Strongly Unforgeable Signatures Based on Computational Diffie-Hellman

Dan Boneh[1]*, Emily Shen[1], and Brent Waters[2]

[1] Computer Science Department, Stanford University, Stanford, CA
{dabo,emily}@cs.stanford.edu
[2] SRI International, Palo Alto, CA
bwaters@csl.sri.com

Abstract. A signature system is said to be strongly unforgeable if the signature is existentially unforgeable and, given signatures on some message m, the adversary cannot produce a new signature on m. Strongly unforgeable signatures are used for constructing chosen-ciphertext secure systems and group signatures. Current efficient constructions in the standard model (i.e. without random oracles) depend on relatively strong assumptions such as Strong-RSA or Strong-Diffie-Hellman. We construct an efficient strongly unforgeable signature system based on the standard Computational Diffie-Hellman problem in bilinear groups.

1 Introduction

A digital signature system is said to be secure if it is existentially unforgeable under a chosen-message attack [20]. Roughly speaking, this means that an adversary who is given a signature for a few messages of his choice should not be able to produce a signature for a new message. For a variety of applications, however, a stronger security property called *strong unforgeability* is needed [1]. Strong unforgeability ensures the adversary cannot even produce a new signature for a previously signed message. In other words, suppose an adversary obtains a message-signature pair (m, σ) along with other message-signature pairs of his choice. The signature system is strongly unforgeable if the adversary cannot produce a new signature $\hat{\sigma}$ for m. We give a precise definition in the next section.

Strongly unforgeable signatures have a number of applications. They are useful for building chosen-ciphertext secure encryption systems [14, 8] as well as group signatures [2, 5]. To see the relation to chosen-ciphertext security recall that chosen-ciphertext secure systems in the standard model often incorporate a (one-time) signature in the ciphertext. This signature is generated by the encryptor and is a signature on the ciphertext. Strong unforgeability is needed to ensure that the adversary cannot somehow modify the signature in the challenge ciphertext and come up with an alternate valid signature on the same ciphertext. This alternate signature would give the adversary a valid ciphertext that is different from the challenge ciphertext. The adversary could then issue

* Supported by NSF and the Packard Foundation.

a decryption query for this new ciphertext and break the system. Consequently, a signature system that is existentially unforgeable but not strongly unforgeable would result in an insecure encryption system. A similar issue comes up in several group signature constructions.

Several existing signature systems are strongly unforgeable. In the random oracle model, constructions based on the full domain hash [3, 9, 6] and other methods [3, 18, 26] are strongly unforgeable.

Without random oracles, several constructions can be shown to be strongly unforgeable; however, they typically depend on relatively strong assumptions:

- Gennaro, Halevi, and Rabin [17] and Cramer and Shoup [11] construct strongly unforgeable signatures based on the Strong-RSA assumption.
- Boneh and Boyen [4] construct a strongly unforgeable signature based on the Strong-Diffie-Hellman assumption.
- A Verifiable Unpredictable Function (VUF) gives a signature system where each message has a unique signature. Such signatures are clearly strongly unforgeable. VUFs were defined by Micali, Rabin, and Vadhan [25] where they give a proof-of-concept construction based on the (large exponent) RSA assumption. A different VUF was proposed by Lysyanskaya [23] using the Many-Diffie-Hellman assumption (a.k.a. the n-party Diffie-Hellman assumption) in bilinear groups. This construction was extended by Dodis [13] to obtain a Verifiable Random Function under a much stronger assumption.
- Tree-based signatures [20, 19, 27, 15, 10, 7] can be proven secure without random oracles and based on standard assumptions. However, they generally tend to be less efficient than signatures presented in this paper.

Our contribution. In this paper we construct a strongly unforgeable signature system (without random oracles) based on the *standard Computational Diffie-Hellman* (CDH) problem in bilinear groups. The system is simple, efficient, and produces signatures that are only 2 group elements plus a short string.

Currently, the only (efficient) signature that is known to be existentially unforgeable based on CDH (in the standard model) is due to Waters [28]. This signature, however, is not strongly unforgeable — given a signature on some message m it is easy to derive many other signatures on the same message. Nevertheless, we use the Waters signature scheme as our starting point. We show how to strengthen the signature to obtain a strongly unforgeable signature based on the standard CDH. We actually do a little more — we provide a general transformation that converts any unforgeable signature of a certain type into a strongly unforgeable signature. We then apply this transformation to the Waters signature to obtain a strongly unforgeable signature based on CDH.

2 Preliminaries

Before presenting our construction we briefly review the security definitions, a few facts about bilinear maps, and our complexity assumptions.

