Fast scalar multiplication on elliptic curves

Tanja Lange

Technische Universiteit Eindhoven

tanja@hyperelliptic.org

08.05.2007

Overview

- Why scalar multiplication
- Elliptic curves
  - Definition and group law in affine coordinates
  - Other coordinate systems
  - Comparison
- Side-channel attacks
  - Why unified group laws?
  - Edwards coordinates
  - Comparison
- Multi-scalar multiplication

Why scalar multiplication?

Diffie-Hellman Key exchange

Alice		**Bob**
1. secretly generates $a < \vert\langle P\rangle\vert$		1. secretly generates $b < \vert\langle P\rangle\vert$
2. computes $Q_1 = [a]P$		2. computes $Q_2 = [b]P$
3. transmits Q_1		3. transmits Q_2
4. computes $[a]Q_2$	$= [ab]P =$	4. computes $[b] \cdot Q_1$

Common Key: the group element $k = [ab]P \in \langle P\rangle$ can be used in symmetric encryption.

ElGamal encryption

Public parameters:
Group G, generator P, ord$(P) = l$, some invertible embedding function $H : m \to G$.
Receiver has secret key s_A and public key $P_A = [s_A]P$.

- Encrypt message m
  - choose random integer k
  - compute $R = [k]P$ and $c = H(m) + [k]P_A$
- Decrypt ciphertext (R, c)
  - compute $S = [s_A]R$
  - obtain $m = H^{-1}(c - S)$

  (This gives m since $S = [s_A]R = [ks_A]P = [k]P_A$).

Disclaimer: this is the school-book method, do not implement as shown.

Elliptic curve Digital Signature Algorithm

Elliptic curve E, point $P \in E$, ord$(P) = l$, some cryptographic hash function $h : m \to \mathbb{Z}$. Point R has coordinate x_R.

Sign message m:

- choose random integer k
- compute $R = [k]P$ and put $r = x_R \pmod{l}$
- put $s = k^{-1}(h(m) + rs_A) \pmod{l}$.

Verify signature (r, s):

- compute $w = s^{-1} \pmod{l}$
- compute $Q_1 = [wr]P_A$, $Q_2 = [wh(m)]P$ and $Q = Q_1 \oplus Q_2$
- accept signature if and only if $x_Q \equiv r \bmod l$.

This accepts valid signatures since

$$[s^{-1}rs_A]P \oplus [s^{-1}h(m)]P = [(h(m)+rs_A)^{-1}k(rs_A+h(m))]P = [k]P.$$

DL systems

These systems assume that the Discrete Logarithm Problem (DLP) is hard to solve, i.e.

- given P and $P_A = [s_A]P$
- it is hard to find s_A.

The Computational Diffie-Hellman Problem (CDHP) is the problem

- given P, $P_A = [s_A]P$, and $P_B = [s_B]P$
- compute $[s_A s_B]P$.

The Decisional Diffie-Hellman Problem (DDHP) is the problem

- given P, $P_A = [s_A]P$, $P_B = [s_B]P$ and $R = [r]P$
- decide whether $R = [s_A s_B]P$.

Elliptic curves

Elliptic curve

$$E : y^2 + \underbrace{(a_1 x + a_3)}_{h(x)} y = \underbrace{x^3 + a_2 x^2 + a_4 x + a_6}_{f(x)}, \;\; h, f \in \mathbb{F}_q[x].$$

Group: $E(\mathbb{F}_q) = \{\, (x, y) \in \mathbb{F}_q^2 \; : \; y^2 + h(x)y = f(x) \,\} \cup \{\, P_\infty \,\}$

