J. Cryptology (1998) 11: 219–234

Journal of
CRYPTOLOGY

Fast Multiplication on Elliptic Curves over Small Fields of Characteristic Two*

Volker Müller
Fachbereich Informatik, Technische Universität Darmstadt,
Alexanderstrasse 10, 64283 Darmstadt, Germany
vmueller@cdc.informatik.tu-darmstadt.de

Communicated by Johannes Buchmann

Received 14 January 1997 and revised 4 September 1997

Abstract. We discuss new algorithms for multiplying points on elliptic curves defined over small finite fields of characteristic two. This algorithm is an extension of previous results by Koblitz, Meier, and Staffelbach. Experimental results show that the new methods can give a running time improvement of up to 50% compared with the ordinary binary algorithm for multiplication. Finally, we present a table of elliptic curves, which are well suited for elliptic curve public key cryptosystems, and for which the new algorithm can be used.

Key words. Elliptic curves, Characteristic two, Multiplication, Frobenius expansion, Frobenius endomorphism.

1. Introduction

Elliptic curves over finite fields have gained much attention in public key cryptography in recent years [3], [10]. For practical reasons, elliptic curves over fields of characteristic two are of special interest. Diffie–Hellman type cryptosystems using elliptic curves over $\mathbb{F}_{2^{155}}$ were implemented and compared with RSA (see [12]). The most time consuming operation of these cryptosystems is multiplication of a point on the elliptic curve with an integer, whose size is approximately equal to the order of the underlying group. In [4] Koblitz proposed distinguishing between the field of definition for the elliptic curve E and the field for the group of points on E. He suggested using the group of points on so-called "anomalous" elliptic curves for such cryptosystems. In [7] the authors showed how to speed up point multiplication on the anomalous curve $y^2 + yx = x^3 + 1$, defined over $\mathbb{F}_2$.

* Part of this work was done when the author was a Postdoctoral Fellow at the University of Waterloo, Canada. The support of the Natural Sciences and Engineering Research Council of Canada (NSERC) is gratefully acknowledged.

In this paper we extend the ideas presented in [7] to arbitrary nonsupersingular elliptic curves defined over small extensions of $\mathbb{F}_2$. We describe a new algorithm for multiplying points that are defined over arbitrary extensions of the definition field. The new algorithm is up to 50% faster than the ordinary binary multiplication algorithm. We examine its running time both in theory and in practice. Finally, we list some examples of nonsupersingular elliptic curves defined over $\mathbb{F}_4$, $\mathbb{F}_8$, $\mathbb{F}_{16}$, respectively, and corresponding extension fields that are well suited for public key cryptosystems. All extension fields are chosen in such a way that the order of the group of points over the given extension field has a large prime factor. Therefore neither the algorithm of Pohlig and Hellman (see [11]) nor the attacks on supersingular curves (see [8]) can be used to compute discrete logarithms in the group of points on the given elliptic curves.

We start with a short introduction to elliptic curves. More information on elliptic curves over fields of characteristic two and their use in cryptography can be found in [9].

Let $\mathbb{F}_q$ be the finite field with q elements, where q is a "small" power of two, and let $\overline{\mathbb{F}}_q$ be its algebraic closure. A (nonsupersingular) elliptic curve E over $\mathbb{F}_q$ can be defined by an equation of the form

$$y^2 + x\,y = x^3 + a_2\,x^2 + a_6, \tag{1}$$

where a_2, $a_6 \in \mathbb{F}_q$ and $a_6 \neq 0$. The set $E(\mathbb{F}_{q^k})$ of points on E over an extension field $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$ is given by the set of solutions in $\mathbb{F}_{q^k}^2$ to (1) together with a "point at infinity" $\mathcal{O}$. There exist simple algebraic formulas for adding two arbitrary points in $E(\mathbb{F}_{q^k})$ (see [12]); with this addition, $E(\mathbb{F}_{q^k})$ forms an additive abelian group with zero element $\mathcal{O}$. The disadvantage of this *affine representation* is the fact that addition involves one inversion in $\mathbb{F}_{q^k}$. There exists another representation of points, the so-called *projective representation*, which eliminates this problem. On the other hand, point addition in projective representation needs more multiplications and squarings of elements in $\mathbb{F}_{q^k}$ as point addition in affine representation (see [9] for a detailed description of these representations and corresponding addition formulas).

The Frobenius endomorphism of E is given as

$$\begin{aligned} \Phi\colon E(\overline{\mathbb{F}}_q) &\longrightarrow E(\overline{\mathbb{F}}_q), \\ (x,\,y) &\longmapsto (x^q,\,y^q). \end{aligned} \tag{2}$$

This endomorphism satisfies the equation

$$\Phi^2 - c\cdot\Phi + q = 0, \tag{3}$$

where $c \in \mathbb{Z}$ and $|c| \leq 2\sqrt{q}$. There is a strong connection between this equation in the endomorphism ring of E and the order of the group of points on E over the field of definition, namely, we have $\#E(\mathbb{F}_q) = q + 1 - c$. Since E is nonsupersingular, the endomorphism ring of E is an order in the imaginary quadratic field $\mathbb{Q}(\sqrt{c^2 - 4q})$. Moreover, the trace c of Φ must be odd. Obviously, the ring $\mathbb{Z}[\Phi]$ is a subring of the endomorphism ring of E.

In Section 2 we show how to represent rational integers as power sums of the Frobenius endomorphism Φ. The main result of this paper is given in Theorem 1, where we prove the existence of such expansions and sharp upper bounds for their length. In Section 3 we use these expansions to develop a new algorithm for multiplying points on suitable elliptic curves. In Section 4 we discuss extensions to this algorithm which use block ideas. Moreover, we report on some practical results achieved with an implementation of the algorithm (Section 5). Finally, we list several elliptic curves which are well suited for cryptographic purposes.

2. Representing Integers as Power Sums of the Frobenius Endomorphism

Let q be a power of two, let c be an odd integer with $|c| \leq 2\sqrt{q}$, and let Φ be an element in the maximal order of the imaginary quadratic field $\mathbb{Q}(\sqrt{c^2 - 4q})$ with minimal polynomial given in (3). The first lemma proves the existence of a "division by Φ with remainder" in the ring $\mathbb{Z}[\Phi]$.

