Efficient Finite Field and Elliptic Curve Arithmetic

Laurent Imbert

CNRS, LIRMM, Université Montpellier 2

Summer School ECC 2011 – Nancy, September 12-16, 2011

Part 2

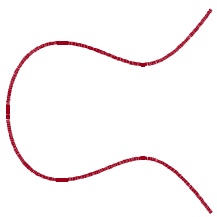
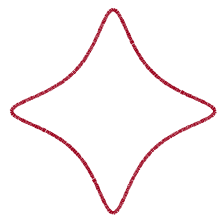
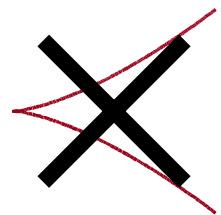
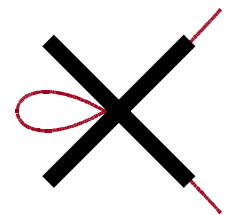
Elliptic curve arithmetic

The two facets of an elliptic curve

An elliptic curve is:

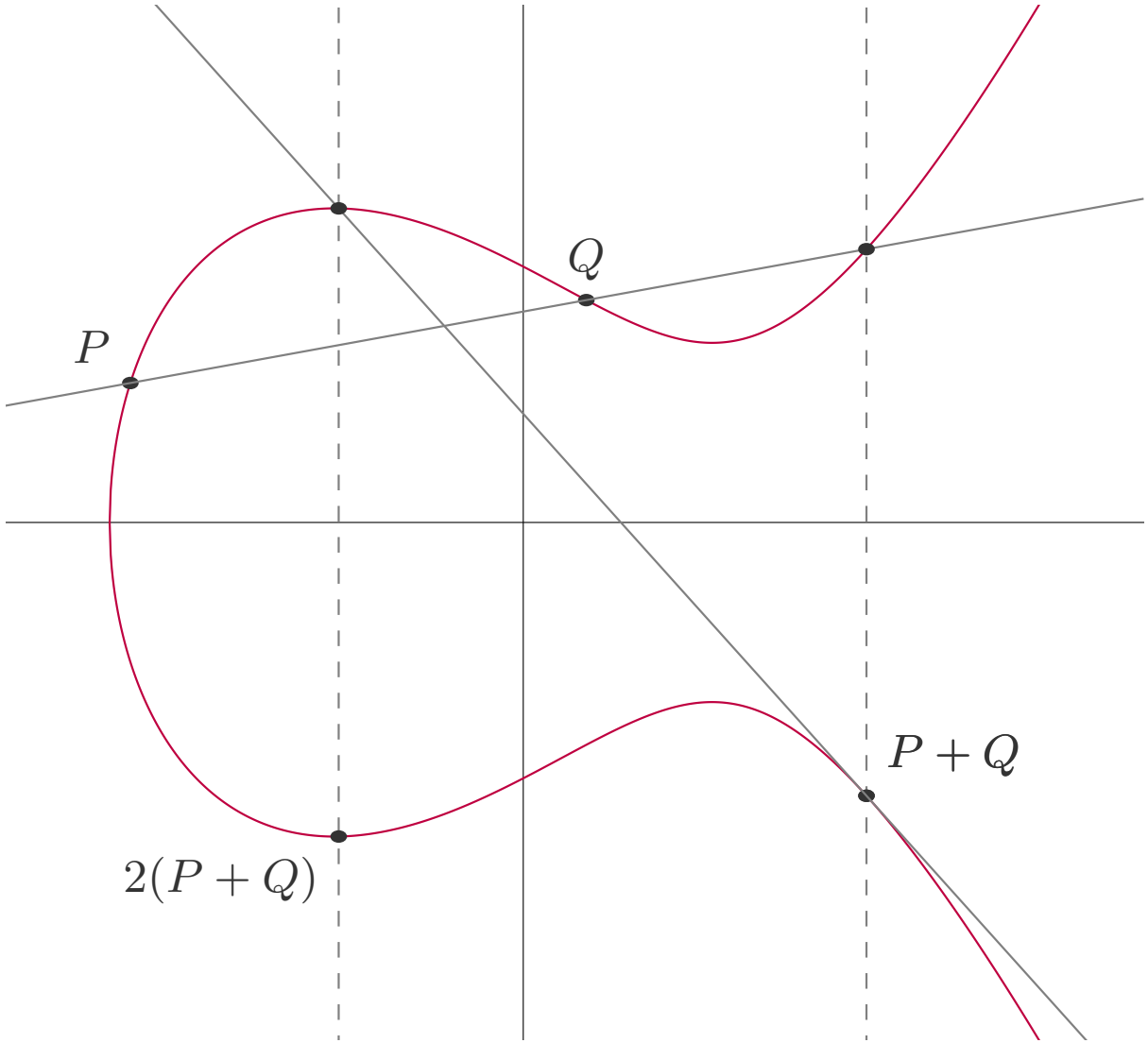
- a geometrical object: a nonsingular curve given by an equation

