Faster Attacks on Elliptic Curve Cryptosystems

Michael J. Wiener & Robert J. Zuccherato

Entrust Technologies
750 Heron Road
Ottawa, Ontario
Canada K1V 1A7
{wiener, robert.zuccherato}@entrust.com

Abstract. The previously best attack known on elliptic curve cryptosystems used in practice was the parallel collision search based on Pollard's ρ-method. The complexity of this attack is the square root of the prime order of the generating point used. For arbitrary curves, typically defined over $GF(p)$ or $GF(2^m)$, the attack time can be reduced by a factor or $\sqrt{2}$, a small improvement. For subfield curves, those defined over $GF(2^{ed})$ with coefficients defining the curve restricted to $GF(2^e)$, the attack time can be reduced by a factor of $\sqrt{2d}$. In particular for curves over $GF(2^m)$ with coefficients in $GF(2)$, called anomalous binary curves or Koblitz curves, the attack time can be reduced by a factor of $\sqrt{2m}$. These curves have structure which allows faster cryptosystem computations. Unfortunately, this structure also helps the attacker. In an example, the time required to compute an elliptic curve logarithm on an anomalous binary curve over $GF(2^{163})$ is reduced from 2^{81} to 2^{77} elliptic curve operations.

1 Introduction

Public-key cryptography based on elliptic curves over finite fields was proposed by Miller [9] and Koblitz [6] in 1985. Elliptic curves over finite fields have been used to implement the Diffie-Hellman key passing scheme [2, 4] and also the elliptic curve variant of the Digital Signature Algorithm [1, 10]. The security of these cryptosystems relies on the difficulty of solving the *elliptic curve discrete logarithm problem*. If P is a point with order n on an elliptic curve, and Q is some other point on the same curve, then the elliptic curve discrete logarithm problem is to determine an l such that $Q = lP$ and $0 \leq l \leq n - 1$ if such an l exists. If this problem can be solved efficiently, then elliptic curve based cryptosystems can be broken efficiently.

There are known classes of elliptic curves in which solving the discrete logarithm problem is (relatively) easy. These classes include supersingular curves [8] and anomolous curves [12, 14, 15]. The elliptic curve discrete logarithm problem for supersingular curves can be reduced to the discrete logarithm problem in a small finite extension of the underlying finite field. The discrete logarithm problem in the finite field can then be solved in subexponential time. Anomolous curves are curves defined over the field $GF(p)$ and have exactly p points. The elliptic curve discrete logarithm problem for anomolous curves can be solved in $O(\ln p)$ operations. Both supersingular and anomolous curves are easily identified and excluded from use in cryptographic operations.

The best attack known on the general elliptic curve discrete logarithm problem is parallel collision search [18] based on Pollard's ρ algorithm [11] which has running time proportional to the square root of the largest prime factor dividing the curve order. This method works for any cyclic group and does not make use of any additional structure present in elliptic curve groups. We show how this method can be improved for any elliptic curve logarithm computation by exploiting the fact that the negative of a point can be computed very rapidly.

Certain classes of elliptic curves have been proposed for use in cryptography because of their ability to provide efficiencies in implementation. Among these have been subfield curves and anomalous binary or Koblitz curves [7, 16]. Using the Frobenius endomorphism, we show that these curves also allow a further speed-up for the parallel collision search algorithm and therefore provide less security than was originally thought. This is the first time that the extra structure provided by these curves has actually been used to attack the cryptosystems upon which they are based. Independant work in this area has also been performed by Robert Gallant, Robert Lambert and Scott Vanstone [5] and by Robert Harley, who has used his results to solve the ECC2K-95 Certicom challenge problem.

2 Background

This section will provide the necessary background material on various properties of elliptic curves and will also describe the parallel collision search method for computing discrete logarithms.