2.1 Strong Existential Unforgeability

A signature system consists of three algorithms: *KeyGen*, *Sign*, and *Verify*. Strong existential unforgeability under an adaptive chosen-message attack is defined using the following game:

Setup. The challenger runs *KeyGen*. It gives the adversary the resulting public key PK and keeps the private key SK to itself.

Signature Queries. The adversary issues signature queries $m_1, \ldots, m_q$. To each query m_i the challenger responds by running *Sign* to generate a signature σ_i of m_i and sending σ_i to the adversary. These queries may be asked adaptively so that each query m_i may depend on the replies to $m_1, \ldots, m_{i-1}$.

Output. Finally the adversary outputs a pair (m, σ). The adversary wins if σ is a valid signature of m according to *Verify* and (m, σ) is not among the pairs (m_i, σ_i) generated during the query phase.

We define the advantage of an adversary $\mathcal{A}$ in attacking the signature scheme as the probability that $\mathcal{A}$ wins the above game, taken over the random bits of the challenger and the adversary.

Definition 1. *A signature scheme is (t, q, ϵ)-strongly existentially unforgeable under an adaptive chosen-message attack if no t-time adversary $\mathcal{A}$ making at most q signature queries has advantage at least ϵ in the above game.*

2.2 Existential Unforgeability

We will also use the traditional security property of (weak) existential unforgeability under an adaptive chosen-message attack [20]. It is defined using the following game.

Setup and Signature Queries. Same as in the strong unforgeability game.

Output. The adversary outputs a pair (m, σ). The adversary wins if σ is a valid signature of m according to *Verify* and m is not among the messages m_i queried during the query phase.

We define the advantage of an adversary $\mathcal{A}$ in weakly attacking a signature scheme as the probability that $\mathcal{A}$ wins the above game, taken over the random bits of the challenger and the adversary.

Definition 2. *A signature scheme is (t, q, ϵ)-existentially unforgeable under an adaptive chosen-message attack if no t-time adversary $\mathcal{A}$ making at most q signature queries has advantage at least ϵ in the above game.*

2.3 Bilinear Groups

We use the following notation:

1. $\mathbb{G}$ and $\mathbb{G}_1$ are two (multiplicative) cyclic groups of prime order p;
2. g is a generator of $\mathbb{G}$;

3. e is a computable map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ with the following properties:
   - Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
   - Non-degenerate: $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group [21] if the group operation in $\mathbb{G}$ is efficiently computable and there exists a group $\mathbb{G}_1$ and an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ as above.

2.4 Computational Diffie-Hellman (CDH) Assumption

The computational Diffie-Hellman problem in a cyclic group $\mathbb{G}$ of order p is defined as follows. Given $g, g^a, g^b \in \mathbb{G}$, output $g^{ab} \in \mathbb{G}$. We say that algorithm $\mathcal{A}$ has advantage ϵ in solving CDH in $\mathbb{G}$ if

$$\Pr[\mathcal{A}(g, g^a, g^b) = g^{ab}] \geq \epsilon \ ,$$

where the probability is over the random choice of generator $g \in \mathbb{G}$, the random choice of $a, b \in \mathbb{Z}_p$, and the random bits of $\mathcal{A}$.

Similarly, we say that algorithm $\mathcal{A}$ has advantage ϵ in solving discrete log in $\mathbb{G}$ if

$$\Pr[\mathcal{A}(g, g^a) = a] \geq \epsilon \ ,$$

where the probability is over the random choice of generator $g \in \mathbb{G}$, the random choice of $a \in \mathbb{Z}_p$, and the random bits of $\mathcal{A}$.

Definition 3. *The (t, ϵ)-CDH assumption holds in $\mathbb{G}$ if no t-time adversary has advantage at least ϵ in solving CDH in $\mathbb{G}$. Similarly, the (t, ϵ)-Dlog assumption holds in $\mathbb{G}$ if no t-time adversary has advantage at least ϵ in solving discrete log.*

2.5 Collision-Resistant Hashing

Let $\mathcal{H} = \{H_k\}$ be a keyed hash family of functions $H_k : \{0,1\}^* \to \{0,1\}^n$ indexed by $k \in \mathcal{K}$. We say that algorithm $\mathcal{A}$ has advantage ϵ in breaking the collision-resistance of $\mathcal{H}$ if

$$\Pr[\mathcal{A}(k) = (m_0, m_1) \ \ : \ \ m_0 \neq m_1, \ \ H_k(m_0) = H_k(m_1)] \geq \epsilon \ ,$$

where the probability is over the random choice of $k \in \mathcal{K}$ and the random bits of $\mathcal{A}$.