$$y^2 = f(x), \quad \text{with } \deg f \in \{3, 4\}$$

- an algebraic object: one can "add" stuff! This composition operation has a nice geometrical interpretation.

Adding points on an elliptic curve



Weierstrass model

- An elliptic curve over a field K of characteristic $\neq 2, 3$ is given by an equation of the form

$$E : y^2 = x^3 + ax + b, \quad \text{with } a, b \in K \tag{1}$$

and $\Delta = -16(4a^3 + 27b^2) \neq 0$

- The set of K-rational points of an elliptic curve is

$$E(K) = \{(x, y) \in K \times K \ ; \ y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

- In the general case, we consider the long Weierstrass form

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

where $a_1, a_2, a_3, a_4, a_6 \in K$.

Algebraic description of the addition operation

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two points on

$$E : y^2 = x^3 + ax + b$$

The slope of the line (P_1, P_2) is given by

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P_1 \neq \pm P2 \\ \dfrac{3x_1^2 + a}{2y_1} & \text{if } P_1 = P_2 \end{cases}$$

The sum of P and Q is the point

$$P + Q = (\lambda^2 - x_1 - x_2, \quad \lambda(x_1 - x_3) - y_1).$$

Properties of the addition operation

For all $P, Q, R \in E$, the addition law has the following properties:

- $P + \mathcal{O} = \mathcal{O} + P = P$
- $P + (-P) = \mathcal{O}$
- $(P + Q) + R = P + (Q + R)$
- $P + Q = Q + P$

Thus, $(E, +)$ forms an Abelian group, with the properties required for group-based cryptography:

- the group operation is easy to implement (basic algebraic operations)
- if K is a well chosen finite field, the computation of discrete logarithms is hard (much harder than over $\mathbb{F}_q^*$)[1]

[1] Vanessa's lecture

What finite field should we use?

Efficiency considerations

- Prime fields $\mathbb{F}_p$, where p is a nice prime (software implementations)

  Ex: $M_{521} = 2^{521} - 1$, $2^{255} - 19$

- Binary fields $\mathbb{F}_{2^m} = \mathbb{F}_2[X]/(f(X))$, where $f(X)$ is a nice ireducible polynomial of degree m (hardware implementations)

  Ex: trinomials, pentanomials, all-one polynomials

- Alternatives: $\mathbb{F}_{p^m}$ where both p and the irreducible polynomial have nice properties

