# Short Signatures from the Weil Pairing*

Dan Boneh†
dabo@cs.stanford.edu

Ben Lynn
blynn@cs.stanford.edu

Hovav Shacham
hovav@cs.stanford.edu

**Abstract**

We introduce a short signature scheme based on the Computational Diffie-Hellman assumption on certain elliptic and hyper-elliptic curves. For standard security parameters, the signature length is half that of a DSA signature with a similar level of security. Our short signature scheme is designed for systems where signatures are typed in by a human or are sent over a low-bandwidth channel. We survey a number of properties of our signature scheme such as signature aggregation and batch verification.

## 1 Introduction

Short digital signatures are needed in environments with strong bandwidth constraints. For example, product registration systems often ask users to key in a signature provided on a CD label. When a human is asked to type in a digital signature, the shortest possible signature is needed. Similarly, due to space constraints, short signatures are needed when one prints a bar-coded digital signature on a postage stamp [41, 37]. As a third example, consider legacy protocols that allocate a fixed short field for non-repudiation [1, 25]. One would like to use the most secure signature that fits in the alloted field length.

The two most frequently used signatures schemes, RSA and DSA, produce relatively long signatures compared to the security they provide. For example, when one uses a 1024-bit modulus, RSA signatures are 1024 bits long. Similarly, when one uses a 1024-bit modulus, standard DSA signatures are 320 bits long. Elliptic curve variants of DSA, such as ECDSA, are also 320 bits long [2]. A 320-bit signature is too long to be keyed in by a human.

We propose a signature scheme whose length is approximately 170 bits and which provides a level of security similar to that of 320-bit DSA signatures. Our signature scheme is secure against existential forgery under a chosen-message attack (in the random oracle model) assuming the Computational Diffie-Hellman problem (CDH) is hard on certain elliptic curves over a finite field. Generating a signature is a simple multiplication on the curve. Verifying the signature is done using a bilinear pairing on the curve. Our signature scheme inherently uses properties of curves. Consequently, there is no equivalent of our scheme in $\mathbb{F}_p^*$.

Constructing short signatures is an old problem. Several proposals show how to shorten DSA while preserving the same level of security. Naccache and Stern [37] propose a variant of DSA where the signature length is approximately 240 bits. Mironov [35] suggests a DSA variant with a similar length and gives a concrete security analysis of the construction (in the random oracle model). Another technique proposed for reducing DSA signature length is signatures with message recovery [38, 41]. In such systems one encodes a part of the message into the signature thus

---

*This is the full version of a paper that appeared in Asiacrypt '01 [12].

†Supported by NSF and the Packard Foundation.

shortening the total length of the message-signature pair. For long messages, one can then achieve a DSA signature overhead of 160 bits. However, for very short messages (e.g., 64 bits) the total length remains 320 bits. Using our signature scheme, the signature length is always on the order of 160 bits, however short the message. When the message is not transmitted along with the signature, DSA signatures with message recovery are just as long as standard DSA signatures. We also note that Patarin *et al.* [40] construct short signatures whose security depends on the Hidden Field Equation (HFE) problem.

Our signature scheme uses groups where the CDH problem is hard, but the Decision Diffie-Hellman problem (DDH) is easy. The first example of such groups was given in [27] and was previously used in [26, 10]. We call such groups Gap Diffie-Hellman groups, or GDH groups for short. We show how to construct a signature scheme from GDH groups; prove security of the scheme; and show how to build GDH groups that lead to short signatures. The signature scheme resembles the undeniable signature scheme of Chaum and Pederson [13]. Our signature scheme has several useful properties, described in Section 5. For example, signatures generated by different people on different messages can be aggregated into a single signature [11]. The signature also supports standard extensions such as threshold signatures and blind signatures [9].

**Notation.** We use $E/\mathbb{F}_q$ to denote an elliptic curve $y^2 = x^3 + ax + b$ with coefficients $a, b \in \mathbb{F}_q$. For $r \geq 1$, we use $E(\mathbb{F}_{q^r})$ to denote the group of points on $E$ in $\mathbb{F}_{q^r}$. We use $|E(\mathbb{F}_{q^r})|$ to denote the number of points in $E(\mathbb{F}_{q^r})$.

## 2 Gap Diffie-Hellman groups and bilinear maps

Before presenting the signature scheme, we first review a few concepts related to bilinear maps and Gap Diffie-Hellman groups. We use the following notation:

1. $G_1$ and $G_2$ are two (multiplicative) cyclic groups of prime order $p$;
2. $g_1$ is a generator of $G_1$ and $g_2$ is a generator of $G_2$;
3. $\psi$ is an isomorphism from $G_2$ to $G_1$, with $\psi(g_2) = g_1$; and
4. $e$ is a bilinear map $e : G_1 \times G_2 \to G_T$.

One can set $G_1 = G_2$, but we allow for the more general case where $G_1 \neq G_2$ so that we can take advantage of certain families of non-supersingular elliptic curves as described in Section 4.3.

The proofs of security require an efficiently computable isomorphism $\psi : G_2 \to G_1$. When $G_1 = G_2$ and $g_1 = g_2$ one could take $\psi$ to be the identity map. When $G_1 \neq G_2$ we will need to describe explicitly an efficiently computable isomorphism $\psi : G_2 \to G_1$. The map $\psi$ is essential for security. To illustrate this, we give in the next section an example of a bilinear map that engenders an insecure signature scheme precisely because $\psi$ does not exist.

With this setup we obtain natural generalizations of the CDH and DDH problems:

**Computational co-Diffie-Hellman (co-CDH) on** $(G_1, G_2)$**:** Given $g_2, g_2^a \in G_2$ and $h \in G_1$ compute $h^a \in G_1$.

**Decision co-Diffie-Hellman (co-DDH) on** $(G_1, G_2)$**:** Given $g_2, g_2^a \in G_2$ and $h, h^b \in G_1$ output `yes` if $a = b$ and `no` otherwise. When the answer is `yes` we say that $(g_2, g_2^a, h, h^a)$ is a co-Diffie-Hellman tuple.

When $G_1 = G_2$ these problems reduce to standard CDH and DDH.

Next we a define a co-GDH gap group pair to be a pair of groups $(G_1, G_2)$ on which co-DDH is easy but co-CDH is hard. We define the advantage of an algorithm $\mathcal{A}$ in solving the Computational co-Diffie-Hellman problem on $(G_1, G_2)$ as

$$\mathsf{Adv\ co\text{-}CDH}_{\mathcal{A}} \stackrel{\text{def}}{=} \Pr\left[\mathcal{A}(g_2, g_2^a, h) = h^a : a \stackrel{\text{R}}{\leftarrow} \mathbb{Z}_p, h \stackrel{\text{R}}{\leftarrow} G_1\right].$$

In other words, the probability is over the uniform random choice of $a$ from $\mathbb{Z}_p$ and $h$ from $G_1$, and over the coin tosses of $\mathcal{A}$. We say that an algorithm $\mathcal{A}$ $(t, \epsilon)$-breaks Computational co-Diffie-Hellman on $(G_1, G_2)$ if $\mathcal{A}$ runs in time at most $t$, and $\mathsf{Adv\ co\text{-}CDH}_{\mathcal{A}}$ is at least $\epsilon$.

**Definition 2.1.** Two groups $(G_1, G_2)$ are a $(t, \epsilon)$-Gap co-Diffie-Hellman pair (co-GDH pair) if they satisfy the following properties:

1. The group action on both $G_1$ and $G_2$ and the map $\psi$ from $G_2$ to $G_1$ can be computed in one time unit.
2. The Decision co-Diffie-Hellman problem on $(G_1, G_2)$ can be solved in one time unit.
3. No algorithm $(t, \epsilon)$-breaks Computational co-Diffie-Hellman on $(G_1, G_2)$.

When $(G_1, G_1)$ is a $(t, \epsilon)$ co-GDH pair we say $G_1$ is a $(t, \epsilon)$-Gap-Diffie-Hellman group (GDH group).

Note that in the above definition we are normalizing time so that all the above algorithms take one time unit, and under this normalization there is no algorithm that $(t, \epsilon)$-breaks CDH on $(G_1, G_2)$.

## 2.1 Bilinear maps

Currently, the only examples of Gap Diffie-Hellman groups arise from bilinear maps [27]. We briefly define bilinear groups and show how they give GDH groups. It is possible that other constructions for Gap Diffie-Hellman groups exist.

Let $G_1$ and $G_2$ be two groups as above, with an additional group $G_T$ such that $|G_1| = |G_2| = |G_T|$. A bilinear map is a map $e : G_1 \times G_2 \to G_T$ with the following properties:

1. Bilinear: for all $u \in G_1, v \in G_2$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degenerate: $e(g_1, g_2) \neq 1$.

**Definition 2.2.** Two order-$p$ groups $(G_1, G_2)$ are a $(t, \epsilon)$-bilinear group pair if they satisfy the following properties:

1. The group action on both $G_1$ and $G_2$ and the map $\psi$ from $G_2$ to $G_1$ can be computed in one time unit.
2. A group $G_T$ of order $p$ and a bilinear map $e : G_1 \times G_2 \to G_T$ exist, and $e$ is computable in one time unit.
3. No algorithm $(t, \epsilon)$-breaks Computational co-Diffie-Hellman on $(G_1, G_2)$.

Joux and Nguyen [26] showed that an efficiently-computable bilinear map $e$ provides an algorithm for solving the Decision co-Diffie-Hellman problem as follows: For a tuple $(g_2, g_2^a, h, h^b)$ where $h \in G_1$ we have

$$a = b \bmod p \quad \Longleftrightarrow \quad e(h, g_2^a) = e(h^b, g_2).$$

Consequently, if two groups $(G_1, G_2)$ are a $(t, \epsilon)$-bilinear group pair, then they are also a $(t/2, \epsilon)$-co-GDH group pair. The converse is probably not true.