Often $q = 2^r$ or $q = p$, prime. Isomorphic transformations lead to

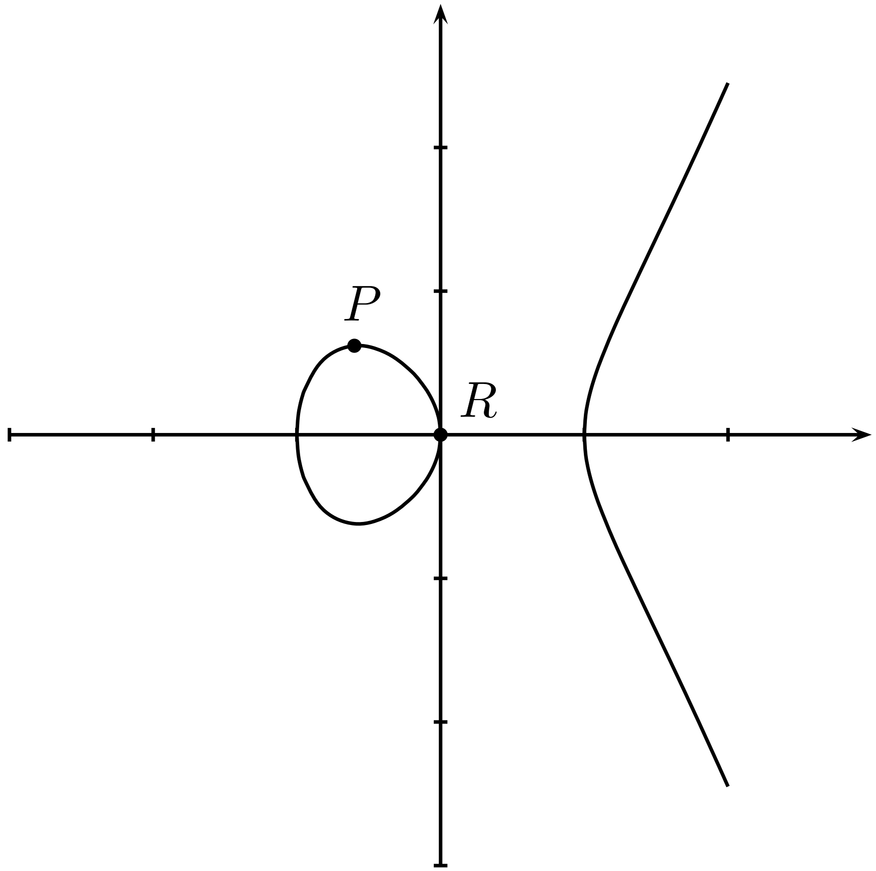
$y^2 = f(x)$ $\quad$ q odd,

for

$$\begin{array}{rcl} y^2 + xy & = & x^3 + a_2 x^2 + a_6 \\ y^2 + y & = & x^3 + a_4 x + a_6 \end{array} \qquad q = 2^r, \qquad \begin{array}{l} \text{curve non-supersingular} \\ \text{curve supersingular} \end{array}$$

