Hardware Arithmetic Operators for Elliptic Curve Cryptography (ECC)

Arnaud Tisserand

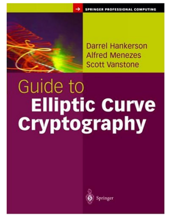
CNRS, IRISA laboratory, CAIRN research team

Journée Sécurité Numérique, GDR SoC-SiP
Paris, November 16th, 2011

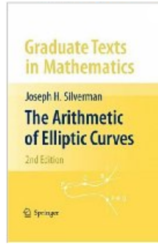
References on Elliptic Curves

Most of examples/notations used in this presentation come from:

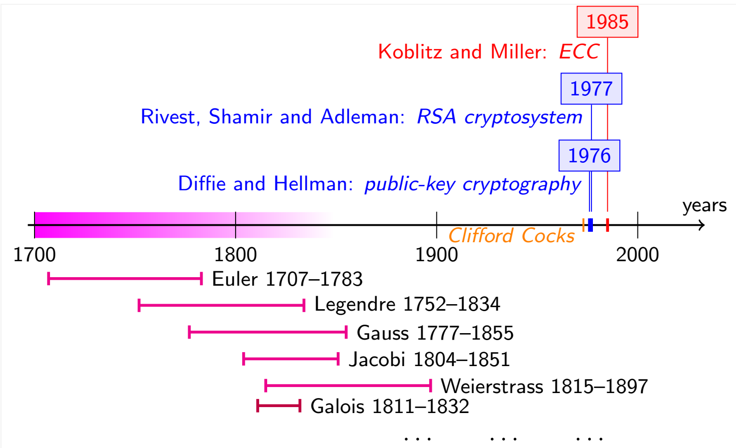
Guide to Elliptic Curve Cryptography
D. Hankerson, A. Menezes and S. Vanstone
2004. Springer
ISBN: 0–387–95273–X



The Arithmetic of Elliptic Curves
Joseph H. Silverman
2009. Springer
ISBN: 978-0-387-09493-9



Some Historical Aspects



Question in the 18th century: arc length of an ellipse?
$\rightsquigarrow$ study of integrals involving $\sqrt{f(x)}$ where $\deg f \in \{3,4\}$

Notations

- Elliptic curve E
- Underlying field K ($\mathbb{R}$, $\mathbb{F}_p$, $\mathbb{F}_{2^m}$, ...)
- Finite field $\mathbb{F}_q$ ($q = p$ or $q = 2^m$ in this presentation)
- Points P, Q, ...
- Coordinates $(x, y, [z])$ $(x, y, [z] \in K)$
- Point at infinity denoted ∞
- Number of points on E: $\#E$

Elliptic Curves

Set of points (x, y) defined by the Weierstrass equation:

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

where

- $a_1, a_2, a_3, a_4, a_6 \in K$
- discriminant of E: $\Delta \neq 0$ and

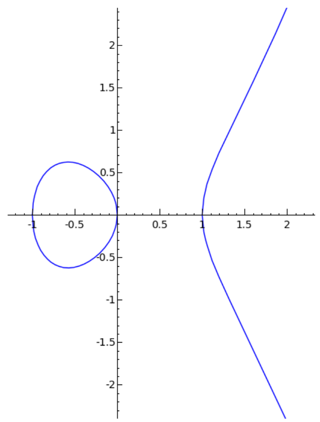
$$\begin{aligned}
\Delta &= -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \\
d_2 &= a_1^2 + 4a_2 \\
d_4 &= 2a_4 + a_1a_3 \\
d_6 &= a_3^2 + 4a_6 \\
d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2
\end{aligned}$$

Condition $\Delta \neq 0$ ensures that E is smooth

Set of points where ∞ denotes the point at infinity:
$E(K) = \{(x, y) \in K \times K; y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{\infty\}$

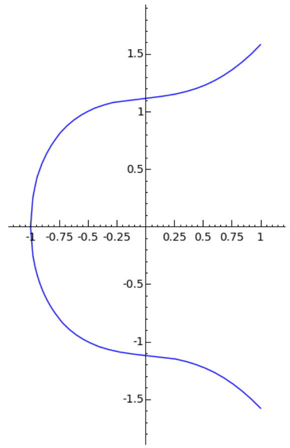
Elliptic Curves Examples on $\mathbb{R}$

$ER_1 : y^2 = x^3 - x$



$(a_1, a_2, a_3, a_4, a_6) = (0, 0, 0, -1, 0)$

$ER_2 : y^2 = x^3 + \frac{x}{4} + \frac{5}{4}$



$(a_1, a_2, a_3, a_4, a_6) = (0, 0, 0, \frac{1}{4}, \frac{5}{4})$

Group Law

Point addition using the *chord-and-tangent rule*: the addition of 2 points of E gives a third point also on E

$$P + Q \quad \text{and} \quad P + P = [2]P$$

Elliptic curves as algebraic objects: $(E, +)$ forms an abelian group

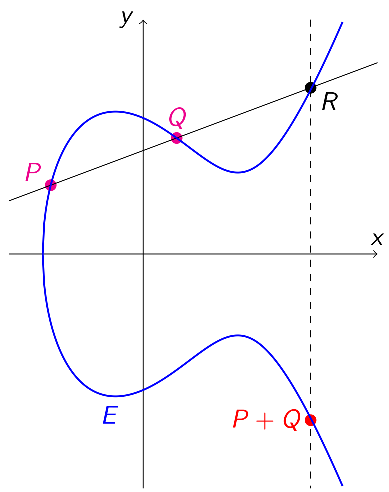
The set of points on E (over field K) and the "point addition" operation forms an abelian group with ∞ as its identity

- $P + \infty = \infty + P = P$
- $P + (-P) = \infty$
- $(P + Q) + R = P + (Q + R)$
- $P + Q = Q + P$

Abelian groups in public-key cryptography:

- operation on the group should be easy to implement
- computation of the discrete logarithm on the group should be hard

Point Addition $P + Q$



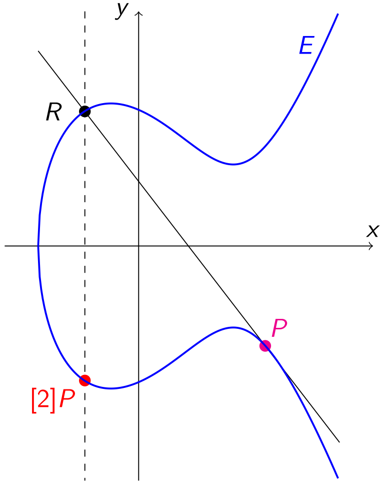
Geometrical explanation:

1. draw P and Q
2. draw the line through P and Q, this line intersects E on a **third point** R
3. $P + Q$ is the reflection of R w.r.t. the x-axis.

