^*$.

Simulation of random oracle h_3. This random oracle takes as input a query (r_1, x_1, x_2) and proceeds as follows. If $h_3(r_1, x_1, x_2)$ has been determined, B answers the value. Otherwise, B answers a random value in $\mathbb{Z}_q^*$.

The simulation of KSA2 with respect to (m^*, r_1^*, r_2^*, s^*). In the first step, B sends (y_1, y_2) to $\mathtt{A}_2$.

In the second step, B receives w and proceeds as follows. If $h_1(m^*, r_1^*, r_2^*, y_1, y_2, w)$ has never been determined then B sets $h_1(m^*, r_1^*, r_2^*, y_1, y_2, w) = c^*$ and resets γ as 0. Else if $h_1(m^*, r_1^*, r_2^*, y_1, y_2, w) \neq c^*$ then B outputs $\perp$ and stops. B then sends $x_2(c^* - c_\beta)$. [11]

In the third step, B receives α and provides $\mathtt{A}_2$ the public key $pk = (y_1, y_2, \alpha, w)$. Note that α should satisfy $g^{H(c^*, r^*, y)} \equiv y^{r^* \alpha} r^{* y s^*} \pmod p$, where $y = y_1 y_2^{c^*} \bmod p (= g^{x_2(c^* - c_\beta)} \bmod p)$, $r^* = r_1^* r_2^{*d^*} \bmod p$, and $d^* = h_2(m^*, r_1^*, r_2^*, y_1, y_2, w)$.

[11] The correctness of this simulation:

$$\begin{aligned} y_1 y_2^{c^*} &\equiv r_0^{a^{-1}(d^* - d'_\beta)^{-1}} y_2^{c^* - c_\beta} \\ &\equiv (y_2 g^{-x_2})^{c_\beta - c^*} y_2^{c^* - c_\beta} \\ &\equiv g^{x_2(c^* - c_\beta)} \end{aligned}$$

where $c^* = h_1(m^*, r_1^*, r_2^*, y_1, y_2, w)$.

Simulation of signing oracle. In the j-th signing query from $\mathtt{A}_3$, B proceeds as follows. In particular, we denote the j-th signing query m as m_j.

If $j \neq \beta$, B simulates as the same way to the proof of Theorem 7. B chooses $c_j, v_j, d_j, d'_j (d_j \neq d'_j) \xleftarrow{R} \mathbb{Z}_q^*$ and $u_j, e_j \xleftarrow{R} \mathbb{Z}_q$. B sets $r_{2j} = g^{e_j}(y_1 y_2^{c_j})^{v_j} \bmod p$, $r_{1j} = g^{u_j} r_{2j}^{-d'_j} \bmod p$, $r_j = r_{1j} r_{2j}^{d_j} \bmod p$, $y_j = y_1 y_2^{c_j} \bmod p$, and $s_j = -r_j \alpha v_j^{-1} y_j^{-1} (d_j - d'_j)^{-1} \bmod q$. If $h_1(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, $h_2(m_j, r_{1j}, r_{2j}, y_1, y_2, w$ or $H(c_j, r_j, y_j)$ have been determined then B outputs $\perp$ and stops, else B sets $H(c_j, r_j, y_j) = (u_j + e_j(d_j - d'_j)) y_j s_j \bmod q$, $h_1(m_j, r_{1j}, r_{2j}, y_1, y_2, w) = c_j$, and $h_2(m_j, r_{1j}, r_{2j}, y_1, y_2, w) = d_j$. Finally, B answers (r_{1j}, r_{2j}, s_j). [12]

If $j = \beta$, B answers $(r_{1\beta}, r_{2\beta}, s_\beta)$. If $h(m_j, r_{1\beta}, r_{2\beta}, y_1, y_2, w)$ has been determined then B outputs $\perp$ and stops, otherwise, B sets $h_1(m_\beta, r_{1\beta}, r_{2\beta}, y_1, y_2, w) = c_\beta$ and $h_2(m_\beta, r_{1\beta}, r_{2\beta}, y_1, y_2, w) = d_\beta$. [13]

Simulation of KSA1 with respect to $(m_j, r_{1j}, r_{2j}, s_j)$ in the list L_s. If $j \neq \beta$, B simulates as the same way to the proof of Theorem 7. In the first step of KSA1, B receives $(\overline{y_1}, \overline{y_2})$ and selects $c'_j \xleftarrow{R} \mathbb{Z}_q^*$ and $\overline{w} \xleftarrow{R} \{0,1\}^\lambda$. If $h_1(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ or $h_2(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ has been determined then B outputs $\perp$ and stops, else B sets $h_1(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w}) = c'_j$ and $h_2(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w}) = d'_j$. B then sends $\overline{w}$ to B.

