SAC 2013 – SFU Burnaby
16th August 2013

Faster Repeated Doublings on Binary Elliptic Curves

C. Doche and D. Sutantyo

Macquarie University

Sydney Australia

Main Contributions

Faster scalar multiplication in Char. 2

Relies on a new way to exploit x-coordinate doublings

Useful for network with limited bandwidth

Window-NAF precomputations in Char. 2 with a single inversion

Generalization of the work of Dahmen et al.

Elliptic Curve

We consider an ordinary elliptic curve defined in Weierstraß form by

$$E : y^2 + xy = x^3 + a_2x^2 + a_6$$

with $a_2 \in \{0, 1\}$ and $a_6 \in \mathbb{F}_{2^d}^*$

Rational Points

A point $P = (x_1, y_1)$ satisfying

$$y_1^2 + x_1 y_1 = x_1^3 + a_2 x_1^2 + a_6$$

is an affine point on E

Together with P_∞, the set of all the affine points with coordinates in $\mathbb{F}_{2^d}$, is the set of rational points $E(\mathbb{F}_{2^d})$

Addition Law

The set $E(\mathbb{F}_{2^d})$ can be endowed with an abelian group structure

The group law is denoted by $+$

A doubling corresponds to $P + P = [2]P$

Scalar Multiplication

Given an integer n and a point P on a curve, the scalar multiplication $[n]P$ consists in computing

$$[n]P = \underbrace{P + \cdots + P}_{n \text{ times}}$$

It is the core operation in many ECC protocols

Scalar Multiplication

There is a vast literature devoted to scalar multiplication

Focus on different aspects:

– Exponentiation techniques

– Arithmetic on the elliptic curve

Doubling

Let $P = (x_1, y_1)$

The coordinates x_2 and y_2 of $[2]P$ satisfy

$$x_2 = \lambda^2 + \lambda + a_2$$

$$y_2 = \lambda(x_1 + x_2) + x_2 + y_1$$

with $\lambda = x_1 + \dfrac{y_1}{x_1}$

Complexity to compute $[2]P$: $\mathrm{I} + 2\mathrm{M} + \mathrm{S}$

LD Representation

The equation

$$Y^2 + XYZ = X^3Z + a_2X^2Z^2 + a_6Z^4$$

is an homogenized version of E

A point is represented by the triple $(X_1 : Y_1 : Z_1)$
in projective-like López–Dahab coordinates

LD Representation

When $Z_1 \neq 0$, $(X_1 : Y_1 : Z_1)$ corresponds to the affine point $(X_1/Z_1, Y_1/Z_1^2)$

The point $(X_1 : Y_1 : 0)$ is the point at infinity

LD Arithmetic

Doubling: $4\mathrm{M} + 5\mathrm{S}$

Addition: $13\mathrm{M} + 4\mathrm{S}$

Mixed Addition: $8\mathrm{M} + 5\mathrm{S}$

Formulas and much more on the EFD

`http://www.hyperelliptic.org/EFD/`

Doubling

There is something remarkable about doublings

The x-coordinate of $[2]P$ only depends on the x-ccordinate of P

In affine coordinates $$x_2 = \frac{x_1^4 + a_6}{x_1^2}$$

In LD coordinates $$X_2 = X_1^4 + a_6 Z_1^4$$

$$Z_2 = X_1^2 Z_1^2$$

Montgomery's Method

Montgomery develops an x-coordinate arithmetic system by noting that

The x-coordinate of $P + Q$ only depends on the x-coordinates of P, of Q, and of $P - Q$

Montgomery's ladder computes $[n]P$ with an addition and a doubling at each step

The complexity of the scheme is $(6\mathrm{M} + 4\mathrm{S})\ell$

x-doubling

We call this operation

$$X_2 = X_1^4 + a_6 Z_1^4$$

$$Z_2 = X_1^2 Z_1^2$$

an x-doubling

Its complexity is $2\mathrm{M} + 3\mathrm{S}$

We propose another way to use it

x-doubling

Let us perform k successive x-doublings

We end up with X_{2^k} and Z_{2^k} such that X_{2^k}/Z_{2^k} is equal to the x-coordinate of the affine point $[2^k]P$

We don't know the y-coordinate but $[2^k]P$ is on the curve

So we just need to solve a quadratic equation

x-doubling

Fix $Q_0 = P$ and $Q_1 = [2^k]P$

Write n as $n = 2^k n_1 + n_0$

We can compute

$$[n]P = [n_1]Q_1 + [n_0]Q_0$$

using the JSF

We call this approach x-JSF

x-JSF

If we precompute $Q_1 + Q_0$ and $Q_1 - Q_0$

(can be done simultaneously with $\mathrm{I} + 4\mathrm{M} + 2\mathrm{S}$)