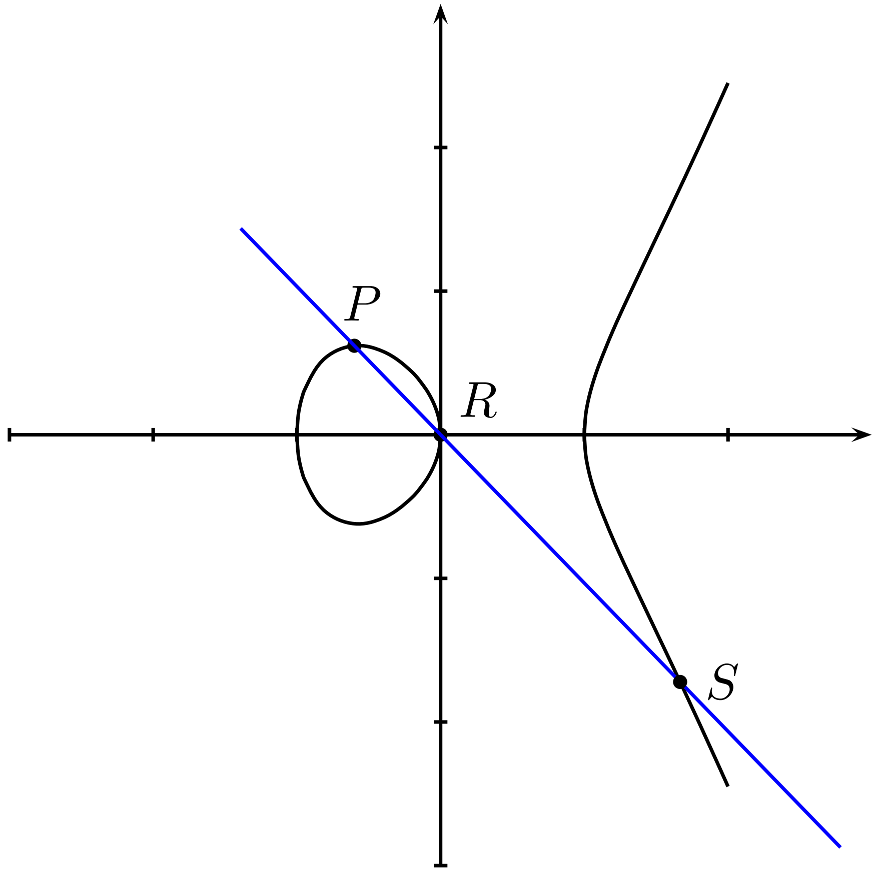
Group Law in $E(\mathbb{R}), h = 0$

$y^2 = x^3 - x$



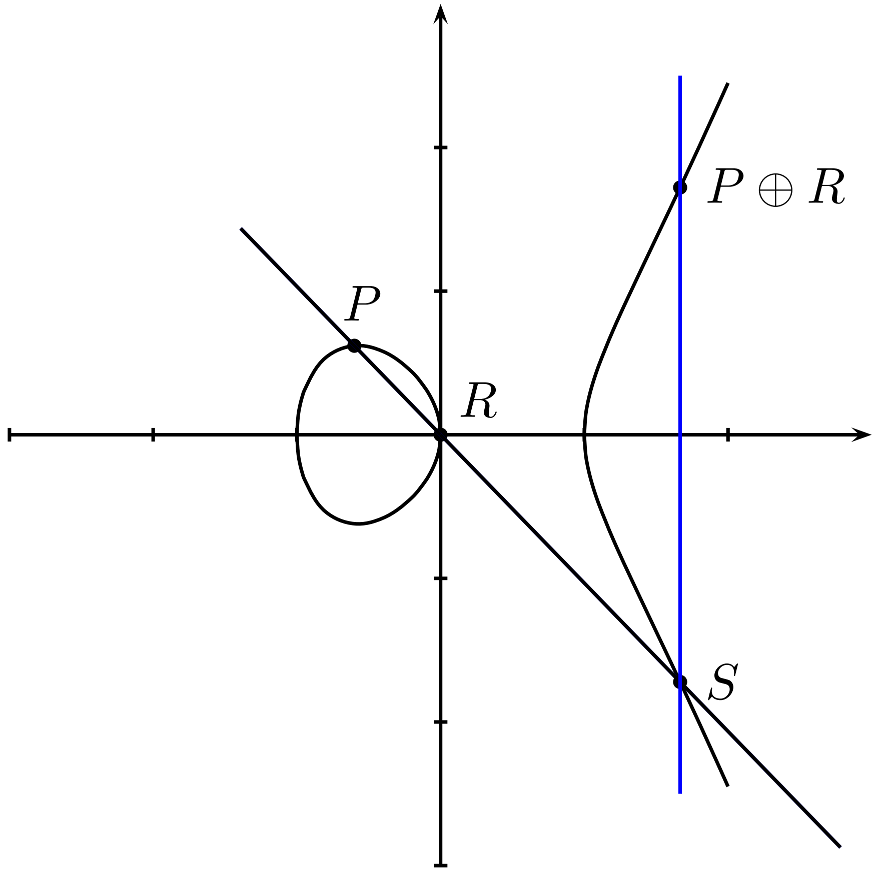
Group Law in $E(\mathbb{R}), h = 0$

$y^2 = x^3 - x$



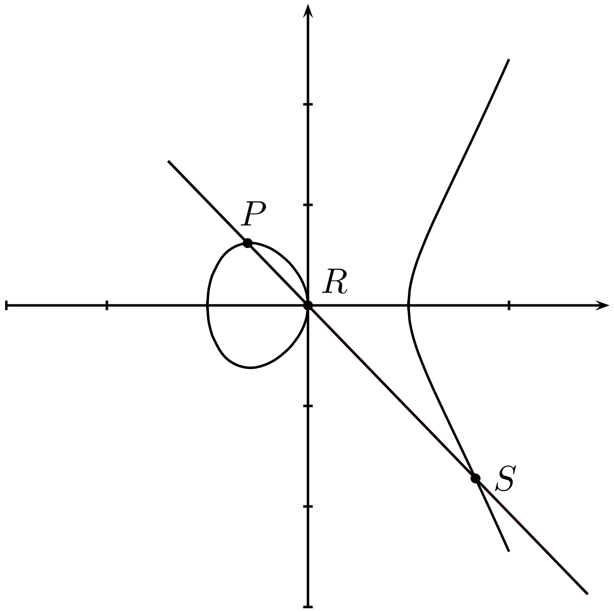
Group Law in $E(\mathbb{R}), h = 0$

$y^2 = x^3 - x$



Group Law (q odd)

$$E: y^2 = x^3 + a_4 x + a_6,\ a_i \in \mathbb{F}_q$$



Line $y = \lambda x + \mu$ has slope

$$\lambda = \frac{y_R - y_P}{x_R - x_P}.$$

Equating gives

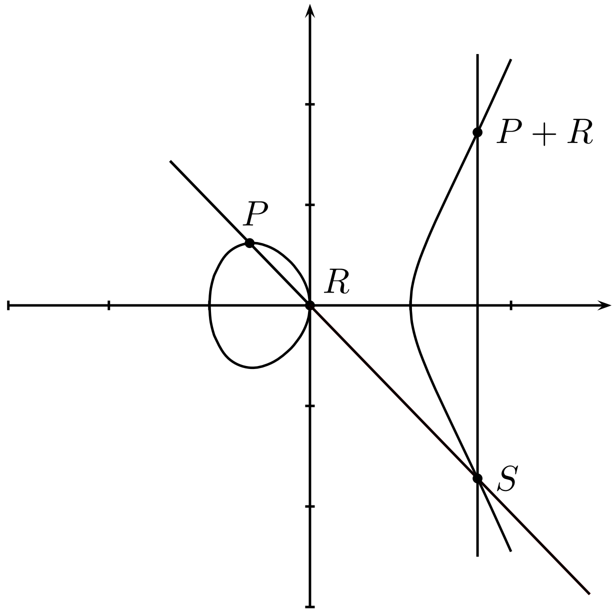
$$(\lambda x + \mu)^2 = x^3 + a_4 x + a_6.$$

This equation has 3 solutions, the x-coordinates of P, R and S, thus

$$\begin{aligned} (x - x_P)(x - x_R)(x - x_S) &= x^3 - \lambda^2 x^2 + (a_4 - 2\lambda\mu)x + a_6 - \mu^2 \\ x_S &= \lambda^2 - x_P - x_R \end{aligned}$$

Group Law (q odd)

$$E : y^2 = x^3 + a_4 x + a_6, \; a_i \in \mathbb{F}_q$$



Point P is on line, thus

$y_P = \lambda x_P + \mu$, i.e.
$\mu = y_P - \lambda x_P$,

and

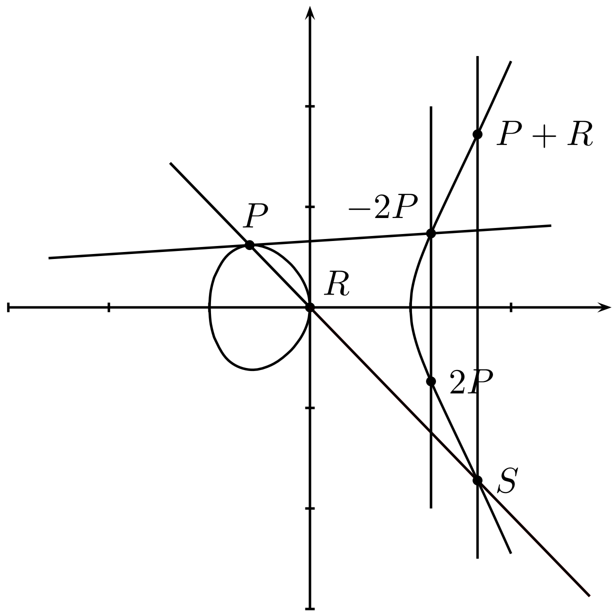
$$\begin{aligned} y_S &= \lambda x_S + \mu \\ &= \lambda x_S + y_P - \lambda x_P \\ &= \lambda(x_S - x_P) + y_P \end{aligned}$$

