Generic Transformation to Strongly Unforgeable Signatures[⋆]

Qiong Huang[1], Duncan S. Wong[1], and Yiming Zhao[2]

[1] Dept. of Computer Science,
City University of Hong Kong
Hong Kong, China
{csqhuang,duncan}@cityu.edu.hk

[2] Dept. of Computer Science and Engineering,
Fudan University
Shanghai 200433, China
zhym@fudan.edu.cn

Abstract. Recently, there are two generic transformation techniques proposed for converting unforgeable signature schemes (the message in the forgery has not been signed yet) into strongly unforgeable ones (the message in the forgery could have been signed previously). Both techniques are based on trapdoor hash functions and require to add supplementary components onto the original key pair of the signature scheme. In this paper, we propose a new generic transformation which converts *any* unforgeable signature scheme into a strongly unforgeable one, and also keeps the key pair of the signature scheme unchanged. Our technique is based on *strong one-time signature schemes*. We show that they can be constructed efficiently from any one-time signature schemes that follow the one-way function paradigm. The performance of our technique also compares favorably with that of those trapdoor-hash-function-based ones. In addition, this new generic transformation can also be used for attaining strongly unforgeable signature schemes in other cryptographic settings which include certificateless signature, identity-based signature, and many others. To the best of our knowledge, similar extent of versatility is not known to be supported by any of those comparable techniques. Finally and of independent interest, we show that our generic transformation technique can be modified to an *on-line/off-line* signature scheme, which possesses a very efficient signing process.

1 Introduction

When considering the security of a signature scheme, we usually refer to the existential unforgeability against adaptive chosen message attacks [13]. The security requirement is to prevent forgery of signatures on new messages not previously

⋆ The first two authors are supported by a grant from CityU (Project No. 7001844). The third author is supported by National Natural Science Foundation of China under Grant No. 60573054.

signed. However, most signature schemes are randomized and allow many possible signatures for a single message. In some applications, a stronger security notion, called *strong unforgeability*, is desirable. It prevents forgery of signatures on messages that could have been signed previously. Applications of strongly unforgeable signature schemes include signcryption [2], encryption secure against chosen ciphertext attacks [10,7], group signature [5,3], authenticated group key exchange [15] and etc. [6]. Unfortunately, many signature schemes in the literature are not strongly unforgeable. Recently, some techniques [6,26] have been proposed to convert existing schemes to strongly unforgeable ones. However, these techniques require to add some supplementary parameters onto the original key pairs of the signature schemes. This may introduce some inconvenience or operational issue in practice, for example, new public key certificates may need to be requested for those augmented public keys.

A *Generic* and *Universal* Transformation. In this paper, we present a new generic transformation which converts *any* signature scheme to a strongly unforgeable one. When comparing with existing techniques [6,26] which are based on trapdoor hash functions, our method has the following merits.

1. The transformation adds *no* additional component into the original public/private key pair; and
2. the transformation is *universal* in the sense that the same transformation technique can be used to convert schemes in other cryptographic settings to strongly unforgeable ones. These cryptographic settings include identity-based signature [24], certificateless signature [1] and many others (Sec. 4).

Furthermore, a strongly-unforgeable signature scheme obtained from our transformation can also be used as an *on-line/off-line* signature [11,25]. Most of the computational-intensive part of the signing process can be done off-line, and this leaves only a little work to be carried out on-line (essentially, only one hash evaluation is left to be done). This helps improve the efficiency of the signing process significantly.

Strong One-time Signature. Our transformation is based on strong one-time signature. A strong one-time signature scheme is a signature scheme which prevents the adversary, making *at most one* signing query, from producing a new signature on a message that could have already been signed. Currently, almost all the one-time signature schemes in the literature [20,16,11,21] have only been shown to be one-time unforgeable rather than strongly one-time unforgeable, that is, they are only ensured to prevent forgery of signatures on new messages not previously signed. The transformation technique to strong one-time signature proposed in [12] requires $O(\ell)$ universal one-way hash functions [18] where ℓ is the length of messages to be signed. In this paper, we propose a simple modification of the method in [12] that improves the efficiency greatly by requiring only *one* collision-resistant hash function.

Related Work. At PKC 2006, Boneh, Shen and Waters [6] presented a transformation technique which converts a large class of existentially unforgeable signature schemes (in the sense of [13]) into strongly unforgeable ones. Their transformation is based on trapdoor hash functions and applies to a class of signature schemes, named *partitioned* signatures. A signature is said to be partitioned if (1) part of the signature, denoted by σ_2, is independent of the message m, and (2) given m and σ_2, the signature can be fully determined. Although many standard signature schemes fall into this class, as the authors pointed out in [6], DSS [19] may not be partitioned[3].

Recently, Steinfeld, Pieprzyk and Wang [26] proposed another transformation technique based on trapdoor hash functions. Their technique can convert *any* (standard) signature scheme to a strongly unforgeable one. Their idea is to use two trapdoor hash functions and apply the '*hash-then-switch*' method to protect the entire signature (rather than only part of it) from modification. They showed that any valid forgery against strong unforgeability would contradict either the existential unforgeability of the original scheme or the collision-resistance of the underlying trapdoor hash functions. In both of the transformations, additional public and private key components for the underlying trapdoor hash functions have to be added into the public and private keys of the original signature scheme, respectively. Furthermore, it is not known if their techniques can be applied to signature schemes in other cryptographic settings, for example, in certificateless cryptography [1].

Earlier in [12], Goldreich showed the existence of strongly unforgeable signature schemes based on one-way functions. First, a *strong* one-time signature scheme is constructed from a one-time signature scheme (that follows the 'one-way function paradigm' [11,12]). The construction is based on universal one-way hash functions [18,12] which in turn can be constructed from one-way functions. Then, by applying the 'authentication-tree' method [12], a strongly unforgeable signature scheme can be constructed. However, this is only a theoretical construction for the feasibility, and is inefficient.