2.1 Elliptic Curves Over $GF(p)$

Let $GF(p)$ be a finite field of characteristic $p \neq 2, 3$, and let $a, b \in GF(p)$ satisfy the inequality $4a^3 + 27b^2 \neq 0$. An *elliptic curve*, $E_{(a,b)}(GF(p))$, is defined as the set of points $(x, y) \in GF(p) \times GF(p)$ which satisfy the equation

$$y^2 = x^3 + ax + b,$$

together with a special point, $\mathcal{O}$, called the *point at infinity*. These points form an abelian group under a well-defined addition operation which we now describe.

Let $E_{(a,b)}(GF(p))$ be an elliptic curve and let P and Q be two points on $E_{(a,b)}(GF(p))$. If $P = \mathcal{O}$, then $-P = \mathcal{O}$ and $P + Q = Q + P = Q$. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$. Then $-P = (x_1, -y_1)$ and $P + (-P) = \mathcal{O}$. If $Q \neq -P$ then $P + Q = (x_3, y_3)$ where

$$\begin{aligned} x_3 &= \mu^2 - x_1 - x_2 \\ y_3 &= \mu(x_1 - x_3) - y_1, \end{aligned}$$

and

$$\mu = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} \text{ if } P \neq Q \\ \\ \dfrac{3x_1^2 + a}{2y_1} \text{ if } P = Q. \end{cases}$$

2.2 Elliptic Curves Over $GF(2^m)$

We now consider non-supersingular elliptic curves defined over fields of characteristic 2. Let $GF(2^m)$ be such a field for some $m \geq 1$. Then a non-supersingular elliptic curve is defined to be the set of solutions $(x, y) \in GF(2^m) \times GF(2^m)$ to the equation

$$y^2 + xy = x^3 + ax^2 + b$$

where $a, b \in GF(2^m)$ and $b \neq 0$, together with the point on the curve at infinity, $\mathcal{O}$. We denote this elliptic curve by $E_{(a,b)}(GF(2^m))$ or (when the context is understood) E.

The points on an elliptic curve form an abelian group under a well defined group operation. The identity of the group operation is the point $\mathcal{O}$. For $P = (x_1, y_1)$ a point on the curve, we define $-P$ to be $(x_1, y_1 + x_1)$, so $P + (-P) = (-P) + P = \mathcal{O}$. Now suppose P and Q are not $\mathcal{O}$, and $P \neq -Q$. Let P be as above and $Q = (x_2, y_2)$, then $P + Q = (x_3, y_3)$, where

$$\begin{aligned} x_3 &= \mu^2 + \mu + x_1 + x_2 + a \\ y_3 &= \mu(x_1 + x_3) + x_3 + y_1, \end{aligned}$$

and

$$\mu = \begin{cases} \dfrac{y_2 + y_1}{x_2 + x_1} \text{ if } P \neq Q \\[2ex] \dfrac{x_1^2 + y}{x_1} \text{ if } P = Q. \end{cases}$$

2.3 Anomalous Binary and Subfield Curves

Anomalous binary curves (also known as Koblitz curves) are elliptic curves over $GF(2^n)$ that have coefficients a and b either 0 or 1. Since it is required that $b \neq 0$, they must be defined by either the equation

$$y^2 + xy = x^3 + 1$$

or the equation

$$y^2 + xy = x^3 + x^2 + 1.$$

Since these curves allow very efficient implementations of certain elliptic curve cryptosystems, they have been particularly attractive to implementors of these schemes [7, 16]. Anomalous binary curves are just a special case of subfield curves which have also been proposed for use in elliptic curve cryptography because they also give efficient implementations.

If $m = ed$ for $e, d \in \mathbb{Z}_{>0}$, then $GF(2^e) \subset GF(2^m)$. Using underlying fields of this type provide very efficient implementations [3, 13]. If a and b are actually elements of $GF(2^e)$, then we say that E is a *subfield curve.* Notice in this case that $E_{(a,b)}(GF(2^e)) \subset E_{(a,b)}(GF(2^m))$.

