Short signatures from the Weil pairing

Dan Boneh*, Ben Lynn, and Hovav Shacham

Computer Science Department, Stanford University
{dabo,blynn,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. The signature length is half the size of a DSA signature for a similar level of security. Our short signature scheme is designed for systems where signatures are typed in by a human or signatures are sent over a low-bandwidth channel.

1 Introduction

Short digital signatures are needed in environments where a human is asked to manually key in the signature. For example, product registration systems often ask users to key in a signature provided on a CD label. More generally, short signatures are needed in low-bandwidth communication environments. For example, short signatures are needed when printing a signature on a postage stamp [21, 19]. Currently, the two most frequently used signatures schemes, RSA and DSA, provide 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 [1]. A 320-bit signature is too long to be keyed in by a human.

We propose a signature scheme whose length is approximately 160 bits and provides a level of security similar to 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 of characteristic three. 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 elliptic curves. Consequently, there is no equivalent of our scheme in $\mathbb{F}_p^*$.

Due to the properties of the curves we use, currently we can only provide signatures of the lengths given below. The best known algorithm for solving the CDH problem in these groups requires a discrete-log on a finite field of characteristic three. The size of this field is given (in bits) in the rightmost column of the table below.

* Supported by NSF and the Packard Foundation.

Signature size (bits)	EC group size (bits)	Discrete-log Security (bits)
126	126	752
154	151	923
237	220	1417
259	256	1551
265	262	1589

The second row shows that we can get a signature of length 154 bits with security comparable to 320-bit DSA or 320-bit ECDSA. The best known algorithm to forge a 154-bit signature requires one to solve a CDH problem in a finite field of size 923 bits or on an elliptic curve group of size 151 bits. In Section 3.5 we outline an approach for generalizing our technique and building signatures of any length.

Constructing short signatures is an old problem. Several proposals show how to shorten the DSA signature scheme while preserving the same level of security. Naccache and Stern [19] propose a variant of DSA where the signature length is approximately 240 bits. Mironov [18] 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 the DSA signature length is signatures with message recovery [21]. In such systems one encodes a part of the message into the signature thus shortening the total length of the message-signature pair. For long messages, one can then achieve a DSA signature overhead of length 160 bits. However, for very short messages (e.g., 64 bits) the total length is still 320 bits. Using our signature scheme, the signature length is always on the order of 160 bits, no matter how short the message is. Note that when the only transmitted data is the signature (the message is not transmitted) DSA signatures with message recovery are not any shorter than standard DSA signatures.

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 [12] and was previously used in [11, 4]. We call such groups Gap Diffie-Hellman groups, or GDH groups for short. Okamoto and Pointcheval [20] commented that a Gap Diffie-Hellman group gives rise to a signature scheme. However, most Gap Diffie-Hellman groups are relatively long and do not lead to short signatures. We prove the security of signatures schemes derived from GDH groups and show how they lead to very short signatures. We experiment with our proposed signature scheme and give running times in Section 5.

2 Signature schemes based on Gap-Diffie-Hellman

We present a signature scheme that works in any Gap Diffie-Hellman group. As mentioned above, this scheme is described implicitly by Okamoto and Pointcheval [20]. The scheme resembles the undeniable signature scheme proposed by Chaum and Pederson [5]. In the next section we show how this signature scheme gives rise to very short signatures.

2.1 Gap Diffie-Hellman Groups (GDH groups)

Consider a (multiplicative) cyclic group $G = \langle g \rangle$, with $p = |G|$ a prime. We are interested in three problems on G.

Group Action Given $u, v \in G$, find uv.
Decision Diffie-Hellman For $a, b, c \in \mathbb{Z}_p^*$, given (g, g^a, g^b, g^c) decide whether $c = ab$.
Computational Diffie-Hellman For $a, b \in \mathbb{Z}_p^*$, given (g, g^a, g^b), compute g^{ab}.

We define a Gap Diffie-Hellman group, in stages.

Definition 1. *G is a τ-decision group for Diffie-Hellman if the group action can be computed in one time unit, and Decision Diffie-Hellman can be computed on G in time at most τ.*

Definition 2. *The advantage of an algorithm $\mathcal{A}$ in solving the Computational Diffie-Hellman problem in a group G is*

$$\mathsf{Adv\,CDH}_{\mathcal{A}} \overset{\text{def}}{=} \Pr\left[\mathcal{A}(g, g^a, g^b) = g^{ab} : a, b \overset{\text{R}}{\leftarrow} \mathbb{Z}_p^*\right]$$

Where the probability is over the choice of a and b, and the coin tosses of $\mathcal{A}$. We say that an algorithm $\mathcal{A}$ (t, ϵ)-breaks Computational Diffie-Hellman in G if $\mathcal{A}$ runs in time at most t, and $\mathsf{Adv\,CDH}_{\mathcal{A}} \geq \epsilon$.

Definition 3. *A prime order group G is a (τ, t, ϵ)-GDH group if it is a τ-decision group for Diffie-Hellman and no algorithm (t, ϵ)-breaks Computational Diffie-Hellman on it.*

2.2 The GDH Signature Scheme

The GDH Signature Scheme allows the creation of signatures on arbitrary messages $m \in \{0,1\}^*$. A signature σ is an element of G. The base group G and the generator g are system parameters. We denote by G^* the set $G^* = G \setminus \{1\}$ where 1 is the identity of G.

The signature scheme comprises three algorithms, *KeyGen*, *Sign*, and *Verify*. It makes use of a full-domain hash function $h : \{0,1\}^* \to G^*$. The security analysis views h as a random oracle [3]. In Section 3.3 we weaken the requirement on the full-domain hash.

Key Generation Pick random $x \overset{\text{R}}{\leftarrow} \mathbb{Z}_p^*$, and compute $v \leftarrow g^x$. The public key is v. The secret key is x.
Signing Given a secret key x, and a message $M \in \{0,1\}^*$, Compute $h \leftarrow h(M)$, and $\sigma \leftarrow h^x$. The signature is $\sigma \in G^*$.
Verification Given a public key v, a message M, and a signature σ, compute $h \leftarrow h(M)$ and verify that (g, v, h, σ) is a valid Diffie-Hellman tuple.

Note that a GDH signature is a single element of G^*. Hence, to construct short signatures we need a GDH group where elements have a short representation. We construct such groups in Section 3.

2.3 Security

We show the security of the GDH signature scheme against existential forgery, under chosen-message attacks.

Definition 4. *The advantage in existentially forging a signature of a forger algorithm $\mathcal{F}$, given access to a signing oracle S, is*

$$\mathsf{Adv\,Sig}_{\mathcal{F}} \overset{\mathrm{def}}{=} \Pr\left[\mathit{Verify}(PK, M, \sigma) = \texttt{valid} \ : \ \begin{array}{l} (PK, SK) \overset{\mathrm{R}}{\leftarrow} \mathit{KeyGen}, \\ (M, \sigma) \overset{\mathrm{R}}{\leftarrow} \mathcal{F}^S(PK) \end{array} \right]$$

The probability is taken over the coin tosses of the key-generation algorithm, and of the forger.