Definition 4. *A hash family $\mathcal{H}$ is (t, ϵ)-collision-resistant if no t-time adversary has advantage at least ϵ in breaking the collision-resistance of $\mathcal{H}$.*

Our construction makes use of collision-resistant hashing. We note, however, that collision-resistant hashing can be easily built based on the CDH assumption [12]. Therefore, in theory, assuming the existence of collision-resistant functions does not strengthen the complexity assumption we are making. In practice, of course, one would use a standard hash function such as SHA-256 and assume that it is collision-resistant.

3 From Weak Unforgeability to Strong Unforgeability

Our goal is to construct a strongly unforgeable signature based on CDH. We begin by presenting a general transformation that converts any *partitioned* unforgeable signature (defined below) into a strongly unforgeable signature. In the next section we apply this transformation to the Waters signature.

Definition 5. *We say that a signature system is* partitioned *if it satisfies two properties:*

- **Property 1:** *The signing algorithm can be broken into two deterministic algorithms F_1 and F_2 so that a signature on a message m using secret key SK is computed as follows:*
  1. *Select a random r in $\mathcal{R}$.*
  2. *Set $\sigma_1 \leftarrow F_1(m, r, SK)$ and $\sigma_2 \leftarrow F_2(r, SK)$.*
  3. *Output the signature $\sigma \leftarrow (\sigma_1, \sigma_2)$.*
- **Property 2:** *Given m and σ_2 there is at most one σ_1 so that (σ_1, σ_2) verifies as a valid signature on m under PK.*

In other words, a signature is partitioned if half the signature, namely σ_2, does not depend on m. Furthermore, given m and σ_2 the signature is fully determined. Many standard discrete-log-based signature systems in the literature can be partitioned. For example, for DSS [24] using x to denote the secret key, the functions F_1, F_2 are:

$$F_1(m, r, x) = r^{-1}\big(m + xF_2(r, x)\big) \bmod q$$
$$F_2(r, x) = (g^r \bmod p) \bmod q$$

We note, however, that property 2 may not hold for DSS.

Next, we present our transformation. Let $\mathbb{G}$ be a group of prime order p and let $\mathcal{H} = \{H_k\}$ be a collision-resistant hash family of functions $H_k : \{0,1\}^* \rightarrow \{0,1\}^n$ indexed by $k \in \mathcal{K}$. We assume $p \geq 2^n$ so that hash outputs can be viewed as elements of $\mathbb{Z}_p$. Furtheremore, we assume that each element of $\mathbb{Z}_p$ has a unique encoding, say as an integer in $[0, p)$. In describing the system we use the notation $x\|y$ to denote the marked concatenation of the two strings x and y.

Let $\Sigma = (KeyGen, Sign, Verify)$ be a partitioned signature where the signing algorithm is partitioned using functions F_1 and F_2. Suppose the randomness for signature generation is picked from some set $\mathcal{R}$. We build a new strongly unforgeable signature system $\Sigma_{\text{new}} = (KeyGen_{\text{new}}, Sign_{\text{new}}, Verify_{\text{new}})$ as follows:

***KeyGen*$_{\text{new}}$:** To generate the public key, select random generators $g, h \in \mathbb{G}$ and a random hash key $k \in \mathcal{K}$. Next, run *KeyGen* to obtain a secret key SK and public key PK. The public and secret keys for the new system are:

$$\text{PK}' = (\text{PK}, g, h, k) \qquad \text{and} \qquad \text{SK}' = (\text{SK})$$

***Sign*$_{\text{new}}$(SK, M):** A signature on a message $M \in \{0,1\}^\ell$ is generated as follows.

1. Select a random exponent $s \in \mathbb{Z}_p$ and a random $r \in \mathcal{R}$.

2. Set $\sigma_2 \leftarrow F_2(r, \mathrm{SK})$.
3. Compute $t \leftarrow H_k(M \| \sigma_2) \in \{0,1\}^n$ and view t as an element of $\mathbb{Z}_p$.
4. Compute $m \leftarrow g^t h^s \in \mathbb{G}$.
5. Compute $\sigma_1 \leftarrow F_1(m, r, \mathrm{SK})$ and output the signature $\sigma \leftarrow (\sigma_1, \sigma_2, s)$.

***Verify*$_{\mathbf{new}}$(PK, M, σ):** A signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ on a message M is verified as follows:

1. Compute $\tilde{t} \leftarrow H_k(M \| \sigma_2)$ and view $\tilde{t}$ as an element of $\mathbb{Z}_p$.
2. Compute $\tilde{m} \leftarrow g^{\tilde{t}} h^{\sigma_3}$.
3. Output *Verify*$\big(\mathrm{PK},\ \tilde{m},\ (\sigma_1, \sigma_2)\big)$.