Paper organization. In next section, we review the definitions of unforgeable and strongly unforgeable signature schemes and the respective definitions for one-time signature schemes. Our generic transformation technique is proposed and shown to be secure in Sec. 3. In Sec. 4, the generic transformation is extended to certificateless signatures and identity-based signatures, and extensions to other settings are discussed. In Sec. 5, we propose a method to convert any one-time signature scheme following the one-way function paradigm into a strong one-time unforgeable one, and discuss its efficiency. In Sec. 6, we show how to use our generic transformation to construct an efficient *on-line/off-line* signature scheme, and conclude the paper.

2 Preliminaries

A signature scheme SIG consists of three (probabilistic) polynomial-time algorithms, KG, Sign and Vrfy, which are key generation, signature generation

[3] Readers may also refer to [26] for some additional discussions about this.

and verification, respectively. *Existential unforgeability against adaptive chosen message attacks* [13] for SIG can be defined using the following game called **Game-General**:

Setup: A public/private key pair $(pk, sk) \leftarrow \mathrm{KG}(1^k)$ is generated and adversary $\mathcal{A}$ is given the public key pk.

Query: $\mathcal{A}$ runs for time t and issues q signing queries to a signing oracle in an adaptive manner, that is, for each i, $1 \leq i \leq q$, $\mathcal{A}$ chooses a message $m^{(i)}$ based on the message-signature pairs that $\mathcal{A}$ has already seen, and obtains in return a signature $\sigma^{(i)}$ on $m^{(i)}$ from the signing oracle (i.e., $\sigma^{(i)} = \mathrm{Sign}(sk, m^{(i)})$).

Forge: $\mathcal{A}$ outputs a forgery (m^*, σ^*) and halts. $\mathcal{A}$ wins if
- σ^* is a valid signature on message m^* under the public key pk, i.e., $\mathrm{Vrfy}(pk, \sigma^*, m^*) = 1$; and
- m^* has never been queried, i.e., $m^* \notin \{m^{(1)}, m^{(2)}, \cdots, m^{(q)}\}$.

Definition 1 (Unforgeability). *A signature scheme SIG = (KG, Sign, Vrfy) is (t, q, ε)-existentially unforgeable against adaptive chosen message attacks (or **unforgeable**, in short), if any adversary with run-time t wins in **Game-General** with probability at most ε after issuing at most q signing queries.*

One of the restrictions for adversary $\mathcal{A}$ in **Game-General** is that the forging message m^* must be new and has not been signed. We can relax this restriction to obtain the notion of ***strong*** *existential unforgeability against adaptive chosen message attacks*, such that $\mathcal{A}$ forges a new valid signature on a message that could have been signed previously. We refer to this new game as **Game-Strong** which is defined as follows.

The **Setup** phase and **Query** phase are the same as in **Game-General**.

Forge: $\mathcal{A}$ outputs a forgery (m^*, σ^*) and halts. $\mathcal{A}$ wins if
- σ^* is a valid, i.e., $\mathrm{Vrfy}(pk, \sigma^*, m^*) = 1$; and
- $(m^*, \sigma^*) \notin \{ (m^{(i)}, \sigma^{(i)}) \}_{i \in \{1,2,\cdots,q\}}$.

Definition 2 (Strong Unforgeability). *A signature scheme SIG = (KG, Sign, Vrfy) is (t, q, ε)-strongly existentially unforgeable against adaptive chosen message attacks (or **strongly unforgeable**, in short), if any adversary with run-time t wins in **Game-Strong** with probability at most ε after issuing at most q signing queries.*

In our generic transformation proposed later in this paper, one of the primitives we use is the ***strong*** *one-time signature*. Informally, a strong one-time signature scheme is a signature scheme, but each private key is used only once for signature generation. We require that given a (one-time) public key, the adversary is only allowed to make *at most one* signing query before producing a forgery on a message that could have been queried previously. Formally, we define the following game called **Game-StrongOneTime**.

The **Setup** phase and **Forge** phase are the same as in **Game-Strong**.
Query: same as in **Game-Strong**, except that $q = 1$.

Definition 3 (Strong One-Time Unforgeability). *A signature scheme SIG = (KG, Sign, Vrfy) is a (t, ε)-*strong one-time signature *scheme, if any adversary with run-time t wins **Game-StrongOneTime** with probability at most ε.*

Similarly, a one-time signature (rather than strong) can be defined by strengthening the restriction for $\mathcal{A}$ so that the forgery must contain a new message which has not been signed previously.

3 Our Generic Transformation

In this section, we describe our generic transformation which converts *any* unforgeable signature scheme to a *strongly unforgeable* one. This transformation can be considered as a sequential composition of the original (standard) signature and a strong one-time signature. First, we use the original signature scheme to generate a "certificate" on a freshly generated one-time public key. Then, we use the strong one-time signature scheme to generate a signature on some message and the "certificate". Below are the details.

Let $\mathrm{SIG}' = (\mathrm{KG}', \mathrm{Sign}', \mathrm{Vrfy}')$ be a signature scheme which is unforgeable (Def. 1). Let $\mathrm{SIG}_{OT} = (\mathrm{KG}_{OT}, \mathrm{Sign}_{OT}, \mathrm{Vrfy}_{OT})$ be a strong one-time signature scheme (Def. 3). The generic transformation is described as follows.

KG: Generate a public/private key pair $(pk', sk') \leftarrow \mathrm{KG}'(1^k)$, and set public key $pk = pk'$ and private key $sk = sk'$.