# 3 Signature schemes based on Gap Diffie-Hellman groups

We present a signature scheme that works in any Gap co-Diffie-Hellman group pair $(G_1, G_2)$. We prove security of the scheme and, in the next section, show how it leads to short signatures. The scheme resembles the undeniable signature scheme proposed by Chaum and Pederson [13]. Okamoto and Pointcheval [39] briefly note that gap problems can give rise to signature schemes. However, most gap problems will not lead to short signatures.

Let $(G_1, G_2)$ be $(t, \epsilon)$ co-Gap Diffie-Hellman pair where $|G_1| = |G_2| = p$. A signature $\sigma$ is an element of $G_1$. The signature scheme comprises three algorithms, *KeyGen*, *Sign*, and *Verify*. It makes use of a full-domain hash function $H : \{0,1\}^* \to G_1$. The security analysis views $H$ as a random oracle [7]. In Section 3.2 we weaken the requirement on the hash function $H$.

**Key generation.** Pick random $x \stackrel{\text{R}}{\leftarrow} \mathbb{Z}_p$, and compute $v \leftarrow g_2^x$. The public key is $v \in G_2$. The secret key is $x$.

**Signing.** Given a secret key $x \in \mathbb{Z}_p$, and a message $M \in \{0,1\}^*$, Compute $h \leftarrow H(M) \in G_1$, and $\sigma \leftarrow h^x$. The signature is $\sigma \in G_1$.

**Verification.** Given a public key $v \in G_2$, a message $M \in \{0,1\}^*$, and a signature $\sigma \in G_1$, compute $h \leftarrow H(M) \in G_1$ and verify that $(g_2, v, h, \sigma)$ is a valid co-Diffie-Hellman tuple. If so, output `valid`; if not, output `invalid`.

A signature is a single element of $G_1$. To construct short signatures, therefore, we need co-GDH pairs where elements in $G_1$ have a short representation. We construct such groups in Section 4.

## 3.1 Security

We prove the security of the Signature Scheme against existential forgery under a chosen-message attacks in the random oracle model. Existential unforgeability under a chosen message attack [24] for a signature scheme (*KeyGen*, *Sign*, and *Verify*) 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 private key *SK*. The adversary $\mathcal{A}$ is given *PK*.

**Queries.** Proceeding adaptively, $\mathcal{A}$ requests signatures with *PK* on at most $q_S$ messages of his choice $M_1, \ldots, M_{q_s} \in \{0,1\}^*$. The challenger responds to each query with a signature $\sigma_i = \mathit{Sign}(SK, M_i)$.

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

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

**Definition 3.1.** A forger $\mathcal{A}$ $(t, q_S, q_H, \epsilon)$-breaks a signature scheme if $\mathcal{A}$ runs in time at most $t$; $\mathcal{A}$ makes at most $q_S$ signature queries and at most $q_H$ queries to the hash function; and $\mathsf{Adv\,Sig}_{\mathcal{A}}$ is at least $\epsilon$. A signature scheme is $(t, q_S, q_H, \epsilon)$-existentially unforgeable under an adaptive chosen-message attack if no forger $(t, q_S, q_H, \epsilon)$-breaks it.

The following theorem shows that the signature scheme is secure.

**Theorem 3.2.** *Let $(G_1, G_2)$ be a $(t', \epsilon')$-co-GDH group pair of order $p$. Then the signature scheme on $(G_1, G_2)$ is $(t, q_S, q_H, \epsilon)$-secure against existential forgery under an adaptive chosen-message attack (in the random oracle model) for all $t$ and $\epsilon$ satisfying*

$$\epsilon \geq e(q_S + 1) \cdot \epsilon' \qquad \textit{and} \qquad t \leq t' - c_{G_1}(q_H + 2q_S),$$

*and $c_{G_1}$ is a constant that depends on $G_1$. Here $e$ is the base of the natural logarithm.*

Hence, security of the signature scheme follows from the hardness of co-CDH on $(G_1, G_2)$. When $G_1 = G_2$ security is based on the standard Computational Diffie-Hellman assumption in $G_1$.

**Proof of Theorem 3.2.** Suppose $\mathcal{A}$ is a forger algorithm that $(t, q_S, q_H, \epsilon)$-breaks the signature scheme. We show how to construct a $t'$-time algorithm $\mathcal{B}$ that solves co-CDH in $(G_1, G_2)$ with probability at least $\epsilon'$. This will contradict the fact that $(G_1, G_2)$ are a $(t', \epsilon')$-co-GDH group pair.

Let $g_2$ be a generator of $G_2$. Algorithm $\mathcal{B}$ is given $g_2, u \in G_2$ and $h \in G_1$, where $u = g_2^a$. Its goal is to output $h^a \in G_1$. Algorithm $\mathcal{B}$ simulates the challenger and interacts with forger $\mathcal{A}$ as follows.

**Setup.** Algorithm $\mathcal{B}$ starts by giving $\mathcal{A}$ the generator $g_2$ and the public key $u \cdot g_2^r \in G_2$, where $r$ is random in $\mathbb{Z}_p$.

***H*-queries.** At any time algorithm $\mathcal{A}$ can query the random oracle $H$. To respond to these queries algorithm $\mathcal{B}$ maintains a list of tuples $\langle M_j, w_j, b_j, c_j \rangle$ as explained below. We refer to this list as the $H$-list. The list is initially empty. When $\mathcal{A}$ queries the oracle $H$ at a point $M_i \in \{0,1\}^*$, algorithm $\mathcal{B}$ responds as follows:

1. If the query $M_i$ already appears on the $H$-list in a tuple $\langle M_i, w_i, b_i, c_i \rangle$ then algorithm $\mathcal{B}$ responds with $H(M_i) = w_i \in G_1$.
2. Otherwise, $\mathcal{B}$ generates a random coin $c_i \in \{0,1\}$ so that $\Pr[c_i = 0] = 1/(q_S + 1)$.
3. Algorithm $\mathcal{B}$ picks a random $b_i \in \mathbb{Z}_p$.
   If $c_i = 0$, $\mathcal{B}$ computes $w_i \leftarrow h \cdot \psi(g_2)^{b_i} \in G_1$. If $c_i = 1$, $\mathcal{B}$ computes $w_i \leftarrow \psi(g_2)^{b_i} \in G_1$.
4. Algorithm $\mathcal{B}$ adds the tuple $\langle M_i, w_i, b_i, c_i \rangle$ to the $H$-list and responds to $\mathcal{A}$ by setting $H(M_i) = w_i$.

Note that either way $w_i$ is uniform in $G_1$ and is independent of $\mathcal{A}$'s current view as required.

**Signature queries.** Let $M_i$ be a signature query issued by $\mathcal{A}$. Algorithm $\mathcal{B}$ responds to this query as follows:

1. Algorithm $\mathcal{B}$ runs the above algorithm for responding to $H$-queries to obtain a $w_i \in G_1$ such that $H(M_i) = w_i$. Let $\langle M_i, w_i, b_i, c_i \rangle$ be the corresponding tuple on the $H$-list. If $c_i = 0$ then $\mathcal{B}$ reports failure and terminates.
2. We know $c_i = 1$ and hence $w_i = \psi(g_2)^{b_i} \in G_1$. Define $\sigma_i = \psi(u)^{b_i} \cdot \psi(g_2)^{rb_i} \in G_1$. Observe that $\sigma_i = w_i^{a+r}$ and therefore $\sigma_i$ is a valid signature on $M_i$ under the public key $u \cdot g_2^r = g_2^{a+r}$. Algorithm $\mathcal{B}$ gives $\sigma_i$ to algorithm $\mathcal{A}$.

**Output.** Eventually algorithm $\mathcal{A}$ produces a message-signature pair $(M_f, \sigma_f)$ such that no signature query was issued for $M_f$. If there is no tuple on the $H$-list containing $M_f$ then $\mathcal{B}$ issues a query itself for $H(M_f)$ to ensure that such a tuple exists. We assume $\sigma_f$ is a valid signature on $M_f$ under the given public key; if it is not, $\mathcal{B}$ reports failure and terminates. Next, algorithm $\mathcal{B}$ finds the tuple $\langle M_f, w, b, c \rangle$ on the $H$-list. If $c = 1$ then $\mathcal{B}$ reports failure and terminates. Otherwise, $c = 0$ and therefore $H(M_f) = w = h \cdot \psi(g_2)^b$. Hence, $\sigma = h^{a+r} \cdot \psi(g_2)^{b(a+r)}$. Then $\mathcal{B}$ outputs the required $h^a$ as $h^a \leftarrow \sigma / (h^r \cdot \psi(u)^b \cdot \psi(g_2)^{rb})$.

This completes the description of algorithm $\mathcal{B}$. It remains to show that $\mathcal{B}$ solves the given instance of the co-CDH problem in $(G_1, G_2)$ with probability at least $\epsilon'$. To do so, we analyze the three events needed for $\mathcal{B}$ to succeed:

$\mathcal{E}_1$**:** $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}$'s signature queries.
$\mathcal{E}_2$**:** $\mathcal{A}$ generates a valid message-signature forgery $(M_f, \sigma_f)$.
$\mathcal{E}_3$**:** Event $\mathcal{E}_2$ and $c = 0$ for the tuple containing $M_f$ on the $H$-list.

$\mathcal{B}$ succeeds if all of these events happen. The probability $\Pr[\mathcal{E}_1 \wedge \mathcal{E}_3]$ decomposes as

$$\Pr[\mathcal{E}_1 \wedge \mathcal{E}_3] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \Pr[\mathcal{E}_3 \mid \mathcal{E}_1 \wedge \mathcal{E}_2]. \tag{1}$$

The following claims give a lower bound for each of these terms.