The basic idea. To give some intuition for signature generation, note that in Step 4 we derive a new message m that is then signed by the underlying signature system in Step 5. This m is derived from the original message M and from σ_2. The σ_2 is derived from the randomness r. Hence, in effect, the signer is signing both the message M and the secret randomness r that is used to create the signature. The adversary, as a result, cannot "re-randomize" a given signature without invalidating the signature. This may suggest that the resulting signature scheme is strongly unforgeable. Unfortunately, in creating this circularity — making the message m being signed depend on the randomness r — we break the proof of security for the underlying signature. Because of Steps 3 and 4 we can no longer prove that the system is secure.

To repair the damage we introduce an additional hashing step (Step 4) where we hash again using a chameleon hash [22]. The extra randomness s of the chameleon hash lets us break the circularity in the proof of security. This lets us repair the proof and prove strong unforgeability based strictly on the weak unforgeability of the underlying system. In particular, the randomness of the chameleon hash is crucial for responding to signature queries from a Type III adversary in the proof of security below.

In summary, the high level structure of the signing algorithm is as follows: (1) first, hash $M \| \sigma_2$ using a chameleon hash to obtain a new message m, (2) then, use the underlying signature to sign m with randomness r, (3) finally, output the resulting signature along with the randomness s of the chameleon hash. The proof of security in the next subsection shows that the resulting signature is strongly unforgeable.

3.1 Security

Let $\Sigma = (\mathit{KeyGen}, \mathit{Sign}, \mathit{Verify})$ be a partitioned signature scheme and let $\Sigma_{\mathrm{new}} = (\mathit{KeyGen}_{\mathrm{new}}, \mathit{Sign}_{\mathrm{new}}, \mathit{Verify}_{\mathrm{new}})$ be the signature system resulting from the transformation described above. The following theorem proves strong unforgeability of Σ_{new}.

Theorem 1. *The signature scheme Σ_{new} is (t, q, ϵ)-strongly existentially unforgeable assuming the underlying signature scheme Σ is $(t, q, \epsilon/3)$-existentially unforgeable, the $(t, \epsilon/3)$-Dlog assumption holds in $\mathbb{G}$, and $\mathcal{H}$ is $(t, \epsilon/3)$-collision-resistant.*

Proof. Suppose $\mathcal{A}$ is a forger that (t, q, ϵ)-breaks strong unforgeability of Σ_{new}. Forger $\mathcal{A}$ is first given a public key (PK, g, h, k).

Forger $\mathcal{A}$ asks for signatures on messages $M_1, \ldots, M_q$ and is given signatures $\sigma_i = (\sigma_{i,1}, \sigma_{i,2}, s_i)$ for $i = 1, \ldots, q$ on these messages. Let $t_i = H_k(M_i \| \sigma_{i,2})$ and $m_i = g^{t_i} h^{s_i}$ for $i = 1, \ldots, q$. Let $\big(\hat{M}, \hat{\sigma} = (\hat{\sigma}_1, \hat{\sigma}_2, \hat{s})\big)$ be the forgery produced by $\mathcal{A}$, let $\hat{t} = H_k(\hat{M} \| \hat{\sigma}_2)$, and let $\hat{m} = g^{\hat{t}} h^{\hat{s}}$. We distinguish among three types of forgeries:

Type I A forgery where $\hat{m} = m_i$ and $\hat{t} = t_i$ for some $i \in \{1, \ldots, q\}$.
Type II A forgery where $\hat{m} = m_i$ and $\hat{t} \neq t_i$ for some $i \in \{1, \ldots, q\}$.
Type III Any other forgery ($\hat{m} \neq m_i$ for all $i \in \{1, \ldots, q\}$).

A successful forger must output a forgery of Type I, Type II, or Type III. We show that a Type I forgery can be used to break the collision-resistance of $\mathcal{H}$, a Type II forgery can be used to solve discrete log in $\mathbb{G}$, and a Type III forgery can be used to break existential unforgeability of the underlying signature scheme Σ. Our simulator can flip a coin at the beginning of the simulation to guess which type of forgery the adversary will produce and set up the simulation appropriately. In all three cases the simulation is perfect. We start by describing how to use a Type III forgery which is the more interesting case.

Type III forger: Suppose algorithm $\mathcal{A}$ is a Type III forger that (t, q, ϵ)-breaks strong unforgeability of Σ_{new}. We construct a simulator $\mathcal{B}$ that (t, q, ϵ)-breaks existential unforgeability of Σ. $\mathcal{B}$ is given a public key PK. $\mathcal{B}$'s goal is to produce a pair (m, σ) where σ is a valid signature on m and m is not among $\mathcal{B}$'s chosen message queries. $\mathcal{B}$ runs $\mathcal{A}$ as follows.