If e is small, so that the number of points in $E_{(a,b)}(GF(2^e))$ can be easily counted, there is an easy way to determine the number of points in $E_{(a,b)}(GF(2^m))$. Denote by $\#E$ the number of points in E. Then it is well known that $\#E_{(a,b)}(GF(2^e)) = 2^e + 1 - t$ for some $t \leq 2\sqrt{2^e}$. The value t is known as the *trace* of the curve. If α and β are the two roots of the equation $X^2 - tX + 2^e = 0$, then $\#E_{(a,b)}(GF(2^m)) = 2^m + 1 - \alpha^d - \beta^d$. This is known as *Weil's Theorem.*

2.4 The Frobenius Endomorphism

An interesting property of anomalous binary curves is that if $P = (x, y)$ is a point on the curve, then so is (x^2, y^2). In fact $(x^2, y^2) = \lambda P$ for some constant λ. We can see this in the general case of subfield curves using the Frobenius endomorphism.

The Frobenius endomorphism is the function ψ that takes x to x^{2^e} for all $x \in GF(2^m)$. Since we are working in a field of characteristic 2, notice that $\psi(r(x)) = r(\psi(x))$ for all $x \in GF(2^m)$ and any rational function r with coefficients in $GF(2^e)$. If $P = (x, y)$ is a point on the subfield curve E, define $\psi(P) = (\psi(x), \psi(y))$. Also define $\psi(\mathcal{O}) = \mathcal{O}$. It can be shown from the curve's defining equation and the fact that $(a+b)^{2^e} = a^{2^e} + b^{2^e}$ for all $a, b \in GF(2^m)$ that if $P \in E$ then $\psi(P) \in E$. Thus if E is a subfield curve and $P, Q \in E$, then $\psi(P+Q) = \psi(P) + \psi(Q)$.

Now, consider a point $P \in E$ where E is a subfield curve and P has prime order p with p^2 not dividing $\#E$. By the above remarks we have $p\psi(P) = \psi(pP) = \psi(\mathcal{O}) = \mathcal{O}$. Hence $\psi(P)$ must also be a point of order P. Since $\psi(P) \in E$, we must have $\psi(P) = \lambda P$ for some $\lambda \in \mathbb{Z}$, $1 \le \lambda \le p-1$. The value λ is constant among all points in the subgroup generated by P and is known as the *eigenvalue* of the Frobenius endomorphism.

It is known that for any point $P \in E$, the Frobenius endomorphism satisfies

$$\psi^2(P) - t\psi(P) + 2^e P = \mathcal{O}$$

where t is the trace as defined in Section 2.3. Therefore, it can also be shown that λ is one of the roots of the quadratic congruence

$$X^2 - tX + 2^e \equiv 0 \pmod{p}.$$

Hence, λ can be efficiently computed.

2.5 Parallel Collision Search

Given a point Q on an elliptic curve which is in a subgroup of order n generated by P, we seek l such that $Q = lP$. Pollard's ρ method [11] proceeds as follows. Partition the points on the curve into three roughly equal size sets S_1, S_2, S_3 based on some simple rule. Define an iteration function on a point Z as follows

$$f(Z) = \begin{cases} 2Z & \text{if } Z \in S_1 \\ Z + P & \text{if } Z \in S_2. \\ Z + Q & \text{if } Z \in S_3. \end{cases}$$

Choose $A_0, B_0 \in [1, n-1]$ at random and compute the starting point $Z_0 = A_0 P + B_0 Q$. Compute the sequence $Z_1 = f(Z_0), Z_2 = f(Z_1), \ldots$ keeping track of A_i, B_i such that $Z_i = A_i P + B_i Q$. Thus,

$$(Z_{i+1}, A_{i+1}, B_{i+1}) = \begin{cases} (2Z_i, 2A_i, 2B_i) & \text{if } Z \in S_1 \\ (Z_i + P, A_i + 1, B_i) & \text{if } Z \in S_2. \\ (Z_i + Q, A_i, B_i + 1) & \text{if } Z \in S_3. \end{cases}$$