**Claim 1:** The probability that algorithm $\mathcal{B}$ does not abort as a result of $\mathcal{A}$'s signature queries is at least $1/e$. Hence, $\Pr[\mathcal{E}_1] \geq 1/e$.

*Proof.* Without loss of generality we assume that $\mathcal{A}$ does not ask for the signature of the same message twice. We prove by induction that after $\mathcal{A}$ makes $i$ signature queries the probability that $\mathcal{B}$ does not abort is at least $(1 - 1/(q_S + 1))^i$. The claim is trivially true for $i = 0$. Let $M_i$ be $\mathcal{A}$'s $i$'th signature query and let $\langle M_i, w_i, b_i, c_i \rangle$ be the corresponding tuple on the $H$-list. Then prior to issuing the query, the bit $c_i$ is independent of $\mathcal{A}$'s view — the only value that could be given to $\mathcal{A}$ that depends on $c_i$ is $H(M_i)$, but the distribution on $H(M_i)$ is the same whether $c_i = 0$ or $c_i = 1$. Therefore, the probability that this query causes $\mathcal{B}$ to abort is at most $1/(q_S + 1)$. Using the inductive hypothesis and the independence of $c_i$, the probability that $\mathcal{B}$ does not abort after this query is at least $(1 - 1/(q_S + 1))^i$. This proves the inductive claim. Since $\mathcal{A}$ makes at most $q_S$ signature queries the probability that $\mathcal{B}$ does not abort as a result of all signature queries is at least $(1 - 1/(q_S + 1))^{q_S} \geq 1/e$. $\square$

**Claim 2:** If algorithm $\mathcal{B}$ does not abort as a result of $\mathcal{A}$'s signature queries then algorithm $\mathcal{A}$'s view is identical to its view in the real attack. Hence, $\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq \epsilon$.

*Proof.* The public key given to $\mathcal{A}$ is from the same distribution as a public key produced by algorithm *KeyGen*. Responses to $H$-queries are as in the real attack since each response is uniformly and independently distributed in $G_1$. All responses to signature queries are valid. Therefore, $\mathcal{A}$ will produce a valid message-signature pair with probability at least $\epsilon$. Hence, $\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq \epsilon$. $\square$

**Claim 3:** The probability that algorithm $\mathcal{B}$ does not abort after $\mathcal{A}$ outputs a valid forgery is at least $1/(q_S + 1)$. Hence, $\Pr[\mathcal{E}_3 \mid \mathcal{E}_1 \wedge \mathcal{E}_2] \geq 1/(q_S + 1)$.

*Proof.* Given that events $\mathcal{E}_1$ and $\mathcal{E}_2$ happened, algorithm $\mathcal{B}$ will abort only if $\mathcal{A}$ generates a forgery $(M_f, \sigma_f)$ for which the tuple $\langle M_f, w, b, c \rangle$ on the $H$-list has $c = 1$. At the time $\mathcal{A}$ generates its output it knows the value of $c_i$ for those $M_i$ for which it issued a signature query. All the remaining $c_i$'s are independent of $\mathcal{A}$'s view. Indeed, if $\mathcal{A}$ did not issue a signature query for $M_i$ then the only value given to $\mathcal{A}$ that depends on $c_i$ is $H(M_i)$, but the distribution on $H(M_i)$ is the same whether $c_i = 0$ or $c_i = 1$. Since $\mathcal{A}$ could not have issued a signature query for $M_f$ we know that $c$ is independent of $\mathcal{A}$'s current view and therefore $\Pr[c = 0 \mid \mathcal{E}_1 \wedge \mathcal{E}_2] \geq 1/(q_S + 1)$ as required. $\square$

Using the bounds from the claims above in equation (1) shows that $\mathcal{B}$ produces the correct answer with probability at least $\epsilon/e(q_S + 1) \geq \epsilon'$ as required. Algorithm $\mathcal{B}$'s running time is the same as $\mathcal{A}$'s running time plus the time is takes to respond to $(q_H + q_S)$ hash queries and $q_S$ signature queries. Each query requires an exponentiation in $G_1$ which we assume takes time $c_{G_1}$. Hence, the total running time is at most $t + c_{G_1}(q_H + 2q_S) \leq t'$ as required. This completes the proof of Theorem 3.2. $\square$

The analysis used in the proof of Theorem 3.2 resembles Coron's analysis of the Full Domain Hash (FDH) signature scheme [15]. We note that Probabilisitc Full Domain Hash (PFDH) signatures [16] have a tighter security reduction than FDH signatures. The same improvement to the security reduction can be applied to our signature scheme. However, randomizing our signature scheme as in PFDH would increase the length of the signature, defeating our main goal of constructing short signatures.

**The necessity of $\psi : G_2 \to G_1$.** Recall that the proof of security relied on the existence of an efficiently computable isomorphism $\psi : G_2 \to G_1$. To show the necessity of $\psi$ we give an example of a bilinear map $e : G_1 \times G_2 \to G_T$ for which the co-CDH problem is believed to be hard on $(G_1, G_2)$ and yet the resulting signature scheme is insecure.

Let $q$ be a prime and let $G_2$ be a subgroup of $\mathbb{Z}_q^*$ of prime order $p$ with generator $g$. Let $G_1$ be the group $G_1 = \{0, 1, \ldots, p-1\}$ with addition modulo $p$. Define the map $e : G_1 \times G_2 \to G_2$ as $e(x, y) = y^x$. The map is clearly bilinear since $e(ax, y^b) = e(x, y)^{ab}$. The co-CDH problem on $(G_1, G_2)$ is as follows: Given $g, g^a \in G_2$ and $x \in G_1$ compute $ax \in G_1$. The problem is believed to be hard since an algorithm for computing co-CDH on $(G_1, G_2)$ gives an algorithm for computing discrete log in $G_2$. Hence, $(G_1, G_2)$ satisfies all the conditions of Theorem 3.2 except that there is no known computable isomorphism $\psi : G_2 \to G_1$. It is is easy to see that the resulting signature scheme from this bilinear map is insecure: Given one message-signature pair, it is easy to recover the secret key.

We comment that one can avoid using $\psi$ at the cost of making a stronger complexity assumption. Without $\psi$ the necessary assumption for proving security is that no polynomial time algorithm can compute $h^a \in G_1$ given $g_2, g_2^a \in G_2$ and $g_1, g_1^a, h \in G_1$. Since $\psi$ naturally exists in all the group pairs $(G_1, G_2)$ we are considering, there is no reason to rely on this stronger complexity assumption.

### 3.2 Hashing onto elliptic curves

The signature scheme needs a hash function $H : \{0,1\}^* \to G_1$. In the next section we use elliptic curves to construct co-GDH groups and therefore we need a hash function $H : \{0,1\}^* \to G_1$ where $G_1$ is a subgroup of an elliptic curve. Since it is difficult to build hash functions that hash directly onto a subgroup of an elliptic curve we slightly relax the hashing requirement.

Let $E/\mathbb{F}_q$ be an elliptic curve defined by $y^2 = f(x)$ and let $E(\mathbb{F}_q)$ have order $m$. Let $P \in E(F_q)$ be a point of prime order $p$. We wish to hash onto the subgroup $G_1 = \langle P \rangle$. Suppose we are given a hash function $H' : \{0,1\}^* \to \mathbb{F}_q \times \{0,1\}$. Such hash functions $H'$ can be built from standard cryptographic hash functions. The security analysis will view $H'$ as a random oracle. We use the following deterministic algorithm called *MapToGroup* to hash messages in $\{0,1\}^*$ onto $G_1$. Fix a small parameter $I = \lceil \log_2 \log_2(1/\delta) \rceil$, where $\delta$ is some desired bound on the failure probability.

***MapToGroup***$_{H'}$**:** The algorithm defines $H : \{0,1\}^* \to G_1$ as follows:

1. Given $M \in \{0,1\}^*$, set $i \leftarrow 0$;
2. Set $(x, b) \leftarrow H'(i \parallel M) \in \mathbb{F}_q \times \{0,1\}$, where $i$ is represented as an $I$-bit string;
3. If $f(x)$ is a quadratic residue in $\mathbb{F}_q$ then do:
   3a. Let $y_0, y_1 \in \mathbb{F}_q$ be the two square roots of $f(x)$. We use $b \in \{0,1\}$ to choose between these roots. Choose some full ordering of $\mathbb{F}_q$ and ensure that $y_1$ is greater than $y_0$ according

to this ordering (swapping $y_0$ and $y_1$ if necessary). Set $\tilde{P}_M \in E(\mathbb{F}_q)$ to be the point $\tilde{P}_M = (x, y_b)$.

3b. Compute $P_M = (m/p)\tilde{P}_M$. Then $P_M$ is in $G_1$. Output $MapToGroup_{H'}(M) = P_M$ and stop.

4. Otherwise, increment $i$, and go to Step 2; if $i$ reaches $2^I$, report failure.

The failure probability can be made arbitrarily small by picking an appropriately large $I$. For each $i$, the probability that $H'(i \parallel M)$ leads to a point on $G$ is approximately 1/2 (where the probability is over the choice of the random oracle $H'$). Hence, the expected number of calls to $H'$ is approximately 2, and the probability that a given message $M$ will be found unhashable is $1/2^{(2^I)} \leq \delta$.

