To appear in Journal of Cryptology, Springer-Verlag, 2007.

Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups *

Dan Boneh † Xavier Boyen ‡

December 26, 2006

Abstract

We describe a short signature scheme that is strongly existentially unforgeable under an adaptive chosen message attack in the standard security model. Our construction works in groups equipped with an efficient bilinear map, or, more generally, an algorithm for the Decision Diffie-Hellman problem. The security of our scheme depends on a new intractability assumption we call *Strong Diffie-Hellman* (SDH), by analogy to the Strong RSA assumption with which it shares many properties. Signature generation in our system is fast and the resulting signatures are as short as DSA signatures for comparable security. We give a tight reduction proving that our scheme is secure in any group in which the SDH assumption holds, without relying on the random oracle model.

1 Introduction

Short signatures have always been desirable in applications with constrained space or bandwidth, such as those with printed material or a human typist in the loop. Although several approaches have been proposed over the years, it has remained a challenge to construct short signatures with generic provable security guarantees, especially without the use of random oracles. Towards this goal, Boneh, Lynn, and Shacham (BLS) [11] proposed a short digital signature scheme where signatures are about half the size of DSA signatures with the same level of security. The BLS scheme is shown to be existentially unforgeable under an adaptive chosen message attack in the random oracle model; its security is based on the Computational Diffie-Hellman (CDH) assumption on certain elliptic curves equipped with a bilinear map that effectively renders the Decisional Diffie-Hellman (DDH) problem easy.

In this paper we describe a signature scheme in a similar setting, but whose security does not require random oracles. In addition, our signatures can be made as short as BLS signatures, and are much more efficient. We prove security of our scheme using a complexity assumption we call the Strong Diffie-Hellman assumption, or SDH for short. Roughly speaking, for some parameter q, the SDH assumption in a group $\mathbb{G}$ of prime order p states that the following problem is intractable:

Given $g, g^x, g^{(x^2)}, \ldots, g^{(x^q)} \in \mathbb{G}$ as input, output a pair $(c,\ g^{1/(x+c)})$ where $c \in \mathbb{Z}_p$.

---

*An extended abstract entitled "Short Signatures Without Random Oracles" [9] appears in Eurocrypt 2004.

†Stanford University; supported by NSF and the Packard Foundation — *dabo@cs.stanford.edu*

‡Voltage, Palo Alto — *xb@boyen.org*

Asymptotically, we say that the SDH assumption holds in some infinite family of groups, if the q-SDH problem above is hard for any function q that is polynomially bounded in the security parameter, where the group size p grows exponentially with the security parameter. Using this assumption, we construct a signature scheme that is existentially unforgeable under an adaptive chosen message attack, without relying on the random oracle methodology.[1]

Currently, the most practical signature schemes secure without random oracles, such as [23, 18], are based on the Strong RSA assumption. It states that, given an RSA modulus N and $s \in \mathbb{Z}_N^\times$, it is difficult to construct a non-trivial pair $(c, s^{1/c})$ where $c \in \mathbb{Z}$. Roughly speaking, what makes Strong RSA useful for constructing secure signature schemes is the following property: given a problem instance (N, s) it is possible to construct a new instance (N, s') with some number q of known solutions $(c_i, (s')^{1/c_i})$, such that learning any additional solution $(c, (s')^{1/c})$ makes it possible to solve the original problem instance. This property provides a way to prove security against an adaptive chosen message attack.

In this paper, we exploit a similar property of the q-SDH problem as a foundation for our constructions, albeit in a vastly different algebraic setting. Hence, the SDH assumption may be viewed as a discrete logarithm analogue of the Strong RSA assumption. We believe that the properties of SDH make it a useful tool for constructing cryptographic systems, and we expect to see many other systems based on it in the future. Some examples can already be found in [10, 19]. We remark that Mitsunari, Sakai, and Kasahara [35] previously used a weaker assumption related to SDH to construct a traitor tracing scheme (see also [47] for an analysis of the system). Complexity assumptions and their properties are discussed in Section 3.

Our short signatures necessitate a special kind of group in which the Diffie-Hellman problem can be *decided* efficiently. The only well-known examples of this are (sub)groups of points on certain algebraic curves equipped with a bilinear map such as the Weil pairing. For concreteness, we shall thus describe our constructions in terms of bilinear pairings, although it should be noted that any efficient method for testing the Diffie-Hellman property can be substituted in all our constructions. We provide some background on secure signatures and bilinear maps in Section 2.

We construct our main secure signature scheme from bilinear pairings, and prove its security against existential forgery under adaptive chosen message attack in the standard model. Our full signatures are as short as DSA signatures, but are provably secure in the absence of random oracles. They support efficient off-line/on-line signing, and a limited form of message recovery which makes it possible to further reduce the total length of the signed message by embedding a few bits of the message in the signature itself. We present all these constructions in Section 4.

We further show that with random oracles, the SDH assumption gives even shorter signatures. We do so using a generalization of the Katz-Wang construction [29]. The resulting scheme produces signatures whose length is about the same as BLS signatures, but which can be verified twice as fast, and created five to ten times faster. A related system using random oracles has been independently described by Zhang et al. [48]. These random oracle signatures are discussed in Section 5.

To gain some confidence in the SDH assumption in the presence of bilinear maps, we provide a lower bound on the computational complexity of solving the q-SDH problem in a generic group model [45]. This shows that, even with the help of pairings, no generic attack on the SDH assumption is possible. In other words, any system provably reducible to SDH can only be defeated in the mathematical sense by considering the structure of the particular groups in which it is implemented. Any attack on SDH in the elliptic curve bilinear groups used in practice would have to

[1] We mention that the SDH assumption studied in this article is slightly weaker and more general than the original assumption proposed in the Eurocrypt 2004 extended abstract [9].

expose heretofore unknown structural properties of these curves. Our lower bound shows that the time to break q-SDH in a generic group of size p is at least $\Omega(\sqrt{p/q})$ provided that $q < O(\sqrt[3]{p})$. Brown and Gallant [12] and Cheon [14] presented a matching generic upper bound: when q is approximately $\sqrt[3]{p}$, they gave generic group algorithms that can solve the q-SDH problem in time $\tilde{O}(\sqrt{p/q})$ under certain conditions. We prove our generic lower bound in Section 6.

We refer to [11] for applications of short signatures. We merely mention that short digital signatures are needed in environments with stringent bandwidth constraints, such as bar-coded digital signatures on postage stamps [37, 41]. In fact, our short signatures with limited message recovery in the standard model are particularly well-suited for these types of applications where the message itself is very compact, such as a serial number. Other short signature systems, proposed by Patarin et al. [39, 17], are based on the Hidden Field Equation (HFE) problem.

2 Preliminaries

Before presenting our results we briefly review two notions of security for signature schemes and give a succinct refresher on groups equipped with a bilinear map.

2.1 Secure Signature Schemes

A signature scheme is made up of three algorithms, *KeyGen*, *Sign*, and *Verify*, for generating keys, signing, and verifying signatures, respectively. For a fixed security parameter, these algorithms work as follows:

KeyGen outputs a random key pair (PK, SK);

Sign takes a private key SK and a message M from some set $\mathcal{M}$, and returns a signature σ;

Verify takes a public key PK and a signed message (M, σ), and returns `valid` or `invalid`.

The signature scheme is said to be correct, or consistent, if it satisfies the following property: $\forall M \in \mathcal{M},\ \forall (PK, SK) \leftarrow \mathit{KeyGen}(),\ \forall \sigma \leftarrow \mathit{Sign}(SK, M)\ :\ \Pr[\mathit{Verify}(PK, M, \sigma) = \texttt{valid}] = 1$.

Strong Existential Unforgeability

The standard notion of security for a signature scheme is called existential unforgeability under an adaptive chosen message attack [25]. We consider a slightly stronger notion of security, called strong existential unforgeability [1], which is defined using the following game between a challenger and an adversary $\mathcal{A}$:

Setup: The challenger runs algorithm *KeyGen* to obtain a public key PK and a private key SK. The adversary is given PK.

Queries: Proceeding adaptively, the adversary requests signatures on at most q_S messages of its choice $M_1, \ldots, M_{q_s} \in \{0,1\}^\star$, under PK. The challenger responds to each query with a signature $\sigma_i \leftarrow \mathit{Sign}(SK, M_i)$.

Output: Eventually, the adversary outputs a pair (M, σ) and wins the game if:
1. (M, σ) is not any of $(M_1, \sigma_1), \ldots, (M_{q_s}, \sigma_{q_s})$; and
2. $\mathit{Verify}(PK, M, \sigma) = \texttt{valid}$.

We define $\mathsf{Sig\,Adv}_{\mathcal{A}}$ to be the probability that the adversary $\mathcal{A}$ wins in the above game, taken over the coin tosses made by $\mathcal{A}$ and the challenger.

Definition 1. A forger $\mathcal{A}$ is said to (t, q_S, ϵ)-break a signature scheme if $\mathcal{A}$ runs in time at most t, $\mathcal{A}$ makes at most q_S signature queries, and $\mathsf{Sig\,Adv}_{\mathcal{A}}$ is at least ϵ. A signature scheme is (t, q_S, ϵ)-strongly existentially unforgeable under an adaptive chosen message attack if there exists no forger that (t, q_S, ϵ)-breaks it.

When discussing security in the random oracle model, we add a fourth parameter q_H to denote an upper bound on the number of queries that the adversary $\mathcal{A}$ makes to the random oracle.

We note that the definition above captures a stronger version of existential unforgeability than the standard one, as it requires that the adversary cannot even generate a new signature on a previously signed message. This property is required for some applications [1, 42, 13]. All our signature schemes satisfy this stronger security notion.

Weak Chosen Message Attacks

We will also use a less stringent notion of security which we call existential unforgeability under a weak chosen message attack (this is sometimes called a generic chosen message attack). In a weak chosen message attack, we require that the adversary submit all signature queries before seeing the public key. This notion is defined using the following game between a challenger and an adversary $\mathcal{A}$:

Query: The adversary sends to the challenger a list of q_S messages $M_1, \ldots, M_{q_s} \in \{0,1\}^\star$.

Response: The challenger runs algorithm *KeyGen* to generate a public key PK and a private key SK. Next, the challenger generates signatures $\sigma_i \leftarrow \mathit{Sign}(SK, M_i)$ for $i = 1, \ldots, q_S$. The challenger then gives to the adversary the public key PK and the q_S signatures $\sigma_1, \ldots, \sigma_{q_s}$.

Output: The adversary outputs a pair (M, σ) and wins the game if:

1. (M, σ) is not any of $(M_1, \sigma_1), \ldots, (M_{q_s}, \sigma_{q_s})$; and
2. $\mathit{Verify}(PK, M, \sigma) = \texttt{valid}$.

We define $\mathsf{W\text{-}Sig\,Adv}_{\mathcal{A}}$ to be the probability that the adversary $\mathcal{A}$ wins in the above game, taken over the coin tosses made by $\mathcal{A}$ and the challenger.

Definition 2. A forger $\mathcal{A}$ (t, q_S, ϵ)-weakly breaks a signature scheme if $\mathcal{A}$ runs in time at most t, $\mathcal{A}$ makes at most q_S signature queries, and $\mathsf{W\text{-}Sig\,Adv}_{\mathcal{A}}$ is at least ϵ. A signature scheme is (t, q_S, ϵ)-existentially unforgeable under a weak chosen message attack if there exists no forger that (t, q_S, ϵ)-weakly breaks it.

2.2 Bilinear Groups

Signature verification in our scheme requires a decision procedure for the Diffie-Hellman problem, which can be implemented using a bilinear map. We briefly review the necessary facts about bilinear maps and groups, in the notation of [11]:

- $(\mathbb{G}_1, *)$, $(\mathbb{G}_2, *)$, and $(\mathbb{G}_T, *)$ are three cyclic groups of prime order p;
- g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$;
- e is a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, i.e., a map satisfying the following properties:
  Bilinearity: $\forall u \in \mathbb{G}_1$, $\forall v \in \mathbb{G}_2$, $\forall a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$;
  Non-degeneracy: $e(g_1, g_2) \neq 1$ and is thus a generator of $\mathbb{G}_T$.

