MATHEMATICS OF COMPUTATION
Volume 00, Number 0, Pages 000–000
S 0025-5718(XX)0000-0

THE SEA ALGORITHM IN CHARACTERISTIC 2

FREDERIK VERCAUTEREN

Abstract. The Schoof-Elkies-Atkin algorithm counts the number of rational points on elliptic curves over finite fields. This paper presents a number of optimizations specific for the characteristic two case. We give a detailed description of the computation of modular polynomials over $\mathbb{F}_{2^n}$ and the search for an eigenvalue in the Elkies algorithm. With our implementation, we were able to count the number of rational points on a curve defined over $\mathbb{F}_{2^{1999}}$, which is the current world record for the characteristic two case.

1. Introduction

Public-key cryptosystems based on elliptic curves (ECC) over finite fields, first proposed by Koblitz [15] and Miller [24], offer a similar level of security as other systems such as RSA [28], but with the benefit of much smaller key sizes. This is due to the fact that there is no known sub-exponential algorithm to solve the discrete logarithm problem on a general elliptic curve. The best known general attacks on ECC's have running time proportional to the square root of the largest prime factor dividing the group order. Thus the security of an ECC mainly depends on the number of rational points on the elliptic curve and therefore it is necessary to explicitly determine this number of rational points whilst generating secure elliptic curves. Furthermore, most cryptographic schemes based on elliptic curves rely on the order of the curve.

Several methods for constructing elliptic curves with known cardinality, such as the complex multiplication method [5, 17, 25] or supersingular curves [2, 23], introduce an extra structure on the curve which could be used to break the cryptosystem [11, 22, 29, 36]. Therefore it is believed that the Schoof-Elkies-Atkin (`SEA`) algorithm is the best way to generate secure elliptic curves with nearly prime cardinality, because it can determine the group order of a randomly chosen curve.

Although the original Schoof algorithm [31] has polynomial running time, it turns out to be inefficient in practice. Thanks to the ideas of Elkies [9, 10] and Atkin [1, 32] and the improvements of Morain [26], Couveignes [6], Dewaghe [8], Lercier [19], Müller [27], ..., it is now possible to generate secure elliptic curves in a reasonable time. In this article we present an overview of the `SEA` algorithm and give optimizations specific for the characteristic two case. We describe an implementation of this algorithm and give accurate statistics in the range of interest to cryptography, i.e., for elliptic curves defined over $\mathbb{F}_{2^n}$, with $163 \leq n \leq 431$. We

Received by the editor January 18, 2000.
2000 *Mathematics Subject Classification.* Primary 14H52, 11G20; Secondary 11T71, 11Y40.
Key words and phrases. Elliptic curves, finite fields, SEA algorithm.
The author is sponsored by the Fund for Scientific Research - Flanders.

©1997 American Mathematical Society

also report on the use of our implementation to count the number of rational points on a curve defined over $\mathbb{F}_{2^{1999}}$ in approximately 65 days on a Pentium II 400 MHz.

The remainder of the article is structured as follows: in Section 2 we recall the basic facts about elliptic curves over $\mathbb{F}_{2^n}$ and give a high level description of the `SEA` algorithm. In Section 3 we explain in detail how to compute the modular polynomials over $\mathbb{F}_{2^n}$ and present an optimized version of Müller's baby-step giant-step algorithm for finding an eigenvalue in the Elkies case. Section 4 describes details of our implementation and gives running times for elliptic curves suitable for public-key cryptography.

2. Overview of the `SEA` algorithm

2.1. **Elliptic curves over $\mathbb{F}_{2^n}$.** Let $\mathbb{F}_q$ be the finite field of characteristic two with $q = 2^n$ elements. Let E_a be an elliptic curve defined over $\mathbb{F}_q$ by the affine equation

$$E_a(x, y) \;:\; y^2 + xy - x^3 - a = 0, \tag{2.1}$$

where $a \in \mathbb{F}_q^*$. Then E_a is non-supersingular with j-invariant $j_a = 1/a$ and discriminant $\Delta_a = a$.

Let $\overline{\mathbb{F}}_q$ be the algebraic closure of $\mathbb{F}_q$ and for every field $\mathbb{F}_q \subset \mathbb{K} \subset \overline{\mathbb{F}}_q$ define the set of $\mathbb{K}$-rational points as

$$E_a(\mathbb{K}) = \{(x, y) \in \mathbb{K} \times \mathbb{K} \mid E_a(x, y) = 0\} \cup \{\mathcal{O}\}, \tag{2.2}$$

where $\mathcal{O}$ denotes the point at infinity. Every set $E_a(\mathbb{K})$ forms an Abelian group with addition law given by the tangent-and-chord method and with $\mathcal{O}$ acting as neutral element.

The Frobenius map is defined as

$$\varphi \;:\; E_a(\overline{\mathbb{F}}_q) \longrightarrow E_a(\overline{\mathbb{F}}_q) \;:\; (x, y) \mapsto (x^q, y^q). \tag{2.3}$$

It is easy to show that this is a group endomorphism of E_a over $\mathbb{F}_q$, hence the name Frobenius endomorphism. This map satisfies the quadratic relation in the endomorphism ring of E_a

$$\varphi^2 - [t]\varphi + [q] = [0], \tag{2.4}$$

where $[m]$ is the multiplication-by-m map. The integer t is called the trace of Frobenius and is linked to the number of $\mathbb{F}_q$-rational points $\#E_a(\mathbb{F}_q)$ by

$$\#E_a(\mathbb{F}_q) = q + 1 - t. \tag{2.5}$$

Furthermore, the theorem of Hasse [35] guarantees that $|t| \leq 2\sqrt{q}$.

2.2. **The `SEA` algorithm.** Schoof considered the restriction φ_l of φ to the l-torsion subgroup $E_a[l] = \{P \in E_a(\overline{\mathbb{F}}_q) \mid [l]P = \mathcal{O}\}$ of E_a, where l is an odd prime number. It is well known that $E_a[l] \cong \mathbb{Z}/l\mathbb{Z} \times \mathbb{Z}/l\mathbb{Z}$, so $E_a[l]$ has exactly $l+1$ cyclic subgroups $C_i, 1 \leq i \leq l+1$ of order l. Equation (2.4) reduces over $E_a[l]$ to

$$\varphi_l^2 - [t_l]\varphi_l + [q_l] = 0, \tag{2.6}$$

where $t_l \equiv t \bmod l$ and $q_l \equiv q \bmod l$. Schoof determines $t \bmod l$ by searching an integer $0 \leq \tau_l < l$ which satisfies

$$\varphi_l^2(P) + [q_l](P) = [\tau_l]\varphi_l(P), \tag{2.7}$$

for a point $P \in E_a[l]^*$. By Hasse's theorem we know that $|t| \leq 2\sqrt{q}$, so it is sufficient to compute $t \bmod l_i$ for small primes l_i until their product exceeds $4\sqrt{q}$. Finally, by means of the Chinese Remainder Theorem one can deduce t.

Denote by f_l the l-th division polynomial [16], which has degree $(l^2-1)/2$ and vanishes on the non-zero x-coordinates of the l-torsion points $P \in E_a[l]$. Then all computations for verifying Equation (2.7) take place in the polynomial ring

$$\mathbb{F}_q[x,y]/(f_l(x), E_a(x,y)). \tag{2.8}$$

The complexity of the original Schoof algorithm is $O(\log^8 q)$, but despite this polynomial running time, it turns out to be inefficient for cryptographic purposes. This inefficiency is caused by the quadratic growth of the degree of f_l.

