# Efficient Construction of Cryptographically Strong Elliptic Curves

Harald Baier and Johannes Buchmann
Darmstadt University of Technology
Department of Computer Science
Alexanderstr. 10
64283 Darmstadt
{hbaier,buchmann}@cdc.informatik.tu-darmstadt.de

March 1, 2001

**Abstract**

We present a very efficient algorithm which given a negative integer $\Delta$, $\Delta \equiv 1 \bmod 8$, $\Delta$ not divisible by 3, finds a prime number $p$ and a cryptographically strong elliptic curve $E$ over the prime field $\mathbb{F}_p$ whose endomorphism ring is the quadratic order $\mathcal{O}$ of discriminant $\Delta$. If the class number of $\mathcal{O}$ is 200, then the running time is about half a minute on a SUN UltraSPARC-IIi.

**Keywords:** class field theory, complex multiplication, cryptography, elliptic curve, factoring polynomials, finite field

## 1 Introduction

Elliptic curve cryptography is a very efficient basic technology for public key infrastructures. An important computational problem in elliptic curve cryptography is the construction of cryptographically strong curves over finite prime fields. One approach is to randomly generate a curve, determine its cardinality by point counting, and then check whether it is cryptographically strong [MP97]. Unfortunately, point counting over large prime fields is rather slow.

Another method uses complex multiplication. It first searches for the cardinality of a cryptographically strong curve and it then constructs a curve of that cardinality. If the endomorphism ring of the curve has small class number, this is faster than searching for a strong curve by point counting. However, the German National Security agency (BSI), for example, considers small class numbers of the endomorphism ring as a possible security risk (see [BSI00]) since there are only very few. The BSI recommends using curves whose endomorphism ring is an order in an imaginary quadratic number field of discriminant $\geq 200$. For such curves, the standard complex multiplication method [AM93] is rather inefficient since the curve is constructed using a polynomial whose degree is the class number and which has extremely large coefficients.

In this paper we present a variant of the complex multiplication construction and show that it efficiently determines cryptographically strong elliptic curves whose endomorphism ring has large class number. Our algorithm constructs curves with endomorphism ring of class number 200 in 33 seconds on a SUN UltraSPARC-IIi. For class number 500 the algorithm requires less than two minutes. The algorithm uses very little storage. It can be used on any personal computer. This means that the users of a public key infrastructure can generate their own cryptographically strong elliptic curve.

Input of our algorithm is the discriminant $\Delta$ of an imaginary quadratic order $\mathcal{O}$. First, the algorithm searches for a prime number $p$ which is the norm of an element of $\mathcal{O}$, and such that there exists a cryptographically strong curve over the prime field $\mathbb{F}_p$. We show how to make this search fast. Once $p$ is found, the algorithm calculates a zero mod $p$ of a polynomial suggested by Yui and Zagier [YZ97], also used in [LZ94] which has much smaller coefficients than the polynomial from [AM93]. Using that zero, it is easy to construct the cryptographically strong curve and a point of large prime order on that curve.

The paper is organized as follows: In Section 2 we define cryptographically strong elliptic curves over prime fields. In Section 3 we present our algorithm. Finally we give examples and running times in Section 4.

## 2 Cryptographically strong elliptic curves

In this section we review a few basic facts concerning elliptic curves over finite fields and define cryptographically strong curves.

Let $p$ be a prime number, $p > 3$. An *elliptic curve* over the prime field $\mathbb{F}_p$ of characteristic $p$ is a pair $E = (a, b) \in \mathbb{F}_p^2$ with $4a^3 + 27b^2 \neq 0$. A *point* on $E$ is a solution $(x, y) \in \mathbb{F}_p^2$ of $y^2 = x^3 + ax + b$ or the point at infinity $O$ obtained by considering the projective closure of this equation. The set of points on $E$ over $\mathbb{F}_p$ is denoted by $E(\mathbb{F}_p)$. It carries a group structure with the point at infinity acting as the identity element.

We call the elliptic curve $E$ cryptographically strong if it satisfies the following conditions which make the cryptosystems, in which $E$ is used, secure and efficient.

We first consider security. If the elliptic curve $E$ is used in a cryptosystem, the security of this cryptosystem is based on the intractability of the discrete logarithm problem in the group of points $E(\mathbb{F}_p)$. Several discrete logarithm algorithms are known. To make their application impossible, we require that $E$ satisfies the following conditions.

1. We have $|E(\mathbb{F}_p)| = c \cdot l$ with a prime $l > 2^{160}$ and a positive integer $c$.
2. The primes $l$ and $p$ are different.
3. The order of $p$ in the multiplicative group $\mathbb{F}_l^\times$ of $\mathbb{F}_l$ is at least $\left\lceil \frac{2000}{\log_2(p)} \right\rceil$.