All group operations and the bilinear map must be efficiently computable. Formally, one defines a bilinear group generation algorithm $\mathcal{G}$ that takes as input a security parameter $\lambda \in \mathbb{Z}^+$ and outputs the description of groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We then require the existence of probabilistic polynomial time algorithms (in λ) for computing the group operation in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and the bilinear map e.

For simplicity one can set $\mathbb{G}_1 = \mathbb{G}_2$. However, as in [11], we consider the more general case where $\mathbb{G}_1 \neq \mathbb{G}_2$; this allows us to take advantage of certain families of algebraic curves in order to obtain the shortest possible signatures. Specifically, when $\mathbb{G}_1$ and $\mathbb{G}_2$ are realized as (sub)groups of points on certain elliptic curves over a finite field $\mathbb{F}$, the coordinates of the points in $\mathbb{G}_1$ may live in the ground field $\mathbb{F}$, whereas those of the points in $\mathbb{G}_2$ may live in an extension of $\mathbb{F}$. Since the elements in $\mathbb{G}_1$ may have a shorter representation than those of $\mathbb{G}_2$, it is useful for us to distinguish the two.

Several papers that use a bilinear map where $\mathbb{G}_1 \neq \mathbb{G}_2$ also assume the existence of a homomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$. Such a homomorphism, known as the trace map, often exists when $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of an elliptic curve. However there are several cases, called type 3 groups [22], where this homomorphism is degenerate. In this paper we will not need a homomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ or its inverse. This enables our signatures to operate with any bilinear group construction currently known.

The bilinear map e can be realized on Abelian varieties using variants of the Weil or Tate pairing [40, 21]. On algebraic curves in particular, these are very efficiently computable using an algorithm proposed in 1986 by Miller [34]. In the usual instantiations, the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of order p of the groups, defined by a curve E over a finite field $\mathbb{F}$, of points with coordinates in $\mathbb{F}$ or in an extension of $\mathbb{F}$. The target group $\mathbb{G}_T$ is then the multiplicative subgroup of order p in a large enough extension of $\mathbb{F}$ to contain all p-th roots of unity. Recent advances in the area of bilinear pairings have produced new and more efficient variants of the Tate pairing, known as "Eta" [2] and "Ate" [27]. All these pairings can be efficiently computed using variants of the Miller algorithm on the appropriate curves. We note that in all known examples, $\mathbb{G}_T$ is always different from $\mathbb{G}_1$ and $\mathbb{G}_2$.

We define the general notion of bilinear group as follows.

Definition 3. We say that $(\mathbb{G}_1, \mathbb{G}_2)$ are a bilinear group pair if there exists a group $\mathbb{G}_T$ and a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, such that the group order $p = |\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T|$ is prime, and the pairing e and the group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are all efficiently computable.

Joux and Nguyen [28] showed that an efficiently computable bilinear map provides an algorithm for solving the Decision Diffie-Hellman (DDH) problem when $\mathbb{G}_1 = \mathbb{G}_2$. More generally, when $\mathbb{G}_1 \neq \mathbb{G}_2$, we can build from the bilinear map a predicate for a "cross-group" DDH relation between pairs of elements of $\mathbb{G}_1$ and $\mathbb{G}_2$. The cross-group DDH problem is as follows:

> Given $g_1, g_1^a \in \mathbb{G}_1$ and $g_2, g_2^b \in \mathbb{G}_2$, decide whether $a = b \pmod{p}$.

This problem is easily solved by using the pairing to test whether $e(g_1^a, g_2) = e(g_1, g_2^b)$ in $\mathbb{G}_T$. Pairings are merely used for that purpose in all of our constructions. For the sake of concreteness, our exposition shall make explicit use of the bilinear map, although we emphasize that all our results can be restated using a generic predicate (given as an oracle) for cross-group DDH in $(\mathbb{G}_1, \mathbb{G}_2)$, without the need for an explicit pairing.

3 The Strong Diffie-Hellman Assumption

We now give a formal definition of the SDH computational complexity assumption. The definition we propose presently is weaker and cleaner than in the Eurocrypt 2004 original paper [9], as it no longer requires the existence of an efficiently computable group homomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$.

We first define the concrete q-SDH problem, stated with respect to a specific bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ and a parameter q. We then give an asymptotic definition of the SDH assumption.

3.1 Concrete Formulation of SDH

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order p, respectively generated by g_1 and g_2. In the bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$, the q-SDH problem is stated as follows:

> Given as input a $(q+3)$-tuple of elements $(g_1, g_1^x, g_1^{(x^2)}, \ldots, g_1^{(x^q)}, g_2, g_2^x) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$,
> output a pair $(c,\, g_1^{1/(x+c)}) \in \mathbb{Z}_p \times \mathbb{G}_1$ for a freely chosen value $c \in \mathbb{Z}_p \setminus \{-x\}$.

Note that when $\mathbb{G}_1 = \mathbb{G}_2$ the pair (g_2, g_2^x) is redundant since in that case (g_2, g_2^x) can be generated by raising (g_1, g_1^x) to a random power. An algorithm $\mathcal{A}$ solves the q-SDH problem in the bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ with advantage ϵ if

$$\mathsf{SDH\,Adv}_{q,\mathcal{A}} := \Pr\left[\mathcal{A}\big(g_1, g_1^x, \ldots, g_1^{(x^q)}, g_2, g_2^x\big) = \Big(c,\, g_1^{\frac{1}{x+c}}\Big)\right] \geq \epsilon \tag{1}$$

where the probability is over the random choice of generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, the random choice of $x \in \mathbb{Z}_p^\times$, and the random bits consumed by $\mathcal{A}$.

Definition 4. We say that the (q, t, ϵ)-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no t-time algorithm has advantage at least ϵ in solving the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$.

The concrete security results we shall obtain are based on Definition 4 and will depend on four parameters: p, q, t, ϵ. Occasionally we drop the t and ϵ and assume that the groups are understood from context, and speak of the q-SDH problem and the SDH assumption.

3.2 Asymptotic Formulation of SDH

To formulate the SDH assumption asymptotically, we need a bilinear group generation algorithm $\mathcal{G}$.

Definition 5. A bilinear group generator $\mathcal{G}$ is a Probabilistic Polynomial Time (PPT) algorithm that, on input 1^λ, outputs the description of groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, so that $(\mathbb{G}_1, \mathbb{G}_2)$ form a bilinear group pair.

We now define the asymptotic SDH assumption. In these settings, the quantity $\mathsf{SDH\,Adv}_{q,\mathcal{A}}$ from Equation (1) becomes a function of λ. It is easy to interpret our results based on Definition 6.

Definition 6. Let $\mathcal{G}$ be a bilinear group generator. We say that the SDH assumption holds for $\mathcal{G}$ if, for every PPT algorithm $\mathcal{A}$ and for every polynomially bounded function $q : \mathbb{Z} \to \mathbb{Z}$ the function $\mathsf{SDH\,Adv}_{q(\lambda),\mathcal{A}}(\lambda)$ is a negligible function in λ.

3.3 Properties of SDH and Related Assumptions

As we will see and exploit in the next section, the SDH problem shares with the Strong RSA problem the very useful property of admitting a large number of solutions that are "insulated" from each other. Therefore, we view the SDH assumption as a discrete logarithm analogue of the Strong RSA assumption, even though the latter is based on the hardness of factoring.

Random Self Reduction. We observe that the Strong Diffie-Hellman problem has a simple random self-reduction. Consider an instance $(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x)$ given as input. To reduce this input to a random q-SDH instance in the same groups, we select random $y, z_1, z_2 \in \mathbb{Z}_p^\times$, define $\tilde{g}_1 \leftarrow g_1^{z_1}$ and $\tilde{g}_2 \leftarrow g_2^{z_2}$, and compute $\tilde{g}_1^{(xy)^i} \leftarrow (g_1^{x^i})^{y^i z_1}$ for $i = 1, \ldots, q$, and $\tilde{g}_2^{xy} \leftarrow (g_2^x)^{yz_2}$. Consequently, $(\tilde{g}_1, \tilde{g}_1^{(xy)}, \ldots, \tilde{g}_1^{(xy)^q}, \tilde{g}_2, \tilde{g}_2^{(xy)})$ is a random q-SDH instance, statistically independent from the first. Then, given any valid solution $(\tilde{c}, \tilde{h}) = (\tilde{c}, \tilde{g}_1^{1/(xy+\tilde{c})})$ to the second instance, we can extract a solution (c, h) to the first by letting $c \leftarrow \tilde{c}/y \pmod p$ and $h \leftarrow (\tilde{h})^{y/z_1} = g_1^{1/(x+c)}$.

Relation to DHI. We mention that a weaker version of the SDH assumption was previously used by Mitsunari, Sakai, and Kasahara [35] to construct a traitor tracing system (see also [47]), by Boneh and Boyen [8] to construct an identity-based encryption system, and by Dodis and Yampolskiy [19] to construct a verifiable pseudo-random function.

Using our notation, this weaker assumption requires the solver to output $g_1^{1/(x+c)}$ for a *prescribed* value of c listed in the problem instance. Recall that in SDH the solver may freely choose c. The weaker assumption can be shown to be equivalent to the hardness of the following problem (stated concretely in a single group $\mathbb{G}$): given $g, g^x, g^{(x^2)}, \ldots, g^{(x^q)}$, output $g^{1/x}$. This problem is called the q-Diffie-Hellman Inversion problem or q-DHI for short. We note that each instance of the DHI problem admits exactly one solution, whereas an SDH instance in a group of order p admits $p-1$ distinct solutions. Indeed, despite their superficial resemblance, the SDH and DHI assumptions exhibit rather different properties.

Generic Group Analysis. To provide some confidence in the SDH assumption, we gave in [9] a lower bound on the complexity of solving the q-SDH problem for any suitably bounded q in a generic group [45] of prime order p. In particular, we showed that any generic solver for the q-SDH problem with $q < O(\sqrt[3]{p})$ must run in expected time $\Omega(\sqrt{p/q})$.

Analyses due to Brown and Gallant [12] and to Cheon [14] imply that for special p, the secret exponent x in the q-SDH problem (and related problems such as q-DHI) can be recovered generically in less time than needed to compute the discrete log. More precisely, Cheon shows that if either $p-1$ or $p+1$ has a factor $r \leq q$, then the secret x can be computed generically in time $\tilde{O}(\sqrt{p/r}+\sqrt{r})$ or $\tilde{O}(\sqrt{p/r}+r)$ respectively, rather than $O(\sqrt{p})$ per the generic discrete log. For values of $q \approx \sqrt[3]{p}$, Cheon's method is most efficient when there is a large factor $r \approx q$, in which case the time and space complexity is $O(\log p \cdot \sqrt[3]{p}) = \tilde{O}(\sqrt{p/q})$. Hence, these generic algorithms can be viewed as a matching upper bound to our generic lower bound, showing that the lower bound is tight, at least for certain p. We prove the generic lower bound for all primes p in Section 6.

4 Short Signatures Without Random Oracles

We now construct a fully secure short signature scheme in the standard model using the SDH assumption. We consider this to be the main result of the paper.

4.1 The Full Signature Scheme

Let $(\mathbb{G}_1, \mathbb{G}_2)$ be bilinear groups where $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ for some prime p. For the moment we assume that the messages m to be signed are elements in $\mathbb{Z}_p$, but as we mention in Section 4.7, the domain can be extended to all of $\{0,1\}^\star$ using (target) collision resistant hashing.

Key Generation: Select random generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, and random integers $x, y \in \mathbb{Z}_p^\times$. Compute $u \leftarrow g_2^x \in \mathbb{G}_2$ and $v \leftarrow g_2^y \in \mathbb{G}_2$. Also compute $z \leftarrow e(g_1, g_2) \in \mathbb{G}_T$. The public key is the tuple (g_1, g_2, u, v, z). The secret key is the triple (g_1, x, y).

Signing: Given a secret key (g_1, x, y) and a message $m \in \mathbb{Z}_p$, pick a random $r \in \mathbb{Z}_p \setminus \{-\frac{x+m}{y}\}$ and compute $\sigma \leftarrow g_1^{1/(x+m+yr)} \in \mathbb{G}_1$. Here, the inverse $1/(x+m+yr)$ is computed modulo p. The signature is the pair (σ, r).

