A comment on "Comparing the MOV and FR reductions in elliptic curve cryptography" from EUROCRYPT'99

Qiping Lin, and Fengmei Liu
linqp@126.com

Data Communication Science and Technology Research Institute, Beijing 100191, China

Abstract. In general the discrete logarithm problem is a hard problem in the elliptic curve cryptography, and the best known solving algorithm have exponential running time. But there exists a class of curves, i.e. supersingular elliptic curves, whose discrete logarithm problem has a subexponential solving algorithm called the MOV attack. In 1999, the cost of the MOV reduction is still computationally expensive due to the power of computers. We analysis the cost of the MOV reduction and the discrete logarithm problem of the curves in [2] using Magma with an ordinary computer.

1 A wrong curve

We recall the example 4 of [2] in the following.

Example 4(Supersingular-EC) Suppose that the curve $E/F_p : y^2 = x^3+ax+b$, the base point $P = (x_0, y_0) \in E(F_p)$, the order n of P, and a point $R = [l]P = (x_1, y_1)$ are given as follows:

$p = 10202130657668293802865103277946942060930683196983$ (binary 163-bits, $p+1 = 2^3 \times 3^3 \times 59 \times 113 \times 7084458733777404048453899025845195282548847$),

$a = 1, b = 0$

$n = 7084458733777404048453899025845195282548847$,

$x_0 = 6361408431660145018472734964469918949727993631117$,

$y_0 = 2224285726125163515264642109319596318772261492 91$,

$x_1 = 1791400202383882094094972648523798358242766050148$,

$y_1 = 6662282879825452479945554028296857282243572635001$.

$T = (x_3, y_3)$

$x_3 = 338530611385145171186893854505822118617259793743 6,$

$y_3 = 498677065440695353174518618475802696104861959899 2.$

Using Magma [1], we found that $P = (x_0, y_0)$, $R = (x_1, y_1)$ and $T = (x_3, y_3)$ are not points on $E/F_p : y^2 = x^3 + ax$ in example 4.

```
> p:=1020213065766829380286510327794694206093068319698 3;
> K:=GF(p);
> a:=1;
> b:=0;
> E:=EllipticCurve([K!a,b]);
> x0:=6361408431660145018472734964469918949727993631117;
> y0:=2224285726125163515264642109319596318772261492 91;
> x1:=1791400202383882094094972648523798358242766050148;
> y1:=6662282879825452479945554028296857282243572635001;
> x3:=3385306113851451711868938545058221186172597937436;
> y3:=4986770654406953531745186184758026961048619598992;
> P:=E![x0,y0];

>> P:=E![x0,y0];
       ^

Runtime error in '!': Illegal coercion
> Q:=E![x1,y1];

>> Q:=E![x1,y1];
       ^

Runtime error in '!': Illegal coercion
> T:=E![x3,y3];

>> T:=E![x3,y3];
       ^

Runtime error in '!': Illegal coercion
```

2 The corrected curve

And we tried to find out the right elliptic curve function $E/F_p : y^2 = x^3 + ax + b$. First, we assumed points P and Q were correct. Then we inserted P and Q into the function $E/F_p : y^2 = x^3 + ax + b$.

$$a * x_0 + b = y_0^2 - x_0^3 \ (mod\ p)$$
$$a * x_1 + b = y_1^2 - x_1^3 \ (mod\ p)$$

And we got the parameters a and b as follows:

$a = 1020213065766829380286510327794694206093068319698 2 \equiv -1\ mod\ p$,

$b = 0\ mod\ p$.

We verified the new elliptic curve function $E/F_p : y^2 = x^3 - x$ with point T.

```
> p:=10202130657668293802865103277946942060930683196983;
> K:=GF(p);
> a:=-1;
> b:=0;
> E:=EllipticCurve([K!a,b]);
> x3:=33853061138514517118689385450582211861725979 37436;
> y3:=49867706544069535317451861847580269610486195 98992;
> T:=E![x3,y3];
>
```

Furthermore, we verified that the order of the new elliptic curve was the same as describing in example 4. And the order of P equals the one of Q.

```
> p:=10202130657668293802865103277946942060930683196983;
> K:=GF(p);
> a:=-1;
> b:=0;
> E:=EllipticCurve([K!a,b]);
> x0:=63614084316601450184727349644699189497279936 31117;
> y0:=22242857261251635152646421093195963187722614 9291;
```

```
> x1:=1791400202383882094094972648523798358242766050148;
```

