I S E C 9 2 – 9

Construction of Elliptic Curves for Secure Cryptosystems

Atsuko Miyaji

June 18, 1992

Matsushita Electric Industrial Co. , LTD.
miyaji@isl.mei.co.jp

Abstract

Koblitz ([6]) and Miller ([8]) described how the group of points on an elliptic curve over a finite field can be used to construct public key cryptosystems. In this paper, we show how to construct elliptic curves for secure cryptosystems. The elliptic curve analog of the ElGamal signature scheme is also analyzed.

1 Introduction

To construct elliptic curves for secure cryptosystems, there are two ways to construct the elliptic curves. One way, by Beth, Schaefer([1]) and Koblitz([7]), is to construct E/F_{2^r} where the extension degree of a finite field, in which EDLP is reduced to DLP by the MOV reduction ([11]), is sufficiently large. The other way, by the author, is to construct E/F_p (p is a large prime) which makes any embedding to DLP, including the MOV reduction inapplicable ([9, 10]). The purpose of this paper is to show the construction time and application of such elliptic curves for cryptosystems.

The results of this paper are following.

- We show what determines the running time of the algorithm to construct an elliptic curve E/F_p and give an approximate formula of the running time.
- Implementation of the algorithm on a 32 bit personal computer shows that we can construct elliptic curves with 30 digits $\sim$ 70 digits prime elements in a practical time.
- The elliptic curve analog of the ElGamal signature scheme shows that the elliptic curve cryptosystems is more efficient than finite field cryptosystems.

Section 2 summarizes how to construct such an elliptic curve ([10]). Section 3 shows the experimental results for constructing elliptic curves. Section 4 studies the elliptic curve analog of the ElGamal signature scheme.

Notation

p : a prime
r : a positive integer
q : a power of p
$\#A$: the cardinality of a set A
$o(t)$: the order of an element t of a group
F_q : a finite field with q elements
K : a field (including a finite field)
$ch(K)$: the characteristic of a field K
E : an elliptic curve
If we remark a field of definition K of E , we write E/K.
$\mathcal{O}$: the identity element of E
$E(K)$: the set of K-rational points on the elliptic curve E

2 Construction of elliptic curves

This section summarizes how to construct the elliptic curves that make the embedding to DLP, including the MOV reduction, inapplicable ([10]). Such an elliptic curve is given as E/F_p with p elements. There are two phases for the construction of E/F_p with p elements.

The first phase is, for a given d, to find a prime p such that $4p = 1 + b^2 d$ (b is an integer). It is equivalent to find a prime p such that $p = db^2 + db + \frac{d+1}{4}$ (b is an integer). We use only $d \in \{11, 19, 43, 67, 163\}$. Given an integer d, the j-invariant of E/F_p with p elements, where p satisfies $p = db^2 + db + \frac{d+1}{4}$, is uniquely determined. It doesnot depend on p. Table 1 lists integers d and the j-invariant.

In the case of $d = 11$, an algorithm to find p more than $N(\geq 10^{29})$ is as follows.

Algorithm 1 .

1. *Let b be the minimal integer satisfying the next two conditions:*

   Condition1: $b \geq \lceil(-1 + \sqrt{\frac{4*N-1}{11}})/2\rceil$;

   Condition2: $b \equiv 4, 7, 10 \pmod{15}$.

   Then goto 2.

2. *Test the primality of $p = 11b^2 + 11b + 3$. If p is prime, then stop. If p isn't prime, then take the next smallest b satisfying the condition 2 and repeat 2.*

This algorithm tests the primality of p one by one not for all b but for b that satisfies condition 2. We denote the reduction ratio of the number of the primality tests by ε_d. The condition2 differs for each integer d. Table 1 shows the condition of b to test and ε_d for each d.

The second phase is, for a given prime p and j-invariant, to make E/F_p with p elements. Here we denote $(\frac{c}{p})$ as following:

$$(\frac{c}{p}) = \begin{cases} 1 & \text{if } c \text{ is a quadratic residue} \pmod p \\ -1 & \text{if } c \text{ is a non-quadratic residue} \pmod p \end{cases}.$$

An algorithm to make E/F_p with p elements is following.