Verification: Given a public key (g_1, g_2, u, v, z), a message m, and a signature (σ, r), verify that $(g_1,\ g_2,\ \sigma,\ g_2^m v^r u)$ is a DDH tuple by testing whether $e(\sigma,\ u \cdot g_2^m \cdot v^r) = z$. If the equality holds the signature is declared `valid`; otherwise it is declared `invalid`.

We note that including both g_1 and z in the public key is redundant. It is convenient to carry them both in the description of the system since g_1 is needed for signing and z is used for fast verification. In practice, g_1 need not be included in the public key.

Theorem 7. *The signature scheme is consistent.*

Proof. We need to show that for all key pairs and all messages, any signature generated by the signing procedure verifies as `valid` under the corresponding public key. Indeed, we have

$$e(\sigma,\ u \cdot g_2^m \cdot v^r) = e(g_1^{1/(x+m+yr)},\ g_2^x \cdot g_2^m \cdot g_2^{yr}) = e(g_1,\ g_2)^{\frac{x+m+yr}{x+m+yr}} = z$$

provided that $x + m + yr \neq 0 \pmod{p}$, as required. □

4.2 Main Features and Security

We list below the principal properties of our signature scheme. Additional features, extensions, and related constructions will be discussed in later sections.

Bandwidth. A signature contains two elements (σ, r), each of length approximately $\log_2 p$ bits, therefore the total signature length is approximately $2 \log_2 p$. When we instantiate the pairing using the elliptic curves described in [11, 3], we obtain a signature whose length is approximately the same as a DSA signature with the same security, but which is proven secure without the random oracle model.

Performance. Key generation times are comparable to the BLS scheme [11]. Signature times are much faster than BLS, by up to an order of magnitude, because our signing algorithm only makes one exponentiation to the fixed base g_1, and this can be greatly accelerated with a moderate amount of reusable pre-computation. Verification times are also faster than BLS since verification requires only one pairing and one multi-exponentiation, instead of two pairing computations in BLS. Since exponentiation tends to be faster than pairing, signature verification is faster than in the BLS system. We note that BLS verification time can be improved using multi-pairing [26], but the result is still slower than the system in this paper.

Security. The following theorem shows that the scheme above is existentially unforgeable in the strong sense under adaptive chosen message attacks, provided that the SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$. We consider an attacker who makes up to q_S adaptive signature queries, and reduce the forgery to the resolution of a random q-SDH instance for $q = q_S$. Our reduction is *tight* (up to a small constant factor approximately equal to 2).

Theorem 8. *Suppose the (q, t', ϵ')-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$. Then the signature scheme above is (t, q_S, ϵ)-secure against strong existential forgery under an adaptive chosen message attack provided that*

$$q_S \leq q \qquad , \qquad \epsilon \geq 2\epsilon' + q_S/p \approx 2\epsilon' \qquad \textit{and} \qquad t \leq t' - \Theta(q^2 T)$$

where T is the maximum time for an exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{Z}_p$.

Proof. We prove the theorem using two lemmas. In Lemma 9, we first describe a simplified signature scheme and prove its existential unforgeability against *weak* chosen message attacks under the SDH assumption. In Lemma 10, we then show that the security of the basic scheme implies the security of the full scheme. From these results (Lemmas 9 and 10), Theorem 8 follows easily. We present the proof in two steps since we will make another use of the construction used to prove Lemma 9 later on in this paper. □

4.3 A Weakly Secure Short Signature Scheme

We first show how the SDH assumption can be used to construct an existentially unforgeable scheme under a *weak* chosen message attack. This construction demonstrates the main properties of the SDH assumption. In the next section we show that the security of this basic scheme implies the security of the full scheme.

The weakly secure short signature scheme is as follows. As before, let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair where $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ for some prime p. For the moment we assume that the messages m to be signed are elements in $\mathbb{Z}_p$.

Key Generation: Select random generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, and a random integer $x \in \mathbb{Z}_p^\times$. Compute $v \leftarrow g_2^x \in \mathbb{G}_2$ and $z \leftarrow e(g_1, g_2) \in \mathbb{G}_T$. The public key is the tuple (g_1, g_2, v, z). The secret key is (g_1, x).

Signing: Given a secret key (g_1, x) and a message $m \in \mathbb{Z}_p$, output the signature $\sigma \leftarrow g_1^{1/(x+m)} \in \mathbb{G}_1$. Here $1/(x+m)$ is computed modulo p. By convention in this context we define $1/0$ to be 0 so that in the unlikely event that $x + m = 0$ we have $\sigma \leftarrow 1 \in \mathbb{G}_1$.

Verification: Given a public key (g_1, g_2, v, z), a message m, and a signature σ, verify the equality $e(\sigma,\ v \cdot g_2^m) = z$. If the equality holds, or if $\sigma = 1$ and $v \cdot g_2^m = 1$, the result is `valid`. Otherwise, the result is `invalid`.

Again, in practice one would omit g_1 from the public key, and keep it instead with the private key.

We show that the basic signature scheme above is existentially unforgeable under a *weak* chosen message attack (Definition 2). The proof of the following lemma uses a similar method to the proof of Theorem 3.5 of Mitsunari et al. [35].

Lemma 9. *Suppose the (q, t', ϵ)-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$. Then the basic signature scheme above is (t, q_S, ϵ)-secure against existential forgery under a weak chosen message attack provided that*

$$q_S \leq q \qquad \textit{and} \qquad t \leq t' - \Theta(q^2 T)$$

where T is the maximum time for an exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{Z}_p$.

Proof. Assume that $\mathcal{A}$ is a forger that (t, q_S, ϵ)-breaks the signature scheme. We construct an algorithm $\mathcal{B}$ that, by interacting with the forger $\mathcal{A}$, solves the q-SDH problem in time t' with advantage ϵ.

Algorithm $\mathcal{B}$ is given a random instance $(g_1, d_1, \ldots, d_q, g_2, h)$ of the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$, where $d_i = g_1^{(x^i)} \in \mathbb{G}_1$ for $i = 1, \ldots, q$, and $h = g_2^x \in \mathbb{G}_2$, for some unknown $x \in \mathbb{Z}_p$. For convenience we set $d_0 \leftarrow g_1$. The objective of $\mathcal{B}$ is to produce a pair $(c, g_1^{1/(x+c)})$ for some value $c \in \mathbb{Z}_p \setminus \{-x\}$ of its choice. Algorithm $\mathcal{B}$ does so by interacting with the forger $\mathcal{A}$ as follows:

Query: The attacker $\mathcal{A}$ outputs a list of q_S distinct messages $m_1, \ldots, m_{q_s} \in \mathbb{Z}_p$, where $q_S \leq q$. Since $\mathcal{A}$ must reveal its queries up front, we may assume that $\mathcal{A}$ outputs exactly q messages to be signed. (If fewer queries are made, we can always virtually reduce the value of q to $q' = q_S$, since the hardness of q-SDH entails that of q'-SDH for all $q' < q$.)

Response: The simulator $\mathcal{B}$ must respond with a public key and q signatures on the respective messages from $\mathcal{A}$. Let f be the univariate polynomial defined as $f(X) = \prod_{i=1}^{q}(X + m_i)$. Expand f and write $f(X) = \sum_{i=0}^{q} \alpha_i X^i$ where $\alpha_0, \ldots, \alpha_q \in \mathbb{Z}_p$ are the coefficients of the polynomial f. Algorithm $\mathcal{B}$ picks a random $\theta \in \mathbb{Z}_p^\times$, and computes

$$g_1' \leftarrow \prod_{i=0}^{q} d_i^{\alpha_i \theta} \in \mathbb{G}_1 \qquad \text{hence} \qquad g_1' = g_1^{\theta f(x)}$$

Algorithm $\mathcal{B}$ also computes $z' = e(g_1', g_2)$. The public key given to $\mathcal{A}$ is (g_1', g_2, h, z'). It has the correct distribution provided that $f(x) \neq 0$; in particular, g_1' and g_2 are independently and uniformly distributed random generators of their respective groups, thanks to the action of θ. If, however, $f(x) = 0$, then $x = -m_i$ for some i, in which case $\mathcal{B}$ can easily recover the secret key x, and hence solve the given instance of the SDH problem with no further help from the forger $\mathcal{A}$.

Next, for each $i = 1, \ldots q$, the simulator $\mathcal{B}$ must generate a signature σ_i on m_i. To do so, let f_i be the polynomial $f_i(X) = f(X)/(X + m_i) = \prod_{j=1, j\neq i}^{q}(X + m_j)$. As before, we expand f_i and write $f_i(X) = \sum_{j=0}^{q-1} \beta_j X^j$ while calculating its coefficients. Algorithm $\mathcal{B}$ computes

$$\sigma_i \leftarrow \prod_{j=0}^{q-1} d_j^{\beta_j \theta} \in \mathbb{G}_1 \qquad \text{hence} \qquad \sigma_i = g_1^{\theta f_i(x)} = (g_1')^{1/(x+m_i)}$$

Observe that σ_i is a valid signature on message m_i under the public key (g_1', g_2, h, z'), since $e\big(\sigma_i,\ h \cdot (g_2')^{m_i}\big) = e\big((g_1')^{1/(x+m_i)},\ (g_2^x)(g_2)^{m_i}\big) = z'$. Algorithm $\mathcal{B}$ performs these steps for each message, and gives to $\mathcal{A}$ the $q - 1$ resulting signatures $\sigma_1, \ldots, \sigma_q$. Since each message admits only a unique signature, the output distribution is trivially correct.

Output: The forger $\mathcal{A}$ returns a forgery (m_*, σ_*) such that $\sigma_* \in \mathbb{G}_1$ is a valid signature on $m_* \in \mathbb{Z}_p$. Note that, necessarily, $m_* \notin \{m_1 \ldots, m_q\}$ since the pair (m_*, σ_*) is novel and to each message corresponds only one valid signature. Since a successful forgery entails $e(\sigma_*,\ h \cdot g_2^{m_*}) = z'$ where $h = g_2^x$ and $z' = e(g_1', g_2)$ we deduce that $e(\sigma_*,\ g_2^{x+m_*}) = e(g_1', g_2)$ and therefore

$$\sigma_* = (g_1')^{1/(x+m_*)} = (g_1)^{\theta \cdot f(x)/(x+m_*)}$$

We use long division to compute the ratio $f(x)/(x + m_*)$ that appears in the exponent. Using long division we rewrite the polynomial f as $f(X) = (X + m_*)\gamma(X) + \gamma_*$ for some

easily computable polynomial $\gamma(X) = \sum_{i=0}^{q-1} \gamma_i X^i$ and constant $\gamma_* \in \mathbb{Z}_p$. Then the ratio $f(X)/(X+m_*)$ can be written as $f(X)/(X+m_*) = \frac{\gamma_*}{X+m_*} + \sum_{i=0}^{q-1} \gamma_i X^i$ and the expression of σ_* becomes

$$\sigma_* = g_1^{\theta \cdot \left(\frac{\gamma_*}{x+m_*} + \sum_{i=0}^{q-1} \gamma_i x^i\right)}$$

Observe that $\gamma_* \neq 0$, since $f(X) = \prod_{i=1}^{q}(X+m_i)$ and $m_* \notin \{m_1, \ldots, m_q\}$, as thus $(X+m_*)$ does not divide $f(X)$. Taking roots of order θ and γ_* modulo p, the simulator $\mathcal{B}$ can compute

$$w \leftarrow \left((\sigma_*)^{1/\theta} \cdot \prod_{i=0}^{q-1} (d_i)^{-\gamma_i} \right)^{1/\gamma_*} \in \mathbb{G}_1$$

Hence, we obtain

$$w = \left(g_1^{\frac{\gamma_*}{x+m_*}} \cdot g_1^{\sum_{i=0}^{q-1} \gamma_i x^i} \cdot \prod_{i=0}^{q-1} g_1^{-\gamma_i x^i} \right)^{1/\gamma_*} = g_1^{1/(x+m_*)}$$

$\mathcal{B}$ returns the pair (m_*, w) as the solution to the submitted instance of the SDH problem.

The claimed bounds are obvious by construction of the reduction. □

4.4 From Weak Security To Full Security

We now present a reduction from the security of the basic scheme we just described to the security of our full signature scheme presented in Section 4.1. This will complete the proof of Theorem 8.

