How to Strengthen Any Weakly Unforgeable Signature into a Strongly Unforgeable Signature

Ron Steinfeld[1], Josef Pieprzyk[1], and Huaxiong Wang[1,2]

[1] Centre for Advanced Computing – Algorithms and Cryptography (ACAC)
Dept. of Computing, Macquarie University, North Ryde, Australia
[2] Nanyang Technological University, Singapore
{rons,josef,hwang}@comp.mq.edu.au
http://www.ics.mq.edu.au/acac/

Abstract. Standard signature schemes are usually designed only to achieve *weak* unforgeability – i.e. preventing forgery of signatures on new messages not previously signed. However, most signature schemes are randomised and allow many possible signatures for a single message. In this case, it may be possible to produce a new signature on a previously signed message. Some applications require that this type of forgery also be prevented – this requirement is called *strong* unforgeability.

At PKC2006, Boneh Shen and Waters presented an efficient transform based on any randomised trapdoor hash function which converts a weakly unforgeable signature into a strongly unforgeable signature and applied it to construct a strongly unforgeable signature based on the CDH problem. However, the transform of Boneh et al only applies to a class of so-called *partitioned* signatures. Although many schemes fall in this class, some do not, for example the DSA signature. Hence it is natural to ask whether one can obtain a truly generic efficient transform based on any randomised trapdoor hash function which converts *any* weakly unforgeable signature into a strongly unforgeable one. We answer this question in the positive by presenting a simple modification of the Boneh-Shen-Waters transform. Our modified transform uses two randomised trapdoor hash functions.

Keywords: Digital signature, strong unforgeability, trapdoor hash function, provable security, transform.

1 Introduction

Background. Standard signature schemes are usually designed only to achieve *weak* unforgeability – i.e. preventing forgery of signatures on new messages not previously signed. However, most signature schemes are randomised and allow many possible signatures for a single message. In this case, it may be possible to produce a new signature on a previously signed message. Some applications (such as constructing chosen-ciphertext secure public key encryption schemes [3] and authenticated key exchange protocols [9]) require that this type of forgery also be prevented – this requirement is called *strong* unforgeability.

M. Abe (Ed.): CT-RSA 2007, LNCS 4377, pp. 357–371, 2007.

At PKC2006, Boneh Shen and Waters [2] presented an efficient transform based on a randomised trapdoor hash function which converts a weakly unforgeable signature into a strongly unforgeable signature, and applied it to construct a strongly unforgeable signature based on the CDH problem. However, the transform of Boneh et al only applies to a class of so-called *partitioned* signatures. Although many schemes fall in this class, some do not, for example the DSA signature [12]. Hence it is natural to ask whether one can obtain a truly generic efficient transform which converts *any* weakly unforgeable signature into a strongly unforgeable one.

Our Result. We answer the above question in the positive by presenting a general efficient transform which uses randomised trapdoor hash functions to strengthen any weakly unforegeable signature into a strongly unforgeable signature. Our transform makes use of *two* randomised trapdoor hash functions (rather than just one in the less general transform of [2]). Like the transform of [2], our transform requires the randomised trapdoor hash functions to be *strongly* collision-resistant (by the word *strong* we mean here that it is even hard to find two randomisers $r \neq r'$ and a message m such that (m, r) and (m, r') are a collision-pair for the randomised hash function, whereas usually only *weak* collision resistance is needed, i.e. it is only hard to find collisions with distinct message inputs). For this purpose, we show that a small modification of the efficient VSH randomised trapdoor function, which was shown to be *weakly* collision-resistant in [5], gives a strongly collision-resistant function which can be used in this application.

Relation to Previous Work. The problem of converting a weakly unforgeable signature into a strongly unforgeable one can be trivially "solved" in two known ways. The first solution is to construct a one-way function from the weakly unforgeable signature scheme, and then apply the generic construction of a strongly unforgeable signature from any one-way function (see Section 6.5.2 in [8]), but this results in a very inefficient scheme. The second trivial "solution" is to completely ignore the original weakly unforgeable scheme, make additional assumptions, and directly construct the strongly unforgeable scheme from those assumptions. For example, strongly unforgeable signature schemes from the Strong RSA assumption [7,6], Strong Diffie-Hellman assumption [1] or Computational Diffie-Hellman in a bilinear group [2] are known. However, these all seem quite strong and non-classical additional assumptions, and do not make use of the given weakly unforgeable signature.

In contrast to the above trivial solutions, our weak-to-strong transform (like the BSW transform [2]) makes non-trivial use of the given weakly unforgeable signature scheme, and efficiently reduces the problem of strengthening it to the problem of constructing a seemingly simpler cryptographic primitive, namely a randomised trapdoor hash function. As evidence for the practicality of our transform, we note that randomised trapdoor hash functions are known to be efficiently constructible under the classical factoring or discrete-log assumptions, whereas no efficient direct constructions for strongly unforgeable signatures based on these classical assumptions are known (without random oracles). As an example application, we show (in Section 4.2) how to strengthen the standard Digital Signature Algorithm (DSA) [12], assuming only the weak unforgeability of DSA.

2 Preliminaries

Weak and Strong Unforgeability for Signature Schemes. A signature scheme Σ consists of three efficient algorithms: a *key generation* algorithm KG, a signing algorithm S and a verification algorithm V.

The strong and weak unforgeability of a signature scheme Σ are defined using the following game. A key pair $(sk, pk) = \mathsf{KG}(k)$ is generated, and unforgeability attacker A is given the public key pk. A can run for time t and can issue q signing queries $m_1, \ldots, m_q$ to a signing oracle $\mathsf{S}(sk, \cdot)$, which upon each query m_i returns the signature $\sigma_i = \mathsf{S}(sk, m_i)$ to A. At the end, A outputs a forgery message/signature pair (m^*, σ^*). We say that A succeeds in breaking the *strong unforgeability* of Σ if (m^*, σ^*) passes the verification test V with respect to public key pk, and $(m^*, \sigma^*) \neq (m_i, \sigma_i)$ for all $i = 1, \ldots, q$. We say that A succeeds in breaking the *weak unforgeability* of Σ if (m^*, σ^*) passes the verification test V with respect to public key pk, and $m^* \neq m_i$ for all $i = 1, \ldots, q$. A signature scheme Σ is called (t, q, ϵ) *strongly (respectively, weakly) existentially unforgeable under an adaptive chosen-message attack* if any efficient attacker A with run-time t has success probability at most ϵ in breaking the strong (respectively, weak) unforgeability of Σ.

Randomised Trapdoor (Chameleon) Hash Functions [10,13]. A randomised trapdoor hash function scheme consists of three efficient algorithms: a *key generation* algorithm KG_F, a *hash function evaluation* algorithm F, and a *trapdoor collision finder* algorithm TC_F. On input a security parameter k, the (randomised) key generation algorithm $\mathsf{KG}_F(k)$ outputs a secret/public key pair (sk, pk). On input a public key pk, message $m \in \mathcal{M}$ and random $r \in \mathcal{R}$ (here $\mathcal{M}$ and $\mathcal{R}$ are the message and randomness spaces, respectively), the hash function evaluation algorithm outputs a hash value $h = F_{pk}(m; r) \in \mathcal{H}$ (here $\mathcal{H}$ is the hash string space). On input a secret key sk, a message/randomiser pair $(m_1, r_1) \in M \times R$ and a second message $m_2 \in M$, the trapdoor collision finder algorithm outputs a second randomiser $r_2 = \mathsf{TC}_F(sk, (m_1, r_1), m_2) \in R$ such that (m_1, r_1) and (m_2, r_2) constitute a collision for F_{pk}, i.e. $F_{pk}(m_1; r_1) = F_{pk}(m_2; r_2)$.