In the second step of KSA1, B receives z, computes $\overline{y_j} = \overline{y_1 y_2}^{c'_j} \bmod p$, and checks $\overline{y_j} \neq g^z \bmod p$. B then computes $\overline{r_j} = r_{1j} r_{2j}^{d'_j} \bmod p$ and $\overline{\alpha} \leftarrow (H(c'_j, \overline{r_j}, \overline{y_j}) - u_j \overline{y_j} s_j)(\overline{r_j} z)^{-1} \bmod q$ and sends $\overline{\alpha}$ to $\mathtt{A}$. [14]

If $j = \beta$, B proceeds as follows. In the first step of KSA1, B receives (y'_1, y'_2) and selects $c'_\beta \xleftarrow{R} \mathbb{Z}_q^*$ and $w' \xleftarrow{R} \{0,1\}^\lambda$. If $h_1(m_\beta, r_{1\beta}, r_{2\beta}, y'_1, y'_2, w')$ or $h_2(m_\beta, r_{1\beta}, r_{2\beta}, y'_1, y'_2, w')$ has been determined then B outputs $\perp$ and stops, else B sets $h_1(m_\beta, r_{1\beta}, r_{2\beta}, y'_1, y'_2, w') = c'_\beta$ and $h_2(m_\beta, r_{1\beta}, r_{2\beta}, y'_1, y'_2, w') = d'_\beta$. B then sends w' to B.

In the second step of KSA1, B receives z, computes $y'_\beta = y'_1 y_2^{\prime c'_\beta} \bmod p$, and checks $y'_\beta \neq g^z \bmod p$. B then computes $r'_\beta = r_{1\beta} r_{2\beta}^{d'_\beta} \bmod p$ and $\alpha' \leftarrow (H(c'_\beta, r'_\beta, y'_\beta) - (t + b(d'_\beta - d^*)) y'_\beta s_\beta)(r'_\beta z)^{-1} \bmod q$ and sends α' to $\mathtt{A}$. [15]

Output. B obtains $\mathtt{A}_3$'s output $(m, r_1, r_2, s, \overline{y_1}, \overline{y_2}, \overline{\alpha}, \overline{w})$. If $(m, r_1, r_2, s) \neq (m_\beta, r_{1\beta}, r_{2\beta}, s_\beta)$, B outputs $\perp$ and stops. If $h_2(m, r_1, r_2, \overline{y_1}, \overline{y_2}, \overline{w})$ has never been determined then B sets $h_2(m, r_1, r_2, \overline{y_1}, \overline{y_2}, \overline{w}) =$

[12] The correctness of this $i \neq \beta$-th signing oracle is the same as Theorem 7.

[13] The correctness of this β-th signing oracle:

$$\begin{aligned}
y_\beta^{r_\beta \alpha} r_\beta^{y_\beta s_\beta} &= (r^{*a^{-1}(d^* - d'_\beta)^{-1}})^{r_\beta \alpha} (g^t r^* r_{2\beta}^{d_\beta - d^*})^{y_\beta s_\beta} \\
&= (r^{*a^{-1}(d^* - d'_\beta)^{-1}})^{r_\beta \alpha} (g^t r^* (r^{*(d^* - d'_\beta)^{-1}} g^b)^{d_\beta - d^*})^{y_\beta s_\beta} \\
&= r^{*a^{-1}(d^* - d'_\beta)^{-1} r_\beta \alpha + (1 + (d^* - d'_\beta)^{-1}(d_\beta - d^*)) y_\beta s_\beta} g^{(t + b(d_\beta - d^*)) y_\beta s_\beta} \\
&= g^{(t + b(d_\beta - d^*)) y_\beta s_\beta} \\
&= g^{H(c_\beta, r_\beta, y_\beta)},
\end{aligned}$$

where $c_\beta = h(m_\beta, r_{1\beta}, r_{2\beta}, y_1, y_2, w)$, $d_\beta = h_2(m_\beta, r_{1\beta}, r_{2\beta}, y_1, y_2, w)$ $r_\beta = r_{1\beta} r_{2\beta}^{d_\beta} \bmod p$, and $y_\beta = y_1 y_2^{c_\beta} \bmod p$.