Lemma 10. *Suppose that the basic signature scheme of Section 4.3 is (t', q_S, ϵ')-weakly secure. Then the full signature scheme is (t, q_S, ϵ)-secure against strong existential forgery under an adaptive chosen message attack provided that*

$$\epsilon \geq 2\epsilon' + q_S/p \approx 2\epsilon' \qquad \text{and} \qquad t \leq t' - \Theta(q_S T)$$

where T is the maximum time for an exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{Z}_p$.

We first give some intuition for the proof. Suppose $\mathcal{A}$ is a forger for the full scheme under an adaptive chosen message attack. We build a forger $\mathcal{B}$ for the basic scheme under a weak chosen message attack. Forger $\mathcal{B}$ starts by requesting signatures on random messages $w_1, \ldots, w_q \in \mathbb{Z}_p$. In response, it is given a public key (g_1, g_2, u, z) and signatures $\sigma_1, \ldots, \sigma_{q_s} \in \mathbb{G}_1$ for the basic scheme.

In principle, $\mathcal{B}$ could create a public key for the full scheme by picking a random $y \in \mathbb{Z}_p^\times$ and giving $\mathcal{A}$ the public key (g_1, g_2, u, g_2^y, z). Now, when $\mathcal{A}$ issues a signature query for an adaptively chosen message $m_i \in \mathbb{Z}_p$, forger $\mathcal{B}$ could choose an $r_i \in \mathbb{Z}_p$ such that $m_i + yr_i$ equals w_i. Then (σ_i, r_i) would be a valid signature on m_i for the full scheme and hence a proper response to $\mathcal{A}$'s query. Eventually, $\mathcal{A}$ would output a forgery (m_*, σ_*, r_*). Since $(m_* + yr_*, \sigma_*)$ would then be a valid message/signature pair for the basic scheme, $\mathcal{B}$ could output that pair as an existential forgery against the basic scheme.

The only problem is that $m_* + yr_*$ might be in $\{w_1, \ldots, w_{q_s}\}$ in which case $(m_* + yr_*, \sigma_*)$ would not be a valid existential forgery for the basic scheme. However, when this happens $\mathcal{B}$ learns the value of y, which would be useful information if it did not know it already, i.e., if the public key had been constructed differently. Dealing with this case forces us to consider two types of adversaries, which complicates the proof by requiring us to build a different reduction for either adversary. The full proof follows.

Proof of Lemma 10. Assume that $\mathcal{A}$ is a forger that (t, q_S, ϵ)-breaks the full signature scheme. We construct an algorithm $\mathcal{B}$ that $(t',\ q_S,\ \frac{1}{2}(\epsilon - q_S/p))$-weakly breaks the basic signature scheme of Section 4.3.

Before describing the algorithm $\mathcal{B}$ we distinguish between two types of forgers that $\mathcal{A}$ can emulate. Let (h_1, h_2, U, V, z) be the public key given to $\mathcal{A}$, where $U = g_2^x$ and $V = g_2^y$. First, we note that by adding dummy queries as necessary, we may always assume that $\mathcal{A}$ makes exactly q_S signature queries. Suppose then that $\mathcal{A}$ adaptively asks for signatures on messages $m_1, \ldots, m_{q_s} \in \mathbb{Z}_p$ and is given signatures (σ_i, r_i) for $i = 1, \ldots, q_S$ in response. Let $w_i = m_i + yr_i$ for each i, and denote by (m_*, σ_*, r_*) the forgery eventually produced by $\mathcal{A}$.

1. We say that $\mathcal{A}$ is a type-1 forger, denoted $\mathcal{A}_1$, if it either
   (a) makes a signature query for the message $m = -x$, or
   (b) outputs a forgery where $m_* + yr_* \notin \{w_1, \ldots, w_{q_s}\}$.

2. We say that $\mathcal{A}$ is a type-2 forger, denoted $\mathcal{A}_2$, if it both
   (a) never makes a signature query for the message $m = -x$, and
   (b) outputs a forgery where $m_* + yr_* = w_i$ for some $i \in \{1, \ldots, q_S\}$.

These cases form a partition that exhausts all possible successful forgeries. We show that in either case the forger can be exploited to forge a signature in the weak signature scheme of Section 4.3. However, the reduction works differently for each forger type. For each type of forger $\mathcal{A}_1$, $\mathcal{A}_2$, we show how to construct a suitable simulator $\mathcal{B}_1$, $\mathcal{B}_2$.

Type-1 forger. First, we describe the simulator $\mathcal{B}_1$, which interacts with a type-1 forger $\mathcal{A}_1$ to produce a forgery for the signature scheme of Section 4.3, as follows:

Setup: Algorithm $\mathcal{B}_1$ selects a list of q_S random messages $w_1, \ldots, w_{q_s} \in \mathbb{Z}_p$, which it sends to the challenger. The challenger responds with a valid public key (g_1, g_2, u, z) and a list of q_S signatures $\sigma_1, \ldots, \sigma_{q_s} \in \mathbb{G}_1$ on these messages. Algorithm $\mathcal{B}_1$ checks whether all $\sigma_i \neq 1 \in \mathbb{G}_1$. If some $\sigma_i = 1 \in \mathbb{G}_1$, then $\mathcal{B}_1$ just learned the challenger's private key, $x = -w_i$, which it can then use to produce a valid forgery. Otherwise, we know that all w_i are uniform in $\mathbb{Z}_p \setminus \{-x\}$ and that $e(\sigma_i, g_2^{w_i} u) = e(g_1, g_2) = z$ for $i = 1, \ldots, q_S$.

To proceed, $\mathcal{B}_1$ picks a random $y \in \mathbb{Z}_p^\times$ and gives to $\mathcal{A}_1$ the public key $PK_1 = (g_1, g_2, U, V, z) \leftarrow (g_1, g_2, u, g_2^y, z)$. Note that PK_1 does not depend on the w_i or σ_i.

Queries: The forger $\mathcal{A}_1$ issues q_S signature queries in an adaptive fashion. In order to respond, $\mathcal{B}_1$ maintains a query counter ℓ which is initially set to 0.

Upon receiving a signature query for $m \in \mathbb{Z}_p$, the simulator $\mathcal{B}_1$ increments ℓ by one, sets $r_\ell \leftarrow (w_\ell - m)/y \in \mathbb{Z}_p$, and gives $\mathcal{A}_1$ the signature (σ_ℓ, r_ℓ). We claim that (σ_ℓ, r_ℓ) is a valid signature on m under PK_1. First, r_ℓ is uniform in $\mathbb{Z}_p \setminus \{-\frac{x+m}{y}\}$ since w_ℓ is uniform in $\mathbb{Z}_p \setminus \{-x\}$. Second,

$$e(\sigma_\ell,\ U \cdot g_2^m \cdot V^{r_\ell}) = e(\sigma_\ell,\ u \cdot g_2^m \cdot g_2^{yr_\ell}) = e(\sigma_\ell,\ u \cdot g_2^{w_\ell}) = z$$

as required. The reason this works is that $\mathcal{B}_1$ chose an r_ℓ such that $m_\ell + yr_\ell = w_\ell$. We set $m_\ell \leftarrow m$.

A type-1 forger may issue a signature query for $m \in \mathbb{Z}_p$ where $m = -x$. If this ever happens then Algorithm $\mathcal{B}_1$ obtains the private key for the public key (g_1, g_2, u, z) it was given. This

allows $\mathcal{B}_1$ to forge the signature on any message of its choice without further interaction with $\mathcal{A}_1$. It terminates the simulation and wins the game.

Output: Eventually, suppose $\mathcal{A}_1$ returns a forgery (m_*, σ_*, r_*), where (σ_*, r_*) is a valid forgery distinct from any previously given signature on message m_*. Since the forgery is valid, we have

$$e(g_1, g_2) = e(\sigma_*,\ U \cdot g_2^{m_*} \cdot V^{r_*}) = e(\sigma_*,\ u \cdot g_2^{m_* + y r_*})$$

Let $w_* = m_* + y r_*$. It follows that (w_*, σ_*) is a valid message/signature pair in the basic signature scheme. Furthermore, the pair is a valid existential forgery in that scheme since for a type-1 forger we have $w_* \notin \{w_1, \ldots, w_{q_s}\}$.

It is easy to see that, if the forger $\mathcal{A}_1$ outputs a valid forgery with probability ϵ in time t, then the reduction $\mathcal{B}_1$ succeeds in time $t + \Theta(q_S T)$ with the same probability ϵ.

Type-2 forger. Second, we describe the simulator $\mathcal{B}_2$, which interacts with a type-2 forger $\mathcal{A}_2$ to produce a forgery for the signature scheme of Section 4.3, as follows:

Setup: Algorithm $\mathcal{B}_2$ starts by sending a list of q_S random messages $w_1, \ldots, w_{q_s} \in \mathbb{Z}_p^\times$ to its challenger, and receives in response a valid public key (g_1, g_2, u, z) and a list of q_S signatures $\sigma_1, \ldots, \sigma_{q_s} \in \mathbb{G}_1$ on these messages. We write $u = g_2^y$ for some $y \in \mathbb{Z}_p^\times$. Notice that for this reduction we restricted the w_i to non-zero values.

Algorithm $\mathcal{B}_2$ checks whether some $\sigma_i = 1 \in \mathbb{G}_1$, in which case $u = g_2^{-w_i}$, which would allow $\mathcal{B}_2$ to produce a valid forgery using $-w_i$ as private key. In this case $\mathcal{B}_2$ stops the simulation and wins the game. Otherwise, for all $i = 1, \ldots, q_S$ the w_i are uniform in $\mathbb{Z}_p^\times \setminus \{-y\}$ and satisfy $e(\sigma_i, g_2^{w_i} u) = e(g_1, g_2) = z$.

To proceed, $\mathcal{B}_2$ picks a random $x \in \mathbb{Z}_p^\times$ and gives to $\mathcal{A}_2$ the public key $PK_2 = (g_1, g_2, U, V, z) \leftarrow (g_1, g_2, g_2^x, u, z)$.

Queries: The forger $\mathcal{A}$ issues q_S signature queries in an adaptive fashion. In order to respond, $\mathcal{B}$ maintains a list L of tuples (m_i, r_i, W_i) which is initially empty, and a query counter ℓ which is initially set to 0.

Upon receiving a signature query for m, the algorithm $\mathcal{B}_2$ increments ℓ by one, and defines $r_\ell \leftarrow (x + m)/w_\ell \in \mathbb{Z}_p$. Note that $r_\ell \neq 0$ since $m \neq -x$ in a type-2 forgery. $\mathcal{B}_2$ then adds the tuple $(m, r_\ell, g_2^m V^{r_\ell})$ to the list L, and responds to the query by giving $\mathcal{A}_2$ the signature $(\sigma_\ell^{1/r_\ell}, r_\ell)$. This is a valid signature on m under PK_2 since

$$e(\sigma_\ell^{1/r_\ell},\ U \cdot g_2^m \cdot V^{r_\ell}) = e(\sigma_\ell^{1/r_\ell},\ g_2^x \cdot g_2^m \cdot u^{r_\ell}) = e(\sigma_\ell,\ g_2^{(x+m)/r_\ell} \cdot u) = e(\sigma_\ell,\ g_2^{w_\ell} \cdot u) = z$$

Since r_ℓ is uniform over $\mathbb{Z}_p^\times \setminus \{-\frac{x+m}{y}\}$ instead of $\mathbb{Z}_p \setminus \{-\frac{x+m}{y}\}$, the signature is almost correctly distributed with a statistical distance of $1/p$ from the correct distribution. Taken as a whole, the q_S signatures are jointly distributed with statistical distance at most q_S/p from the specification.

Output: Eventually, suppose $\mathcal{A}_2$ returns a forgery (m_*, σ_*, r_*), where (σ_*, r_*) is a valid forgery distinct from any previously given signature on message m_*. Let $W_* \leftarrow g_2^{m_*} V^{r_*}$, and let (m_j, r_j, W_j) be a tuple on the list L such that $W_j = W_*$; in a type-2 forgery such a tuple always exists. Since $V = u$ we know that $g_2^{m_j} u^{r_j} = g_2^{m_*} u^{r_*}$. Write $V = g_2^y$ for some $y \in \mathbb{Z}_p^\times$ so that $m_j + y r_j = m_* + y r_*$. We know that $(m_j, r_j) \neq (m_*, r_*)$, otherwise the forgery would be

identical to a previously given signature on the query message m_j. Since $g_2^{m_j} u^{r_j} = g_2^{m_*} u^{r_*}$, it follows that $m_j \neq m_*$ and $r_j \neq r_*$. Therefore, $\mathcal{B}_2$ can compute $y \leftarrow (m_* - m_j)/(r_j - r_*) \in \mathbb{Z}_p^\times$, thus recovering the private key corresponding to the public key (g_1, g_2, u, z) it was given. Algorithm $\mathcal{B}_2$ can then forge a signature on any message of its choice.