There are two desirable security properties for a trapdoor hash function scheme $\mathcal{F} = (\mathsf{KG}_F, F, \mathsf{TC}_F)$. The scheme $\mathcal{F}$ is called (t, ϵ) *strongly collision-resistant* if any efficient attacker A with run-time t has success probability at most ϵ in the following game. A key pair $(sk, pk) = \mathsf{KG}_F(k)$ is generated, and A is given the public key pk. A can run for time t and succeeds if it outputs a collision $(m_1, r_1), (m_2, r_2)$ for F_{pk} satisfying $F_{pk}(m_1, r_1) = F_{pk}(m_2, r_2)$ and $(m_1, r_1) \neq (m_2, r_2)$. The scheme $\mathcal{F}$ has the *random trapdoor collision* property if for each fixed secret key sk and fixed message pair (m_1, m_2), if r_1 is chosen uniformly at random from $\mathcal{R}$, then $r_2 \stackrel{\text{def}}{=} \mathsf{TC}_F(sk, (m_1, r_1), m_2)$ has a uniform probability distribution on $\mathcal{R}$.

3 Converting Weak Unforgeability to Strong Unforgeability

We begin by reviewing the Boneh-Shen-Waters (BSW) transform that applies to the class of *partitioned* signatures. We then explain the problem that arises in trying to apply the BSW transform to an arbitrary signature scheme while preserving the security proof, and how we resolve the problem with our generic transform.

3.1 The Boneh-Shen-Waters Transform for Partitioned Signatures

The BSW transform [2] converts any weakly unforgeable *partitioned* signature into a strongly unforgeable signature. First we recall the definition of partitioned signatures from [2].

Definition 1 (Partitioned Signature). *A signature scheme Σ is called* partitioned *if it satisfies the following two properties:*

1. *The signing algorithm S can be split into two deterministic subalgorithms S_1 and S_2, such that a signature $\sigma = (\sigma_1, \sigma_2)$ on a message m using secret key sk can be computed as follows:*
   - *choose a random $\omega \in \Omega_\Sigma$,*
   - *compute $\sigma_1 = \mathsf{S}_1(sk, m; \omega)$ and $\sigma_2 = \mathsf{S}_2(sk; \omega)$ (note that σ_2 does not depend on m),*
   - *return signature $\sigma = (\sigma_1, \sigma_2)$.*
2. *For each m and σ_2, there exists at most one σ_1 such that $\sigma = (\sigma_1, \sigma_2)$ verifies as a valid signature on message m under public key pk.*

The BSW transform converts a partitioned signature scheme $\Sigma = (\mathsf{KG}, \mathsf{S}, \mathsf{V})$ (where the signing algorithm S is partitioned into subalgorithms S_1 and S_2, and the signing algorithm randomness space is denoted Ω_Σ) into a new signature scheme $\Sigma_{\mathsf{BSW}} = (\mathsf{KG}_{\mathsf{BSW}}, \mathsf{S}_{\mathsf{BSW}}, \mathsf{V}_{\mathsf{BSW}})$. The transform also makes use of a randomised trapdoor hash function scheme $\mathcal{F} = (\mathsf{KG}_F, F, \mathsf{TC}_F)$ (where the randomness space is denoted $\mathcal{R}_\mathcal{F}$). We remark that in [2] the authors present their transform with a specific trapdoor hash construction for $\mathcal{F}$ based on the discrete-log problem, but here we present it more generally. The new signature scheme Σ_{BSW} is defined as follows:

1. $\mathsf{KG}_{\mathsf{BSW}}(k)$. On input security parameter k, run $\mathsf{KG}(k)$ to generate a secret/public key pair (sk, pk) for signature scheme Σ, and run $\mathsf{KG}_F(k)$ to generate secret/public key pair (sk_F, pk_F) for trapdoor hash scheme $\mathcal{F}$. The secret and public keys for the new signature scheme Σ_{BSW} are:
$$sk_{\mathsf{BSW}} = (sk, pk_F) \text{ and } pk_{\mathsf{BSW}} = (pk, pk_F).$$
2. $\mathsf{S}_{\mathsf{BSW}}(sk_{\mathsf{BSW}}, m)$. On input message m and secret key $sk_{\mathsf{BSW}} = (sk, pk_F)$, a signature is generated as follows:

(a) choose random $\omega \in \Omega_{\Sigma}$ and $s \in \mathcal{R}_{\mathcal{F}}$,
(b) compute $\sigma_2 = \mathsf{S}_2(sk;\omega)$,
(c) compute $\bar{m} = F_{pk_F}(m\|\sigma_2;s)$,
(d) compute $\sigma_1 = \mathsf{S}_1(sk,\bar{m};\omega)$ and return signature $\sigma = (\sigma_1,\sigma_2,s)$.

3. $\mathsf{V}_{\mathsf{BSW}}(pk,m,\sigma)$. A signature $\sigma = (\sigma_1,\sigma_2,s)$ on a message m is verified as follows:
   (a) compute $\bar{m} = F_{pk_F}(m\|\sigma_2;s)$,
   (b) return $\mathsf{V}(pk,\bar{m},(\sigma_1,\sigma_2))$.

The security result proven in [2] can be stated as follows (when generalised to the case of an arbitrary trapdoor hash function in place of the composition of a standard collision-resistant hash function and trapdoor hash function in [2]).

Theorem 1 (Boneh–Shen–Waters). *The signature scheme Σ_{BSW} is (t,q,ϵ) strongly existentially unforgeable, assuming the underlying signature scheme Σ is $(t,q,\epsilon/2)$ weakly existentially unforgeable and the randomised trapdoor hash function $\mathcal{F}$ is $(t,\epsilon/2)$ strongly collision-resistant and has the random trapdoor collision property.*

Intuition. The basic idea of the BSW transform (as also explained in [2]) is that the message-independent signature portion σ_2 of a generated signature is protected from modification by appending it to the message m before hashing with F_{pk_F} and signing with the S_1 algorithm. As a consequence, any 'strong unforgeability' attacks which modify σ_2 will lead to either a collision for the hash function F or a 'weak unforgeability' forgery for the underlying signature scheme. However (to set the scene for our generalised construction) we wish to highlight two important issues and how they were addressed in [2]:

(1) *Security Proof Issues:* Following the above intuition, the security proof involves using the strong unforgeability attacker A on Σ_{BSW} to construct attackers $\mathsf{A}_{\mathcal{F}}$ and A_{Σ} against the collision resistance and weak unforgeability of schemes $\mathcal{F}$ and Σ, respectively. But note that to answer A's signing queries:
   (1.1) $\mathsf{A}_{\mathcal{F}}$ needs to be able to simulate signatures of Σ_{BSW} without the trapdoor key sk_F for trapdoor hash scheme $\mathcal{F}$.
   (1.2) A_{Σ} needs to be able to simulate signatures of Σ_{BSW} using the signing algorithm $\mathsf{S}(sk,\cdot)$ as a black box (i.e. without individual access to the internal subalgorithms $\mathsf{S}_1(\cdot,sk)$ and $\mathsf{S}_2(\cdot,sk)$).