The first condition excludes the application of discrete logarithm algorithms whose running time is roughly the square root of the largest prime factor of the group order (see for example [vOM99]). The second condition makes the anomalous curve attack impossible (see [Sma99]). The last condition excludes the attack of Menezes, Okamoto, and Vanstone [MOV91] which reduces the discrete logarithm problem in $E(\mathbb{F}_p)$ to the discrete logarithm problem in a finite extension field of $\mathbb{F}_p$. The degree of this extension over $\mathbb{F}_p$ is at least the order of $p$ in $\mathbb{F}_l^\times$. The third condition is based on the assumption that the discrete logarithm problem in a finite field, whose cardinality is a 2000-bit number, is intractable. We remark that the German National Security Agency [BSI00] requires the order of $p$ in $\mathbb{F}_l^\times$ to be at least $10^4$.

Let us now consider efficiency. Suppose that an elliptic curve $E$ over a prime field $\mathbb{F}_p$ satisfies the security conditions. If this curve is used in a cryptosystem, the efficiency of this system depends on the efficiency of the arithmetic in $\mathbb{F}_p$. So $p$ should be as small as possible. It follows from a theorem of Hasse that

$$(\sqrt{|E(\mathbb{F}_p)|} - 1)^2 \leq p \leq (\sqrt{|E(\mathbb{F}_p)|} + 1)^2. \tag{1}$$

Hence, we try to make $|E(\mathbb{F}_p)|$ as small as possible. Now the first security condition implies

$$|E(\mathbb{F}_p)| = c \cdot l \tag{2}$$

with a prime number $l > 2^{160}$ and a positive integer $c$, the so called *cofactor*. The security of the cryptosystem, in which $E$ is used, is based on the intractability of the discrete logarithm problem in the subgroup of order $l$ in $E(\mathbb{F}_p)$. This security is independent of $c$. Therefore, $c$ can be as small as possible. In our algorithm $c = 1$ is not possible. But we require the following condition which implies the first security condition.

4. We have $|E(\mathbb{F}_p)| = c \cdot l$ with a prime number $l > 2^{160}$ and a positive integer $c \leq 4$.

We say that an elliptic curve $E$ over a finite prime field $\mathbb{F}_p$ is *cryptographically strong* if it satisfies the four conditions from above.

We explain an additional security condition required by the German National Security Agency BSI ([BSI00]). The third condition implies that the endomorphism ring $\mathrm{End}(E(\overline{\mathbb{F}}_p))$ of the elliptic curve over the algebraic closure of $\mathbb{F}_p$ is an imaginary quadratic order. The BSI requires the following.

5. The class number of the maximal order which contains $\mathrm{End}(E(\overline{\mathbb{F}}_p))$ is at least 200.

The reason for this condition is that among all curves over a prime field only very few have endomorphism rings with small class numbers. So those curves may be subject to specific attacks. However, no such attacks are known.

# 3 The algorithm

Let $\Delta$ be a negative integer, $\Delta \equiv 0, 1 \bmod 4$ and let $\mathcal{O}$ be the imaginary quadratic order of discriminant $\Delta$. This order is denoted by $\mathcal{O}_\Delta$. Also, let $p$ be a prime number which is the norm of an element in $\mathcal{O}$ that is, there are positive integers $t, y$ such that

$$t^2 - \Delta y^2 = 4p. \tag{3}$$

Using complex multiplication, elliptic curves $E_{1,p}$ and $E_{2,p}$ over $\mathbb{F}_p$ with endomorphism ring $\mathcal{O}$ and

$$|E_{1,p}(\mathbb{F}_p)| = p+1-t, \quad |E_{2,p}(\mathbb{F}_p)| = p+1+t \tag{4}$$

can be constructed as follows (see [AM93], [BSS99]).

Let $H \in \mathbb{Z}[X]$ be the the minimal polynomial of $j(\frac{\Delta+\sqrt{\Delta}}{2})$ where $j = 12^3 J$ with Klein's modular function $J$ (see [Apo90]). Modulo $p$ the polynomial $H$ splits into linear factors. Let $j_p$ be a zero of $H \bmod p$ that is, $j_p$ is an integer such that $H(j_p) \equiv 0 \bmod p$. We assume $\Delta < -4$ in what follows. Then we have $j_p \notin \{0; 1728\}$. Let $s_p$ be a quadratic nonresidue mod $p$. With

$$\kappa_p = \frac{j_p}{1728 - j_p}, \quad (a_p, b_p) = (3\kappa_p, 2\kappa_p) \tag{5}$$

we have

$$\{E_{1,p}, E_{2,p}\} = \{(a_p, b_p), (a_p s_p^2, b_p s_p^3)\}. \tag{6}$$

After this construction it is not known which of the curves is $E_{1,p}$ and which is $E_{2,p}$. However by choosing points on each curve and testing whether their order is a divisor of $p+1+t$ or $p+1-t$, the curves $E_{1,p}$ and $E_{2,p}$ can be identified.

