Fast hashing onto elliptic curves over fields of characteristic 3

Paulo S. L. M. Barreto[1] and Hae Y. Kim[2]

[1] Scopus Tecnologia S. A.
Av. Mutinga, 4105
BR 05110–000 São Paulo (SP), Brazil.
pbarreto@scopus.com.br
[2] Universidade de São Paulo, Escola Politécnica.
Av. Prof. Luciano Gualberto, tr. 3, 158
BR 05508-900, São Paulo(SP), Brazil.
hae@lps.usp.br

Abstract. We describe a fast hash algorithm that maps arbitrary messages onto points of an elliptic curve defined over a finite field of characteristic 3. Our new scheme runs in time $O(m^2)$ for curves over $\mathbb{F}_{3^m}$. The best previous algorithm for this task runs in time $O(m^3)$. Experimental data confirms the speedup by a factor $O(m)$, or approximately a hundred times for practical m values. Our results apply for both standard and normal basis representations of $\mathbb{F}_{3^m}$.

1 Introduction

The recent discovery of groups where the Decision Diffie-Hellman (DDH) problem is easy while the Computational Diffie-Hellman (CDH) problem is hard [13] has given rise to a digital signing scheme by Boneh et al. [3], here dubbed simply BLS, where the signature size is essentially the same as the underlying finite field. Contrary to all previously known schemes the new algorithm uses intrinsic properties of elliptic curves and have no equivalent in more conventional groups like $\mathbb{Z}_p^*$. Although other kinds of signature algorithms are known that produce smaller signatures [21, 6], the BLS scheme is much faster in practice, and to the best of our knowledge not covered by patents.

Currently, the BLS algorithm has only been instantiated for supersingular elliptic curves over $\mathbb{F}_{3^m}$, and it depends on the existence of a hash function that maps arbitrary messages directly onto curve points in such a way that the discrete logarithm of the hash value remains unknown. The original description of the BLS algorithm suggests for this hash function an iterated probabilistic construction whose running time is cubic in the extension degree of the underlying finite field, i.e. $O(m^3)$, which derives from the need to compute the square roots in the finite field.

We propose a different, but conceptually simple approach that entirely avoids the square root evaluations and takes $O(m^2)$ steps. The computational complexity derives from the squaring of a field element and from solving a system of linear equations over $\mathbb{F}_{3^m}$ with coefficients in $\mathbb{F}_3$.

Compared to the original BLS hash, our scheme needs a slightly higher number of coin tosses to produce a hash value, namely, about 3 instead of 2 hash oracle queries, but this is completely offset by the increased efficiency of each query. The signature size is also increased by 2 bits (or rather by an element from $\mathbb{F}_3$).

In a sense, our results shed some light on the practicality of working with the seldom used $\mathbb{F}_{3^m}$ fields, where operations can be surprisingly efficient if properly implemented.

This document is organized as follows. Section 2 discusses the Decision Diffie-Hellman problem. Section 3 describes BLS signatures, and section 4 the associated hashing algorithm. We present our improved hash scheme in section 5, analyze its security properties in section 6, and show how to use it in a signature scheme in section 7. In section 8 we provide experimental results, and we conclude in section 9.

2 The Decision Diffie-Hellman problem

Consider the tuple (P, aP, bP, cP) where P is a point of order q of an elliptic curve $E/\mathbb{F}_{p^m}$. The Decision Diffie-Hellman (DDH) problem on the subgroup $\langle P \rangle$ consists of deciding whether $ab \equiv c \pmod{q}$.

We begin by defining the security multiplier of an elliptic curve group (or a subgroup thereof).