  Ex: optimal extension fields

What finite field should we use?
Security considerations

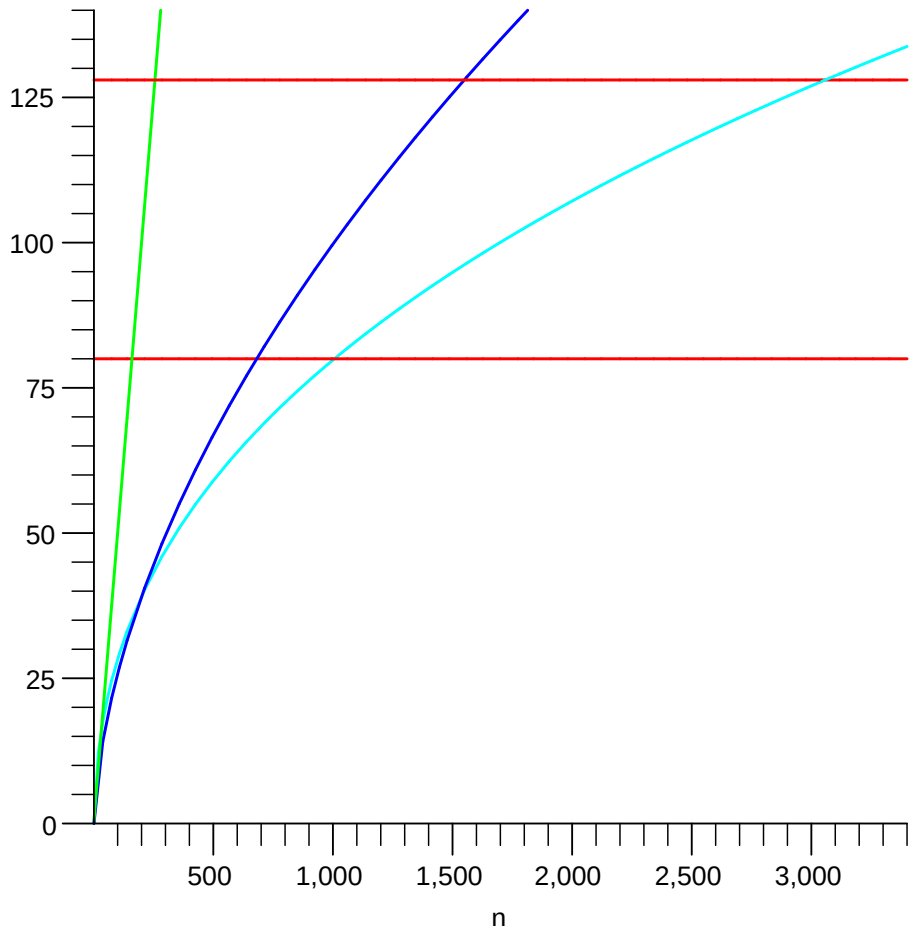
For cryptographic usage, an elliptic curve E defined over $\mathbb{F}_q$, $q = p^m$ should satisfy:

- $\#E(\mathbb{F}_q) = h \times n$
- n is prime, h is small (ideally $h = 1$)
- $n > 2^{160}$ to avoid BSGS/Pollard ρ attacks in $O(\sqrt{n})$
- $n \neq p$ to avoid anomalous attack
- $q^t \not\equiv 1 \pmod{n}$ for all $t \leq 20$ to avoid the MOV attack
- m is prime to avoid Weil descent attacks

How big should $\mathbb{F}_q$ be?

Hasse-Weil bounds: $|\#E(\mathbb{F}_q) - (q+1)| \leq 2\sqrt{q}$ $\qquad$ $(\#E(\mathbb{F}_q) \approx q)$

Best known attacks: $O(\sqrt{N})$



Cost estimation

We want to compute the group operation as fast as possible (*)

A not-too-bad estimation of the time can be obtained by counting the number of field operations of each type:

- # field addition/subtraction (A)
- # field multiplications (M)
- # field squarings (S)
- # field inversions (I)
- # "small" field multiplications (e.g. (a))

Estimates:

- $A \ll M$
- Over $\mathbb{F}_p$: $S \approx 0.8M$, $I \gg M$ ($I > 30M$)
- Over $\mathbb{F}_{2^m}$: $S \ll M$ (negligible)

Cost of the group law for $E/K : y^2 = x^3 + ax + b$

- Identity: $P + \mathcal{O} = P$ and $P + (-P) = \mathcal{O}$ for all $P \in E(K)$
- Point negation: Let $P = (x_1, y_1) \in E(K)$. Then $-P = (x, -y)$
- Point addition: Let $P = (x_1, y_1)$, $Q = (x_2, y_2)$ with $P \neq \pm Q$. Then $P + Q = (x_3, y_3)$, with

$$1I + 2M + 1S$$

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2, \qquad y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1$$

- Point doubling: If $P = (x_1, y_1)$, where $P \neq -P$. Then $[2]P = (x_3, y_3)$ with

$$1I + 2M + 2S$$

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1, \qquad y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1$$

Projective coordinates

Let $c, d > 0$. Define an equivalence relation on $K^3 \setminus \{(0,0,0)\}$

$$(X, Y, Z) \sim (\lambda^c X, \lambda^d Y, \lambda Z) \text{ for all } \lambda \in K^*$$

A projective point, denoted $(X : Y : Z)$, is a class of $K^3 \setminus \{0,0,0\}$ modulo the equivlence relation $\sim$

The set of projective points is called the 2-dimensional projective space over K, denoted $\mathbb{P}^2(K)$.

$$\mathbb{A}^2(K) \longrightarrow \mathbb{P}^2(K)^* = \{(X : Y : Z) \ : \ X, Y, Z \in K, \ Z \neq 0\}$$
$$(x, y) \longmapsto (X/Z^c : Y/Z^d : 1)$$