Algorithm 2 .

1. *Find* $x_0 \in F_p$ *such that* $y_0 = x_0^3 + 3\frac{j}{1728-j}x_0 + 2\frac{j}{1728-j}$ *is a quadratic residue. Then* $X_{y_0} = (y_0x_0, y_0^2) \in E_{y_0}(F_p)$.

2. *Calculate* pX_{y_0}. *If* $pX_{y_0} = \mathcal{O}$, *then* $E_{y_0}(F_p)$ *has exactly* p *elements. If not, choose* t *such that* $(\frac{t}{p}) = -1$ *and find* $x_1 \in F_p$ *such that* $y_1 = x_1^3 + 3t^2\frac{j}{1728-j}x_1 + 2t^3\frac{j}{1728-j}$ *is a quadratic residue. Then* $E_{y_1t}(F_p) \ni (y_1x_1, y_1^2)$ *and* $E_{y_1t}(F_p)$ *has just* p *elements.*

With calculation of the Jacobi symbols we can easily determine whether an element is a quadratic residue or not. For the Jacobi symbol, see [5]. So we can construct E/F_p only by 4 calculations of the Jacobi symbol on the average plus a calculation of a p-fold element of E/F_p. Therefore the running time of this algorithm is insignificant compared to the time of Algorithm 1.

3 The running time of Algorithm 1

First we define the density $Dens(d, x)$ of primes represented by $db^2 + db + \frac{d+1}{4}$ for a given integer d,

$$Dens(d, x) = \frac{\pi_d(2x) - \pi_d(x)}{x}, \text{ where}$$
$$\pi_d(x) = \#\{b < x | p = db^2 + db + \tfrac{d+1}{4} \text{ is prime } \}.$$

Algorithm 1 reduces the number of tests with a factor of ε_d effectively. We define the effective density $Dens'(d, x)$,

$$Dens'(d, x) = Dens(d, x)/\varepsilon_d .$$

We further define $T_1(x)$ as the average time to judge that an integer $t \approx x^2$ is prime and $T_2(x)$ as the average time to judge that an integer $t \approx x^2$ isn't prime. $T_1(x)$ and $T_2(x)$ depend on a machine and the type of the primality test, deterministic or probabilistic. Now

we can write the approximate time *Aptime* $1(x)$ to find a prime $p = db^2 + db + \frac{d+1}{4} (b \geq x)$ with Algorithm 1 approximately as

$$Aptime\ 1(x) = (\frac{1}{Dens'(d,x)} - 1)T_2(x) + T_1(x). \quad (1)$$

The experimental results in section 4 will show this is a good approximation. The approximate formula (1) shows the running time of Algorithm 1 depends on the density and the time for primality test. It also shows how the reduction ratio ε_d contributes to reduce the time to find p.

There is a conjecture that there are infinitely many primes $p = db^2 + db + \frac{d+1}{4}$ ([4]). If there is a density function of primes $p = db^2 + db + \frac{d+1}{4}$, we will be able to calculate *Aptime* $1(x)$ with the density function in the same way.

As we mentioned in Section 2, the time to construct elliptic curves mainly consists of the time to implement the Algorithm 1. So the above results show it depends on the density $Dens'(d,x)$ and the time for primality test. Section 4 gives concrete examples of the running time.

4 Implementation of Algorithm 1 and 2

We implemented Algorithm 1 and 2 to construct E/F_p with p elements where p is a prime from 30 digits to 70 digits. Then up to the present, EDLP on such an elliptic curve is secure for all considerable attacks. We wrote a program in UBASIC that runs on a 32bit personal computer (20MHz). We used the deterministic primality test. The experimental results will show that such condition is enough practical for the construction of such elliptic curves.