[14] The correctness of this simulation is the same as Theorem 7.

[15] The correctness of this simulation:

$$\begin{aligned}
y_\beta^{\prime r'_\beta \alpha'} r_\beta^{\prime y'_\beta s_\beta} &= (g^z)^{r'_\beta (H(c'_\beta, r'_\beta, y'_\beta) - (t + b(d'_\beta - d^*)) y'_\beta s_\beta)(r'_\beta z)^{-1}} (g^{t + b(d'_\beta - d^*)})^{y'_\beta s_\beta} \\
&= g^{H(c'_\beta, r'_\beta, y'_\beta)},
\end{aligned}$$

where $c'_\beta = h(m_\beta, r_{1\beta}, r_{2\beta}, y'_1, y'_2, w')$, $d'_\beta = h_2(m_\beta, r_{1\beta}, r_{2\beta}, y'_1, y'_2, w')$ $r'_\beta = r_{1\beta} r_{2\beta}^{d'_\beta} (= g^{t + b(d'_\beta - d^*)}) \bmod p$, and $y'_\beta = y'_1 y_2^{\prime c'_\beta} \bmod p$.

d^*. Else if $h_2(m, r_1, r_2, \overline{y_1}, \overline{y_2}, \overline{w}) \neq d^*$ then B outputs $\perp$ and stops.

If $h_2(m, r_1, r_2, \overline{y_1}, \overline{y_2}, \overline{w})$ has been determined as $h_2(m, r_1, r_2, \overline{y_1}, \overline{y_2}, \overline{w}) \neq d^*$, B also outputs $\perp$ and stops.

B sets $\overline{y} = \overline{y_1 y_2}^c \bmod p$, $\overline{r} = r_1 r_2^{d^*} \bmod p$, and $\overline{\alpha'} = \overline{r\alpha} s_\beta^{-1} \bmod q$ where $c = h_1(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$. B then outputs $(c, \overline{y}, \overline{\alpha'})$.

Analysis of success probability. We first show that if A$_3$'s output $(m, r_1, r_2, s, \overline{y_1}, \overline{y_2}, \overline{\alpha}, \overline{w})$ satisfies (1) the winning conditions, (2) $(m, r_1, r_2, s) = (m_\beta, r_{1\beta}, r_{2\beta}, s_\beta)$, and (3) $d^* = h_2(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$ then B's output the winning conditions. We assume the follows.

- $g^{H(c,\overline{r},\overline{y})} = \overline{y}^{\overline{r}\alpha} r_0^{\overline{y} s_\beta}$ where $c = h_1(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$, $d^* = h_2(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$, $\overline{y} = \overline{y_1 y_2}^c \bmod p$, and $\overline{r} = r_{1\beta} r_{2\beta}^{d^*} \bmod p$.
- $(m, r_1, r_2, s) \in L_s$.
- $(m, r_1, r_2, s) \notin L_k$.
- $(\overline{y_1}, \overline{y_2}, \overline{\alpha}, \overline{w}) \notin \{(y'_1, y'_2, \alpha', w')\}$. [16]
- $(y_1, y_2, \alpha, w) \neq (\overline{y_1}, \overline{y_2}, \overline{\alpha}, \overline{w})$ In our case, if $(y_1, y_2, w) = (\overline{y_1}, \overline{y_2}, \overline{w})$ then $\alpha = \overline{\alpha}$. Therefore, we obtain $(y_1, y_2, w) \neq (\overline{y_1}, \overline{y_2}, \overline{w})$.