(2) *No Protection for σ_1:* Since the σ_1 signature portion is not hashed, it is not protected from modification by the transform.

These issues were addressed in [2] as follows. The issue (1.1) was easily resolved by just using the signing algorithm $\mathsf{S}_{\mathsf{BSW}}$ since the latter does not make use of sk_F. The issue (1.2) was resolved using an alternative signing algorithm which uses the trapdoor key of hash function F_k and the 'sign-then-switch' paradigm [13] to sign using $S(sk,\cdot)$ as a black box (namely, to sign m, one picks a random $s' \in \mathcal{R}_{\mathcal{F}}$ and an arbitrary string σ_2' and signs $\bar{m} = F_{pk_F}(m\|\sigma_2';s')$ to obtain $\sigma = (\sigma_1,\sigma_2) = \mathsf{S}(sk,\bar{m})$, and then 'switch' s' to $s = \mathsf{TC}_F(sk_F,(m\|\sigma_2',s'),(m\|\sigma_2))$ using the trapdoor key sk_F, yielding the signature (σ_1,σ_2,s)). Finally, the issue (2) was resolved vt using Property 2 of partitioned signatures (see Def. 1), which implies that protection of σ_1 is not needed, because one cannot modify σ_1 alone without making the signature invalid.

3.2 Our Generic Transform for Arbitrary Signatures

Intuition. Our goal is to construct a generic transform which can convert any weakly unforgeable signature to a strongly unforgeable one, i.e. we seek a transform which does not rely on the properties of partitioned signatures. Suppose we attempt to modify the BSW transform for this purpose. To address the issue (2) in the previous section, we must protect the *whole* signature from modification. Referring to Fig 1(a), suppose we modify the BSW construction by feeding back the whole signature σ (not just σ_2) into the hash function F_{pk_F} input. The problem is that we obtain a closed loop, where message input $\bar{m}$ of $\mathsf{S}(sk, \cdot)$ depends on the output signature σ. Using the trapdoor key sk_F of the hash function F_{pk_F} and the black box $\mathsf{S}(sk, \cdot)$, we can still produce valid signatures of Σ_{BSW} using the 'sign-then-switch' method outlined in the previous section, but we can no longer produce signatures of Σ_{BSW} without the trapdoor key sk_F (even if we know sk). Therefore, the proof of security for the modified construction collapses due to issue (1.1) discussed in the previous section. Our solution for resolving this issue is to introduce a *second* trapdoor hash function H in this closed loop as shown in Fig. 1(b). This resolves the issue (1.1) by allowing us to use the 'hash-then-switch' method to simulate signatures of Σ_{BSW} using sk_H and sk (without knowing sk_F), or using sk_F and sk (without knowing sk_H), and the last two signing methods produce identically distributed signatures thanks to the 'random trapdoor collision' property of F and H.

Construction. Following the above intuition, we now give a precise description and security proof for our generic transform GBSW. The GBSW transform converts an arbitrary signature scheme $\Sigma = (\mathsf{KG}, \mathsf{S}, \mathsf{V})$ (where the signing algorithm randomness space is denoted Ω_Σ) into a new signature scheme $\Sigma_{\mathsf{GBSW}} = (\mathsf{KG}_{\mathsf{GBSW}}, \mathsf{S}_{\mathsf{GBSW}}, \mathsf{V}_{\mathsf{GBSW}})$. The transform makes use of two randomised trapdoor hash function schemes $\mathcal{F} = (\mathsf{KG}_F, F, \mathsf{TC}_F)$ (with randomness space $\mathcal{R}_\mathcal{F}$) and $\mathcal{H} = (\mathsf{KG}_H, H, \mathsf{TC}_H)$ (with randomness space $\mathcal{R}_\mathcal{H}$). The new signature scheme Σ_{GBSW} is defined as follows:

1. $\mathsf{KG}_{\mathsf{GBSW}}(k)$. On input security parameter k, run $\mathsf{KG}(k)$ to generate a secret/public key pair (sk, pk) for signature scheme Σ, and run $\mathsf{KG}_F(k)$ and $\mathsf{KG}_H(k)$ to generate secret/public key pairs (sk_F, pk_F) and (sk_H, pk_H) for trapdoor hash schemes $\mathcal{F}$ and $\mathcal{H}$, respectively. The secret and public keys for the new signature scheme Σ_{GBSW} are:
$$sk_{\mathsf{GBSW}} = (sk, sk_H, pk_F, pk_H) \text{ and } pk_{\mathsf{GBSW}} = (pk, pk_F, pk_H).$$
2. $\mathsf{S}_{\mathsf{GBSW}}(sk_{\mathsf{GBSW}}, m)$. On input message m and secret key $sk_{\mathsf{GBSW}} = (sk, sk_H, pk_F, pk_H)$, a signature is generated as follows:

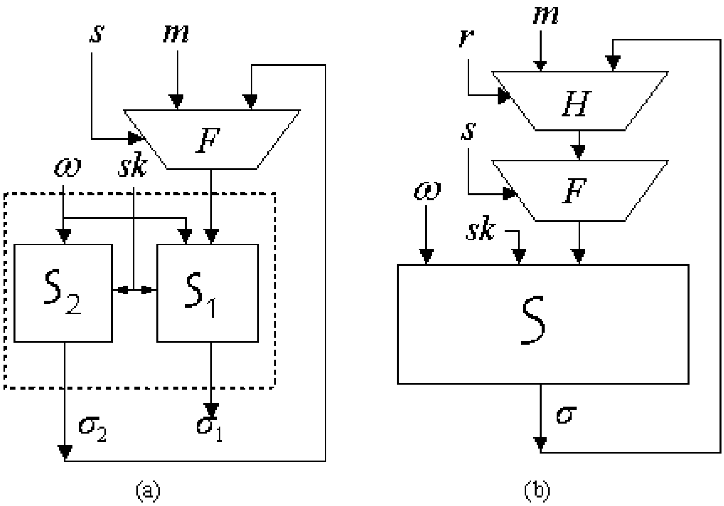

Fig. 1. (a) Boneh–Shen–Waters (BSW) transform for partitioned signature schemes. (b) Our Generalised BSW transform for arbitrary signature schemes.

   (a) choose random elements $\omega \in_R \Omega_\Sigma$, $s \in_R \mathcal{R}_\mathcal{F}$, and $r' \in_R \mathcal{R}_\mathcal{H}$,
   (b) compute $h = H_{pk_H}(m' \| \sigma'; r')$, for some arbitrary fixed strings m' and σ'.
   (c) compute $\bar{m} = F_{pk_F}(h; s)$,
   (d) compute $\sigma = \mathsf{S}(sk, \bar{m}; \omega)$,
   (e) using trapdoor collision finder for H to compute $r = \mathsf{TC}_H(sk_H, (m' \| \sigma', r'), m \| \sigma)$ such that $H_{pk_H}(m \| \sigma; r) = h$ and return signature $\sigma_{\mathsf{GBSW}} = (\sigma, r, s)$.