4.1 Approximate time and running time

Let us construct elliptic curves over 30 digits prime fields one by one from the smallest. In the following, let d be an integer of $\{11, 19, 43, 67, 163\}$ and $N = 10^{29}$. First we show the result of implementation Algorithm 1. We use Cohen-Lenstra method([2], [3]) which is a deterministic primality test. The values $T_1(\sqrt{N})$ and $T_2(\sqrt{N})$ to represent the performance of the primality test on the machine are obtained experimentally as,

$$T_1(\sqrt{N}) = 29.26sec \text{ and } T_2(\sqrt{N}) = 2.92sec.$$

We check the primality of the numbers $p = db^2 + db + \frac{d+1}{4}$ for each d . The integer b is chosen out of the following three intervals

$$\begin{aligned} I_{d,N,1} &= [x_{d,N}, x_{d,N} + L] \\ I_{d,N,2} &= [\frac{3}{2}x_{d,N}, \frac{3}{2}x_{d,N} + L] \\ I_{d,N,3} &= [2 * x_{d,N} - L, 2 * x_{d,N}] \end{aligned} \quad (2)$$

where $L = 5 * 10^3$ is a condition to stop and $x_{d,N}$ is set to $\lceil(-1 + \sqrt{\frac{4*N-1}{d}})/2\rceil$. The column (b) of Table 2 shows the average execution time to find a prime for the b in three range of (2).

Next we investigate the relation between *Aptime* $1(x)$ and the actual execution time. The density $Dens'(d, x_{d,N})$ is shown in the column (a) of Table 2. It is obtained approximately as the average of each density that $p = db^2 + db + \frac{d+1}{4}$ is prime for b of the above three intervals of (2). Each density of the three intervals are found to be nearly equal. The column (e) of Table 2 shows the approximate time *Aptime* $1(x_{d,N})$ obtained by the above $Dens'(d, x_{d,N}), T_1(N)$ and $T_2(N)$. We see that the column (e) of Table 2 is nearly equal to the execution time of the column (b) of Table 2.

Next we construct $E/F_p, E(F_p) \ni X$ for all primes p given by the previous experiment with Algorithm 2. The column (c) of Table 2 presents the running time for the Algorithm 2. We see the running time for the Algorithm 2 is insignificant compared to the Algorithm 1. As we described in section 2, we can construct E/F_p only by 4 calculations of the Jacobi symbol on the average plus a calculation of a p-fold element of E/F_p for a given p. This means the running time depends only on the order of p. So the running time for each d is nearly equal (the column (c) of Table 2). We list the total average time for Algorithm 1 and 2 in the column (d) of Table 2. We see the running time to make E/F_p with p elements nearly equals *Aptime* $1(x_{d,N})$. We also see we can make an elliptic curve for secure cryptosystem, where p is 30 digits prime, in $59.9 \sim 83.2$ seconds on a 32bit personal computer(20 MHz).

Finally we list the experimental results for $40 \sim 70$ digits. They were done in the same way as the above case of 30 digits. Table 3 shows the experimental results of 40 digits. Table 4 shows the experimental results of 50 digits. Table 5 shows the experimental results of 60 digits. Table 6 shows the experimental results of 70 digits. Table 7 shows the values $T_1(\sqrt{N})$ and $T_2(\sqrt{N})$ obtained experimentally for each N. In these cases, we see that *Aptime* $1(x)$ is also a good approximation of Algorithm 1 and the running time for the Algorithm 2 is insignificant compared to the Algorithm 1.

4.2 Comparison between the ordinary p and $p = db^2 + db + \frac{d+1}{4}$

Finding a prime p of the form of $p = db^2 + db + \frac{d+1}{4}$ is more difficult than finding an ordinary prime p? We know that the more primes exist in a fixed search range, the less the necessary search time is. So we compare the number of the primes $p = db^2 + db + \frac{d+1}{4}$ and the ordinary p for the first $5 * 10^3$ range in 30 digits $\sim$ 70 digits. The search range is as following,

$$\begin{array}{ll} I_{d,N,1} & \text{for } p = db^2 + db + \frac{d+1}{4} \\ I_{N,1} = [N, N + L] & \text{for ordinary } p \end{array}$$

