Introduction to Elliptic Curve Cryptography: Implementation Aspects

Francisco Rodríguez-Henríquez

Cinvestav-IPN. Computer Science Department.
francisco@cs.cinvestav.mx

Material adapted from joint-works with Luis J. Domínguez Pérez, Armando Faz-hernández, Laura Fuentes-Castañeda and Ana Helena Sánchez Ramírez

October 28^{rd} 2012 .

Cinvestav

Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional

Outline

1 Context

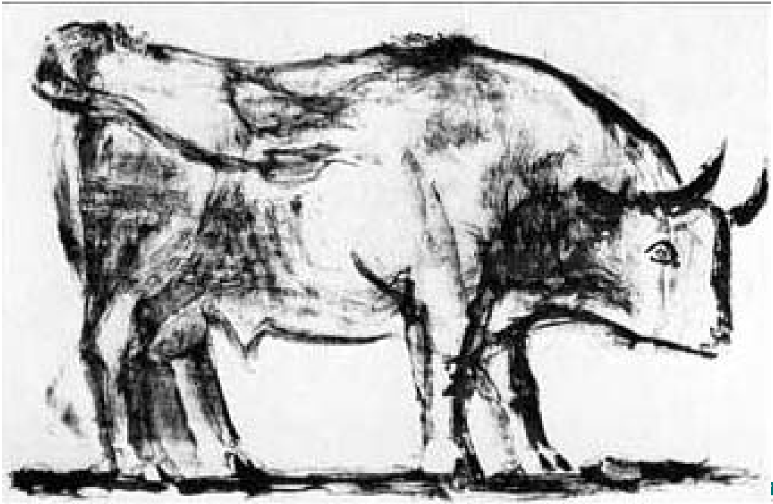
2 Field Arithmetic

3 Elliptic curve Arithmetic

4 Elliptic curve defined over prime fields

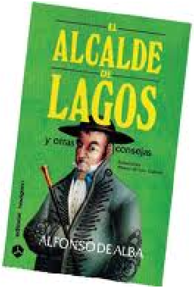
5 Other tricks

Pablo Picasso: The bull challenge (1/11)



Importancia del contexto

Orden del señor Alcalde:
"Desde hoy, el que tenga puercos que los amarre y el que no que no"



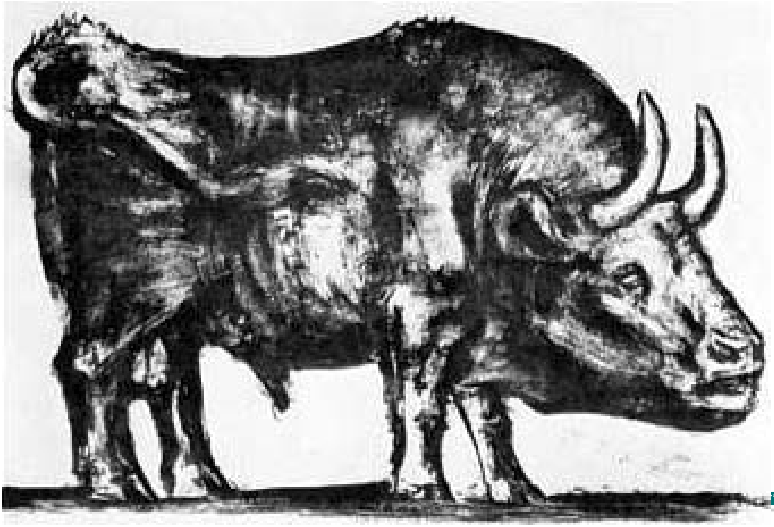
Importancia del contexto

Orden del señor Alcalde:

"Desde hoy, el que tenga puercos que los amarre y el que no que no"

Cinvestav
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional
50 AÑOS

Pablo Picasso: The bull challenge (2/11)



Flynn's Taxonomy [IEEE TC'72]

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

Flynn's Taxonomy [IEEE TC'72]

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

SISD

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

- 1 instruction at a time on 1 item of data at a time
- M. Flynn included pipelined architectures in this category
- Processors Intel $<$ 1996 and AMD $<$ 1998

SISD

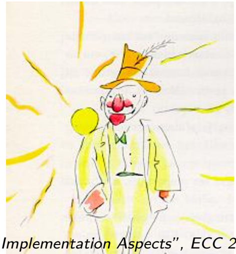
	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

- 1 instruction at a time on 1 item of data at a time
- M. Flynn included pipelined architectures in this category
- Processors Intel $<$ 1996 and AMD $<$ 1998

MISD

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

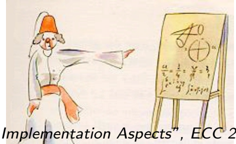
- Executing different instructions on the same data set
- Not common! To detect and mask errors...