It is easy to see that, if the forger $\mathcal{A}_2$ outputs a valid forgery with probability ϵ in time t, then the reduction $\mathcal{B}_2$ succeeds in time $t + \Theta(q_S T)$ with probability at least $\epsilon - q_S/p$.

This completes the description of the two reduction algorithms, $\mathcal{B}_1$ and $\mathcal{B}_2$. Regardless of which reduction is used, a standard argument shows that if the algorithm does not abort, then, from the viewpoint of the adversary, $\mathcal{A}_1$ or $\mathcal{A}_2$, the simulation is indistinguishable from a real attack environment. In particular, the public keys and signatures are correctly distributed, and the adversary cannot tell whether it is interacting with $\mathcal{B}_1$ and $\mathcal{B}_2$.

Therefore, given an arbitrary adversary $\mathcal{A}$ that (t, q_S, ϵ)-breaks the full signature scheme but whose type is unknown, it suffices to let $\mathcal{B}$ be an algorithm that randomly executes one of $\mathcal{B}_1$ and $\mathcal{B}_2$ with equal probability. We obtain an algorithm that breaks the basic signature scheme with probability $\frac{1}{2}\min(\epsilon, \epsilon - q_S/p) = \frac{1}{2}(\epsilon - q_S/p) \geq \epsilon'$ in time $t + \Theta(q_S T) \geq t'$. This completes the proof of Lemma 10. □

Since in the full scheme a single message has many valid signatures, it is worth repeating that the full signature scheme is existentially unforgeable in the strong sense: the adversary cannot make any forgery, even on messages which are already signed.

4.5 Relation to Chameleon Hash Signatures

It is instructive to consider the relation between the full signature scheme of Section 4.1 and an elegant signature construction based on the Strong RSA assumption due to Gennaro, Halevi, and Rabin (GHR) [23]. GHR signatures are pairs $(r, s^{1/H(m,r)})$ where H is a Chameleon hash [30], r is random in some range, and arithmetic is done modulo an RSA modulus N. Looking closely, one can see some parallels between the proof of security in Lemma 10 and the proof of security in [23]. There are three interesting points to be made:

- The $m + yr$ component in our signature scheme provides us with the functionality of a Chameleon hash: given m, we can choose r so that $m + yr$ maps to some predefined value of our choice. This makes it possible to handle adaptive chosen message attacks. Embedding the hash $m + yr$ directly within the signature scheme results in a much more efficient construction than using an explicit Chameleon hash (which requires additional exponentiations). Such an easy embedding is not known to be possible with Strong RSA signatures, though we refer to the work of Fischlin [20] for ideas in that direction.

- One difficulty with GHR signatures is that given a "non-prime" solution such as $(6, s^{1/6})$ to the Strong RSA problem, one can deduce another solution, e.g., $(3, s^{1/3})$. Thus, given a GHR signature on one message it possible to deduce a GHR signature on another message (see [23, 16] for details). Gennaro et al. [23] solve this problem by ensuring that $H(m, r)$ always maps to a prime; however, that makes it difficult to compute the hash (a different solution is given in [18]). This issue does not come up at all in the SDH assumption and in our constructions.

- We obtain short signatures since, unlike Strong RSA, the SDH assumption applies to groups with a short element representation. This is especially true at high security levels.

Thus, we see that Strong Diffie-Hellman leads to signatures that are simpler and shorter than their Strong RSA counterparts.

4.6 Limited Message Recovery

We now describe another useful property of the signature schemes whereby the total size of signed messages can be further reduced at the cost of increasing the verification time. The technique applies equally well to the fully secure signature scheme as to the weakly secure one. Unlike all the other constructions in the paper, it also makes fuller use of the pairing than as the mere provider of a DDH predicate.

A common technique for shortening the total length of a signed message is to encode a part of the message in the signature [33, §11]. In the standard terminology, such schemes are called signatures with message recovery, as opposed to signatures with appendix. Signatures based on trapdoor permutations support very efficient message recovery. At the other end of the spectrum, any signature scheme can support a very inefficient form of message recovery, based on the following trivial signature compression mechanism. Rather than transmit a message/signature pair (M, σ), the sender transmits $(\hat{M}, \sigma)$ where $\hat{M}$ is a truncated version of M that omits its last t bits. To verify $(\hat{M}, \sigma)$ and recover M, the verifier tries to verify the signature on all concatenations of $\hat{M}$ with the 2^t possible values for the missing bits; if one verification succeeds, the signature is accepted and the corresponding reconstitution is output as the message M. This trivial method shows that a signed message (M, σ) can be shortened by t bits at the cost of increasing verification time by a factor of 2^t.

For our signature scheme we obtain a better tradeoff than with the trivial method above, although not as good as provided by trapdoor permutations. The signed message (M, σ) can be shortened by t bits at the cost of increasing verification time by a factor of $2^{t/2}$ only. We refer to this property as limited message recovery. Our technique applies to both the full strongly secure signature scheme of Section 4.1 and the basic weakly secure signature scheme of Section 4.3. The main requirement is that whichever scheme is used be implemented using an actual pairing (instead of a generic DDH predicate).

Achieving Limited Message Recovery. For simplicity, we only show how limited message recovery applies to the full signature scheme. Assume messages are k-bit strings represented as integers in $\mathbb{Z}_p$. Let (g_1, g_2, u, v, z) be a public key in the full scheme—although for this application one might prefer to abbreviate the public key as (g_2, u, v) and let the verifier derive g_1 and z. Suppose we are given the signed message $(\hat{m}, \sigma, r)$ where $\hat{m}$ is a truncation of the last t bits of $m \in \mathbb{Z}_p$. Thus $m = \hat{m} \cdot 2^t + \delta$ for some integer $0 \leq \delta < 2^t$. Our goal is to verify the signed message $(\hat{m}, \sigma, r)$ and to reconstruct the missing bits δ in time $2^{t/2}$. To do so, we first rewrite the verification equation $e(\sigma,\ u \cdot v^r \cdot g_2^m) = e(g_1, g_2)$ as

$$e(\sigma, g_2)^m = \frac{e(g_1, g_2)}{e(\sigma,\ u \cdot v^r)}$$

Then, substituting $m = \hat{m} \cdot 2^t + \delta$ we obtain

$$e(\sigma, g_2)^\delta = \frac{e(g_1, g_2)}{e(\sigma,\ u \cdot v^r \cdot g_2^{\hat{m}2^t})} \tag{2}$$

Now, we say that $(\hat{m}, \sigma, r)$ is valid if there exists an integer $\delta \in [0, 2^t)$ satisfying Equation (2). Finding such a δ takes time approximately $2^{t/2}$ using Pollard's Lambda method [33, p.128] for

computing discrete logarithms. Thus, we can verify the signature and recover the t missing message bits in time $2^{t/2}$, as required.

Very Short Weakly Secure Signatures. Obvious applications of limited message recovery are situations where bandwidth is extremely limited, such as when the signature is an authenticator that is to be typed-in by a human. The messages in such applications are typically chosen and signed by a central authority, so that adaptive chosen message attacks are typically not a concern. It is safe in those cases to use the weakly secure signature scheme of Section 4.3, and apply limited message recovery to further shrink the already compact signatures it produces. Specifically, using t-bit truncation as above we obtain a total signature overhead of $(160 - t)$ bits for common security parameters, at the cost of requiring $2^{t/2}$ arithmetic operations for signature verification. The total bandwidth requirement is thus even smaller than BLS signatures in the random oracle model [11], even though the security of our application does not rely on random oracles.

4.7 Arbitrary Message Signing

We can extend our signature schemes to sign arbitrary messages in $\{0,1\}^\star$, as opposed to merely messages in $\mathbb{Z}_p$, by first hashing the message using a collision resistant hash function $H : \{0,1\}^\star \to \mathbb{Z}_p$ prior to both signing and verifying. A standard argument shows that if the scheme above is secure against existential forgery under an adaptive chosen message attack (in the strong sense) then so is the scheme with the hash. The result is a signature scheme for arbitrary messages in $\{0,1\}^\star$. We note that there is no need for a full domain hash into $\mathbb{Z}_p$; a collision resistant hash function $H : \{0,1\}^\star \to \{1,\ldots,2^b\}$ such that $2^b < p$ is sufficient for the security proof. This transformation applies to both the fully and the weakly secure signature schemes described above.

More rigorously in the asymptotic setting, a target collision resistant hash (TCR) [38, 6, 46] may be substituted for H at the expense of a slightly longer signature to accommodate the extra random seed or index.

4.8 Off-line/On-line Signatures

Even though our signature scheme already provides a very efficient signing operation—dominated by a single inversion in $\mathbb{Z}_p$ and a single exponentiation to a fixed base—, it is possible to make it almost instantaneous for on-line signing by doing essentially all of the work off-line.

The idea is very simple. In the off-line phase, the signer would pick any number of random integers $\rho_i \in \mathbb{Z}_p^\times$, and for each of them compute $\sigma_i = g_1^{1/\rho_i}$ ahead of time; the pairs (ρ_i, σ_i) must be stored securely. In the on-line phase, to sign a message or message hash $m \in \mathbb{Z}_p$, the signer would select an unused pair (ρ, σ), erase it from its secure storage, compute $r = (\rho - m - x)\, y^{-1}$, and output the signature (σ, r). The entire on-line signature process requires only two subtractions and a single multiplication in $\mathbb{Z}_p$, provided that the signer stores his private key as (x, y^{-1}).

Observe that the private key is not needed in the off-line phase. In particular, this means that exposure of unused pair (ρ, σ) causes no harm as long as the compromised pairs are not subsequently used to create signatures. A similar approach was proposed by Shamir and Halevi [44].

4.9 Efficiency Considerations

As noted earlier, the signing operation in the full scheme is dominated by an inversion in $\mathbb{Z}_p^\times$ and an exponentiation with a fixed base $g_1 \in \mathbb{G}$. With some pre-computations, a fixed-base exponentiation

can be made much faster than a general exponentiation by eliminating the sequence of repeated squarings and grouping the ancillary multiplications into a few large chunks or windows.

Window-Ladder Pre-computation. Specifically, we would pre-compute a "power ladder" of w-bit "windows" for the fixed base $g_1 \in \mathbb{G}_1$, comprising the $2^w\lceil \log_2(p)/w \rceil$ group elements

$$(g_1)^{y \cdot 2^{kw}} \qquad \text{for all} \quad y = 0, \ldots, 2^w - 1 \qquad \text{and all} \quad k = 0, \ldots, \lceil \log_2(p)/w \rceil - 1$$

Using these values, raising g_1 to any power in $\mathbb{Z}_p$ requires no more than $\lceil \log_2(p)/w \rceil$ group operations in $\mathbb{G}_1$. The main cost of the pre-computation is the storage requirement, which grows as $2^w/w$; however, even for small windows of width $w \in \{4, \ldots, 8\}$ the efficiency gains are very substantial. To illustrate, a general exponentiation using the fast "signed m-ary windows" method [7, §IV.2] requires on average $(2^{m-2} - 2) + \lceil \log_2(p) \rceil + \lfloor \log_2(p) + 1 \rfloor/(m + 1)$ group operations, or about $6 + \frac{7}{6}\log_2(p)$ for the optimal choice $m = 5$ under common values of $\log_2(p)$. Hence, with our window-ladder method for $w = 8$, a fixed-base exponentiation will be close to 10 times faster than the fastest known general exponentiation algorithm. Notice also that the pre-computed values are not secret and are thus easy to handle. Since the value of g_1 rarely ever changes for a given signer, the (rather benign) pre-computation effort can be amortized over many signatures.

The off-line phase in the off-line/on-line signature process described in Section 4.8 benefits equally from pre-computations for all exponentiations (and, as noted earlier, the on-line phase is already almost instantaneous).