Sign: On input private key sk and a message m, the following steps are carried out and a signature σ is generated.

$$\begin{aligned}(vk_{OT}, sk_{OT}) &\leftarrow \mathrm{KG}_{OT}(1^k)\\ \sigma_1 &\leftarrow \mathrm{Sign}'(sk, vk_{OT})\\ \sigma_2 &\leftarrow \mathrm{Sign}_{OT}(sk_{OT}, m\|\sigma_1)\\ \sigma &\leftarrow (\sigma_1, \sigma_2, vk_{OT})\end{aligned}$$

Vrfy: On input public key pk, message m and signature $\sigma = (\sigma_1, \sigma_2, vk_{OT})$, $b_1 \wedge b_2$ is returned where

$$\begin{aligned}b_1 &\leftarrow \mathrm{Vrfy}'(pk, \sigma_1, vk_{OT})\\ b_2 &\leftarrow \mathrm{Vrfy}_{OT}(vk_{OT}, \sigma_2, m\|\sigma_1).\end{aligned}$$

Theorem 1. *The generic transformation described above is a (t, q, ε)-strongly unforgeable scheme (Def. 2), provided that SIG' is a $(t, q, \varepsilon/2)$-unforgeable signature scheme (Def. 1) and SIG_{OT} is a $(t, \varepsilon/2q)$-strong one-time signature scheme (Def. 3).*

Proof. Suppose there exists an adversary $\mathcal{A}$ in **Game-Strong** that runs for time t, issues at most q signing queries[4] and breaks the strong unforgeability (Def. 2) of the generic transformation with probability at least ε. We show how to construct adversaries $\mathcal{B}$ and $\mathcal{C}$ that break the strong one-time unforgeability (Def. 3) of SIG_{OT} and the existential unforgeability (Def. 1) of SIG', respectively, such that either $\mathcal{B}$ wins in **Game-StrongOneTime** with probability at least $\varepsilon/2q$ or $\mathcal{C}$ wins in **Game-General** with probability at least $\varepsilon/2$, and both of them run for time negligibly greater than t.

Let (m^*, σ^*) be the forgery of $\mathcal{A}$, where $\sigma^* = (\sigma_1^*, \sigma_2^*, vk_{OT}^*)$. For $i = 1, 2, \cdots, q$, let $m^{(i)}$ be the i-th (distinct) query message of $\mathcal{A}$ and $\sigma^{(i)} = (\sigma_1^{(i)}, \sigma_2^{(i)}, vk_{OT}^{(i)})$ the corresponding signature. We define two events, E_1 and E_2. E_1 is that (m^*, σ^*) is valid and $vk_{OT}^* = vk_{OT}^{(i)}$ for some i $(1 \leq i \leq q)$. E_2 is that (m^*, σ^*) is valid and $vk_{OT}^* \neq vk_{OT}^{(i)}$ for all $1 \leq i \leq q$. As $\Pr[\text{E}_1] + \Pr[\text{E}_2] = \Pr[\mathcal{A} \text{ wins}]$, if $\mathcal{A}$ wins in **Game-Strong**, it must be that either event E_1 or event E_2 occurs. Since $\mathcal{A}$ wins with probability ε, it follows that one of the two events occurs with probability at least $\varepsilon/2$. In the simulations below, $\mathcal{A}$ will be run by each of the adversaries $\mathcal{B}$ and $\mathcal{C}$ which we will construct. If E_1 (respectively, E_2) occurs with probability $\varepsilon/2$, then $\mathcal{B}$ breaks the strong one-time unforgeability of SIG_{OT} with probability $\varepsilon/2q$ (respectively, $\mathcal{C}$ breaks the existential unforgeability of SIG' with probability $\varepsilon/2$).

Adversary $\boldsymbol{\mathcal{B}}$. Given a challenge one-time public key vk_{OT}, which is a random instance in the corresponding key space, and a (one-time) signing oracle $\text{OSign}_{vk_{OT}}$, adversary $\mathcal{B}$ proceeds as below to attack against the strong one-time unforgeability of SIG_{OT}:

Setup: $\mathcal{B}$ runs $\text{KG}(1^k)$ to generate a key pair (pk, sk) for the generic transformation, selects uniformly at random i from $\{1, 2, \cdots, q\}$, and runs $\mathcal{A}$ on input the public key pk.

Query: When $\mathcal{A}$ issues the j-th $(j \neq i)$ signing query, $\mathcal{B}$ simulates the signing oracle as if the answer is generated by the real signer. That is, $\mathcal{B}$ responds as follows:

- Run $\text{KG}_{OT}(1^k)$ to generate a one-time key pair $(vk_{OT}^{(j)}, sk_{OT}^{(j)})$;
- Compute $\sigma_1^{(j)} \leftarrow \text{Sign}'(sk, vk_{OT}^{(j)})$;
- Compute $\sigma_2^{(j)} \leftarrow \text{Sign}_{OT}(sk_{OT}^{(j)}, m^{(j)} \| \sigma_1^{(j)})$;
- Return $\sigma^{(j)} \leftarrow (\sigma_1^{(j)}, \sigma_2^{(j)}, vk_{OT}^{(j)})$ to $\mathcal{A}$.

When $\mathcal{A}$ issues the i-th signing query, $\mathcal{B}$ responds as follows:

- Set $vk_{OT}^{(i)} = vk_{OT}$;
- Compute $\sigma_1^{(i)} \leftarrow \text{Sign}'(sk, vk_{OT}^{(i)})$;
- Obtain a signature $\sigma_2^{(i)}$ on $m^{(i)} \| \sigma_1^{(i)}$ by querying the one-time signing oracle $\text{OSign}_{vk_{OT}}$.
- Return $\sigma^{(i)} \leftarrow (\sigma_1^{(i)}, \sigma_2^{(i)}, vk_{OT}^{(i)})$ to $\mathcal{A}$.