Here the adversary $\mathcal{F}$ is allowed to query the signing oracle adaptively: any of its queries may depend on previous answers, but it may not emit a signature for a message on which it had previously queried the oracle. The adversary also has access to the full-domain hash function, which is treated as a random oracle.

Definition 5. *A forger $\mathcal{F}$ (t, q_H, q_S, ϵ)-breaks a signature scheme if $\mathcal{F}$ runs in time at most t, makes at most q_H queries to the hash function and at most q_S queries to the signing oracle S, and* $\mathsf{Adv\,Sig}_{\mathcal{F}} \geq \epsilon$.

Definition 6. *A signature scheme is (t, q_H, q_S, ϵ)-secure against existential forgery on adaptive chosen-message attacks if no forger (t, q_H, q_S, ϵ)-breaks it.*

The following theorem shows that the GDH signature scheme is secure. The proof of the theorem is given in Section 4.

Theorem. *Let G be a (τ, t', ϵ')-gap group for Diffie-Hellman of order p. Then the Gap Signature Scheme on G is (t, q_H, q_S, ϵ)-secure against existential forgery on adaptive chosen-message attacks, where*

$$t \leq t' - 2c_{\mathcal{A}}(\lg p)(q_H + q_S) \quad \textit{and} \quad \epsilon \geq 2e \cdot q_S \epsilon',$$

and $c_{\mathcal{A}}$ is a small constant. Here e is the base of the natural logarithm.

3 Building Gap-Diffie-Hellman groups with small representations

Using the Weil pairing, certain elliptic curves may be used as GDH groups. We recall some necessary facts about elliptic curves (see, e.g., [14, 22]), and then show how to use certain curves for GDH signatures. In particular, we describe the curves $y^2 = x^3 + 2x \pm 1$ over $\mathbb{F}_{3^\ell}$.

3.1 Elliptic Curves and the Weil Pairing

An elliptic curve can serve as the basis for a GDH signature scheme if we can use it to construct some group G with large prime order on which Computational Diffie-Hellman is difficult, but Decision Diffie-Hellman is easy. First, we characterize a necessary condition for CDH intractability on a subgroup of E.

Definition 7. *Let p be a prime, l a positive exponent, and E an elliptic curve over $\mathbb{F}_{p^l}$ with m points. Let P in E be a point of prime order q where $q^2 \nmid m$. We say that the subgroup $\langle P \rangle$ has a security multiplier α, for some integer $\alpha > 0$, if the order of p^l in $\mathbb{F}_q^*$ is α. In other words:*

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

It is well known (as shown below) that for CDH to be hard in the subgroup $\langle P \rangle$ we must have that the security multiplier, α, for this subgroup is not too small. On the other hand, to get an efficient Decision Diffie-Hellman algorithm in $\langle P \rangle$ we need that α is not too large. Therefore, the problem in constructing short signatures is to find curves for which α is sufficiently large for security, but sufficiently small for efficiency. Using current security parameters, $\alpha = 6$ is sufficient for obtaining short signatures. It is an open problem to build elliptic curves with slightly higher α, say $\alpha = 10$ (see Section 3.5).

Discrete-log on elliptic curves: Let $\langle P \rangle$ be a subgroup of $E/\mathbb{F}_{p^l}$ of order q with security multiplier α. We briefly discuss two standard ways for computing discrete-log in $\langle P \rangle$.

1. **MOV:** Use an efficiently computable homomorphism, as in the Menezes-Okamoto-Vanstone reduction [15], to map the discrete log problem in $\langle P \rangle$ to a discrete log problem in some extension of $\mathbb{F}_{p^l}$, say $\mathbb{F}_{p^{li}}$. We require that the image of $\langle P \rangle$ under this homomorphism is a subgroup of $\mathbb{F}_{p^{li}}^*$ of order q. Thus we have $q | (p^{il} - 1)$, which by the definition of α 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 $\mathbb{F}_{p^{l\alpha}}^*$. Therefore, to ensure that discrete log is hard in $\langle P \rangle$ we want curves with large α.
2. **Generic:** Generic discrete log algorithms such as the Baby-Step-Giant-Step and Pollard's Rho method [16] have a running time proportional to $\sqrt{q}$. Therefore, we must ensure that q is sufficiently large.

Decision Diffie-Hellman on elliptic curves: Let $P \in E/\mathbb{F}_{p^l}$ be a point of prime order q. Suppose the subgroup $\langle P \rangle$ has security multiplier α. We assume $q \nmid p^l - 1$. A result of Balasubramanian and Koblitz [2] shows that $E/\mathbb{F}_{p^{l\alpha}}$ contains a point Q that is linearly independent of P. Such a point $Q \in E/\mathbb{F}_{p^{l\alpha}}$ can be efficiently found. Note that linear independence of P and Q can be verified via the Weil pairing described below.

With two linearly independent points $P \in E/\mathbb{F}_{p^l}$ and $Q \in E/\mathbb{F}_{p^{l\alpha}}$, each of order q, we can use the Weil pairing to answer certain questions that will allow us to construct a DDH oracle [12]. Let $E[q]$ denote the subgroup of $E/\mathbb{F}_{p^{l\alpha}}$ generated by P and Q. The Weil pairing is a map $e : E[q] \times E[q] \to \mathbb{F}^*_{p^{l\alpha}}$ with the following properties:

1. Identity: for all $R \in E[q]$, $e(R, R) = 1$.
2. Bilinear: for all $R_1, R_2 \in E[q]$ and $a, b \in \mathbb{Z}$ we have that $e(aR_1, bR_2) = e(R_1, R_2)^{ab}$.
3. Non-degenerate: if for $R \in E[q]$ we have $e(R, R') = 1$ for all $R' \in E[q]$, then $R = \mathcal{O}$.
4. Computable: for all $R_1, R_2 \in E[q]$, the pairing $e(R_1, R_2)$ can be computed efficiently [17].

Note that $e(R_1, R_2) = 1$ if and only if R_1 and R_2 are linearly dependent.

For the linearly independent points P and Q, both of order q, the Weil pairing allows us to determine whether the tuple (P, aP, Q, bQ) is such that $a = b \bmod q$; indeed,

$$a = b \bmod q \quad \Longleftrightarrow \quad e(P, bQ) = e(aP, Q).$$

Suppose we also have a computable isomorphism ϕ from $\langle P \rangle$ to $\langle Q \rangle$. Necessarily, ϕ is such that, for all a, $\phi(aP) = axQ$, where $xQ = \phi(P)$. In this case, the Weil pairing allows us to determine whether the tuple (P, aP, bP, cP) is such that $ab = c \bmod q$:

$$ab = c \bmod q \quad \Longleftrightarrow \quad e(P, \phi(cP)) = e(aP, \phi(bP)).$$

With the isomorphism ϕ, the Weil pairing provides an algorithm for Decision Diffie-Hellman. Note that the algorithm for DDH requires two evaluations of the Weil pairing for points over $\mathbb{F}_{p^{l\alpha}}$.