Lemma 1. *Let $s \in \mathbb{Z}[\Phi]$. There exists an integer $r \in \mathbb{Z}$, $-q/2 \leq r \leq q/2$, and an element $t \in \mathbb{Z}[\Phi]$ such that*

$$s = t \cdot \Phi + r.$$

If we choose $r \in \{-q/2 + 1, \dots, q/2\}$, then r and t are unique.

Proof. Let $s = s_1 + s_2 \cdot \Phi$ with $s_1,\ s_2 \in \mathbb{Z}$. We want to find integers $t_1,\ t_2,\ r \in \mathbb{Z}$, such that

$$s_1 + s_2 \cdot \Phi = (t_1 + t_2 \cdot \Phi) \cdot \Phi + r.$$

Using the minimal polynomial of Φ given in (3), we transform the right-hand side of this equation into

$$(t_1 + t_2 \cdot \Phi) \cdot \Phi + r = t_1 \cdot \Phi + t_2 \cdot (c\,\Phi - q) + r.$$

Comparing coefficients, we get $s_1 = -t_2\, q + r$. This equation must be solvable in the rational integers, therefore we have $r \equiv s_1 \bmod q$. Choosing r as the absolute smallest residue of $s_1 \bmod q$, we compute

$$t_2 = \frac{r - s_1}{q} \quad \text{and} \quad t_1 = s_2 - c\, t_2 = s_2 - c\, \frac{r - s_1}{q}.$$

The uniqueness of r and t follows from the fact that $\{-q/2 + 1, \dots, q/2\}$ forms a minimal complete set of residues modulo q. □

Obviously, one can iterate the process of divisions by Φ with remainder. This procedure leads to so-called Frobenius expansions for an element $s \in \mathbb{Z}[\Phi]$. We prove the existence of such expansions in the following theorem. Note that $\mathbb{Z}[\Phi]$ can be embedded into the complex numbers; define $||s||$ to be the euclidean length of s.

Theorem 1. *Let $q \geq 4$, and let $s \in \mathbb{Z}[\Phi]$. If we set $k = \lceil 2\log_q ||s|| \rceil + 3$, then there exist rational integers $r_j \in \{-q/2, \dots, q/2\}$, $0 \leq j \leq k$, such that*

$$s = \sum_{j=0}^{k} r_j \cdot \Phi^j.$$

Proof. We set $s_0 = s$, and define for $j \geq 0$ inductively the elements $s_{j+1} \in \mathbb{Z}[\Phi]$ by

$$s_j = s_{j+1} \cdot \Phi + r_j, \tag{4}$$

where $r_j \in \{-q/2, \ldots, q/2\}$. By Lemma 1, such integers r_j always exist. The elements s_j, $0 \leq j \leq i+1$, lead to an expansion of the form

$$s_0 = s_1 \cdot \Phi + r_0 = (s_2 \cdot \Phi + r_1) \cdot \Phi + r_0 = \cdots = \sum_{j=0}^{i} r_j \cdot \Phi^j + s_{i+1} \cdot \Phi^{i+1}.$$

Using the triangle inequality in (4) and observing $||r_i|| \leq q/2$, we obtain

$$||s_{i+1}|| \leq \frac{||s_i|| + ||r_i||}{\sqrt{q}} \leq \frac{||s_i|| + q/2}{\sqrt{q}}.$$

If we iterate this process, we can show per induction that

$$||s_{i+1}|| \leq \frac{||s_0||}{q^{(i+1)/2}} + \frac{1}{2} \cdot \sum_{j=-1}^{i-1} q^{-j/2}.$$

For $q \geq 4$, we bound this expression as

$$||s_{i+1}|| < \frac{||s_0||}{q^{(i+1)/2}} + \frac{1}{2}\left(\sqrt{q} + \frac{\sqrt{q}}{\sqrt{q}-1}\right) \leq \frac{||s_0||}{q^{(i+1)/2}} + \sqrt{q}.$$

Therefore we know that for $i \geq \lceil 2 \log_q ||s_0|| \rceil - 1$ the length of s_{i+1} is smaller than $\sqrt{q} + 1$.

Now we try to bound the length of expansions for elements of euclidean length bounded by $\sqrt{q} + 1$. Note that the norm of an arbitrary element $a + b \cdot \Phi \in \mathbb{Z}[\Phi]$ is given as $N(a + b \cdot \Phi) = ||a + b \cdot \Phi||^2$ and that

$$N(a + b \cdot \Phi) = a^2 + c\,a\,b + q\,b^2 = \left(a + \frac{c\,b}{2}\right)^2 + \tfrac{1}{4}(4q - c^2)\,b^2. \tag{5}$$

Therefore we have to examine the length of expansions for all elements in $\mathbb{Z}[\Phi]$ of norm smaller than $(\sqrt{q} + 1)^2$. Assume that $a + b \cdot \Phi$ is such an element. Since c is odd, it is easy to see by (5) that $|b| \leq q/2$. For $b = 0$, we have $|a| < \sqrt{q} + 1$. Since $\sqrt{q} + 1 < q/2$ for all $q \geq 8$ and $\sqrt{4} + 1 = 3$, we conclude that we have $|a| \leq q/2$ for all q, and the expansion length is therefore one.

So consider the situation $|b| = 1$. Then we obtain the upper bound $|a| < |c|/2 + \sqrt{q} + 1 \leq 2\sqrt{q} + 1 \leq 3/2\,q$. If $|a| \leq q/2$, then obviously $a \pm \Phi$ is itself a valid Frobenius expansion of length two. Otherwise we use the minimal polynomial (3) of Φ and obtain for $a > q/2$ (the case $a < -q/2$ can be treated symmetrically)

$$a \pm \Phi = (a - q) + (c \pm 1) \cdot \Phi - \Phi^2.$$

Now $a - q$ is in the correct range, but $|c \pm 1|$ might be bigger than $q/2$. However, note that there are only two situations when this can happen: $q = 4,\ c = \pm 3$ or $q = 8,\ c = \pm 5$.