Definition 1. *Let p be a prime, m a positive exponent, and E an elliptic curve over $\mathbb{F}_{p^m}$ with n points. Let $P \in E$ be a point of prime order q where $q^2 \not| m$. We say that the subgroup $\langle P \rangle$ has a* security multiplier *α for some integer $\alpha > 0$ if $q \mid p^{m\alpha} - 1$ and $q \not| p^{mk} - 1$ for all $k - 1, 2, \ldots, \alpha - 1$ (this means that the order of p^m in $\mathbb{F}_q^*$ is α).*

Given $P \in E$ of order q, let Q be another curve point of the same order but linearly independent from P, and let $E[q]$ be the subgroup of $E/\mathbb{F}_{p^{m\alpha}}$ generated by P and Q, where α is the security multiplier of $\langle P \rangle$. The *Weil pairing* [13, 18, 20, 24] is a mapping $e : E[q] \times E[q] \leftarrow \mathbb{F}_{p^{m\alpha}}^*$ satisfying the following properties:

1. Bilinearity: for all $P, Q \in E[q]$ and $a, b \in \mathbb{Z}$, $e(aP, bQ) = e(P, Q)^{ab}$.
2. Alternation: for all $P, Q \in E[q]$, $e(P, Q) = e(Q, P)^{-1}$, so in particular $e(P, P) = 1$.
3. Non-degenerate: if $e(P, Q) = 1$ for all $Q \in E[q]$, then $P = O$.

An efficient, very clearly presented algorithm to compute the Weil pairing on supersingular elliptic curves is given in [4, appendices B and C], based on work by Miller [20].

The Weil pairing allows one to determine whether the tuple (P, aP, Q, bQ) is such that $a \equiv b \pmod{q}$, since this is equivalent to $e(P, bQ) = e(aP, Q)$. Suppose there is a computable isomorphism $\phi : \langle P \rangle \to \langle Q \rangle$; then the Weil pairing solves the DDH defined by the tuple (P, aP, bP, cP) by means of the relation:

$$ab \equiv c \pmod{q} \Leftrightarrow e(P, \phi(cP)) = e(aP, \phi(bP)).$$

3 The BLS signature algorithm

Cryptosystems based on group arithmetic are all susceptible, at least to some extent, to generic group attacks, of which the most powerful known is Pollard's rho attack [22]. Two very powerful dedicated attacks are known against elliptic curve cryptosystems: the Weil descent attack and the Menezes-Okamoto-Vanstone (MOV for short) attack.

Weil descent attacks [8–10] map the elliptic curve group to a subgroup of a hyperelliptic curve of higher genus, where the discrete logarithm problem can be solved by subexponential algorithms. This attack is not applicable if the extension degree of the finite field over which the curve is defined is prime.

The MOV attack [18] maps the elliptic curve discrete logarithm problem to an analogous problem in the multiplicative group of a finite field (an extension field of the underlying field over which the curve is defined). Resistance against the MOV attack can be quantified by the curve's security multiplier α: an elliptic curve cryptosystem resists the MOV attack if the discrete logarithm in the extension field $\mathbb{F}_{p^{m\alpha}}$ is infeasible. In practice, the MOV attack is only effective against certain supersingular curves whose security multiplier is very small.

The BLS scheme uses supersingular elliptic curves over $\mathbb{F}_{3^m}$ defined by $E^\pm : y^2 = x^3 - x \pm 1$. Only finite fields of prime extension degree m are used to avoid Weil descent attacks. Furthermore, only curves with security multiplier $\alpha = 6$ (the maximum achievable value for supersingular curves) are allowed to prevent the MOV attack. Values of m of practical interest are 79, 97, 149, 163, and 169.

3.1 Key generation

Given one of the values m above, let $E/\mathbb{F}_{3^m}$ be the corresponding curve and let q be the largest prime factor of the order of the curve. Let $P \in E$ be a point of order q. The private signing key is a statistically unique, uniformly chosen element $s \in \mathbb{F}_q^*$, and the corresponding public key is the tuple (m, P, V) where V is the curve point $V = sP$ (the public value q is implicit in the public key because the value of m and the curve equation uniquely determine q).