Note that A_i and B_i can be computed modulo n so that they do not grow out of control. Because the number of points on the curve is finite, the sequence of points must begin to repeat. Upon detection that $Z_i = Z_j$ we have $A_iP + B_iQ = A_jP + B_jQ$, which gives $l = \frac{A_i - A_j}{B_j - B_i} \bmod n$, unless we are very unlucky and $B_i \equiv B_j \pmod{n}$.

Actually, Pollard's function is not an optimal choice. In [17] it is recommended that the points be divided into about 20 sets of equal size $S_1, \ldots, S_{20}$ and that the iteration function be

$$f(Z) = \begin{cases} Z + c_1P + d_1Q & \text{if } Z \in S_1 \\ Z + c_2P + d_2Q & \text{if } Z \in S_2 \\ \quad \vdots & \quad \vdots \\ Z + c_{20}P + d_{20}Q & \text{if } Z \in S_{20} \end{cases} \tag{1}$$

where the c_i and d_i are random integers between 1 and $n-1$. The use of this iteration function gives a running time very close to that expected by theoretical estimates. In order to make computation of the values A_i and B_i more efficient, we suggest that constants $c_{11}, \ldots, c_{20}$ and $d_1, \ldots, d_{10}$ could be zero so that only one of the values A_i or B_i need to be updated at each stage.

Pollard's ρ method is inherently serial and cannot be directly parallelized over several processors efficiently. Parallel collision search [18] provides a method for efficient parallelization. Several processors each create their own starting points Z_0 and iterate until a "distinguished point" Z_d is reached. A point is considered distinguished if it satisfies some easily tested property such as having several leading zero bits. The triples (Z_d, A_d, B_d) are contributed to a memory common to all processors. When the memory holds two triples containing the same point Z_d, then the logarithm l can be computed as with Pollard's ρ method.

The expected number of iterations required to find the logarithm is $\sqrt{\frac{\pi n}{2}}$. The object of this paper is to reduce this number.

3 Faster Attacks for Arbitrary Curves

Notice that for elliptic curves over both $GF(p)$ and $GF(2^m)$, given a point $P = (x, y)$ on the curve it is trivial to determine its negative. Either $-P = (x, -y)$ (in the $GF(p)$ case) or $-P = (x, x + y)$ (in the $GF(2^m)$ case). Thus, at every stage of the parallel collision search algorithm, both Z_i and $-Z_i$ could be easily computed.

We would like to reduce the size of the space that is being searched by parallel collision search by a factor of 2. We can do this by replacing Z_i with $\pm Z_i$ at each step in a canonical way. A simple way to do this is to choose the one that has smallest y coordinate when its binary representation is interpreted as an integer.

When performing a parallel collision search, Z_i, A_i and B_i should be computed as normal. However, $-Z_i$ should also be computed, and whichever one of Z_i and $-Z_i$ has the smallest y coordinate should be taken to be Z_i. If Z_i has the smallest y coordinate, then everything progresses as normal. If $-Z_i$ has the smallest y coordinate then $-Z_i$ should replace Z_i, $-A_i$ should replace A_i and $-B_i$ should replace B_i. Notice that the equation $Z_i = A_iP + B_iQ$ is still maintained.

Thus, the search space for the parallel collision search is reduced to only those points which have a smaller y coordinate than their negative. Since exactly half of the points ($\neq \mathcal{O}$) have this property we have reduced the search space by a factor of 2. Because the extra computational effort in determining which of Z_i and $-Z_i$ to accept is negligible, the expected running time of the algorithm will be reduced by a factor of $\sqrt{2}$. This improvement in attack time is valid for any elliptic curve.