The ideas of Atkin [1] and Elkies [9, 10] focus on the splitting of the characteristic equation of φ_l over $\mathbb{F}_l$. If the discriminant $\Delta_l = t_l^2 - 4q_l$ is a square modulo l, then l is called an Elkies prime, otherwise l is called an Atkin prime. If l is an Elkies prime, then φ_l has an eigenvalue λ in $\mathbb{F}_l$ and the corresponding eigenspace $E_a[l]_\lambda$ (one of the subgroups C_i) is stable under the Frobenius endomorphism. Define

$$g_l(x) = \prod_{\pm P \in E_a[l]^*_\lambda} (x - x(P)) \tag{2.9}$$

where only one of each pair $\pm P$ is included, because both P and $-P$ have the same x-coordinate. Then it follows that $g_l \in \mathbb{F}_q[x]$ and has degree $(l-1)/2$. If C is a finite subgroup of $E_a(\overline{\mathbb{F}}_q)$ which is Galois stable over $\mathbb{F}_q$, then there always exists an elliptic curve E'_a defined over $\mathbb{F}_q$ and an isogenie $\mathcal{I} : E_a \longrightarrow E'_a$ with kernel equal to C. Applying this to $E_a[l]_\lambda$, we can compute g_l once we have found the corresponding elliptic curve $E'_{a,\lambda}$ and an isogenie $\mathcal{I}_\lambda$, such that $\ker(\mathcal{I}_\lambda) = E_a[l]_\lambda$. The problem of finding $E'_{a,\lambda}$ is solved by the following theorem taken from Schoof [32].

Theorem 2.1. *Let $\overline{\Phi}_l(x,y) = 0$ be the l-th modular polynomial over $\mathbb{F}_q$ and let E be a non-supersingular elliptic curve over $\mathbb{F}_q$. Then there exists an isogenous curve E' and an isogenie from E to E' whose kernel C is cyclic of order l if and only if $\overline{\Phi}_l(j_{E'}, j_E) = 0$. Furthermore, $j_{E'} \in \mathbb{F}_{q^r}$ if and only if the kernel C is a one dimensional eigenspace of φ^r in $E[l]$.*

The above theorem provides a way to classify l as an Elkies or an Atkin prime. If $\overline{\Phi}_l(x, j_E) = 0$ has a root in $\mathbb{F}_q$, then l is an Elkies prime, otherwise l is an Atkin prime. Note that the degree of $\overline{\Phi}_l(x, j_E)$ is $l+1$.

The following theorem by Atkin is the actual heart of the `SEA` algorithm. A proof of the theorem can be found in Schoof [32].

Theorem 2.2 (Atkin). *Let E be a non-supersingular elliptic curve defined over $\mathbb{F}_q$ and let $\overline{\Phi}_l(x, j_E) = h_1 h_2 \cdots h_s$ be the factorization of $\Phi_l(x, j_E) \in \mathbb{F}_q[x]$ as a product of irreducible polynomials. Then there are the following possibilities for the degrees of $h_1, \ldots, h_s$:*

1. $(\mathbf{1}, \mathbf{l})$ *or* $(\mathbf{1}, \mathbf{1}, \ldots \mathbf{1})$ *— in either case we have $t^2 - 4q \equiv 0 \mod l$. In the former we set $r = l$ and in the latter $r = 1$.*
2. $(\mathbf{1}, \mathbf{1}, \mathbf{r}, \mathbf{r}, \ldots, \mathbf{r})$ *— in this case $t^2 - 4q$ is square modulo l, r divides $l-1$ and φ_l acts on $E[l]$ as a diagonal matrix $\begin{pmatrix} \lambda & 0 \\ 0 & \mu \end{pmatrix}$ with $\lambda, \mu \in \mathbb{F}_l^*$.*
3. $(\mathbf{r}, \mathbf{r}, \ldots, \mathbf{r})$ *for some $r > 1$ — in this case $t^2 - 4q$ is a non-square modulo l, r divides $l+1$ and φ_l has an irreducible characteristic polynomial over $\mathbb{F}_l$.*

In all cases r is the order of φ_l in the projective general linear group $\mathrm{PGL}_2(\mathbb{F}_l)$ and the trace of Frobenius t satisfies

$$t^2 = q(\xi + \xi^{-1})^2 \bmod l, \tag{2.10}$$

for some primitive r-th root of unity $\xi \in \overline{\mathbb{F}}_l$.

Note that Equation (2.10) limits the number of possible values for $t \bmod l$ to $\phi_{\mathrm{Eul}}(r)$, where ϕ_{Eul} is the Euler totient function. To determine the exact value of t, one merges the information found from both types of primes using a baby-step giant-step strategy called the match and sort algorithm. A detailed description can be found in Müller's thesis [27].

The original Schoof algorithm, combined with the above improvements of Elkies and Atkin, is called the SEA algorithm and has complexity $O(\log^6 q)$. Theorems 2.1 and 2.2 directly lead to Algorithm 2.1, which we describe in more detail in the following section.

Algorithm 2.1 (SEA).

```
IN:   Elliptic curve $E_a : y^2 + xy = x^3 + a$ over $\mathbb{F}_q$
OUT:  The order $\#E_a(\mathbb{F}_q)$ of $E_a(\mathbb{F}_q)$
```

```
 1. $l = 2$, $M_A = 1$, $A = \{\}$, $M_E = 1$ and $E = \{\}$
 2. While ($M_E \times M_A < 4\sqrt{q}$) do:
 3.     Compute the modular polynomial $\overline{\Phi}_l(x, y)$
 4.     Find the splitting of $\overline{\Phi}_l(x, j_a)$
 5.     If $l$ is an Elkies prime then do:
 6.         Find a zero $j_b$ of $\overline{\Phi}_l(x, j_a)$ in $\mathbb{F}_q$
 7.         Determine an isogenie $\mathcal{I}_l : E_a \longrightarrow E_b$ and $g_l$
 8.         Find an eigenvalue $\lambda$ of $\varphi_l$ in $\mathbb{F}_l$
 9.         $t = \lambda + q/\lambda \bmod l$
10.         $E = E \cup \{(t, l)\}$, $M_E = M_E \times l$
11.     else $l$ is an Atkin prime, then do:
12.         Determine the set $T_l$ of possible values for $t \bmod l$
13.         $A = A \cup \{(T_l, l)\}$, $M_A = M_A \times l$
14.     $l$ = nextprime($l$)
15. Use match and sort algorithm to determine $t$
16. Return $\#E_a(\mathbb{F}_q) = q + 1 - t$.
```

3. Optimizations in characteristic 2

3.1. **Computing modular polynomials $\overline{\Phi}_l(x, y)$.** The SEA algorithm computes information about the Frobenius trace modulo odd primes l based on the splitting of the l-th modular polynomial over $\mathbb{F}_q$. We know that $\overline{\Phi}_l(x, y)$ is the reduction of the l-th modular polynomial over $\mathbb{C}$ modulo the characteristic of the field. Thus, in this section we briefly recall the notion of modular polynomials over $\mathbb{C}$ and describe a very space- and time-efficient algorithm for computing these in characteristic two, based on an algorithm by Müller [4]. The proofs of the properties of these polynomials can be found in [30], [32] and [35].