Point at infinity:

$$P + Q + R = \infty$$

Point Doubling [2]P



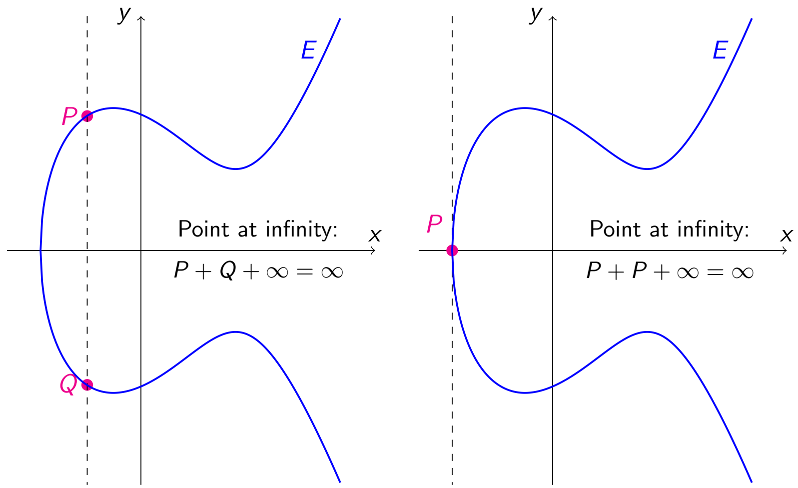
Geometrical explanation:

1. draw P
2. draw the tangent to E at point P, this tangent intersects E on a **second point** R
3. $[2]P$ is the reflection of R w.r.t. the x-axis.

Point at infinity:

$$P + P + R = \infty$$

Specific Cases



Addition and Doubling Equations

Notations:

- elliptic curve $E : y^2 = x^3 + ax + b$
- P coordinates (x_1, y_1)
- Q coordinates (x_2, y_2)

The slope of line (P, Q) is

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq \pm Q \qquad [ADD] \\[2ex] \dfrac{3x_1^2 + a}{2y_1} & \text{if } P = Q \qquad [DBL] \end{cases}$$

The addition $P + Q$ (or doubling $[2]P$) gives the point (x_3, y_3) where:

$$x_3 = \lambda^2 - x_1 - x_2 \quad \text{and} \quad y_3 = \lambda(x_1 - x_3) - y_1$$

Simplified Weierstrass Equations

Depending on the characteristic of the field K, the equation can be significantly simplified.

Characteristic p: with $p \notin \{2, 3\}$, fields $\mathbb{F}_p$

$$y^2 = x^3 + ax + b \quad \text{and} \quad \Delta = -16(4a^3 + 27b^2) \neq 0$$

Characteristic 2: fields $\mathbb{F}_{2^m}$

$a_1 \neq 0$: *non-supersingular* curve

$$y^2 + xy = x^3 + ax^2 + b \quad \text{and} \quad \Delta = b \neq 0$$

$a_1 = 0$: *supersingular* curve

$$y^2 + cy = x^3 + ax + b \quad \text{and} \quad \Delta = c^4 \neq 0$$

Characteristic 3: fields $\mathbb{F}_{3^m}$

$a_1^2 \neq -a_2$: *non-supersingular* curve

$$y^2 = x^3 + ax^2 + b \quad \text{and} \quad \Delta = -a^3 b \neq 0$$

$a_1^2 = -a_2$: *supersingular* curve

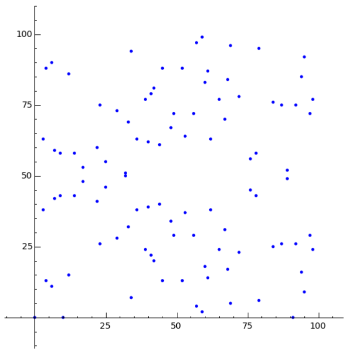
$$y^2 = x^3 + ax + b \quad \text{and} \quad \Delta = -a^3 \neq 0$$

Notation: $a, b, c \in K$

Elliptic Curves Examples on $\mathbb{F}_{101}$

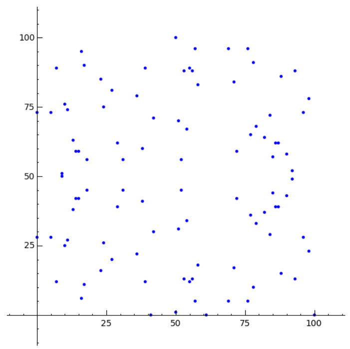
For $K = \mathbb{F}_{101}$:

$EF_1 : y^2 = x^3 - 100x$



$(a_1, a_2, a_3, a_4, a_6) = (0, 0, 0, -100, 0)$

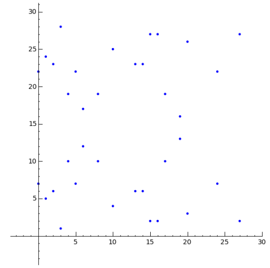
$EF_2 : y^2 = x^3 + 76x + 77$



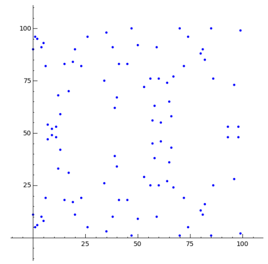
$(a_1, a_2, a_3, a_4, a_6) = (0, 0, 0, 76, 77)$