The points at infinity

The set $\mathbb{P}^2(K)^0 = \{(X : Y : Z) : X, Y, Z \in K, Z = 0\}$ is called the line at infinity

The points at infinity on E are the points of $\mathbb{P}^2(K)^0$ which lie on E

Projective form of the Weierstrass equation ($c = d = 1$)

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

For $Z = 0$, the point $(X : Y : Z)$ must satisfy $0 = X^3 + 0$, which implies $X = 0$, and $Y \neq 0$ (since $(0, 0, 0) \notin \mathbb{P}^2(K)$)

$\mathcal{O} = (0 : 1 : 0)$ is the only point at infinity on E

Jacobian projective coordinate

Let $c = 2$ and $d = 3$. The projective point $(X : Y : Z)$, $Z \neq 0$ corresponds to the affine point $(X/Z^2, Y/Z^3)$

Projective form of the Weierstrass equation:

replace x by X/Z^2 and y by Y/Z^3 in $y^2 = x^3 + ax + b$ and clear denominators

$$Y^2 = X^3 + aXZ^4 + bZ^6$$

Point at infinity corresponds to $\mathcal{O} = (1 : 1 : 0)$

The negative of $(X : Y : Z)$ is $(X : -Y : Z)$

Inversions free formula

Let $P = (X_1 : Y_1 : Z_1)$ with $P \neq -P$.

Replace x_1 by X_1/Z_1^2 and y_1 by Y_1/Z_1^3 in the affine doubling formula leads to $[2]P = (X_3' : Y_3' : 1)$, with

$$X_3' = \frac{(3X_1^2 + aZ_1^4)^2 - 8X_1Y_1^2}{4Y_1^2Z_1^2}, \qquad Y_3' = \frac{3X_1^2 + aZ_1^4}{2Y_1Z_1}\left(\frac{X_1}{Z_1^2} - X_3'\right) - \frac{Y_1}{Z_1^3}$$

Use the equivalence relation to clear denominators. Set $X_3 = X_3'Z_3^2$ and $Y_3 = Y_3'Z_3^3$ to get $(X_3 : Y_3 : Z_3)$

$$\begin{aligned} X_3 &= (3X_1^2 + aZ_1^4)^2 - 8X_1Y_1^2 \\ Y_3 &= (3X_1^2 + aZ_1^4)(4X_1Y_1^2 - X_3) - 8Y_1^4 \\ Z_3 &= 2Y_1Z_1 \end{aligned}$$

Implement!

Common implementation techniques

- Common-subexpression elimination
- Trade multiplication for squarings: $2XY = (X + Y)^2 - X^2 - Y^2$
- Curve parameters: If $a = -3$, $3X_1^2 + aZ_1^4$ can be computed as a difference of two squares
- Add redundancy: Modified Jacobian: $(X : Y : Z : T)$, $x = X/Z^2$, $y = Y/Z^3$, $T = aZ^4$
  Chudnovsky coordinates: $(X : Y : Z : Z^2 : Z^3)$, $x = X/Z^2$, $y = Y/Z^3$
- Mixed formula ($Z = 1$)
- Readdition
- co-Z formula
- etc.

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

Main computations

Scalar multiplication is the main operation

$$k, P \longrightarrow [k]P = P + \cdots + P, \qquad (k \text{ times})$$

Various situations occur:

$[k\|r]P$	First step ECDH
$[r]P$	ECDSA signature
$[k\|r]Q$	Second step ECDH
$[r]Q$	ECIES encryption
$[u]P + [u']Q$	ECDSA verification

k: known scalar (domain parameter, private key)
r: generated online at random
u: unknown in advance, result of online computations
P: point known in advance (domain parameter, private key)
Q: point unknown in advance

Addition chains

An addition chain computing k is a sequence $1 = u_0 < \cdots < u_n = k$ such that, for all $m \geq 1$, $u_m = u_i + u_j$ with $0 \leq i \leq j < m$

Finding optimal addition chain is very difficult, but good heuristics exists to get raisonably short addition chains

$$[k]P = \begin{cases} [2]([k/2]P) & \text{if } k \equiv 0 \pmod 2 \\ [2]([k/2]P) + P & \text{if } k \equiv 1 \pmod 2 \end{cases}$$

Example: 289 : $1, 2, 4, 8, 9, 18, 36, 72, 144, 288, 289$

When the scalar k is known in advance, "short" addition chains can be computed offline

When the scalar k is generated online at random, it may be generated directly in a non-standard, convenient representation (be careful!)