We can decide whether one of the curves $E_{1,p}$ or $E_{2,p}$ is cryptographically strong before we actually construct those curves. We only need to know the prime number $p$ and its representation (3). Then we know the orders of $E_{1,p}$ and $E_{2,p}$ from (4). Using those orders we can check whether one of the curves is cryptographically strong because the conditions from the previous section only depend on $p$ and the order of the curve.

Our algorithm `cryptoCurve`$(\Delta, v, w)$ for computing cryptographically strong curves proceeds as follows. Input is a negative integer $\Delta$ with $\Delta \equiv 1 \bmod 8$, $\Delta \not\equiv 0 \bmod 3$, and positive integers $v, w$ with $w > v > 2^{162}$. It uses the algorithm `findPrime`$(\Delta, v, w)$ explained below which determines a prime number $p \in [v, w]$ with a representation (3) such that $E_{1,p}$ or $E_{2,p}$ is cryptographically strong. It also returns the prime number $l$ from (2) with $l > 2^{160}$. Once $p$ is found, the algorithm `findCurve`$(\Delta, p, l)$ constructs $E_{1,p}$ and $E_{2,p}$ and returns one of those curves which is cryptographically strong. `findCurve` also returns a point $P$ of order $l$ on the curve. We only use discriminants $\Delta \equiv 1 \bmod 8$, $\Delta \not\equiv 0 \bmod 3$ because for those discriminants `findCurve` can be implemented very efficiently. By Lemma 1, the order of a curve with endomorphism ring $\mathcal{O}_\Delta$ is then divisible by 4.

`cryptoCurve`$(\Delta, v, w)$

---

**Input:** $\Delta \in \mathbb{Z}_{<0}$, $\Delta \equiv 1 \bmod 8$, $\Delta \not\equiv 0 \bmod 3$, $v, w \in \mathbb{Z}$, $w > v > 2^{162}$.
**Output:** A prime $p \in [v, w]$.
A cryptographically strong elliptic curve $E$ over $\mathbb{F}_p$ with endomorphism ring $\mathcal{O}_\Delta$.
A prime divisor $l$ of $|E(\mathbb{F}_p)|$ with $l > 2^{160}$ such that $|E(\mathbb{F}_p)| = 4l$.
A point $P$ in $E(\mathbb{F}_p)$ of order $l$.

$(p, l) \leftarrow$ `findPrime`$(\Delta, v, w)$;
$(E, P) \leftarrow$ `findCurve`$(\Delta, p, l)$;
return $(p, E, l, P)$;

In order for the algorithm `findPrime`$(\Delta, v, w)$ to be successful it is necessary that the interval $[v, w]$ is sufficiently large. We use intervals of length $2^{122}$.

To satisfy condition 5 of the previous section, we choose $\Delta$ such that $\mathcal{O}_\Delta$ is a maximal order and the class number of $\mathcal{O}_\Delta$ is at least 200. Also, if we want to generate many different curves for a fixed $\Delta$, then we choose many pairwise disjoint intervals $[v, w]$.

Next, we explain `findPrime`$(\Delta, v, w)$. So let $\Delta, v, w$ be as in `cryptoCurve`$(\Delta, v, w)$. We have to find positive integers $t$ and $y$ such that $p = (t^2 - \Delta y^2)/4$ is a prime number, $p \in [v, w]$, and $(p+1-t)/4$ or $(p+1+t)/4$ is a prime $l > 2^{160}$. As the next Lemma shows, those requirements imply congruence conditions for $t$ and $y$.

**Lemma 1** *If* $\Delta \equiv 1 \bmod 8$ *and* $t, y$ *are positive integers such that* $p = (t^2 - \Delta y^2)/4$ *is a prime, then* $(t \bmod 4, y \bmod 4) \in \{(0,2),(2,0)\}$ *and* $p+1+t \equiv p+1-t \equiv 0 \bmod 4$. *Also, if* $(p+1+t)/4$ *is prime, then* $(t \bmod 8, y \bmod 8) \in \{(2,0),(6,4)\}$ *and if* $(p+1-t)/4$ *is prime, then* $(t \bmod 8, y \bmod 8) \in \{(2,4),(6,0)\}$.