It can be shown that an elliptic curve over $\mathbb{C}$ is isomorphic to $\mathbb{C}/L$, where $L = \mathbb{Z} + \mathbb{Z}\tau$ is a two-dimensional lattice in $\mathbb{C}$. The parameter τ can be chosen to lie in the upper half of the complex plane $\mathcal{H} = \{z \in \mathbb{C} \mid \Im(z) > 0\}$. In this way we can consider the j-invariant of the corresponding elliptic curve E_τ as a function on $\mathcal{H}$ and set $j(\tau) = j(E_\tau)$. Because the function $j(\tau)$ is invariant under $\mathrm{SL}_2(\mathbb{Z})$ transformations, it is sufficient to consider the restriction of $j(\tau)$ to the fundamental domain

$$\mathcal{F} = \{z \in \mathbb{C} \mid \Im(z) > 0, -1/2 \le \Re(z) < 1/2, |z| \ge 1\}. \tag{3.1}$$

Furthermore $j(\tau)$ is periodic of period one and has the Laurent series

$$j(\tau) = \frac{1}{q} + 744 + \sum_{n\ge 1} c_n q^n, \tag{3.2}$$

where $q = e^{2\pi i \tau}$ and c_n are positive integers. This Laurent series can be computed by using the relation between $j(\tau)$ and $\Delta(\tau)$, where $\Delta(\tau)$ is the discriminant of E_τ and has q-expansion

$$\Delta(\tau) = q\prod_{n\ge 1}(1-q^n)^{24} = q\left(1 + \sum_{n\ge 1}(-1)^n\left(q^{n(3n-1)/2} + q^{n(3n+1)/2}\right)\right)^{24}. \tag{3.3}$$

The relation between $j(\tau)$ and $\Delta(\tau)$ is given by the formula

$$j(\tau) = \frac{(256f(\tau)+1)^3}{f(\tau)}, \tag{3.4}$$

where $f(\tau) = \frac{\Delta(2\tau)}{\Delta(\tau)}$.

Define the l-th modular polynomial by the equation

$$\Phi_l(x,j) = (x - j(l\tau))\prod_{i=0}^{l-1}\left(x - j\left(\frac{\tau+i}{l}\right)\right), \tag{3.5}$$

then we know that $\Phi_l \in \mathbb{Z}[j][x]$ and, considered as a polynomial in two variables $\Phi_l(x,y) \in \mathbb{Z}[x,y]$, is symmetric and has degree $l+1$. Furthermore, $\Phi_l(x,y)$ has the following structure:

$$x^{l+1} - x^l y^l + y^{l+1} + R\ , \tag{3.6}$$

where R is of the form $\sum a_{ij}x^iy^j$, with $i,j \le l, i+j < 2l, a_{ij} \in \mathbb{Z}$. Reduced modulo two, this gives us the l-th modular polynomial $\overline{\Phi}_l(x,y)$ over $\mathbb{F}_q$ as in Theorem 2.1.

The computation of these polynomials is based on the reduced q-expansion of $j(\tau)$ and $j(l\tau)$ and the equality

$$\Phi_l(j(l\tau), j(\tau)) = 0. \tag{3.7}$$

Equation (3.4) reduced modulo two leads to

$$\bar{\jmath}(\tau) = \frac{\overline{\Delta}(\tau)}{\overline{\Delta}(2\tau)} = q^{-1}\frac{1 + \sum_{n\ge 1} q^{4n(3n-1)} + q^{4n(3n+1)}}{1 + \sum_{n\ge 1} q^{16n(3n-1)} + q^{16n(3n+1)}}. \tag{3.8}$$

The q-expansion of $\bar{\jmath}(l\tau)$ is obtained by substituting q^l for q in the q-expansion of $\bar{\jmath}(\tau)$. Now we can compute the l-th modular polynomial

$$\overline{\Phi}_l(x,y) = \sum_{i=0}^{l+1}\sum_{j=0}^{l+1} a_{ij}x^iy^i,\ a_{ij} \in \mathbb{F}_2 \tag{3.9}$$

by systematically comparing leading powers in the q-expansion of

$$\bar{\jmath}^{l+1}(\tau)+\bar{\jmath}^{l+1}(l\tau)=\sum_{i=0}^{l}\sum_{j=0}^{l}a_{ij}\bar{\jmath}^{i}(l\tau)\bar{\jmath}^{j}(\tau). \tag{3.10}$$

The leading power of $\bar{\jmath}^i(l\tau)\bar{\jmath}^j(\tau)$ equals $-(j+il)$ and because of the symmetry of $\overline{\Phi}_l(x,y)$, we have $a_{ij}=a_{ji}$. The set $S=\{(j+il,i+jl)\mid 0\le i\le j\le l\}$ can be given a strictly decreasing order by sorting on $\max(j+il,i+jl)$. Rewriting the right hand side of Equation (3.10) in this order, leads to the following algorithm:

Algorithm 3.1 (`Modular Polynomial`).

```
IN:   Prime $l$
OUT:  Modular polynomial $\overline{\Phi}_l(x,y)$

 1. Determine $\bar{\jmath}(\tau)$ with precision $l(l+1)+1$ using Equation (3.8)
 2. Compute $\bar{\jmath}^i(\tau)$ for $1<i\le l+1$
 3. Determine $\bar{\jmath}^j(l\tau)$ by substituting $q^l$ in $\bar{\jmath}^j(\tau)$ for $1\le j\le l+1$
 4. Set $L=\bar{\jmath}^{l+1}(\tau)+\bar{\jmath}^{l+1}(l\tau)$
 5. Set $A=0$, where $A[i,j]=a_{ij}$
 6. Set $A[l+1,0]=A[0,l+1]=1$
 7. While $(p(L)<0)$, where $p(L)$ is the leading power of $L$, do
 8.     Determine the pair $(i,j)$, with $p(L)=-\max(j+il,i+jl)$
 9.     Set $A[i,j]=A[j,i]=1$
10.     If $(i=j)$ then do $L=L+\bar{\jmath}^i(l\tau)\bar{\jmath}^j(\tau)$
11.     else set $L=L+\bar{\jmath}^i(l\tau)\bar{\jmath}^j(\tau)+\bar{\jmath}^j(l\tau)\bar{\jmath}^i(\tau)$
12. Return $\sum_{i=0}^{i=l+1}\sum_{j=0}^{j=l+1}A[i,j]x^iy^i$
```

The following tricks can be used to improve the time and space efficiency:

– Every power occurring in the reduced q-expansion of $\bar{\jmath}(\tau)$ given by Equation (3.8) is congruent to -1 modulo 8, as illustrated by the first few terms

$$\bar{\jmath}(\tau)=\frac{1}{q}+q^7+q^{15}+q^{31}+q^{47}+q^{55}+\cdots. \tag{3.11}$$

Therefore it is natural to represent every reduced power series S as

$$S=q^{-v_s}\sum_{n\ge 0}s_nq^{m_sn}, \tag{3.12}$$

with $v_s\in\mathbb{Z},m_s\in\mathbb{N}_0,s_n\in\mathbb{F}_2$ and $s_0=1$. Furthermore, to obtain the most space-efficient representation, m_s is taken as large as possible, i.e., m_s is the greatest common divisor of the powers occurring in the shifted power series $q^{v_s}S$. For instance, the reduced q-expansion of $\bar{\jmath}(\tau)$ can be written as $\bar{\jmath}(\tau)=q^{-1}\sum_{n\ge 0}\jmath_nq^{8n}$, with $\jmath_n\in\mathbb{F}_2$.

Now consider the arithmetic of these series:

- Addition: $C=A+B$, then $v_c\le\max(v_a,v_b)$ and $m_c=\gcd(v_a-v_b,m_a,m_b)$.
- Multiplication: $C=A\cdot B$, then $v_c=v_a+v_b$ and $m_c=\gcd(m_a,m_b)$.
- Squaring: $C=A^2$, then $v_c=2v_a$ and $m_c=2m_a$.