In all other situations, the expansion for $a + \Phi$ satisfies all conditions, and its length is exactly 3. For the two special situations we get the expansions $a+\Phi = (a-4)+2\cdot\Phi^2-\Phi^3$ and $(a-8)+2\cdot\Phi+4\cdot\Phi^2-\Phi^3$ (and the symmetric cases), i.e., the length of these expansions is bounded by four. Therefore we have proven that all elements in $\mathbb{Z}[\Phi]$ of length smaller $\sqrt{q}+1$ have a Frobenius expansion of length at most four.

For $|b| \geq 2$ and $q = 4, 8, 16$, we can check per hand that all these elements have a Frobenius expansion of length at most three. For $q \geq 32$, we sharpen the upper bound for b (note that $(4q-c^2) \geq 3$) and obtain $|b| \leq 2(\sqrt{q}+1)/3$ and $|a| \geq q$. Testing the different possibilities for a, we check that there exists a Frobenius expansion for $a + b\Phi$ ofr length at most three.

Thus we can conclude that for $k \geq \lceil 2\log_q ||s_0|| \rceil + 3$ we have $s_{k+1} = 0$, and the theorem is proven. □

Theorem 1 states the existence of a Frobenius expansion of the given length. Nevertheless, it is not clear whether we really can compute this expansion in reasonable time. An algorithm for computing such an expansion will iterate the division with remainder given in (4). Unfortunately, there are two possibilities for the remainder r_j if $r_j \equiv q/2 \bmod q$. If we remember the proof of Theorem 1, then it does not matter which candidate we choose as long as the iteration index is smaller than $\lceil 2\log_q ||s_0|| \rceil - 1$. The shortest expansions for the remaining elements $a + b \cdot \Phi$ with $|a| \leq 3/2\, q$ and $|b| \leq 1$ can be precomputed and stored, such that it is possible to compute Frobenius expansions whose length is bounded by the values given in Theorem 1.

In the following corollaries, we sharpen the upper bound for the length of Frobenius expansions for the different values $q = 4$, $q = 8$, $q = 16$, and all possible traces. These better bounds are obtained by using the actual value of q, c in the proof of Theorem 1. Moreover, we restrict ourselves to computing expansions for rational integers s.

Corollary 1. *Let $q = 4$, and let $s \in \mathbb{Z} \subseteq \mathbb{Z}[\Phi]$. If $c = \pm 1$, then there exists a Frobenius expansion for s such that the index k of the "last" nonzero coefficient of this expansion satisfies $k \leq \lceil \log_2 |s| \rceil + 1$. For $c = \pm 3$, there exists a Frobenius expansion with $k \leq \lceil \log_2 |s| \rceil + 4$.*

Proof. We sharpen the bounds given in the proof of Theorem 1. First, we note that for $i = \lceil \log_2 |s| \rceil - 1$ the elements s_{i+1} have length smaller than three. For $c = \pm 1$, we get the following possibilities: $s_{i+1} \in \{0, \pm 1, \pm 2, \pm\Phi, \pm 1 \pm \Phi, -2 \pm \Phi\}$. For $c = \pm 3$, we compute this set as $\{0, \pm 1, \pm 2, \pm\Phi, \pm 1 \pm \Phi, -2 \pm \Phi, -2\Phi + \Phi^2, 1 - 2\Phi + \Phi^2, -2\Phi^2 + \Phi^3, 1 - 2\Phi^2 + \Phi^3\}$. □

Corollary 2. *Let $q = 8$, and let $s \in \mathbb{Z} \subseteq \mathbb{Z}[\Phi]$. There exists a Frobenius expansion for s such that the index k of the "last" nonzero coefficient of this expansion satisfies*

$$k \leq \begin{cases} \lceil \frac{2}{3} \log_2 |s| \rceil + 1 & \textit{for} \quad c = \pm 1, \pm 3, \\ \lceil \frac{2}{3} \log_2 |s| \rceil + 2 & \textit{for} \quad c = \pm 5. \end{cases}$$

Corollary 3. *Let $q = 16$, and let $s \in \mathbb{Z} \subseteq \mathbb{Z}[\Phi]$. There exists a Frobenius expansion for s such that the index k of the "last" nonzero coefficient of this expansion satisfies $k \leq \lceil \frac{1}{2} \log_2|s| \rceil + 1$.*

In the next section we show how we can use these Frobenius expansions to multiply points on elliptic curves defined over $\mathbb{F}_q$. We give a formal description of the algorithm, and we analyze the expected number of additions of this algorithm.

3. Multiplication of Points Using Frobenius Expansions

The main computational problem in elliptic curve public key cryptosystems is the computation of $m \cdot P$ for an integer $m \in \mathbb{Z}$ and a point $P \in E(\mathbb{F}_{q^k})$. Usually, the size of m is approximately equal to q^k, since we can reduce m modulo the group order of $E(\mathbb{F}_{q^k})$. The theorem of Hasse states that this group order is bounded by $q^k + 1 + 2\sqrt{q^k}$. Moreover, we can assume that m is positive (otherwise we negate m and P). In this section we explain how we can use Frobenius expansions for speeding up this multiplication on elliptic curves defined over small extensions of $\mathbb{F}_2$.

Let E be an elliptic curve defined over the field $\mathbb{F}_q$, where q is a "small" power of two, and let $\#E(\mathbb{F}_q) = q + 1 - c$. Assume that we want to compute $m \cdot P$ for some integer $m \in \mathbb{N}$ and a point $P \in E(\mathbb{F}_{q^k})$, where $\mathbb{F}_{q^k}$ is an extension field of $\mathbb{F}_q$. Let Φ be the Frobenius endomorphism of E. In Theorem 1 we showed that we can expand the integer m as a power sum of the Frobenius endomorphism Φ:

$$m = \sum_{j=0}^{k} m_j \cdot \Phi^j,$$

where the coefficients satisfy the condition $m_j \in \{-q/2, \ldots, q/2\}$. Since multiplication by m is an endomorphism and $\mathbb{Z} \subseteq \mathbb{Z}[\Phi] \subseteq \mathrm{End}(E)$, we can interpret this equation as an equation in the endomorphism ring of E. The most important observation in our context is the fact that the evaluation of $\Phi(P)$ can be done in $2 \log_2(q)$ squarings (compare (2)), which is—for small q—usually faster than doubling a point.