3. $\mathsf{V}_{\mathsf{GBSW}}(pk_{\mathsf{GBSW}}, m, \sigma_{\mathsf{GBSW}})$. A signature $\sigma_{\mathsf{GBSW}} = (\sigma, r, s)$ on a message m using public key $pk_{\mathsf{GBSW}} = (pk, pk_F, pk_H)$ is verified as follows:
   (a) compute $h = H_{pk_H}(m \| \sigma; r)$,
   (b) compute $\bar{m} = F_{pk_F}(h; s)$,
   (c) return $\mathsf{V}(pk, \bar{m}, \sigma)$.

Remark 1: We implicitly assume of course, that the verification algorithm $\mathsf{V}_{\mathsf{GBSW}}$ *immediately rejects any signature* (σ, r, s) *for which* $r \notin \mathcal{R}_\mathcal{H}$ *or* $s \notin \mathcal{R}_\mathcal{F}$*.*

Remark 2: One can set the schemes $\mathcal{F}$ *and* $\mathcal{H}$ *to be identical – the important requirement is that the key pair instances* (sk_F, pk_F) *and* (sk_H, pk_H) *are generated by two independent runs of the key generation algorithms of* $\mathcal{F}$ *and* $\mathcal{H}$*, respectively.*

The following theorem proves the strong unforgeability of the scheme Σ_{GBSW}, assuming the weak unforgeability of the underlying signature scheme Σ.

Theorem 2. *The signature scheme* Σ_{GBSW} *is* (t, q, ϵ) *strongly existentially unforgeable, assuming the underlying signature scheme* Σ *is* $(t, q, \epsilon/3)$ *weakly existentially unforgeable and the randomised trapdoor hash functions* $\mathcal{F}$ *and* $\mathcal{H}$

are both $(t, \epsilon/3)$ *strongly collision-resistant and both have the random trapdoor collision property.*

Proof. Let A denote a (t, q, ϵ) attacker against the strong unforgeability of Σ_{GBSW}. We show how to construct attackers A_Σ, $\mathsf{A}_\mathcal{F}$ and $\mathsf{A}_\mathcal{H}$ against the weak unforgeability of Σ and strong collision-resistance of $\mathcal{F}$ and $\mathcal{H}$, respectively, such that at least one of A_Σ, $\mathsf{A}_\mathcal{F}$ or $\mathsf{A}_\mathcal{H}$ succeeds with probability at least $\epsilon/3$, and all have run-time t, which establishes the theorem.

We first classify the forgery produced by attacker A. Suppose that A is run on input $pk_{\mathsf{GBSW}} = (pk, pk_F, pk_H) = \mathsf{KG}_{\mathsf{GBSW}}(k)$. For $i = 1, \ldots, q$, let m_i denote the ith sign query of A and (σ_i, r_i, s_i) the answer to that query. Furthermore, let $h_i = H_{pk_H}(m_i \| \sigma_i; r_i)$ and $\bar{m}_i = F_{pk_F}(h_i; s_i)$ be the values computed by the signing algorithm in answering the ith sign query. Finally, let $(m^*, (\sigma^*, r^*, s^*))$ denote the output message/signature forgery of A and define $h^* = H_{pk_H}(m^* \| \sigma^*; r^*)$ and $\bar{m}^* = F_{pk_F}(h^*; s^*)$. Let Succ denote the event that A succeeds to break the strong unforgeability of Σ_{GBSW}. If Succ occurs then it easy to see that at least one of the following subevents must occur:

– subevent Succ_I (Type I forgery): $\bar{m}^* \notin \{\bar{m}_1, \ldots, \bar{m}_q\}$,
– subevent Succ_{II} (Type II forgery): there exists $i^* \in \{1, \ldots, q\}$ such that $\bar{m}^* = \bar{m}_{i^*}$ but $(h^*, s^*) \neq (h_{i^*}, s_{i^*})$,
– subevent Succ_{III} (Type III forgery): there exists $i^* \in \{1, \ldots, q\}$ such that $\bar{m}^* = \bar{m}_{i^*}$ and $(h^*, s^*) = (h_{i^*}, s_{i^*})$ but $(m^*, \sigma^*, r^*) \neq (m_{i^*}, \sigma_{i^*}, r_{i^*})$.

Since event Succ occurs with probability ϵ, it follows that one of the above 3 subevents occur with probability at least $\epsilon/3$. Accordingly, our attackers A_Σ, $\mathsf{A}_\mathcal{F}$ and $\mathsf{A}_\mathcal{H}$ described below will each run A and succeed if subevents Succ_I, Succ_{II} and Succ_{III} occur, respectively. In each of those three runs of A we show that the distribution of A's view is perfectly simulated as in the real attack, so that the subevents Succ_I, Succ_{II} and Succ_{III} occur with the same probability as in the real attack, and hence at least one of attackers A_Σ, A_F and A_H succeeds with probability $\epsilon/3$, as claimed.

Attacker A_Σ. The attacker A_Σ against the weak unforgeability of Σ runs as follows on input public key pk (where $(pk, sk) = \mathsf{GK}(k)$ is a challenge key pair for Σ).

Setup. A_Σ runs $\mathsf{KG}_F(k)$ and $\mathsf{KG}_H(k)$ to generate secret/public key pairs (sk_F, pk_F) and (sk_H, pk_H) for trapdoor hash schemes $\mathcal{F}$ and $\mathcal{H}$, respectively, and runs A with input public key $pk_{\mathsf{GBSW}} = (pk, pk_F, pk_H)$.

Sign Queries. When attacker A makes its ith sign query message m_i, A_Σ responds as follows:

– choose random elements $s_i \in_R \mathcal{R}_\mathcal{F}$, and $r'_i \in_R \mathcal{R}_\mathcal{H}$,
– compute $h_i = H_{pk_H}(m' \| \sigma'; r'_i)$, for some arbitrary fixed strings m' and σ',
– compute $\bar{m}_i = F_{pk_F}(h_i; s_i)$,
– query message $\bar{m}_i$ to sign oracle of A_Σ to obtain answer $\sigma_i = \mathsf{S}(sk, \bar{m}_i; \omega)$ for a random $\omega \in_R \Omega_\Sigma$,

– use trapdoor collision finder for H to compute $r_i = \mathsf{TC}_H(sk_H, (m'\|\sigma', r'_i), (m_i\|\sigma_i))$ such that $H_{pk_H}(m_i\|\sigma_i; r_i) = h_i$ and return signature (σ_i, r_i, s_i) to A.

Output. After A outputs its forgery $(m^*, (\sigma^*, r^*, s^*))$, A_Σ computes $h^* = H_{pk_H}(m^*\|\sigma^*; r^*)$ and $\bar{m}^* = F_{pk_F}(h^*; s^*)$, and outputs $(\bar{m}^*, \sigma^*)$ as its forgery for Σ.