Point $P \oplus R$ has the same x-coordinate as S but negative y-coordinate:

$$x_{P \oplus R} = \lambda^2 - x_P - x_R, \quad y_{P \oplus R} = \lambda(x_P - x_{P \oplus R}) - y_P$$

Group Law (q odd)

$$E : y^2 = x^3 + a_4 x + a_6,\ a_i \in \mathbb{F}_q$$



In general, for $(x_P, y_P) \neq (x_R, -y_R)$:

$$\begin{aligned}(x_P, y_P) \oplus (x_R, y_R) &= \\ &= (x_{P\oplus R}, y_{P\oplus R}) = \\ &= (\lambda^2 - x_P - x_R, \lambda(x_P - x_{P\oplus R}) - y_P),\end{aligned}$$

where

$$\lambda = \begin{cases} (y_R - y_P)/(x_R - x_P) & \text{if } x_P \neq x_R, \\ (3x_P^2 + a_4)/(2y_P) & \text{else.} \end{cases}$$

$\Rightarrow$ Addition and Doubling need
1 I, 2M, 1S and 1 I, 2M, 2S, respectively

Weierstraß equation

$$E : y^2 + \underbrace{(a_1x + a_3)}_{h(x)}y = \underbrace{x^3 + a_2x^2 + a_4x + a_6}_{f(x)}, \; h, f \in \mathbb{F}_q[x].$$

- Negative of $P = (x_P, y_P)$ is given by $-P = (x_P, -y_P - h(x_P))$.
- $(x_P, y_P) \oplus (x_R, y_R) = (x_3, y_3) =$ $= (\lambda^2 + a_1\lambda - a_2 - x_P - x_R, \lambda(x_P - x_3) - y_P - a_1x_3 - a_3)$, where

$$\lambda = \begin{cases} (y_R - y_P)/(x_R - x_P) & \text{if } x_P \neq x_R, \\ \frac{3x_P^2+2a_2x_P+a_4-a_1y_P}{2y_P+a_Px_P+a_3} & \text{else.} \end{cases}$$

Projective Coordinates

$P = (X_1 : Y_1 : Z_1)$, $Q = (X_2 : Y_2 : Z_2)$, $P \oplus Q = (X_3 : Y_3 : Z_3)$
on $E : Y^2Z = X^3 + a_4XZ^2 + a_6Z^3$

Addition: $P \neq \pm Q$

$A = Y_2Z_1 - Y_1Z_2, B = X_2Z_1 - X_1Z_2,$

$C = A^2Z_1Z_2 - B^3 - 2B^2X_1Z_2$

$X_3 = BC, Z_3 = B^3Z_1Z_2$

$Y_3 = A(B^2X_1Z_2 - C) - B^3Y_1Z_2,$

Doubling $P = Q \neq -P$

$A = a_4Z_1^2 + 3X_1^2, B = Y_1Z_1,$

$C = X_1Y_1B, D = A^2 - 8C$

$X_3 = 2BD, Z_3 = 8B^3$.

$Y_3 = A(4C - D) - 8Y_1^2B^2$

No inversion is needed and the computation times are $12M + 2S$ for a general addition and $7M + 5S$ for a doubling.

Jacobian Coordinates

$P = (X_1 : Y_1 : Z_1)$, $Q = (X_2 : Y_2 : Z_2)$, $P \oplus Q = (X_3 : Y_3 : Z_3)$
on $Y^2 = X^3 + a_4 XZ^4 + a_6 Z^6$ by

Addition: $P \neq \pm Q$

$A = X_1 Z_2^2, B = X_2 Z_1^2, C = Y_1 Z_2^3,$
$D = Y_2 Z_1^3, E = B - A, F = D - C$
$X_3 = -E^3 - 2AE^2 + F^2, Z_3 = Z_1 Z_2 E,$
$Y_3 = -CE^3 + F(AE^2 - X_3),$

Doubling $P = Q \neq -P$

$A = 4X_1 Y_1^2, B = 3X_1^2 + a_4 Z_1^4$
$X_3 = -2A + B^2, Z_3 = 2Y_1 Z_1$
$Y_3 = -8Y_1^4 + B(A - X_3)$.

No inversion is needed and the computation times are
$12M + 4S$ for a general addition and $4M + 6S$ for a doubling.

Different coordinate systems $y^2 = x^3 + ax + b$

system	points	correspondence
affine ($\mathcal{A}$)	(x, y)	
projective ($\mathcal{P}$)	(X, Y, Z)	$(X/Z, Y/Z)$
jacobian ($\mathcal{J}$)	(X, Y, Z)	$(X/Z^2, Y/Z^3)$
Chudnovsky jacobian ($\mathcal{J}^C$)	(X, Y, Z, Z^2, Z^3)	$(X/Z^2, Y/Z^3)$
modified jacobian ($\mathcal{J}^m$)	(X, Y, Z, aZ^4)	$(X/Z^2, Y/Z^3)$

system	addition			doubling		
affine ($\mathcal{A}$)	2M	1S	1I	2M	2S	1I
projective ($\mathcal{P}$)	12M	2S	–	7M	5S	–
jacobian ($\mathcal{J}$)	12M	4S	–	4M	6S	–
Chudnovsky jacobian ($\mathcal{J}^C$)	11M	3S	–	5M	6S	–
modified jacobian ($\mathcal{J}^m$)	13M	6S	–	4M	4S	–

Mixed coordinates

(Cohen, Miyaji, Ono, Asiacrypt ’98)

affordable inversions:
precomputations in $\mathcal{A}$ (with Montgomery),
main doublings in $\mathcal{J}^m$,
final doublings $2\mathcal{J}^m = \mathcal{J}$,
additions $\mathcal{A} + \mathcal{J} = \mathcal{J}^m$

expensive inversions:
precomputations in $\mathcal{J}^C$,
main doublings in $\mathcal{J}^m$,
final doublings $2\mathcal{J}^m = \mathcal{J}$,
additions $\mathcal{J} + \mathcal{J}^C = \mathcal{J}^m$

Side-channel attacks

–

Why can't we always go for the fastest coordinate systems?