Real-World Performance. For comparison, a signature in our scheme can be generated in less than 1/10-th of the time needed for a BLS signature, since the latter involves a full domain hash directly into $\mathbb{G}_1$ as well as a general exponentiation (whose base is the unpredictable hash output). The speed-up of our scheme over RSA-based signatures is even more pronounced since not only do the latter require a general exponentiation, they also need a much larger modulus for a given security level.

The following table, courtesy of Shacham [43], compares the time required to perform general and fixed-base exponentiations in a group $\mathbb{G}_1$ of 158-bit prime order, using various methods. Here, an exponentiation in $\mathbb{G}_1$ is a point multiplication on an MNT [36] elliptic curve over a 159-bit finite field with embedding degree 6, resulting in 79-bit brute force and 953-bit finite field MOV [32] security levels. The reported timings pertain to the `c159` curve from Lynn's PBC [31] library running on top of Gnu GMP [24], on a G4 model CPU clocked at 1.25GHz with 512MB of RAM.

Method	General	Fixed base	Time	Storage
Double-and-add exponentiation	✓		5.92 ms	—
Sliding windows ($m = 4$) on the fly	✓		5.13 ms	—
Sliding windows ($m = 8$) from cache		✓	4.29 ms	255 elts.
Window-ladder pre-computation ($w = 5$)		✓	**0.74** ms	1024 elts.

Similar ideas can be used to speed up the verification process. Recall from Section 4.1 that signature verification amounts to testing whether $e(\sigma, u\, g_2^m\, v^r) = z$. Since g_2 and v (as well as u) are fixed for each signatory, the window-ladder pre-computation trick can be used exactly as in the signing process. The main difference is that the verifier needs one set of pre-computations for each new signer public key it encounters, whereas the signer only has one private key to contend with. In practice, pre-computation for verification might require a queuing strategy to delay the pre-computation investment for a particular public key until there is evidence that it might pay off.

More realistically, pre-computations will only be used for signature and not for verification. In that case, the verification expression $u\, g_2^m\, v^r$ should be evaluated as a one multi-exponentiation rather than two exponentiations, with only one sequence of repeated doublings instead of two. Verification time will then amount to slightly more than the time of one exponentiation and one pairing, versus two pairings (or one multi-pairing [26]) for BLS verification.

5 Shorter Signatures With Random Oracles

For completeness we show that the weakly secure signature scheme of Section 4.3 can also be transformed into particularly efficient and fully secure short signatures in the random oracle model [4]. To do so, we show a general transformation from any existentially unforgeable signature scheme under a *weak* chosen message attack into an existentially unforgeable signature scheme under an *adaptive* chosen message attack (in the strong sense), in the random oracle model. This gives a very efficient short signature scheme based on q-SDH in the random oracle model. We analyze our construction using a method of Katz and Wang [29] which gives a very tight reduction to the security of the underlying signature. We note that a closely related system was independently proposed by Zhang et al. [48].

5.1 Strongly Secure Tight Conversion

Let (*KeyGen*, *Sign*, *Verify*) be an existentially unforgeable signature under a *weak* chosen message attack. We assume that the scheme signs messages in some finite set Σ and that the private keys are in some set Π. We need two hash functions $H_1 : \Pi \times \{0,1\}^\star \to \{0,1\}$ and $H_2 : \{0,1\} \times \{0,1\}^\star \to \Sigma$ that will be viewed as random oracles in the security analysis. The hash-signature scheme is as follows:

Key Generation: Same as *KeyGen*. The public key is *PK*; the secret key is $SK \in \Pi$.

Signing: Given a secret key *SK*, and given a message $M \in \{0,1\}^\star$, compute $b \leftarrow H_1(SK, M) \in \{0,1\}$ and $m \leftarrow H_2(b, M) \in \Sigma$. Output the signature $(b, \mathit{Sign}(m))$. Note that signatures are one bit longer than in the underlying signature scheme.

Verification: Given a public key *PK*, a message $M \in \{0,1\}^\star$, and a signature (b, σ), output `valid` if $\mathit{Verify}(PK, H_2(b, M), \sigma) = \texttt{valid}$.

Theorem 11 below proves security of the scheme by leveraging the weak scheme from Section 4.3. The security reduction in Theorem 11 is tight and generic, in the sense that an attacker on the hash-signature scheme with success probability ϵ is converted to an attacker on the underlying signature with success probability approximately $\epsilon/2$.

Theorem 11. *Suppose* $(\mathit{KeyGen}, \mathit{Sign}, \mathit{Verify})$ *is* (t', q'_S, ϵ')*-existentially unforgeable under a weak chosen message attack. Then the corresponding hash-signature scheme is* (t, q_S, q_H, ϵ)*-secure against strong existential forgery under an adaptive chosen message attack, in the random oracle model, whenever* $q_S + q_H \leq q'_S$*, and for all* t *and* ϵ *satisfying*

$$\epsilon \geq 2\epsilon' / (1 - \frac{q'_S}{|\Sigma|}) \approx 2\epsilon' \qquad \textit{and} \qquad t \leq t' - o(t')$$

Proof. Assume $\mathcal{A}$ is a forger that (t, q_S, q_H, ϵ)-breaks the hash-signature scheme (in the random oracle model). We construct an algorithm $\mathcal{B}$ that interacts with $\mathcal{A}$ and (t', q'_S, ϵ')-breaks the underlying signature scheme. Algorithm $\mathcal{B}$ works as follows:

Setup: Algorithm $\mathcal{B}$ picks q'_S random and independent messages $m_1, \ldots, m_{q'_s}$ in Σ and sends them to the challenger. The challenger responds with a public key PK and signatures $\sigma_1, \ldots, \sigma_{q'_s}$ on $m_1, \ldots, m_{q'_s}$. Algorithm $\mathcal{B}$ gives PK to the adversary $\mathcal{A}$.

Hash queries: At any time, the forger $\mathcal{A}$ can query the hash functions H_1 and H_2. It can query these functions q_H times each. Since $\mathcal{B}$ can maintain tables to ensure that repeated queries are answered consistently, we assume without loss of generality that $\mathcal{A}$ never queries on the same input twice.

To respond to a query for $H_1(K, M)$ our algorithm $\mathcal{B}$ first checks if $K = SK$ by attempting to sign a random message using K. If the signature is valid then $\mathcal{B}$ outputs that message/signature pair as an existential forgery and terminates. Otherwise, $\mathcal{B}$ picks a random bit $b \in \{0, 1\}$ and tells $\mathcal{A}$ that $H_1(K, M) = b$.

To respond to a query for $H_2(c, M)$ our algorithm $\mathcal{B}$ maintains a list of tuples (M_i, b_i, i) called the H-list, and a counter ℓ which is initially set to 0. The H-list is initially empty. When responding to a query for $H_2(c, M)$ we set things up so that we know the signature on either $H_2(0, M)$ or $H_2(1, M)$ but $\mathcal{A}$ will not know which one. More precisely, to respond to the query $H_2(c, M)$ the simulator $\mathcal{B}$ does the following:

1. If M does not appear as first component in any tuple in the H-list then pick a random bit $b \in \{0, 1\}$, set $\ell \leftarrow \ell + 1$, and add (M, b, ℓ) to the H-list.
2. Let then (M, b, j) be the entry on the H-list corresponding to M. If $b = c$, output $H_2(c, M) = m_j$ (for which we know that σ_j is a valid signature). Otherwise, pick a random message $m \in \Sigma$ and output $H_2(c, M) = m$. Observe that $j \leq q_S + q_H \leq q'_S$ (since ℓ is always less than this value) and hence m_j is well defined.

Signature queries: $\mathcal{A}$ can issue up to q_S signature queries. To respond to a signature query for $M \in \Sigma$, the simulator $\mathcal{B}$ first runs the algorithm for responding to a hash query for $H_2(0, M)$, which incidentally is why the total number of H_2 queries is $q_S + q_H$. Let (M, b, j) be the entry on the H-list corresponding to M. Algorithm $\mathcal{B}$ responds with (b, σ_j) as the signature on M. This is a valid signature on M since $H_2(b, M) = m_j$ and σ_j is a valid signature on m_j. Note that this defines $H_1(SK, M) = b$ even though $\mathcal{B}$ does not know SK.

Output: Eventually, suppose $\mathcal{A}$ returns a forgery, $(M_*, (b_*, \sigma_*))$, such that (b_*, σ_*) is a valid signature on M_* in the hash-signature scheme and $\mathcal{A}$ did not previously obtain (b_*, σ_*) from $\mathcal{B}$ in response to a signature query on M_*. It follows that σ_* is a valid signature in the underlying signature scheme for the message $m_* = H_2(b_*, M_*)$. If $m_* \in \{m_1, \ldots, m_{q'_s}\}$ then $\mathcal{B}$ reports failure and aborts. Otherwise, it outputs (m_*, σ_*) as the existential forgery for the underlying signature scheme.

Algorithm $\mathcal{B}$ simulates the random oracles and signature oracle perfectly for $\mathcal{A}$. Therefore $\mathcal{A}$ produces a valid forgery for the hash-signature scheme with probability at least ϵ. It remains to bound the probability that $m_* \in \{m_1, \ldots, m_{q'_s}\}$. Let (M_*, b, j) be the entry on the H-list corresponding to M_*. First, consider the case where $\mathcal{A}$ never issued a signature query for M_*. In this case the bit b is independent of $\mathcal{A}$'s view. Therefore, $\Pr[b_* = b] = 1/2$. Next, note that if $b_* = b$ then, by construction, $m_* = H_2(b_*, M_*) = m_j$ and therefore in this case $\mathcal{B}$ will fail. When $b_* \neq b$, by construction, $H_2(b_*, M_*)$ is chosen at random in Σ and therefore, in this case, $\mathcal{B}$ will fail with probability at most $q'_S/|\Sigma|$. Now, in the case where $\mathcal{A}$ did issue a signature query for M_*, we necessarily have $b_* \neq b$, otherwise $\mathcal{A}$'s forgery would be a replay of $\mathcal{B}$'s response. $\mathcal{B}$'s failure rate in

this case is thus also at most $q'_S/|\Sigma|$. Thus, in all cases, it follows that $\mathcal{B}$ succeeds with probability at least

$$\Pr[\text{success}(\mathcal{B})] \geq \frac{\epsilon}{2} \cdot (1 - \frac{q'_S}{|\Sigma|}) \geq \epsilon'$$

as required. □

We note that in the proof above H_1 can be replaced with a Pseudo Random Function (PRF) and does not need to be modeled as a random oracle. However, modeling H_2 as a random oracle appears to be unavoidable.

Concrete Short Hash-Signature Scheme with Random Oracles. Applying Theorem 11 to the weakly secure scheme of Section 4.3 gives an efficient short signature that is strongly existentially unforgeable under an *adaptive* chosen message attack in the random oracle model assuming the hardness of the $(q_S + q_H)$-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$. For a public key $(g_1, g_2, v = g_2^x, z)$ and a hash function $H : \{0,1\}^\star \to \mathbb{Z}_p$ a signature on a message m is defined as the group element $\sigma \leftarrow g_1^{1/(x+H(b,m))} \in \mathbb{G}_1$ concatenated with the bit $b \in \{0,1\}$. To verify the signature, one checks that $e(\sigma, v \cdot g_2^{H(b,m)}) = z = e(g_1, g_2)$. We see these signatures are essentially as short as BLS signatures, and can be verified in approximately half the time. As before, signature time is where our scheme really shines compared to BLS, being up to an order of magnitude faster with pre-computations on g_1, as discussed in Section 4.9.

We note that the random oracle (RO) construction relies on the hardness of the q-SDH problem for $q = q_S + q_H$, whereas the system of Section 4.1 in the standard model only required $q = q_S$. In practice, one often considers $q_S = 2^{40}$ and $q_H = 2^{80}$ as bounds on the attacker's capabilities. Hence, we see that the RO system relies on a stronger SDH assumption than the non-RO system. For the RO system, this q_H is sufficiently large compared to p that one has to take special care in choosing p to avoid the Brown-Gallant [12] and Cheon [14] generic algorithms for SDH discussed in Section 3.3. Alternatively, for the RO system one could choose a larger group size p that satisfies $p \approx q_H^3$, in accordance with the generic complexity lower bounds we give in Section 6. The resulting signatures are still shorter than the non-RO system, but longer than BLS signatures.