Double-and-add algorithms

Input: $P \in E(K), k = (k_{n-1}, \ldots, k_0)_2$, with $k_{n-1} = 1$
Output: $[k]P$

Right-to-left (RL)

1: $R \leftarrow \mathcal{O}$
2: For $i = 0$ to $n-1$ do
3: If $k_i = 1$ then
4: $R \leftarrow R + P$
5: $P \leftarrow [2]P$
6: return R

Left-to-right (LR)

1: $R \leftarrow P$
2: For $i = n-2$ downto 0 do
3: $R \leftarrow [2]R$
4: If $k_i = 1$ then
5: $R \leftarrow R + P$
6: return R

RL: $[k]P = [k_0]P + [2k_1]P + [2^2k_2]P + \cdots + [2^{n-1}k_{n-1}]P$

LR: $[k]P = [2]([k/2]P) + [k_0]P$
$= [2]([2]([k/4]P) + [k_1]P) + [k_0]P = [2]([2]([2](\ldots$

Average cost: $(n-1)$DBL + $(n/2)$ADD (mixed additions for LR)

Signed digits representations

In the group of points of an elliptic curve, computing the inverse of an element ($P \rightarrow -P$) is almost free.

It may therefore be advantageous to consider addition/subtraction chains using signed digits (SD) representations

$$k = \sum_{i=0}^{n} k_i 2^i, \quad \text{with } k_i \in \{-1, 0, 1\}$$

Length: at most $n + 1$ digits if n is the binary length of k

Canonical SD: no consecutive nonzero digits ($k_i k_{i+1} = 0$), also called the Non Adjacent Form (NAF)

NAF properties: unique representation, minimal density:
for $k_i \in \{-1, 0, 1\}$, the average density of nonzero digits is 1/3

Computing NAF(k)

Booth's algorithm: bits are scanned from right to left; block of consecutive 1s are replaced by a block of 0s and $\overline{1}$; e.g. $(1, 1, 0, 1, 1, 1)_2 \rightarrow (1, 0, \overline{1}, 1, 0, 0, \overline{1})_{SD2}$.

Reitweitzner's variant: ensures the NAF property when blocks of 1s are separated by an isolated 0; e.g. $(1, 1, 0, 1, 1, 1)_2 \rightarrow (1, 0, 0, \overline{1}, 0, 0, \overline{1})_{SD2}$

Reitweitzner's algorithm

Input: $k = (k_{n-1}, \ldots, k_0)_2$

Output: $k' = (k'_n, k'_{n-1}, \ldots, k'_0)_{SD2}$

1: $c_0 \leftarrow 0$, $k_{n+1} \leftarrow 0$, $k_n \leftarrow 0$
2: For $i = 0$ to n do
3: $\quad c_{i+1} \leftarrow \lfloor (c_i + k_i + k_{i+1})/2 \rfloor$
4: $\quad k'_i \leftarrow c_i + k_i - 2c_{i+1}$
5: Return $k' = (k'_n, k'_{n-1}, \ldots, k'_0)$

Idea: compute $3n - n$ with the additional rule $0 - 1 = \overline{1}$ and discard the least significant bit

$O(\log k)$

Classical NAF recoding

Input: $k > 0$

Output: NAF(k)

1: $i \leftarrow 0$
2: While $k \geq 1$ do
3: If k is odd then
4: $k_i \leftarrow 2 - (k \bmod 4)$
5: else
6: $k_i \leftarrow 0$
7: $k \leftarrow (k - k_i)/2$, $i \leftarrow i + 1$
8: Return $(k_{i-1}, \ldots, k_1, k_0)$

Idea: if k is odd, k_i is chosen in $\{-1, 1\}$ so that $(k - k_i)/2$ is even

$O(\log k)$

NAF left-to-right scalar multiplication

Input: k, P
Output: $[k]P$

1: Compute NAF(k) = $(k_n, \ldots, k_0)$
2: $R \leftarrow P$
3: For $i = n - 1$ downto 0 do
4: $\quad R \leftarrow [2]R$
5: $\quad$ If $k_i \neq 0$ then
6: $\quad\quad R \leftarrow R + k_i P$
7: Return R

NAF(k), obtained from right to left, has to be fully computed before any left-to-right scalar multiplication algorithm.