Forge: After $\mathcal{A}$ outputs a forgery (m^*, σ^*) where $\sigma^* = (\sigma_1^*, \sigma_2^*, vk_{OT}^*)$, $\mathcal{B}$ outputs $((m^* \| \sigma_1^*), \sigma_2^*)$ as its forgery for SIG_{OT}.

[4] W.l.o.g., we assume that $\mathcal{A}$ makes exactly q distinct signing queries.

Since $\mathcal{B}$'s run is essentially a run of $\mathcal{A}$, if $\mathcal{A}$ runs for time t, so does $\mathcal{B}$. Also, $\mathcal{B}$ perfectly simulates the signing oracle for $\mathcal{A}$ as $\mathcal{B}$ follows exactly the signing process except when answering the i-th query. For the i-th query, $\mathcal{B}$ makes a black-box access to its one-time signing oracle $\mathrm{OSign}_{vk_{OT}}$ and the oracle's answer is indistinguishable from those signatures generated by a real signer with respect to the same one-time public key vk_{OT}. Thus, $\mathcal{A}$'s view is identical to that in a real attack (i.e. an exact simulation of **Game-Strong**) and is independent of the choice of i. This implies that $\mathcal{A}$ will succeed with the same probability as in a real attack.

Now we analyze the validity of $\mathcal{B}$'s output under the conditions that event E_1 occurs and $\mathcal{B}$'s guess of i is correct (i.e. $vk^*_{OT} = vk^{(i)}_{OT} = vk_{OT}$). If $(m^* \| \sigma^*_1) \neq (m^{(i)} \| \sigma^{(i)}_1)$, by the validity of (m^*, σ^*), we have that $\mathrm{Vrfy}_{OT}(vk^*_{OT}, \sigma^*_2, m^* \| \sigma^*_1) = 1$, hence, $((m^* \| \sigma^*_1), \sigma^*_2)$ is certainly a valid forgery for SIG_{OT}. Then we come to the case that $(m^* \| \sigma^*_1) = (m^{(i)} \| \sigma^{(i)}_1)$. Due to the validity of (m^*, σ^*), it must be that $\sigma^*_2 \neq \sigma^{(i)}_2$. Therefore, $((m^* \| \sigma^*_1), \sigma^*_2)$ is also a valid forgery for SIG_{OT}, which contradicts the strong unforgeability of SIG_{OT}.

The probability that the choice of i is exactly the one such that $vk^*_{OT} = vk^{(i)}_{OT}$ is $1/q$. Therefore, if event E_1 occurs with probability at least $\varepsilon/2$, $\mathcal{B}$ which runs for time t breaks the security of SIG_{OT} with probability at least $\varepsilon/2q$.

Adversary $\mathcal{C}$. Given a public key pk' of SIG', which is chosen from the output space of $\mathrm{KG}'(1^k)$ at random, and a signing oracle $\mathrm{OSign}_{pk'}$, adversary $\mathcal{C}$ proceeds as below to attack against the existential unforgeability of SIG'.

Setup: $\mathcal{C}$ sets $pk = pk'$, and runs $\mathcal{A}$ on input public key pk. Note that $\mathcal{C}$ does not know the corresponding private key sk.

Query: When $\mathcal{A}$ issues a signing query on some message m, $\mathcal{C}$ simulates the answer as follows:
- Run $\mathrm{KG}_{OT}(1^k)$ to generate a one-time key pair (vk_{OT}, sk_{OT});
- Query the signing oracle $\mathrm{OSign}_{pk'}$ for a signature σ_1 on vk_{OT};
- Compute $\sigma_2 \leftarrow \mathrm{Sign}_{OT}(sk_{OT}, m \| \sigma_1)$;
- Return $\sigma \leftarrow (\sigma_1, \sigma_2, vk_{OT})$.

Forge: After $\mathcal{A}$ outputs a forgery (m^*, σ^*) where $\sigma^* = (\sigma^*_1, \sigma^*_2, vk^*_{OT})$, $\mathcal{C}$ outputs (vk^*_{OT}, σ^*_1) as its forgery for SIG'.

If event E_2 occurs, vk^*_{OT} is a new one-time public key which has not been used by $\mathcal{C}$ in any of the previous queries to its signing oracle $\mathrm{OSign}_{pk'}$. By the validity of $(m^*, (\sigma^*_1, \sigma^*_2, vk^*_{OT}))$ under the public key pk, we have $\mathrm{Vrfy}'(pk, vk^*_{OT}) = 1$. Therefore, (vk^*_{OT}, σ^*_1) is a valid forgery for SIG'.

Since $\mathcal{C}$'s run is essentially a run of $\mathcal{A}$, if $\mathcal{A}$ runs for time t, so does $\mathcal{C}$. Also, $\mathcal{C}$ issues only one signing query to its own oracle $\mathrm{OSign}_{pk'}$ when answering a signing query issued by $\mathcal{A}$, if $\mathcal{A}$ issues q signing queries, so does $\mathcal{C}$. Furthermore, $\mathcal{C}$ perfectly simulates the signing oracle for $\mathcal{A}$ because $\mathcal{C}$ simply follows the signing procedure with the only exception that $\mathcal{C}$ uses its signing oracle $\mathrm{OSign}_{pk'}$ to generate σ_1, and the oracle's output is perfectly indistinguishable from signatures generated by real signers of SIG' with respect to the same public key. Therefore,

$\mathcal{A}$ will succeed with the same probability as that in a real attack. If event E_2 occurs with probability $\varepsilon/2$, $\mathcal{C}$ breaks the existential unforgeability of SIG$'$ with probability $\varepsilon/2$ as well.