**Lemma 3.3.** *Let $E/\mathbb{F}_q$ be an elliptic curve and let $E(\mathbb{F}_q)$ have order $m$. Let $G_1$ be a subgroup of $E(\mathbb{F}_q)$ of order $p$. We assume $p^2$ does not divide $m$. Suppose the co-GDH signature scheme is $(t, q_S, q_H, \epsilon)$-secure in the groups $(G_1, G_2)$ when a random hash function $H : \{0,1\}^* \to G_1$ is used. Then it is $(t - 2^I c_{G_1} q_H, q_S, q_H, \epsilon)$-secure when the hash function $H$ is computed with $MapToGroup_{H'}$ and $H'$ is a random hash function $H' : \{0,1\}^* \to \mathbb{F}_q \times \{0,1\}$.*

*Proof Sketch.* Suppose a forger algorithm $\mathcal{F}'$ $(t, q_S, q_H, \epsilon)$-breaks the signature scheme on $(G_1, G_2)$ when the hash function $H$ is computed using $MapToGroup_{H'}$. We build an algorithm $\mathcal{F}$ that $(t + 2^I c_{G_1} q_H, q_S, q_H, \epsilon)$-breaks the signature scheme when $H$ is a random oracle $H : \{0,1\}^* \to G_1$.

Our new forger $\mathcal{F}$ will run $\mathcal{F}'$ as a black box. Algorithm $\mathcal{F}$ passes signatures queries made by $\mathcal{F}'$ to its own signature oracle. $\mathcal{F}$ uses its hash oracle $H : \{0,1\}^* \to G_1$ to simulate for $\mathcal{F}'$ the behavior of $MapToGroup_{H'}$. It uses an array $s_{ij}$, whose entries are elements of $\mathbb{F}_q \times \{0,1\}$. The array has $q_H$ rows and $2^I$ columns. On initialization, $\mathcal{F}$ fills $s_{ij}$ with uniformly-selected elements of $\mathbb{F}_q \times \{0,1\}$.

$\mathcal{F}$ then runs $\mathcal{F}'$, and keeps track (and indexes) all the unique messages $M_i$ for which $\mathcal{F}'$ requests an $H'$ hash. When $\mathcal{F}'$ asks for an $H'$ hash of a message $w \parallel M_i$ whose $M_i$ the forger $\mathcal{F}$ had not previously seen (and whose $w$ is an arbitrary $I$-bit string), $\mathcal{F}$ scans the row $s_{ij}$, $0 \leq j < 2^I$. For each $(x, b) = s_{ij}$, $\mathcal{F}$ follows Step 3 of *MapToGroup*, above, seeking points in $G$. For the smallest $j$ for which $s_{ij}$ maps into $G_1$, $\mathcal{F}$ replaces $s_{ij}$ with a different point $(x_i, b_i)$ defined as follows. Let $Q_i = H(M_i) \in G_1$. Then $\mathcal{F}$ constructs a random point $\tilde{Q}_i \in E(\mathbb{F}_q)$ satisfying $(m/p)\tilde{Q}_i = Q_i$ as follows:

1. Let $w = (m/p)^{-1} \bmod p$. Note that $m/p$ is integer since $p$ divides $m$. Furthermore, $m/p$ has an inverse modulo $p$ since $p^2$ does not divide $m$ and hence $m/p$ is relatively prime to $p$.
2. Pick a random point $T_i \in E(\mathbb{F}_q)$.
3. Set $\tilde{Q}_i = (x_i, y_i) = pT_i + wQ_i$.

Then $\tilde{Q}_i$ is a random point in $E(\mathbb{F}_q)$ such that $(m/p)\tilde{Q}_i = Q_i$. $\mathcal{F}$ sets $s_{ij} = (x_i, b_i)$ where $b_i \in \{0,1\}$ is set so that $(x_i, b_i)$ maps to $\tilde{Q}_i$ in Step 3a of *MapToGroup*. Then $MapToGroup_{H'}(M_i) = H(M_i)$ as required.

Once this preliminary patching has been completed, $\mathcal{F}$ is able to answer $H'$ hash queries by $\mathcal{F}'$ for strings $w \parallel M_i$ by simply returning $s_{iw}$. The simulated $H'$ which $\mathcal{F}'$ sees is statistically indistinguishable from that in the real attack. Thus, if $\mathcal{F}'$ succeeds in breaking the signature scheme using $MapToGroup_{H'}$ then $\mathcal{F}$, in running $\mathcal{F}'$ while consulting $H$, succeeds with the same likelihood, and suffers only a running-time penalty from maintaining the additional information and running the exponentiation in Step 3 of the *MapToGroup* algorithm. We again assume that exponentiation in $G_1$ takes time $c_{G_1}$. $\square$

# 4 Building co-GDH groups with small representations

Using the Weil [29, pp. 243–245] and Tate [18] pairings, we obtain co-GDH groups from certain elliptic curves. We recall some necessary facts about elliptic curves (see, e.g., [29, 44]), and then show how to use certain curves for short signatures.

## 4.1 Elliptic curves and the Weil pairing

Our goal is to construct bilinear groups $(G_1, G_2)$ which lead to co-GDH groups as discussed in Section 2.1. Let $E/\mathbb{F}_q$ be an elliptic curve. We first define a useful constant called the security multiplier of a subgroup $\langle P \rangle \subseteq E(\mathbb{F}_q)$.

**Definition 4.1.** Let $q$ be a prime power, and $E/\mathbb{F}_q$ an elliptic curve with $m$ points in $E(\mathbb{F}_q)$. Let $P$ in $E(\mathbb{F}_q)$ be a point of prime order $p$ where $p^2 \nmid m$. We say that the subgroup $\langle P \rangle$ has a security multiplier $\alpha$, for some integer $\alpha > 0$, if the order of $q$ in $\mathbb{F}_p^*$ is $\alpha$. In other words:

$$p \mid q^\alpha - 1 \quad \text{and} \quad p \nmid q^k - 1 \quad \text{for all } k = 1, 2, \ldots, \alpha - 1.$$

The security multiplier of $E(\mathbb{F}_q)$ is the security multiplier of the largest prime order subgroup in $E(\mathbb{F}_q)$.

We describe two families of curves that provide $\alpha = 6$. For standard security parameters this is sufficient for obtaining short signatures. It is an open problem to build useful elliptic curves with slightly higher $\alpha$, say $\alpha = 10$ (see Section 4.5).

Our first step is to define $G_1$ and $G_2$. We will then describe a bilinear map $e : G_1 \times G_2 \to G_T$, describe an isomorphism $\psi : G_2 \to G_1$, and discuss the intractability of co-CDH on $(G_1, G_2)$.

**Balasubramanian-Koblitz.** Let $E/\mathbb{F}_q$ be an elliptic curve and let $P \in E(\mathbb{F}_q)$ be a point of prime order $p$ with $p \neq q$. Suppose the subgroup $\langle P \rangle$ has security multiplier $\alpha > 1$, i.e. $p \nmid q - 1$. Then, a useful result of Balasubramanian and Koblitz [3] shows that $E(\mathbb{F}_{q^\alpha})$ contains a point $Q$ of order $p$ that is linearly independent of $P$. We set $G_1 = \langle P \rangle$ and $G_2 = \langle Q \rangle$. Then $|G_1| = |G_2| = p$. Note that $G_1 \subseteq E(\mathbb{F}_q)$ and $G_2 \subseteq E(\mathbb{F}_{q^\alpha})$.

**The Weil and Tate pairings.** Let $E[p]$ be the group of points of order dividing $p$ in $E(F_{q^\alpha})$. Then the group $E[p]$ is isomorphic to $\mathbb{Z}_p \times \mathbb{Z}_p$ [44] and also $G_1, G_2 \subseteq E[p]$. The Weil pairing is a map $e : E[p] \times E[p] \to \mathbb{F}_{q^\alpha}^*$ with the following properties:

1. Identity: for all $R \in E[p]$, $e(R, R) = 1$.
2. Bilinear: for all $R_1, R_2 \in E[p]$ and $a, b \in \mathbb{Z}$ we have $e(aR_1, bR_2) = e(R_1, R_2)^{ab}$.
3. Non-degenerate: if for $R \in E[p]$ we have $e(R, R') = 1$ for all $R' \in E[p]$, then $R = \mathcal{O}$.
4. Computable: for all $R_1, R_2 \in E[p]$, the pairing $e(R_1, R_2)$ is computable in polynomial time [34].

Note that $e(R_1, R_2) = 1$ if and only if $R_1$ and $R_2$ are linearly dependent. See [31, 10] for a definition of the Weil pairing and a description of the algorithm for computing it. The Tate pairing [18] is another useful bilinear map on $E[p]$. It has properties similar to those of the Weil pairing, but does not necessarily satisfy Property 1 (identity).

The Weil pairing on the curve $E$ gives a computable, non-degenerate bilinear map $e : G_1 \times G_2 \to \mathbb{F}_{q^\alpha}^*$ which enables us to solve the Decision co-Diffie-Hellman problem on the groups $(G_1, G_2)$. When the Tate pairing is non-degenerate on $G_1 \times G_2$ it can also be used to solve Decision co-Diffie-Hellman on $(G_1, G_2)$.