SIMD

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

- Parallelism : execute the same operation on different data
- First Pentium : Intel Pentium MMX (1996), MMX Instructions
- First AMD : AMD K6-2 (1998), 3DNow! Instructions

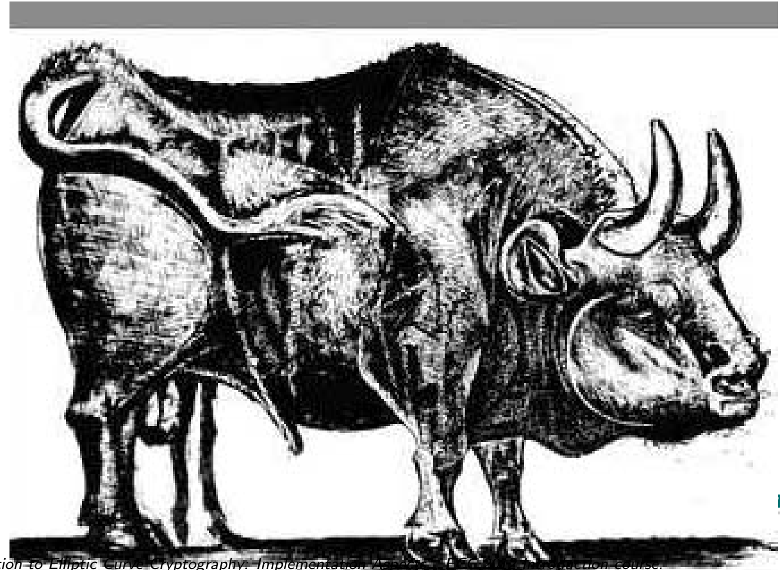


MIMD

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

- Parallelism : execute asynchronously different set of instructions independently on different set of data elements
- Multiprocessor architectures, clusters, etc.

Pablo Picasso: The bull challenge (3/11)



SSE

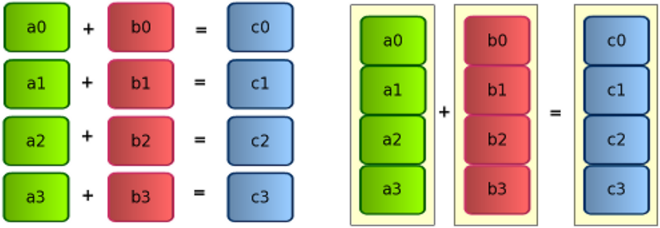
- SSE : Streaming SIMD Extensions
- The initial targets were multimedia applications as image processing, audio/video encoding or decoding, HD
- The extended targets : scientific purposes, cryptography, ...

SIMD Instructions

SIMD instructions, (*single instruction, multiple data*), perform one logic/arithmetic operation over multiple data

SIMD Instructions

SIMD instructions, (*single instruction, multiple data*), perform one logic/arithmetic operation over multiple data



Vector instructions

- Nowadays vector instructions are present in contemporary desktop processors.

Vector instructions

- Nowadays vector instructions are present in contemporary desktop processors.
- Latest architectures have special register and instruction sets that are able to perform one single operation over a set of data. Resulting in a vector-wise processing.

Vector instructions

- Nowadays vector instructions are present in contemporary desktop processors.
- Latest architectures have special register and instruction sets that are able to perform one single operation over a set of data. Resulting in a vector-wise processing.
- Intel's Sandy Bridge architecture provides sixteen 256-bit registers, which can store up to four 64-bit integers.

Vector instructions

Some relevant vector instructions:

- **Bit-wise XOR, AND, OR.** These instructions operate with 256-bit registers performing bit-wise operations.

Vector instructions

Some relevant vector instructions:

- **Bit-wise XOR, AND, OR.** These instructions operate with 256-bit registers performing bit-wise operations.
- **64-bit shifts.** It processes four parallel shifts on each 64-bit integer allocated in the register.