3.2 A Special Supersingular Curve

Using the machinery of Section 3.1, we derive GDH groups with small representation from the supersingular elliptic curves E given by $y^2 = x^3 + 2x \pm 1$ over $\mathbb{F}_{3^l}$. As we will see, these are unique supersingular elliptic curves with security multiplier 6. Hence, the MOV reduction maps the discrete log problem in $E/\mathbb{F}_{3^l}$ to $\mathbb{F}^*_{3^{6l}}$. This means that we can use relatively small values of l to obtain short signatures, but the security is dependent on a discrete log problem in a large finite field. We use two simple lemmas to describe the behavior of these curves (see also [23, 13]).

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

$$\#E^+/\mathbb{F}^{3^l} = \begin{cases} 3^l + 1 + \sqrt{3 \cdot 3^l} \text{ when } l = \pm 1 \bmod 12, \text{ and} \\ 3^l + 1 - \sqrt{3 \cdot 3^l} \text{ when } l = \pm 5 \bmod 12 \end{cases}$$

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

$$\#E^-/\mathbb{F}^{3^l} = \begin{cases} 3^l + 1 - \sqrt{3 \cdot 3^l} \text{ when } l = \pm 1 \bmod 12, \text{ and} \\ 3^l + 1 + \sqrt{3 \cdot 3^l} \text{ when } l = \pm 5 \bmod 12 \end{cases}$$

Proof. See [13, section 2]. □

We have thus shown how to construct an elliptic curve with $3^l + 1 \pm \sqrt{3 \cdot 3^l}$ points over $\mathbb{F}_{3^l}$, simply by selecting one of E^- and E^+ as appropriate, whenever $l \bmod 12$ equals ± 1 or ± 5.

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

Proof. We have $x^6 - 1 = (x^3-1)(x^3+1) = (x-1)(x^2+x+1)(x+1)(x^2-x+1)$, so for any integer x it follows that $(x^2 - x + 1) \mid (x^6 - 1)$. In particular, when $x = 3^l$, we see that $(3^{2l} - 3^l + 1) \mid (3^{6l} - 1)$. Now when E is an elliptic curve as above, we know that $\#(E/\mathbb{F}_{3^l})$ is either $3^l + 1 + \sqrt{3 \cdot 3^l}$ or $3^l + 1 - \sqrt{3 \cdot 3^l}$. But $\left((3^l+1) + \sqrt{3 \cdot 3^l}\right)\left((3^l+1) - \sqrt{3 \cdot 3^l}\right) = 3^{2l} - 3^l + 1$. Thus $\#(E/\mathbb{F}_{3^l}) \mid (3^{6l} - 1)$.

Together, Lemmas 1 and 2 show that, for the relevant values of l, the curves $E^+/\mathbb{F}_{3^l}$ and $E^-/\mathbb{F}_{3^l}$ will have security parameters α 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 $E^+, E^-/\mathbb{F}_{3^{6l}}$:* For l such that $l \bmod 12$ equals ± 1 or ± 5, compute three elements of $\mathbb{F}_{3^{6l}}$, 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 $\mathbb{F}_{3^{6l}}$:

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

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

Proof. See Silverman [22, p. 326]. □

For a point P of order q on any of these curves, the appropriate automorphism allows us to solve a Decision Diffie-Hellman question on $G = \langle P \rangle$, as we have shown in the previous section.

3.3 Hashing onto Elliptic Curves

The GDH signature scheme needs a hash function $h : \{0,1\}^* \to G^*$ where G is a GDH group. We are proposing to use a subgroup of an elliptic curve as a GDH group. 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 $E/\mathbb{F}_{p^l}$ be an elliptic curve of order m defined by $y^2 = f(x)$. Let $P \in E/\mathbb{F}_{p^l}$ be a point of prime order q, where $q^2 \nmid m$. We wish to use the subgroup $G = \langle P \rangle$ as a GDH group for the GDH signature scheme. Suppose we are given a hash function $h' : \{0,1\}^* \to \mathbb{F}_{p^l} \times \{0,1\}$. 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^*. Fix a small parameter $I = \lceil \log_2 \log_2(1/\delta) \rceil$, where δ is some desired bound on the probability of failure.