Notice that in the above game A_Σ perfectly simulates the real signing oracle $\mathsf{S}_{\mathsf{GBSW}}$ of A (because A_Σ simply follows the real signing procedure, exploiting the fact that $\mathsf{S}_{\mathsf{GBSW}}$ makes only black box access to the signing oracle $\mathsf{S}(sk, \cdot)$ of Σ, and that A_Σ knows the trapdoor key sk_H for H). Furthermore, if A succeeds and outputs a type I forgery, i.e. if subevent Succ_I occurs, then $(\bar{m}^*, \sigma^*)$ verifies as a valid message/signature pair for Σ and $\bar{m}^* \notin \{\bar{m}_1, \ldots, \bar{m}_q\}$, meaning that A breaks the weak unforgeability of Σ, as required.

Attacker $\mathsf{A}_\mathcal{F}$. The attacker $\mathsf{A}_\mathcal{F}$ against the strong collision-resistance of $\mathcal{F}$ runs as follows on input public key pk_F (where $(pk_F, sk_F) = \mathsf{GK}_F(k)$ is a challenge key pair for $\mathcal{F}$).

Setup. $\mathsf{A}_\mathcal{F}$ runs $\mathsf{KG}(k)$ and $\mathsf{KG}_H(k)$ to generate secret/public key pairs (sk, pk) and (sk_H, pk_H) for schemes Σ and $\mathcal{H}$, respectively, and runs A with input public key $pk_{\mathsf{GBSW}} = (pk, pk_F, pk_H)$.

Sign Queries. When attacker A makes its ith sign query message m_i, $\mathsf{A}_\mathcal{F}$ responds with $(\sigma_i, r_i, s_i) = \mathsf{S}_{\mathsf{GBSW}}(sk_{\mathsf{GBSW}}, m_i)$, where $sk_{\mathsf{GBSW}} = (sk, sk_H, pk_F, pk_H)$. $\mathsf{A}_\mathcal{F}$ also stores $(m_i, \sigma_i, r_i, s_i)$ in a table for later use.

Output. After A outputs its forgery $(m^*, (\sigma^*, r^*, s^*))$, $\mathsf{A}_\mathcal{F}$ computes $h^* = H_{pk_H}(m^*\|\sigma^*; r^*)$ and then $\bar{m}^* = F_{pk_F}(h^*; s^*)$, and searches its table of A's queries for entry $i^* \in \{1, \ldots, q\}$ such that $\bar{m}^* = m_{i^*}$ but $(h^*, s^*) \neq (h_{i^*}, s_{i^*})$. If such entry is found, $\mathsf{A}_\mathcal{F}$ outputs strong collision $(h^*; s^*), (h_{i^*}; s_{i^*})$ for $\mathcal{F}$, else $\mathsf{A}_\mathcal{F}$ fails.

In the above game $\mathsf{A}_\mathcal{F}$ perfectly simulates the real signing oracle $\mathsf{S}_{\mathsf{GBSW}}$ of A (because $\mathsf{A}_\mathcal{F}$ knows both sk and sk_H and follows the real signing algorithm). Furthermore, $\mathsf{A}_\mathcal{F}$ succeeds in breaking the strong collision-resistance of $\mathcal{F}$ if A outputs a type II forgery, i.e. if subevent Succ_{II} occurs (because $(h^*, s^*) \neq (h_{i^*}, s_{i^*})$ but $\bar{m}^* = F_{pk_F}(h^*; s^*) = F_{pk_F}(h_{i^*}; s_{i^*}) = \bar{m}_{i^*}$), as required.

Attacker $\mathsf{A}_\mathcal{H}$. The attacker $\mathsf{A}_\mathcal{H}$ against the strong collision-resistance of $\mathcal{H}$ runs as follows on input public key pk_H (where $(pk_H, sk_H) = \mathsf{GK}_H(k)$ is a challenge key pair for $\mathcal{H}$).

Setup. $\mathsf{A}_\mathcal{H}$ runs $\mathsf{KG}(k)$ and $\mathsf{KG}_F(k)$ to generate secret/public key pairs (sk, pk) and (sk_F, pk_F) for schemes Σ and $\mathcal{F}$, respectively, and runs A with input public key $pk_{\mathsf{GBSW}} = (pk, pk_F, pk_H)$.

Sign Queries. When attacker A makes its ith sign query message m_i, $\mathsf{A}_\mathcal{H}$ responds as follows:

– choose random elements $s'_i \in_R \mathcal{R}_\mathcal{F}$, and $r_i \in_R \mathcal{R}_\mathcal{H}$,
– compute $\bar{m}_i = F_{pk_F}(h'_i; s'_i)$, for some arbitrary fixed string h'_i,

– compute $\sigma_i = \mathsf{S}(sk, \bar{m}_i; \omega)$ for a random $\omega \in_R \Omega_\Sigma$,
– compute $h_i = H_{pk_H}(m_i \| \sigma_i; r_i)$,
– use trapdoor collision finder for $\mathcal{F}$ to compute $s_i = \mathsf{TC}_F(sk_F, (h'_i, s'_i), h_i)$ such that $F_{pk_F}(h_i; s_i) = \bar{m}_i$ and return signature (σ_i, r_i, s_i) to A. $\mathsf{A}_\mathcal{H}$ also stores $(m_i, \sigma_i, r_i, s_i)$ in a table for later use.

Output. After A outputs its forgery $(m^*, (\sigma^*, r^*, s^*))$, $\mathsf{A}_\mathcal{H}$ computes $h^* = H_{pk_H}(m^* \| \sigma^*; r^*)$ and searches its table of A's queries for entry $i^* \in \{1, \ldots, q\}$ such that $h^* = h_{i^*}$ but $(m^*, \sigma^*, r^*) \neq (m_{i^*}, \sigma_{i^*}, r_{i^*})$. If such entry is found, $\mathsf{A}_\mathcal{H}$ outputs strong collision $(m^* \| \sigma^*; r^*), (m_{i^*} \| \sigma_{i^*}; r_{i^*})$ for $\mathcal{H}$, else $\mathsf{A}_\mathcal{H}$ fails.

In the above game, $\mathsf{A}_\mathcal{H}$ succeeds in breaking the strong collision-resistance of $\mathcal{H}$ if A outputs a type III forgery, i.e. if subevent Succ_{III} occurs (because $(m^* \| \sigma^*, r^*) \neq (m_{i^*} \| \sigma_{i^*}, r_{i^*})$ but $h^* = H_{pk_H}(m^* \| \sigma^*; r^*) = H_{pk_H}(m_{i^*} \| \sigma_{i^*}; r_{i^*}) = h_{i^*}$), as required.