Vector instructions

Some relevant vector instructions:

- **Bit-wise XOR, AND, OR.** These instructions operate with 256-bit registers performing bit-wise operations.
- **64-bit shifts.** It processes four parallel shifts on each 64-bit integer allocated in the register.
- **128-bit shifts.** Processes two parallel shifts on each 128-bit data in the register, under the restriction that the shifts can only be 8-bit multiples.

Vector instructions

Some relevant vector instructions:

- **Bit-wise XOR, AND, OR.** These instructions operate with 256-bit registers performing bit-wise operations.
- **64-bit shifts.** It processes four parallel shifts on each 64-bit integer allocated in the register.
- **128-bit shifts.** Processes two parallel shifts on each 128-bit data in the register, under the restriction that the shifts can only be 8-bit multiples.
- **Memory alignment.** This instruction concatenates two vector registers and shift them by an 8-bit multiple.

Vector instructions

Some relevant vector instructions:

- **Bit-wise XOR, AND, OR.** These instructions operate with 256-bit registers performing bit-wise operations.
- **64-bit shifts.** It processes four parallel shifts on each 64-bit integer allocated in the register.
- **128-bit shifts.** Processes two parallel shifts on each 128-bit data in the register, under the restriction that the shifts can only be 8-bit multiples.
- **Memory alignment.** This instruction concatenates two vector registers and shift them by an 8-bit multiple.
- Carry-less multiplier.

Carry less multiplication

As illustrated in the following example, this operation acts without generating carries, hence its name.

$$\begin{array}{cccccccc}
 & & & 1 & 1 & 0 & (6) \\
 & \times & & 1 & 1 & 1 & (7) \\
\hline
 & & & 1 & 1 & 0 & \\
 & & 1 & 1 & 0 & 0 & \\
 & 1 & 1 & 0 & 0 & 0 & \\
\hline
1 & 0 & 1 & 0 & 1 & 0 & (42)
\end{array}$$

(a) with carry

$$\begin{array}{cccccc}
 & & 1 & 1 & 0 & (6) \\
\times & & 1 & 1 & 1 & (7) \\
\hline
 & & 1 & 1 & 0 & \\
 & 1 & 1 & 0 & 0 & \\
1 & 1 & 0 & 0 & 0 & \\
\hline
1 & 0 & 0 & 1 & 0 & (18)
\end{array}$$

(b) without carry

Cinvestav
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional
50 AÑOS

Carry-less multiplier

- The instruction `PCLMULQDQ`, included in the AES-NI instruction set, performs a polynomial multiplication over $\mathbb{F}_2[x]$.

Carry-less multiplier

- The instruction PCLMULQDQ, included in the AES-NI instruction set, performs a polynomial multiplication over $\mathbb{F}_2[x]$.
- Unlike integer multiplication, this instruction performs intermediate additions regardless carry bits, hence its name.

Carry-less multiplier

- The instruction PCLMULQDQ, included in the AES-NI instruction set, performs a polynomial multiplication over $\mathbb{F}_2[x]$.
- Unlike integer multiplication, this instruction performs intermediate additions regardless carry bits, hence its name.
- Recent applications of this instruction on binary field arithmetic have shown dramatic throughput speedups, getting cutting-edge high speed implementations. For example, in the computation of the scalar multiplication operation over binary elliptic curves.

Fields

A field $\mathbb{F}$ is a set of elements equipped with two binary operations $(\star)$ y $(\bullet)$, such that,

$$\langle \mathbb{F}, \star, 0 \rangle \;\text{ and }\; \langle \mathbb{F} \setminus \{0\}, \bullet, 1 \rangle$$

are abelian groups.
The binary operation $\bullet$ can be distributed over $\star$, i. e., for all $a, b, c \in \mathbb{F}$, the following identity holds,

$$a \bullet (b \star c) = (a \bullet b) \star (a \bullet c)$$

Finite fields

Every prime number p defines a finite field of order p, denoted as, $\mathbb{F}_p$.

The smallest finite field is $\langle \mathbb{F}_2, \oplus, \odot \rangle$, that contains only two elements $\{0, 1\}$ and its binary operations act as the Boolean operators `XOR` and `AND`, respectively.