This concludes that if $\mathcal{A}$ (t, q, ε)-breaks the strong unforgeability of SIG, either $\mathcal{B}$ $(t, \varepsilon/2q)$-breaks the strong one-time unforgeability of SIG_{OT}, or $\mathcal{C}$ $(t, q, \varepsilon/2)$-breaks the existential unforgeability of SIG$'$. □

The efficiency of the generic transformation depends very much on that of the underlying strong one-time signature scheme SIG_{OT}. As we can see, the generic transformation adds one key generation and one signing operation of SIG_{OT} onto the original signing process of SIG$'$, and one verification operation of SIG_{OT} onto the original verification process. According to [11,21], SIG_{OT} can usually be implemented with very efficient key generation, signing and verifying processes, and short signatures. In addition, the two verification operations of the generic transformation, one for checking σ_1 and the other for σ_2, can be carried out in parallel, that may also be used to improve efficiency. In the next section, we show that the generic transformation can also be extended to transform signatures in other settings such as certificateless signature, identity-based signature and many others, to strongly unforgeable ones. To the best of our knowledge, it is not known if this extent of versatility can also be supported by comparable methods such as [6,26].

4 Extensions to Other Cryptographic Settings

In the above, we show how to transform an unforgeable signature scheme to a strongly unforgeable one, under the conventional public key infrastructure. That is, the public key of an entity is assumed to be publicly known, for example due to the presence of a certificate issued by a Certification Authority. In this section, we show that the generic transformation technique can be extended directly for converting signature schemes in other cryptographic settings to their strongly unforgeable ones in some similar context.

4.1 Certificateless Signature and ID-based Signature

Certificateless cryptography, introduced by Al-Riyami and Paterson [1], is intended to solve the key escrow issue that is inherent in ID-based cryptography. In the following, we adopt the simplified five-algorithm definition of [14] for specifying a certificateless signature scheme. These five algorithms are: key generation, KG, user partial key generation, PartialKeyGen, user secret key generation, UserKeyGen, signature generation, Sign and verification, Vrfy. Two games are considered for the security of a certificateless signature: Game-I and Game-II. Adversary $\mathcal{A}_I$ in Game-I can compromise user secret key, replace user public key, but cannot get master secret key nor user partial key. Adversary $\mathcal{A}_{II}$ in Game-II models a dishonest KGC which knows master secret key and partial keys of all users but cannot get access to user secret key nor replace user public key. Informally, a certificateless signature is said to be $(t, q_s, q_o, \varepsilon)$-existentially unforgeable (in Game-I, or Game-II) if no adversary with run-time t issuing at most q_s signing queries and at most q_o other oracle queries (such as CreateUser, RevealPartialKey and others specified in the games) succeeds in forging a signature on a new identity-message pair (ID^*, m^*) with probability at least ε. A certificateless signature is said to be *strongly unforgeable* (in Game-I, or Game-II) if it can prevent forgery of new signatures on identity-message pairs which could have been signed previously.

To the best of our knowledge, no certificateless signature scheme in the literature has formally considered the strong unforgeability. Also note that the generic composition of certificateless signature scheme from a standard signature scheme and an ID-based signature scheme proposed in [14] does not ensure strong unforgeability even if we assume that both of the underlying primitives are strongly unforgeable. It remains open to construct a generic composition of strongly unforgeable certificateless signature scheme. In the following, we show that the generic transformation technique proposed in Sec. 3 can be used directly to solve this problem.

Let KG_{CL}, $\mathrm{PartialKeyGen}_{CL}$, $\mathrm{UserKeyGen}_{CL}$, Sign_{CL} and Vrfy_{CL} constitute a certificateless signature scheme SIG_{CL}.

KG: $(mpk, msk) \leftarrow \mathrm{KG}_{CL}(1^k)$
PartialKeyGen: $partialkey[ID] \leftarrow \mathrm{PartialKeyGen}_{CL}(msk, ID)$.
UserKeyGen: $(upk[ID], usk[ID]) \leftarrow \mathrm{UserKeyGen}_{CL}(mpk, ID)$.
Sign: For a message m and identity ID, a signature σ is generated as follows.

$$\begin{aligned}(vk_{OT}, sk_{OT}) &\leftarrow \mathrm{KG}_{OT}(1^k)\\ \sigma_1 &\leftarrow \mathrm{Sign}_{CL}(usk[ID], partialkey[ID], vk_{OT})\\ \sigma_2 &\leftarrow \mathrm{Sign}_{OT}(sk_{OT}, m\|ID\|\sigma_1)\\ \sigma &\leftarrow (\sigma_1, \sigma_2, vk_{OT})\end{aligned}$$

Vrfy: Given master public key mpk, message m, identity ID, user public key $upk[ID]$ and signature $\sigma = (\sigma_1, \sigma_2, vk_{OT})$, $b_1 \wedge b_2$ is returned, where $b_1 \leftarrow \mathrm{Vrfy}_{CL}(mpk, ID, upk[ID], \sigma_1, vk_{OT})$ and $b_2 \leftarrow \mathrm{Vrfy}_{OT}(vk_{OT}, \sigma_2, m\|ID\|\sigma_1)$.

Theorem 2. *The generic transformation above is $(t, q_s, q_o, \varepsilon)$-strongly unforgeable in Game-I (respectively, Game-II) if SIG_{CL} is $(t, q_s, q_o, \varepsilon/2)$-existentially unforgeable in Game-I (respectively, Game-II), and SIG_{OT} is a $(t, \varepsilon/2q_s)$-strong one-time signature scheme, where q_s and q_o are the maximum numbers of signing queries and all the other oracle queries, respectively.*

Similar to the proof of Theorem 1, to prove the strong unforgeability of the generic transformation in Game-I (respectively, Game-II), we distinguish between two events: (1) the forgery of $\mathcal{A}_I$ is valid and the one-time public key in the forgery appears in some previous answer of the signing queries; (2) the forgery is valid but the one-time public key in the forgery is new. For the first event, we can construct an efficient adversary $\mathcal{B}_{CL}$ to break the strong one-time unforgeability of SIG_{OT}. Note that, with the knowledge of master secret key *msk*, $\mathcal{B}_{CL}$ can answer queries to all the other oracles (i.e., CreateUser, RevealPartialKey, RevealSecretKey and ReplaceKey) besides the Signing oracle, and it can issue a signing query to its own oracle in this case. For the second event, we can construct an efficient adversary $\mathcal{C}_{CL}$ to break the existential unforgeability of SIG_{CL}. Detailed proof is similar to that of Theorem 1, so we omit it here.