5.2 Full Domain Hash Conversion

Another method for converting a signature scheme secure under a weak chosen message attack into a scheme secure under an adaptive chosen message attack is to simply apply *Sign* and *Verify* to $H(M)$ rather than M. In other words, we hash $M \in \{0,1\}^\star$ using a full domain hash H prior to signing and verifying. Security in the random oracle model is shown using a similar argument to Coron's analysis [15] of the Full Domain Hash [5]. However, the resulting reduction is not tight: an attacker on this hash-then-sign signature with success probability ϵ yields an attacker on the underlying signature with success probability approximately ϵ/q_S. We note, however, that these proofs are set in the random oracle model and therefore it is not clear whether the efficiency of the security reduction is relevant to actual security in the real world. Therefore, since this full domain hash conversion is slightly simpler that the tight conversion of Theorem 11 it might be preferable to use it rather than the system of Section 5.1. When we apply the full domain hash to the weakly secure scheme of Section 4.3, we obtain a secure signature under an adaptive chosen message attack assuming that the $(q_S + q_H)$-SDH problem is hard in $(\mathbb{G}_1, \mathbb{G}_2)$. A signature is one element, namely $\sigma \leftarrow g_1^{1/(x+H(m))} \in \mathbb{G}_1$. As before, signature verification is twice as fast as in BLS signatures, and signing five to ten times faster.

As mentioned above, a similar scheme was independently proposed by Zhang et al. [48], with a different reduction: in the random oracle model, security of this full domain hash scheme can be proven under Mitsunari's et al. [35] slightly weaker complexity assumption, rather than SDH. That assumption amounts to pre-specifying the value c in the q-SDH instance instead of letting it be chosen by the adversary. However, the resulting security reduction is far less efficient.

6 Generic Security of the SDH Assumption

To provide more confidence in the SDH assumption we prove a lower bound on the computational complexity of the q-SDH problem for generic groups in the sense of Shoup [45]. We slightly extend the original model to account for the multiple groups and the bilinearity.

In the generic bilinear group model, elements of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ appear to be encoded as arbitrary unique strings, so that no property other than equality can be directly tested by the adversary. The representation may use random-looking strings, or even sequential integers where the i-th string that the adversary sees is represented by the number i. The adversary performs operations on group elements by interacting with various oracles: three oracles for the group operation in each of the three groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, two oracles for the homomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ and its inverse ψ^{-1}, and one oracle for the bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We remark that this model gives too much power to the adversary in bilinear groups where ψ or ψ^{-1} cannot be computed efficiently.

To represent and simulate the working of the oracles, we model the opaque encoding of the elements of $\mathbb{G}_1$ using an injective function $\xi_1 : \mathbb{Z}_p \to \{0,1\}^{\lceil \log_2 p \rceil}$, where p is the group order. Internally, the simulator represents the elements of $\mathbb{G}_1$ not as themselves but as their discrete logarithms relative to some arbitrary generator g_1. This is captured by the function ξ_1, which maps any integer $a \in \mathbb{Z}_p$ to the external string representation $\xi_1 \in \{0,1\}^{\lceil \log_2 p \rceil}$ of the element $g_1^a \in \mathbb{G}_1$. We similarly define a second function $\xi_2 : \mathbb{Z}_p \to \{0,1\}^{\lceil \log_2 p \rceil}$ to represent $\mathbb{G}_2$, and a third function $\xi_T : \mathbb{Z}_p \to \{0,1\}^{\lceil \log_2 p \rceil}$ to represent $\mathbb{G}_T$. The adversary communicates with the oracles using the string representation of the group elements exclusively. Note that the adversary is given $p = |\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T|$.

The following theorem establishes the unconditional hardness of the q-SDH problem in the generic bilinear group model.

Theorem 12. *Suppose $\mathcal{A}$ is an algorithm that solves the q-SDH problem in generic bilinear groups of order p, making at most q_G oracle queries for the group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, the homomorphisms ψ and ψ^{-1}, and the bilinear pairing e, all counted together. Suppose also that the integer $x \in \mathbb{Z}_p^\times$ and the encoding functions ξ_1, ξ_2, ξ_T are chosen at random. Then, the probability, ϵ, that $\mathcal{A}$ on input $(p, \xi_1(1), \xi_1(x), \ldots, \xi_1(x^q), \xi_2(1), \xi_2(x))$ outputs $(c, \xi_1(\frac{1}{x+c}))$ with $c \in \mathbb{Z}_p \setminus \{-x\}$,*

$$\epsilon = \Pr\left[\mathcal{A}^G\begin{pmatrix} p, & \xi_1(1), \xi_1(x), \ldots, \xi_1(x^q), \\ & \xi_2(1), \xi_2(x) \end{pmatrix} = \left(c, \xi_1(\frac{1}{x+c})\right)\right]$$

is bounded as

$$\epsilon \leq \frac{(q_G + q + 3)^2 (q+1)}{p-1}$$

Asymptotically we have, $\epsilon \leq O\left(\frac{q_G^2\, q + q^3}{p}\right)$.

Proof. Consider an algorithm $\mathcal{B}$ that plays the following game with $\mathcal{A}$.

$\mathcal{B}$ maintains three lists of pairs $L_1 = \{(F_{1,i}, \xi_{1,i}) : i = 1, \ldots, \tau_1\}$, $L_2 = \{(F_{2,i}, \xi_{2,i}) : i = 1, \ldots, \tau_2\}$, and $L_T = \{(F_{T,i}, \xi_{T,i}) : i = 1, \ldots, \tau_T\}$, such that, at step τ in the game, $\tau_1 + \tau_2 + \tau_T = \tau + q + 3$. The entries $F_{1,i}$ and $F_{2,i}$ will be univariate polynomials of degree $\leq q$ in $\mathbb{Z}_p[X]$, and the $F_{T,i}$ polynomials of degree $\leq 2q$ in $\mathbb{Z}_p[X]$. The entries $\xi_{1,i}$, $\xi_{2,i}$, $\xi_{T,i}$ will be all the strings given out to the adversary. The lists are initialized at step $\tau = 0$ by setting $\tau_1 = q + 1$, $\tau_2 = 2$, and $\tau_T = 0$, and assigning $F_{1,i} = X^{i-1}$ for $i = 1, \ldots, q+1$ and $F_{2,i} = X^{i-1}$ for $i = 1, 2$. The corresponding $\xi_{1,i}$ and $\xi_{2,i}$ are set to random distinct strings. All polynomials are stored as coefficients of powers of X.

We assume that $\mathcal{A}$ only makes oracle queries on strings previously obtained from $\mathcal{B}$, a rule that is easy for $\mathcal{B}$ to enforce. Hence, given any query string $\xi_{1,i}$, it is easy for $\mathcal{B}$ to determine its index i into the table L_1, and from there the corresponding polynomial $F_{1,i}$. If the same string appears multiple times in the list L_1, ties are broken arbitrarily. (The same applies to L_2 and L_T.)

To start the game, $\mathcal{B}$ provides $\mathcal{A}$ with the $q + 3$ strings $\xi_{1,1}, \ldots, \xi_{1,q+1}, \xi_{2,1}, \xi_{2,2}$ that correspond to the challenge SDH instance. $\mathcal{B}$ answers $\mathcal{A}$'s queries as follows:

Group operations: $\mathcal{A}$ may request a group operation in $\mathbb{G}_1$ as a multiplication or as a division. Before answering a $\mathbb{G}_1$ query, the simulator $\mathcal{B}$ starts by incrementing the τ_1 counter by one. $\mathcal{A}$ gives $\mathcal{B}$ two operands $\xi_{1,i}$, $\xi_{1,j}$ with $1 \leq i, j < \tau_1$, and a multiply/divide selection bit. To respond, $\mathcal{B}$ creates a polynomial $F_{1,\tau_1} \in \mathbb{Z}_p[x]$ which it sets to $F_{1,\tau_1} \leftarrow F_{1,i} + F_{1,j}$ for a multiplication or to $F_{1,\tau_1} \leftarrow F_{1,i} - F_{1,j}$ for a division. If the result is identical to an earlier polynomial $F_{1,l}$ for some $l < \tau_1$, the simulator $\mathcal{B}$ duplicates its string representation: $\xi_{1,\tau_1} \leftarrow \xi_{1,l}$; otherwise, it lets ξ_{1,τ_1} be a fresh random string in $\{0,1\}^{\lceil \log_2 p \rceil}$ distinct from $\xi_{1,1}, \ldots, \xi_{1,\tau_1 - 1}$. The simulator appends the pair $(F_{1,\tau_1}, \xi_{1,\tau_1})$ to the list L_1 and gives the string ξ_{1,τ_1} to $\mathcal{A}$.

Group operation queries in $\mathbb{G}_2$ and $\mathbb{G}_T$ are answered in a similar manner, based on the lists L_2 and L_T respectively.

Homomorphisms: To answer a homomorphism query from $\mathbb{G}_2$ to $\mathbb{G}_1$, the simulator starts by incrementing the τ_1 counter by one. $\mathcal{A}$ gives to $\mathcal{B}$ a string operand $\xi_{2,i}$ with $1 \leq i < \tau_2$. To respond, $\mathcal{B}$ makes a copy of the associated L_2 polynomial into L_1: it sets $F_{1,\tau_1} \leftarrow F_{2,i}$. If L_1 already contained a copy of the polynomial, i.e., $F_{1,\tau_1} = F_{1,l}$ for some $l < \tau_1$, then $\mathcal{B}$ duplicates its existing string representation: $\xi_{1,\tau_1} \leftarrow \xi_{1,l}$; otherwise, it sets ξ_{1,τ_1} to a random string in $\{0,1\}^{\lceil \log_2 p \rceil} \setminus \{\xi_{1,1}, \ldots, \xi_{1,\tau_1 - 1}\}$. The pair $(F_{1,\tau_1}, \xi_{1,\tau_1})$ is added to the list L_1, and the string ξ_{1,τ_1} is given to $\mathcal{A}$ as answer to the query.

Inverse homomorphism queries from $\mathbb{G}_1$ to $\mathbb{G}_2$ are answered similarly. Note that in this case the counter τ_2 is to be incremented, and a string from L_2 is to be returned.

Pairing: A pairing query consists of two operands $\xi_{1,i}$ and $\xi_{2,j}$ with $1 \leq i \leq \tau_1$ and $1 \leq j \leq \tau_2$ for the current values of τ_1 and τ_2. Upon receipt of such a query from $\mathcal{A}$, the counter τ_T is incremented. The simulator then computes the product of polynomials $F_{T,\tau_T} \leftarrow F_{1,i} \cdot F_{2,j}$. The result is a polynomial of degree at most $2q$ in $\mathbb{Z}_p[X]$. If the same polynomial was already present in L_T, i.e., if $F_{T,\tau_T} = F_{T,l}$ for some $l < \tau_T$, then $\mathcal{B}$ simply clones the associated string: $\xi_{T,\tau_T} \leftarrow \xi_{T,l}$; otherwise, it sets ξ_{T,τ_T} to a new random string in $\{0,1\}^{\lceil \log_2 p \rceil} \setminus \{\xi_{T,1}, \ldots, \xi_{T,\tau_T - 1}\}$. The simulator then adds the pair $(F_{T,\tau_T}, \xi_{T,\tau_T})$ to the list L_T, and gives the string ξ_{T,τ_T} to $\mathcal{A}$.