**The trace map.** We present a computable isomorphism $\psi : G_2 \to G_1$, using the trace map, tr, which sends points in $E(\mathbb{F}_{q^\alpha})$ to $E(\mathbb{F}_q)$. Let $\sigma_1, \ldots, \sigma_\alpha$ be the Galois maps of $\mathbb{F}_{q^\alpha}$ over $\mathbb{F}_q$. Also, for $R = (x, y) \in E(\mathbb{F}_{q^\alpha})$ define $\sigma_i(R) = (\sigma_i(x), \sigma_i(y))$. Then the trace map $\mathrm{tr} : E(\mathbb{F}_{q^\alpha}) \to E(\mathbb{F}_q)$ is defined by:

$$\mathrm{tr}(R) = \sigma_1(R) + \ldots + \sigma_\alpha(R).$$

**Fact 4.2.** *Let $P \in E(\mathbb{F}_q)$ be a point of prime order $p \neq q$ and let $\langle P \rangle$ have security multiplier $\alpha > 1$. Let $Q \in E(\mathbb{F}_{q^\alpha})$ be a point of order $p$ that is linearly independent of $P$. If $tr(Q) \neq \mathcal{O}$ then tr is an isomorphism from $\langle Q \rangle$ to $\langle P \rangle$.*

*Proof.* Suppose $R \in E(\mathbb{F}_q)$ is a point of order $p$. If $R$ is not in $\langle P \rangle$ then $P$ and $R$ generate $E[p]$ and therefore $E[p] \subseteq E(\mathbb{F}_q)$. It follows that $e(P, R) \in \mathbb{F}_q^*$ has order $p$ since otherwise $e$ would be degenerate on $E[p]$. But since $\alpha > 1$ we know that $p$ does not divide $q - 1$ and consequently there are no elements of order $p$ in $\mathbb{F}_q^*$. Hence, we must have $R \in \langle P \rangle$. It follows that all the points in $E(\mathbb{F}_q)$ of order $p$ are contained in $\langle P \rangle$. Since $\mathrm{tr}(Q) \neq \mathcal{O}$, we know that $\mathrm{tr}(Q) \in E(\mathbb{F}_q)$ has order $p$ and therefore $\mathrm{tr}(Q) \in \langle P \rangle$. Hence, tr is an isomorphism from $\langle Q \rangle$ to $\langle P \rangle$. □

Hence, when $\mathrm{tr}(Q) \neq \mathcal{O}$, the trace map is an isomorphism from $G_2$ to $G_1$ and is computable in polynomial time in $\alpha$ and $\log q$ as required.

**Intractability of co-CDH on $(G_1, G_2)$.** The remaining question is the difficulty of the co-CDH problem on $(G_1, G_2)$. we review necessary conditions for CDH intractability. The best known algorithm for solving co-CDH on $(G_1, G_2)$ is to compute discrete-log in $G_1$. In fact, the discrete-log and CDH problems in $G_1$ are known to be computationally equivalent given some extra information about the group $G_1$ [30]. Therefore, it suffices to consider necessary conditions for making the discrete-log problem on $E(\mathbb{F}_q)$ intractable.

Let $\langle P \rangle$ be a subgroup of $E(\mathbb{F}_q)$ of order $p$ with security multiplier $\alpha$. We briefly discuss two standard ways for computing discrete-log in $\langle P \rangle$.

1. **MOV:** Use an efficiently computable homomorphism, as in the MOV reduction [32], to map the discrete log problem in $\langle P \rangle$ to a discrete log problem in some extension of $\mathbb{F}_q$, say $\mathbb{F}_{q^i}$. We then solve the discrete log problem in $\mathbb{F}_{q^i}^*$ using the Number Field Sieve algorithm [43]. The image of $\langle P \rangle$ under this homomorphism must be a subgroup of $\mathbb{F}_{q^i}^*$ of order $p$. Thus we have $p | (q^i - 1)$, which by the definition of $\alpha$ implies that $i \geq \alpha$. Hence, the MOV method can, at best, reduce the discrete log problem in $\langle P \rangle$ to a discrete log problem in a subgroup of $\mathbb{F}_{q^\alpha}^*$. Therefore, to ensure that discrete log is hard in $\langle P \rangle$ we want curves where $\alpha$ is sufficiently large to make discrete log in $\mathbb{F}_{q^\alpha}^*$ intractable.

2. **Generic:** Generic discrete log algorithms such as Baby-Step-Giant-Step and Pollard's Rho method [33] have a running time proportional to $\sqrt{p}$. Therefore, we must ensure that $p$ is sufficiently large.

In summary, we want curves $E/\mathbb{F}_q$ where both a generic discrete log algorithm in $E(\mathbb{F}_q)$ and the Number Field Sieve in $\mathbb{F}_{q^\alpha}^*$ are intractable.

## 4.2 Co-GDH signatures from elliptic curves

We summarize the construction for co-GDH groups and adapt the signature scheme to use a group of points on an elliptic curve.

The co-GDH groups $(G_1, G_2)$ we use are defined as follows:

1. Let $E/\mathbb{F}_q$ be an elliptic curve and let $P \in E(\mathbb{F}_q)$ be a point of prime order $p$ where (1) $p \neq q$, (2) $p \nmid q-1$, and (3) $p^2$ does not divide $|E(\mathbb{F}_q)|$.
2. Let $\alpha > 1$ be the security multiplier of $\langle P \rangle$. By Balasubramanian and Koblitz [3] there exists a point $Q \in E(\mathbb{F}_{q^\alpha})$ that is linearly independent of $P$. It is easy to construct such a $Q$ in expected polynomial time once the number of points in $E(\mathbb{F}_{q^\alpha})$ is known. Since $\alpha > 1$ we know that $Q \notin E(\mathbb{F}_q)$. We ensure that $\mathrm{tr}(Q) \neq \mathcal{O}$. If $\mathrm{tr}(Q) = \mathcal{O}$ replace $Q$ by $Q + P$. Then $Q + P$ is of order $p$, it is linearly independent of $P$, and $\mathrm{tr}(Q+P) \neq \mathcal{O}$ since $\mathrm{tr}(P) \neq \mathcal{O}$.
3. Set $G_1 = \langle P \rangle$ and $G_2 = \langle Q \rangle$.
4. Since $P$ and $Q$ are linearly independent, the Weil pairing gives a non-degenerate bilinear map $e : G_1 \times G_2 \to \mathbb{F}^*_{q^\alpha}$. It can be computed in polynomial time in $\alpha$ and $\log q$. When the Tate pairing is non-degenerate on $G_1 \times G_2$ it can also be used as a bilinear map.
5. Since $\mathrm{tr}(Q) \neq \mathcal{O}$ the trace map on $E(\mathbb{F}_{q^\alpha})$ is an isomorphism from $G_2$ to $G_1$ computable in polynomial time in $\alpha$ and $\log q$.

With these subgroups $G_1, G_2$ of the elliptic curve $E/\mathbb{F}_q$ the signature scheme works as follows. Recall that $\mathit{MapToGroup}_{H'}$ is a hash function $\mathit{MapToGroup}_{H'} : \{0,1\}^* \to G_1$ built from a hash function $H' : \{0,1\}^* \to \mathbb{F}^*_q \times \{0,1\}$ as described in Section 3.2.

**Key generation** Pick random $x \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and compute $V \leftarrow xQ$. The public key is $V \in E(\mathbb{F}_{q^\alpha})$. The secret key is $x$.

**Signing** Given a secret key $x \in \mathbb{Z}_p$, and a message $M \in \{0,1\}^*$, do:
1. Compute $R \leftarrow \mathit{MapToGroup}_{H'}(M) \in G_1$,
2. $\sigma \leftarrow xR \in E(\mathbb{F}_q)$, and
3. output the $x$-coordinate of $\sigma$ as the signature $s$ on $M$. Then $s \in \mathbb{F}_q$.

**Verification** Given a public key $V \in G_2$, a message $M \in \{0,1\}^*$, and a signature $s \in \mathbb{F}_q$ do:
1. Find a $y \in \mathbb{F}_q$ such that $\sigma = (s, y)$ is a point of order $p$ in $E(\mathbb{F}_q)$. If no such $y$ exists, output `invalid` and stop.
2. Compute $R \leftarrow \mathit{MapToGroup}_{H'}(M) \in G_1$,
3. Test if either $e(\sigma, Q) = e(R, V)$ or $e(\sigma, Q)^{-1} = e(R, V)$.
   If so, output `valid`; Otherwise, output `invalid`.

The signature length is $\lceil \log_2 q \rceil$. Note that during verification we accept the signature if $e(\sigma, Q)^{-1} = e(R, V)$. This is to account for the fact that the signature $s \in \mathbb{F}_q$ could have come from either the point $\sigma$ or $-\sigma$ in $E(\mathbb{F}_q)$.

**Security.** By Theorem 3.2 it suffices to study the difficulty of co-CDH on $(G_1, G_2)$. The best known algorithm for solving the co-CDH problem on $(G_1, G_2)$ requires the computation of a discrete log in $G_1$ or the computation of a discrete log in $\mathbb{F}^*_{q^\alpha}$.

### 4.3 Using non-supersingular curves over fields of high characteristic

It remains to build elliptic curves with the desired security multiplier $\alpha$. In the next two sections we show curves with security multiplier, $\alpha = 6$. We begin by describing a family of non-supersingular elliptic curves with $\alpha = 6$. This family is outlined by Miyaji *et al.* [36]. We call these MNT curves.

The idea is as follows: Suppose $q = (2\ell)^2 + 1$ and $p = (2\ell)^2 - 2\ell + 1$ for some $\ell \in \mathbb{Z}$. Then it can be verified that $p$ divides $q^6 - 1$, but does not divide $q^i - 1$ for $0 < i < 6$. So, when $p$ is prime, a curve $E/\mathbb{F}_q$ with $p$ points is likely to have security multiplier $\alpha = 6$.

| Discriminant $D$ | Signature Size $\lceil \log_2 q \rceil$ | DLog Security $\lceil \log_2 p \rceil$ | MOV Security $\lceil 6 \log_2 q \rceil$ |
|---|---|---|---|
| 13368643 | 149 | 149 | 894 |
| 254691883 | 150 | 147 | 900 |
| 8911723 | 157 | 157 | 942 |
| 62003 | 159 | 158 | 954 |
| 12574563 | 161 | 161 | 966 |
| 1807467 | 163 | 163 | 978 |
| 6785843 | 168 | 166 | 1008 |
| 28894627 | 177 | 177 | 1062 |
| 153855691 | 185 | 181 | 1110 |
| 658779 | 199 | 194 | 1194 |
| 1060147 | 203 | 203 | 1218 |
| 20902979 | 204 | 204 | 1224 |
| 9877443 | 206 | 206 | 1236 |

Table 1: Non-supersingular elliptic curves for co-GDH Signatures. $E$ is a curve over the prime field $\mathbb{F}_q$ and $p$ is the largest prime dividing its order. The MOV reduction maps the curve onto the field $\mathbb{F}_{q^6}$. $D$ is the discriminant of the complex multiplication field of $E/\mathbb{F}_q$.