Elliptic Curve $y^2 = x^3 + 4x + 20$, $(0, 0, 0, 4, 20)$

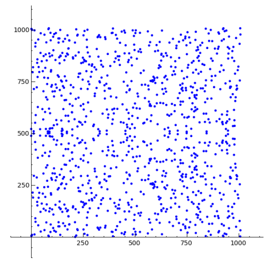
$\mathbb{F}_{29}$
37



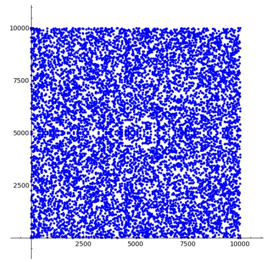
$\mathbb{F}_{101}$
99



$\mathbb{F}_{1009}$
1004



$\mathbb{F}_{10007}$
10055



Number of Points in the Elliptic Curve

Notations:

- $\mathbb{F}_q$ is a finite field ($q = p$ or $q = 2^m$ in this presentation)
- $\#E$ is the number of points in E over $\mathbb{F}_q$ (also called the *order* of E over $\mathbb{F}_q$)

First bounds: Weierstrass equation has at most 2 solutions for each $x \in \mathbb{F}_q$ then

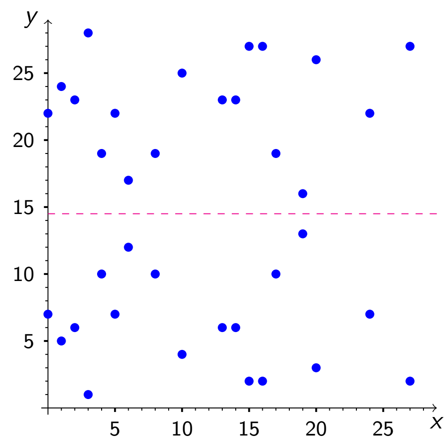
$$1 \leq \#E \leq 2q + 1$$

Tighter bounds: Hasse's theorem bounds $\#E$ of an elliptic curve over a finite field $\mathbb{F}_q$

$$q + 1 - 2\sqrt{q} \leq \#E \leq q + 1 + 2\sqrt{q}$$

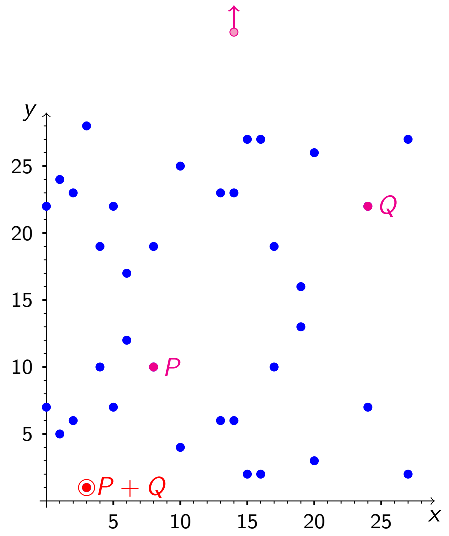
In practice, $\#E$ is close to q

Example $E : y^2 = x^3 + 4x + 20$ on $\mathbb{F}_{29}$ (1/4)



There are 37 points on E:
∞ and
$(0, 7), (0, 22), (1, 5),$
$(1, 24), (2, 6), (2, 23),$
$(3, 1), (3, 28), (4, 10),$
$(4, 19), (5, 7), (5, 22),$
$(6, 12), (6, 17), (8, 10),$
$(8, 19), (10, 4), (10, 25),$
$(13, 6), (13, 23), (14, 6),$
$(14, 23), (15, 2), (15, 27),$
$(16, 2), (16, 27), (17, 10),$
$(17, 19), (19, 13), (19, 16),$
$(20, 3), (20, 26), (24, 7),$
$(24, 22), (27, 2), (27, 27)$

Example $E : y^2 = x^3 + 4x + 20$ on $\mathbb{F}_{29}$ (2/4)



Point addition example:
$P = (8, 10)$
$Q = (24, 22)$

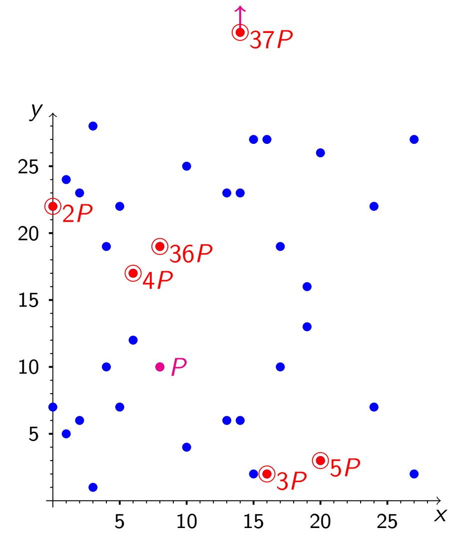
$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{22 - 10}{24 - 8} = 8$$

$$\begin{cases} x_3 &= \lambda^2 - x_1 - x_2 \\ &= 8^2 - 8 - 24 = 3 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \\ &= 8 \times (8 - 3) - 10 = 1 \end{cases}$$

Verification using sage:

```
P = EEF29(8,10)
Q = EEF29(24,22)
P+Q
```


Example $E : y^2 = x^3 + 4x + 20$ on $\mathbb{F}_{29}$ (3/4)