In the generic transformation above, we include identity ID in the message when generating σ_2. This allows us to follow the two-event approach in the proof of Theorem 1 and therefore simplifies the proof for this theorem.

An ID-based signature scheme, introduced by Shamir [24], comprises four efficient algorithms, master key generation, KG, user secret key generation, Extract, signature generation, Sign and verification, Vrfy. Such a scheme is said to be $(t, q_s, q_e, \varepsilon)$-existentially unforgeable against adaptive chosen message and identity attacks if no adversary, which runs for time t and issues at most q_e Extract queries and at most q_s Signing queries, succeeds in forging a signature on a new identity-message pair with probability at least ε. Readers may refer to [4] for details.

Let SIG_{IBS}=(KG_{IBS}, Extract_{IBS}, Sign_{IBS}, Vrfy_{IBS}) be an ID-based signature scheme. We can apply the same transformation technique described above to convert SIG_{IBS} to a strongly unforgeable one (i.e. preventing adversaries from forging new signatures on identity-message pairs that could have been signed previously). We omit the details of the transformation as it can easily be obtained from the above. Similarly, we have the following theorem:

Theorem 3. *The new signature scheme obtained from the generic transformation in the setting of ID-based cryptography is $(t, q_s, q_e, \varepsilon)$-strongly existentially unforgeable against adaptive chosen message and identity attacks, provided that SIG_{IBS} is an ID-based signature scheme that is $(t, q_s, q_e, \varepsilon/2)$-existentially unforgeable against adaptive chosen message and identity attacks, and SIG_{OT} is a $(t, \varepsilon/2q_s)$-strong one-time signature scheme, where q_e and q_s are the maximum numbers of the Extract queries and Signing queries, respectively.*

The proof is similar to that of Theorem 1 and is omitted here.

4.2 Other Signatures

In our generic transformation and its extensions to certificateless and ID-based settings, we can see that our technique makes no modification on the internal of the original signature scheme, but uses it as a *black-box* to sign a freshly-generated one-time public key. This does not rely on any additional property of the original scheme except the existential unforgeability. Besides, our transformation does not modify the public/private key pair nor information concerning users' identities. Therefore, after describing the generic transformation in Sec. 3, the extensions to certificateless signature and ID-based signature become straightforward. We believe that this generic transformation technique can also be applied to other types of signature schemes, such as group signature [8], ring signature [22], proxy signature [17] and some others. We leave this as our further studies.

5 Strong One-Time Signature

The security of our generic transformation relies on the existence of strong one-time signature schemes. In this section, we show how to transform any one-time signature scheme which follows the 'one-way function paradigm' [11,12] to a strong one-time version. We also evaluate the performance of an instantiation which is based on a scheme by Reyzin and Reyzin [21].

5.1 From One-Time to Strong One-Time

Since the introduction of one-time signature [20,16], there have been many schemes of this type proposed, and many of them follow the *one-way function paradigm* [11,12]. Let $f : \{0,1\}^k \to \{0,1\}^\kappa$ be a one-way function. Informally, a scheme that follows the 'one-way function paradigm' has the private key composed of a set of random elements from the domain of f, and the public key composed of evaluations of those private key elements using f. To generate a signature, a subset of private key elements is chosen in accordance with the message, and the subset is considered to be the signature, which can then be verified through evaluations of f and comparison with the corresponding elements in the public key. Below is an example from [12].

KG: On input 1^k, randomly select 2ℓ strings of length k, $s_1^0, s_1^1, \cdots, s_\ell^0, s_\ell^1$, where $\ell = \ell(k)$ for some polynomial $\ell : \mathbb{N} \to \mathbb{N}$, and compute $v_i^b = f(s_i^b)$, for $b = 0,1$ and $i = 1,2,\cdots,\ell$. The public key vk_{OT} is $((v_1^0, v_1^1), \cdots, (v_\ell^0, v_\ell^1))$ and the private key sk_{OT} is $((s_1^0, s_1^1), \cdots, (s_\ell^0, s_\ell^1))$.

Sign: For an ℓ-bit message $m = b_1 b_2 \cdots b_\ell$ where $b_i \in \{0,1\}$ for $i = 1,\cdots,\ell$, the signature σ is $(s_1^{b_1}, \cdots, s_\ell^{b_\ell})$.

Vrfy: For message $m = b_1 b_2 \cdots b_\ell$ and signature $\sigma = (\sigma_1, \sigma_2, \cdots, \sigma_\ell)$, if $v_i^{b_i} = f(\sigma_i)$ for all $i = 1,2,\cdots,\ell$, output 1; otherwise, output 0.

It is easy to show that the scheme above is a one-time signature scheme as any forgery on a new message would lead to the inversion of f. However, it is not ensured that no forgery can be made on a message that has already been signed.

To transform a one-time signature that follows the one-way function paradigm to a strong one-time signature (in the sense of Def. 3), a method is proposed in [12], which is based on Universal One-Way Hash Functions (UOWHF, in short) [18]. Although UOWHF can be constructed directly from one-way functions, the resulting strong one-time signature scheme suffers from a much larger public key, which includes the description of 2ℓ randomly selected UOWHFs in addition to the one-way-function evaluations of the 2ℓ private key elements.