To build $E/\mathbb{F}_q$ with $p$ points as above we use complex multiplication [8, chapter VIII]. We briefly explain how to do so. Suppose we had integers $y, t$ and another positive integer $D = 3 \bmod 4$ such that

$$q = (t^2 + Dy^2)/4 \tag{2}$$

is an integer prime. Then the complex multiplication method will produce an elliptic curve $E/\mathbb{F}_q$ with $q + 1 - t$ points in time $O(D^2(\log q)^3)$. The value $t$ is called the trace of the curve.

We want a curve over $\mathbb{F}_q$ with $p$ points where $q = (2\ell)^2 + 1$ and $p = (2\ell)^2 - 2\ell + 1$. Therefore, $t = q + 1 - p = 2\ell + 1$. Plugging these values into (2) we get $4((2\ell)^2 + 1) = (2\ell + 1)^2 + Dy^2$ which leads to:

$$(6\ell - 1)^2 - 3Dy^2 = -8. \tag{3}$$

For a fixed $D = 3 \bmod 4$, we need integers $\ell, y$ satisfying the equation above such that $q = (2\ell)^2 + 1$ is prime and $p = (2\ell)^2 - 2\ell + 1$ is prime (or is a small multiple of a prime). For any such solution we can verify that we get a curve $E(\mathbb{F}_q)$ with security multiplier $\alpha = 6$. Finding integer solutions $\ell, y$ to an equation of type (3) is done by reducing it to Pell's equation, whose solution is well known [45].

Table 1 gives some values of $D$ that lead to suitable curves for our signature scheme. For example, we get a curve $E/\mathbb{F}_q$ where $q$ is a 168-bit prime. Signatures using this curve are 168-bits while the best algorithm for co-CDH on $E(\mathbb{F}_q)$ requires either (1) a generic discrete log algorithm taking time approximately $2^{83}$, or (2) a discrete log in a 1008-bit finite field of large characteristic.

### 4.4 A special supersingular curve

Another method for building curves with security multiplier $\alpha = 6$ is to use a special supersingular curve $E/\mathbb{F}_3$. Specifically, we use the curve $E : y^2 = x^3 + 2x \pm 1$ over $\mathbb{F}_3$. The MOV reduction maps the discrete log problem in $E(\mathbb{F}_{3^\ell})$ to $\mathbb{F}^*_{3^{6\ell}}$. We use two simple lemmas to describe the behavior of these curves. (See also [47, 28].)

**Lemma 4.3.** *The curve $E^+$ defined by $y^2 = x^3 + 2x + 1$ over $\mathbb{F}_3$ satisfies*

$$|E^+(\mathbb{F}_{3^\ell})| = \begin{cases} 3^\ell + 1 + \sqrt{3 \cdot 3^\ell} & \textit{when} \quad \ell = \pm 1 \bmod 12, \textit{and} \\ 3^\ell + 1 - \sqrt{3 \cdot 3^\ell} & \textit{when} \quad \ell = \pm 5 \bmod 12. \end{cases}$$

*The curve $E^-$ defined by $y^2 = x^3 + 2x - 1$ over $\mathbb{F}_3$ satisfies*

$$|E^-(\mathbb{F}_{3^\ell})| = \begin{cases} 3^\ell + 1 - \sqrt{3 \cdot 3^\ell} & \textit{when} \quad \ell = \pm 1 \bmod 12, \textit{and} \\ 3^\ell + 1 + \sqrt{3 \cdot 3^\ell} & \textit{when} \quad \ell = \pm 5 \bmod 12. \end{cases}$$

*Proof.* See [28, section 2]. □

**Lemma 4.4.** *Let $E$ be an elliptic curve defined by $y^2 = x^3 + 2x \pm 1$ over $\mathbb{F}_{3^\ell}$, where $\ell \bmod 12$ equals $\pm 1$ or $\pm 5$. Then $|E(\mathbb{F}_{3^\ell})|$ divides $3^{6\ell} - 1$.*

*Proof.* See [47]. □

Together, Lemmas 4.3 and 4.4 show that, for the relevant values of $\ell$, groups on the curves $E^+/\mathbb{F}_{3^\ell}$ and $E^-/\mathbb{F}_{3^\ell}$ will have security multiplier $\alpha$ at most 6 (more specifically: $\alpha \mid 6$). Whether the security parameter actually is 6 for a particular prime subgroup of a curve must be determined by computation.

**Automorphism of $E^+, E^-/\mathbb{F}_{3^{6\ell}}$:** Both curves $E^+$ and $E^-$ have a useful automorphism that make the prime-order subgroups of $E^+(\mathbb{F}_{3^\ell})$ and $E^-(\mathbb{F}_{3^\ell})$ into GDH groups (as opposed to co-GDH groups). This fact can be used to shrink the size of the public key since it makes it possible for for the public key to live in $E(\mathbb{F}_{3^\ell})$ as opposed to $E(\mathbb{F}_{3^{6\ell}})$.

The automorphism is defined as follows. For $\ell$ such that $\ell \bmod 12$ is $\pm 1$ or $\pm 5$, compute three elements of $\mathbb{F}_{3^{6\ell}}$, $u$, $r^+$, and $r^-$, satisfying $u^2 = -1$, $(r^+)^3 + 2r^+ + 2 = 0$, and $(r^-)^3 + 2r^- - 2 = 0$. Now consider the following maps over $\mathbb{F}_{3^{6\ell}}$:

$$\phi^+(x, y) = (-x + r^+, uy) \quad \text{and} \quad \phi^-(x, y) = (-x + r^-, uy).$$

**Lemma 4.5.** *Let $\ell \bmod 12$ equal $\pm 1$ or $\pm 5$. Then $\phi^+$ is an automorphism of $E^+/\mathbb{F}_{3^{6\ell}}$ and $\phi^-$ is an automorphism of $E^-/\mathbb{F}_{3^{6\ell}}$. Moreover, if $P$ is a point of order $p$ on $E^+/\mathbb{F}_{3^\ell}$ (or on $E^-/\mathbb{F}_{3^\ell}$) then $\phi^+(P)$ (or $\phi^-(P)$) is a point of order $p$ that is linearly independent of $P$.*

*Proof.* See Silverman [44, p. 326]. □

Let $E/\mathbb{F}_{3^\ell}$ be one of $E^+$ or $E^-$ and let $P \in E(\mathbb{F}_{3^\ell})$ be a point of prime order $p$. Set $G_1 = \langle P \rangle$, the group generated by $P$. Let $\phi : E(\mathbb{F}_{3^\ell}) \to E(\mathbb{F}_{3^{6\ell}})$ be the automorphism of the curve from above. Define the modified Weil pairing $\hat{e} : G_1 \times G_1 \to \mathbb{F}^*_{3^{6\ell}}$ as follows: $\hat{e}(P_1, P_2) = e(P_1, \phi(P_2))$ where $e$ is the standard Weil pairing on $E[p]$. By Lemma 4.5 we know that $\phi(P)$ is linearly independent of $P$. Therefore, $\hat{e}$ is non-degenerate. It follows that $G_1$ is a GDH group. This has two implications for the signature scheme:

- Security of the signature scheme is based on the difficulty of the standard Computational Diffie-Hellman problem in $G_1$ (as opposed to the co-CDH problem).
- Public keys are elements of $G_1$ and, hence, are shorter than public keys should the automorphism not exist.

| curve | $l$ | Sig Size $\lceil \log_2 3^\ell \rceil$ | DLog Security $\lceil \log_2 p \rceil$ | MOV Security $\lceil 6 \log_2 3^\ell \rceil$ |
|---|---|---|---|---|
| $E^-$ | 79 | 126 | 126 | 752 |
| $E^+$ | 97 | 154 | 151 | 923 |
| $E^+$ | 121 | 192 | 155 | 1151 |
| $E^+$ | 149 | 237 | 220 | 1417 |
| $E^+$ | 163 | 259 | 256 | 1551 |
| $E^-$ | 163 | 259 | 259 | 1551 |
| $E^+$ | 167 | 265 | 262 | 1589 |

Table 2: Supersingular elliptic curves for GDH signatures. Here $p$ is the largest prime divisor of $|E(\mathbb{F}_{3^\ell})|$. The MOV reduction maps the curve onto a field of characteristic 3 of size $3^{6\ell}$.

**Useful curves.** Some useful instantiations of these curves are presented in Table 2. Note that we restrict these instantiations to those where $\ell$ is prime, to avoid Weil-descent attacks [21, 22], except for $\ell = 121$. It has recently been shown that certain Weil-descent attacks are not effective for this case [17], suggesting that it may be safe to use.

**Performance.** Galbraith *et al.* [20] and Baretto *et al.* [4] show that the Frobenius map on the curves $E^+, E^-$ can be used to speed the computation of the Weil and Tate pairings on these curves. This results in a significant speed-up to the signature-verification algorithm. Consequently, the signature scheme using these curves is much faster than the scheme using the curves from the previous section.