It remains to show that in the above game, $\mathsf{A}_\mathcal{H}$ perfectly simulates the real signing oracle $\mathsf{S}_{\mathsf{GBSW}}$ of A. For any fixed message m and fixed signature triple $(\widehat{\sigma}, \widehat{r}, \widehat{s})$, let $P_{real}(\widehat{\sigma}, \widehat{r}, \widehat{s})$ denote the probability that the real signing algorithm $\mathsf{S}_{\mathsf{GBSW}}$ outputs $(\widehat{\sigma}, \widehat{r}, \widehat{s})$ for input message m (over the random choices $r' \in_R \mathcal{R}_H$, $s \in_R \mathcal{R}_F$, $\omega \in_R \Omega_\Sigma$ of the real signing oracle). Similarly, let $P_{sim}(\widehat{\sigma}, \widehat{r}, \widehat{s})$ denote the probability that the sign oracle simulator of $\mathsf{A}_\mathcal{H}$ outputs $(\widehat{\sigma}, \widehat{r}, \widehat{s})$ for input message m (over the random choices $r \in_R \mathcal{R}_H$, $s' \in_R \mathcal{R}_F$, $\omega \in_R \Omega_\Sigma$ of the simulator). Then, defining $\widehat{h} = H_{pk_H}(m \| \widehat{\sigma}; \widehat{r})$ and $\widehat{\bar{m}} = F_{pk_F}(\widehat{h}; \widehat{s})$, we have:

$$\begin{aligned}
&P_{real}(\widehat{\sigma}, \widehat{r}, \widehat{s}) = \\
&\quad \Pr_{r', s, \omega}[(\mathsf{TC}_H(sk_H, (m' \| \sigma', r'), m \| \widehat{\sigma}) = \widehat{r}) \wedge (s = \widehat{s}) \wedge (\mathsf{S}(sk, \widehat{\bar{m}}; \omega) = \widehat{\sigma})] \\
&= \Pr_{r' \in_R \mathcal{R}_H}[\mathsf{TC}_H(sk_H, (m' \| \sigma', r'), m \| \widehat{\sigma}) = \widehat{r}] \cdot \Pr_{s \in_R \mathcal{R}_F}[s = \widehat{s}] \\
&\quad \cdot \Pr_{\omega \in_R \Omega_\Sigma}[\mathsf{S}(sk, \widehat{\bar{m}}; \omega) = \widehat{\sigma}] \\
&= \left(\frac{1}{|\mathcal{R}_\mathcal{H}|}\right) \cdot \left(\frac{1}{|\mathcal{R}_\mathcal{F}|}\right) \cdot \Pr_{\omega \in_R \Omega_\Sigma}[\mathsf{S}(sk, \widehat{\bar{m}}; \omega) = \widehat{\sigma}], \qquad (1)
\end{aligned}$$

where in the second-last row we used the independence of the r', s, ω and in the last row we used the random trapdoor collision property of H and the uniform distribution of s in $\mathcal{R}_F$ chosen by the real signing algorithm.

On the other hand, for the simulated signatures, we have:

$$\begin{aligned}
&P_{sim}(\widehat{\sigma}, \widehat{r}, \widehat{s}) = \\
&\quad \Pr_{r, s', \omega}[(r = \widehat{r}) \wedge (\mathsf{TC}_F(sk_F, (h', s'), \widehat{h}) = \widehat{s}) \wedge (\mathsf{S}(sk, \widehat{\bar{m}}; \omega) = \widehat{\sigma})] \\
&= \Pr_{r \in_R \mathcal{R}_H}[r = \widehat{r}] \cdot \Pr_{s' \in_R \mathcal{R}_F}[\mathsf{TC}_F(sk_F, (h', s'), \widehat{h}) = \widehat{s}] \cdot \Pr_{\omega \in_R \Omega_\Sigma}[\mathsf{S}(sk, \widehat{\bar{m}}; \omega) = \widehat{\sigma}] \\
&= \left(\frac{1}{|\mathcal{R}_\mathcal{H}|}\right) \cdot \left(\frac{1}{|\mathcal{R}_\mathcal{F}|}\right) \cdot \Pr_{\omega \in_R \Omega_\Sigma}[\mathsf{S}(sk, \widehat{\bar{m}}; \omega) = \widehat{\sigma}], \qquad (2)
\end{aligned}$$

where in the second-last row we used the independence of r, s', ω and in the last row we used the random trapdoor collision property of F and the uniform distribution of r in $\mathcal{R}_H$ chosen by the simulator.

Comparing (1) and (2), we conclude that $P_{real}(\widehat{\sigma}, \widehat{r}, \widehat{s}) = P_{sim}(\widehat{\sigma}, \widehat{r}, \widehat{s})$, so $\mathsf{A}_{\mathcal{H}}$ perfectly simulates the real signing algorithm, as required.

It follows that at least one of the attackers A_{Σ}, $\mathsf{A}_{\mathcal{F}}$ and $\mathsf{A}_{\mathcal{H}}$ succeeds with probability at least $\epsilon/3$, completing the proof of the theorem. □

Remark (Non-Adaptive to Adaptive Security). It is known [13,11] that randomised trapdoor hash functions can also be used to generically upgrade non-adaptive chosen message attack security to adaptive chosen message attack security for signature schemes. Suppose we start with a weakly unforgeable signature secure against non-adaptive message attack and we wish to upgrade it to a strongly unforgeable signature secure against adaptive message attack. A generic solution is to apply our weak-to-strong transform above followed by the non-adaptive-to-adaptive transform from [13,11]. However, it is easy to see (by modifying the attacker A_{Σ} in our proof of Theorem 2 using the technique in [13,11]) that our GBSW transform simultaneously also achieves non-adaptive-to-adaptive conversion, so there is no need to apply the second transform. Similarly, like the transforms in [13,11], our GBSW transform also gives an 'on-line/off-line' signature scheme, where the only on-line operation is collision-finding for trapdoor hash scheme $\mathcal{H}$ (for this application, $\mathcal{H}$ would have to be chosen appropriately to have a collision-finding algorithm faster than signing algorithm S). Finally, we remark that the 'dual' of the above observation does *not* hold, namely it is easy to see that the non-adaptive-to-adaptive transforms in [13,11] *do not* upgrade weak unforgeability to strong unforgeability in general.

4 Implementation Issues and Application

4.1 Implementation of the Randomised Trapdoor Hash Function

We discuss some possible provably secure concrete implementations of the randomised trapdoor hash functions used in our transform.

Discrete-Log Based Construction. A well known Discrete-log based strongly collision-resistant randomised trapdoor hash function is the Chaum–van Heijst–Pfitzmann (CHP) function [4], also used in [2]. This construction $\mathcal{H}_{DL}$ works in any group G of prime order q where discrete-log is hard. Let g denote a generator for G and let J denote a collision-resistant hash function from $\{0,1\}^*$ to $\mathbb{Z}_q$. The key generation algorithm $\mathsf{KG}_{\mathcal{H}_{DL}}$ chooses $x \in_R \mathbb{Z}_q$ and outputs public/secret key pair $pk_{\mathcal{H}} = (g, g_1 = g^x)$ and $sk_{\mathcal{H}} = x$. Given randomiser $r \in \mathbb{Z}_q$ and message m, we define its hash value $H_{DL}(m;r) = g^r g_1^{J(m)}$. Given a message/randomiser pair (m, r) and a second message m', the collision-finder algorithm computes a second randomiser $r' = r + (J(m) - J(m'))x \bmod q$ such that $H_{DL}(m;r) = H_{DL}(m';r')$. Any 'strong' collision $(m;r) \neq (m';r')$ for H_{DL} (with $r, r' \in \mathbb{Z}_q$) implies that $m \neq m'$ (because g has order q) and hence $x = (r - r')/(J(m') - J(m)) \bmod q$, revealing the discrete-log of g_1 to base g. Hence $\mathcal{H}_{DL}$ is strongly collision-resistant (with randomiser space $\mathbb{Z}_q$) as long as discrete-log is hard in G and J is collision-resistant, and $\mathcal{H}_{DL}$ also has the random trapdoor collision property.