TABLE 1. Running time (s) on a Pentium III 600 MHz and memory usage (MB) for Algorithm 3.1

B	100	300	500	750	1000	2000
Precomp. $\overline{\jmath}(\tau)$ with $p = B(B+1)+1$	0	0.01	0.07	0.34	1.18	22.4
Precomp. $\overline{\jmath}^i(\tau)$ for $i = 2, \ldots, B$	0.02	1.99	21.9	161	772	26651
Compute $\overline{\Phi}_l(x, y)$ for all $l < B$	0.05	6.58	79.9	598	2649	108144
Total Time	0.07	8.60	101	759	3423	135442
Memory usage	0.61	0.98	1.28	4.08	8.75	62.7

- Inversion: $C = 1/A$, then $v_c = -v_a$ and $m_c = m_a$.

Thus a power series S can be represented as a triple (v_s, m_s, C_s), where C_s is a bit-array which contains the coefficients s_n. When working with precision p, the length of this array equals $\lceil p/m_s \rceil$.

– With the above representation, squaring a power series A actually is free and so is computing every 2^n-th power $B = A^{2^n}$, because we only need to set $v_b = 2^n v_a$, $m_b = 2^n m_a$ and take the bit-array C_b equal to the first $\lceil p/m_b \rceil$ elements in C_a. So, instead of computing $\overline{\jmath}^i(\tau)$ for $1 < i \leq l+1$ in step 2, it suffices to compute only the odd powers of $\overline{\jmath}$.

– In step 8 we need the unique pair (i, j), with $p(L) = -\max(j + il, i + jl)$. One easily verifies that $j = \lfloor -p(L)/l \rfloor$ and $i = -p(L) - jl$ satisfy these conditions.

In Table 1 we give running times and memory usage for the computation of $\overline{\Phi}_l(x, y)$ for *all* primes $l \leq B$, obtained on a Pentium III 600 MHz. Because for every l we need the q-series of $\overline{\jmath}^i(\tau)$ for $1 \leq i \leq l+1$ with precision $l(l+1)+1$, we first precompute these powers with precision $B(B+1)+1$. Note that if we had used a list containing only the powers of the non-zero terms, i.e., the conventional sparse series representation, the memory usage for computing $\overline{\Phi}_l(x, y)$ for all $l < 2000$ would exceed 2 GB. Furthermore, the running times would be a factor 100 slower.

3.2. **Determining the splitting of $\overline{\Phi}_l(x, j_a)$.** Theorem 2.2 classifies l as an Elkies prime iff $\overline{\Phi}_l(x, j_a)$ has a root in $\mathbb{F}_q$. Because every element of $\mathbb{F}_q$ is a zero of $x^q - x$, l is an Elkies prime iff the degree of

$$\gcd\left(x^q - x, \overline{\Phi}_l(x, j_a)\right) \tag{3.13}$$

equals 1, 2 or $l+1$. This gcd computation needs $x^q \bmod \overline{\Phi}_l(x, j_a)$, which is one of the dominant steps in the `SEA` algorithm, so it should be implemented as efficiently as possible.

The best known algorithm is one by Kaltofen and Shoup [14] and uses a combination of modular composition and squaring. Define $\beta_j \equiv x^{2^j} \equiv \sum_{i=0}^{d-1} c_i x^i \bmod f(x)$, with $d = \deg(f)$, then we need $x^q \equiv \beta_n \bmod f(x)$. Note that $\beta_{2j} \equiv \beta_j^{2^j} \bmod f(x)$ and

$$\begin{aligned} \beta_j^{2^j} &\equiv \left(\sum_{i=0}^{d-1} c_i x^i\right)^{2^j} \\ &\equiv \sum_{i=0}^{d-1} c_i^{2^j} \beta_j^i \mod f(x), \end{aligned} \tag{3.14}$$

so we can compute β_{2j} using β_j with d modular compositions over $\mathbb{F}_2$ and one modular composition over $\mathbb{F}_q$. Starting with $j = \lfloor n/2 \rfloor$, the algorithm recursively computes β_j and if $n = 2j$ it finishes with $\beta_n \equiv \beta_{2j} \bmod f(x)$, else we have $n = 2j+1$ and $\beta_n \equiv \beta_{2j}^2 \bmod f(x)$.

The algorithm by Brent and Kung [3] for polynomial modular composition has complexity $O(d^{1.69})$ and the product of two $\mathbb{F}_q$ elements can be computed in $O(n \log n \log\log n)$ bit operations. This leads to a total cost of

$$O(d^{1.69} n \log^2 n \log\log n + dn^{1.69} \log n) \tag{3.15}$$

bit operations for computing $x^q \bmod f(x)$. Note that this algorithm is faster than repeated modular squaring provided d is small compared to n. The data in Section 4 show that this is always the case in the SEA algorithm because the maximal prime $l_{\max}$ used is about $n/2$.

For an Atkin prime l we need the equal degree r of the irreducible factors of $\overline{\Phi}_l(x, j_a)$. Obviously $r | l+1$, because $\overline{\Phi}_l(x, j_a)$ has degree $l+1$, but a theorem by Schoof [32] states that the number of irreducible factors $s = (l+1)/r$ satisfies

$$(-1)^s = \left(\frac{q}{l}\right). \tag{3.16}$$

This theorem limits the possibilities for r to the set $R = \{\rho \in \mathbb{N} \mid \left(\frac{q}{l}\right) = (-1)^{\frac{l+1}{\rho}}\}$.

For nonnegative integers a and b, the polynomial $x^{q^a} - x^{q^b}$ is divisible precisely by the irreducible polynomials in $\mathbb{F}_q[x]$ whose degree divides $a-b$. This fact follows, for example, from [20, Theorem 3.20]. Let B be the largest integer in R strictly smaller than $\max(R)$, $k = \lfloor\sqrt{B}\rfloor$ and $m = \lceil B/k \rceil$, then it is sufficient to test for all integers $i = k, \ldots, 1$ and $j = 1, \ldots, m$ if

$$x^{q^{jk}} \equiv x^{q^i} \bmod \overline{\Phi}_l(x, j_a). \tag{3.17}$$

When a match occurs, we set $r = jk - i$ else $r = \max(R)$. The algorithm needs k modular compositions $x^{q^i} \circ x^q$ and at most m modular compositions $x^{q^{jk}} \circ x^{q^k}$. Using the algorithm of Brent and Kung this leads to a total complexity of $O(l^{2.19})$ $\mathbb{F}_q$ operations, because both k and m are $O(\sqrt{l})$ and $\deg(\overline{\Phi}_l(x, j_a)) = l+1$.

3.3. **Computing isogenies over $\mathbb{F}_{2^n}$.**

Currently three algorithms exist to compute an isogenie $\mathcal{I}$ between two elliptic curves E_a and E_b over $\mathbb{F}_{2^n}$. A first algorithm was proposed by Couveignes [6] and works in the formal group defined by E_a. Although it has complexity $O(l^3)$, it turns out to be quite slow in practice, because of the huge series computations involved.

The second algorithm was proposed by Lercier [18] and is based on the fact that $\mathcal{I} \circ [2]_a = [2]_b \circ \mathcal{I}$. It only works in characteristic two and performs much better in practice than Couveignes' algorithm. By using algebraic properties it determines a set of non-linear equations in binary variables, which eventually leads to the construction of the isogenie $\mathcal{I}$. We combined this algorithm with some heuristic improvements which become important for large primes l. Accurate statistics are given in Section 4 and show that our implementation reduces the time for isogenie computations to a negligible 1.5% of the total running time of the SEA algorithm.