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

```
1. Given $M \in \{0,1\}^*$, set $i \leftarrow 0$;
2. Set $(x,b) \leftarrow h'(i \parallel M) \in \mathbb{F}_{p^l} \times \{0,1\}$;
3. If $f(x)$ is a quadratic residue in $\mathbb{F}_{p^l}$ then do:
   3a. Let $y_0, y_1 \in \mathbb{F}_{p^l}$ be the two square roots of $f(x)$. We use $b \in \{0,1\}$ to
       choose between these roots. View $y_0, y_1$ as polynomials of degree $l-1$
       over $\mathbb{F}_p$. Then ensure that the constant term of $y_0$ is not greater than the
       constant term of $y_1$ when viewed as integers in $[0,p]$ (swapping $y_0$ and $y_1$
       if necessary). Set $\tilde{P}_M \in E/\mathbb{F}_{p^l}$ to be the point $\tilde{P}_M = (x, y_b)$.
   3b. Compute $P_M = (m/q)\tilde{P}_M$. Then $P_M$ is in $G$.
       If $P_M$ is in $G^*$ then output $MapToGroup_{h'}(M) = P_M$ and stop.
4. Otherwise, increment $i$, and goto 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 $1/2^{2^I} \le \delta$.

Lemma 4. *Suppose the GDH signature scheme is (t, q_H, q_S, ϵ)-secure in the subgroup G when using a random hash function $h : \{0,1\}^* \to G^*$. Then it is $(t - 2^I q_H \lg m, q_H, q_S, \epsilon)$-secure when the hash function h is computed with MapToGroup$_{h'}$ where h' is a random hash function $h' : \{0,1\}^* \to \mathbb{F}_{p^l} \times \{0,1\}$.*

Proof Sketch: Suppose a forger algorithm $\mathcal{F}'$ (t, q_H, q_S, ϵ)-breaks the Gap Signature Scheme on the subgroup G when the hash function h is computed using *MapToGroup*$_{h'}$. We construct an algorithm $\mathcal{F}$ that $(t + 2^I q_H \lg m, q_H, q_S, \epsilon)$-breaks the scheme when h is a random oracle $h : \{0,1\}^* \to G^*$.

Our new forger $\mathcal{F}$ will run $\mathcal{F}'$ as a black box. $\mathcal{F}$ will use its own hash oracle $h : \{0,1\}^* \to G^*$ to simulate for $\mathcal{F}'$ the behavior of *MapToGroup*$_{h'}$. It uses an array s_{ij}, of elements of $\mathbb{F}_{p^l} \times \{0,1\}$. The array has q_H rows and 2^I columns. On initialization, $\mathcal{F}$ fills s_{ij} with uniformly-selected elements of $\mathbb{F}_{p^l} \times \{0,1\}$.

$\mathcal{F}$ 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 $\mathcal{F}$ had not previously seen (and whose w is an arbitrary I-bit string), $\mathcal{F}$ scans the row s_{ij}, $0 \le j < 2^I$. 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^*, $\mathcal{F}$ replaces s_{ij} with a different point (x_i, b_i) defined as follows. Let $Q_i = h(M_i) \in G^*$. Then $\mathcal{F}$ constructs a random $\tilde{Q}_i = (x_i, y_i) \in E/\mathbb{F}_{p^l}$ such that $(m/q)\tilde{Q}_i = Q_i$. It 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 $\mathit{MapToGroup}_{h'}(M_i) = h(M_i)$ 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 $\mathit{MapToGroup}_{h'}$ 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 *MapToGroup*. □

3.4 A concrete short signature scheme

To summarize things so far, we describe a concrete signature scheme using the GDH group derived from the curve $E/\mathbb{F}_{3^l}$ defined by $y^2 = x^3+2x\pm1$. Some useful instantiations of these curves are presented in Table 1. Note that we restrict these instantiations to those where l is prime, to avoid Weil-descent attacks [9, 10]. As explained in Section 3.3, we use $\mathit{MapToGroup}_{h'}$ to map arbitrary bit strings to points of order q on E, using a hash function h' from arbitrary strings to elements of $\mathbb{F}_{p^l}$ and an extra bit.

curve	l	Signature Size	DLog Security	Multiplier	MOV Security
		$\lceil \lg_2 m \rceil$	$\lceil \lg_2 q \rceil$	α	$\lceil \lg_2 x \rceil$
E^-	79	126	126	6	752
E^+	97	154	151	6	923
E^+	149	237	220	6	1417
E^+	163	259	256	6	1551
E^-	163	259	259	6	1551
E^+	167	265	262	6	1589

Table 1. Supersingular elliptic curves for GDH Signatures. Here $m = \#(E/\mathbb{F}_{3^l})$, and q is the largest prime dividing m. The MOV reduction maps the curve onto a field with x elements.

A concrete signature scheme:

Key generation Given one of the values l in Table 1, let $E/\mathbb{F}_{3^l}$ be the corresponding curve and let q be the largest prime factor of the order of the curve. Let $P \in E/\mathbb{F}_{3^l}$ be a point of order q. pick a random $x \in \mathbb{Z}_q^*$ and set $R \leftarrow xP$. Then (l, q, P, R) is the public key and x is the private key.

Signing To sign a message $M \in \{0,1\}^*$ use algorithm $\mathit{MapToGroup}_{h'}$ to map M to a point $P_M \in \langle P \rangle$. Set $S_M \leftarrow xP_M$. The signature σ is the x coordinate of S_M. Therefore, $\sigma \in \mathbb{F}_{3^l}$.

Verification Given a public key (l, q, P, R), a message M, and a signature σ do:

1. Find a point $S \in E/\mathbb{F}_{3^l}$ of order q whose x-coordinate is σ and whose y-coordinate is y for some $y \in \mathbb{F}_{3^l}$. If no such point exists reject the signature as invalid.
2. Set $u \leftarrow e(P, \phi(S))$ and $v \leftarrow e(R, \phi(h(M)))$, where e is the Weil pairing on the curve $E/\mathbb{F}_{3^{6l}}$ and $\phi : E \to E$ is the automorphism of the curve described in Lemma 3.
3. If either $u = v$ or $u^{-1} = v$, accept the signature. Otherwise, reject.

Note that both (σ, y) and $(\sigma, -y)$ are points on $E/\mathbb{F}_{3^l}$ that have σ as their x-coordinate. Either one of these two points can be the point S_M used to generate the signature in the signing algorithm. Indeed, since $(\sigma, y) = -(\sigma, -y)$ on the curve, we have that $e(P, \phi(-S)) = e(P, \phi(S))^{-1}$. Therefore, $u = v$ tests that $(P, R, h(M), S)$ is a Diffie-Hellman tuple, while $u^{-1} = v$ tests that $(P, R, h(M), -S)$ is a Diffie-Hellman tuple.

The next lemma shows that an attacker capable of existential forgery under a chosen message attack (in the random oracle model) is also capable of solving the Diffie-Hellman problem in $E/\mathbb{F}_{3^l}$.

Lemma 5. *Suppose $E/\mathbb{F}_{3^l}$ is one of the curves given in Table 1, q is the largest prime dividing $\#E$, P is a point of order q on E, and no algorithm (t_0, ϵ_0)-breaks Computational Diffie-Hellman on $G = \langle P \rangle$. Let $h' : \{0,1\}^* \to \mathbb{F}_{3^l} \times \{0,1\}$ be a random oracle. Then the concrete signature scheme described above is (t, q_H, q_S, ϵ)-secure against existential forgery on adaptive chosen-message attacks (in the random oracle model), where*

$$t \leq t_0 - 2c_{\mathcal{A}}(\lg q)(q_H + q_S) - 2^I q_H \lg m - 2\tau \quad \textit{and} \quad \epsilon \geq 2e \cdot q_S \epsilon_0,$$

and $c_{\mathcal{A}}$ is a small constant.

Proof. By assumption, G is a (τ, t_0, ϵ_0)-GDH group, where τ is equal to twice the time necessary to compute the Weil pairing on G. Assuming the existence of a random oracle h from arbitrary bit strings to G^*, the generic GDH signature scheme (given in Section 2.2) on G is $(t_1, q_H, q_S, \epsilon_1)$-secure against existential forgery on adaptive chosen-message attacks by the main theorem (Section 4), where