Factoring-based Construction. The above DL-based construction has a fast collision-finding algorithm but relatively slow hash evaluation algorithm. Some constructions based on a standard factorization problem are given in [13]. A variant of the recent VSH randomised trapdoor hash function [5] can also be used and has the opposite performance tradeoff: a fast evaluation algorithm but relatively slow collision-finding algorithm. Although the randomised trapdoor hash function described in [5] is not *strongly* collision-resistant, we show how to easily modify it to achieve this property. The original construction H_{VSH} in [5] has public key $n = pq$, where p,q are primes congruent to 3 modulo 4. The secret key is (p, q). Let $J : \{0,1\}^* \to \{0,1\}^k$ be a collision-resistant hash function. The randomiser space is $\mathbb{Z}_n^*$. Given message m and randomiser $r \in \mathbb{Z}_n^*$, the hash value is $H_{VSH}(m; r) = (r^2 \prod_{i=1}^{k} p_i^{J(m)_i})^2 \bmod n$, where $J(m)_i$ denotes the ith bit of $J(m) \in \{0,1\}^k$ and p_i denotes the ith prime. Given a message/randomiser pair (m, r) and a second message m', the collision-finder algorithm computes a second randomiser r' such that $H_{VSH}(m; r) = H_{VSH}(m'; r')$ by choosing uniformly at random among the 4 fourth roots of $(r^2 \prod_i p_i^{J(m)_i - J(m')_i})^2 \bmod n$ in $\mathbb{Z}_n^*$. The function H_{VSH} is weakly collision resistant assuming hardness of the factoring-related 'Very Smooth Number Non-Trivial Modular Square-Root' (VSSR) problem, but is not strongly collision-resistant because $(m; (-r \bmod n))$ collides with $(m; r)$ for any m, r. However, the function H'_{VSH} defined in the same way but with randomiser space restricted to $\mathbb{Z}_n^* \bigcap (0, n/2)$ is strongly collision-resistant under the VSSR assumption. This follows from the fact that any quadratic residue in $\mathbb{Z}_n^*$ has two of its square roots less than $n/2$ and two above (the negatives modulo n of each of the first two square-roots). The two square-roots r,r' below $n/2$ are congruent modulo one of the prime factors of n but not modulo the other prime factor, so finding both r and r' is as hard as factoring n (since $\gcd(r' - r, n)$ gives either p or q). The random trapdoor collisions property also is preserved by this modification (note that the modified collision-finder algorithm chooses r' uniformly at random among the two fourth roots of $(r^2 \prod_i p_i^{J(m)_i - J(m')_i})^2 \bmod n$ in $\mathbb{Z}_n^* \bigcap (0, n/2)$).

4.2 Application to Strengthen the Standard Digital Signature Algorithm (DSA)

The Digital Signature Standard [12] (DSA) is an example of a randomised signature scheme which probably does *not* fall within the class of partitioned signature schemes, as noted in [2]. In this scheme, the public key is $(g, y = g^x \bmod p)$, where p is prime and $g \in \mathbb{Z}_p^*$ is an element of prime order q, and $x \in \mathbb{Z}_q$ is the secret key. The signature on message m using randomiser $r \in \mathbb{Z}_q$ is (σ_1, σ_2), where $\sigma_2 = (g^r \bmod p) \bmod q$ and $\sigma_1 = r^{-1}(SHA(m) + x\sigma_2) \bmod q$ (here $SHA : \{0,1\}^* \to \mathbb{Z}_q$ is the SHA-1 hash function). To verify signature (σ_1, σ_2) on message m under public key (g, y), one checks whether $((g^{SHA(m)} y^{\sigma_2})^{1/\sigma_1} \bmod p) \bmod q$ equals σ_2.

Although DSA clearly satisfies Property (1) of partitioned signatures, it probably does not satisfy Property (2). The reason is that given a valid signature (σ_1, σ_2) on a message m, the number of σ'_1 values such that (σ'_1, σ_2) verifies as a valid signature on m is the number of elements in the group G of order q generated by g which are congruent to σ_2 mod q. As σ_1' runs through all $q-1$ values of $\mathbb{Z}_q$ except σ_1, we heuristically expect the values of $((g^{SHA(m)}y^{\sigma_2})^{1/\sigma_1} \bmod p) \bmod q$ to behave like $q-1$ independent uniformly random elements in $\mathbb{Z}_q$. This heuristic suggests that with "high probability" of about $1-(1-1/q)^{q-1} \approx 0.63$, we expect there exists at least one other $\sigma_1' \neq \sigma_1$ such that (σ_1', σ_2) is also a valid signature on m. Although we do not know how to efficiently find such 'strong forgeries' for DSA, the fact that DSA is not partitioned means that the BSW transform does not provably guarantee the strong unforgeability of DSA, even assuming that DSA is weakly unforgeable.

Applying our generalised transform to DSA with two CHP [4] randomised trapdoor hash functions for $\mathcal{F}$ and $\mathcal{H}$ based on the hardness of discrete-log in the group G used by DSA, we can construct a strengthened DSA signature which is provably strongly unforgeable, assuming only the weak unforgeability of DSA (which immediately implies the hardness of discrete-log in G and hence the strong collision-resistance of $\mathcal{F}$ and $\mathcal{H}$). The resulting concrete system, called SDSA, is as follows.

1. $\mathsf{KG}_{\mathsf{SDSA}}(k)$. On input security parameter k:
   (a) run DSA key generation on input k to generate a DSA key pair $sk_{\mathsf{DSA}} = (p, q, g, x)$ and $pk_{\mathsf{DSA}} = (p, q, g, y)$, where p is prime, q is a divisor of $p-1$, $g \in \mathbb{Z}_p^*$ is an element of order $q > 2^{159}$, x is uniformly random in $\mathbb{Z}_q$ and $y = g^x \bmod p$,
   (b) choose uniformly random $x_H \in \mathbb{Z}_q$ and compute $v = g^{x_H} \bmod p$,
   (c) choose uniformly random $x_F \in \mathbb{Z}_q$ and compute $u = g^{x_F} \bmod p$,
   (d) the secret and public keys for signature scheme SDSA are:
$$sk_{\mathsf{SDSA}} = (p, q, g, x, v, u, x_H) \text{ and } pk_{\mathsf{SDSA}} = (p, q, g, y, v, u).$$
2. $\mathsf{S}_{\mathsf{SDSA}}(sk_{\mathsf{SDSA}}, m)$. On input message m and secret key $sk_{\mathsf{SDSA}} = (p, q, g, x, x_H)$, a signature is generated as follows:
   (a) compute $h = g^{\eta'} v^{SHA(0)} \bmod p$, for uniformly random $\eta' \in \mathbb{Z}_q$ and fixed bit string 0 (e.g. an all zero byte),
   (b) compute $\bar{m} = g^s u^{SHA(h)} \bmod p$ for uniformly random $s \in \mathbb{Z}_q$.
   (c) compute DSA signature (σ_1, σ_2) on "message" $\bar{m}$, where $\sigma_2 = (g^r \bmod p) \bmod q$ for uniformly random $r \in \mathbb{Z}_q$ and $\sigma_1 = r^{-1}(SHA(\bar{m}) + x \cdot \sigma_2) \bmod q$,
   (d) compute $\eta = \eta' + (SHA(0) - SHA(m \| \sigma_1 \| \sigma_2)) \cdot x_H \bmod q$,
   (e) return signature $\sigma_{\mathsf{SDSA}} = (\sigma_1, \sigma_2, \eta, s)$.
3. $\mathsf{V}_{\mathsf{SDSA}}(pk_{\mathsf{SDSA}}, m, \sigma_{\mathsf{SDSA}})$. A signature $\sigma_{\mathsf{SDSA}} = (\sigma_1, \sigma_2, \eta, s)$ on a message m is verified as follows:
   (a) compute $h = g^{\eta} v^{SHA(m \| \sigma_1 \| \sigma_2)} \bmod p$,
   (b) compute $\bar{m} = g^s u^{SHA(h)} \bmod p$,
   (c) accept only if DSA signature (σ_1, σ_2) verifies on "message" $\bar{m}$, namely accept only if $\sigma_2 = ((g^{SHA(\bar{m})} y^{\sigma_2})^{1/\sigma_1} \bmod p) \bmod q$ holds.