A third algorithm also by Couveignes [7] and valid over any field characteristic p, consists in computing $E_a[p^k]$ and $E_b[p^k]$ and is based on the fact that $\mathcal{I}(E_a[p^k]) = E_b[p^k]$. Using fast multiplication techniques, the complexity drops to $O(l^2)$, but since all computations take place in an extension of degree $p^{k-1}(p-1)/2$ it is not obvious to implement this algorithm efficiently in practice.

3.4. **Looking for an eigenvalue.** If l is an Elkies prime, there exists at least one cyclic subgroup $C_i \subset E_a[l]$ of order l which is stable under the Frobenius-morphism, i.e., $\varphi_l(P) = \lambda P$ for all $P \in C$ and $1 \leq \lambda \leq l-1$. Using Theorem 2.1 we can compute an isogenie $\mathcal{I}_l \,:\, E_a \longrightarrow E'_a$ with $\ker(\mathcal{I}_l) = C_i$. The construction of this isogenie results in a polynomial g_l that vanishes on all the non-zero x-coordinates of points in C_i. Hence, to find this eigenvalue we simply test for which integer $1 \leq \lambda \leq l-1$ the equation $(x^q, y^q) = [\lambda](x,y)$ holds in the ring $\mathbb{F}_q[x,y]/(g_l(x), y^2+xy-x^3-a)$.

Müller [21] proposed a multiplicative version of the baby-step giant-step algorithm, by writing λ as a fraction of two smaller integers modulo l. It turns out that every element of the set $\{0,\dots,l-1\}$ can be written as

$$\pm\frac{i}{j} \bmod l \qquad i,j \in [0,B], \tag{3.18}$$

where $B = \lfloor\sqrt{l}\rfloor$. Thus the search for λ is replaced by the search for the pair (i,j) and the correct sign s such that

$$[j](x^q,y^q) = s[i](x,y) \bmod (g_l(x), y^2+xy-x^3-a). \tag{3.19}$$

To avoid the costly computation of y^q in the ring $\mathbb{F}_q[x,y]/(g_l(x), y^2+xy-x^3-a)$, we only compute the x-coordinates of both sides of Equation (3.19) using the following theorem by Koblitz [16].

Theorem 3.1. *Let f_n denote the n-th division polynomial, $P=(x,y) \in E \setminus E[n]$, then for $n \geq 2$*

$$nP = \left(x + \frac{f_{n-1}f_{n+1}}{f_n^2}, x + y + \frac{f_{n-1}f_{n+1}}{f_n^2} + \frac{f_{n-2}f_{n+1}^2}{xf_n^3} + (x^2+y)\frac{f_{n-1}f_{n+1}}{xf_n^2}\right). \tag{3.20}$$

Using only the x-coordinates has the disadvantage that we no longer can distinguish between a point P and its negative $-P$, because both have the same x-coordinate. Thus we can only determine the eigenvalue up to the sign.

In each step of the algorithm we have to test for a given integer $j \leq B$ whether

$$\frac{A_j(x^q)}{B_j(x^q)} \equiv \frac{A_i(x)}{B_i(x)} \bmod g_l(x) \tag{3.21}$$

for $1 \leq i \leq B$. The obvious way is to multiply out denominators and check for equality of $A_j(x^q)B_i(x) - A_i(x)B_j(x^q) \equiv 0 \bmod g_l(x)$, which requires $2B$ modular multiplications for every $1 \leq j \leq B$.

To speed up this process, we use a variant of a probabilistic method by Shoup [34]. Suppose $\deg(g_l) = d$ and that all polynomials A_j, B_j, A_i, B_i are reduced modulo g_l. Then we define the linear mapping

$$M \;:\; \mathbb{F}_q[x]/(g_l(x)) \longrightarrow \mathbb{F}_q \;:\; f(x) \mapsto f(0), \tag{3.22}$$

so M maps every polynomial to its constant term. Obviously, the map M is linear and $\#\ker(M) = q^{d-1}$. This means that a non-zero polynomial $f(x) \not\equiv 0 \bmod g_l(x)$ will have a non-zero image under M with probability about $1-1/q$. Because in our case q is at least 2^{150}, this probability is very high. Thus the map M can be used to check Equation (3.21) with a very high probability by testing whether

$$M\left(A_j(x^q)B_i(x) \bmod g_l(x)\right) = M\left(A_i(x)B_j(x^q) \bmod g_l(x)\right). \tag{3.23}$$

Note that in every iteration $A_j(x^q)$ and $B_j(x^q)$ are kept constant, whereas $A_i(x)$ and $B_i(x)$ vary, so we need an efficient way to compute $M(A(x)B(x) \bmod g_l(x))$, where $A(x)$ is kept constant. Let $\overline{A}$ and $\overline{B}$ be the coefficient vectors of A and B, then using the linearity of M we get

$$M(A(x)B(x) \bmod g_l(x)) = \sum_{i=0}^{d-1} \overline{B}[i] M\left(A(x)x^i \bmod g_l(x)\right). \tag{3.24}$$

Define the vector $\overline{M}_A$ by $\overline{M}_A[i] = M\left(A(x)x^i \bmod g_l(x)\right)$ for $0 \le i < d$. Then we can compute $M(A(x)B(x) \bmod g_l(x))$ as the inner product of the vectors $\overline{B}$ and $\overline{M}_A$. Algorithm 3.2 gives an efficient way to compute $\overline{M}_A$, which can be verified by an easy calculation. The complexity of this algorithm is $O(d^2)$ $\mathbb{F}_q$-operations, but it is at least a factor four faster than using the original random linear mapping proposed by Maurer and Müller [21].

Algorithm 3.2 (Compute $\overline{M}_A$).

IN: *Polynomial $g_l(x)$ of degree d and polynomial $A(x)$ with $\deg(A) < d$*
OUT: *Vector $\overline{M}_A[i] = M\left(A(x)x^i \bmod g_l(x)\right)$ for $0 \le i < d$*

```
1. Compute $\overline{X}_d$, the coefficient vector of $x^d \bmod g_l(x)$
2. Set $\overline{M}_A[0] = \overline{A}[0]$
3. For $i = 1, \ldots, d-1$ do:
4.     Set $\overline{M}_A[i] = \overline{A}[d-i] \times \overline{X}_d[0]$
5.     For $j = 1, \ldots, i-1$ do:
6.         $\overline{M}_A[i] += \overline{X}_d[d-i+j] \times \overline{M}_A[j]$
7. Return $\overline{M}_A$
```

The above strategy is very efficient, but only determines the eigenvalue up to the sign. In most cases we can resolve this non-uniqueness by a theorem of Dewaghe [8].

Theorem 3.2. *Let E_a be an elliptic curve over $\mathbb{F}_q$ given by the equation $y^2 + xy = x^3 + a$, l an odd prime and $g_l = \sum_{i=0}^{d} g_i x^i$ a factor of degree $d = (l-1)/2$ of f_l corresponding to an eigenvalue λ and a cyclic subgroup C of $E[l]$. Suppose $[j]\varphi(P) = \pm[i]P$ for $P \in C$, then $\lambda \equiv \pm\lambda_0 \bmod l$, with $\lambda_0 \equiv i/j \bmod l$. Let s be the semi-order of λ_0 in $\mathbb{F}_l$, i.e. the smallest $n \in \mathbb{N}_0$ such that $\lambda_0^n \equiv \pm 1 \mod l$ and $\overline{g}_l = \sum_{i=0}^{r} \overline{g}_i x^i$ be a factor of g_l of degree s. If s is odd, then*

$$\lambda = \begin{cases} \lambda_0^s \lambda_0 & \text{if } \operatorname{Tr}\left(\overline{g}_{s-1} + a\frac{\overline{g}_1^2}{\overline{g}_0^2}\right) = 0 \\ -\lambda_0^s \lambda_0 & \text{otherwise.} \end{cases} \tag{3.25}$$