Setup. Algorithm $\mathcal{B}$ generates the public key PK' as follows.
1. Select a random generator $g \in \mathbb{G}$.
2. Select a random exponent $a \in {\mathbb{Z}_p}^*$ and set $h \leftarrow g^a$.
3. Select a random hash key $k \in \mathcal{K}$.
4. Provide the public key $\text{PK}' \leftarrow (\text{PK}, g, h, k)$ to $\mathcal{A}$.

Signature Queries. Algorithm $\mathcal{A}$ issues up to q signature queries. Algorithm $\mathcal{B}$ responds to a query on a message M as follows.
1. Select a random exponent $w \in \mathbb{Z}_p$ and set $m \leftarrow g^w$.
2. Ask $\mathcal{B}$'s challenger for a signature on message m. Obtain a signature (σ_1, σ_2) on m.
3. Compute $t \leftarrow H_k(M \| \sigma_2)$.
4. Set $s \leftarrow (w - t)/a$.
5. Return $\sigma \leftarrow (\sigma_1, \sigma_2, s)$ to $\mathcal{A}$.

Indeed, $m = g^w = g^{as+t} = g^t h^s$ and s is uniform in $\mathbb{Z}_p$ as required. Hence, σ is a valid signature on M.

Output. Finally, algorithm $\mathcal{A}$ outputs a forgery $\big(\hat{M}, (\hat{\sigma}_1, \hat{\sigma}_2, \hat{s})\big)$. Algorithm $\mathcal{B}$ produces a weak forgery on the underlying scheme as follows.
1. Compute $\hat{t} \leftarrow H_k(\hat{M} \| \hat{\sigma}_2)$.
2. Compute $\hat{m} \leftarrow g^{\hat{t}} h^{\hat{s}}$.

3. Output $\big(\hat{m},\ (\hat{\sigma}_1, \hat{\sigma}_2)\big)$.

Note that $\hat{m} \notin \{m_1, \ldots, m_q\}$ because if $\hat{m} = m_i$ for some $i \in \{1, \ldots, q\}$ then, either $\hat{t} = t_i$ (a Type I forgery) or $\hat{t} \neq t_i$ (a Type II forgery). Therefore $\mathcal{B}$ produces a forgery on some new message $\hat{m}$ for the underlying scheme whenever $\mathcal{A}$ produces a Type III forgery, as required.

Type I forger: Next we show how to use a Type I forger. Suppose $\mathcal{A}$ is a Type I forger that (t, q, ϵ)-breaks strong unforgeability of Σ_{new}. We construct an algorithm $\mathcal{B}$ that (t, ϵ)-breaks the collision-resistance of $\mathcal{H}$. Algorithm $\mathcal{B}$ is given a random key $k' \in \mathcal{K}$. $\mathcal{B}$'s goal is to output a pair of messages (m_1, m_2) such that $m_1 \neq m_2$ and $H_{k'}(m_1) = H_{k'}(m_2)$. $\mathcal{B}$ runs $\mathcal{A}$ as follows.

Setup. Algorithm $\mathcal{B}$ sets $k \leftarrow k'$ and generates the remaining elements of the public key and the private key according to *KeyGen*$_{\text{new}}$. $\mathcal{B}$ gives $\mathcal{A}$ the resulting public key $\text{PK}' = (\text{PK}, g, h, k)$ and keeps the secret key SK'.

Signature Queries. $\mathcal{A}$ issues up to q signature queries. $\mathcal{B}$ responds to a query on a message M_i by running *Sign*$_{\text{new}}(\text{SK}', M_i)$ and returning the signature σ_i to $\mathcal{A}$.

Output. $\mathcal{A}$ outputs a forgery $\big(\hat{M},\ \hat{\sigma} = (\hat{\sigma}_1, \hat{\sigma}_2, \hat{s})\big)$ such that

$$(\hat{M}, \hat{\sigma}) \notin \big\{(M_1, \sigma_1), \ldots, (M_q, \sigma_q)\big\} \quad \text{and} \quad \hat{m} = m_i \quad \text{and} \quad \hat{t} = t_i$$

for some $i \in \{1, \ldots, q\}$. More precisely, $\hat{t} = t_i$ means that $H_k(\hat{M} \| \hat{\sigma}_2) = H_k(M_i \| \sigma_{i,2})$. Similarly, $\hat{m} = m_i$ means that $g^{\hat{t}} h^{\hat{s}} = g^{t_i} h^{s_i}$.

Then $\mathcal{B}$ outputs the pair $(\hat{M} \| \hat{\sigma}_2,\ M_i \| \sigma_{i,2})$ as a collision on H_k.