We have:

Corollary 1. *The signature scheme* SDSA *is* (t, q, ϵ) *strongly existentially unforgeable assuming that the* DSA *signature is* $(t, \max(q, 1), \epsilon/6)$ *weakly existentially unforgeable.*

Proof. Applying Theorem 2 to the GBSW transform applied to the DSA signature with two CHP trapdoor hash functions $\mathcal{F}$ and $\mathcal{H}$, we can convert any (t, q, ϵ) attacker against the strong unforgeability of SDSA into a $(t, q, \epsilon/3)$ attacker against the weak unforgeability of DSA or a $(t, \epsilon/3)$ attacker against the strong collision-resistance of $\mathcal{F}$ or $\mathcal{H}$ respectively. In turn, any $(t, \epsilon/3)$ attacker against collision-resistance of $\mathcal{F}$ (or $\mathcal{H}$) can be converted into either a $(t, \epsilon/6)$ attacker against the discrete-log problem in the group generated by g, or a $(t, \epsilon/6)$ attacker against the collision-resistance of SHA. Finally, any $(t, \epsilon/6)$ discrete-log attacker can be easily converted into a $(t, 0, \epsilon/6)$ attacker against weak unforgeability of DSA, while any $(t, \epsilon/6)$ attacker against collision-resistance of SHA can be easily converted into a $(t, 1, \epsilon/6)$ attacker against the weak unforgeability of DSA. So in any case, we can construct a $(t, \max(q, 1), \epsilon/6)$ attacker against weak unforgeability of DSA, which gives the claimed result. □

The SDSA scheme requires an extra computation of two products of two exponentiations each in both verification and signature generation over the DSA scheme, the public key contains two additional elements of $\mathbb{Z}_p$ and the signature contains two additional elements of $\mathbb{Z}_q$. A feature of SDSA which may be of use in practice is that it uses the key generation, signature generation and verification algorithms of DSA as subroutines; hence existing implementations of these subroutines can be used without modification to build SDSA implementations.

5 Conclusion

We presented a modification of the Boneh–Shen–Waters transform to strengthen *arbitrary* weakly unforgeable signatures into strongly unforgeable signatures, and presented applications to the Digital Signature Standard (DSA) with suggested concrete implementations of the randomised trapdoor hash functions needed by our transform.

Finally, we have recently learnt (by private communication with I. Teranishi) that, independently and in parallel with our work, Teranishi, Oyama and Ogata [14] propose a 'weak to strong' unforgeability transform which uses a similar idea to our transform, but is less general in its implementation. In particular, the standard model transform in [14] assumes the hardness of the discrete-log problem, whereas our transform works with any randomised trapdoor hash function (for example, our transform can be used with the efficient factoring-based trapdoor hash function from [5]). On the other hand, the discrete-log based transform in [14] has a more efficient verification algorithm compared to our general transform applied using the discrete-log based trapdoor hash function from [4]. A more efficient transform assuming the random-oracle model along with the discrete-log assumption is also described in [14].

Acknowledgements. The authors would like to thank Duncan Wong for interesting discussions and the anonymous referees for their useful comments. This work was supported by Australian Research Council Discovery Grants DP0663452, DP0451484 and DP0665035.

References

1. D. Boneh and X. Boyen. Short Signatures without Random Oracles. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73, Berlin, 2004. Springer-Verlag.
2. D. Boneh, E. Shen, and B. Waters. Strongly Unforgeable Signatures Based on Computational Diffie-Hellman. In *PKC 2006*, volume 3958 of *LNCS*, pages 229–240, Berlin, 2006. Springer-Verlag.
3. R. Canetti, S. Halevi, and J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. In *Eurocrypt 2004*, volume 3027 of *LNCS*, pages 207–222, Berlin, 2004. Springer-Verlag.
4. D. Chaum, E. van Heijst, and B. Pfitzmann. Cryptographically Strong Undeniable Signatures, Unconditionally Secure for the Signer. In *CRYPTO '91*, volume 576 of *LNCS*, pages 470–484, Berlin, 1991. Springer-Verlag.
5. S. Contini, A.K. Lenstra, and R. Steinfeld. VSH, an Efficient and Provable Collision-Resistant Hash Function. In *Eurocrypt 2006*, volume 4004 of *LNCS*, pages 165–182, Berlin, 2006. Springer-Verlag.
6. R. Cramer and V. Shoup. Signature Schemes Based on the Strong RSA Assumption. *ACM Transactions on Information and System Security (ACM TISSEC)*, 3:161–185, 2000.
7. R. Gennaro, S. Halevi, and T. Rabin. Secure Hash-and-Sign Signatures Without the Random Oracle. In *EUROCRYPT '99*, volume 1592 of *LNCS*, pages 123–139, Berlin, 1999. Springer-Verlag.
8. O. Goldreich. *Foundations of Cryptography, Volume II.* Cambridge University Press, Cambridge, 2004.
9. J. Katz and M. Yung. Scalable Protocols for Authenticated Group Key Exchange. In *CRYPTO 2003*, volume 2729 of *LNCS*, pages 110–125, Berlin, 2003. Springer-Verlag.
10. H. Krawczyk and T. Rabin. Chameleon Signatures. In *NDSS 2000*, 2000. Available at `http://www.isoc.org/isoc/conferences/ndss/2000/proceedings/`.
11. K. Kurosawa and S. Heng. The Power of Identification Schemes. In *PKC 2006*, volume 3958 of *LNCS*, pages 364–377, Berlin, 2006. Springer-Verlag.
12. National Institute of Standards and Technology (NIST). *Digital Signature Standard (DSS). Federal Information Processing Standards Publication 186-2*, January 2000.
13. A. Shamir and Y. Tauman. Improved Online/Offline Signature Schemes. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 355–367, Berlin, 2001. Springer-Verlag.
14. I. Teranishi, T. Oyama, and W. Ogata. General Conversion for Obtaining Strongly Existentially Unforgeable Signatures. In *INDOCRYPT 2006.* To Appear.