Side Channels

Attacker can measure

- Time to perform operations,
- Power consumption during operations,
- Electro-magnetic radiation during computation,
- Noise produced during computation.
- ...

Obviously, integer addition is cheaper than multiplication
$\Rightarrow$ needs more clock cycles, different characteristics of power trace.

Attacker might be able to reconstruct sequence of operations (power & EM) or at least learn how many of each kind were performed (timing).

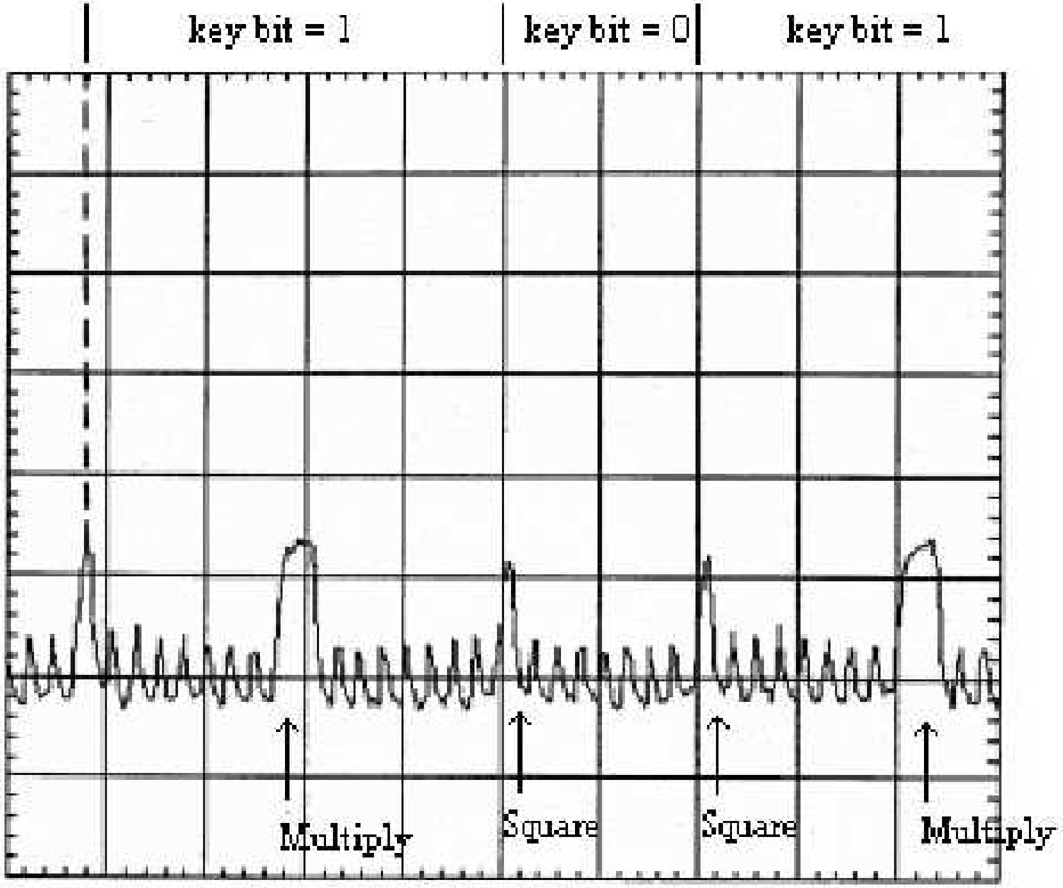
Consequences

If sequence of operations depends on the secret key and this is directly translated to the observed data, one can reconstruct the key

⇒ Simple Side-Channel Analysis (SSCA) (often SPA= Simple Power Analysis).

(e. g. in binary square-and-multiply one has

S M S S M $\sim$ $(1101)_2 = 13$).



Scalar Multiplication – Double-and-Add

IN: $P \in E(\mathbb{F}_q)$, $n \in \mathbb{Z}$, $n = \sum_{i=0}^{l} n_i 2^i$
OUT: $Q = nP$

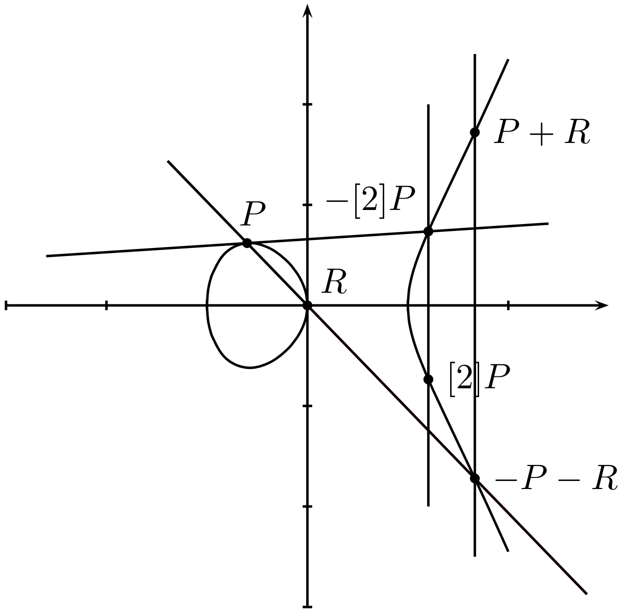
1. $Q = P$
2. for $i = l - 1$ down to 0 do
3. $\quad Q = 2Q$
4. $\quad$ if $(n_i = 1)$ then $Q = Q + P$
5. output Q