In [7] the authors showed, for the special choice $q = 2$, $c = 1$, how to compute a Frobenius expansion for an integer $m \in \mathbb{N}$, where the length of the expansion (i.e., the number of coefficients in the expansion) is bounded by $\lfloor 2 \log_2 m \rfloor + 1$. By a suitable reduction modulo $\#E(\mathbb{F}_{2^k})$, they could reduce the expansion length to $\min\{k - 1, \lfloor 2 \log_2 m \rfloor + 1\}$. The coefficients in this expansion are absolutely bounded by one. Therefore they could develop an ordinary "right-to-left binary" multiplication algorithm, which examines the bits of m from the low order to the high order bit, computes the corresponding coefficient of the Frobenius expansion of m, and uses this coefficient "on the fly" (see p. 339 of [7]).

If q is bigger than two, the situation is more complicated. Since in this case the coefficients of the Frobenius expansion might be bigger than one, additional point additions would be necessary if we would use a "right-to-left" technique. Therefore such an algorithm would in general be of no advantage compared with the binary method. Fortunately, these additions can be replaced by a table lookup if we use a "left-to-right" exponentiation variant. On the other hand, we know no way to compute a Frobenius expansion for an integer m by scanning the bits of m from the high order to the low order bit. Therefore the new algorithm consists of three different parts:

1. precompute a "small" table of multiples of P,
2. compute and store a Frobenius expansion for m, i.e., compute (and store) the coefficients $m_0, \ldots, m_k$ in ascending order ("right-to-left"),
3. use the table and the stored coefficients of this expansion in descending order to compute $m \cdot P$ ("left-to-right").

A C^{++}-like description of this idea is given in Algorithm 1. Let $r' = r \operatorname{Mod} q$ denote the rational integer $r' \in \{-q/2 + 1, \ldots, q/2\}$ that satisfies $r' \equiv r \bmod q$.

Algorithm 1 (Fast Multiplication using Frobenius Expansions)

E is an elliptic curve defined over $\mathbb{F}_q$, Frobenius endomorphism Φ for $E(\mathbb{F}_q)$ satisfies $\Phi^2 - c\,\Phi + q = 0$.

INPUT: $m \in \mathbb{N}$ and $P \in E(\mathbb{F}_{q^k})$.
OUTPUT: $m \cdot P$.

```
Precompute table of points
   (1)  compute and store $i \cdot P$ for all $1 \le i \le q/2$.
Compute Frobenius expansion
   (2)  set $s_1 = m,\ s_2 = 0$ and $i = 0$.
   (3)  while ($|s_1| > q$ or $|s_2| > q/2$) do
   (4)     compute and store $m_i = s_1 \operatorname{Mod} q$.
   (5)     set $h = (m_i - s_1)/q,\ i = i + 1,\ s_1 = s_2 - c \cdot h$ and $s_2 = h$.
   (6)  od
"Left-to-right" multiplication
   (7)  determine $H = s_2 \cdot \Phi(P) + s_1 \cdot P$.
   (8)  for ($j = i - 1$ downto 0) do
   (9)     if ($m_j \ge 0$) then
  (10)        set $H = \Phi(H) + m_j \cdot P$.
  (11)     else
  (12)        set $H = \Phi(H) - |m_j| \cdot P$.
  (13)  od
  (14)  return $H$
```

The correctness of Algorithm 1 follows directly from the observation that the **while**-loop in steps (3)–(6) is a direct translation of the sequence introduced in the proof of Theorem 1. Then observe that

$$\begin{aligned} m \cdot P &= \left(\sum_{i=0}^{k} m_i \cdot \Phi^i \right)(P) \\ &= \Phi(\cdots \Phi(\Phi(m_k \cdot \Phi(P) + m_{k-1} \cdot P) + m_{k-2} \cdot P) \cdots + m_1 \cdot P) + m_0 \cdot P. \end{aligned}$$

A time critical part of Algorithm 1 is the precomputation of the table in step (1). All rational integers r_j in steps (10) and (12) and s_1 in step (7) are absolutely bounded by

$q/2$. Therefore we do not have to compute the points $s_1 \cdot P$, $|r_j| \cdot P$, but we can take these points out of the precomputed table.

As already mentioned above, there is one ambiguity in the computation of the Frobenius expansion for m. In step (4) we might have two choices for r_i if $s_1 \equiv q/2 \bmod q$. Unfortunately, we do not know a priori what choice is the better one. One possibility to overcome this problem is to choose the value for r_i which minimizes the norm of the new element $s_1 + s_2 \cdot \Phi$. Another possible solution to this problem is not to care about this ambiguity and to examine the computed coefficients r_i in the second part. As a first step of the **for**-loop in step (8), we test whether $r_{j-2} = q/2$. If this condition is fulfilled, we check whether adding c to r_{j-1} and -1 to r_j does not violate the range condition for r_j, r_{j-1}. Such a substitution should then be done if the number of zero r-values increases (note that in this case we do not have to add a point in steps (10) and (12)). Obviously, there can be recursive effects, such that this heuristic is not necessary optimal.

We analyze the crossover point of Algorithm 1 compared with the ordinary binary method. First note that we do not have to care about the arithmetic operations for computing the Frobenius expansions.

Lemma 2. *All rational integer operations in Algorithm* 1 *with input* (m, P) *can be done in time* $O((\log_2 m)^2)$.

Proof. Note that all operations on rational integers (subtractions and division with remainder by q) can be done in linear time, since q is a power of two. The result follows by observing that the absolute maximum of s_1, s_2 in the **while**-loop (3)–(6) is bounded by the input m (proof per induction) and that the number of loop iterations is $O(\log m)$ (see Theorem 1). □

We can assume that the multiplier $m \in \mathbb{N}$ is reduced modulo the group order of the used curve. By Hasse's theorem, we know that $m \le q^k + 1 + 2\sqrt{q^k}$, i.e., all integer operations take approximately the same time as one multiplication in the finite field $\mathbb{F}_{q^k}$. Therefore we can ignore the complexity of the integer operations in the remainder of this section.