**Proof:** Since $\Delta$ is odd, we have that $t^2 - \Delta y^2 \equiv 0 \bmod 4$ if and only if $t \equiv y \equiv 0 \bmod 2$ or $t \equiv y \equiv 1 \bmod 2$. Let $t$ and $y$ both be odd. Then $t^2 \equiv y^2 \equiv 1 \bmod 8$ and $t^2 - \Delta y^2 \equiv 1 - 1 \cdot 1 \equiv 0 \bmod 8$. Therefore, $(t^2 - \Delta y^2)/4$ is even. Now let $t$ and $y$ both be even. If $t \equiv y \equiv 0 \bmod 4$, then $t^2 - \Delta y^2 \equiv 0 \bmod 16$, and again $(t^2 - \Delta y^2)/4$ is even. The same is true if $t \equiv y \equiv 2 \bmod 4$ since we have $t^2 \equiv y^2 \equiv 4 \bmod 8$ and $t^2 - \Delta y^2 \equiv 0 \bmod 8$. A necessary condition for $(t^2 - \Delta y^2)/4$ to be an odd prime is therefore $(t \bmod 4, y \bmod 4) \in \{(0,2),(2,0)\}$. Then it is easy to see that $p+1+t \equiv p+1-t \equiv 0 \bmod 4$. As both $t$ and $y$ are even, there are integers $t_0$, $y_0$ with $t = 2t_0$ and $y = 2y_0$. Let $(t \bmod 4, y \bmod 4) = (0,2)$. Then we have $t_0 = 2t_1$ with an integer $t_1$ and $y_0^2 \equiv 1 \bmod 8$. It follows that $p + 1 \pm t = t_0^2 - \Delta y_0^2 + 1 \pm 2t_0 \equiv 4t_1^2 - 1 + 1 \pm 4t_1 = 4t_1(t_1 \pm 1) \equiv 0 \bmod 8$. Hence, neither $(p+1+t)/4$ nor $(p+1-t)/4$ can be prime. Now let $(t \bmod 4, y \bmod 4) = (2,0)$. Furthermore, assume that $y_0 \equiv 0 \bmod 4$. Then we have $y_0^2 \equiv 0 \bmod 8$ and $p + 1 \pm t = t_0^2 - \Delta y_0^2 + 1 \pm 2t_0 \equiv 1 - 1 \cdot 0 + 1 \pm 2t_0 = 2(1 \pm t_0) \bmod 8$. If $t_0 \equiv 1 \bmod 4$, that is $(t \bmod 8, y \bmod 8) = (2,0)$, we get $p+1+t \equiv 4 \bmod 8$ and $p+1-t \equiv 0 \bmod 8$. Hence, only $(p+1+t)/4$ can be prime. If $t_0 \equiv 3 \bmod 4$, that is $(t \bmod 8, y \bmod 8) = (6,0)$, we get $p+1+t \equiv 0 \bmod 8$ and $p+1-t \equiv 4 \bmod 8$. Hence, only $(p+1-t)/4$ can be prime. The case $y_0 \equiv 2 \bmod 4$ is treated analogously.$\square$

It follows from Lemma 1 that the order of a cryptographically strong elliptic curve $E$ over $\mathbb{F}_p$ is $4l$ with a prime $l > 2^{160}$. In the algorithm `findPrime` we first initialize $y$ to a random integer in $[1, \lfloor\sqrt{4v/|\Delta|}\rfloor]$ which is 0 mod 8. For all possible $t \equiv 2 \bmod 4$ we try to find a cryptographically strong curve. By Lemma 1 the number $(p+1+t)/4$ can only be prime if $t \equiv 2 \bmod 8$. If this is true, we test whether $p+1+t$ is the order of a cryptographically strong curve by using the function `isStrong`. If the order happens to be cryptographically strong, this function returns the prime $l$ such that $p+1+t = 4l$. Otherwise, the function returns 0. If $t \equiv 6 \bmod 8$, then we test whether $p+1-t$ is the order of a cryptographically strong curve. If no cryptographically strong curve is found, then the algorithm chooses a new $y$ and repeats the above procedure. Since $y$ is chosen randomly, the algorithm returns a different curve each time it is called.

`findPrime`$(\Delta, v, w)$

---

**Input:** $\Delta \in \mathbb{Z}_{<0}$, $\Delta \equiv 1 \bmod 8$, $\Delta \not\equiv 0 \bmod 3$, $v, w \in \mathbb{Z}$, $w > v > 2^{162}$.
**Output:** Primes $p \in [v, w]$ and $l > 2^{160}$
  such that $|E(\mathbb{F}_p)| = 4l$ for a cryptographically strong curve $E$ over $\mathbb{F}_p$
  with endomorphism ring $\mathcal{O}_\Delta$.