If ADD $\neq$ DBL one can easily determine n from the sequence of ADD and DBL:

DBL DBL ADD DBL ADD DBL DBL $\Leftrightarrow (101100)_2 = 44$

Weierstrass form (q odd)

$$E: y^2 = x^3 + a_4x + a_6,\ a_i \in \mathbb{F}_q$$



$$(x_1, y_1) + (x_2, y_2) =$$
$$= (x_3, y_3) =$$
$$= (\lambda^2 - x_1 - x_2, \lambda(x_1 - x_3) - y_1),$$

where

$$\lambda = \begin{cases} (y_2 - y_1)/(x_2 - x_1) & \text{if } x_1 \neq x_2, \\ (3x_1^2 + a_4)/(2y_1) & \text{else.} \end{cases}$$

$\Rightarrow$ Addition and Doubling differ considerably.

ADD: 1 I, 2M, 1S vs. DBL: 1 I, 2M, 2S

Unprotected arithmetic prone to SSCA.

Double-and-always-Add

This is the obvious countermeasure …

`IN:` $P \in E(\mathbb{F}_q)$, $n \in \mathbb{Z}$, $n = \sum_{i=0}^{l} n_i 2^i$

`OUT:` $Q = nP$

1. $Q = P, R = [2]P$
2. `for` $i = l - 1$ `down to` 0 `do`
3. $\qquad Q = [2]Q$
4. $\qquad$ `if` $n_i == 1$ `then` $Q = Q \oplus P$
   `else` $R = Q \oplus P$ //dummy operation
5. `output` Q

…but it is very inefficient.

Caution: If an active adversary is allowed, the dummy operations might be detected (fault attacks)

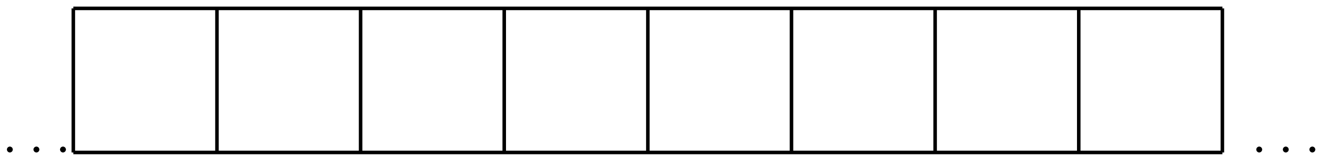
Common countermeasures

- Double-and-always-add
  very inefficient
- Side-channel atomicity (Chevallier-Mames, Ciet, Joye)
  build group operation from identical blocks.

  Each block consists of:

  1 multiplication, 1 addition, 1 negation, 1 addition;

  fill with cheap dummy additions and negations
  ADD $(\mathcal{A} + \mathcal{P})$ needs 11 blocks
  DBL $(2\mathcal{P})$ needs 10 blocks



- Brier and Joye, uniform Jacobian coordinates

Common countermeasures

- Double-and-always-add
  very inefficient
- Side-channel atomicity (Chevallier-Mames, Ciet, Joye)
  build group operation from identical blocks.

  Each block consists of:

  1 multiplication, 1 addition, 1 negation, 1 addition;

  fill with cheap dummy additions and negations
  ADD $(\mathcal{A}+\mathcal{P})$ needs 11 blocks
  DBL $(2\mathcal{P})$ needs 10 blocks

$\ldots$	ADD$_9$	ADD$_{10}$	ADD$_{11}$	DBL$_1$	DBL$_2$	DBL$_3$	DBL$_4$	DBL$_5$	$\ldots$

- Brier and Joye, uniform Jacobian coordinates

Uniform Group Operations

- Liardet and Smart CHES 2001: Jacobi intersection
- Billet and Joye AAECC 2003: Jacobi-Model

$$E_J : Y^2 = \epsilon X^4 - 2\delta X^2 Z^2 + Z^4.$$

- Joye and Quisquater suggested Hessian Curves

$$E_H : X^3 + Y^3 + Z^3 = cXYZ.$$

They achieve uniformity by

$$[2](X_1 : Y_1 : Z_1) = (Z_1 : X_1 : Y_1) + (Y_1 : Z_1 : X_1)$$

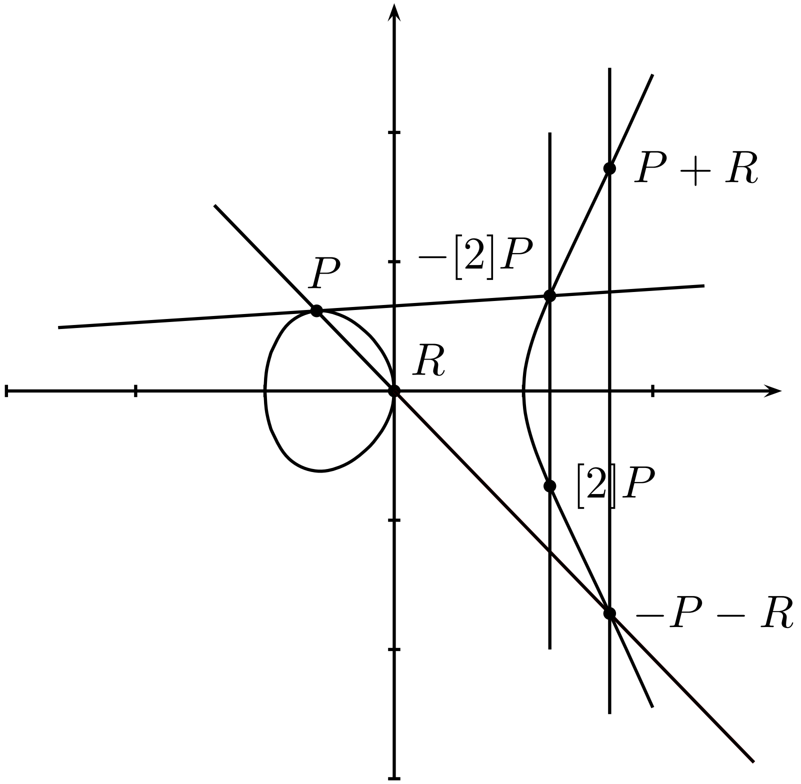
and $(Z_1 : X_1 : Y_1) \neq (Y_1 : Z_1 : X_1)$.