where $L = 5 * 10^3, N = 10^{29}, 10^{39}, 10^{49}, 10^{59}, 10^{69}$. Table 8 shows the results. We see the number of prime $p = db^2 + db + \frac{d+1}{4}$ for $d = 19, 43, 67, 163$ is always more than an ordinary prime in the same digit. Practically we don't test the primality of number divisible by 2 to find an ordinary prime. So we can't compare the necessary search time from Table 8

immediately. If we search an ordinary prime except the number divisible by 2,3 and 5 and a prime $p = db^2 + db + \frac{d+1}{4}$ for $\forall b \in I_{d,N,1}$, then we can compare $\frac{15}{4}$ times of the number of ordinary prime in Table 8 with the number of prime $p = db^2 + db + \frac{d+1}{4}$. In such condition, we see that searching a prime p of $p = 163b^2 + 163b + 41$ is faster than searching the ordinary prime p.

4.3 Further discussion

We have seen the running time to find p nearly equals $Aptime\ 1(x_{d,N})$. This fact leads two things. One is that the higher the density $Dens'(d, x)$ is, the faster the time to find p is. The column (a) and (b) of Table 2 show this. The other is that $Dens'(d, x)$ and the primality test determine the running time to find p, where p is nearly equal to x^2. Figure 1 shows the graph of change of $Dens'(d, x_{d,N})$. We see that the order of $Dens'(d, x_{d,N})$ doesn't change in all digits. We can make E/F_p with p elements fastest in the case of $d = 163$.

5 ElGamal signature scheme

We discribe one application of elliptic curve cryptosystems. Let E/F_q be an elliptic curve, and let $P \in E(F_q)$ be a publicly known point. We denote $o(P)$ by l. User A randomly chooses an integer a , a secret key, and make public the point $P_a = aP$ as a public key. We assume a message $m \in \{m|0 < m < l\}$. User A sends the messeage m to a user B with her or his signature of m. The procedure for A to make a signature (R, s) is as follows.

Procedure for signature generation

1 Pick a random number $k \in \{1, ..., l\}$ and compute

$$R = kP = (r_x, r_y). \tag{3}$$

Here r_x is a $x-$coordinate of R and r_y is a $y-$coordinate of R.

2 Compute s satisfying that

$$s * k \equiv (m - a * r_x) \pmod{l} \tag{4}$$

and output signature (R, s) to B.

The procedure for B to verify a signature (R, s) is as follows.

Procedure for signature verificatoin

1 Compute mP and $r_x P_a + sR$ and check that they are equal as an element of elliptic curve E.

2 A signature (R, s) is considered to be valid if it withstands the signature.

A signature generated according to the protocol is always valid since

$$\begin{aligned} r_x P_a + s * R &= r_x P_a + s * kP \\ &= r_x P_a + (m - a * r_x)P \\ &= mP . \end{aligned}$$

The security of the ElGamal signature scheme depends on EDLP on E/F_p. Suppose an intruder tries to get a or k with the knowledge of R, m, s, P_a and P. Then from the equation 4, he gets k if and only if he gets a. So he must solve EDLP on E/F_q.

Using E/F_p of section 2, where p is more than 30 digit (97 bit) and

$$E : y^2 = x^3 + ax + b \ \ (a, b \in F_p),$$

as the above elliptic curve, we have next advantages.

- **The length of signatures**

Given the $x-$coordinate r_x of $R \in E(F_P)$, we can determine the $y-$coordinate r_y of R except the sign as following,

$$r_y = \pm\sqrt{r_x{}^3 + ar_x + b}.$$

So we transmit only $(r_x, s, sign(r_y))$ instead of (R, s). The calculation of square root is easy. So user B can restore R easily. The length of signature is about 195 bit, that is about $\frac{2}{3}$ times as long as the length of signature (R, s). This idea is used only when we construct an elliptic curve over F_p ($p \geq 5$). If we construct E over F_{2^r} ([1, 7]), we cannot determine the y-coordinate from the x-coordinate of an element easily.