A technicality which affects the most obvious application of this technique is the appearance of trivial 2-cycles. Suppose that Z_i and Z_{i+1} both belong to the same S_j and that in both cases after f is applied, the negative of the resulting point is used. This is when $Z_{i+1} = -(Z_i + c_jP + d_jQ)$ (say) and $Z_{i+2} = -(Z_{i+1} + c_jP + d_jQ) = Z_i$. The occurrence of these 2-cycles is reduced by using the iteration function given in Equation (1) since it gives more choices for the multipliers. It does not reduce it enough so that efficient implementations are possible however. To reduce the occurrence of 2-cycles even further, we can use a look-ahead technique which proceeds as follows. Define $f_w(Z) \equiv Z + c_wP + d_wQ$. Suppose that $Z_i \in S_j$. Then $f(Z_i) = f_j(Z_i)$. Begin by computing $R = \pm f_j(Z_i)$, a candidate for Z_{i+1}. If $R \notin S_j$ then $Z_{i+1} = R$. If $R \in S_j$, then we treat Z_i as though it were in S_{j+1} (where $j+1$ is reduced modulo 20), and compute a new candidate $R = \pm f_{j+1}(Z_i)$. If $R \notin S_{j+1}$, then $Z_{i+1} = R$, otherwise continue trying $j+2, j+3, \ldots$. If all 20 choices fail (a very low probability event), then just use $Z_{i+1} = \pm f_j(Z_i)$. The idea is to reduce the probability that two successive points will belong to the same set. Note that Z_{i+1} still depends solely on Z_i, a requirement for parallel collision search to work.

This modified iteration function causes the amount of computation to increase by an expected factor of approximately $\frac{20}{19}$, a small penalty which can be reduced by using more than 20 cases. The occurrence of 2-cycles is not completely eliminated, but is significantly reduced. If necessary, it can be reduced further by using more than 20 cases or by looking ahead two steps instead of just one. Another way to deal with 2-cycles is to consider them to be distinguished points.

4 Faster Attacks for Subfield Curves

We will now describe an attack on subfield curves that again uses parallel collision search and will reduce the running time by a factor of $\sqrt{d}$ when considering curves over $GF(2^{ed})$.

Let $E_{(a,b)}(GF(2^{ed}))$ be a subfield curve with $a, b \in GF(2^e)$ and let P be a point on the curve such that not both coordinates are from a proper subfield of $GF(2^{ed})$. In other words $P \in E_{(a,b)}GF(2^{ed})$, but $P \notin E_{(a,b)}(GF(2^{ef}))$ for any f, $1 \leq f \leq d-1$. Let P have prime order p such that p^2 does not divide the order of the curve and let d be odd. These conditions are not restrictive since most elliptic curve cryptosystems require the use of points P with prime order very close to the curve order, which usually implies the above conditions.

By these conditions we get that

$$\begin{aligned} \psi(P) &= \lambda P \neq P, \\ \psi^2(P) &= \lambda^2 P \neq P, \\ &\vdots \\ \psi^{d-1}(P) &= \lambda^{d-1} P \neq P, \\ \psi^d(P) &= \lambda^d P = P \end{aligned}$$

which implies that $d | p-1$.

Remember that $\psi(x) = x^{2^e}$. Since we are working over a subfield of characteristic 2, squaring is always a very efficient operation. In particular when a normal basis representation is used, it is just a cyclic shift of the binary representation of the field element. Thus $\psi(P)$ can be computed very efficiently.

Similar to Section 3, we will use a parallel collision search and compute Z_i, A_i and B_i as usual. We can now also compute the $2d$ different points on the curve $\pm\psi^j(Z_i)$ for $0 \leq j \leq d-1$. We would like to choose a "distinguished" or "canonical" representative from this set. We will first consider the d points $\psi^j(Z_i)$ and use the one whose x coordinate's binary representation has smallest value when interpreted as an integer. We can then choose either that point or its negative depending on which has smaller y coordinate when interpreted as an integer. This point will now replace Z_i. If we have chosen $\pm\psi^j(Z_i)$ to replace Z_i, we must then replace A_i with $\pm\lambda^j A_i$ and also replace B_i with $\pm\lambda^j B_i$ to maintain the relationship $Z_i = A_iP + B_iQ$. The powers of λ^j can be precomputed to obtain further efficiencies.