$$t_1 \leq t_0 - 2c_{\mathcal{A}}(\lg q)(q_H + q_S) \quad \text{and} \quad \epsilon_1 \geq 2e \cdot q_S \epsilon_0, \qquad (*)$$

and $c_{\mathcal{A}}$ is a small constant.

By Section 3.3, we can construct a hash function h onto G^* from the hash function h'. By Lemma 4, the generic GDH signature scheme on G, using algorithm *MapToGroup*$_{h'}$ is $(t_2, q_H, q_S, \epsilon_2)$-secure against existential forgery on adaptive chosen-message attacks by the main theorem (Section 4), where

$$t_2 = t_1 - 2^I q_H \lg m \quad \text{and} \quad \epsilon_2 = \epsilon_1. \qquad (**)$$

The only difference between the generic GDH signature scheme on G and the concrete scheme on G described above is that signatures in the latter scheme are elements of $\mathbb{F}_{3^l}$, rather than G. Given an adversary $\mathcal{F}$ that breaks the concrete scheme, we can construct an algorithm $\mathcal{A}$ that breaks the generic scheme, as follows. The public key is identical in the two schemes, so $\mathcal{A}$ simply provides $\mathcal{F}$ with the R given to it. Hashes are identical in the two schemes, so $\mathcal{A}$ passes $\mathcal{F}$'s hash requests to its own hash oracle, and provides $\mathcal{F}$ with the answer. When $\mathcal{F}$ requests a signature on a message M, $\mathcal{A}$ obtains the signature $S \in E$ from its signature oracle, and gives $\mathcal{F}$ the x-coordinate σ of S. Finally, when $\mathcal{F}$ outputs a forgery σ^* (for the concrete scheme) on a message M^*, $\mathcal{A}$ finds a point $S^* \in E$ whose x-coordinate is σ^*. By the discussion above, either $(P, R, h(M^*), S^*)$ is a Diffie-Hellman tuple, in which case S^* is a signature on M^* in the concrete scheme, or $(P, R, h(M^*), -S^*)$ is a Diffie-Hellman tuple, in which case $-S^*$ is a signature on M^* in the concrete scheme. $\mathcal{A}$ outputs M^* along with the appropriate one of S^* and $-S^*$.

The additional time required for this simulation is dominated by the two additional signature verifications, each of which takes time τ. Thus if the generic GDH scheme is $(t_2, q_H, q_S, \epsilon_2)$-secure, the concrete GDH scheme is $(t_3, q_H, q_S, \epsilon_3)$-secure, where

$$t_3 = t_2 - 2\tau \quad \text{and} \quad \epsilon_3 = \epsilon_2. \tag{$***$}$$

Combining $(*)$, $(**)$, and $(***)$ yields the required reduction. $\square$

3.5 An open problem: short signatures with high security

In the previous section we proposed using a supersingular curve over $\mathbb{F}_{3^\ell}^*$ to build a short signature scheme as secure as discrete log in $\mathbb{F}_{3^{6\ell}}^*$. However, there is no reason to stick with supersingular curves. Using other elliptic or hyper-elliptic curves it might be possible to achieve even higher security multipliers.

In Section 3.2, we showed that the curves E^+ and E^- over $\mathbb{F}_{3^l}$ have security parameter α at most 6. This is, in fact, the maximum value of α for any supersingular curve [15, 23]. Instantiating the GDH signature scheme on (necessarily non-supersingular) elliptic curves with slightly higher values of α would increase the work required for verification, but also increase security against MOV-related attacks at comparable signature bit lengths.

Consider an elliptic curve $E/\mathbb{F}_{p^l}$ with m points, a large prime $q \mid m$, a security parameter α for the subgroup of order q, and two linearly independent points, P and Q, of order q, where $P \in E/\mathbb{F}_{p^l}$, and $Q \in E/\mathbb{F}_{p^{l\alpha}}$. Note that a point $Q \in E/\mathbb{F}_{p^{l\alpha}}$ linearly independent of P must exist by [2] assuming $q \nmid p^l - 1$. For such a curve, there is not necessarily an automorphism that maps between $\langle P \rangle$ and $\langle Q \rangle$. We therefore slightly modify the Gap Signature Scheme to use the two groups together.

It is easy to decide whether a tuple (P, aP, Q, bQ) is such that $a = b$, using the Weil pairing. We call this the co-Decision Diffie-Hellman problem, and it has an obvious computational variant: given the tuple (P, Q, aQ), compute aP. The modified (co-gap) signature scheme is as follows.

Key Generation Let $P \in E/\mathbb{F}_{p^l}$ and $Q \in E/\mathbb{F}_{p^{l\alpha}}$ be two linearly independent points of prime order q as described above. Pick $x \xleftarrow{\text{R}} \mathbb{Z}_q^*$, and compute $R \leftarrow xQ$. The public key is $(E/\mathbb{F}_{p^l}, q, Q, R)$. The secret key is x.

Signing Given a secret key x, and a message $M \in \{0,1\}^*$ use $\mathit{MapToGroup}_{h'}$ to map M to a point $P_M \in \langle P \rangle$. Set $S_M \leftarrow xP_M$. The signature σ is the x-coordinate of S_M, an element of $\mathbb{F}_{p^l}$.

Verification Given a public key $(E/\mathbb{F}_{p^l}, q, Q, R)$, a message M, and a purported signature σ, let S be a point on $E/\mathbb{F}_{p^l}$ of order q whose x-coordinate is σ and whose y-coordinate is y for some $y \in \mathbb{F}_{p^l}$ (if no such point exists reject the signature as invalid). Set $u \leftarrow e(Q, S)$ and $v \leftarrow e(R, h(M))$. If either $u = v$ or $u^{-1} = v$, accept the signature. Otherwise, reject.

By reasoning analogous to that in Section 3.4, the tests in the verification phase ensure that either $(Q, R, h(M), S)$ or $(Q, R, h(M), -S)$ is a valid co-Diffie-Hellman tuple. While the public key, R, is an element of $E/F_{p^{l\alpha}}$, and thus long, a signature σ is an element of E/F_{p^l}, and thus relatively short. The security of this scheme follows from the assumption that no adversary (t, ϵ)-breaks the co-Computational Diffie-Hellman problem.

The challenge, therefore, is to construct elliptic curves with larger values of α, say $\alpha = 10$. It is currently an open problem to build a family of elliptic curves with security multiplier $\alpha = 10$.

Galbraith [8] constructs supersingular curves of higher genus with a "large" security multiplier. For example, the supersingular curve $y^2 + y = x^5 + x^3$ has security multiplier 12 over $\mathbb{F}_{2^l}$. Since a point on the Jacobian of this curve of genus two is characterized by two values in $\mathbb{F}_{2^l}$ (the two x-coordinates in a reduced divisor) the length of the signature is $2l$ bits. Hence, we might obtain a signature of length $2l$ with security of computing CDH in the finite field $\mathbb{F}_{2^{12l}}$. 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 signature lengths beyond those given in Table 1. Since this curve is defined over a field of characteristic two it is better suited for computation than curves defined over of fields of characteristic three. Galbraith shows that Jacobians of genus 2 supersingular curves have a maximum security multiplier of 12. Therefore, genus 2 supersingular curves will not give short signature with higher security. It is an open problem whether one can build a family of hyper-elliptic curves of genus 3 that would give short signatures with higher security.

4 Proof of Security Theorem