- **The computation complexity of generating the signature**

The elliptic curve that we've constructed in Section 4 has a good property that the order l of P equals to p. So both the euation 3 and 4 require computation in the residue of p. If we use a pre-computation table for the necessary residue computation, the computation will be more quickly done. By the good property, we need only the residue computation of one prime p. So we have only to use a remainder table of p, the computation of generating signature will be more quickly done.

6 Conclusion

- We have shown that the density $Dens'(d, x_{d,N})$ and the time for primality test determines the running time of the algorithm to construct an elliptic curve E/F_p. We have also given an approximate formula of the running time.
- We have shown that we can construct E/F_p with p elements in a practical time with the deterministic primality test on a 32 bit personal computer.
- We have shown the elliptic curve analog of the ElGamal signature scheme.

d	j	condition of b	ε_d
11	$(-2^5)^3$	$b \equiv 4, 7, 10 \pmod{15}$	1/5
19	$(-2^5 * 3)^3$	$b \equiv 1, 2, 3 \pmod{5}$	3/5
43	$(-2^6 * 3 * 5)^3$	$b \equiv 1, 2, 3, 4, 5, 6, 7, 8, 9 \pmod{11}$	9/11
67	$(-2^5 * 3 * 5 * 11)^3$	all b	1
163	$(-2^6 * 3 * 5 * 23 * 29)^3$	all b	1

Table 1: Integers d and j-invariant and the condition of b

d	(a) $Dens'(d, x_{d,N})$	(b) Average time of algorithm 1	(c) Average time of algorithm 2	(d) Total average time of algorithm 1 and 2	(e) Approximate time $Time1(x_{d,N})$
11	0.0820	62.4	2.81	65.21	61.9
19	0.0538	80.16	3.03	83.19	80.6
43	0.0613	73.56	2.99	76.55	74.0
67	0.0645	72.06	2.85	74.91	71.6
163	0.0981	56.94	2.99	59.93	56.1

Table 2: Running time and density for 30 dgits (in seconds)

d	(a) $Dens'(d, x_{d,N})$	(b) Average time of algorithm 1	(c) Average time of algorithm 2	(d) Total average time of algorithm 1 and 2	(e) Approximate time $Time1(x_{d,N})$
11	0.064	105.72	4.97	110.69	105.24
19	0.037	137.28	5.43	142.71	138.97
43	0.0449	124.44	5.44	129.88	124.90
67	0.0452	124.08	5.13	129.21	124.46
163	0.0737	100.2	5.33	105.33	99.16

Table 3: Running time and density for 40 dgits (in seconds)

d	(a) $Dens'(d, x_{d,N})$	(b) Average time of algorithm 1	(c) Average time of algorithm 2	(d) Total average time of algorithm 1 and 2	(e) Approximate time $Time1(x_{d,N})$
11	0.0533	163.40	7.90	171.3	163.38
19	0.0257	220.18	8.23	228.41	224.43
43	0.0361	188.92	8.43	197.35	190.46
67	0.0373	186.20	7.93	194.13	187.76
163	0.0592	158.99	8.61	167.6	157.71

Table 4: Running time and density for 50 dgits (in seconds)

d	(a) $Dens'(d, x_{d,N})$	(b) Average time of algorithm 1	(c) Average time of algorithm 2	(d) Total average time of algorithm 1 and 2	(e) Approximate time $Time1(x_{d,N})$
11	0.0397	258.76	11.57	270.33	259.29
19	0.025	298.96	12.04	311	304.79
43	0.0299	281.21	11.83	293.04	284.65
67	0.0301	281.29	12.02	293.31	283.97
163	0.0499	243.70	11.97	255.67	243.47

Table 5: Running time and density for 60 dgits (in seconds)

d	(a) $Dens'(d, x_{d,N})$	(b) Average time of algorithm 1	(c) Average time of algorithm 2	(d) Total average time of algorithm 1 and 2	(e) Approximate time $Time1(x_{d,N})$
11	0.036	349.10	16.8	365.9	351.17
19	0.02	409.39	17.04	426.43	421.17
43	0.0249	381.9	16.57	398.47	390.18
67	0.0266	373.83	16.64	390.47	382.09
163	0.0431	333.76	17.39	351.15	336.76