We show that algorithm $\mathcal{B}$ succeeds in producing an H_k-collision whenever $\mathcal{A}$ produces a Type I forgery. Since $H_k(\hat{M} \| \hat{\sigma}_2) = H_k(M_i \| \sigma_{i,2})$ we only need to show that $\hat{M} \| \hat{\sigma}_2 \neq M_i \| \sigma_{i,2}$.

Suppose towards a contradiction that $\hat{M} = M_i$ and $\hat{\sigma}_2 = \sigma_{i,2}$. Since $\hat{t} = t_i$ and $\hat{m} = m_i$ we know that $\hat{s} = s_i$. (We require that any exponent $s \in \mathbb{Z}_p$ has a unique encoding.) Furthermore, since $\hat{\sigma}_2 = \sigma_{i,2}$ and $\hat{m} = m_i$, the second property of partitioned signatures implies that $\hat{\sigma}_1 = \sigma_{i,1}$. Hence, we have just shown that $\hat{M} = M_i$ and $\hat{\sigma} = \sigma_i$ which contradicts the fact that $(\hat{M}, \hat{\sigma})$ is a strong existential forgery. Therefore, $\hat{M} \| \hat{\sigma}_2 \neq M_i \| \sigma_{i,2}$, implying that whenever $\mathcal{A}$ produces a Type I forgery, $\mathcal{B}$ produces an H_k-collision.

Type II forger: Finally, we show how to use a Type II forger. Suppose $\mathcal{A}$ is a Type II forger that (t, q, ϵ)-breaks strong unforgeability of Σ_{new}. We construct an algorithm $\mathcal{B}$ that (t, ϵ)-solves discrete log in $\mathbb{G}$. Algorithm $\mathcal{B}$ is given a random pair (g', h') and its goal is to output a such that $h' = (g')^a$. $\mathcal{B}$ runs $\mathcal{A}$ as follows.

Setup. Algorithm $\mathcal{B}$ sets $g \leftarrow g'$, $h \leftarrow h'$, and generates the remaining elements of the public key and the private key according to *KeyGen*$_{\text{new}}$. $\mathcal{B}$ gives $\mathcal{A}$ the resulting public key $\text{PK}' = (\text{PK}, g, h, k)$ and keeps the private key SK'.

Signature Queries. $\mathcal{A}$ issues up to q signature queries. $\mathcal{B}$ responds to a query on a message M_i by running $Sign_{\text{new}}(\text{SK}', M_i)$ and returning the signature σ_i to $\mathcal{A}$.

Forgery. $\mathcal{A}$ outputs a forgery $\big(\hat{M}, \hat{\sigma} = (\hat{\sigma}_1, \hat{\sigma}_2, \hat{s})\big)$ such that $\hat{m} = m_i$ and $\hat{t} \neq t_i$ for some $i \in \{1, \ldots, q\}$. Then we have $g^{\hat{t}} h^{\hat{s}} = g^{t_i} h^{s_i}$, which can be written as $g^{\hat{t}} (g^a)^{\hat{s}} = g^{t_i} (g^a)^{s_i}$. Then $\mathcal{B}$ computes $a = (t_i - \hat{t})/(\hat{s} - s_i) \in \mathbb{Z}_p$ and outputs a in response to its discrete log challenge. Note that $\hat{s} - s_i \neq 0$ since $\hat{s} = s_i$ and $g^{\hat{t}} h^{\hat{s}} = g^{t_i} h^{s_i}$ imply $\hat{t} = t_i$.

Algorithm $\mathcal{B}$ succeeds in solving its discrete log challenge whenever $\mathcal{A}$ produces a Type II forgery, as required.

In summary, we showed how to use all three forgery types to break existential unforgeability of the underlying signature scheme, collision-resistance of $\mathcal{H}$, or discrete log. This completes the proof of Theorem 1. $\square$

4 A Concrete Construction: Strong Unforgeability from CDH

We now apply Theorem 1 to the Waters signature which is based on CDH without random oracles. It is straightforward to verify that the Waters signature is partitioned. The functions F_1 and F_2 are:

$$F_1(m, r, \text{SK}) = \text{SK} \cdot \big(u' \prod_{i=1}^{n} u_i^{m_i}\big)^r \quad \in \mathbb{G}$$

$$F_2(r, \text{SK}) = g^r \quad \in \mathbb{G}$$

where $u', u_1, \ldots, u_n \in \mathbb{G}$ are part of the public key and $m = m_1 \ldots m_n \in \{0,1\}^n$. The second property of partitioned signatures holds since given m and $\sigma_2 = F_2(r, \text{SK})$ there is only one σ_1 for which the verification equation will hold. Note that we are assuming that each element $g \in \mathbb{G}$ has a unique encoding (otherwise an attacker can invalidate property 2 by simply changing the encoding of a group element).