If $l \equiv 3 \bmod 4$ we can take $\overline{g}_l = g_l$, so the above theorem provides a very efficient way to determine the correct sign of the eigenvalue. Otherwise $l \equiv 1 \bmod 4$ and if s is odd, we could find the sign by computing a factor $\overline{g}_l$ of degree s. If s is even, we still have to compare the y-coordinates of $[j]\varphi(P)$ and $[i]P$, so we need $y^q \bmod (g_l(x), y^2 + xy - x^3 - a)$. Usually this is done by iterated squaring and substitution of $y^2 \equiv (x^3 + a) + xy \bmod (g_l(x), y^2 + xy - x^3 - a)$. However, an easy calculation shows that

$$y^q \equiv x^q \sum_{i=0}^{n-1} \left(x + ax^{-2}\right)^{2^i} + x^{q-1}y \bmod (g_l(x), y^2 + xy - x^3 - a), \tag{3.26}$$

which can be computed by evaluating a trace-like map of $x + ax^{-2} \bmod g_l(x)$, using an efficient algorithm by Kaltofen and Shoup [14]. Not only is this algorithm more efficient than iterated squaring, it also generates $x^q \bmod g_l(x)$, which we need anyway in the search for the eigenvalue. Thus, whenever $l \equiv 1 \bmod 4$ we compute $x^q \bmod g_l(x)$ and $y^q \bmod g_l(x)$ at once, find $\pm\lambda$ and compare the y-coordinates of $[j]\varphi(P)$ and $[i]P$ to determine the correct sign. This strategy is even faster than factoring in the case where s is odd. Finally, note that $x^{-1} \bmod g_l(x)$ exists because l is an odd prime.

Algorithm 3.3 (`Find Eigenvalue`).

`IN:` *Odd prime l and polynomial $g_l(x) = \prod_{\pm P \in E[l]^*_\lambda}(x - x(P))$*
`OUT:` *Eigenvalue λ*

```
 1. Set $B = \lfloor\sqrt{l}\rfloor$
 2. For $i = 1, \ldots, B$ do:
 3.     Compute and store the $x$-coordinate of
        $i(x,y) \bmod g_l(x)$ as $\frac{A_i(x)}{B_i(x)} \bmod g_l(x)$
 4. For $j = 1, \ldots, B$ do:
 5.     Compute the $x$-coordinate of $j(x^q, y^q)$ as $\frac{A_j(x^q)}{B_j(x^q)} \bmod g_l(x)$
 6.     Compute $\overline{M}_{A_j}$ and $\overline{M}_{B_j}$
 7.     For $i = 1, \ldots, B$ do:
 8.         If $\overline{M}_{A_j} \otimes \overline{B}_i \equiv \overline{M}_{B_j} \otimes \overline{A}_i$ then
 9.             If $A_j(x^q)B_i(x) - A_i(x)B_j(x^q) \equiv 0 \bmod g_l(x)$ then
10.                 set $\lambda_0 \equiv ij^{-1} \bmod l$ and break
11. Compute sign of $\lambda$ using Theorem 3.2 or Equation (3.26)
12. Return $\lambda$
```

4. Implementation and results

The `SEA` algorithm with the above optimizations was implemented as part of `C++PEC` (Counting Points on Elliptic Curves), a package which was developed during the author's master thesis [37]. The program is written in standard `C++`, using Victor Shoup's `NTL` [33] for the finite field arithmetic.

4.1. **Elliptic curves and cryptography.** The main use of the `SEA` algorithm is the generation of cryptographically strong elliptic curves. In this section we provide timings for computing the number of points on elliptic curves defined over $\mathbb{F}_{2^n} \cong \mathbb{F}_2[t]/(f(t))$ in the cryptographic range, i.e. $163 \leq n \leq 431$. The extension degrees of the finite fields used are all prime, because of the recent attack by Gaudry, Hess and Smart [12] and the irreducible polynomial $f(t)$ is either a trinomial or a pentanomial. All timings were obtained on a Pentium III 600 MHz, with 128 MB RAM, running Linux as operating system.

TABLE 2. Running times (s) for finite fields $\mathbb{F}_{2^n}$, $163 \le n \le 239$.

$\mathbb{F}_{2^{163}}$	min	avg	max	$\mathbb{F}_{2^{191}}$	min	avg	max	$\mathbb{F}_{2^{239}}$	min	avg	max
$l_{\max}$	47	55	67	$l_{\max}$	53	64	73	$l_{\max}$	71	80	97
$\#E$	6	9	12	$\#E$	6	10	15	$\#E$	8	12	16
$\#A$	3	7	12	$\#A$	2	7	14	$\#A$	5	9	15
$\#C$	1.98	6.70	8.10	$\#C$	2.15	7.54	8.73	$\#C$	2.40	8.88	9.71
Frob	3.95	5.69	7.88	Frob	6.11	8.61	11.3	Frob	15.6	22.1	30.3
Ord	0.2	1.27	2.78	Ord	0.28	1.84	3.32	Ord	0.31	4.13	10.9
Iso	0.04	0.11	0.23	Iso	0.03	0.17	0.3	Iso	0.14	0.41	0.75
Eig	0.35	1.02	2.53	Eig	0.46	1.85	6.17	Eig	0.82	3.74	10.9
Sign	0.06	0.35	0.92	Sign	0.02	0.52	0.99	Sign	0.23	1.15	2.71
M-S	0.23	0.53	2.85	M-S	0.08	1.08	5.31	M-S	0.46	4.98	20.8
Total	6.97	9.45	14.1	Total	10.9	14.7	19.8	Total	25.2	37.7	56.9

As a first test, we determined the group order of 100 random elliptic curves $y^2 + xy = x^3 + a$ over the finite fields $\mathbb{F}_{2^{163}} \cong \mathbb{F}_2[t]/(t^{163} + t^7 + t^6 + t^3 + 1)$, $\mathbb{F}_{2^{191}} \cong \mathbb{F}_2[t]/(t^{191} + t^9 + 1)$ and $\mathbb{F}_{2^{239}} \cong \mathbb{F}_2[t]/(t^{239} + t^{36} + 1)$. Table 2 indicates the maximum prime used $l_{\max}$; the number of Elkies ($\#E$) and Atkin ($\#A$) primes; the 10-log of the number of possibilities ($\#C$) in the match and sort phase; the cumulated times Frob for computing $x^q \bmod f(x)$, Ord for determining the order r of φ_l in $\mathrm{PGL}_2(\mathbb{F}_l)$, Iso for computing isogenies, Eig for searching the eigenvalue in case of an Elkies prime and Sign for computing the sign of the eigenvalue; the time M-S for the match and sort phase and finally the total time. For each of these timings, we give minimal, average and maximal values.

Table 2 indicates that in this range the running time of our implementation grows less than $\log^4 q$. The reason for this low practical complexity is twofold: first, `NTL` uses fast multiplication techniques and second, in this range the search space in the match and sort algorithm is large enough to include every Atkin prime. For larger fields the available search space saturates and therefore the information found from some of the Atkin primes will be redundant.

A second test consisted in computing the number of points on 50 random elliptic curves defined over the larger fields $\mathbb{F}_{2^{367}} \cong \mathbb{F}_2[t]/(t^{367} + t^{21} + 1)$, $\mathbb{F}_{2^{401}} \cong \mathbb{F}_2[t]/(t^{401} + t^{152} + 1)$ and $\mathbb{F}_{2^{431}} \cong \mathbb{F}_2[t]/(t^{431} + t^{120} + 1)$. For these larger fields the run time grows less than $O(\log^5 q)$ as indicated by Table 3. Furthermore Table 2 and 3 suggest that, although the practical complexity differs in both ranges, the percentage of time taken up by the different sub-algorithms stays constant.