**while** true **do**
  Choose a random $y$ in $[1, \lfloor\sqrt{4v/|\Delta|}\rfloor]$ with $y \equiv 0 \bmod 8$;
  $t \leftarrow 4\lceil(\sqrt{4v - |\Delta|y^2} - 2)/4\rceil + 2$;
  **while** $p \leftarrow (t^2 + |\Delta|y^2)/4 \leq w$ **do**
    **if** $p$ is prime **then**
      **if** $t \equiv 2 \bmod 8$ and $l \leftarrow$ `isStrong`$(p, p+1+t) \neq 0$ **then**
        return $(p, l)$;
      **else if** $l \leftarrow$ `isStrong`$(p, p+1-t) \neq 0$ **then**
        return $(p, l)$;
      **end if**
    **end if**
    $t \leftarrow t + 4$;
  **end while**
**end while**

If the interval $[v, w]$ used in `findPrime` is too small then `findPrime` may not terminate. According to our experiments, intervals of length $2^{122}$ seem to be sufficiently large.

The function `isStrong` decides whether a curve, whose order is determined in `findPrime`, is cryptographically strong. The order of $p$ in the multiplicative group $\mathbb{F}_l^\times$ is denoted by $\mathrm{ord}_{\mathbb{F}_l^\times}(p)$. The algorithm implements the conditions from Section 2.

`isStrong`$(p, N)$

**Input:** A prime $p$ and $N \in \mathbb{N}$, $N \equiv 0 \bmod 4$.
**Output:** A prime $l$ with $N = 4l$ if $N$ is the order of a cryptographically strong elliptic curve over $\mathbb{F}_p$,
  $l = 0$ otherwise.

**if** $l \leftarrow N/4$ is prime and $l > 2^{160}$ and $p \neq l$ and $\mathrm{ord}_{\mathbb{F}_l^\times}(p) \geq \left\lceil \frac{2000}{\log_2(p)} \right\rceil$ **then**
  return $l$;
**else**
  return 0;
**end if**

The function `findCurve` implements the construction of the curves as explained in (5) and (6).

`findCurve`$(\Delta, p, l)$

**Input:** $\Delta \in \mathbb{Z}_{<0}$, $\Delta \equiv 1 \bmod 8$, $\Delta \not\equiv 0 \bmod 3$. A prime $p$ such that there exists a curve of order $4l$ over $\mathbb{F}_p$.
**Output:** An elliptic curve $E$ over $\mathbb{F}_p$ with $|E(\mathbb{F}_p)| = 4l$ and endomorphism ring $\mathcal{O}_\Delta$.
  A point $P$ in $E(\mathbb{F}_p)$ of order $l$.

$j_p \leftarrow$ `findRoot`$(\Delta, p)$;
Select a quadratic nonresidue $s$ mod $p$;
$E_1 \leftarrow (a_p, b_p)$;
$E_2 \leftarrow (a_p s_p^2, b_p s_p^3)$;
**while** true **do**
  Choose $Q_1$ randomly in $(E_1(\mathbb{F}_p)) \setminus \{O\}$;
  Choose $Q_2$ randomly in $(E_2(\mathbb{F}_p)) \setminus \{O\}$;

**if** $4Q_1 \neq O$ and $4lQ_1 = O$ **then**
  return $(E_1, 4Q_1)$;
**else if** $4Q_2 \neq O$ and $4lQ_2 = O$ **then**
  return $(E_2, 4Q_2)$;
**end if**
**end while**

The function `findRoot`$(\Delta, p)$ determines a zero mod $p$ of the polynomial $H$ from above. If the class number of $\mathcal{O}_\Delta$ is large then the coefficients of $H$ are extremely large. Instead, following [YZ97], we use a polynomial $W$ of the same degree which is computed by `findPolynomial(`$\Delta$`)`. This polynomial has much smaller coefficients. For example, if $\Delta = -71$ then

$$\begin{aligned} H &= X^7 + 313645809715X^6 - 3091990138604570X^5 + \\ &\quad 98394038810047812049302X^4 - 823534263439730779968091389X^3 + \\ &\quad 5138800366453976780323726329446X^2 - 425319473946139603274605151187659X + \\ &\quad 737707086760731113357714241006081263 \end{aligned}$$

and

$$W = X^7 - X^6 - X^5 + X^4 - X^3 - X^2 + 2X + 1.$$

Also, if $w_p$ is a zero of $W$ mod $p$ then

$$j_p = (w_p^{24} - 16)^3 / w_p^{24}$$

is a zero of $H$ mod $p$.