By performing the above operation at every step of the parallel collision search, we will be reducing the size of our search space by a factor of $2d$. Thus, the expected running time to compute the discrete logarithm will decrease by a factor of $\sqrt{2d}$.

The iteration function f used in the parallel collision search must be chosen carefully. In particular, notice that if the function is chosen to be a choice between just $2Z$, $Z + P$ and $Z + Q$ (as in the basic parallel collision search algorithm), then in some situations trivial cycles are likely to occur. Notice that for $i < j$, Z_j can be written as $Z_j = p_1(\lambda)Z_i + p_2(\lambda)P + p_3(\lambda)Q$ where p_1, p_2 and p_3 are polynomials in λ. Also notice that these polynomials will have small coefficients if $j - i$ is not too big. When using anomalous binary curves, the value λ satisfies $\lambda^2 + \lambda + 2$ or $\lambda^2 - \lambda + 2$. In either case, λ will be likely to be a root of the polynomials in the expression for Z_j, and hence a trivial cycle will be encountered. Experimentation shows that the modified iteration function described in Section 3 reduces the occurrences of these trivial cycles sufficiently for practical purposes.

4.1 Anomalous Binary Curves

Now consider the situation created by using anomalous binary curves. If $E_{(a,b)}(GF(2^m))$ is such a curve, then $a, b \in \{0, 1\}$, so we are actually using subfield curves with $e = 1$ and $d = m$.

These curves are particularly well suited to this attack because the size of the space searched is reduced by a factor of $2m$, which reduces the expected running time by a factor of $\sqrt{2m}$. Thus the attacks on anomalous binary curves using this method are the most efficient among all subfield curves.

As an example, consider the anomalous binary curve $E_{(1,1)}(GF(2^{163}))$. This curve has been considered particularly attractive for implementing elliptic curve cryptosystems since its order is twice a prime close to 2^{162}. Many standards recommend that elliptic curve cryptosystems use curves divisible by a prime of at least 160 bits to obtain an expected attack time of at least 2^{80} operations [1, 2].

The conventional parallel collision search method for computing discrete logarithms on this curve is expected to take approximately 2^{81} operations. Using the improvements suggested above will reduce this expected running time by a factor of $\sqrt{2 \cdot 163}$ to approximately 2^{77} operations. This is below the required level of security imposed by the standards. Thus, this curve should not be used if a security level of 2^{80} is desired.

5 Efficiency Considerations

It has been shown that the number of group operations required to perform an elliptic curve logarithm can be reduced, but this is not much good if too much added computation is required in each step. In this section we show how to keep computation low. At each stage of the algorithm we know that the equation $Z_i = A_i P + B_i Q$ holds. We have at each stage that $A_{i+1} = \pm\lambda^j (A_i + c)$ (say) for some $0 \leq j \leq d-1$ and some multiplier c. If we represent A_i as $A_i = (-1)^{u_i}\lambda^{v_i} w_i$, $u_i \in \{0, 1\}$, $0 \leq v_i \leq d-1$, $0 \leq w_i \leq n-1$, then we can compute w_{i+1} as $w_i + (-1)^{u_i}\lambda^{-v_i} c$, v_{i+1} as $v_i + j$, and u_i is negated if necessary. The coefficient B_i can be tracked similarly. If there is a precomputed table of $\lambda^{-j} c$ for each $j = 0, \ldots, d-1$ and each multiplier c, then the computation on each step consists of additions or subtractions modulo n, additions modulo d and sign changes. This is much cheaper than an elliptic curve addition and is not a significant part of the algorithm run-time.