The more important part for the running time of Algorithm 1 is the number of point additions in $E(\mathbb{F}_{q^k})$ and the number of evaluations of the Frobenius endomorphism. It is obvious that the number of point additions performed by Algorithm 1 with input (m, P) is bounded by $q/2 + k_{q,c}(m) - 1$, where $k_{q,c}(m)$ is the maximal index in the Frobenius expansion for m (note that this index depends on q, c, see Corollaries 1, 2, and 3). Moreover, the number of Frobenius evaluations is bounded by $k_{q,c}(m)$. The following lemma counts the number of field operations for an addition and a doubling of a point (the exact formulas can be found in [12] and [9]).

Lemma 3. *In affine representation, one addition of two different points of* $E(\mathbb{F}_{q^k})$ *can be done with one inversion, one squaring, two multiplications, and eight additions in* $\mathbb{F}_{q^k}$*; doubling a point needs one inversion, two squarings, two multiplications, and four additions in* $\mathbb{F}_{q^k}$*. An addition of different points in projective representation involves two squarings, thirteen multiplications, and seven additions, doubling of points takes five squarings, seven multiplications, and four additions in* $\mathbb{F}_{q^k}$.

If we combine the previous results, we obtain the following upper bounds for the number of (quadratic) field operations for the different versions.

Theorem 2. *Assume that we want to compute $m \cdot P$ for an integer $m \in \mathbb{N}_{>1}$ and a point $P \in E(\mathbb{F}_{q^k})$ in affine representation. Then Algorithm* 1 *performs at most $q/2 + k_{q,c}(m)$ inversions, $q + 2\,k_{q,c}(m)$ multiplications, and $(2\log_2(q)+1)\cdot k_{q,c} + q/2$ many squarings in $\mathbb{F}_{q^k}$. If P is given in projective representation, Algorithm* 1 *needs at most $13(k_{q,c}(m) + q/2)$ multiplications and $2\,\log_2(q)\cdot k_{q,c}(m) + q - 2$ squarings in $\mathbb{F}_{q^k}$.*

We compare the maximal and expected number of field operations used in Algorithm 1 for $q = 4$, $q = 8$, and $q = 16$ to the ordinary binary method. The expected number of operations is computed under the assumption that about half the bits of a "random" multiplier are zero. For the new method, we assume that the numbers s_1 in the **for**-loop of Algorithm 1 behave like random integers such that about $1/(q+1)$ of all coefficients in a Frobenius expansion are zero. Note that we indeed observe this behavior in practice. The first table describes the affine situation, then we present these values for points in projective representation.

Operation	Binary method	$q=4$	$q=8$	$q=16$
Inversion				
Maximum	$2\log_2(m)$	$\log_2(m)+6$	$\frac{2}{3}\log_2(m)+7$	$\frac{1}{2}\log_2(m)+10$
Average	$\frac{3}{2}\log_2(m)$	$\frac{4}{5}\log_2(m)+4$	$\frac{16}{27}\log_2(m)+6$	$\frac{8}{17}\log_2(m)+8$
Squaring				
Maximum	$3\log_2(m)$	$5\log_2(m)+22$	$\frac{14}{3}\log_2(m)+25$	$\frac{9}{2}\log_2(m)+26$
Average	$\frac{5}{2}\log_2(m)$	$4\log_2(m)+13$	$\frac{112}{27}\log_2(m)+20$	$\frac{72}{17}\log_2(m)+16$
Multiplication				
Maximum	$4\log_2(m)$	$2\log_2(m)+12$	$\frac{4}{3}\log_2(m)+14$	$\log_2(m)+20$
Average	$3\log_2(m)$	$\frac{8}{5}\log_2(m)+7$	$\frac{32}{27}\log_2(m)+12$	$\frac{16}{17}\log_2(m)+16$

Operation	Binary Method	$q=4$	$q=8$	$q=16$
Squaring				
Maximum	$7\log_2(m)$	$6\log_2(m)+26$	$\frac{16}{3}\log_2(m)+30$	$5\log_2(m)+54$
Average	$6\log_2(m)$	$\frac{28}{5}\log_2(m)+19$	$\frac{124}{27}\log_2(m)+24$	$\frac{76}{17}\log_2(m)+39$
Multiplication				
Maximum	$20\log_2(m)$	$13\log_2(m)+65$	$\frac{26}{3}\log_2(m)+78$	$\frac{13}{2}\log_2(m)+117$
Average	$\frac{27}{2}\log_2(m)$	$\frac{52}{5}\log_2(m)+52$	$\frac{208}{27}\log_2(m)+70$	$\frac{105}{17}\log_2(m)+111$

Obviously, the ordinary binary method can also be accelerated by using a precomputed table and a q-ary decomposition of the multiplier. However, such a strategy does not decrease the number of doublings of points. The table only decreases the number of points additions. In Section 5 we give timings for both variants of the binary algorithm.

Nevertheless, the main advantage of Algorithm 1 is the fact that the number of quadratic field operations in $\mathbb{F}_{q^k}$ is smaller than in the binary algorithm. Note especially that in affine representation the number of inversions decreases by about a half. These theoretical complexity improvements can also be seen in practice, as we show in Section 5.

4. Variants of Algorithm 1

4.1. *The Block Variant*

Several authors have used block techniques for accelerating the ordinary binary method for multiplication of points (for an overview, see [1]). In these methods, several bits of the multiplier are processed in one iteration step. We can use similar techniques to develop a block Frobenius expansion algorithm.

Again, the first part of the block version consists of computing a Frobenius expansion for the given integer m. Instead of using this expansion coefficient per coefficient, we simultaneously use blocks of coefficients in the second part of the algorithm. Assume that the Frobenius expansion for $m \in \mathbb{N}$ is given as in Section 3, and that we use a blocking factor $s \geq 1$. Then we have

$$m \cdot P = \sum_{i=0}^{\lfloor k/s \rfloor} \Phi^{is} \left(\sum_{j=0}^{s-1} m_{is+j} \cdot \Phi^{j}(P) \right).$$

In the precomputation step we have to compute and store all nonzero points which might occur in the "inner sum." It is easy to see that the number of points which we have to store is $((q+1)^s - 1)/2$ (note that we should make use of the fact that the point $-Q$ can easily be derived from Q).