`findRoot`$(\Delta, p)$

---

**Input:** $\Delta \in \mathbb{Z}_{<0}$, $\Delta \equiv 1 \bmod 8$, $\Delta \not\equiv 0 \bmod 3$.
A prime $p$ which is the norm of an element of $\mathcal{O}_\Delta$.
**Output:** A root $j_p$ of $H$ mod $p$.

$W \leftarrow$ `findPolynomial(`$\Delta$`)`;
$w_p \leftarrow$ `find_root`$(p, W)$;
return $j_p \leftarrow (w_p^{24} - 16)^3 / w_p^{24}$;

Next, we explain `findPolynomial(`$\Delta$`)`. In this algorithm, we first compute the set $R_\Delta$ of reduced integral primitive forms $(a, b, c) = ax^2 + bxy + cy^2$ of discriminant $\Delta$. An algorithm for computing this set can be found in [Coh95], p.228. Using $R_\Delta$ we define the polynomial $W$. For $\tau$ in the upper half plane of $\mathbb{C}$ we set $q_\tau = e^{2\pi i \tau}$. The Weber functions (see [Web02]) are

$$\mathfrak{f}(\tau) = q_\tau^{-\frac{1}{48}} \prod_{n=1}^{\infty} (1 + q_\tau^{n - \frac{1}{2}}) \, , \tag{7}$$

$$\mathfrak{f}_1(\tau) = q_\tau^{-\frac{1}{48}} \prod_{n=1}^{\infty} (1 - q_\tau^{n - \frac{1}{2}}) \, , \tag{8}$$

$$\mathfrak{f}_2(\tau) = \sqrt{2} q_\tau^{\frac{1}{24}} \prod_{n=1}^{\infty} (1 + q_\tau^{n}) \, . \tag{9}$$

For $g = (a,b,c) \in R_\Delta$ set $\tau_g = \frac{-b+i\sqrt{|\Delta|}}{2a}$. Then $\tau_g$ lies in the upper half plane of $\mathbb{C}$. We set

$$f(g) := \begin{cases} \zeta^{b(a-c-ac^2)} \cdot \mathfrak{f}(\tau_g) & \text{if} \quad 2 \mid a, 2 \mid c, \\ (-1)^{\frac{\Delta-1}{8}} \cdot \zeta^{b(a-c-ac^2)} \cdot \mathfrak{f}_1(\tau_g) & \text{if} \quad 2 \mid a, 2 \nmid c, \\ (-1)^{\frac{\Delta-1}{8}} \cdot \zeta^{b(a-c+a^2c)} \cdot \mathfrak{f}_2(\tau_g) & \text{if} \quad 2 \nmid a, 2 \mid c \end{cases} \tag{10}$$

with $\zeta = e^{2\pi i/48}$. Then the polynomial $W$ is

$$W = \prod_{g \in R_\Delta} (X - f(g)). \tag{11}$$

This polynomial has integer coefficients. In `findPolynomial`($\Delta$), we compute $W$ using (11). The zeros $f(g)$ of $W$, $g \in R_\Delta$ are defined as infinite products. We approximate them by truncated products using the LiDIA type `bigcomplex` with precision (as in [LZ94])

$$F = \frac{G + \frac{h}{4} + 5}{47} + 1$$

where

$$G = \frac{\pi\sqrt{|\Delta|}}{\log 10} \cdot \sum_{(a,b,c) \in R_\Delta} \frac{1}{a}.$$

More precisely, we keep multiplying the factors of $f(g)$ in (7), (8), or (9) until five successive partial products are equal within the chosen precision. The following observation can be used to speed up the algorithm. If $(a,b,c) \in R_\Delta$ then $f(a,-b,c) = \overline{f(a,b,c)}$. Hence, if $(a,-b,c)$ also belongs to $R_\Delta$, then we obtain $f(a,-b,c)$ almost for free. By multiplying all linear factors $X - f(g)$ using our truncated products $f(g)$, we obtain a polynomial whose coefficients are close to integers. We determine $W$ by rounding each coefficient to the closest integer.

To compute a zero $w_p$ of $W$ mod $p$ we use the LiDIA-function `find_root`$(p, W)$. As input this function requires a prime $p$ and a polynomial $W \in \mathbb{Z}[X]$ which splits into linear factors modulo $p$. It returns a zero of $W$ mod $p$. `find_root` uses the Cantor-Zassenhaus split (see [Coh95]) and a polynomial arithmetic due to Shoup [Sho95].

# 4 Examples and running times

We have implemented our algorithm in C++ using the library LiDIA 2.0 ([LiDIA]) and the GNU compiler 2.95.2 . The timings were measured on a SUN UltraSPARC-IIi running Solaris 2.6 at 333 MHz and having 512 MB of main memory.