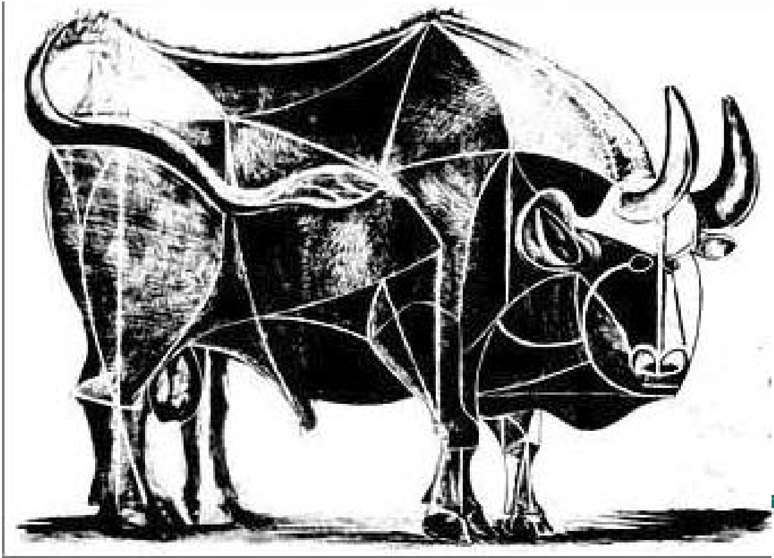
Field Extensions

Given a positive integer $m > 1$,the field $\mathbb{F}_{p^m}$ is a field extension of $\mathbb{F}_p$.

It can be shown that $\mathbb{F}_{p^m}$ is isomorphic to $\mathbb{F}_p[x]/(f(x))$, where $f(x)$ is a monic polynomial of degree $m > 1$, irreducible over $\mathbb{F}_p$.

We denote by $\mathbb{F}_p[x]/(f(x))$ the set of equivalence classes of the polynomials $\mathbb{F}_p[x] \pmod{f(x)}$.

Pablo Picasso: The bull challenge (4/11)



Addition

A field element $\mathbb{F}_{2^m}$ can be represented as a vector of m bits.

The addition of two field elements, $a(x), b(x) \in \mathbb{F}_{2^m}$ can be performed just with bit-wise XOR [no carries are generated],

$$c(x) = a(x) + b(x) = \sum_{i=0}^{m-1} (a_i \oplus b_i) x^i$$

This operation directly benefits from the parallel processing of the `XOR` operation over a vector of data

Multiplication

Field multiplication is usually performed in two steps: polynomial multiplication followed by polynomial reduction

The first phase consists on multiplying two polynomials of degree $m-1$ to obtain a polynomial of degree $2m-2$, where the arithmetic operations are performed over $\mathbb{F}_2$.

The second phase performs modular reduction using $f(x)$, the irreducible polynomial that generated the field.

Polynomial multiplication

We use the Karatsuba multiplier at a computational cost of just $O(m^{\log_2 3})$.

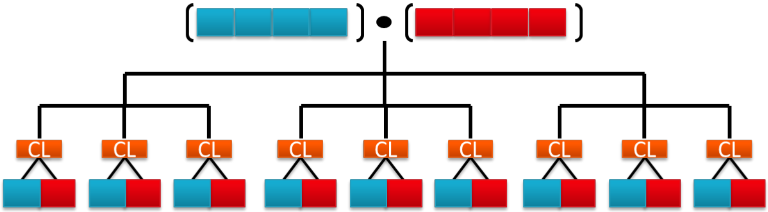
Given the polynomials A y B of degree $m-1$, the product $C = A \cdot B$ of degree $2m-2$ can be computed as,

$$\begin{aligned} C &= A \cdot B \\ &= (a_0 + a_1 x^{\frac{m-1}{2}})(b_0 + b_1 x^{\frac{m-1}{2}}) \\ &= a_0 b_0 + [(a_0 + a_1)(b_0 + b_1) + a_0 b_0 + a_1 b_1]\, x^{\frac{m-1}{2}} \\ &\quad + a_1 b_1 x^{m-1} \end{aligned}$$

This operation can be recursively repeated until the bit level

Polynomial multiplication

Using the new carry-less multiplication instruction PCLMULQDQ one can multiply 64-bit binary polynomials and stop at that level the recursion



Field Squaring

Due to the action of the Frobenius map, polynomial squaring of an element $a \in \mathbb{F}_{2^m}$ is a linear operation over binary fields,

$$\begin{aligned} a(x)^2 &= \left[\sum_{i=0}^{m-1} a_i x^i\right]^2 \\ &= \sum_{i=0}^{m-1} a_i x^{2i} \end{aligned}$$