We then need: $\frac{\ell}{2}$ x-doublings

$\frac{\ell}{2}$ LD regular doublings

$\frac{\ell}{4}$ LD mixed additions

Solving the Quadratic Equation

Consider the equation

$$T^2 + TX_{2^k}Z_{2^k} = X_{2^k}^3 Z_{2^k} + a_2 X_{2^k}^2 Z_{2^k}^2 + a_6 Z_{2^k}^4$$

It has two solutions: Y_{2^k} and $Y_{2^k} + X_{2^k}Z_{2^k}$

Corresponding to the points $[2^k]P$ and $-[2^k]P$

How do we select the right root?

Solving the Quadratic Equation

The good news is that we just need one bit to decide

Namely, the last bit of y_{2^k}/x_{2^k} allows to identify the correct root

The bad news is that it does not seem possible to compute this last bit faster than computing all the bits of y_{2^k}/x_{2^k}

Solving the Quadratic Equation

If we cannot compute this last bit ...

Maybe this bit can be made available with the point itself

The point P must be known in advance, it cannot be a random point

Solving the Quadratic Equation

If the point P is known in advance, why not precomputing and making $[2^k]P$ available in full rather than just one bit?

We could, we could even precompute more points of the form $[2^k]P$

But this represents a lot of information to transmit

Our scheme just needs one extra bit per extra point

Solving the Quadratic Equation

If the bandwidth is limited, there is an advantage in using our approach

The owner of P makes the x-coordinate of P, i.e. x_1, available as well as the last bits of x_1/y_1 and of x_{2^k}/y_{2^k}

Solving the Quadratic Equation

We need $\mathrm{I} + 2\mathrm{H} + 12\mathrm{M} + 4\mathrm{S}$ to form the equations

$$T^2 + TX_1Z_1 = X_1^3Z_1 + a_2X_1^2Z_12 + a_6Z_1^4$$

and

$$T^2 + TX_{2^k}Z_{2^k} = X_{2^k}^3Z_{2^k} + a_2X_{2^k}^2Z_{2^k}^2 + a_6Z_{2^k}^4$$

and solve them

H is the complexity of the Half-Trace function

In practice, $\mathrm{H} \sim \mathrm{M}$

Solving the Quadratic Equation

More generally, once the X and Z coordinates of Q_0, Q_1, $\ldots$, Q_{t-1} have been obtained using x-doublings

It takes

$$\mathrm{I} + t\mathrm{H} + (7t - 2)\mathrm{M} + 2t\mathrm{S}$$

to fully recover them in affine coordinates

Trading-Off More Doublings

Since x-doublings are faster, we should try to use more

Split n in three as $n = 2^{2k}n_2 + 2^k n_1 + n_0$

Consider $Q_2 = [2^{2k}]P$, $Q_1 = [2^k]P$, and $Q_0 = P$

The x-coordinates of Q_2 and Q_1 are obtained with $2k$ successive x-doublings

Each one of the three points is reconstructed from its x-coordinate and one bit of information

Trading-Off More Doublings

Then perform $[n]P$ using interleaving with NAFs

A window w of size 3, 4, or 5 gives good results in practice

$$\begin{pmatrix} n_2 \\ n_1 \\ n_0 \end{pmatrix} = \begin{pmatrix} w_k & \dots & w_0 \\ v_k & \dots & v_0 \\ u_k & \dots & u_0 \end{pmatrix}$$

We call this approach w-w-w

w-*w*-*w*

If we precompute Q_i, $[3]Q_i$, $\ldots$, $[2^{w-1}-1]Q_i$ for $i = 0, 1, 2$

We then need: $2\frac{\ell}{3}$ x-doublings

$\frac{\ell}{3}$ regular LD doublings

$\frac{\ell}{3(w+1)}$ LD mixed additions

Trading-Off More Doublings

Even more?