We present three examples. Timings can be found in Tables 1, 2 and 3

Let us first consider the fundamental discriminant $\Delta = -21311$ with $h = 200$. We remark that $\Delta$ is maximal with this property. We use the floating point precision $F = 48$. Running times can be found in table 2. We get the following output:

$$\begin{aligned}
p &= 1124388104722008836025816920368001511855982082 1349 \\
\lfloor \log_2 p \rfloor + 1 &= 163 \\
l &= 2810970261805022090064542583157653944571338913691 \\
\lfloor \log_2 l \rfloor + 1 &= 161 \\
E &= (9664333206544914885970277381389792397205945721715, \\
&\phantom{= (}9712273578800196724738364136255277465294004742516) \\
|E(\mathbb{F}_p)| &= 4 \cdot l \\
P &= (9734858823530657227741101030084051747415204761009, \\
&\phantom{= (}986958879780570993447621164009014806868 5)
\end{aligned}$$

In the second example we use the fundamental discriminant $\Delta = -96599$ with $h = 500$. Again, $\Delta$ is maximal with this property. We use $F = 129$. We obtain the following output:

$$\begin{aligned}
p &= 1123936402202444560239507107629140916141800479044 9 \\
\lfloor \log_2 p \rfloor + 1 &= 163 \\
l &= 2809841005506111400598767640310076774940711944009 \\
\lfloor \log_2 l \rfloor + 1 &= 161 \\
E &= (3056338225511409442512995201958098190832410825281, \\
&\phantom{= (}9530468165023903363272044185499671568166943743820) \\
|E(\mathbb{F}_p)| &= 4 \cdot l \\
P &= (7706333343307924406946634579911418698936319570270, \\
&\phantom{= (}404612800210025285669535480167217967868 60)
\end{aligned}$$

To show the limit of the algorithm we also present an example with the fundamental discriminant $\Delta = -10000031$ and $h = 5426$. We use the floating point precision $F = 1986$. We get the following output:

$$\begin{aligned}
p &= 1157522434958478330458658960293003539758595639708 9 \\
\lfloor \log_2 p \rfloor + 1 &= 163 \\
l &= 2893806087396195826146647734591894354891848869089 \\
\lfloor \log_2 l \rfloor + 1 &= 161 \\
E &= (975815186120457023995003896354016930095933053 88, \\
&\phantom{= (}6505434574136380159966692642360112867306220359 2) \\
|E(\mathbb{F}_p)| &= 4 \cdot l \\
P &= (4475160835035949933240439608822439678768106791477, \\
&\phantom{= (}6992349469932163330795423460823116723197169323796)
\end{aligned}$$

The total running time was 1 day 13 hour 41 min 39 sec. The time to compute $W$ and $w_p$ was 1 day 13 hour 21 min 56 sec and 19 min 13 sec, respectively.

| $\Delta$ | $h$ | Number of tests | Time of `cryptoCurve` | Time of `findPrime` | Time of `findPolynomial` | Time of `find_root` |
|---|---|---|---|---|---|---|
| -455 | 20 | 100 | 6.235 | 5.275 | 0.05463 | 0.6242 |
| | | | 5.172 | 5.166 | 0.03266 | 0.05251 |
| -1271 | 40 | 100 | 8.278 | 5.871 | 0.1245 | 2.004 |
| | | | 5.345 | 5.347 | 0.04943 | 0.1157 |
| -2159 | 60 | 100 | 10.03 | 5.079 | 0.2492 | 4.421 |
| | | | 4.901 | 4.869 | 0.08125 | 0.2091 |
| -5183 | 80 | 100 | 15.08 | 6.238 | 0.3778 | 8.174 |
| | | | 7.269 | 7.213 | 0.1382 | 0.3276 |
| -7991 | 100 | 100 | 17.02 | 5.823 | 0.6289 | 10.28 |
| | | | 5.806 | 5.805 | 0.2033 | 0.4606 |
| -11879 | 150 | 100 | 27.52 | 6.418 | 1.272 | 19.46 |
| | | | 5.665 | 5.654 | 1932 | 0.7965 |

Table 1: Timings on UltraSPARC-IIi for fundamental discriminants of class number 20 up to 150. All timings are given in seconds. Each first row contains the running time, each second row the standard deviation.