We have implemented these ideas on the anomalous binary curve $E_{(0,1)}(GF(2^{41}))$. The iteration function used 20 multipliers and used the look-ahead scheme described in Section 3. Over 15 trials, the experimental run-times were consistent with the expected run-time of $\sqrt{\frac{(\pi/2)2^{41}}{2 \cdot 41}}$.

6 Other Attempts for Faster Attacks

Another way that one might try to take advantage of the Frobenius endomorphism is to use parallel collision search as usual, but to check whether any stored distinguished points are negatives of each other or can be mapped to each other with the Frobenius endomorphism. This is easiest when using a method for choosing distinguished points which leaves a point distinguished if the Frobenius map is applied.

Unfortunately, this approach will not work unless the iteration function is carefully chosen so that all members of one equivalence class map to the same new equivalence class. The principle behind parallel collision search is that each distinguished point stands for the entire set of points in the trail leading to the distinguished point. A collision occurs because one trail runs into another trail and is lead to the common distinguished point. When a collision occurs and is detected, the two distinguished points are identical. The probability of encountering two unequal distinguished points which have a Frobenius map and/or negation map is very low.

Another way to think of this is that the iteration function acts as a random mapping and not all distinguished points are equally likely to appear. In fact, distinguished points tend to have radically different sized tree structures leading into them. The conditional probabilities are such that if a distinguished point occurs, it is very likely to have a large tree structure leading into it, making it a likely candidate to appear again. However, the distinguished points which are Frobenius and/or negation maps of the one which has occurred are not likely to have large tree structures.

It should be noted that the methods presented in Section 4 may also apply to any elliptic curve that has an easily computed automorphism that has small order modulo the order of the point. For example, consider the curve

$$y^2 = x^3 - ax$$

over $GF(p)$, with point $P = (x_0, y_0)$ of prime order n. This curve has complex multiplication by $\mathbb{Z}[i]$. Let i_p be a solution to $x^2 \equiv -1 \pmod{p}$. Then, $\psi_i(P) = (-x_0, i_p y_0) = \lambda_i P$ where λ_i is a solution to $x^2 \equiv -1 \pmod{n}$. Since

$$\begin{aligned} \psi_i^0(P) &= P \\ \psi_i^1(P) &= \lambda_i P \\ \psi_i^2(P) &= -P \\ \psi_i^3(P) &= -\lambda_i P \end{aligned}$$

are all distinct we can reduce the size of our search space by a factor of 4 to get a speed up of 2 over the general parallel collision search.

Also, consider the curve

$$y^2 = x^3 + b$$

over $GF(p)$, with point $P = (x_0, y_0)$ of prime order n. This curve has complex multiplication by $\mathbb{Z}[\omega]$ where ω is a cube root of unity. Let ω_p be a solution to $x^3 \equiv 1 \pmod{p}$. Then, $\psi_\omega(P) = (\omega_p x_0, y_0) = \lambda_\omega P$ where λ_ω is a solution to $x^3 \equiv 1 \pmod{n}$. Since

$$\begin{aligned} \pm\psi_\omega^0(P) &= \pm P \\ \pm\psi_\omega^1(P) &= \pm\lambda_\omega P \\ \pm\psi_\omega^2(P) &= \pm\lambda_\omega^2 P \end{aligned}$$

are all distinct we can reduce the size of our search space by a factor of 6 to get a speed up of $\sqrt{6}$ over the general parallel collision search.

7 Conclusion

Subfield and anomalous binary curves have been attractive to cryptographers for quite some time because of the efficiencies they provide both in curve generation and in the implementation of cryptographic algorithms. There have also been unsubstantiated warnings for quite some time that these curves may be more open to attack because of the greater structure that these curves have. The results of this paper show that this structure can in fact be used to obtain faster attacks. While the attack presented here still has a fully exponential running time, care should be exercised when choosing these curves regarding their expected security level. In certain circumstances these curves may still be attractive because of their efficiencies with respect to curves of similar security levels.