We prove, in the random oracle model, that GDH signatures are secure in GDH groups. The proof is similar to that given for full-domain hash RSA signatures by Coron [6], but the presentation is different. The point of this method is that the break-probability ϵ for the signature scheme does not depend on the number of hash queries a forger makes, but only depends on the number of signature queries made by the adversary.

Theorem (Gap Signature Security). *If G is a (τ, t', ϵ')-GDH group, then the Gap Signature Scheme on G is (t, q_H, q_S, ϵ)-secure against existential forgery on* *adaptive chosen-message attacks, where*

$$t \leq t' - 2\tau c_{\mathcal{A}}(q_H + q_S) \quad \textit{and} \quad \epsilon \geq 2e \cdot q_S \epsilon',$$

and $c_{\mathcal{A}}$ is a small constant (in practice, at most 2).

The proof follows, in stages.

4.1 Overview

Assume an algorithm $\mathcal{F}$ (t, q_H, q_S, ϵ)-breaks the Gap Signature Scheme on G. We will use $\mathcal{F}$ to construct an algorithm $\mathcal{A}$ that (τ, t', ϵ') breaks Computational Diffie-Hellman on G, where t' and ϵ' are as above.

Given a forger $\mathcal{F}$ for the GDH group G, we build an algorithm $\mathcal{A}$ that uses $\mathcal{F}$ to break CDH on G. $\mathcal{A}$ is given a challenge (g, g^a, g^b). It uses this challenge to construct a public key that it provides to $\mathcal{F}$. It then allows $\mathcal{F}$ to run. At times, $\mathcal{F}$ makes queries to two oracles, one for message hashes and one for message signatures. These oracles are puppets of $\mathcal{A}$, which it manipulates in constructive ways. Finally, if all goes well, the forgery which $\mathcal{F}$ outputs is transformed by $\mathcal{A}$ into an answer to the CDH challenge.

We assume that $\mathcal{F}$ is well-behaved in the sense that it always requests the hash of a message M before it requests a signature for M, and that it always requests a hash of the message M^* that it outputs as its forgery. It is trivial to modify any forger algorithm $\mathcal{F}$ to have this property.

$\mathcal{A}$ needs to engage in a certain amount of bookkeeping. In particular, it must maintain a list of the messages on which $\mathcal{F}$ requests hashes or signatures. Each message M, as it arrives from $\mathcal{F}$, is assigned an index i; i is obviously bounded above by q_H. The message is stored in M_i, its hash in h_i, and its signature (if available) in σ_i.

4.2 Construction of $\mathcal{A}$

Rather than describe $\mathcal{A}$'s behavior and prove its efficacy *in toto*, we will construct $\mathcal{A}$ in a series of "games," in which increasingly sophisticated $\mathcal{A}$-variants run $\mathcal{F}$; the final variant, $\mathcal{A}_6$, is the $\mathcal{A}$ we seek.

(Each of the $\mathcal{A}$-variants will depend on a probability constant ζ, which will be optimized later, to yield the best possible reduction. Define B_ζ to be the probability distribution over $\{0, 1\}$ where 1 is drawn with probability ζ, and 0 with probability $1 - \zeta$.)

Game 1. $\mathcal{A}_1$ is given a challenge (g, g^a, g^b). In setup, it constructs $PK \leftarrow (g^a)$. Then, for each i, $1 \leq i \leq q_H$, $\mathcal{A}_1$ picks a random bit $s_i \overset{\text{R}}{\leftarrow} B_\zeta$, and a random number $r_i \overset{\text{R}}{\leftarrow} \mathbb{Z}_p^*$. It then sets $h_i \leftarrow g^{r_i}$, and $\sigma_i \leftarrow (g^a)^{r_i}$. Note that (g, g^a, h_i, σ_i) is a valid Diffie-Hellman tuple, so σ_i is a signature on any message whose hash is h_i. $\mathcal{A}_1$ then runs $\mathcal{F}$ with public key PK.

When $\mathcal{F}$ requests a hash on a message M_i, $\mathcal{A}_1$ responds with h_i; when $\mathcal{F}$ requests a signature on a message M_i, $\mathcal{A}_1$ responds with σ_i.

Finally, $\mathcal{F}$ halts, either conceding failure or returning a a forged signature (M^*, σ^*), where $M^* = M_{i^*}$ for some i^* (on which $\mathcal{F}$ had not requested a signature). If $\mathcal{F}$ succeeds in forging, $\mathcal{A}_1$ outputs "`success`"; otherwise, it outputs "`failure`".

The hashes h_i are uniformly distributed in G, so $\mathcal{A}_1$'s hash oracle is a random oracle. Moreover, the signatures σ_i are all valid. In the random oracle model, therefore, $\mathcal{F}$, when run by $\mathcal{A}_1$, behaves exactly as it would when running on its own. Thus

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}_1} &= \Pr\left[\mathcal{A}_1^{\mathcal{F}}(g, g^a, g^b) = \texttt{success} : a, b \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_p^*\right] \\ &= \Pr\left[\mathit{Verify}(PK, M^*, \sigma^*) = \texttt{valid} \; : \; \begin{array}{l}(PK, SK) \overset{\mathrm{R}}{\leftarrow} \mathit{KeyGen}, \\ (M^*, \sigma^*) \overset{\mathrm{R}}{\leftarrow} \mathcal{F}(PK)\end{array}\right] = \epsilon,\end{aligned}$$

where the first probability is taken over the coin tosses of $\mathcal{A}_1$ and $\mathcal{F}$, and over the choices of a and b. Since a is chosen uniformly from $\mathbb{Z}_p^*$, g^a, the public key $\mathcal{A}_1$ provides $\mathcal{F}$, is uniformly distributed in G.

Game 2. $\mathcal{A}_2$ functions as does $\mathcal{A}_1$, with a single exception. If $\mathcal{F}$ fails, $\mathcal{A}_2$ outputs "`failure`"; if $\mathcal{F}$ succeeds, outputting a forgery (M^*, σ^*), where i^* is the index of M^*, then $\mathcal{A}_2$ outputs "`success`" if $s_{i^*} = 1$, but "`failure`" if $s_{i^*} = 0$.

Clearly, $\mathcal{F}$ can get no information about any s_i, so its behavior cannot depend on their values. Thus the final trip test $\mathcal{A}_2$ performs is independent of the game to that point. Thus we have

$$\mathsf{Adv}_{\mathcal{A}_2} = \mathsf{Adv}_{\mathcal{A}_1} \cdot \Pr\left[s_{i^*} = 1\right] = \zeta\epsilon,$$

since each s_i is drawn from B_ζ.

Game 3. $\mathcal{A}_3$ functions as does $\mathcal{A}_2$, but, again, with a modification. If $\mathcal{F}$ fails to create a forgery, $\mathcal{A}_3$ also fails. If $\mathcal{F}$ succeeds in finding a forgery on M_{i^*}, $\mathcal{A}$ claims success only if $s_{i^*} = 1$, and $\mathcal{F}$ asked for signatures only on messages M_i for which $s_i = 0$.

Again, $\mathcal{F}$ can get no information about any s_i. Each of its signature requests can cause $\mathcal{A}$ to declare failure at the game's end, with probability ζ, but it cannot know, during the game, whether any of them did. The s_i's are independent, so each of $\mathcal{F}$'s signature requests is an independent trial insofar as disqualification by s_i is concerned. Moreover, s_{i^*} is independent of any s_i's for which $\mathcal{F}$ requests signatures, so the test that s_{i^*} equals 1 is again an independent trial, and the analysis of Game 2 is not affected.

The probability of $\mathcal{F}$'s not being disqualified because of any particular signature request is $1 - \zeta$. If $\mathcal{F}$ makes k signature oracle queries, where k necessarily is at most q_S, and if, moreover, it makes those queries on the messages with indices $i_1, \ldots, i_k$, then