Thus, applying Theorem 1 to the Waters signature system we obtain a strongly unforgeable scheme based on CDH without random oracles. The resulting system is as follows. Let $\mathbb{G}$ be a bilinear group of prime order p and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ denote the bilinear map and g be the corresponding generator. Let $\mathcal{H} = \{H_k\}$ be a collision-resistant hash family of functions $H_k : \{0,1\}^* \to \{0,1\}^n$ indexed by $k \in \mathcal{K}$. We assume $p \geq 2^n$ so that hash outputs can be viewed as elements of $\mathbb{Z}_p$.

KeyGen. To generate the public key, select a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$ and set $g_1 = g^\alpha$. Next, select random $g_2, h \in \mathbb{G}$. Select random $u', u_1, \ldots, u_n \in \mathbb{G}$ and let $U = (u_1, \ldots, u_n)$. Finally, select a random hash key $k \in \mathcal{K}$. The public and secret keys are:

$$\text{PK} = (g, g_1, g_2, h, u', U, k) \qquad \text{and} \qquad \text{SK} = (g_2^\alpha)$$

Note that the secret key is a single group element, but the public key contains $n+5$ group elements where n is the hash output size.

Sign. A signature on a message $M \in \{0,1\}^\ell$ is generated as follows.

1. Select random exponents $r, s \in \mathbb{Z}_p$.
2. Set $\sigma_2 \leftarrow g^r \in \mathbb{G}$.
3. Compute $t \leftarrow H_k(m\|\sigma_2) \in \{0,1\}^n$ and view t as an element of $\mathbb{Z}_p$.
4. Compute $m \leftarrow H_k(g^t h^s)$ and write m as $m_1 \dots m_n \in \{0,1\}^n$.
5. Compute $\sigma_1 \leftarrow g_2^\alpha \cdot (u' \prod_{i=1}^n u_i^{m_i})^r$ and output the signature (σ_1, σ_2, s).

Verify. A signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ on a message M is verified as follows:

1. Compute $\tilde{t} \leftarrow H_k(M\|\sigma_2)$ and view $\tilde{t}$ as an element of $\mathbb{Z}_p$.
2. Compute $\tilde{m} \leftarrow H_k(g^{\tilde{t}} h^{\sigma_3})$ and write $\tilde{m}$ as $\tilde{m}_1 \dots \tilde{m}_n \in \{0,1\}^n$.
3. Check that
$$e(\sigma_1, g) \stackrel{?}{=} e(\sigma_2, u' \prod_{i=1}^{n} u_i^{\tilde{m}_i}) \cdot e(g_1, g_2) \ .$$
Accept if this holds and reject otherwise.

Corollary 1. *The signature system above is (t, q, ϵ)-strongly existentially unforgeable assuming the $(t, \epsilon/24(n+1)q)$-CDH assumption holds in $\mathbb{G}$, and $\mathcal{H}$ is $(t, \epsilon/3)$-collision-resistant.*

Proof. The Waters system is known to be (t, q, ϵ)-existentially unforgeable assuming $(t, \epsilon/8(n+1)q)$-CDH holds in $\mathbb{G}$. It follows that the system is $(t, q, \epsilon/3)$-existentially unforgeable assuming $(t, \epsilon/24(n+1)q)$-CDH holds in $\mathbb{G}$. When $(t, \epsilon/3)$-CDH holds in $\mathbb{G}$ then $(t, \epsilon/3)$-Dlog must also hold in $\mathbb{G}$. Hence, since the system is partitioned, all the requirements of Theorem 1 are satisfied. Consequently, the signature system above is strongly unforgeable.

Efficiency. Our signature system is only slightly worse than the Waters signature system in terms of performance. The signing operation in our scheme takes four exponentiations and $n/2+2$ group operations in $\mathbb{G}$ on average. The verification algorithm consists of two pairings, two exponentiations, $n/2+1$ group operations in $\mathbb{G}$ and one group operation in $\mathbb{G}_1$ on average. Like the Waters signature scheme public keys are approximately n group elements. However, we note that the values $u', U = (u_1, \dots, u_n)$ can actually come from a common reference string and be shared by all users in a system. If this is the case each user's public key can be short.

5 Conclusions

We constructed a strongly unforgeable signature system based on the standard Computational Diffie-Hellman problem in bilinear groups. The signature is efficient and contains only two group elements (plus a short random string). The public key size is proportional to the output size of the hash function used. We presented the construction in two steps. First, we showed a general mechanism for transforming any partitioned (weakly) unforgeable system into a strongly unforgeable system. We then applied this transformation to a specific system.