Theorem 3. *Let $m \in \mathbb{N}$ and $P \in E(\mathbb{F}_{q^k})$. If we use a block variant of Algorithm* 1 *with blocking factor $s \geq 1$, then this variant performs at most*

$$\frac{(q+1)^s - 3}{2} + \frac{2\log_q(m) + 4}{s} - 1$$

many point additions and $2\log_q(m) + s + 3$ *Frobenius evaluations.*

Since the number of Frobenius evaluations is approximately the same for different blocking factors, we concentrate on the number of point additions. The following table lists the maximal number of points additions for a few possibilities for q and s, and the sizes of the corresponding tables.

s	$q = 4$	$\#T_4$	$q = 8$	$\#T_8$	$q = 16$	$\#T_{16}$
1	$\log_2(m) + 4$	2	$\frac{2}{3}\log_2(m) + 7$	4	$\frac{1}{2}\log_2(m) + 11$	8
2	$\frac{1}{2}\log_2(m) + 12$	12	$\frac{1}{3}\log_2(m) + 40$	40	$\frac{1}{4}\log_2(m) + 144$	144
3	$\frac{1}{3}\log_2(m) + 61$	62	$\frac{2}{9}\log_2(m) + 363$	364	$\frac{1}{6}\log_2(m) + 2455$	2456

It is obvious that large block sizes lead to a huge precomputed table. Therefore this variant of Algorithm 1 is restricted to the situation where multiples of one fixed point are computed. Another algorithm for this special situation is described in the following section.

4.2. *Several Multiplications of a Fixed Point*

In [6] the authors present a fast exponentiation techniques which reduces the number of multiplications if we want to exponentiate a fixed element several times. We can extend these ideas to our situation.

Assume that the point $P \in E(\mathbb{F}_{q^k})$ is fixed, and that we want to compute $m \cdot P$ for several integers $m \in \mathbb{N}$. Again we use a precomputation part, which reduces the number of additions by about a half. Let k be an upper bound for the maximal length of all Frobenius expansions for the possible multipliers m. We set $k' = \lfloor k/2 \rfloor$ and precompute the point $P' = \Phi^{k'}(P)$. Additionally, we precompute a table of points $i \cdot P + j \cdot P'$ for $-q/2 \leq i,\ j \leq q/2$. Since the base point P is fixed, these tables can be initialized before the actual multiplication starts.

For computing $m \cdot P$, we again use a two round method: first we compute all coefficients $m_0, \ldots, m_k$ of the Frobenius expansion for m. The second part of the algorithm uses the equation

$$\begin{aligned} m \cdot P &= \sum_{i=0}^{k} m_i\, \Phi^i(P) = \sum_{i=0}^{k'-1} m_i\, \Phi^i(P) + \sum_{i=0}^{k-k'} m_{i+k'}\, \Phi^i(P') \\ &= \sum_{i=0}^{k'-1} \Phi^i(m_i \cdot P + m_{i+k'} \cdot P') + \sum_{i=2k'}^{k} \Phi^i(m_i \cdot P). \end{aligned} \tag{6}$$

If we evaluate these sums “from the high index to the low index,” we can use the table of precomputed points to reduce the maximal number of required additions to approximately half of the number of additions which Algorithm 1 would need.

Obviously, it is again possible to develop a block variant of this idea. In such a variant, k' would be chosen as $\lfloor k/s \rfloor$ for some blocking factor $s \in \mathbb{N}_{\geq 2}$. Then we can split the sum (6) into s subsums and thus reduce the number of additions to $\lfloor k/s \rfloor$. On the other hand, the size of the precomputed table will grow.

5. Running Times

We present some running times achieved with our implementation of Algorithm 1. As underlying field arithmetic, we use an implementation written by Kirsch (see [2], also included in LiDIA, see [5]). We have implemented the fast multiplication algorithms of this paper for points in both affine and projective representation.

The following table presents average running times for multiplication of a point in affine representation using the different methods. We choose the elliptic curve $y^2 + x\,y = x^3 + x^2 + 1$ with group order $\#E(\mathbb{F}_2) = 2$. If we set $\#E(\mathbb{F}_{2^i}) = 2^i + 1 - c_i$, then we have the following values for the Frobenius traces: $c_1 = 1$, $c_2 = -3$, $c_3 = -5$, $c_4 = 1$, and $c_5 = 11$. We use the following test: we multiply a random point (in affine representation) in the group of points $E(\mathbb{F}_{2^k})$ for the given elliptic curve with 100 randomly chosen integers of size $< 2^k$. In addition to the usual binary method, we also give timings for a binary method with precomputed tables of size 4 and 8, respectively. The table shows the average time for one such multiplication in milliseconds (on a sparc4).

		Binary	Binary				
Field	Binary	with T_4	with T_8	$q = 4$	$q = 8$	$q = 16$	$q = 32$
$\mathbb{F}_{2^{60}}$	27.2	25.1	24.7	16.5	13.6	12.4	13.8
$\mathbb{F}_{2^{120}}$	85.3	78.6	75.0	48.9	40.2	35.1	34.2
$\mathbb{F}_{2^{180}}$	210.3	193.8	184.0	117.4	94.2	79.8	75.0

The results show that the running time improvements of Algorithm 1 are not an effect of the usage of tables, but of the Frobenius expansions of the multipliers. In the following table we compare the average number of operations used in the different methods. We list the average number of point additions (A), point doublings (D), and Frobenius evaluations (F_q) (note that these evaluations depend on q), respectively.