3.2 Signing

To sign a message $M \in \{0,1\}^*$, map M to a point $P_M \in \langle P \rangle$. Set $S_M \leftarrow sP_M$. The signature σ is the abscissa of S_M. Notice that $\sigma \in \mathbb{F}_{3^m}$.

3.3 Verification

The BLS scheme verifies signatures by solving the DDH problem with the Weil pairing.

Let u, $r^\pm$ be elements of $\mathbb{F}_{3^{m\alpha}}$ satisfying $u^2 + 1 = 0$, $(r^\pm)^3 - r^\pm \pm 2 = 0$, and let $\phi^\pm : E^\pm \to E^\pm$ such that $\phi^\pm(x, y) \equiv (-x + r^\pm, uy)$ for any point $P = (x, y) \in E^\pm$ of order q. Then $Q = \phi^\pm(P)$ is a point of the same order q linearly independent from P [24, p. 326].

Given a public key (m, P, V), a message M, and a signature σ do:

1. Find a point $S \in E/\mathbb{F}_{3^m}$ of order q whose abscissa is σ and whose ordinate is y for some $y \in \mathbb{F}_{3^m}$. If no such point exists, reject the signature.
2. Set $u \leftarrow e(P, \phi(S))$ and $v \leftarrow e(V, \phi(h(M)))$, where e is the Weil pairing on the curve $E/\mathbb{F}_{3^{6m}}$.
3. Accept the signature if, and only if, either $u = v$ or $u^{-1} = v$.

Note that both (σ, y) and $(\sigma, -y)$ are points on $E/\mathbb{F}_{3^m}$ with abscissa σ. 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 $(P, V, h(M), S)$ is a Diffie-Hellman tuple, while $u^{-1} = v$ tests that $(P, V, h(M), -S)$ is a Diffie-Hellman tuple. The fact that the signature consists exclusively of the abscissa of S causes a slight degradation in the ability of detecting forgery, since S and $-S$ are indistinguishable to the verifying algorithm.

4 Hashing onto curves

To complete the above specification one needs a hash function to map messages onto $\langle P \rangle$. In what follows we assume for simplicity that $\langle P \rangle$ spans the whole elliptic curve group; the case of a proper subgroup is easy to derive, and is explained in detail in [3, section 3.3].

One can view the field $\mathbb{F}_{3^m}$ as a vector space of dimension m over $\mathbb{F}_3$, in which case an element $u \in \mathbb{F}_{3^m}$ is represented in a basis by a tuple $(u_0, u_1, \ldots, u_{m-1})$, $u_i \in \mathbb{F}_3$ for $i = 0, 1, \ldots, m-1$. Besides standard polynomial bases, we also consider normal bases defined as follows.

Definition 2. *A* normal basis *is a linearly independent set* $\{\theta^{3^i} \mid 0 \leqslant i < m\}$ *where* θ *is a root in* $\mathbb{F}_{3^m}$ *of an irreducible polynomial of degree* m.

For convenience, define $\tau(u) \equiv u_0$, i.e. the independent term in standard polynomial basis, or the coefficient of θ in normal basis. We also define $\tilde{u} \equiv (u_1, \ldots, u_{m-1})$.

The original *Map2Group* function maps a message M to a curve point (x, y) using a more conventional hash function $h : \mathbb{Z} \times \{0,1\}^*$ as follows. Let the elliptic curve equation be $y^2 = x^3 - x + b$ over $\mathbb{F}_{3^m}$, where $b = \pm 1$. Fix a small parameter $I = \lceil \lg \lg(1/\delta) \rceil$, where δ is some desired bound on the probability of failure. Then compute function *Map2Group* as follows:

Map2Group$_h$(M)

1. Set $i = 0$.
2. Hash the pair (i, M) to a pair $h(i, M) = (x, t)$ where $x \in \mathbb{F}_{3^m}$ and $t \in \{0, 1\}$.
3. Compute $u = x^3 - x + b$.
4. Solve the quadratic equation $y^2 = u$ in $\mathbb{F}_{3^m}$.
5. If no solution is found, increment i and try again from step 2.
6. Otherwise, use t to choose between the solutions y_0 and y_1, and return (x, y_t).