Point doubling example:
$P = (8, 10)$

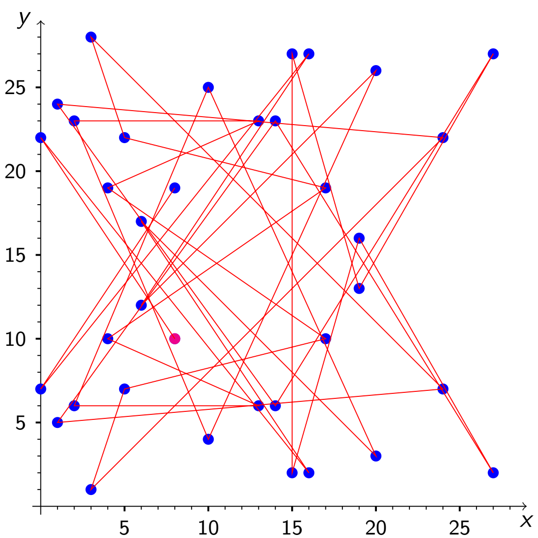
$$\lambda = \frac{3x_1^2 + a}{2y_1} = \frac{3 \cdot 8^2 + 4}{2 \cdot 10} = 4$$

$$\begin{cases} x_3 &= \lambda^2 - x_1 - x_2 \\ &= 4^2 - 8 - 8 = 0 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \\ &= 4 \times (8 - 0) - 10 = 22 \end{cases}$$

Verification using sage:

```
P = EEF29(8,10)
2*P
```


Example $E : y^2 = x^3 + 4x + 20$ on $\mathbb{F}_{29}$ (4/4)



Scalar Multiplication $Q = kP$

Point multiplication or scalar multiplication:

Inputs: a point $P \in E$ and $k \in \mathbb{N}$
Output: the point $E \ni Q = kP = \underbrace{P + P + \ldots + P}_{k \text{ times}}$ (also denoted $[k]P$)

This is the main operation in ECC protocols

Choice for k:

- $\#E(\mathbb{F}_q) = nh$ where n is prime and h is small ($n \approx q$)
- k random integer in $[1, n-1]$
- k binary representation $(k_{t-1}k_{t-2}...k_1k_0)_2$ where $t \approx \lceil \log_2 q \rceil$

Remark: computing efficiently multiple point multiplication $[k]P + [l]Q$ may be useful in some protocols

Discrete Logarithm Problems

Discrete logarithm problem (DLP) on a group G:

Inputs: $a, b \in (G, \times)$
Output: the smallest integer x (> 0) such that $a = b^x$ (if it exists)

Remark: $\#G$ prime $\Longrightarrow$ a discrete logarithm always exists

---

Elliptic curve discrete logarithm problem (ECDLP):

Inputs: $P, Q \in E \mid Q = kP$
Output: the scalar k (long integer), k is the discrete logarithm of Q to the base P

Given P and Q, it is computationally infeasible to obtain k, if k is large enough.

Key Size vs Security Level

security level	RSA	ECC		
	$\|n\|$ [bits]	$\mathbb{F}_p$ $\|p\|$ [bits]	$\mathbb{F}_{2^m}$ m [bits]	
56	512	112	113	
64	704	128	131	
80	1024	160	163	◀
96	1536	192	193	
112	2048	224	233	◀◀
128	3072	256	283	
192	7680	384	409	
256	15360	521	571	

- Security level of h: the best known algorithm takes 2^h steps for breaking the cryptosystem
- RSA: $\mathbb{Z}/n\mathbb{Z}$ with $n = pq$, p and q primes
- ECC: $\mathbb{F}_p$ with p prime or $\mathbb{F}_{2^m}$

Source: SEC2 recommendations from Certicom (v1.0, Jan. 2000)

ECC Challenge (1/2)

Source: `http://www.certicom.com/index.php/the-certicom-ecc-challenge`

Challenge: compute ECC private key from ECC public key and parameters (ECDLP)

challenge	end date	machine days[1]
ECC2-79	Dec. 16, 1997	116
ECC2-89	Feb. 9, 1998	1114
ECC2K-95	May 21, 1998	1709
ECC2-97	Sep. 22, 1999	6118
ECC2K-108	Apr. 4, 2000	166000
ECC2-109	Apr. 8, 2004	
ECCp-79	Dec. 6, 1997	52
ECCp-89	Jan. 12, 1998	716
ECCp-97	Mar. 18, 1998	6412
ECCp-109	Oct. 15, 2002	

[1] Machine days on a 500 MHz alpha workstation.

ECC Challenge (2/2)

New record:

- Challenge: 112 bits (curve secp112r1)
- Dates: 2009.01.13 – 2009.07.08
- Support: 200 PlayStation 3 game consoles
- Location: EPFL
- Corresponding publication:
  J.W. Bos, M.E. Kaihara, T. Kleinjung, A.K. Lenstra and P.L. Montgomery. Solving a 112-bit Prime Elliptic Curve Discrete Logarithm Problem on Game Consoles using Sloppy Reduction. Int. J. Applied Cryptography, 2011.

Source: `http://lacal.epfl.ch/112bit_prime`

Guidelines for Designing “Robust” Cryptosystems

Use recommendations/standards from specialists...

Example : elliptic curve P-521 over a prime finite field, recommendation from NIST (cf. FIPS 186-2)

$p =$	6864797660130609714981900799081393217269435300143305409394463459185543183397656052122559640661454554977296311391480858037121987999716643812574028291115057151
$r =$	6864797660130609714981900799081393217269435300143305409394463459185543183397655394245057746333217197532963996371363321113864768612440380340372808892707005449
$s =$	d09e8800 291cb853 96cc6717 393284aa a0da64ba
$c =$	0b4 8bfa5f42 0a349495 39d2bdfc 264eeeeb 077688e4 4fbf0ad8 f6d0edb3 7bd6b533 28100051 8e19f1b9 ffbe0fe9 ed8a3c22 00b8f875 e523868c 70c1e5bf 55bad637
	$\vdots$

ECC Protocols

Applications:

- encryption
- digital signature
- key agreement

ECC protocols:

- ECIES: Elliptic Curve Integrated Encryption System
- ECDSA: Elliptic Curve Digital Signature Algorithm
- ECDH: Elliptic Curve Diffie-Hellman key agreement
- ...