Left-to-right NAF recoding variants do exist, but are more difficult to implement and require more memory

Average cost: nDBL + $(n/3)$ADD (mixed additions for LR)

Window methods: process w digits of k at a time

Width-w NAF: $k = \sum_{i=0}^{n} k_i 2^i$, with each $k_i \neq 0$ is odd, $|k_i| < 2^{w-1}$, at most one of any w consecutive digits is $\neq 0$.

- k has a unique width-w NAF, denoted $\mathrm{NAF}_w(k)$.
- $\mathrm{NAF}_2(k) = \mathrm{NAF}(k)$
- Length: at most $n + 1$ digits if $|k| = n$
- Average density: $1/(w + 1)$

Example: $n = 314159$. ($\overline{d}$ denotes $-d$)

$$
\begin{aligned}
(n)_2 &= 1\,0\,0\,1\,1\,0\,0\,1\,0\,1\,1\,0\,0\,1\,0\,1\,1\,1\,1 \\
\mathrm{NAF}(n) &= 1\,0\,1\,0\,\overline{1}\,0\,1\,0\,\overline{1}\,0\,\overline{1}\,0\,1\,0\,\overline{1}\,0\,0\,0\,\overline{1} \\
\mathrm{NAF}_3(n) &= 1\,0\,0\,0\,3\,0\,0\,1\,0\,0\,3\,0\,0\,0\,3\,0\,0\,0\,\overline{1} \\
\mathrm{NAF}_4(n) &= 0\,0\,5\,0\,0\,0\,\overline{3}\,0\,0\,0\,\overline{5}\,0\,0\,0\,3\,0\,0\,0\,\overline{1}
\end{aligned}
$$

Window NAF method for point multiplication

Compute $\mathrm{NAF}_w(k)$: replace $2 - (k \bmod 4)$ in NAF by $-2^{w-1} \leq k \bmod 2^w \leq 2^{w-1} - 1$

Precompute $P_i = [i]P$ for $i \in \{1, 3, 5, \ldots, 2^{w-1} - 1\}$
Cost: PRECOMP $\leq$ 1DBL + $(2^{w-2} - 1)$ADD

Scan digits of $\mathrm{NAF}_w(k)$ from left to right ; add P_{k_i} or subtract P_{-k_i} whenever $k_i \neq 0$

Average cost: nDBL + $(n/(w+1))$ADD (mixed?) + PRECOMP

Example: $\mathrm{NAF}_4(314159) = 0\ 0\ 5\ 0\ 0\ 0\ \overline{3}\ 0\ 0\ 0\ \overline{5}\ 0\ 0\ 0\ 3\ 0\ 0\ 0\ \overline{1}$

$\mathcal{O}$, $5P$, $10P$, $20P$, $40P$, $80P$, $77P$, $144P$, $\ldots$, $1232P$, $1227P$, $2454P$, $\ldots$, $19632P$, $19635P$, $39270P$, $\ldots$, $314160P$, $314159P$

Alternatives: sliding window, fractional window methods

Algorithms based on point tripling

Double-base chains:

$$k = \sum_{i=0}^{m-1} k_i \, 2^{a_i} 3^{b_i}, \text{ where } k_i \in \mathcal{S} \text{ and } (a_i, b_i) \searrow$$

$314159 = 2^4 3^9 - 2^0 3^6 - 3^3 - 3^2 - 3 - 1$
$[314159]P = 3(3(3(3^3(2^4 3^3 P - P) - P) - P) - P) - P$

Alternatives:

- Yao/Meloni's algorithm
- Hybrid binary-ternary
- Tree-based approach
- Multi-base (see Patrick Longa's talk at ECC)
- etc.

Isogenies

E_1/K and E_2/K are isogenous over K if there exists a rational map $\phi : E_1 \to E_2$ with coefficients in K such that $\phi(\mathcal{O}_1) = \mathcal{O}_2$

For every (non constant) -isogeny $\phi : E_1 \to E_2$, there exists a unique dual -isogeny $\hat{\phi} : E_2 \to E_1$ such that

$$\hat{\phi} \circ \phi = [\ ]$$

An -isogeny can be described as a rational map involving polynomials of degree $\leq$