**The bad news.** MOV reduces the discrete log problem on $E^+(\mathbb{F}_{3^\ell})$ and $E^-(\mathbb{F}_{3^\ell})$ to a discrete log problem in $\mathbb{F}^*_{3^{6\ell}}$. A discrete-log algorithm due to Coppersmith [14, 43] is specifically designed to compute discrete log in small characteristic fields. Consequently, a discrete-log problem in $\mathbb{F}^*_{3^n}$ is much easier than a discrete-log problem in $\mathbb{F}^*_p$ where $p$ is a prime of approximately the same size as $3^n$. To get security equivalent to DSA using a 1024-bit prime, we would have to use a curve $E(\mathbb{F}_{3^\ell})$ where $3^{6\ell}$ is much larger than 1024 bits. This leads to much longer signatures, defeating the point of using these curves. In other words, for a fixed signature length, these supsersingular curves lead to a signature with reduced security compared to the curves of the previous section.

### 4.5 An open problem: higher security multipliers

With the curves of Section 4.3, a security multiplier of $\alpha = 6$ is sufficient for constructing short signatures with security comparable to DSA using a 1024-bit prime. However, to obtain security comparable to DSA using a 2048-bit prime with $\alpha = 6$ we get signatures of length $2048/6 = 342$ bits. Elliptic curves with higher $\alpha$, say $\alpha = 10$, would result in short signatures when higher security is needed (such as 2048-bit discrete-log security).

Let $q$ be a large prime power, say, $q > 2^{160}$. It is currently an open problem to construct an elliptic curve $E/\mathbb{F}_q$ such that $E(\mathbb{F}_q)$ has $\alpha = 10$ and $E(\mathbb{F}_q)$ has prime order. Baretto *et al.* [5] show how to build elliptic curves $E$ such that $E(\mathbb{F}_q)$ has a given security multipliers $\alpha$. However, the largest prime order subgroup of $E(\mathbb{F}_q)$ is much smaller than $q$. Consequently, these curves cannot be used for secure short signatures — a generic discrete-log algorithm in $E(\mathbb{F}_q)$ will break the scheme in time proportional to $\sqrt{p}$ where $p$ is the largest prime factor of $|E(\mathbb{F}_q)|$.

One could also build GDH groups of higher genus. Galbraith [19] constructs supersingular curves of higher genus with a "large" security multiplier. For example, the Jacobian of the supersingular curve $y^2 + y = x^5 + x^3$ has security multiplier 12 over $\mathbb{F}_{2^\ell}$. Since a point on the Jacobian of this curve of genus two is characterized by two values in $\mathbb{F}_{2^\ell}$ (the two $x$-coordinates in a reduced divisor), the length of the signature is $2\ell$ bits. Hence, we might obtain a signature of length $2\ell$ with security of computing CDH in the finite field $\mathbb{F}_{2^{12\ell}}$. This factor of 6 between the length of the signature and the degree of the finite field is the same as in the elliptic curve case. Hence, this genus 2 curve does not improve the security of the signature, but does give more variety in curves used for short signatures. Discrete log on the Jacobian of these curves is reducible to discrete-log in a field of characteristic 2 and consequently one must take Coppersmith's discrete log algorithm [14] into account, as discussed at the end of Section 4.4.

To obtain larger security multipliers, Rubin and Silverberg [42] propose certain Abelian varieties. Superficially, they show that signatures produced using the curve of Section 4.4 can be shortened by 20%. The result is an $n$-bit signature where the pairing reduces the discrete log problem to a finite field of size approximately $2^{7.5n}$. This is the only useful example we currently know of where the multiplier is greater than 6.

# 5 Extensions

Our signatures support threshold signatures and batch verification. Surprisingly, signatures from distinct people on distinct messages can be aggregated into a single convincing signature. We briefly survey these extensions here and refer to Boldyreva [9], Verheul [46], and Boneh *et al.* [11] for a full description and proofs of security.

## 5.1 Aggregate signatures

Common environments require managing many signatures by different parties on distinct messages. For example, certificate chains contain signatures on distinct certificates issued by various Certificate Authorities. Our signature scheme enables us to aggregate multiple signatures by distinct entities on distinct messages into a single short signature. Any party that has all the signatures can aggregate signatures, and aggregation can be done incrementally: Two signatures are aggregated, then a third is added to the aggregate, and so on. See [11] for more applications.

Let $(G_1, G_2)$ be a bilinear group pair of prime order $p$. Suppose $n$ users each have a public-private key pair. For $i = 1, \ldots, n$, user $i$ has private key $x_i \in \mathbb{Z}_p$ and public key $v_i = g_2^{x_i} \in G_2$.

Suppose user $i$ signs a message $M_i \in \{0,1\}^*$ to obtain the signature $\sigma_i = H(M_i)^{x_i} \in G_1$. The aggregate of all these signatures is computed simply as $\sigma \leftarrow \sigma_1 \sigma_2 \cdots \sigma_n \in G_1$.

Aggregate verification: We are given all public keys $v_1, \ldots, v_n \in G_2$, all messages $M_1, \ldots, M_n \in \{0,1\}^*$, and the aggregate signature $\sigma \in G_1$. To verify that, for all $i = 1, \ldots, n$, user $i$ has signed message $M_i$, we test that

1. The messages $M_1, \ldots, M_n$ are all distinct, and
2. $e(\sigma, g_2) = \prod_{i=1}^{n} e(H(M_i), v_i)$.

If both conditions hold, we accept the aggregate signature. Otherwise, we reject.

We refer to [11] for the exact security model and the proof of security. An attacker who can existentially forge an aggregate signature can be subverted to solve co-CDH on $(G_1, G_2)$. We note that aggregate signature verification requires a bilinear map — a generic Gap Diffie-Hellman group is apparently insufficient. Generic Gap Diffie-Hellman groups are sufficient for verifying aggregate signatures on the same message by different people, or for verifying aggregate signatures on distinct messages by the same person.

### 5.2 Batch verification

Suppose $n$ users all sign the same message $M \in \{0,1\}^*$. We obtain $n$ signatures $\sigma_1, \ldots, \sigma_n$. We show that these $n$ signatures can be verified as a batch much faster than verifying them one by one. A similar property holds for other signature schemes [6].

Let $(G_1, G_2)$ be a co-GDH group pair of prime order $p$. Suppose user $i$'s private key is $x_i \in \mathbb{Z}_p$ and his public key is $v_i = g_2^{x_i} \in G_2$. Signature $\sigma_i$ is $\sigma_i = H(M)^{x_i} \in G_1$. To verify the $n$ signatures as a batch we use a technique due to Bellare *et al.* [6]:

1. Pick random integers $c_1, \ldots, c_n$ from the range $[0, B]$ for some value $B$. This $B$ controls the error probability as discussed below.
2. Compute $V \leftarrow \prod_{i=1}^n v_i^{c_i} \in G_2$ and $U \leftarrow \prod_{i=1}^n \sigma_i^{c_i} \in G_1$.
3. Test that $(g_2, V, H(M), U)$ is a co-DDH tuple. Accept all $n$ signatures if so; reject otherwise.

Theorem 3.3 of [6] shows that we incorrectly accept the $n$ signatures with probability at most $1/B$. Hence, verifying the $n$ signatures as a batch is faster than verifying them one by one. Note that if all signers are required to prove knowledge of their private keys, then taking $c_1 = \ldots = c_n = 1$ is sufficient, yielding even faster batch verification [9]. A similar batch verification procedure can be used to verify quickly $n$ signatures on distinct messages issued by the *same* public key.

### 5.3 Threshold signatures

Using standard secret sharing techniques [33], our signature scheme gives an immediate robust $t$-out-of-$n$ threshold signature [9]. In a threshold signature scheme, there are $n$ parties where each possesses a share of a private key. Each party can use its share of the private key to produce a share of a signature on some message $M$. A complete signature on $M$ can only be constructed if at least $t$ shares of the signature are available.

A robust $t$-out-of-$n$ threshold signature scheme derives from our signature scheme as follows. A central authority generates a public/private key pair. Let $x \in \mathbb{Z}_p$ be the private key and $v = g_2^x \in G_2$ be the public key. The central authority picks a random degree $t-1$ polynomial $\omega \in \mathbb{Z}_p[X]$ such that $\omega(0) = x$. For $i = 1, \ldots, n$, the authority gives user $i$ the value $x_i = \omega(i)$, its share of the private key. The authority publishes the public key $v$ and $n$ values $u_i = g_2^{x_i} \in G_2$.

When a signature a on a message $M \in \{0,1\}^*$ is needed each party that wishes to participate in signature generation publishes its share of the signature as $\sigma_i = H(M)^{x_i} \in G_1$. Without loss of generality, assume users $1, \ldots, t$ participate and generate shares $\sigma_1, \ldots, \sigma_t$. Anyone can verify that share $\sigma_i$ is valid by checking that $(g_2, u_i, H(M), \sigma_i)$ is a co-Diffie-Hellman tuple. When all $t$ shares are valid, the complete signature is recovered as

$$\sigma \leftarrow \prod_{i=1}^{t} \sigma_i^{\lambda_i} \quad \text{where} \quad \lambda_i = \frac{\prod_{i=1, j \neq i}^{t}(0-j)}{\prod_{i=1, j \neq i}^{t}(i-j)} \pmod{p}.$$

If fewer than $t$ users are able to generate a signature on some message $M$ then these users can be used to solve co-CDH on $(G_1, G_2)$ [9]. This threshold scheme is robust: A participant who contributes a bad partial signature $\sigma_i$ will be detected immediately since $(g_2, u_i, H(M), \sigma_i)$ will not be a co-Diffie-Hellman tuple.

We note that there is no need for a trusted third party to generate shares of the private key. The $n$ users can generate shares of the private key without the help of a trusted third party using the protocol due to Gennaro *et al.* [23].

## 6 Conclusions

We presented a short signature based on bilinear maps on elliptic curves. A signature is only one element in a finite field. Standard signatures based on discrete log such as DSA require two elements. Our signatures are much shorter than all current variants of DSA for the same security. We showed that the scheme is existentially unforgeable under a chosen message attack (in the random oracle model) assuming the Computational Diffie-Hellman problem is hard in certain elliptic-curve groups. More generally, the signature scheme can be instantiated on any Gap Diffie-Hellman group or co-GDH group pair.