Notation: D is the set of domain parameters $(E, q, \#E = nh, P \in E, \ldots)$

Elliptic Curve Digital Signature Algorithm

Preprocessing: select random integer $d \in [1, n-1]$, compute $Q = dP$ where $P \in E \implies Q$ public key and d private key

Signature: m is the message, H is the hash function

1. select random integer $k \in [1, n-1]$
2. $(x_1, y_1) = kP$, $r = x_1 \bmod n$, if $r = 0$ then step 1
3. $e = H(m)$, $s = k^{-1}(e + dr) \bmod n$, if $s = 0$ then step 1
4. return (r, s)

Verification:

1. if (r or s not in $[1, n-1]$) then REJECT
2. $e = H(m)$, $w = s^{-1} \bmod n$, $u_1 = ew \bmod n$, $u_2 = rw \bmod n$, $X = (x_1, y_1) = u_1 P + u_2 Q$
3. if $X = \infty$ then REJECT
4. $v = x_1 \bmod n$
5. if $v = r$ then ACCEPT else REJECT

ECC Implementation: Delay Estimation

Counting the number of point operations:

- Point addition $P + Q$ (ADD)
- Point doubling $2P$ (DBL)

Counting the number of field operations:

- addition/subtraction (A)
- multiplication (M)
- squaring (S)
- inversion (I)

Common assumptions for high-level estimation:

- $A \approx 0$
- $S \approx 0.8M$ for $\mathbb{F}_p$ and $S \approx 0$ for $\mathbb{F}_{2^m}$
- $I \approx 30M$

Scalar Multiplication: Double-and-Add Algorithms

Input: $P \in E$, $k = (k_{t-1}k_{t-2}..., k_1k_0)_2 \in \mathbb{N}$
Output: $Q = kP$

1: $Q \longleftarrow \infty$
2: **for** i **from** 0 **to** t-1 **do**
3: **if** $k_i = 1$ **then** $Q \longleftarrow Q + P$ — *ADD*
4: $P \longleftarrow 2P$ — *DBL*

Input: $P \in E$, $k = (k_{t-1}k_{t-2}..., k_1k_0)_2 \in \mathbb{N}$
Output: $Q = kP$

1: $Q \longleftarrow \infty$
2: **for** i **from** t-1 **downto** 0 **do**
4: $Q \longleftarrow 2Q$ — *DBL*
3: **if** $k_i = 1$ **then** $Q \longleftarrow Q + P$ — *ADD*

Double-and-Add Analysis

Assumption on the density of k due to security aspects:

number of 1 in k is $\approx \frac{t}{2}$

Point operations:

$$\frac{t}{2} \cdot ADD + t \cdot DBL$$

Cost of DBL and ADD point operations:

- $DBL \approx I + 2 \cdot M + 2 \cdot S$
- $ADD \approx I + 2 \cdot M + S$

Field operations:

$$\frac{3}{2}t \cdot I + 3t \cdot M + \frac{5}{2}t \cdot S$$

Estimation using previous assumptions:

- $\mathbb{F}_p$: $\text{cost}(kP) \approx 50t \cdot M$
- $\mathbb{F}_{2^m}$: $\text{cost}(kP) \approx 48t \cdot M$

Optimization

Q: Inversions are very expensive, can we remove them?

A: Yes, by changing the representation of the points

In some different coordinate systems, points on a curve can be added without inversions

$$(x, y) \longrightarrow (X, Y, Z)$$

Transformation: x is replaced by X/Z^c and y is replaced by Y/Z^d

Several coordinates systems are used in practice (several transformations and parameters $c, d \in \mathbb{N}^*$)

Remark: affine coordinates are the basic coordinates (x, y)

Projective Coordinates

Equivalence relation $\sim$ on the set $K^3 \setminus (0,0,0)$:

$$(X_1, Y_1, Z_1) \sim (X_2, Y_2, Z_2)$$

$$\text{if } X_1 = \lambda^c X_2, Y_1 = \lambda^d Y_2 \text{ and } Z_1 = \lambda Z_2 \text{ for some } \lambda \in K^*$$

Equivalence class $(X, Y, Z) \in K^3 \setminus (0,0,0)$, projective point:

$$(X : Y : Z) = \{(\lambda^c X, \lambda^d Y, \lambda Z) : \lambda \in K^*\}$$

Example: projective form of the Weierstrass equation using standard projective coordinates ($c = 1$, $d = 1$):

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

becomes

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

Examples of Coordinates Systems

- Affine coordinates, $\mathcal{A}$:

$$P : (x, y) \qquad \infty$$

- Standard projective coordinates, $\mathcal{P}$ ($c = 1, d = 1$):

$$P : (X, Y, Z) \qquad x = \frac{X}{Z}, y = \frac{Y}{Z} \qquad \infty = (0, 1, 0)$$

- Jacobian projective coordinates, $\mathcal{J}$ ($c = 2, d = 3$):

$$P : (X, Y, Z) \qquad x = \frac{X}{Z^2}, y = \frac{Y}{Z^3} \qquad \infty = (1, 1, 0)$$

- Chudnovsky coordinates, $\mathcal{C}$:

$$P : (X, Y, Z, Z^2, Z^3) \qquad \infty = (1, 1, 0)$$

- ...

Remark: $-(X, Y, Z) = (X, -Y, Z)$

Point Addition and Doubling Costs

- Point doubling

$$\begin{aligned} 2\mathcal{A} \to \mathcal{A} &\approx 1 \cdot I + 2 \cdot M + 2 \cdot S \\ 2\mathcal{P} \to \mathcal{P} &\approx 7 \cdot M + 3 \cdot S \\ 2\mathcal{J} \to \mathcal{J} &\approx 4 \cdot M + 4 \cdot S \\ 2\mathcal{C} \to \mathcal{C} &\approx 5 \cdot M + 4 \cdot S \end{aligned}$$

- Point addition

$$\begin{aligned} \mathcal{A} + \mathcal{A} \to \mathcal{A} &\approx 1 \cdot I + 2 \cdot M + 1 \cdot S \\ \mathcal{P} + \mathcal{P} \to \mathcal{P} &\approx 12 \cdot M + 2 \cdot S \\ \mathcal{J} + \mathcal{J} \to \mathcal{J} &\approx 12 \cdot M + 4 \cdot S \\ \mathcal{C} + \mathcal{C} \to \mathcal{C} &\approx 11 \cdot M + 3 \cdot S \\ \\ \mathcal{J} + \mathcal{A} \to \mathcal{J} &\approx 8 \cdot M + 3 \cdot S \\ \mathcal{J} + \mathcal{C} \to \mathcal{J} &\approx 11 \cdot M + 3 \cdot S \\ \mathcal{C} + \mathcal{A} \to \mathcal{C} &\approx 8 \cdot M + 3 \cdot S \end{aligned}$$