These results highlight the fact that more research must be done on the cryptanalysis of elliptic curve cryptosystems before we can be fully confident of the security level different curves offer. Two open questions remain:

- Can the ideas presented here be used, possibly in combination with other methods to reduce the attack time further?
- Can similar ideas be applied to other classes of curves or to curves whose coefficients do not lie in the subfield?

8 Acknowledgement

The authors would like to thank Niels Provos for pointing out the fact that the two curves mentioned in Section 6 also allow a speed up of the parallel collision search algorithm.

References

1. ANSI X9.62-199x: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), January 13, 1998.
2. ANSI X9.63-199x: Public Key Cryptography for the Financial Services Industry: Elliptic Curve Key Agreement and Transport Protocols, October 5, 1997.
3. E. De Win, A. Bosselaers, S. Vandenberghe, P. De Gersem and J. Vandewalle, "A fast software implementation for arithmetic operations in $GF(2^n)$," *Advances in Cryptology, Proc. Asiacrypt96, LNCS 1163*, K. Kim and T. Matsumoto, Eds., Springer-Verlag, 1996, pp. 65-76.

4. W. Diffie and M. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory* **22** (1976), pp. 644-654.
5. R. Gallant, R. Lambert and S. Vanstone, "Improving the Parallelized Pollard Lambda Search on Binary Anomalous Curves", Research Report No. CORR98-15, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada, (1998).
6. N. Koblitz, Elliptic curve cryptosystems, *Mathematics of Computation* **48** (1987), pp. 203-209.
7. N. Koblitz, "CM-curves with good cryptographic properties," *Advances in Cryptology, Proc. Crypto91, LNCS 576*, J. Feigenbaum, Ed., Springer-Verlag, 1997, pp. 279-287.
8. A. Menezes, T. Okamoto and S. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field, *IEEE Transactions on Information Theory*, **39** (1993), pp. 1639-1646.
9. V. Miller, "Uses of elliptic curves in cryptography," in *Advances in Cryptology - CRYPTO '85*, Lecture Notes in Computer Science, **218** (1986), Springer-Verlag, pp. 417-426.
10. National Institute for Standards and Technology, "Digital signature standard," *Federal information processing standard*, U.S. Department of Commerce, FIPS PUB 186, Washington, DC, 1994.
11. J.M. Pollard, Monte Carlo methods for index computation $\pmod p$, *Mathematics of Computation*, **32** (1978), pp. 918-924.
12. T. Satoh and K. Araki, Fermat Quotients and the Polynomial Time Discrete Log Algorithm for Anomalous Elliptic Curves, *preprint*, 1997.
13. R. Schroeppel, H. Orman, S. OMalley and O. Spatscheck, "Fast key exchange with elliptic curve systems," *Advances in Cryptology, Proc. Crypto95, LNCS 963*, D. Coppersmith, Ed., Springer-Verlag, 1995, pp. 43-56.
14. I. Semaev, Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p, *Mathematics of Computation*, **67** (1998), pp. 353-356.
15. N. Smart, The discrete logarithm problem on elliptic curves of trace one, *preprint*, 1997.
16. J. Solinas, "An improved algorithm for arithmetic on a family of elliptic curves," *Advances in Cryptology, Proc. Crypto97, LNCS 1294*, B. Kaliski, Ed., Springer-Verlag, 1997, pp. 357-371.
17. E. Teske, "Speeding up Pollard's rho method for computing discrete logarithms," Technical Report No. TI-1/98, Technische Hochschule Darmstadt, Darmstadt, Germany, (1998).
18. P. van Oorschot and M. Wiener, Parallel collision search with cryptanalytic applications, *Journal of Cryptology*, to appear.