Choosing between the roots in the last step above merely consists of ensuring that $\tau(y_0) < \tau(y_1)$.

If an upper bound is imposed on the counter i, then it is possible that an unhashable message exists. The failure probability can be made arbitrarily small by picking an appropriately large I. Furthermore, it is necessary to compute a square root in a finite field of characteristic 3 to solve the quadratic equation. Actually, the quadratic equation solving at step 4 must be also executed during verification (in step 1).

It is not really necessary to hash (i, M) onto a pair (x, t); mapping to x alone would work as fine. Indeed, t is only needed to distinguish between two curve points $P_M = (x, y)$ and $-P_M = (x, -y)$, which lead to the same signature since both $S_M = sP_M$ and $-S_M = s(-P_M)$ share the same abscissa. Therefore, the actual value of t is irrelevant, and a fixed t would be equally suitable.

5 The new approach

Before we proceed it is convenient to provide some definitions.

Let $\mathcal{C} : \mathbb{F}_{3^m} \to \mathbb{F}_{3^m}$ be defined by $\mathcal{C}(x) = x^3 - x$. The kernel of $\mathcal{C}$ is $\mathbb{F}_3$ [16, chapter 2, section 1], hence the rank of $\mathcal{C}$ is $m - 1$ [11, section 3.1, theorem 2].

Definition 3. *The (absolute)* trace *of an element $a \in \mathbb{F}_{3^m}$ is given by*

$$\mathrm{Tr}(a) = a + a^3 + a^9 + \cdots + a^{3^{m-1}}.$$

The trace will always be in $\mathbb{F}_3$ as one can easily check by noticing from the above definition that $\mathcal{C} \circ \mathrm{Tr} \equiv 0$, i.e. $\mathrm{Tr}(a)^3 = \mathrm{Tr}(a)$, for all $a \in \mathbb{F}_{3^m}$. The trace is also surjective and linear over $\mathbb{F}_3$, so it can always be represented as a matrix in a basis.

Obtaining a full curve point by first specifying the abscissa and then computing a suitable ordinate is commonplace in elliptic curve cryptography. Such a technique is used in all algorithms adopted for existing standards [1, 12].

In fields of characteristic 3, cubing is a *linear* operation. Therefore, it is more advantageous to hash the message M to an *ordinate* instead of an abscissa. This property is exploited by function *Map3Group* below. Fix a small parameter $J = \lceil \lg\lg(1/\delta) - \lg(\lg(3) - 1) \rceil \approx \lceil \lg\lg(1/\delta) \rceil + 1 = I + 1$, where δ is the desired bound on the probability of failure. Then compute function *Map2Group* as follows:

Map3Group$_h$(M)

1. Set $i = 0$.
2. Hash the pair (i, M) to a pair $h(i, M) = (y, t) \in \mathbb{F}_{3^m} \times \mathbb{F}_3$.
3. Compute $u = y^2 - b$.
4. Solve the cubic equation $\mathcal{C}(x) = u$.
5. If no solution is found, increment i and try again from step 2.
6. Otherwise, use t to choose between the solutions y_0 to y_2, and return (x, y_t).

As before, choosing between the roots in the last step above merely consists of ensuring that $\tau(y_0) < \tau(y_1) < \tau(y_2)$.

The equation in step 4 has a solution if, and only if, $\mathrm{Tr}(u) = 0$ [16, theorem 2.25]. This is the case for 1/3 of the elements in $\mathbb{F}_{3^m}$, since the trace function is linear and surjective.

Here too the failure probability can be made arbitrarily small by picking an appropriately large J, which is only one unit larger than the corresponding value of I in the original *Map2Group*. For each i, the probability (over the choice of the random oracle h') that $h'(i||M)$ leads to a point on G^* is that of finding a solution to the cubic equation in Step 4, or 1/3 for uniformly distributed u values. Hence, the expected number of calls to h' is approximately 3, and the probability that a given message M will be found unhashable is $(2/3)^{2^J} \leqslant \delta$.