More Information on Coordinates Systems and Implementations

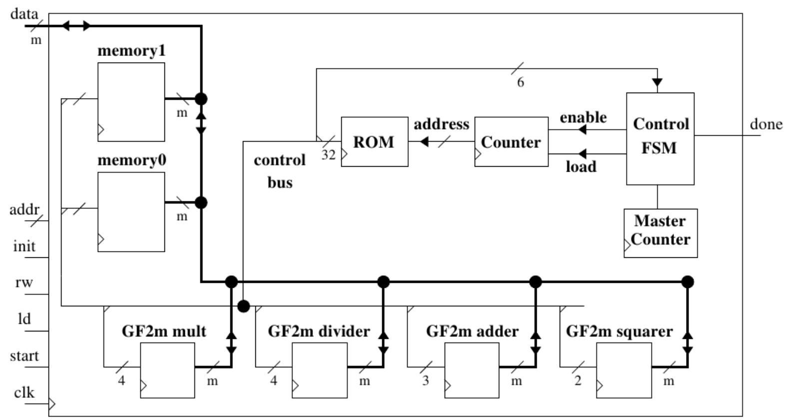
- Paper from D. Bernstein and T. Lange on *Analysis and optimization of elliptic-curve single-scalar multiplication* (PDF on the web)

- Explicit-Formulas Database (EFD): `http://www.hyperelliptic.org/EFD`
  - Collection of explicit formulas (point addition, doubling and tripling) for many coordinate systems
  - Best formulas from the literature
  - Code (sage) for validation purpose

- Proceedings of the workshops on Cryptographic Hardware and Embedded Systems (CHES): `http://www.iacr.org/workshops/ches/` (full-text access via Springer)

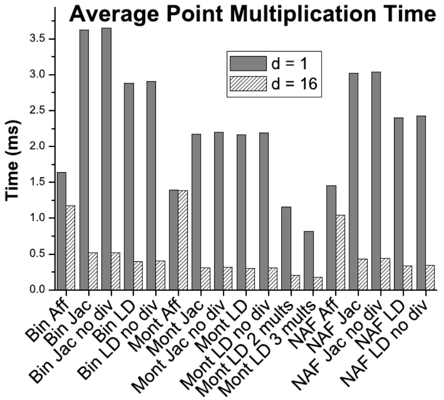
Implementation Example from UCC-CSI (1/3)

Source: Liam Marnane (University College Cork and Claude Shannon Institute), invited talk at ECC 2007: *Comparing Hardware Complexity of Cryptographic Algorithms*

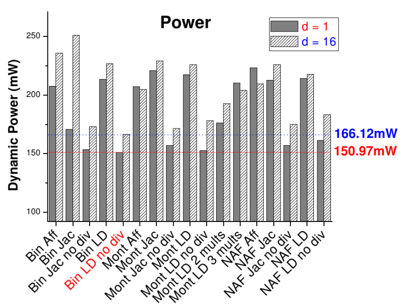


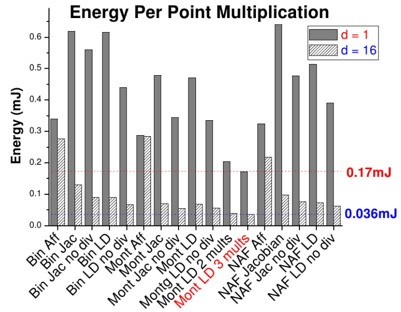
Implementation Example from UCC-CSI (2/3)

- $\mathbb{F}_{2^m}$, $m = 163$, NIST curve, target Xilinx xc3s1000l FPGA
- $\mathbb{F}_{2^m}$ mult.: digit size $d = 1$ ($\approx$ 3000 LUT) or $d = 16$ ($\approx$ 5100 LUT)
- $\mathbb{F}_{2^m}$ divider ($\approx$ 1100 LUT)
- freq: 80 MHz, static power: 92 mW



Implementation Example from UCC-CSI (3/3)





Summary (scalar mutl. using Montgomery ladder):

solution	power	energy	time	area	A×T
3 mult., $d = 16$	203 mW	0.036 mJ	177 μs	9393 LUT	1.66
2 mult., $d = 16$	192 mW	0.039 mJ	201 μs	6711 LUT	1.35

Addition Chains (Work of Nicolas Méloni)

In scalar multiplication $[k]P$, only use point additions on the curve

- robust against SPA
- $ADD(P_1, P_2) = (P_1 + P_2, P_1)$ with P_1 and P_2 already computed
- problem find a short chain

Example: addition chains for $k = 113$

1	1	2	1	1	6	1	1	14	14	14	14	14	14	14	
1	2	3	5	6	7	13	14	15	29	43	57	71	85	99	113

1	1	1	1	4	5	5	14	14	19	47	
1	2	3	4	5	9	14	19	33	47	66	113

Collaboration with UCC code and crypto group (2006–2008)

Signed-Digit Redundant Number Systems

Avizienis 1961: radix β representation

- replace the digit set $\{0, 1, 2, \ldots, \beta - 1\}$
- by the digit set $\{-\alpha, -\alpha + 1, \ldots, 0, \ldots, \alpha - 1, \alpha\}$ with $\alpha \leq \beta - 1$

If $2\alpha + 1 > \beta$ some numbers have several possible representations

Example: radix $\beta = 10$, digits from the set $\mathcal{D} = \{\overline{9}, \ldots, \overline{1}, 0, 1, \ldots, 9\}$

$$\begin{aligned} 2010 &= (2010)_{\beta,\mathcal{D}} \\ &= (21\overline{9}0)_{\beta,\mathcal{D}} \\ &= (3\overline{99}0)_{\beta,\mathcal{D}} \\ &= (1\overline{8}010)_{\beta,\mathcal{D}} \\ &= (1\overline{8}1\overline{9}0)_{\beta,\mathcal{D}} \\ &= \ldots \end{aligned}$$