3 The running time of ECDLP of the curves in [2]

Table 1. The time of computation in Example 1-4($\log q$ and k are the binary size of the definition field and the necessary minimum extension degree, respectively.)[2]

Type	$\log q$	k	Running time(sec)	
Example 1	46	1	RF reduction	419
Example 2	108	1	RF reduction	4105
Example 3	46	2	RF reduction	999
			MOV reduction	1872
Example 4	164	2	RF reduction	161467
			MOV reduction	282426

Comparing with table 1, we can see that the running time of the FR reduction (or MOV reduction) is negligible now. However, the cost of solving discrete logarithm problem is still expensive (the Magma codes are in the appendix).

In 1993, Menezes et al. [3] introduced the MOV attack to reduce the elliptic curve discrete logarithm problem to a discrete logarithm problem in a finite field by using the Weil pairing. And then the discrete logarithm problem in the finite field can be solved efficiently by means of Index Calculus methods. However, we still can't solve the ECDLP of example 4 in [2] in a short time.

Table 2. The cost of the FR reduction (or the MOV reduction) and ECDLP of curves in example 1-4 of [2](see the detail in appendix)

Type	$\log q$	k	Running time(sec) [2]		Running time(sec) now
Example 1	46	1	RF reduction	419	negligible
			DLP	-	DLP 2.592
Example 2	108	1	RF reduction	4105	negligible
			DLP	-	DLP 4.805
Example 3	46	2	RF reduction	999	negligible
			MOV reduction	1872	
			DLP	-	DLP 0.029
Example 4	164	2	RF reduction	161467	negligible
			MOV reduction	282426	
			DLP	-	DLP -

References

1. Wieb Bosma, John Cannon, and Catherine Playoust. MAGMA. The Computational Algebra Group, Sydney, 2007. available from http://magma.maths.usyd.edu.au/.
2. Ryuichi Harasawa, Junji Shikata, Joe Suzuki, and Hideki Imai. Comparing the MOV and FR reductions in elliptic curve cryptography. EUROCRYPT'99, LNCS 1592, pp. 190-205, 1999.
3. Alfred J. Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Transactions on information Theory, 39(5):1639-1646, 1993.

A The running time of the discrete logarithm of example 1-3 in [2]

The following is the running time of discrete logarithm of example 1 in [2] using Magma.

```
> p:=23305425500899;
> F<t>:=GF(p);
> E:=EllipticCurve([F!13079575536215,951241857177]);
> P:=E![17662927853004,1766549410280];
> Q:=E![2072411881257,5560421985272];
> N:=Order(P);
> T:=Random(E);
```

```
> M:=Order(T);
> d:=GCD(M,N);
> T:=Floor(M/d)*T;
> a:=WeilPairing(P,T,N);
> b:=WeilPairing(Q,T,N);
> t1:=Realtime();
> Log(a,b);
709658
> t2:=Realtime();
> t3:=t2-t1;
> t3;
2.592
```

The following is the running time of discrete logarithm of example 2 in [2] using Magma.

```
> p:=93340306032025588917032364977153;
> F<t>:=GF(p);
> E:=EllipticCurve([F!71235469403697021051902688366816,4749031293579801403
4601792244544]);
> P:=E![10362409929965041614317835692463,79529049191468905652172306035573];
> Q:=E![15411349585423321468944221089888,94160529078832780887823358830033];
> N:=Order(P);
> T:=Random(E);
> M:=Order(T);
> d:=GCD(M,N);
> T:=Floor(M/d)*T;
> a:=WeilPairing(P,T,N);
> b:=WeilPairing(Q,T,N);
> t1:=Realtime();
> Log(a,b);
764009
```

```
> t2:=Realtime();
> t3:=t2-t1;
> t3;
4.805
```

The following is the running time of discrete logarithm of example 3 in [2] using Magma.

```
> p:=23305425500899;
> K:=GF(p);
> F<t>:=GF(p^2);
> E:=EllipticCurve([F!1,0]);
> P:=E![18414716422748,9607997424906];
> Q:=E![22829488331658,15463570264423];
> N:=Order(P);
> T:=Random(E);
> M:=Order(T);
> d:=GCD(M,N);
> T:=Floor(M/d)*T;
> a:=WeilPairing(P,T,N);
> b:=WeilPairing(Q,T,N);
> t1:=Realtime();
> Log(a,b);
4500974
> t2:=Realtime();
> t3:=t2-t1;
> t3;
0.029
```