The complexity derives from the squaring in step 3 and the cubic equation solving in step 4. Squaring is obviously no more complex than $O(m^2)$ (this may involve the use of multiplication matrices for a normal basis [17, section 6.1]). We now show how to efficiently solve the cubic equation on $\mathbb{F}_{3^m}$.

5.1 Solving the cubic equation in standard polynomial basis

Trace computation: Since the trace is actually a linear form $\mathrm{Tr} : \mathbb{F}_{3^m} \to \mathbb{F}_3$, precompute its representation T (an usually sparse m-tuple of $\mathbb{F}_3$ elements) in the given basis, and thereafter obtain $\mathrm{Tr}(u)$ as the inner product Tu in $O(m)$ time.

Solving $\mathcal{C}(x) = u$: The cubic equation reduces to a system of linear equations with coefficients in $\mathbb{F}_3$, and can be solved in no more than $O(m^2)$ steps. This is achieved by first checking whether the system has solutions, i.e. whether $\mathrm{Tr}(u) = 0$. If so, since the rank of $\mathcal{C}$ is $m-1$ one obtains an invertible $(m-1) \times (m-1)$ matrix A by leaving out the one row and correspondingly one column of the matrix representation of $\mathcal{C}$ on the given basis. A solution of the cubic equation is then given by an arbitrary element $x_0 \in \mathbb{F}_3$ and by the solution of system $A\tilde{x} = \tilde{u}$, which is obtained as $\tilde{x} = A^{-1}\tilde{u}$ in $O(m^2)$ time.

5.2 Solving the cubic equation in normal basis:

Trace computation: The trace can be computed very easily in a normal basis. From the definition of trace, one sees that computing $\mathrm{Tr}(u)$ amounts to summing up all coefficients of u in the normal basis and multiplying the result by $\mathrm{Tr}(\theta)$. Obviously, it is most advantageous to choose θ so that $\mathrm{Tr}(\theta) = 1$.

Solving $\mathcal{C}(x) = u$: Using a normal basis to represent field elements, it is not difficult to see that the cubic equation can be efficiently solved in $O(m)$ time by the following algorithm (the proof is straightforward and left as an exercise):

$x_0 \leftarrow$ root selector (an arbitrary element from $\mathbb{F}_3$)
for $i \leftarrow 1$ to $m-1$ do {
    $x_i \leftarrow x_{i-1} - u_i$
}
x is a solution if, and only if, $x_{m-1} = x_0 + u_0$.

A minor drawback of hashing onto an ordinate instead of an abscissa is that the convergence is slower, since the probability of finding a solution to the cubic equation in step 4 is only $1/3$ for uniformly distributed hash values in step 2. This means that the expected number of hash queries in step 2 is 3 instead of 2.

6 Proof of security

We now show that our hash proposal is secure in the random oracle model [2] against existential forgery under chosen-message attacks. Both the theorem below and its proof closely follow [3, Lemma 4].

For simplicity, here again we only discuss hashing onto the full elliptic curve group. Modifying the argument for hashing onto a proper subgroup thereof is fairly simple.

Definition 4. *A forger algorithm $\mathcal{F}$ (t, q_H, q_S, ϵ)-*breaks *a signature scheme if $\mathcal{F}$ runs in time at most t, makes at most q_H adaptive queries to a hash oracle and at most q_S adaptive queries to a signing oracle, and produces with probability not smaller than ϵ a message M and a valid signature σ for M under a given, randomly generated key pair (s, V).*

Definition 5. *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.*

Theorem 1. *Suppose the BLS signature scheme is (t, q_H, q_S, ϵ)-secure in the group G when using a random hash function $h : \{0,1\}^* \to G^*$. Then it is $(t - 2^J q_H \lg n, q_H, q_S, \epsilon)$-secure when the hash function h is computed with Map3Group$_{h'}$ where h' is a random hash function $h' : \{0,1\}^* \to \mathbb{F}_{3^m} \times \mathbb{F}_3$.*