In a redundant number system there is constant-time addition algorithm (without carry propagation) where all computations are done in parallel

Recoding k

Recoding: w-NAF (*non-adjacent form*)
With

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

use k with digits in "windows" of w bits

$$|k_i| < 2^{w-1}$$

Example:

$$\begin{array}{rcccccccccl} k = 267 = & (& 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 &)_2 \\ & (& 1 & 0 & 0 & 0 & 1 & 0 & \bar{1} & 0 & \bar{1} &)_{2-NAF} \\ & (& 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 3 &)_{3-NAF} \\ & (& 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \bar{5} &)_{4-NAF} \\ & (& 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 &)_{5-NAF} \end{array}$$

Cost: $(n-1) \cdot DBL$ and $\frac{n}{w+1} \cdot ADD$

Double-Base Number Systems (DBNS) (1/3)

Redundant representation based the sum of powers of 2 AND 3:

$$x = \sum_{i=1}^{n} x_i 2^{a_i} 3^{b_i}, \text{ with } x_i \in \{-1, 1\}, \ a_i, b_i \geq 0$$

Example: $127 = 108 + 16 + 3 = 72 + 54 + 1 = \ldots$

	1	2	4	8	16
1					1
3	1				
9					
27			1		

	1	2	4	8
1	1			
3				
9				1
27		1		

Source: L. Imbert

Double-Base Number Systems (DBNS) (2/3)

Smallest $x > 0$ with n DBNS terms in its decomposition:

n	unsigned	signed
2	5	5
3	23	105
4	431	(4985)
5	18,431	?
6	3,448,733	
7	1,441,896,119	
8	?	

DBNS is a very sparse and redundant representation

Example: 127 has 783 DBNS representations among which 6 are canonic: $127 = (108 + 18 + 1) = (108 + 16 + 3) = (96 + 27 + 4) = (72 + 54 + 1) = (64 + 54 + 9) = (64 + 36 + 27)$

Double-Base Number Systems (DBNS) (3/3)

Application: ECC scalar multiplication

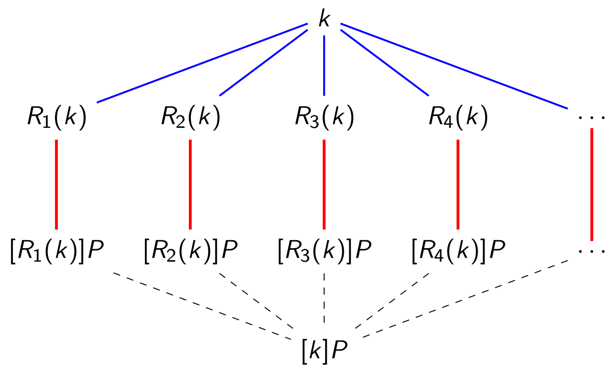
$314159 = 2^4 3^9 + 2^8 3^1 - 1$
$[314159]P = [2^4 3^9]P + [2^8 3^1]P - P$
cost: 12 DBL + 10 TPL + 2 ADD

$314159 = 2^4 3^9 - 2^0 3^6 - 3^3 - 3^2 - 3 - 1$
$[314159]P = 3(3(3(3^3([2^4 3^3]P - P) - P) - P) - P$
cost: 4 DBL + 9 TPL + 5 ADD

Protection at the Arithmetic Level

Redundant number system =

- a way to improve the performance of some operations
- a way to represent a value with different representations

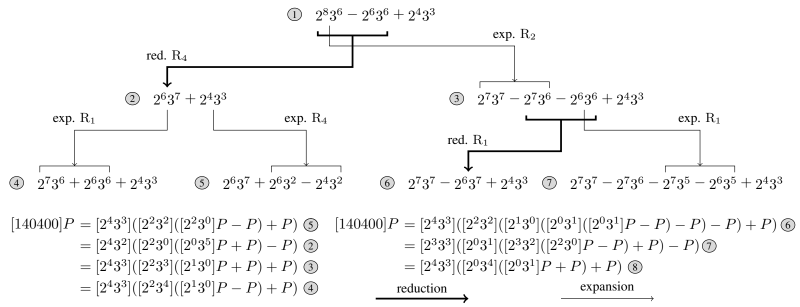


Proposed solution: use random redundant representations of k

PhD Thesis of Thomas Chabrier

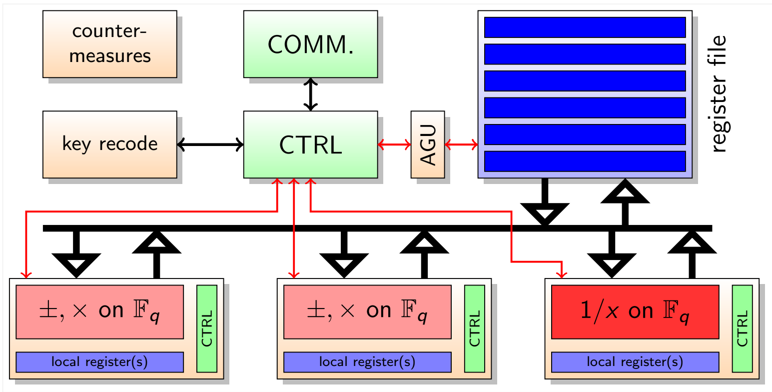
Hardware random recoding of the scalar (NAF-like, DBNS, ...)

Recoding rules: $1+2 \longleftrightarrow 3$, $1+3 \longleftrightarrow 4$, $1+8 \longleftrightarrow 9$, ...