We presented two families of elliptic curves that are suitable for obtaining short signatures. The first, based on [36], is a family of non-supersingular curves over a prime finite field. The second uses supersingular curves over $\mathbb{F}_{3^\ell}$. Both families of curves produce $n$-bit signatures and the discrete log problem on these curves is reducible to a discrete log problem in a finite field of size approximately $2^{6n}$. Hence, for 1024-bit security we get signatures of size $1024/6 = 171$ bits.

We expect that the first family of curves (the non-supersingular curves) will be the one used for short signatures: 171-bit signatures with 1024-bit security. As discussed at the end of Section 4.4, the second family of curves (the supersingular curve over $\mathbb{F}_{3^\ell}$) should not be used for short signatures. The problem is that discrete log on these curves reduces to a discrete log in a finite field of characteristic 3 where Coppersmith's algorithm can be used.

Implementation results [20, 4] indicate that the signature scheme performs well. Signature generation is just a simple multiplication on an elliptic curve and is faster than RSA signature generation. Verification requires two computations of the bilinear map and is slower than RSA signature verification.

In Section 4.5 we outlined an open problem that would enable us to get even better security while maintaining the same length signatures. We hope future work on constructing elliptic curves or higher genus curves will help in solving this problem.

## Acknowledgments

The authors thank Steven Galbraith, Alice Silverberg, Moni Naor, Victor Shoup, Scott Renfro, Paulo Barreto, and Doug Kuhlman for helpful discussions about this work.

## References

[1] ANSI X9 Committee. DSTU X9.59-2000: Electronic Commerce for the Financial Services Industry: Account-Based Secure Payment Objects. `http://www.x9.org/`.

[2] ANSI X9.62 and FIPS 186-2. Elliptic Curve Digital Signature Algorithm, 1998.

[3] R. Balasubramanian and N. Koblitz. The Improbability That an Elliptic Curve Has Subexponential Discrete Log Problem under the Menezes-Okamoto-Vanstone Algorithm. *J. Cryptology*, 11(2):141–5, 1998.

[4] P. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In M. Yung, editor, *Proceedings of Crypto 2002*, volume 2442 of *LNCS*, pages 354–68. Springer-Verlag, 2002.

[5] P. Barreto, B. Lynn, and M. Scott. Constructing Elliptic Curves with Prescribed Embedding Degrees. Cryptology ePrint Archive, Report 2002/088, 2002. `http://eprint.iacr.org/`.

[6] M. Bellare, J. Garay, and T. Rabin. Fast Batch Verification for Modular Exponentiation and Digital Signatures. In K. Nyberg, editor, *Proceedings of Eurocrypt 1998*, volume 1403 of *LNCS*, pages 236–50. Springer-Verlag, 1998.

[7] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures: How to Sign with RSA and Rabin. In U. Maurer, editor, *Proceedings of Eurocrypt 1996*, volume 1070 of *LNCS*, pages 399–416. Springer-Verlag, 1996.

[8] I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic Curves in Cryptography*, volume 265 of *London Mathematical Society Lecture Notes*. Cambridge University Press, 1999.

[9] A. Boldyreva. Efficient Threshold Signature, Multisignature, and Blind Signature Schemes, Based on the Gap Diffie-Hellman Group Signature Scheme. In Y. Desmedt, editor, *Proceedings of PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer-Verlag, 2003.

[10] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing*, 32(3):586–615, 2003. Extended abstract in *Proceedings of Crypto 2001*.

[11] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps, 2003.

[12] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *Proceedings of Asiacrypt 2001*, volume 2248 of *LNCS*, pages 514–32. Springer-Verlag, 2001.

[13] D. Chaum and T. Pederson. Wallet Databases with Observers. In E. Brickell, editor, *Proceedings of Crypto 1992*, volume 740 of *LNCS*, pages 89–105. Springer-Verlag, 1992.

[14] D. Coppersmith. Fast Evaluation of Logarithms in Fields of Charateristic Two. *IEEE Tran. Info. Th.*, 30(4):587–94, 1984.

[15] J.-S. Coron. On the Exact Security of Full Domain Hash. In M. Bellare, editor, *Proceedings of Crypto 2000*, volume 1880 of *LNCS*, pages 229–35. Springer-Verlag, 2000.

[16] J.-S. Coron. Optimal Security Proofs for PSS and Other Signature Schemes. In L. Knudsen, editor, *Proceedings of Eurocrypt 2002*, volume 2332 of *LNCS*, pages 272–87. Springer-Verlag, 2002.

[17] C. Diem. The GHS-Attack in Odd Characteristic. Preprint, 2001. `http://www.exp-math.uni-essen.de/~diem/english.html`.

[18] G. Frey, M. Muller, and H. Ruck. The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems. *IEEE Tran. Info. Th.*, 45(5):1717–9, 1999.

[19] S. Galbraith. Supersingular Curves in Cryptography. In C. Boyd, editor, *Proceedings of Asiacrypt 2001*, volume 2248 of *LNCS*, pages 495–513. Springer-Verlag, 2001.

[20] S. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate Pairing. In C. Fieker and D. Kohel, editors, *Proceedings of ANTS V*, volume 2369 of *LNCS*, pages 324–37. Springer-Verlag, 2002.

[21] S. Galbraith and N. P. Smart. A Cryptographic Application of Weil Descent. In M. Walker, editor, *Cryptology and Coding*, volume 1746 of *LNCS*, pages 191–200. Springer-Verlag, 1999.

[22] P. Gaudry, F. Hess, and N. P. Smart. Constructive and Destructive Facets of Weil Descent on Elliptic Curves. Technical Report CSTR-00-016, Department of Computer Science, University of Bristol, 2000.

[23] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In J. Stern, editor, *Proceedings of Eurocrypt 1999*, volume 1592 of *LNCS*, pages 295–310. Springer-Verlag, 1999.

[24] S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks. *SIAM J. Computing*, 17(2):281–308, 1988.

[25] ISO TC86 Committee. ISO 8583: Financial Transaction Card Originated Messages — Interchange Message Specifications. `http://www.tc68.org/`.

[26] A. Joux. A One Round Protocol for Tripartite Diffie-Hellman. In W. Bosma, editor, *Proceedings of ANTS IV*, volume 1838 of *LNCS*, pages 385–94. Springer-Verlag, 2000.

[27] A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in Cryptographic Groups. Cryptology ePrint Archive, Report 2001/003, 2001. `http://eprint.iacr.org/`.

[28] N. Koblitz. An Elliptic Curve Implementation of the Finite Field Digital Signature Algorithm. In H. Krawczyk, editor, *Proceedings of Crypto 1998*, volume 1462 of *LNCS*, pages 327–33. Springer-Verlag, 1998.

[29] S. Lang. *Elliptic Functions.* Addison-Wesley, 1973.

[30] U. Maurer. Towards the Equivalence of Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms. In Y. Desmedt, editor, *Proceedings of Crypto 1994*, volume 839 of *LNCS*, pages 271–81. Springer-Verlag, 1994.

[31] A. Menezes, editor. *Elliptic Curve Public Key Cryptosystems.* Kluwer Academic Publishers, 1993.

[32] A. Menezes, T. Okamoto, and P. Vanstone. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. *IEEE Trans. Information Theory*, 39(5):1639–46, 1993.

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

[34] V. Miller. Short Programs for Functions on Curves. Unpublished manuscript, 1986.

[35] I. Mironov. A Short Signature as Secure as DSA. Unpublished manuscript, 2001.

[36] A. Miyaji, M. Nakabayashi, and S. Takano. New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. *IEICE Trans. Fundamentals*, E84-A(5):1234–43, 2001.

[37] D. Naccache and J. Stern. Signing on a Postcard. In *Proceedings of Financial Cryptography 2000*, 2000.

[38] K. Nyberg and R. Rueppel. Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem. *Design Codes and Cryptography*, 7:61–81, 1996.

[39] T. Okamoto and D. Pointcheval. The Gap Problems: A New Class of Problems for the Security of Cryptographic Primitives. In K. Kim, editor, *Proceedings of PKC 2001*, volume 1992 of *LNCS*, pages 104–18. Springer-Verlag, 2001.

[40] J. Patarin, N. Courtois, and L. Goubin. QUARTZ, 128-Bit Long Digital Signatures. In D. Naccache, editor, *Proceedings of RSA 2001*, volume 2020 of *LNCS*, pages 282–97. Springer-Verlag, 2001.

[41] L. Pintsov and S. Vanstone. Postal Revenue Collection in the Digital Age. In *Proceedings of Financial Cryptography 2000*, 2000.

[42] K. Rubin and A. Silverberg. Supersingular Abelian Varieties in Cryptology. In M. Yung, editor, *Proceedings of Crypto 2002*, volume 2442 of *LNCS*, pages 336–53. Springer-Verlag, 2002.

[43] O. Schirokauer, D. Weber, and T. Denny. Discrete Logarithms: The Effectiveness of the Index Calculus Method. In H. Cohen, editor, *Proceedings of ANTS II*, volume 1122 of *LNCS*, pages 337–61. Springer-Verlag, 1996.

[44] J. H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, 1986.

[45] N. Smart, editor. *The Algorithmic Resolution of Diophantine Equations*. Cambridge University Press, 1998.

[46] E. R. Verheul. Self-Blindable Credential Certificates from the Weil Pairing. In C. Boyd, editor, *Proceedings of Asiacrypt 2001*, volume 2248 of *LNCS*, pages 533–51. Springer-Verlag, 2001.

[47] W. C. Waterhouse. Abelian Varieties over Finite Fields. *Ann. Sci. École Norm. Sup.*, 2:521–60, 1969.