The main constraint for an elliptic curve E_a over $\mathbb{F}_q$ to be secure is that the group order $\#E_a(\mathbb{F}_q)$ should be divisible by a large prime. Every elliptic curve of the form $y^2 + xy = x^3 + a$ has a point $P_a = (\sqrt[4]{a}, \sqrt{a})$ of order 4 so the cryptographically strongest curves are those with $\#E_a(\mathbb{F}_q) = 4p$, where p is prime. Because for every Elkies prime l we compute $t \bmod l$, we can check if $\#E_a(\mathbb{F}_q) = q + 1 - t$ is divisible by l. In this case we conclude that the curve E_a will not be suitable for cryptographic purposes and continue with the next curve. This technique is called the early abort strategy [19] and works well because if $\#E_a(\mathbb{F}_q)$ is not nearly prime, then most times it is divisible by a small prime.

TABLE 3. Running times (s) for finite fields $\mathbb{F}_{2^n}$, $367 \leq n \leq 431$.

$\mathbb{F}_{2^{367}}$	min	avg	max	$\mathbb{F}_{2^{401}}$	min	avg	max	$\mathbb{F}_{2^{431}}$	min	avg	max
$l_{\max}$	109	147	191	$l_{\max}$	113	164	197	$l_{\max}$	137	180	233
$\#E$	15	19	22	$\#E$	18	21	24	$\#E$	19	22	26
$\#A$	7	15	26	$\#A$	9	17	25	$\#A$	9	19	30
$\#C$	7.25	10.2	10.8	$\#C$	8.62	10.4	11.2	$\#C$	7.72	10.4	10.9
Frob	153	268	518	Frob	197	424	710	Frob	266	597	1115
Ord	16.4	82.7	215	Ord	40.0	127	362	Ord	20.8	174	443
Iso	2.6	4.72	10.7	Iso	2.55	7.64	15.0	Iso	4.56	12.0	25.1
Eig	10.4	24.1	49.5	Eig	11.6	35.6	73.0	Eig	25.9	49.2	97.6
Sign	1.08	16.7	37.3	Sign	11.2	28.0	58.8	Sign	15.4	38.8	79.2
M-S	4.11	52.8	148	M-S	11.7	68.7	174	M-S	6.21	80.4	221
Total	242	457	886	Total	350	704	1179	Total	447	967	1790

TABLE 4. Statistics for the early abort strategy on a Pentium III 600 MHz.

Number of curves with	$\mathbb{F}_{2^{163}}$	$\mathbb{F}_{2^{191}}$	$\mathbb{F}_{2^{239}}$	$\mathbb{F}_{2^{307}}$	$\mathbb{F}_{2^{367}}$	$\mathbb{F}_{2^{401}}$	$\mathbb{F}_{2^{431}}$
$8 \mid \#E_a(\mathbb{F}_q)$	506	502	498	518	480	505	496
$3 \mid \#E_a(\mathbb{F}_q)$	265	233	237	249	372	237	262
$5 \mid \#E_a(\mathbb{F}_q)$	55	69	60	56	64	60	64
$7 \mid \#E_a(\mathbb{F}_q)$	24	31	25	28	37	37	34
$11 \mid \#E_a(\mathbb{F}_q)$	13	24	14	20	15	19	7
$13 \mid \#E_a(\mathbb{F}_q)$	12	16	17	9	15	12	9
$l \mid \#E_a(\mathbb{F}_q), 13 < l \leq l_{\max}$	34	50	63	44	48	59	61
$l \mid \#E_a(\mathbb{F}_q), l_{\max} < l$	91	75	86	76	69	71	67
$\#E_a(\mathbb{F}_q) = 4p$	7	3	6	4	1	0	6
Total time (h):	0.32	0.51	1.27	4.47	14.7	22.3	30.3

In the third test, we used this strategy to search for elliptic curves with nearly prime cardinality over the fields defined above. For each field we tested 1000 random curves and determined the smallest Elkies prime l which divides the group order. When the cardinality of a curve E_a was computed completely, we checked if $\#E_a(\mathbb{F}_q)$ factors as $4p$, with p prime.

Table 4 shows that about 80% of the curves are detected as being bad curves within one second and that only 0.5% of the curves have an optimal cardinality. Furthermore, whilst searching for cryptographically strong elliptic curves, the use of the early abort strategy results in a speed-up of a factor 8 over the naive method.

4.2. **A 1999-bit elliptic curve.** To illustrate the efficiency of our implementation and to study the behavior of the algorithm for huge finite fields, we counted the number of points on a 1999-bit elliptic curve. More precisely, let E be the elliptic curve over $\mathbb{F}_{2^{1999}} \cong \mathbb{F}_2[t]/(t^{1999} + t^{367} + 1)$ defined by

$$y^2 + xy = x^3 + t^{19} + t^{18} + t^{17} + t^{16} + t^{14} + t^{13} + t^{12} + t^8 + t^5, \tag{4.1}$$

Elkies Primes

$x^q \bmod \overline{\Phi}(x, j_a)$	11.88
Isogenie $\mathcal{I}$	5.743
Check isogenie	1.069
$x^q \bmod \mathcal{I}$	3.396
Eigenvalue	3.540
Sign	4.513
Total	30.36

Atkin Primes

$x^q \bmod \overline{\Phi}(x, j_a)$	15.46
Equal degree r	17.60
Total	33.22

Table 5. Run time (days) for a 1999-bit elliptic curve on a Pentium II 400 MHz.

then $\#E(\mathbb{F}_{2^{1999}}) = 2^{1999} + 1 - c$ with

$$\begin{aligned}c =\,&151405908230621180774107306567712619334174721345788259326840773080788\\&772556797641330166425115485164266441847512520006272260565480608080 9\\&310382327170008885866833835871649030149719136567732130780176973616748\\&785759669139650023212173840976394766149549468848703178405560624987 3\\&4022252177154721915844 0743937.\end{aligned}$$

The computation was distributed using a simple bash script over a network of 10 Pentium II 400 MHz all running Linux, where PC i determined $c \bmod l_{10\cdot k+i}$, with l_s the s-th prime and $s \leq 200$. The total run time on one Pentium II would have been about 65 days, of which most of the time was spent on computing the splitting of the modular polynomials as shown in Table 5.

We computed information about $c \bmod l$ for primes up to 1223, of which 96 were Elkies primes and 104 Atkin primes. To merge the information found from both types of primes, a variant of the Chinese & Match algorithm [13] was used to search among the 2.6 10^{13} remaining possibilities for c. The computation took 2.16 days to complete on one Pentium II 400 MHz and used 29 MB of main memory. This algorithm retains the candidates for c which satisfy all the congruences found from the Atkin and Elkies primes. To speed up the algorithm we implemented a phased variant based on a probabilistic argument. In the first phase, we only used 25 Atkin primes to limit the possibilities for c, then in a second phase, we further refined these possibilities by using 35 Atkin primes. Finally, we checked the remaining candidates against the other Atkin primes, which left us with 319 possibilities for c. A simple exhaustive search then completed the computation with the result given above. This result was further verified by computing $c \bmod l$ for larger primes l until enough Elkies primes were found to recover c uniquely from the Chinese Remainder Theorem.

5. Conclusion

In this article we have described various optimizations of the Schoof-Elkies-Atkin algorithm specific for the characteristic two case. We presented a very space- and time-efficient algorithm for computing modular polynomials in characteristic two and for searching the eigenvalue in the Elkies case.