DIK curves

DIK2: Elliptic curves family for which [2] splits into two 2-isogenies

$$y^2 = x^3 + ux^2 + 16ux$$

DIK3: Elliptic curves family for which [3] splits into two 3-isogenies

$$y^2 = x^3 + 3u(x + 1)^2$$

Tripling on DIK3 curves

$(x_1, y_1) \longrightarrow (x_t, y_t)$

$$x_t = x_1 + 4u + 12u\left(\frac{x_1 + 1}{x_1^2}\right)$$

$$y_t = y_1\left(1 - 12u\left(\frac{x_1 + 2}{x_1^3}\right)\right)$$

$(x_t, y_t) \longrightarrow (x_3, y_3) = [3]P$

$$x_3 = \frac{1}{3^2}\left(x_t - 12u + \frac{12u(4u-9)}{x_t} - \frac{4u(4u-9)^2}{x_t^2}\right)$$

$$y_3 = \frac{1}{3^3}y_t\left(1 - \frac{12u(4u-9)}{x_t^2} + \frac{8u(4u-9)^2}{x_t^3}\right)$$

`http://www.hyperelliptic.org/EFD/g1p/auto-3dik.html`

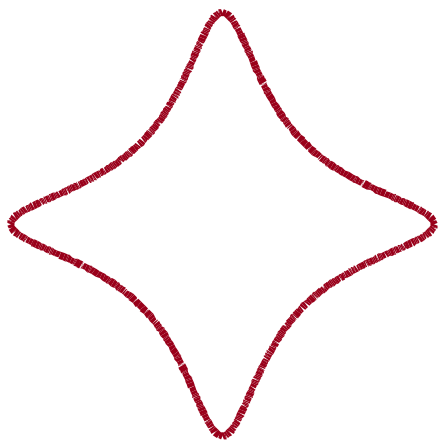
Edwards curves

- H. M. Edwards, *Bulletin of the AMS*, 2007

$$x^2 + y^2 = a^2(1 + x^2y^2), \quad \text{with } a^5 \neq a \tag{2}$$

- D. Bernstein and T. Lange introduced parameter d to cover more curves over K

$$x^2 + y^2 = c^2(1 + dx^2y^2), \text{ with } c, d \neq 0 \text{ and } d \text{ not a square in } K$$



Group law on Edwards curves

- Addition: $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$

$$x_3 = \frac{x_1y_2 + y_1x_2}{c(1 + dx_1x_2y_1y_2)}, \qquad y_3 = \frac{y_1y_2 - x_1x_2}{c(1 - dx_1x_2y_1y_2)}$$

- Neutral element: affine point of coordinates $(0, c)$
- Negative of a point: $-(x, y) = (-x, y)$
- Doubling: $[2](x, y) = \left(\frac{xy + yx}{c(1 + dxxyy)}, \frac{yy - xx}{c(1 - dxxyy)}\right)$
- Unified group operations

Unified operations

- If d is not a square then Edwards addition law is complete
  - if (x_1, y_1) and (x_2, y_2) on the curve then $dx_1x_2y_1y_2 \neq \pm 1$

- Formula is correct for all affine point including $(0, c)$, $P + (-P)$.

- Doubling formula is exactly identical to addition formula
  - no re-arrangement like in Hessian form where
    $[2](X_1 : Y_1 : Z_1) = (Z_1 : X_1 : Y_1) + (Y_1 : Z_1 : X_1)$.

`http://www.hyperelliptic.org/EFD/g1p/auto-edwards.html`

Comparisons with other fast unified formulas

Coordinates	Cost add/dbl	Ref
Projective	11M + 6S + 1D	Brier/Joye 03
Projective ($a = -1$)	13M + 3S	Brier/Joye 03
Jacobi intersection	13M + 2S + 1D	Liardet/Smart 01
Jacobi quartic	10M + 3S + 1D	Billet/Joye 01
Hessian	12M	Joye/Quisquater 01
Edwards ($c = 1$)	10M + 1S + 1D	Bernstein/Lange 07

Optimizing Edwards doubling ($c = 1$)

Affine: $[2](x, y)$

$$\left(\frac{xy + yx}{1 + dxxyy}, \frac{yy - xx}{1 - dxxyy}\right)$$

$$= \left(\frac{2xy}{1 + dx^2y^2}, \frac{y^2 - x^2}{1 - dx^2y^2}\right)$$

$$= \left(\frac{2xy}{x^2 + y^2}, \frac{y^2 - x^2}{2 - x^2 - y^2}\right)$$

$$= \left(\frac{(x + y)^2}{x^2 + y^2} - 1, \frac{y^2 - x^2}{2 - x^2 - y^2}\right)$$