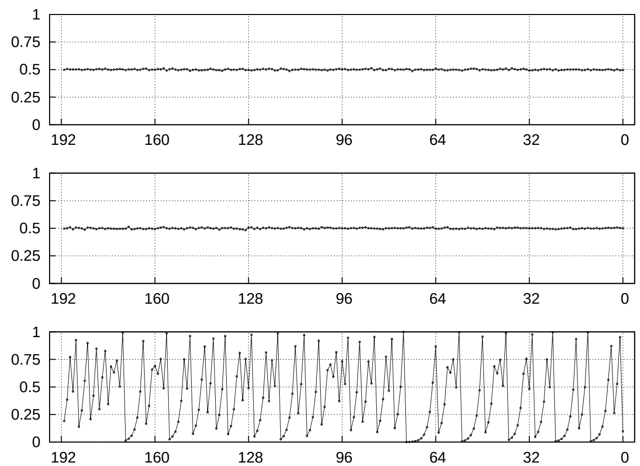
Security evaluation in progress

ECC (Co)Processor under Development



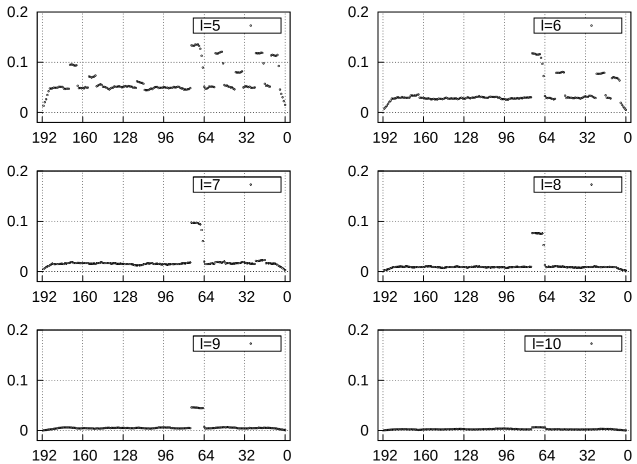
- Functional units (FU): $\pm, \times, 1/x$ for $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$, key recoding
- Memory: register file + internal registers in the FUs
- Control: operations (E and $\mathbb{F}_q$ levels) schedule, parameters management...

Activity in GF(p) Arithmetic Operators (1/2)



top: addition, middle: multiplication, bottom: addition with a constant

Activity in $\mathbb{F}_p$ Arithmetic Operators (2/2)



Other Topics

- Countermeasures against side channel attacks or fault attacks
- Parameters selection (security/performance/cost trade-off. . .)
- Specific operations (e.g. *ReADD*: addition where one of the addends has been added before)
- Unified equations (same equations for *ADD* and *DBL*)
- Montgomery point multiplication
- Multiple point multiplication ($kP + lQ$)
- Point halving
- Specific curves (Edwards, Montgomery, Huff, ...) curves
- . . .

The end, some questions ?

Contact:

- `mailto:arnaud.tisserand@irisa.fr`
- `http://www.irisa.fr/prive/Arnaud.Tisserand/`
- CAIRN Group `http://www.irisa.fr/cairn/`
- IRISA Laboratory, CNRS–INRIA–Univ. Rennes 1
  6 rue Kérampont, BP 80518, F-22305 Lannion cedex, France

Thank you

SAGE Mathematical Software System

Features and information:

- Topics: algebra, combinatorics, geometry, number theory, numerical mathematics, calculus, cryptography. . .
- URL: `http://www.sagemath.org/`
- License: GPL and GNU Free Documentation License
- Language: Python
- Platforms: Linux, OS X and Solaris (both x86 and SPARC)
- History: 0.1 in Jan. 2005, $\approx$ 1 main version/year + several releases/year
- Use: command line or notebook (through a web browser)

Integrated libraries: GMP, NTL, MPFR, MPFI, LinBox, ATLAS. . .

Interfaces to/from: GP/Pari, Gnuplot, Magma, Maple, Matlab, Maxima, Mathematica, Octave. . .

Sage Examples (1/3)

```
----------------------------------------------------------------------
| Sage Version 4.1, Release Date: 2009-07-09                         |
| Type notebook() for the GUI, and license() for information.        |
----------------------------------------------------------------------
sage: 1+1
2

sage: (factor(29),factor(30))
(29, 2 * 3 * 5)

sage: x, b, c = var('x b c')
sage: solve([x^2 + b*x + c == 0],x)
[x == -1/2*b - 1/2*sqrt(b^2 - 4*c), x == -1/2*b + 1/2*sqrt(b^2 - 4*c)]

sage: ER1=EllipticCurve([0,0,0,-1,0])
sage: ER1
Elliptic Curve defined by y^2 = x^3 - x over Rational Field
sage: show(plot(ER1),aspect_ratio=1,xmin=-1,xmax=2,ymin=-2,ymax=2)
```

Remark: the prompts `sage:` or >>> are ignored during cut/paste

Sage Examples (2/3)

```
sage: EF1=EllipticCurve(GF(101),[0,0,0,-100,0])
sage: EF1
Elliptic Curve defined by y^2 = x^3 + x over Finite Field of size 101
sage: show(plot(EF1),aspect_ratio=1)

sage: EEF29=EllipticCurve(GF(29),[0,0,0,4,20])
sage: EEF29
Elliptic Curve defined by y^2 = x^3+4*x+20 over Finite Field of size 29
sage: show(plot(EEF29),aspect_ratio=1)
sage: P=EEF29.random_point()
sage: Q=EEF29.random_point()
sage: P, Q
((3 : 1 : 1), (24 : 7 : 1))
sage: P+Q
(8 : 10 : 1)
sage: 2*P
(24 : 7 : 1)
sage: 2*Q
(5 : 7 : 1)
```


Sage Examples (3/3)

```
sage: F29=GF(29)
sage: F29((22-10)/(24-8))
8
sage: F29(8^2-8-24)
3
sage: F29(8*(8-3)-10)
1
sage:
sage: F29((3*8^2+4)/(2*10))
4
sage: F29(4^2-8-8)
0
sage: F29(4*(8-0)-10)
22

sage: exit
Exiting SAGE (CPU time 0m4.10s, Wall time 18m4.22s).
Exiting spawned Maxima process.
```