$$\mathsf{Adv}_{\mathcal{A}_3} = \mathsf{Adv}_{\mathcal{A}_2} \cdot \Pr\left[s_{i_j} = 0,\; j = 1, \ldots, k\right] = \zeta\epsilon \cdot (1 - \zeta)^k \geq (1 - \zeta)^{q_S}\zeta\epsilon.$$

Game 4. $\mathcal{A}_4$ functions as does $\mathcal{A}_3$, except that, if $\mathcal{F}$ requests a signature on a message M_i for which $s_i = 1$, $\mathcal{A}$ declares failure and halts immediately.

We may fully describe a run of $\mathcal{A}$ by fixing the challenge, $\mathcal{A}$'s random bits, and $\mathcal{F}$'s random bits; these collectively determine the value of each s_i, and the indices on which $\mathcal{F}$ requests signatures. Let us call unlucky any runs in which $\mathcal{F}$ requests a signature on some M_i for which $s_i = 1$. $\mathcal{A}_3$ would already declare failure on any unlucky runs: if $\mathcal{F}$ declares failure, $\mathcal{A}_3$ does also; if $\mathcal{F}$ finds a forgery, $\mathcal{A}_3$ fails anyway because of the unlucky signature query. Thus $\mathcal{A}_3$ and $\mathcal{A}_4$ will agree (with output "`failure`") on all unlucky runs; they will also agree on all lucky runs, since the modification of $\mathcal{A}_4$ relative to $\mathcal{A}_3$ is not invoked on those runs. Thus we have

$$\mathsf{Adv}_{\mathcal{A}_4} = \mathsf{Adv}_{\mathcal{A}_3} \geq (1-\zeta)^{q_S}\zeta\epsilon.$$

The immediate halt in unlucky runs is a shortcut and does not affect the outcome distribution.

Game 5. $\mathcal{A}_5$ is based on $\mathcal{A}_4$. In the setup phase, for each i, if $s_i = 1$, $\mathcal{A}_5$ sets $h_i \leftarrow g^b \cdot g^{r_i}$ and $\sigma_i \leftarrow \star$, a placeholder value; if $s_i = 0$, it sets $h_i \leftarrow g^{r_i}$ and $\sigma_i \leftarrow (g^b)^{r_i}$, as before.

G is a cyclic group of prime order, so multiplication by any element of G, and g^b in particular, induces a permutation on G. Thus if r is uniformly distributed in $\mathbb{Z}_p^*$, g^r and $g^b \cdot g^r$ have identical, uniform distributions in G. $\mathcal{F}$ cannot learn any information about the s_i's from examining the h_i's it is given. $\mathcal{A}_5$ is unable to provide signatures on messages for which $s_i = 1$, but that is unimportant, since any runs in which $\mathcal{F}$ asks for such a signature are failed immediately. Therefore, $\mathcal{F}$ will behave under $\mathcal{A}_5$ exactly as it does under $\mathcal{A}_4$, and

$$\mathsf{Adv}_{\mathcal{A}_5} = \mathsf{Adv}_{\mathcal{A}_4} \geq (1-\zeta)^{q_S}\zeta\epsilon.$$

Game 6. $\mathcal{A}_6$ behaves as does $\mathcal{A}_5$. In those games where $\mathcal{A}_5$ outputs "`success`", however, $\mathcal{A}_6$ outputs "`success`" and, in addition, outputs $\sigma^*/(g^a)^{r_{i^*}}$, where i^* is the index of the message M^* for which $\mathcal{F}$ output a forged signature σ^*. ($\mathcal{A}_6$, like the $\mathcal{A}$'s before it, only succeeds when $\mathcal{F}$ succeeds.)

Clearly, $\mathcal{A}_6$ succeeds with precisely the same probability as $\mathcal{A}_5$, so

$$\mathsf{Adv}_{\mathcal{A}_6} = \mathsf{Adv}_{\mathcal{A}_5} \geq (1-\zeta)^{q_S}\zeta\epsilon.$$

Moreover, $\mathcal{A}_6$ only succeeds if $s_{i^*} = 1$, which means that $h_{i^*} = g^b \cdot g^{r_{i^*}}$. If σ^* is a valid signature on $M^* = M_{i^*}$, then $(g, g^a, h_{i^*}, \sigma^*)$ must be a valid Diffie-Hellman tuple, so σ^* must equal $h_{i^*}^a = g^{ab} \cdot (g^{r_{i^*}})^a$. Thus, in every instance on which $\mathcal{A}_6$ claims to succeed, it also outputs $\sigma^*/(g^a)^{r_{i^*}} = g^{ab}$, which is indeed the answer to the Diffie-Hellman challenge posed to it.

4.3 Optimization and Conclusion

The algorithm $\mathcal{A}_6$ thus uses the GDH-signature forger $\mathcal{F}$ to solve CDH challenges. What remains is to optimize the parameter ζ to achieve a maximal probability of success. The function $(1-\zeta)^{q_S}\zeta\epsilon$ is maximized at $\zeta = 1/(q_S+1)$, where it has the value

$$\frac{1}{q_S+1} \cdot \left(1 - \frac{1}{q_S+1}\right)^{q_S} \cdot \epsilon = \frac{1}{q_S} \cdot \left(1 - \frac{1}{q_S+1}\right)^{q_S+1} \cdot \epsilon.$$

(The latter equality follows from taking partial fractions.) Now $\mathcal{A}$'s success probability ϵ' is at least as great as this. For large q_S, $(1 - 1/(q_S+1))^{q_S+1} \approx 1/e$.

$\mathcal{A}$'s running time includes the running time of $\mathcal{F}$. The additional overhead imposed by $\mathcal{A}$ is dominated by the need to evaluate group exponentiation for each signature and hash request from $\mathcal{F}$. Any one such exponentiation may be computed by using at most $2 \lg p$ group actions, and thus at most $2 \lg p$ time units, on G (see [16]). $\mathcal{A}$ may need to answer as many as $q_H + q_S$ such requests, so its overall running time is $t' \leq t + 2c_{\mathcal{A}}(\lg p)(q_H + q_S)$, Where $c_{\mathcal{A}}$ is a small constant that accounts for the remainder of $\mathcal{A}$'s administrative overhead; in practice, $c_{\mathcal{A}}$ should be at most 2.

To summarize: if there exists a forger algorithm $\mathcal{F}$ that (t, q_H, g_S, ϵ)-breaks the GDH signature scheme on G, then there exists an algorithm $\mathcal{A}$ that (t', ϵ')-breaks CDH on G, where

$$t' = t + 2c_{\mathcal{A}}(\lg p)(q_H + q_S) \quad \text{and} \quad \epsilon' = \frac{1}{q_S} \cdot \left(1 - \frac{1}{q_S+1}\right)^{q_S+1} \cdot \epsilon.$$

Conversely, if G is a (τ, t', ϵ')-GDH group, then there can exist no algorithm $\mathcal{F}$ that (t, q_H, q_S, ϵ)-breaks the GDH signature scheme, where

$$t = t' - 2c_{\mathcal{A}}(\lg p)(q_H + q_S) \quad \text{and} \quad \epsilon = q_S\epsilon' \Big/ \left(1 - \frac{1}{q_S+1}\right)^{q_S+1}.$$

For all positive q_S, the radicand in the latter equation is greater than $1/2e$, so the equation may be rewritten as $\epsilon \leq q_S\epsilon' \cdot 2e$. This completes the proof.

5 Experimental results

5.1 Implementation Details

We experimented with the scheme of Section 3.4. Recall that signing is a single multiplication on the curve $y^2 = x^3 + 2x \pm 1$ over $\mathbb{F}_{3^l}$. Verifying a signature requires two Weil pairing computations over $\mathbb{F}_{3^{6l}}$. Hence, verifying takes more time than signing.

For efficiency, rather than working in $\mathbb{F}_{3^{6l}}$ directly (which involves manipulating polynomials of degree $6l$), we work with extensions of $\mathbb{F}_{3^6}$ of degree l. To speed up arithmetic in $\mathbb{F}_{3^6}$ we construct lookup tables (of size 3^6) for quickly multiplying two elements. Elements of $\mathbb{F}_{3^6}$ are represented by their exponent relative to a chosen generator, so that multiplication and division corresponds to addition modulo $3^6 - 1$. Addition is done using a multiplication, division and table lookup via the identity $a + b = a(1 + a^{-1}b)$. The constants r^+, r^-, u used in the automorphism ϕ also lie in $\mathbb{F}_{3^6}$ and can be quickly found by a brute force search.