Proof. Suppose a forger algorithm $\mathcal{F}'$ (t, q_H, q_S, ϵ)-breaks the BLS algorithm on the bgroup G when the hash function h is computed using *Map3Group*$_{h'}$. We construct an algorithm $\mathcal{F}$ that $(t + 2^J q_H, q_H, q_S, \epsilon)$-breaks the scheme when h is a random oracle $h : \{0,1\}^* \to G^*$.

The forger $\mathcal{F}$ runs $\mathcal{F}'$ 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 *Map3Group*$_{h'}$. It uses an array s_{ij} of elements of $\mathbb{F}_{3^m} \times \mathbb{F}_3$. The array has q_H rows and 2^J columns. On initialization, $\mathcal{F}$ fills s_{ij} with uniformly-selected elements of $\mathbb{F}_{3^m} \times \mathbb{F}_3$.

$\mathcal{F}$ then runs $\mathcal{F}'$, and keeps track of 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, M_i) whose M_i $\mathcal{F}$ had not previously seen (and whose w is an arbitrary J-bit string), $\mathcal{F}$ computes $(x_i, y_i) = h(M_i) \in G^*$ and scans the row s_{ij}, $0 \leqslant j < 2^J$. For each

$(x, b) = \mathsf{s}_{ij}$, $\mathcal{F}$ solves the cubic equation in step 4 of *Map3Group* above, seeking points in G^*. For the smallest j for which s_{ij} maps into G^*, F replaces s_{ij} with a different point (x_i, b_i) where $b \in \mathbb{F}_3$ is set so that (x_i, b_i) corresponds to (x_i, y_i) in step 6 of $\mathit{Map3Group}_{h'}$. This way $\mathit{Map3Group}_{h'}(M_i) = h(M_i)$ as required.

Once this preliminary patching has been completed, F is able to answer h' hash queries by $\mathcal{F}'$ for pairs (w', M_i) by simply returning $\mathsf{s}_{iw'}$. 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 $\mathit{Map3Group}_{h'}$, 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 bookkeeping information. $\square$

In the case of hashing onto a proper subgroup, one can show that the scheme is $(t - 2^J q_H \lg n, q_H, q_S, \epsilon)$-secure, where n is the subgroup order.

7 A modified signature scheme

As we pointed out in section 4, in the original BLS scheme it is necessary to solve a quadratic equation not only at signing time but during verification as well. To completely avoid this, we propose the following modified scheme (key generation is unchanged):

7.1 Signing

To sign a message $M \in \{0, 1\}^*$, map M to a point $P_M \in \langle P \rangle$ using *Map3Group*. Set $(x_\sigma, y_\sigma) \leftarrow sP_M$. The signature σ is the pair (t_σ, y_σ), where $t_\sigma \equiv \tau(x_\sigma)$.

7.2 Verification

Given a public key (m, P, V), a message M, and a signature (t_σ, y_σ) do:

1. Find a point $S = (x_S, y_S) \in E/\mathbb{F}_{3^m}$ of order q satisfying $y_S = y_\sigma$ and $\tau(x_S) = t_\sigma$. If no such point exists, reject the signature.
2. Set $u \leftarrow e(P, \phi(S))$ and $v \leftarrow e(V, \phi(h(M)))$, where e is the Weil pairing on the curve $E/\mathbb{F}_{3^{6m}}$.
3. Accept the signature if, and only if, $u = v$.

This scheme has the drawback that the signature is slightly larger (by an extra $\mathbb{F}_(3)$ element) than the original scheme. Unfortunately this seems avoidable only at the cost of an additional DDH step, which would deteriorate the verification speed. On the other hand, attaching the selector t_σ to the signature increases the ability to detect forgery by a factor of 2, as now only the exact point resulting from the signing process is accepted as valid.