Edwards coordinates

Addition on Elliptic Curves

At Mathematics: Algorithms and Proofs in Leiden, January 2007, Harold M. Edwards gave a talk on Addition on Elliptic Curves

So Dan and I expected . . .



Addition on Elliptic Curves

At Mathematics: Algorithms and Proofs in Leiden, January 2007, Harold M. Edwards gave a talk on Addition on Elliptic Curves

But there it was – the elliptic curve:

$$x^2 + y^2 = a^2(1 + x^2y^2).$$

Addition on Elliptic Curves

At Mathematics: Algorithms and Proofs in Leiden, January 2007, Harold M. Edwards gave a talk on Addition on Elliptic Curves

But there it was – the elliptic curve:

$$x^2 + y^2 = a^2(1 + x^2y^2).$$

Nonsingular if and only if $a^5 \neq a$.

Addition on Elliptic Curves

At Mathematics: Algorithms and Proofs in Leiden, January 2007, Harold M. Edwards gave a talk on Addition on Elliptic Curves

But there it was – the elliptic curve:

$$x^2 + y^2 = a^2(1 + x^2y^2).$$

Nonsingular if and only if $a^5 \neq a$.
To see that this is indeed an elliptic curve, use $z = y(1 - a^2x^2)/a$ to obtain

$$z^2 = x^4 - (a^2 + 1/a^2)x^2 + 1.$$

Edwards' Addition Formulae

- $P=(x_P,y_P), Q=(x_Q,y_Q)$ on $x^2+y^2=a^2(1+x^2y^2)$.
- $$P+Q=\left(\frac{x_Py_Q+y_Px_Q}{a(1+x_Px_Qy_Py_Q)},\frac{y_Py_Q-x_Px_Q}{a(1-x_Px_Qy_Py_Q)}\right).$$
- $$\begin{aligned}[2]P &= \left(\frac{x_Py_P+y_Px_P}{a(1+x_Px_Py_Py_P)},\frac{y_Py_P-x_Px_P}{a(1-x_Px_Py_Py_P)}\right)\\ &= \left(\frac{2x_Py_P}{a(1+(x_Py_P)^2)},\frac{y_P^2-x_P^2}{a(1-(x_Py_P)^2)}\right).\end{aligned}$$
- For much more information on elliptic curves in this shape see Edwards 2007 paper in Bull. AMS., electronic April 9.

Following results are joint work with Daniel J. Bernstein

Edwards form

- Slightly generalized shape:

$$E_E : x^2 + y^2 = c^2(1 + dx^2y^2)$$

  is elliptic curve for $c, d \neq 0$ and $dc^4 \neq 1$.

- Affine formulae

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1y_2 + y_1x_2}{c(1 + dx_1x_2y_1y_2)}, \frac{y_1y_2 - x_1x_2}{c(1 - dx_1x_2y_1y_2)} \right).$$

- Projective version takes

  10M + 1S + 1C + 1D + 7A,

  where C is the cost of multiplying by c, D is the cost of multiplying by d, and A abbreviates addition.

Comparison of unified formulae

System	Cost of unified addition-or-doubling
Jacobian	11M+6S+1C; see Brier/Joye '03
Jacobian if $a_4 = -1$	13M+3S; see Brier/Joye '02
Jacobi intersection	13M+2S+1C; see Liardet/Smart '01
Jacobi quartic	10M+3S+3C; see Billet/Joye '01
Hessian	12M; see Joye/Quisquater '01
Edwards	10M+1S+1C

- Fastest unified addition-or-doubling formulae.
- Exactly the same formulae for doubling (no re-arrangement like in Hessian)
- No exceptional cases – affine input produces correct affine output – if d is not a square, i.e. no points with $dx_1x_2y_1y_2 = \pm 1$.

Multi-scalar multiplication

Idea of joint doublings

- To compute $[n_1]P_1 \oplus [n_2]P_2 \oplus \cdots \oplus [n_m]P_m$ compute the doublings together, i.e. write scalars n_i in binary:

$$
\begin{array}{cccccccc}
n_1 & = & n_{1,l-1}2^{l-1} & +n_{1,l-2}2^{l-2} & +n_{1,l-3}2^{l-3} & \ldots & +n_{1,1}2 & +n_{1,0} \\
n_2 & = & n_{2,l-1}2^{l-1} & +n_{2,l-2}2^{l-2} & +n_{2,l-3}2^{l-3} & \ldots & +n_{2,1}2 & +n_{2,0} \\
\vdots & = & \vdots & \vdots & \vdots & & \vdots & \vdots \\
n_m & = & n_{m,l-1}2^{l-1} & +n_{m,l-2}2^{l-2} & +n_{m,l-3}2^{l-3} & \ldots & +n_{m,1}2 & +n_{m,0}
\end{array}
$$