Surprisingly, our signature system does not seem to naturally extend to give an efficient threshold signature [16]. In fact, the only known efficient strongly unforgeable threshold signatures (in the standard model) appear to be the unique signatures of Lysyanskaya [23] and Dodis [13]. Thresholdizing these signatures, however, requires multiple rounds of interaction with the signing servers and the resulting signatures are somewhat long. We leave as an open problem the question of constructing a threshold unforgeable signature based on a standard assumption.

References

1. J. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In L. R. Knudsen, editor, *Proceedings of Eurocrypt 2002*, volume 2332 of *LNCS*, pages 83–107. Springer-Verlag, 2002.
2. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *Proceedings of Crypto 2000*, volume 1880 of *LNCS*, pages 255–70. Springer-Verlag, Aug. 2000.
3. M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In U. Maurer, editor, *Proceedings of Eurocrypt '96*, volume 1070 of *LNCS*, pages 399–416. Springer-Verlag, 1996.
4. D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 56–73. Springer-Verlag, 2004. Full version at: http://eprint.iacr.org/2004/171.
5. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 41–55. Springer-Verlag, 2004.
6. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *J. of Cryptology*, 17(4):297–319, 2004. Early version in Asiacrypt '01.
7. D. Boneh, I. Mironov, and V. Shoup. A secure signature scheme from bilinear maps. In M. Joye, editor, *Proceedings of RSA-CT '03*, volume 2612 of *LNCS*, pages 98–110. Springer-Verlag, 2003.
8. R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, LNCS, pages 207–222. Springer-Verlag, 2004. http://eprint.iacr.org/2003/182/.
9. J.-S. Coron. On the Exact Security of Full Domain Hash. In M. Bellare, editor, *Proceedings of Crypto 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer-Verlag, 2000.
10. R. Cramer and I. Damgård. New generation of secure and practical rsa-based signatures. In N. Koblitz, editor, *Proceedings of Crypto '96*, volume 1109 of *LNCS*, pages 173–185. Springer-Verlag, 1996.
11. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *ACM TISSEC*, 3(3):161–185, 2000. Extended abstract in Proc. 6th ACM CCS, 1999.
12. I. Damgård. Collision free hash functions and public key signature schemes. In D. Chaum and W. L. Price, editors, *Proceedings of Eurocrypt '87*, volume 304 of *LNCS*, pages 203–216. Springer-Verlag, 1987.

13. Y. Dodis. Efficient construction of (distributed) verifiable random functions. In Y. Desmedt, editor, *Workshop on Public Key Cryptography (PKC)*, volume 2567 of *LNCS*, pages 1–17. Springer-Verlag, 2003.
14. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM J. of Computing*, 30(2):391–437, 2000.
15. C. Dwork and M. Naor. An efficient existentially unforgeable signature scheme and its applications. *J. of Cryptology*, 11(2):187–208, 1998. Early version in Crypto '94.
16. P. Gemmel. An introduction to threshold cryptography. *RSA CryptoBytes*, 2(3):7–12, 1997.
17. R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In J. Stern, editor, *Proceedings of Eurocrypt 1999*, volume 1592 of *LNCS*, pages 123–139. Springer-Verlag, 1999.
18. E.-J. Goh and S. Jarecki. A signature scheme as secure as the Diffie-Hellman problem. In E. Biham, editor, *Proceedings of Eurocrypt 2003*, volume 2656 of *LNCS*, pages 401–415. Springer-Verlag, 2003.
19. O. Goldreich. Two remarks concerning the goldwasser-micali-rivest signature scheme. In A. M. Odlyzko, editor, *Proceedings of Crypto'86*, volume 263 of *LNCS*, pages 104–110. Springer-Verlag, 1987.
20. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281–308, 1988.
21. A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Proceedings of ANTS IV*, volume 1838 of *LNCS*, pages 385–94. Springer-Verlag, 2000.
22. H. Krawczyk and T. Rabin. Chameleon signatures. In *Proceedings of NDSS 2000*. Internet Society, 2000. http://eprint.iacr.org/1998/010/.
23. A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In M. Yung, editor, *Proceedings of Crypto 2002*, volume 2442 of *LNCS*, pages 597–612. Springer-Verlag, 2002.
24. A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
25. S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In *Proceedings of the 40th Annual Symposium on the Foundations of Computer Science*, pages 120–130, New York, NY, October 1999. IEEE.
26. S. Micali and L. Reyzin. Improving the exact security of digital signature schemes. *J. of Cryptology*, 15(1):1–18, 2002.
27. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of STOC'89*, pages 33–43, 1989.
28. B. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *Proceedings of Eurocrypt 2005*, volume 3494 of *LNCS*, pages 114–127. Springer-Verlag, 2005.