Split n in four as $n = 2^{3k}n_3 + 2^{2k}n_2 + 2^k n_1 + n_0$

Consider $Q_3 = [2^{3k}]P$, $Q_2 = [2^{2k}]P$, $Q_1 = [2^k]P$, and $Q_0 = P$

The x-coordinates of Q_3, Q_2, and Q_1 are obtained with $3k$ successive x-doublings

Each one of the four points is reconstructed from its x-coordinates and one bit of information

Trading-Off More Doublings

Then perform $[n]P$ using two independent JSFs

One for n_3 and n_2 and one for n_1 and n_0

$$\begin{pmatrix} n_3 \\ n_2 \end{pmatrix} = \begin{pmatrix} w_k & \dots & w_0 \\ v_k & \dots & v_0 \end{pmatrix}_{\text{JSF}}$$

$$\begin{pmatrix} n_1 \\ n_0 \end{pmatrix} = \begin{pmatrix} u_k & \dots & u_0 \\ t_k & \dots & t_0 \end{pmatrix}_{\text{JSF}}$$

We call this approach x-JSF$_2$

x-JSF$_2$

If we precompute $Q_3 \pm Q_2$ and $Q_1 \pm Q_0$

We then need: $3\frac{\ell}{4}$ x-doublings

$\frac{\ell}{4}$ regular LD doublings

$\frac{\ell}{4}$ LD mixed additions

Trading-Off All the Doublings

Let us try Yao's method

Write n in base 2^k as $(n_{t-1} \dots n_0)_{2^k}$

Consider $Q_i = [2^{ki}]P$, for $i = 0, \dots, t-1$

We have $[n]P = [n_{t-1}]Q_{t-1} + \cdots + [n_0]Q_0$

which we can rewrite

$$[n]P = \sum_{j=1}^{2^k-1} [j] \Big(\sum_{n_i = j} Q_i \Big)$$

Trading-Off All the Doublings

Start with

$T = P_\infty$, $R = P_\infty$, and $j = 2^k - 1$

Repeat

$R = R + Q_i$ for each i such that $n_i = j$

$T = T + R$

$j = j - 1$

Until $j = 0$

Trading-Off All the Doublings

The complexity of this approach is

$k(t-1)$ successive x-doublings

t point reconstructions

$\left(1-\frac{1}{2^k}\right)2^k$ mixed LD additions on average

2^k LD full additions

Trading-Off All the Doublings

Unfortunately, this approach is not faster than Montgomery's method

It is quite expensive to retrieve the points Q_i's

If we try to keep this number of points low, then the number of full LD additions increases

Trading-Off All the Doublings

We also tried Bos–Coster's method with the same outcome and essentially the same reasons

Implementation/Experiments

We have implemented all the methods in `C++` using `NTL 6.0.0` built on top of `GMP 5.1.2`

The program is compiled and executed on a `quad core i7-2620` at `2.70Ghz`

We generate a total of 100 curves of the form

$$E : y^2 + xy = x^3 + x^2 + a_6,$$

where a_6 is a random element of $\mathbb{F}_{2^d}^*$

Implementation/Experiments

For each curve, a random point P is created as well as 100 random scalars selected in the interval $[0, 2^d + 2^{d/2} - 1]$

We assume that the point P needs to be decompressed for all the methods

The different methods are then tested against the same curves, points, and scalars

The computations are timed over 10 repetitions

Results

Degree 233: $I_{233}/M_{233} = 8.651$ and $S_{233}/M_{233} = 0.226$							
	$\#\mathcal{P}$	I_{233}	H_{233}	M_{233}	S_{233}	Time (ms)	Speed-up
Mont.	1	2	0	1402	928	1.102	0%
NAF_5	8	10	0	1253	1360	1.221	−10.81%
x-JSF	4	3	2	1181	1229	1.118	−1.51%
x-JSF_2	8	4	4	1094	1126	1.053	4.43%
4-4-4	12	14	3	1043	1108	1.079	2.05%
Degree 571: $I_{571}/M_{571} = 9.212$ and $S_{571}/M_{571} = 0.153$							
	$\#\mathcal{P}$	I_{571}	H_{571}	M_{571}	S_{571}	Time (ms)	Speed-up
Mont.	1	2	0	3430	2280	10.986	0%
NAF_6	16	18	0	2961	3269	12.154	−10.64%
x-JSF	4	3	2	2871	3004	10.153	7.58%
x-JSF_2	8	4	4	2618	2735	9.014	17.94%
5-5-5	24	26	3	2355	2601	8.843	19.51%

Future Work

We cannot generalize the idea to large odd characteristic

A possible generalization would be to investigate which endomorphisms have an x-coordinate that can be computed efficiently and independently from the y-coordinate

Certain endomorphisms $[k]P$ that can be split as the product of two isogenies on special families of curves are known to have this property

QUESTIONS?