Then, we can see the following equations.

$$\begin{aligned} r_0^{\overline{y}} \overline{y}^{\overline{\alpha'}} &= (\overline{r} g^{-t})^{\overline{y}} \overline{y}^{\overline{\alpha r} s_\beta^{-1}} \\ &= (g^{H(c,\overline{r},\overline{y})} \overline{y}^{-\overline{r\alpha}})^{s_\beta^{-1}} \overline{y}^{\overline{\alpha r} s_\beta^{-1}} g^{-t\overline{y}} \\ &= g^{H'(c,\overline{y})} \end{aligned}$$

where $c = h_1(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$, $d^* = h_2(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$, $\overline{y} = \overline{y_1 y_2}^c \bmod p$, and $r_0 = r_{1\beta} r_{2\beta}^{d^*} \bmod p$.

We next estimate the probability that B failures the simulation of the initialization, KSA2, the signing oracle, KSA1, and the random oracles.

In the simulation of KSA2, A$_1$ outputs (m^*, r_1^*, r_2^*, s^*) such that $h(m^*, r_1^*, r_2^*, s^*, y_1, y_2, w)$ has never been determined or $h(m^*, r_1^*, r_2^*, s^*, y_1, y_2, w)$ has been determined as c^* with advantage $\frac{\epsilon'}{q_{h_1}}$.

In the simulation of the signing oracle, only if $h_1(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, $h_2(m_j, r_{1j}, r_{2j}, y_1, y_2, w)$, or $H(c_j, r_j, y_j)$ have been determined then B fails. The probability that this happens is at most $\frac{2(q_{h_1}+q_k+q_s)}{q^2} + \frac{2(q_{h_2}+q_k+q_s)}{q^2} + \frac{q_H+q_s}{q} = \frac{2(q_{h_1}+q_{h_2}+2q_k+2q_s)}{q^2} + \frac{q_H+q_s}{q}$.

In the simulation of KSA1, only if $h_1(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ or $h_2(m_j, r_{1j}, r_{2j}, \overline{y_1}, \overline{y_2}, \overline{w})$ has been determined then B fails. The probability that this happens is at most $\frac{q_{h_1}+q_k+q_s}{2^\lambda} + \frac{q_{h_2}+q_k+q_s)}{2^\lambda} = \frac{q_{h_1}+q_{h_2}+2q_k+2q_s}{2^\lambda}$.

In the mentioned above, we can see that B failures the simulation of A's environment with at most $(\frac{2(q_{h_1}+q_{h_2}+2q_k+2q_s)}{q^2} + \frac{q_H+q_s}{q}) \cdot q_s + \frac{q_{h_1}+q_{h_2}+2q_k+2q_s}{2^\lambda} \cdot q_k$. We denote the probability as δ. And the successful simulation is properly distributed. Therefore, we can see that A outputs $(m, r_1, r_2, s, \overline{y_1}, \overline{y_2}, \overline{\alpha}, \overline{w})$ such that $(m, r_1, r_2, s) = (m_\beta, r_{1\beta}, r_{2\beta}, s_\beta)$ and $d^* = h_2(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$ with the advantage $(\frac{\epsilon'}{q_h} - \delta)\frac{1}{q_s q_{h_2}}$. As the mentioned above, $\epsilon \geq (\frac{\epsilon'}{q_h} - \delta)\frac{1}{q_s q_{h_2}}$.

[16] This means that $h_2(m_\beta, r_{1\beta}, r_{2\beta}, \overline{y_1}, \overline{y_2}, \overline{w})$ has never setted as $\{d_1, \cdots, d_{q_s}, d'_1, \cdots, d'_{q_s}\}$. That is, it setted as d^* with probability $\frac{1}{q_{h_2}}$.

Analysis of running time. `B` takes the time $O(\lambda^3)$ to the simulation of initialization, $O(\lambda^3)$ to the random oracle oracle H, $O(\lambda)$ to each random oracle oracle h_1, h_2, and h_3, $O(\lambda^3)$ to the key substitution algorithm `KSA2`, $O(\lambda^3)$ to the signing oracle, $O(\lambda^3)$ to the key substitution algorithm `KSA1`, $O(\lambda^3)$ to produce the output, and t' to the running time of `A`. The total is $(q_{h_1} + q_{h_2} + q_{h_3})O(\lambda) + (q_H + q_s + q_k + 2)O(\lambda^3) + t'$.

□