Field	Binary method	$q = 4$	$q = 8$	$q = 16$	$q = 32$
$\mathbb{F}_{2^{60}}$	29.1 A	43.8 A	36.3 A	34.1 A	38.1 A
	58.0 D	58.6 F_4	38.8 F_8	29.0 F_{16}	23.8 F_{32}
$\mathbb{F}_{2^{120}}$	59.9 A	86.8 A	71.7 A	61.4 A	61.3 A
	117.9 D	118.8 F_4	79.5 F_8	58.8 F_{16}	48.0 F_{32}
$\mathbb{F}_{2^{180}}$	90.3 A	131.9 A	107.6 A	89.7 A	84.8 A
	177.9 D	178.8 F_4	119.7 F_8	88.8 F_{16}	72.1 F_{32}

The next table lists running times for the same test as above, but now we use points in projective representation. Since inversion is relatively slow in the chosen field implementation, absolute running times in projective representation are superior to affine representation. Nevertheless, the timing differences between the different methods remain approximately the same.

		Binary	Binary				
Field	Binary	with T_4	with T_8	$q = 4$	$q = 8$	$q = 16$	$q = 32$
$\mathbb{F}_{2^{60}}$	19.7	14.8	13.4	12.3	9.2	8.1	8.9
$\mathbb{F}_{2^{120}}$	79.4	58.2	52.2	46.3	33.6	27.6	26.5
$\mathbb{F}_{2^{180}}$	212.4	154.1	136.3	117.6	84.0	67.2	59.2

Finally we present some timings for the block version of the new algorithm which we have described in Section 4.1. The average running times are determined with exactly the same strategy as before. Note that we only show timings for affine representation of points and block size two.

Field	$q = 4$	Block version for $q = 4$	$q = 8$	Block version for $q = 8$
$\mathbb{F}_{2^{60}}$	16.5	15.0	13.6	21.9
$\mathbb{F}_{2^{120}}$	48.9	40.8	40.2	47.7
$\mathbb{F}_{2^{180}}$	117.4	91.5	94.2	95.7

It is obvious that the size of the precomputed table in the block variant of Algorithm 1 causes these negative impacts on the running time for $q > 4$. In the next section we describe the possible usage of the ideas of this paper in elliptic curve public key cryptosystems.

6. Elliptic Curves for Public Key Cryptography

Algorithm 1 can be used for multiplication in the group of points $E(\mathbb{F}_{q^k})$, when the given elliptic curve is defined over a "small" field $\mathbb{F}_q$. For public key cryptography, one additional requirement is necessary, since the security of these cryptosystems depends on the difficulty of the discrete logarithm problem in $E(\mathbb{F}_{q^k})$. If the group order of $E(\mathbb{F}_{q^k})$ is smooth (i.e., all prime factors of $\#E(\mathbb{F}_{q^k})$ are "small"), then the algorithm of Pohlig–Hellman (see [11]) can be used to solve the DL problem.

The order of the group $E(\mathbb{F}_{q^k})$ can easily be computed if we know the group order of $E(\mathbb{F}_q)$. Let $c_0 = 2$, $c_1 = q + 1 - \#E(\mathbb{F}_q)$, and define, for $i \geq 2$,

$$c_i = c_1 \cdot c_{i-1} - q \cdot c_{i-2}.$$

Then the group order of $E(\mathbb{F}_{q^k})$ is given as $\#E(\mathbb{F}_{q^k}) = q^k + 1 - c_k$. In this section we tabulate "cryptographically good" elliptic curves over the different base fields $\mathbb{F}_q$. We show that there exist nonsupersingular elliptic curves defined over $\mathbb{F}_q$ for $q = 4$, $q = 8$, $q = 16$, and $q = 32$, such that the group order of $E(\mathbb{F}_{q^k})$ for some extension degree k "of reasonable size" is divided by a large prime factor of length at least 155 bit. These groups of points are well suited as the basis for elliptic curve public key cryptosystems, since—as far as I know—no attack on the discrete logarithm problem in these groups is known.

We list the trace of the Frobenius endomorphism for an elliptic curve defined over $\mathbb{F}_q$ and the prime factorization of the order of the group of points $E(\mathbb{F}_{q^k})$ for given extension degree k. Actual equations for these curves can be found in the Appendix.

6.1. *The Case* $q = 4$

c	Extension degree over $\mathbb{F}_4$	Order of group of points over extension field
1	79	$2^2 \cdot 91343852333181432387730573045979447452365303319$
1	97	$2^2 \cdot 14551 \cdot 431386278289236531086233896175787116535934904510\backslash$ 183171

6.2. *The Case $q = 8$*

c	Extension degree over $\mathbb{F}_8$	Order of group of points over extension field
-3	73	$2^2 \cdot 3 \cdot 70208194445704791131945351789113626814156914\backslash$ 7453471605594551875 21
-1	59	$2 \cdot 5 \cdot 1915619426082361072947933791574731837504813 70\backslash$ 80701777
-1	71	$2 \cdot 5 \cdot 1279 \cdot 102924444554883880666456242849560534192\backslash$ 331880885509014284 5407
3	59	$2 \cdot 3 \cdot 3192699043470601788246556321152115972353471 56\backslash$ 89440269

6.3. *The Case $q = 16$*

c	Extension degree over $\mathbb{F}_{16}$	Order of group of points over extension field
-1	47	$2 \cdot 3^2 \cdot 280121 \cdot 778074194250317120378806220180330135343\backslash$ 22754237321
7	47	$2 \cdot 5 \cdot 392318858461667547739736838942997715128064667 93\backslash$ 403150729
7	53	$2 \cdot 5 \cdot 17424917 \cdot 37773598745318696029483966725519639943\backslash$ 987745326964290877

6.4. *The Case $q = 32$*

c	Extension degree over $\mathbb{F}_{32}$	Order of group of points over extension field
-1	47	$2 \cdot 17 \cdot 1693 \cdot 95920869278907109376807697609909669607 7\backslash$ 0758408153512162298132328 21
5	43	$2^2 \cdot 7 \cdot 1880576636938521191034250494351269861683593 68\backslash$ 2897108316059976983
9	41	$2^3 \cdot 3 \cdot 83 \cdot 2581426577122875944645722236692620836442 31\backslash$ 89254186044190931

6.5. *Using Elliptic Curves Proposed by Menezes*

In [9, Example 6.2, p. 89] Menezes proposes using suitable elliptic curves defined over $\mathbb{F}_{32}$ for a public key cryptosystem working in $\mathbb{F}_{2^{155}}$. We tested one of the proposed curves and obtained the following running times: For one multiplication of a random point in $E(\mathbb{F}_{2^{155}})$ with a random integer of size $\approx 2^{155}$, our implementation of the usual binary method needed on average 176.9 milliseconds in affine representation

(174.7 milliseconds in projective representation), Algorithm 1 took on average 62.9 milliseconds (52.3 milliseconds in projective representation). These practical results show that the new algorithm leads to a real improvement for elliptic curve cryptosystems in practical use.