8 Experimental results

We have implemented both the original and the modified forms of *Map3Group* for the curve $E^- : y^2 = x^3 - x - 1$ over $\mathbb{F}_{3^{79}}$ and $\mathbb{F}_{3^{163}}$, and the curve $E^+ : y^2 = x^3 - x + 1$ over $\mathbb{F}_{3^{97}}$. In particular, E^+ gives rise to signatures of about 160 bits in lengths and security roughly equivalent to 320-bit DSA or ECDSA [1] signatures, and has very practical interest.

The improved hash scheme runs noticeably faster (by about two orders of magnitude) than the original algorithm, as we see in table 8.

Table 1. Running times of *Map2Group* and *Map3Group* in ms.

m	*Map2Group*	*Map3Group*
79	3.02	0.037
97	5.53	0.054
163	25.7	0.164

8.1 Techniques for software implementation

The algorithms were coded in the C++ language and run on a 550 MHz Pentium III processor. Only modest attempts were made to optimize the coded algorithms, namely by applying some standard implementation techniques like inner loop unrolling and function inlining.

Contrary to the situation of $\mathbb{F}_{2^m}$ and $\mathbb{F}_p$, current processors in general lack native support for arithmetic in $\mathbb{F}_{3^m}$. Nevertheless, it is possible to exploit existing operations to maximize parallelism within computer words. Our reference implementation represents $\mathbb{F}_{3^m}$ in polynomial basis by packing eight coefficients per 32-bit word, each $\mathbb{F}_3$ coefficient occupying a nibble (4 bits). Although it is possible to store 10 or even 11 coefficients per word (in the latter case, if carry bits are taken care of separately), our choice is more natural and leads to simpler formulas. This representation is also suitable when computing inverses with the almost inverse algorithm [23].

The square root extraction algorithm used by the original BLS scheme is described in [5, section 1.5] and uses the property that, if $x = r^2$ for some $r \in \mathbb{F}_q$ where $q \equiv 3 \pmod 4$, then $r \in \{r_1, r_2\}$ where $r_1 = x^{(q+1)/4}$ and $r_2 = -r_1$. This property, which holds for $q = 3^m$ if m is odd, is exploited in a windowed exponentiation algorithm [19, algorithm 14.82].

9 Conclusions and extensions

We have proposed a variant of the hash scheme used in BLS digital signature algorithm that is substantially faster than the original approach. The main idea is to hash messages onto curve ordinates and then solving a cubic equation over $\mathbb{F}_{3^m}$ to compute the corresponding abscissa, which can be done in time $O(m^2)$. In comparison, the original algorithm hashes messages onto curve abscissas and then solves a quadratic equation to compute a valid ordinate, a process that takes time $O(m^3)$. Experimental results indicate that even simple cubic equation solving methods are very effective, improving the running time by a factor of one hundred or more. The linear system found in the process is fairly sparse, so dedicated solving methods may increase performance to some extent.

Under certain circumstances, it may be undesirable to transmit the full counter used by function *Map3Group* together with the signature. In theory, the counter might be omitted from the signature data altogether, but this would burden the verifier, which already has the heavier part of the work, with the same effort the signer has to spend. A straightforward tradeoff is to attach only a few counter bits to the signature, say the least significant ones, and recompute only the remaining bits upon verification. The speed achievable with our proposed hash scheme reduce the recalculation overhead and makes the penalty for the verification algorithm negligible.

Still, the elliptic curve scalar multiplications necessary to complete the signature are the bottleneck of the BLS algorithm. Some techniques enables practical speedups for these operations. Point halving [14] and the methods proposed by Koblitz [15] for curves over fields of characteristic 3 may be used for this purpose to great effect, though the computational effort incurred is still $O(m^3)$.

Although the overall speedup our technique provides is modest, to the best of our knowledge it is free of patents, which may be an important consideration under several circumstances.