To solve this problem, we propose another method. Our method is to replace f with a randomly selected collision-resistant hash function h. In this conversion, only the description of *one* (collision-resistant) hash function is added into the public key vk_{OT} rather than that of 2ℓ UOWHFs as in the method of [18]. On the security of our conversion, as the minimal assumption currently known for constructing a collision-resistant hash function is the existence of claw-free permutations, which is stronger than that of the existence of one-way functions, it follows that our generic transformation proposed in Sec. 3 is also based on the existence of claw-free permutations. In general, we use signature schemes to sign arbitrary-length messages. The standard technique, so-called '*hash-and-sign*' paradigm [9], we can use it to apply a collision-resistant hash function to the message and then sign the resulting hash value. Therefore, the existence of claw-free permutations is generally an assumption for the security of the original unforgeable signature schemes already.

The following theorem shows that our method yields a strong one-time signature scheme.

Theorem 4. *Let SIG'_{OT} be a secure one-time signature scheme that follows the one-way function paradigm, and let SIG_{OT} be the resulting scheme by replacing the one-way function f with a randomly selected hash function h. Then SIG_{OT} is strongly unforgeable against (adaptive) chosen one-message attack in the sense of Def. 3, provided that h is collison-resistant and preimage-resistant (or, one-way).*

Proof (Sketch). First, as the collision resistance of a hash function implies one-wayness (please also refer to the remark below), if we view the hash function h as a one-way function, then the new scheme SIG_{OT} is equivalent to SIG'_{OT}, thus is also unforgeable against one-time chosen message attacks (i.e., SIG_{OT} is also a one-time signature scheme).

Let $\mathcal{A}$ be an adversary which runs for time t and breaks the strong one-time unforgeability (in the sense of Def. 3) of SIG_{OT} with probability at least ε. Note that in general, if messages are ℓ bits long, by no means the signatures must have exactly ℓ private key components. Hence, we use another notation d to denote the number of private key components in a signature of SIG_{OT}. For example, in HORS [21], the value of d is much less than ℓ. Now, suppose $m = b_1 b_2 \cdots b_\ell$ is the ℓ-bit message in the query of $\mathcal{A}$ and $\sigma = (\sigma_1, \cdots, \sigma_d)$ is the signature on m answered by the Signing oracle, where $\sigma_i \in \{0,1\}^k$, for $i = 1, \cdots, d$. Let (m^*, σ^*) be $\mathcal{A}$'s forgery, where $m^* = b_1^* b_2^* \cdots b_\ell^*$ and $\sigma^* = (\sigma_1^*, \cdots, \sigma_d^*)$. Then either of the following events would occur with probability at least $\varepsilon/2$:

1. If $m^* \neq m$, there exists at least one i such that $b_i^* \neq b_i$. This would lead to the break of the preimage-resistance of function h.
2. If $m^* = m$, it must hold that $\sigma^* \neq \sigma$, which implies that there exists at least one i $(1 \leq i \leq d)$ such that $\sigma_i^* \neq \sigma_i$. Such a pair forms a collision for h. □

Remark: It is worthwhile to notice the relation between collision-resistance and one-wayness. Suppose h is a hash function compressing k-bit strings into κ-bit strings. According to [23], ε-collision-resistance of h implies $(2\varepsilon + 2^{\kappa-k})$-one-wayness. This implies that the input length k should be larger than the output length κ by a sufficient margin for ensuring the one-wayness of h.

5.2 An Instantiation of Strong One-time Signature

Most of the current constructions of one-time signature follow the one-way function paradigm and are very efficient as they do not carry out any public key cryptographic operation. One of the most efficient one-time signature schemes that follows the one-way function paradigm is the HORS proposed by Reyzin and Reyzin [21]. HORS is in fact an r-time signature scheme. A single public key can be used for r times, in contrast to only one time in a one-time signature scheme. The security of HORS relies on the existence of *Subset Resilient* functions [21]. For $r > 1$, realizing such functions using only conventional complexity-theoretic assumptions without random oracles is still an open problem. Fortunately, in our case, each key pair of HORS only needs to be used once (i.e. $r = 1$), and so collision-resistant hash families are enough for realizing the subset resilient functions.

Let k be the security parameter. Two auxiliary parameters, t and d, are chosen such that $d \cdot \log t = \ell$, where ℓ is the output length of a collision-resilient hash function $Hash$. The private key contains essentially t strings of k bits (along with the value of d) and the public key contains the evaluations of these t strings using a one-way function $f : \{0,1\}^k \rightarrow \{0,1\}^\kappa$. The signing process requires just one $Hash$ operation and produces a signature which is a sequence of d out of t strings of the private key. The verification process requires only d evaluations of f and one of $Hash$.

According to Theorem 4, to convert HORS into a strong one-time signature scheme, we can replace f with a randomly chosen collision-resistant (and one-way) hash function h. As suggested in [21], the security parameter k is set 80, and the output length κ of h is 160-bit. According to the remark at the end of Theorem 4, the one-wayness of h may not be ensured by collision-resistance in this case. Hence we suggest that a collision-resistant and one-way hash function h should be used. For example, we employ a collision-resistant hash function $h : \{0,1\}^{240} \rightarrow \{0,1\}^{160}$, that is, setting k to 240. We refer to the resulting scheme as '*strong HORS*'.

Public Key Size and Signature Size. In our generic transformation, signature size depends on the public key size and signature size of the underlying strong one-time scheme, but not on the private key size. Hence in the following, we only evaluate the public key size and signature size of *strong HORS*. As we can see, the public key is $t \cdot \kappa$-bit-long and the signature is $d \cdot k$-bit-long. These sizes could be large in practice, for example when $t = 256$ and $d = 20$ (as suggested in [21]). We note that almost all the current one-time signature schemes suffer from this drawback, and we believe that improving the signature size of a one-time signature scheme is of independent interest.