Projective: $[2](X_1 : Y_1 : Z_1)$

$$B = (X_1 + Y_1)^2$$
$$C = X_1^2$$
$$D = Y_1^2$$
$$E = C + D$$
$$H = Z_1^2$$
$$J = E - 2H$$
$$X_3 = (B - E)J$$
$$Y_3 = E(C - D)$$
$$Z_3 = EJ$$

Cost: 3M + 4S + 6A

Comparisons

Doubling:

System	Cost
Proj.	5M + 6S
Proj. ($a = -3$)	7M + 3S
Hessian	7M + 1S
DIK 3	2M + 7S
Jac.	1M + 8S
Jac. ($a = -3$)	3M + 5S
Jacobi quartic	2M + 6S
Jacobi intersec.	3M + 4S
Edwards	3M + 4S
DIK 2	2M + 5S

Jac-3 vs. Edwards:

	Jac-3	Edwards
Double	3M + 5S	3M + 4S
Triple	7M + 7S	9M + 4S
Add	11M + 5S	10M + 1S + 1D
Re-Add	10M + 4S	10M + 1S + 1D
Mixed	7M + 4S	9M + 1S + 1D

`http://www.hyperelliptic.org/EFD/g1p/index.html`

Side channel attacks

Do we really want to compute $[k]P$ as fast as possible?

Side channel attacks:

- Timing attacks
- Simple power analysis
- Differential power analysis
- Electromagnetic analysis
- etc.

Algorithmic countermeasures:

- Unified formulæ
- Double-and-always-add
- Atomic blocks
- Randomization
- etc.

The Montgomery ladder

Input: $P \in E$, $k = (k_{n-1} \dots k_0)_2$

Output: $[k]P \in E$

1: $P_1 \leftarrow P$, $\quad P_2 \leftarrow [2]P$
2: For $i = k - 1$ downto 0 do
3: $\quad$ If $n_i = 0$ then
4: $\quad\quad P_1 \leftarrow [2]P_1, \quad P_2 \leftarrow P_1 + P_2$
5: $\quad$ else
6: $\quad\quad P_1 \leftarrow P_1 + P_2, \quad P_2 \leftarrow [2]P_2$
7: Return P_1

Note that $P_2 - P_1 = P$.

Cost: $(6M + 4S)(|k|_2 - 1)$

Montgomery curves

Montgomery curve:

$$E_M : \; By^2 = x^3 + Ax^2 + x, \quad A, B \in \mathbb{F}_{p^k}, \; p > 3$$

Differential addition:

$$[m+n]P = [m]P + [n]P = [X_{m+n} : - : Z_{m+n}]$$

$$X_{m+n} = Z_{m-n}\left((X_m - Z_m)(X_n + Z_n) + (X_m + Z_m)(X_n - Z_n)\right)^2$$
$$Z_{m+n} = X_{m-n}\left((X_m - Z_m)(X_n + Z_n) - (X_m + Z_m)(X_n - Z_n)\right)^2$$

Doubling:

$$4X_nZ_n = (X_n + Z_n)^2 - (X_n - Z_n)^2,$$
$$X_{2n} = (X_n + Z_n)^2(X_n - Z_n)^2,$$
$$Z_{2n} = 4X_nZ_n\left((X_n - Z_n)^2 + ((A+2)/4)\,(4X_nZ_n)\right).$$

`http://www.hyperelliptic.org/EFD/g1p/auto-montgom.html`

Conversion to Montgomery curves

$$E_M : By^2 = x^3 + Ax^2 + x \longleftrightarrow E_W : y^2 = x^3 + ax + b$$

$E_M \longrightarrow E_W$: always possible

$a := 1/B^2 - A^2/3B^2$

$b := -A^3/27B^3 - aA/3B$

$E_W \longrightarrow E_M$: conditional

If $\alpha \in \mathbb{F}_p$ is a root of $x^3 + ax + b$

and $3\alpha^2 + a$ is a quadratic residue modulo p

Then set $s := \sqrt{(3\alpha^2 + a)^{-1}}$, $A := 3\alpha s$, $B := s$

The change of variables $(x, y) \longrightarrow (x/s + \alpha, y/s)$ gives a curve E_M isomorphic to E

Speed records

http://www.patricklonga.bravehost.com/speed_ecc.html#speed

http://www.loria.fr/~zimmerma/ecc.html