With our implementation we were able to compute the number of points on an elliptic curve defined over $\mathbb{F}_{2^{1999}}$, which is the largest point-counting computation to date. Furthermore, counting the number of points on an elliptic curve over $\mathbb{F}_{2^{163}}$

takes less than 10 seconds on a Pentium III 600 MHz and with the early abort strategy we can test 1000 such curves in about 20 minutes. This indicates that the cost of generating cryptographically secure elliptic curves no longer is prohibitive in implementing elliptic curve cryptosystems.

Acknowledgments

The author would like to thank Victor Shoup for his library `NTL`, Reynald Lercier for the many fruitful discussions during the implementation of the isogenie computation routine and Volker Müller for providing an algorithm to compute the modular polynomials in characteristic two.

References

[1] A.O. Atkin, *The number of points on an elliptic curve modulo a prime*, Draft, 1998.

[2] A. Bender and G. Castagnoli, *On the implementation of elliptic curve cryptosystems*, Advances in Cryptology – CRYPTO '89, Lect. Notes Comput. Sci., vol. 435, Springer-Verlag, 1990, pp. 186–192.

[3] R.P. Brent and H.T. Kung, *Fast algorithms for manipulating formal power series*, J. Assoc. Comput. Mach. **25** (1978), 581–595.

[4] R. Carls, *Punktezählalgorithmen für ordinäre elliptische Kurven über endlichen Körpern der Charakteristik 2*, Master's thesis, Technischen Universität Darmstadt, September 1999.

[5] J. Chao, K. Tanada, and S. Tsujii, *Design of elliptic curves with controllable lower boundary of extension degree for reduction attacks*, Advances in Cryptology – CRYPTO '94, Lect. Notes Comput. Sci., vol. 839, Springer-Verlag, 1994, pp. 50–55.

[6] J.M. Couveignes, *Quelques calculs en théorie des nombres*, Ph.D. thesis, Université de Bordeaux I, July 1994.

[7] ———, *Computing l-isogenies using the p-torsion*, ANTS-II: Algorithmic Number Theory, Lect. Notes Comput. Sci., vol. 1122, Springer-Verlag, 1996, pp. 59–65.

[8] L. Dewaghe, *Remarks on the Schoof-Elkies-Atkin algorithm*, Math. Comput. **67** (1998), no. 223, 1247–1252.

[9] N.D. Elkies, *Explicit isogenies*, Draft, 1991.

[10] ———, *Elliptic and modular curves over finite fields and related computational issues*, Computational perspectives on number theory, Stud. Adv. Math., vol. 7, AMS/IP, 1998, pp. 21–76.

[11] G. Frey and H.-G. Rück, *A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves*, Math. Comput. **62** (1994), no. 206, 865–874.

[12] P. Gaudry, F. Hess, and N.P. Smart, *Constructive and destructive facets of Weil descent on elliptic curves*, Submitted to J. Cryptology, 2000.

[13] A. Joux and R. Lercier, *Chinese & Match, an alternative to Atkin's Match and Sort method used in the SEA algorithm*, Preprint, 1999.

[14] E. Kaltofen and V. Shoup, *Fast polynomial factorization over high algebraic extensions of finite fields*, Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, ACM Press, 1997, pp. 184–188.

[15] N. Koblitz, *Elliptic curve cryptosystems*, Math. Comp. **48** (1987), 203–209.

[16] ———, *Constructing elliptic curve cryptosystems in characteristic 2*, Advances in Cryptology – CRYPTO '90, Lect. Notes Comput. Sci., vol. 537, Springer-Verlag, 1991, pp. 156–167.

[17] G.J. Lay and H.G. Zimmer, *Constructing elliptic curves with given group order over large finite fields*, ANTS-I: Algorithmic number theory, Lect. Notes Comput. Sci., vol. 877, 1994, pp. 250–263.

[18] R. Lercier, *Computing isogenies in $\mathbb{F}_{2^n}$*, ANTS-II: Algorithmic Number Theory, Lect. Notes Comput. Sci., vol. 1122, Springer-Verlag, 1996, pp. 197–212.

[19] ———, *Algorithmique des courbes elliptiques dans les corps finis*, Ph.D. thesis, L'École Polytechnique, Laboratoire D'Informatique, CNRS, Paris, June 1997.

[20] R. Lidl and H. Niederreiter, *Finite fields*, Encyclopedia of Mathematics and Its Applications, vol. 20, Cambridge Univ. Press, 1996.

[21] M. Maurer and V. Müller, *Finding the eigenvalue in Elkies' algorithm*, Preprint, 1999.

[22] A.J. Menezes, T. Okamoto, and S.A. Vanstone, *Reducing elliptic curve logarithms to a finite field*, IEEE Trans. Inf. Theory **39** (1993), no. 5, 1639–1646.

[23] A.J. Menezes and S.A. Vanstone, *The implementation of elliptic curve cryptosystems*, Advances in Cryptology – AUSCRYPT '90, Lect. Notes Comput. Sci., vol. 453, Springer-Verlag, 1990, pp. 2–13.

[24] V.S. Miller, *Use of elliptic curves in cryptography*, Advances in Cryptology – CRYPTO '85, Lect. Notes Comput. Sci., vol. 218, Springer-Verlag, 1986, pp. 417–426.

[25] A. Miyaji, *Elliptic curves over $\mathbb{F}_p$ suitable for cryptosystems*, Advances in Cryptology – AUSCRYPT '92, Lect. Notes Comput. Sci., vol. 718, Springer-Verlag, 1993, pp. 479–491.

[26] F. Morain, *Calcul du nombre de points sur une courbe elliptique dans un corps fini: aspects algorithmiques*, J. Theor. Nombres Bordx. **7** (1995), no. 1, 255–282.

[27] V. Müller, *Ein Algorithmus zur Bestimmung der Punktzahl elliptischer Kurven über endlichen Körpern der Charakteristik grösser drei*, Ph.D. thesis, Universität des Saarlandes, 1995.

[28] R.L. Rivest, A. Shamir, and L.M. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Comm. ACM **21** (1978), 120–126.

[29] T. Satoh and K. Araki, *Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves*, Comment. Math. Univ. St. Pauli **47** (1998), no. 1, 81–92.

[30] B. Schoeneberg, *Elliptic modular functions. An introduction*, Die Grundlehren der mathematischen Wissenschaften, vol. 203, Springer-Verlag, 1974.

[31] R. Schoof, *Elliptic curves over finite fields and the computation of square roots* mod p, Math. Comput. (1985), no. 44, 483–494.

[32] ———, *Counting points on elliptic curves over finite fields*, J. Theor. Nombres Bordx. **7** (1995), no. 1, 219–254.

[33] V. Shoup, *NTL: A Library for doing Number Theory*, Available at `http://www.shoup.net/ntl/`.

[34] ———, *A new polynomial factorization algorithm and its implementation*, J. Symb. Comput. **20** (1995), no. 4, 363–397.

[35] J.H. Silverman, *The arithmetic of elliptic curves*, Graduate Texts in Mathematics, vol. 106, Springer-Verlag, 1986.

[36] N.P. Smart, *The discrete logarithm problem on elliptic curves of trace one*, J. Cryptology **12** (1999), no. 3, 193–196.

[37] F. Vercauteren, *The number of points on elliptic curves over finite fields of characteristic 2*, Master's thesis, Katholieke Universiteit Leuven, June 1999, (in Dutch).

Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven, Kardinaal Mercierlaan 94, B-3001 Heverlee, Belgium

E-mail address: `Frederik.Vercauteren@esat.kuleuven.ac.be`