Table 6: Running time and density for 70 dgits (in seconds)

digit	$T_1(\sqrt{N})$	$T_2(\sqrt{N})$
30	29.26	2.92
40	61.98	2.96
50	109.56	3.03
60	184.98	3.07
70	266.82	3.15

Table 7: $T_1(\sqrt{N})$ and $T_2(\sqrt{N})$ (in seconds)

digit	prime for $d = 11$	prime for $d = 19$	prime for $d = 43$	prime for $d = 67$	prime for $d = 167$	ordinary prime
30	91	166	246	323	512	89
40	75	103	179	230	377	65
50	53	75	126	188	319	40
60	35	73	122	133	233	36
70	35	57	100	140	220	40

Table 8: Comparison of the number of primes

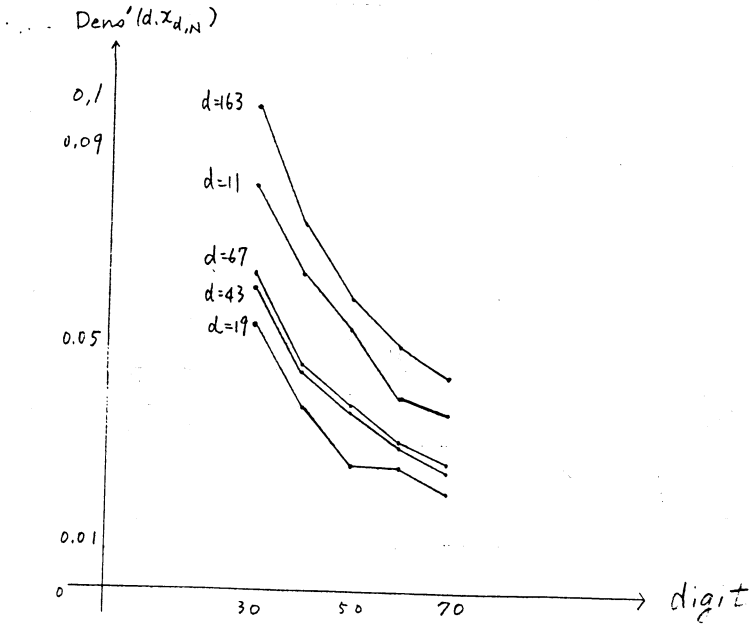

Figure 1

7 Acknowledgements

I wish to thank Makoto Tatebayashi for his helpful advice. I am grateful to all other members of my group for their kind help.

References

[1] T. Beth and F. Schaefer *"Non supersingular elliptic curves for public key cryptosystems", Abstracts for Eurocrypto 91 , Brighton, U.K. 155-159.*

[2] H. Cohen and H. W. Lenstra Jr. *"Primality testing and jacobi sums", Mathematics of computation 42(1984), 297-330.*

[3] H. Cohen and H. W. Lenstra Jr. *"Implementation of a new primality test", Mathematics of computation 48(1987), 103-121.*

[4] G. Hardy and E. Wright *"An Introduction to the Theory of Numbers", Oxford Univ. Press, 1960.*

[5] K. Ireland and M. Rosen *"A classical introduction to modern number theory", GTM 84, Springer-Verlag, New-York, 1982.*

[6] N. Koblitz *"Elliptic curve cryptosystems", Math. Comp. 48(1987), 203-209.*

[7] N. Koblitz *"CM-curves with good cryptographic properties", Proceedings of Crypto'91, to appear.*

[8] V. S. Miller *"Use of elliptic curves in cryptography", Advances in Cryptology-Proceedings of Crypto'85, Lecture Notes in Computer Science, 218 (1986), Springer-Verlag, 417-426.*

[9] A. Miyaji *"On ordinary elliptic curves", Abstract of proceedings of Asiacrypto'91, 1991.*

[10] A. Miyaji *"On the implementation of ordinary elliptic curve cryptosystems", SCIS92-2A, 1992.*

[11] A. Menezes, S. Vanstone and T. Okamoto *"Reducing elliptic curve logarithms to logarithms in a finite field", to appear in Proc. STOC'91.*