We map an element a in $\mathbb{F}_{3^l}$ to an element of $\mathbb{F}_{3^{6l}}$ using the obvious injection: a is represented by a polynomial of degree l with coefficients in $\mathbb{F}_3$, and we simply view it as a polynomial with coefficients in $\mathbb{F}_{3^{6l}}$.

We use the Tate pairing [7] instead of the Weil pairing, since it has similar properties and is easier to compute: the Weil pairing requires two iterations of Miller's algorithm [17] and one division while the Tate pairing needs only one call to Miller's algorithm and an additional exponentiation.

Because Miller's algorithm involves the computation of various quotients, several divisions can be avoided since we may scale the numerator and denominator by arbitrary constants. We used sliding windows for every exponentiation-like operation, that is, exponentiation in $\mathbb{F}_{3^{6l}}$, Miller's algorithm, and multiplication of a point on the curve. Point multiplication can be sped up further by using signed sliding windows, converting to weighted projective coordinates (though this may not help; it depends on the implementation of the field operations), and taking advantage of the fact that some points are fixed for the whole system.

Recall that the output of the Tate pairing is a coset representative in $\mathbb{F}^*_{3^{6l}}$. Signature verification then consists of checking that the output of two Tate pairings lie in the same coset. This could be done by finding the quotient of the outputs, and raising it to the appropriate power (and comparing with the identity element). However, we can replace the division with a multiplication by exploiting the bilinearity of the Tate pairing: dividing by $e(A, B)$ is equivalent to multiplying by $e(A, -B) = 1/e(A, B)$ ($-B$ can be easily computed from B by negating the y-coordinate).

The x-coordinate is an element of $\mathbb{F}_{3^l}$ and is represented as a polynomial of degree at most $l-1$ with coefficients in $\mathbb{F}_3$. For output, it is viewed as a number in base 3, and then encoded in base-64. For $l = 97$, which has 923-bit discrete-log security, an example signature looks as follows: "`KrpIcV0O9CJ8iyBS8MyVkNrMyE`". This is under half the size of the standard 320-bit DSS signature (with 1024-bit discrete security).

5.2 Running Times

The following table shows the time required to verify a signature. Recall that a verification is much more expensive than signature generation because it requires computing two pairings. The program was run on a 1 GHz Pentium III computer running GNU/Linux.

l	sig-length (bits)	Dlog Security $\lceil \log_2 x \rceil$	curve	Running Time (seconds)
79	126	752	E^-	1.6
97	154	923	E^+	2.9
149	237	1417	E^+	9.6
163	259	1551	E^+	13.3
163	259	1551	E^-	13.4
167	265	1589	E^+	14.0

When using elliptic curves to get short GDH signatures we are forced to use a curve over a field of characteristic three. This slows down arithmetic on the curve. It is possible that the running times above can be improved using higher genus curves over fields of characteristic two as discussed at the end of Section 3.5. Similarly, the techniques of [13] for computing on the curves E^+ and E^- over $\mathbb{F}_{3^l}$ may slightly improve these numbers.

6 Conclusions

We presented a short signature based on the Weil pairing. The length of a signature is one element of a finite field. Standard signatures based on discrete log such as DSA require two elements. When working with the curve $y^2 = x^3 + 2x \pm 1$ over $\mathbb{F}_{3^l}$ the MOV attack maps the CDH problem in this curve to a CDH problem in $\mathbb{F}_{3^{6l}}$. Hence, we can use small values of l to obtain short signatures with security comparable to the security of 320-bit DSA. For example, we obtain a signature of length 154 bits where breaking the scheme reduces to solving the Diffie-Hellman problem in a finite field of size approximately 2^{923}. In Section 3.5 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, and Moni Naor for helpful discussions about this work.

References

1. ANSI X9.62 and FIPS 186-2. Elliptic Curve Digital Signature Algorithm, 1998.
2. R. Balasubramanian and N. Koblitz. The Improbability That an Elliptic Curve Has Subexponential Discrete Log Problem under the Menezes-Okamoto-Vanstone Algorithm. *Journal of Cryptology*, 11(2):141–145, 1998.
3. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures: How to Sign with RSA and Rabin. In U. Maurer, editor, *Proceedings of Eurocrypt '96*, volume 1070 of *LNCS*, pages 399–416. Springer-Verlag, 1996.
4. D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In J. Kilian, editor, *Proceedings of Crypto '2001*, volume 2139 of *LNCS*, pages 213–229. Springer-Verlag, 2001.
5. D. Chaum and T. Pederson. Wallet Databases with Observers. In E. Brickell, editor, *Proceedings of Crypto '92*, volume 740 of *LNCS*, pages 89–105. Springer-Verlag, 1992.
6. J.-S. Coron. On the Exact Security of Full Domain Hash. In M. Bellare, editor, *Proceedings of Crypto '2000*, volume 1880 of *LNCS*, pages 229–235. Springer-Verlag, 2000.

7. G. Frey, M. Muller, and H. Ruck. The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems. *IEEE Tran. on Info. Th.*, 45(5):1717–1719, 1999.
8. S. Galbraith. Supersingular curves in cryptography. In *Proceedings of Asiacrypt '2001*, LNCS. Springer-Verlag, 2001.
9. 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.
10. 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.
11. A. Joux. A One Round Protocol for Tripartite Diffie-Hellman. In W. Bosma, editor, *Proceedings of ANTS IV*, volume 1838 of *LNCS*, pages 385–394. Springer-Verlag, 2000.
12. 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/.
13. N. Koblitz. An Elliptic Curve Implementation of the Finite Field Digital Signature Algorithm. In H. Krawczyk, editor, *Proceedings of Crypto '98*, volume 1462 of *LNCS*, pages 327–333. Springer-Verlag, 1998.
14. S. Lang. *Elliptic Functions.* Addison-Wesley, Reading, MA, 1973.
15. A. Menezes, T. Okamoto, and P. Vanstone. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.
16. A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.
17. V. Miller. Short Programs for Functions on Curves. unpublished manuscript, 1986.
18. I. Mironov. A Short Signature as Secure as DSA. Preprint, 2001.
19. D. Naccache and J. Stern. Signing on a Postcard. In *Proceedings of Financial Cryptography '00*, 2000.
20. T. Okamoto and D. Pointcheval. The Gap Problems: A New Class of Problems for the Security of Cryptographic Primitives. In K. Kim, editor, *Public Key Cryptography, PKC 2001*, volume 1992 of *LNCS*, pages 104–118. Springer-Verlag, 2001.
21. L. Pintsov and S. Vanstone. Postal Revenue Collection in the Digital Age. In *Proceedings of Financial Cryptography '00*, 2000.
22. J. H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, 1986.
23. W. C. Waterhouse. Abelian Varieties over Finite Fields. *Ann. Sci. École Norm. Sup.*, 2:521–60, 1969.