Note that $\mathcal{A}$ can implement exponentiation generically using $O(\log p)$ calls to the group operation oracles, so we need not provide an exponentiation oracle. Similarly, $\mathcal{A}$ can obtain the identity element in each group by requesting the division of any element into itself. Observe also that the

following invariant is preserved throughout the game, where τ is the total number of oracle queries that have been answered at any given time:

$$\tau_1 + \tau_2 + \tau_T = \tau + q + 3 \tag{3}$$

When $\mathcal{A}$ terminates it returns a pair $(c,\ \xi_{1,\ell})$ where $c \in \mathbb{Z}_p$ and $1 \leq \ell \leq \tau_1$. Let $F_{1,\ell}$ be the corresponding polynomial in the list L_1. In order to exhibit the correctness of $\mathcal{A}$'s answer within the simulation framework, $\mathcal{B}$ computes the polynomial $F_{T,\star} = F_{1,\ell} \cdot (F_{2,2} + c \cdot F_{2,1}) = F_{1,\ell} \cdot (X + c)$. Notice that if $\mathcal{A}$'s answer is correct for a particular secret SDH exponent $x \in \mathbb{Z}_p$, then for $X = x$ we must necessarily have

$$F_{T,\star}(X) = 1 \tag{4}$$

Indeed, this equality corresponds to the DDH relation "$e(A, g_2^x g_2^c) = e(g_1, g_2)$" where A denotes the element of $\mathbb{G}_1$ represented by $\xi_{1,\ell}$. Observe that since the constant monomial "1" has degree 0 and $F_{T,\star} = F_{1,\ell} \cdot (X + c)$ where $(X + c)$ has degree 1, the above relation (4) cannot be satisfied identically in $\mathbb{Z}_p[X]$ unless $F_{1,\ell}$ has degree $\geq p - 2$. We know that the degree of $F_{1,\ell}$ is at most q, therefore we deduce that there exists an assignment in $\mathbb{Z}_p$ to the variable X for which Equation (4) does not hold. Since Equation (4) is thus a non-trivial polynomial equation of degree $\leq q + 1$, it admits at most $q + 1$ roots in $\mathbb{Z}_p$.

At this point, $\mathcal{B}$ chooses a random $x \in \mathbb{Z}_p^\times$ as the secret SDH exponent, and evaluates all the polynomials under the assignment $X \leftarrow x$. If the assignment causes two non-identical polynomials within either of the lists L_1, L_2, and L_T to assume the same value, then the simulation provided by $\mathcal{B}$ to $\mathcal{A}$ was flawed since it presented as distinct two group elements that were in fact equal. If it causes the non-trivial Equation (4) to be satisfied, then the adversary has won the game. However, if no non-trivial equality emerged from the assignment, then $\mathcal{B}$'s simulation was perfect and nonetheless resulted in $\mathcal{A}$'s failure to solve the instance it was given.

By the above argument, the success probability of $\mathcal{A}$ in the generic model is bounded by the probability that at least one equality among the following collections is satisfied, for random $x \in \mathbb{Z}_p^\times$:

1. $F_{1,i}(x) = F_{1,j}(x)$ in $\mathbb{Z}_p$ — for some i, j such that $F_{1,i} \neq F_{1,j}$ in $\mathbb{Z}_p[X]$,
2. $F_{2,i}(x) = F_{2,j}(x)$ in $\mathbb{Z}_p$ — for some i, j such that $F_{2,i} \neq F_{2,j}$ in $\mathbb{Z}_p[X]$,
3. $F_{T,i}(x) = F_{T,j}(x)$ in $\mathbb{Z}_p$ — for some i, j such that $F_{T,i} \neq F_{T,j}$ in $\mathbb{Z}_p[X]$,
4. $F_{1,\ell}(x) \cdot (x + c) = 1$ in $\mathbb{Z}_p$.

Since each non-trivial polynomial $F_{1,i} - F_{1,j}$ has degree at most q, it vanishes at a random $x \in \mathbb{Z}_p^\times$ with probability at most $q/(p-1)$. In a similar way, each non-trivial polynomial $F_{2,i} - F_{2,j}$ vanishes with probability $\leq q/(p-1)$, and $F_{T,i} - F_{T,j}$ with probability $\leq 2q/(p-1)$ since polynomials in L_T can have degree up to $2q$. The last equality holds with probability $\leq (q+1)/(p-1)$, as already shown. Summing over all valid pairs (i, j) in all four cases, we deduce that $\mathcal{A}$ wins the game with probability

$$\epsilon \leq \binom{\tau_1}{2} \frac{q}{p-1} + \binom{\tau_2}{2} \frac{q}{p-1} + \binom{\tau_T}{2} \frac{2q}{p-1} + \frac{q+1}{p-1}$$

It follows from Equation (3) that the game ended with $\tau_1 + \tau_2 + \tau_T \leq q_G + q + 3$, and we obtain: $\epsilon \leq (q_G + q + 3)^2 q/(p-1) = O(q_G^2\, q/p + q^3/p)$. □

The following corollary restates in a simpler way the asymptotic hardness of the SDH assumption against generic attacks.

Corollary 13. *Any adversary that solves the q-SDH problem with constant probability $\epsilon > 0$ in generic bilinear groups of order p such that $q < O(\sqrt[3]{p})$ requires $\Omega(\sqrt{\epsilon p/q})$ generic operations.*

7 Conclusion

We presented a number of short signature schemes based on the SDH assumption in bilinear groups. Our main result is a short signature which is fully secure without random oracles or hash functions. The signature is very efficient and as short as DSA signatures, but is provably secure in the standard model under a tight security reduction. We also described a number of useful extensions to our scheme, such as limited message recovery for even greater compactness, off-line/on-line operation for instantaneous signing, and a random oracle signature that is as compact as and much more efficient than the BLS scheme.

These constructions are possible thanks to properties of the Strong Diffie-Hellman assumption, which we introduced, motivated, and proved secure in the generic group model. The Strong DH assumption can be regarded as a discrete logarithm analogue of the Strong RSA assumption. We hope that the SDH assumption will establish itself as a useful tool for constructing cryptographic systems.

Acknowledgments

We thank Mihir Bellare, Hovav Shacham, and Nigel Smart for their helpful comments on this paper.

References

[1] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In *Advances in Cryptology—EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 83–107. Springer-Verlag, 2002.

[2] Paulo S. L. M. Barreto, Steven Galbraith, Colm O'hEigeartaigh, and Michael Scott. Efficient pairing computation on supersingular abelian varieties. Cryptology ePrint Archive, Report 2004/375, 2004. `http://eprint.iacr.org/`.

[3] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. Cryptology ePrint Archive, Report 2005/133, 2005. `http://eprint.iacr.org/`.

[4] Mihir Bellare and Phil Rogaway. Random oracle are practical: A paradigm for designing efficient protocols. In *Proceedings of ACM CCS 1993*, pages 62–73. ACM Press, 1993.

[5] Mihir Bellare and Phil Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In *Advances in Cryptology—EUROCRYPT 1996*, volume 1070 of *LNCS*, pages 399–416. Springer-Verlag, 1996.

[6] Mihir Bellare and Phil Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In *Advances in Cryptology—CRYPTO 1997*, volume 1294 of *LNCS*, pages 470–84. Springer-Verlag, 1997.

[7] Ian Blake, Gadiel Seroussi, and Nigel Smart. *Elliptic Curves in Cryptography*, volume 265 of *London Mathematical Society Lecture Notes*. Cambridge University Press, 1999.

[8] Dan Boneh and Xavier Boyen. Efficient selective-ID identity based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–38. Springer-Verlag, 2004.

[9] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer-Verlag, 2004.

[10] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer-Verlag, 2004.

[11] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004. Extended abstract in Proceedings of Asiacrypt 2001, LNCS volume 2248.

[12] Daniel Brown and Robert Gallant. The static Diffie-Hellman problem. Cryptology ePrint Archive, Report 2004/306, 2004. `http://eprint.iacr.org/`.

[13] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–22. Springer-Verlag, 2004.

[14] Jung Hee Cheon. Security analysis of the strong diffie-hellman problem. In *Advances in Cryptology—EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–13. Springer-Verlag, 2006.

[15] Jean-Sébastien Coron. On the exact security of full domain hash. In *Advances in Cryptology—CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–35. Springer-Verlag, 2000.

[16] Jean-Sébastien Coron and David Naccache. Security analysis of the Gennaro-Halevi-Rabin signature scheme. In *Advances in Cryptology—EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 91–101. Springer-Verlag, 2000.

[17] Nicolas Courtois, Magnus Daum, and Patrick Felke. On the security of HFE, HFEv- and Quartz. In *Proceedings of PKC 2003*, volume 2567 of *LNCS*, pages 337–50. Springer-Verlag, 2003.

[18] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. *ACM TISSEC*, 3(3):161–85, 2000. Extended abstract in Proceedings of ACM CCS, ACM Press, 1999.

[19] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *Proceedings of PKC 2005*, volume 3386 of *LNCS*, pages 416–31. Springer-Verlag, 2005.

[20] Marc Fischlin. The Cramer-Shoup strong-RSA signature scheme revisited. In *Proceedings of PKC 2003*, volume 2567 of *LNCS*, pages 116–29. Springer-Verlag, 2003.

[21] Steven Galbraith. Pairings. In Ian F. Blake, Gadiel Seroussi, and Nigel Smart, editors, *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Notes*, chapter IX, pages 183–213. Cambridge University Press, 2005.

[22] Steven Galbraith, Kenneth Paterson, and Nigel Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. `http://eprint.iacr.org/`.

[23] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology—EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 123–39. Springer-Verlag, 1999.

[24] GMP Project. The GnuMP multiprecision arithmetic library. `http://www.swox.com/gmp/`.

[25] Shafi Goldwasser, Silvio Micali, and Ron Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[26] Robert Granger and Nigel Smart. On computing products of pairings. Cryptology ePrint Archive, Report 2006/172, 2006. `http://eprint.iacr.org/`.

[27] Florian Hess, Nigel P. Smart, and Frederik Vercauteren. The Eta pairing revisited. Cryptology ePrint Archive, Report 2006/110, 2006. `http://eprint.iacr.org/`.

[28] Antoine Joux and Kim Nguyen. Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups. *Journal of Cryptology*, 16(4):239–47, 2003.

[29] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In *Proceedings of ACM CCS 2003*, pages 155–64. ACM Press, 2003.

[30] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *Proceedings of NDSS 2000*. Internet Society, 2000.

[31] Ben Lynn. The PBC pairing-based cryptography library. `http://rooster.stanford.edu/~ben/pbc/`.

[32] Alfred Menezes, Tatsuaki Okamoto, and Scott Vanstone. Reducing elliptic curve logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–46, 1993.

[33] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[34] Victor Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4):235–61, 2004.

[35] Shigeo Mitsunari, Ryuichi Sakai, and Masao Kasahara. A new traitor tracing. *IEICE Trans. Fundamentals*, E85-A(2):481–84, 2002.

[36] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Trans. Fundamentals*, E84-A(5):1234–43, 2001.

[37] David Naccache and Jacques Stern. Signing on a postcard. In *Proceedings of Financial Cryptography—FC 2000*, volume 1962 of *LNCS*, pages 121–35. Springer-Verlag, 2000.

[38] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of ACM STOC 1989*, pages 33–43. ACM Press, 1989.

[39] Jacques Patarin, Nicolas Courtois, and Louis Goubin. QUARTZ, 128-bit long digital signatures. In *Proceedings of CT-RSA 2001*, volume 2020 of *LNCS*, pages 282–97. Springer-Verlag, 2001.

[40] Kenneth Paterson. Cryptography from pairings. In Ian F. Blake, Gadiel Seroussi, and Nigel Smart, editors, *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Notes*, chapter X, pages 215–51. Cambridge University Press, 2005.

[41] Leon Pintsov and Scott Vanstone. Postal revenue collection in the digital age. In *Proceedings of Financial Cryptography—FC 2000*, volume 1962 of *LNCS*, pages 105–20. Springer-Verlag, 2000.

[42] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proceedings of IEEE FOCS 1999*. IEEE Press, 1999.

[43] Hovav Shacham. Implementing pairing-based signature schemes. Presentation at the Pairings in Cryptography workshop—PiC 2005. Dublin, Ireland, 2005.

[44] Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In *Advances in Cryptology—CRYPTO 2001*, volume 2139 of *LNCS*, pages 355–67. Springer-Verlag, 2001.

[45] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology—EUROCRYPT 1997*, volume 1233 of *LNCS*, pages 256–66. Springer-Verlag, 1997.

[46] Victor Shoup. A composition theorem for universal one-way hash functions. In *Advances in Cryptology—EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 445–52. Springer-Verlag, 2000.

[47] Vu Dong Tô, Reihaneh Safavi-Naini, and Fangguo Zhang. New traitor tracing schemes using bilinear map. In *Proceedings of DRM Workshop*, 2003.

[48] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An efficient signature scheme from bilinear pairings and its applications. In *Proceedings of PKC 2004*, volume 2947 of *LNCS*, pages 277–90. Springer-Verlag, 2004.