10 Acknowledgements

We are grateful to Dan Boneh, Ben Lynn, Ricardo Komatsu de Almeida, and especially to Frederik Vercauteren for fruitful discussions regarding the contents of this paper.

References

1. ANSI X9.62, "Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature Algorithm (ECDSA)," 1999. Also published in FIPS 186-2.
2. M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," Proc. First Annual Conference on Computer and Communications Security, ACM, 1993.
3. D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," Proceedings of AsiaCrypt'2001, to appear. Preprint available online at `http://crypto.stanford.edu/ dabo/abstracts/weilsigs.html`.
4. D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," Advances in Cryptology, Crypto'2001, Lecture Notes in Computer Science 2139, pp. 213-229, Springer-Verlag, 2001.

5. H. Cohen, "A Course in Computational Algebraic Number Theory," Springer-Verlag, 1993.
6. N. Courtois, M. Finiasz and N. Sendrier, "How to achieve a McEliece-based digital signature scheme," Proceedings of Asiacrypt'2001, to appear. Available online at `http://eprint.iacr.org/2001/010/`.
7. G. Frey, M. Müller, and H. Rück, "The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems," IEEE Transactions on Information Theory, 45(5), pp. 1717–1719, 1999.
8. S.D. Galbraith, F. Hess, N.P. Smart, "Extending the GHS Weil Descent Attack," Cryptology ePrint Archive: Report 2001/054.
9. S.D. Galbraith, N.P. Smart, "A cryptographic application of Weil Descent," *Codes and Cryptography*, Lecture Notes in Computer Science 1746, pp. 191–200, Springer-Verlag, 1999.
10. P. Gaudry, F. Hess and N.P. Smart, "Constructive and destructive facets of Weil descent on elliptic curves," Journal of Cryptology, to appear.
11. K. Hoffman and R. Kunze, "Linear Algebra," 2nd Edition, Prentice-Hall Inc., 1971.
12. IEEE Std 1363-2000, "Standard Specifications for Public-Key Cryptography," 2000.
13. 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/`.
14. E.W. Knudsen, "Elliptic Scalar Multiplication Using Point Halving," Proceedings of AsiaCrypt'99, Lecture Notes in Computer Science 1716, pp. 135–149, Springer-Verlag, 1999.
15. N. Koblitz, "An Elliptic Curve Implementation of the Finite Field Digital Signature Algorithm," Advances in Cryptology – Crypto '98. Lecture Notes in Computer Science 1462, pp. 327–337, Springer-Verlag, 1998.
16. R. Lidl and H. Niederreiter, "Finite Fields," Encyclopedia of Mathematics and its Applications, vol. 20, 2nd Edition, Cambridge University Press, 1997.
17. A.J. Menezes, "Elliptic Curve Public Key Cryptosystems," Kluwer International Series in Engineering and Computer Science, 1993.
18. A. Menezes, T. Okamoto, and P. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field," IEEE Transactions on Information Theory 39(5), pp. 1639–1646, 1993.
19. A.J. Menezes, P.C. Van Oorschot, S.A. Vanstone, "Handbook of Applied Cryptography," CRC Press, 1997.
20. V. Miller, "Short Programs for Functions on Curves," unpublished manuscript, 1986.
21. J. Patarin, N. Courtois and L. Goubin, "Quartz, 128-bit long digital signatures," NESSIE submission, 2000. Available online at `http://www.cryptonessie.org/`.
22. J.M. Pollard, "Monte Carlo methods for index computation $(\bmod p)$," Mathematics of Computation 32(1978), pp. 918–924.
23. R. Schroeppel, H. Orman, S. O'Malley, O. Spatscheck, "Fast Key Exchange with Elliptic Curve Systems," Advances in Cryptology – Crypto '95. Lecture Notes in Computer Science 963, pp. 43–56, Springer-Verlag, 1995.
24. J.H. Silverman, "The Arithmetic of Elliptic Curves," Graduate Texts in Mathematics, vol. 106, Springer-Verlag, 1986.