| $\Delta$ | $h$ | Number of tests | Time of `cryptoCurve` | Time of `findPrime` | Time of `findPolynomial` | Time of `find_root` |
|---|---|---|---|---|---|---|
| -21311 | 200 | 100 | 33.06 | 5.808 | 2.420 | 24.52 |
| | | | 5.463 | 5.458 | 0.009839 | 0.5377 |
| -31031 | 200 | 100 | 31.98 | 4.668 | 2.739 | 24.26 |
| | | | 4.752 | 4.770 | 0.003942 | 0.5232 |
| -40895 | 200 | 100 | 33.70 | 6.456 | 2.720 | 24.20 |
| | | | 5.793 | 5.710 | 0.004976 | 0.5489 |
| -50183 | 200 | 100 | 33.89 | 6.198 | 3.042 | 24.33 |
| | | | 6.166 | 6.111 | 0.008005 | 0.5736 |
| -100015 | 200 | 100 | 33.45 | 5.791 | 3.033 | 24.29 |
| | | | 4.971 | 4.888 | 0.007614 | 0.5901 |
| -500047 | 200 | 100 | 33.77 | 4.540 | 4.368 | 24.43 |
| | | | 3.831 | 3.838 | 0.01136 | 0.5162 |

Table 2: Timings on UltraSPARC-IIi for fundamental discriminants of class number 200. All timings are given in seconds. Each first row contains the running time, each second row the standard deviation.

| Δ | $h$ | Number of tests | Time of `cryptoCurve` | Time of `findPrime` | Time of `findPolynomial` | Time of `find_root` |
|---|---|---|---|---|---|---|
| -96599 | 500 | 25 | 101.2 | 8.146 | 29.30 | 63.36 |
| | | | 7.107 | 6.782 | 0.1116 | 1.201 |
| -148511 | 600 | 25 | 152.9 | 5.846 | 49.03 | 97.56 |
| | | | 5.948 | 4.450 | 0.2956 | 2.642 |
| -185471 | 700 | 25 | 197.8 | 7.426 | 82.29 | 107.7 |
| | | | 9.068 | 8.576 | 0.5049 | 1.572 |
| -233999 | 800 | 25 | 243.2 | 4.862 | 120.6 | 117.2 |
| | | | 4.613 | 4.259 | 0.4135 | 1.034 |
| -299519 | 900 | 25 | 320.5 | 4.702 | 188.8 | 126.5 |
| | | | 5.258 | 5.486 | 1.141 | 1.684 |
| -412079 | 1000 | 25 | 430.0 | 6.282 | 283.1 | 140.0 |
| | | | 6.638 | 6.708 | 0.3415 | 1.580 |

Table 3: Timings on UltraSPARC-IIi for fundamental discriminants of class number 500 up to 1000. All timings are given in seconds. Each first row contains the running time, each second row the standard deviation.

## References


[AM93] A.O.L. Atkin and F. Morain. Elliptic curves and primality proving. *Mathematics of Computation*, 61:29–67, 1993.

[Apo90] T.M. Apostol. *Modular Functions and Dirichlet Series in Number Theory.* Springer-Verlag, 1990.

[BSI00] Geeignete Kryptoalgorithmen gemäß §17(2) SigV, April 2000. Bundesamt für Sicherheit in der Informationstechnik.

[BSS99] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography.* Cambridge University Press, 1999.

[Coh95] H. Cohen. *A Course in Computational Algebraic Number Theory.* Springer-Verlag, 1995.

[LiDIA] LiDIA. *A library for computational number theory.* Technical University of Darmstadt. available via WWW from http://www.informatik.tu-darmstadt.de/TI/LiDIA/Welcome.html.

[LZ94] G.-J. Lay and H.G. Zimmer. Constructing elliptic curves with given group order over large finite fields. In *Proceedings of ANTS I*, LNCS 877, pages 250–263, 1994.

[MOV91] A. Menezes, T. Okamoto, and S. Vanstone. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. In *Proceedings of the 23rd Annual ACM Symposium on the Theory of Computing*, pages 80–89, 1991.

[MP97] V. Müller and S. Paulus. On the Generation of Cryptographically Strong Elliptic Curves. Technical Report, Technical University of Darmstadt, 1997.

[Sho95] V. Shoup. A new polynomial factorization algorithm and its implementation. *Journal of Symbolic Computation*, 20:363–397, 1995.

[Sma99] N.P. Smart. The Discrete Logarithm Problem on Elliptic Curves of Trace One. *Journal of Cryptology*, 12/3:193–196, 1999.

[vOM99] P.C. van Oorschot and M.J.Wiener. Parallel Collision Search with Cryptanalytic Applications. *Journal of Cryptology*, 12/1:1–28, 1999.

[Web02] H. Weber. *Lehrbuch der Algebra III.* Chelsea Publishing Company, 1902.

[YZ97] N. Yui and D. Zagier. On the singular values of Weber modular functions. *Mathematics of Computation*, 66:1645–1662, 1997.