Remark: Also note that any strongly unforgeable signature scheme (in the sense of Def. 2) is also a strong one-time signature scheme (Def. 3). Hence the drawback mentioned above can easily be solved by using a strongly unforgeable signature scheme that has short public key and signature to instantiate SIG_{OT} in our generic transformation. We should note that the tradeoff is on the computational efficiency.

6 Concluding Remarks

On-line/Off-line. As mentioned before, the signature generation of our generic transformation can be considered as a sequential composition of the original signature scheme and a strong one-time signature scheme. In the first phase, a one-time public key is freshly generated and signed to create a 'certificate'. In the second phase, the message and the 'certificate' are then signed using the strong one-time signature scheme. The first phase is independent of the message and therefore, can be used directly as the off-line part of an *on-line/off-line* signature scheme [11]. This phase can also be carried out for multiple times, each time, a triple $(vk_{OT}, sk_{OT}, \sigma_1)$ is generated and stored. When a message m is to be signed on-line, an unused triple $(vk_{OT}, sk_{OT}, \sigma_1)$ is selected and a signature σ_2 on $m\|\sigma_1$ is generated under sk_{OT} using the strong one-time signing algorithm. This yields a very efficient on-line operation. For example, the signing process of (strong) HORS [21] is essentially one hash evaluation.

In this paper, we proposed a universal and generic transformation which converts *any* unforgeable signature scheme into a strongly unforgeable one. It is universal in the sense that it can also be applied to signatures in other settings such as ID-based signature, certificateless signature and many others. Our technique does not add any additional parameter into the original public/private key pair and makes no change to the internals of the original signature scheme. On the conversion to strong one-time signature schemes that follow the one-way function paradigm, our method is efficient and does not introduce any additional security assumption in general. Due to the limitation of currently available one-time signature schemes, our generic transformation could have a large signature size when implemented, which is the main drawback. However, by choosing proper parameters and instantiations, our technique can be very efficient for practical use (Sec. 5.2). Finally, thanks to the efficient key generation, signing and verification processes of the strong one-time signature scheme, these make our transformation much more efficient than the comparable transformation techniques [12,6,26].

Acknowledgements

We are grateful to Man Ho Au and Ron Steinfeld for their valuable comments.

References

1. S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *Proc. ASIACRYPT 2003*, pages 452–473. Springer-Verlag, 2003. LNCS 2894.

2. J. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *Proc. EUROCRYPT 2002*, pages 83–107. Springer-Verlag, 2002. LNCS 2332.
3. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Proc. CRYPTO 2000*, pages 255–270. Springer-Verlag, 2000. LNCS 1880.
4. M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. In *Proc. EUROCRYPT 2004*, pages 268–286. Springer-Verlag, 2004.
5. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Proc. CRYPTO 2004*, pages 41–55, 2004. LNCS 3152.
6. D. Boneh, E. Shen, and B. Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In *Proc. of PKC 2006*, pages 229–240. Springer-Verlag, 2006. LNCS 3958.
7. R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Proc. EUROCRYPT 2004*, pages 207–222. Springer-Verlag, 2004. LNCS 3027.
8. D. Chaum and E. V. Heyst. Group signatures. In *Proc. EUROCRYPT 91*, pages 257–265. Springer-Verlag, 1991. LNCS 547.
9. I. Damgård. Collision free hash functions and public key signature schemes. In *Proc. EUROCRYPT 87*, pages 203–216. Springer-Verlag, 1988. LNCS 304.
10. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM J. Computing*, 30(2):391–437, 2000.
11. S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. *J. of Cryptology*, 9(1), 1996.
12. O. Goldreich. *Foundations of Cryptography, volume II, Basic Applications.* Cambridge University Press, 2004.
13. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attack. *SIAM J. Computing*, 17(2):281–308, Apr. 1988.
14. B. C. Hu, D. S. Wong, Z. Zhang, and X. Deng. Key replacement attack against a generic construction of certificateless signature. In *Information Security and Privacy: 11th Australasian Conference, ACISP 2006*, pages 235–246. Springer-Verlag, 2006. LNCS 4058.
15. J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In *Proc. CRYPTO 2003*, pages 110–125. Springer-Verlag, 2003. LNCS 2729.
16. L. Lamport. Constructing digital signatures from a one way function. Technical Report Technical Report CSL-98, SRI International, Oct. 1979.
17. K. U. M. Mambo and E. Okamoto. Proxy signature: Delegation of the power to sign messages. *IEICE Trans. Fundamentals*, E79-A(9):1338–1353, 1996.
18. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proc. of 21st ACM Symposium on the Theory of Computing*, pages 33–43, 1989.
19. National Institute of Standards and Technology (NIST). *Digital Signature Standard (DSS).* Federal Information Processing Standards Publication 186, Nov. 1994.
20. M. O. Rabin. Digitalized signatures. *Foundations of Secure Computation*, pages 155–168, 1978.
21. L. Reyzin and N. Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In *Information Security and Privacy: 7th Australasian Conference, ACISP 2002*, pages 144–153. Springer-Verlag, 2002. LNCS 2384.
22. R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Proc. ASIACRYPT 2001*, pages 552–565. Springer-Verlag, 2001. LNCS 2248.

23. P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *Proc. Fast Software Encryption 2004*, pages 371–388. Springer-Verlag, 2004. LNCS 3017.
24. A. Shamir. Identity-based cryptosystems and signature schemes. In *Proc. CRYPTO 84*, pages 47–53. Springer, 1984. LNCS 196.
25. A. Shamir and Y. Tauman. Improved online/offline signature schemes. In *Proc. CRYPTO 2001*, pages 355–367. Springer, 2001. LNCS 2139.
26. R. Steinfeld, J. Pieprzyk, and H. Wang. How to strengthen any weakly unforgeable signature into a strongly unforgeable signature. To appear in CT-RSA 2007.