Acknowledgments

I am especially grateful to Scott A. Vanstone who pointed me to the problem of generalizing the results of Koblitz and Meier et al., and I would like to thank Ingrid Biehl from the Universität des Saarlandes, Germany, for helpful discussions concerning the proof of Theorem 1.

Appendix

We present one example of an elliptic curve suitable for cryptographic systems for all choices for q, c as described in Section 6. The finite fields $\mathbb{F}_{q^k}$ are given by a generating irreducible polynomial over $\mathbb{F}_2$, elements in the field are represented as polynomials. We describe these polynomials over $\mathbb{F}_2$ in "decimal representation," i.e., the number $a \in \mathbb{N}$ represents the polynomial induced by the binary representation of a. All the computations were done with the help of LiDIA (for a description, see [5]).

The Case $q = 4$

c	Generating polynomial	a_2	a_6
1	$x^{158} + x^{76} + x^{33} + x^{32} + 1$	0	8598277310257033540808160551297811186\ 46444597228
1	$x^{194} + x^{87} + 1$	0	30910297718908258831771251207717368 7\ 3452974439586799473446

The Case $q = 8$

c	Generating polynomial	a_2	a_6
-3	$x^{219} + x^{54} + x^{33} + x^{32} + 1$	0	28011598142057797575194694158183958110 80\ 614606270824806665177 28685
-1	$x^{177} + x^{88} + 1$	1	3181931642933075904430737403863671449228\ 8809306721799
-1	$x^{213} + x^{75} + x^{33} + x^{32} + 1$	1	3174935408637498283622005592879686207788\ 398581481910767191717662
3	$x^{177} + x^{88} + 1$	1	1526262677713764696299159372597171337406\ 65058561311972

The Case $q = 16$

c	Generating polynomial	a_2	a_6
-1	$x^{188} + x^{46} + x^{33} + x^{32} + 1$	89704831002846141956257\ 35597872677711832704243\ 1878979890	1
7	$x^{188} + x^{46} + x^{33} + x^{32} + 1$	26932309050291623696669\ 96469995611643753389444\ 6053375083648	31565028309007051813343 7\ 22336203883025061822241\ 76443616196
7	$x^{212} + x^{105} + 1$	59130727864844554885839\ 25517904640662821600611\ 8114191535279284548	20492200966434650611611\ 83438375656795599974674\ 683239185599768319

The Case $q = 32$

c	Generating polynomial	a_2	a_6
-1	$x^{235} + x^{34} + x^{33} + x^{32} + 1$	1	28367676456631108659390994280210 59\ 9577533192290815616725673561575651998
5	$x^{215} + x^{51} + 1$	0	13432958756252252067374509727840 92\ 4330763383981049950024826287335
9	$x^{205} + x^{94} + x^{33} + x^{32} + 1$	0	45898028521286731224066053153858 79\ 3402811962616419539897036781

References

[1] D. Fox, A.W. Röhm: Effiziente Digitale Signatursysteme auf der Basis elliptischer Kurven, in *Digitale Signaturen*, P. Horster, ed., DuD Fachberichte, Vieweg, 1996, pp. 256–266.

[2] P. Kirsch: Implementierung einer Arithmetik des Polynomrings $GF(2)[X]$ und des Körpers $GF(2^n)$, Diplomarbeit, Universität des Saarlandes, Saarbrücken, 1996.

[3] N. Koblitz: Elliptic Curve Cryptosystems, *Math. Comp.*, **48** (1987), 203–209.

[4] N. Koblitz: CM-Curves with Good Cryptographic Properties, *Advances in Cryptology—CRYPTO* 91, Lecture Notes in Computer Science, No. 576, Springer-Verlag, Berlin, 1992, pp. 279–287.

[5] LiDIA—A Library for Computational Number Theory, available per ftp from `http://www.informatik.tu-darmstadt.de/TI/`.

[6] C.H. Lim, P.J. Lee: More Flexible Exponentiation with Precomputation, *Advances in Cryptology—CRYPTO* 94, Lecture Notes in Computer Science, No. 839, Springer-Verlag, Berlin, 1994, pp. 95–107.

[7] W. Meier, O. Staffelbach: Efficient Multiplication on Certain Nonsupersingular Elliptic Curves, *Advances in Cryptology—CRYPTO* 92, Lecture Notes in Computer Science, No. 740, Springer-Verlag, Berlin, 1992, pp. 333–344.

[8] A. Menezes, T. Okamoto, S.A. Vanstone: Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field, *Proceedings of the* 23rd *ACM Symp. on Theory of Computing*, 1991, pp. 1639–1646.

[9] A. Menezes: *Elliptic Curve Public Key Cryptosystems*, Kluwer, Dordrecht, 1993.

[10] V.S. Miller: Use of Elliptic Curves in Cryptography, *Advances in Cryptology—CRYPTO* 85, Lecture Notes in Computer Science, No. 218, Springer-Verlag, Berlin, 1986, pp. 417–426.

[11] S.C. Pohlig, M.E. Hellman: An Improved Algorithm for Computing Logarithms over $GF(p)$ and Its Cryptographic Significance, *IEEE Trans. Inform. Theory*, **24** (1978), 106–110.

[12] R. Schroeppel, H. Orman, S. O'Malley, O. Spatschek: Fast Key Exchange with Elliptic Curve Systems, *Advances in Cryptology—CRYPTO* 95, Lecture Notes in Computer Science, No. 963, Springer-Verlag, Berlin, 1995, pp. 43–56.