Idea of joint doublings

- To compute $[n_1]P_1 \oplus [n_2]P_2 \oplus \cdots \oplus [n_m]P_m$ compute the doublings together, i.e. write scalars n_i in binary:

$$
\begin{array}{ccllllll}
n_1 & = & n_{1,l-1}2^{l-1} & +n_{1,l-2}2^{l-2} & +n_{1,l-3}2^{l-3} & \dots & +n_{1,1}2 & +n_{1,0} \\
n_2 & = & n_{2,l-1}2^{l-1} & +n_{2,l-2}2^{l-2} & +n_{2,l-3}2^{l-3} & \dots & +n_{2,1}2 & +n_{2,0} \\
\vdots & = & \vdots & \vdots & \vdots & & \vdots & \vdots \\
n_m & = & n_{m,l-1}2^{l-1} & +n_{m,l-2}2^{l-2} & +n_{m,l-3}2^{l-3} & \dots & +n_{m,1}2 & +n_{m,0}
\end{array}
$$

- Compute as

$$[2](\underbrace{[n_{1,l-1}]P_1 \oplus [n_{2,l-1}]P_2 \oplus [n_{3,l-1}]P_3 \oplus \cdots \oplus [n_{m,l-1}]P_m}_{\text{first column}})$$

Idea of joint doublings

- To compute $[n_1]P_1 \oplus [n_2]P_2 \oplus \cdots \oplus [n_m]P_m$ compute the doublings together, i.e. write scalars n_i in binary:

$$\begin{array}{ccllllll} n_1 & = & n_{1,l-1}2^{l-1} & +n_{1,l-2}2^{l-2} & +n_{1,l-3}2^{l-3} & \ldots & +n_{1,1}2 & +n_{1,0} \\ n_2 & = & n_{2,l-1}2^{l-1} & +n_{2,l-2}2^{l-2} & +n_{2,l-3}2^{l-3} & \ldots & +n_{2,1}2 & +n_{2,0} \\ \vdots & = & \vdots & \vdots & \vdots & & \vdots & \vdots \\ n_m & = & n_{m,l-1}2^{l-1} & +n_{m,l-2}2^{l-2} & +n_{m,l-3}2^{l-3} & \ldots & +n_{m,1}2 & +n_{m,0} \end{array}$$

- Compute as

$$[2]\left([2]([n_{1,l-1}]P_1 \oplus [n_{2,l-1}]P_2 \oplus [n_{3,l-1}]P_3 \oplus \cdots \oplus [n_{m,l-1}]P_m) \oplus \right.$$
$$\left.([n_{1,l-2}]P_1 \oplus [n_{2,l-2}]P_2 \oplus [n_{3,l-2}]P_3 \oplus \cdots \oplus [n_{m,l-2}]P_m\right) \oplus$$
$\cdots$ etc.

- Needs many more additions than doublings, even with precomputations.

Applications

- ECDSA verification uses 2 scalar multiplications ... just to add the results.
- If base point P is fixed, precompute $R = [2^{l/2}]P$ and include in the curve parameters. Split scalar $n = n_1 2^{l/2} + n_0$ and compute

$$[n_1]R \oplus [n_0]P.$$

- GLV curves split scalar in two halves to get faster scalar multiplication.
- Verification in accelerated ECDSA can be extended to use 4 or even 6 scalars. Splitting of the scalar is done by LLL techniques
- Further applications in batch verification of signatures – many scalars – by taking random linear combinations.

Comparison – 1 DBL & 0.5 mixed ADD

System	Cost of 1 DBL & 0.5 mixed ADD
Projective	10.5M+6S+1C $\approx$ 15.3M
Edwards	10.5M+4.5S+1.5C $\approx$ 14.1M
Jacobi quartic	5M+10.5S+4.5C $\approx$ 13.4M
Hessian	11M+3S $\approx$ 13.4M
Jacobian	6M+8.5S+1C $\approx$ 12.8M
Jacobi intersection	9.5M+4S+0.5C $\approx$ 12.7M
Jacobian/Chudnovsky if $a_4 = -3$	7M+6.5S $\approx$ 12.2M

1 DBL & 0.75 ADD & 0.75 mixed ADD

System	Cost of 1DBL & 0.75 ADD & 0.75 mixed ADD
Projective	21.75M+8S+1C $\approx$ 28.15M
Jacobi intersection	22M+6S+1.5C $\approx$ 26.8M
Jacobian	16.25M+13S+1C $\approx$ 26.65M
Jacobian if $a_4 = -3$	17.25M+11S $\approx$ 26.05M
Jacobi quartic	14.5M+13.5S+7.5C $\approx$ 25.3M
Hessian	22.5M+3S $\approx$ 24.9M
Chudnovsky if $a_4 = -3$	16.5M+10.25S $\approx$ 24.7M
Edwards	20.25M+5.5S+2.5C $\approx$ 24.65M

Results

- Most coordinate systems optimized for many doublings, few additions (single scalar multiplication with windowing).
- Projective Edwards formulae offer best speed for addition and are not bad for doubling either.
- Edwards coordinates are an ideal system for batch verification.

Results

- Most coordinate systems optimized for many doublings, few additions (single scalar multiplication with windowing).
- Projective Edwards formulae offer best speed for addition and are not bad for doubling either.
- Edwards coordinates are an ideal system for batch verification.
- Anybody need unified, SSCA resistant multi-scalar multiplication???

The end

http://cr.yp.to/papers.html#newelliptic