This can be implemented by interleaving zeroes among the polynomial coefficients,

$$\begin{aligned} \vec{a} &\rightarrow (\vec{a})^2 \\ (a_{m-1}, a_{m-2}, \ldots, a_1, a_0) &\rightarrow (a_{m-1}, 0, \ldots, a_2, 0, a_1, 0, a_0) \end{aligned}$$

Multi-squaring

Computing a^{2^k}, with k a constant, one can pre-compute a table T that contains $16\lceil \frac{m}{4} \rceil$ field elements, computed as,

$$T[j, i_0 + 2i_1 + 4i_2 + 8i_3] = (i_0 x^{4j} + i_1 x^{4j+1} + i_2 x^{4j+2} + i_3 x^{4j+3})^{2^k} \quad (1)$$

where $i_0, i_1, i_2, i_3 \in \{0, 1\}$ and $0 \leq j < \lceil \frac{m}{4} \rceil$.
Finally, the multi-squaring computation can be performed using,

$$a(x)^{2^k} = \sum_{j=0}^{\lceil \frac{m}{4} \rceil - 1} T[j, \lfloor a/2^{4j} \rfloor \bmod 2^4] \quad (2)$$

Field Inversion

- **Inversion.** The friendliest approach to compute the most costly binary field operation is using Itoh-Tsujii algorithm. Given a field element a, we use the following identity to compute its inverse:

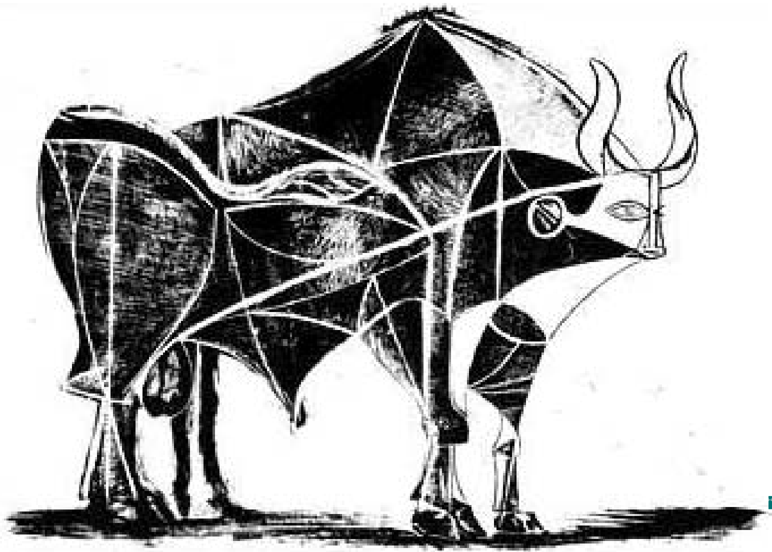
$$a^{-1} = \left(a^{2^{m-1}-1}\right)^2$$

The term $a^{2^{m-1}-1}$ is obtained by sequentially computing intermediate terms of the form:

$$\left(a^{2^i-1}\right)^{2^j} \cdot \left(a^{2^j-1}\right) \qquad i, j \in [0, \lambda]$$

where i, j are elements of an addition chain of λ length. This sequence of powers is done by using multi-squaring operations.

Pablo Picasso: The bull challenge (5/11)



Field towering

$$\mathbb{F}_{p^{12}} \qquad \mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[w]/(w^2 - \gamma)$$

$$\simeq \qquad \mathbb{F}_{p^6} = \mathbb{F}_{p^2}[v]/(v^3 - \xi)$$

$$\mathbb{F}_{p^2} = \mathbb{F}_p[u]/(u^2 - \beta)$$

$$\mathbb{F}_p \qquad \mathbb{F}_p$$

$\mathbb{F}_p$ Arithmetic

$\mathbb{F}_p$ field arithmetic has crucial importance for the performance of any cryptosystem. The field elements $a, b \in \mathbb{F}_p$ are integers in the interval $[0, p-1]$

Addition	$a + b \bmod p$
Multiplication	$a \cdot b \bmod p$
Multiplicative inversion	$a^{-1} \bmod p$

Montgomery Multiplier

The problem of performing a division by p is traded with divisions by r, where $r = 2^k$ with $k - 1 < |p| < k$.

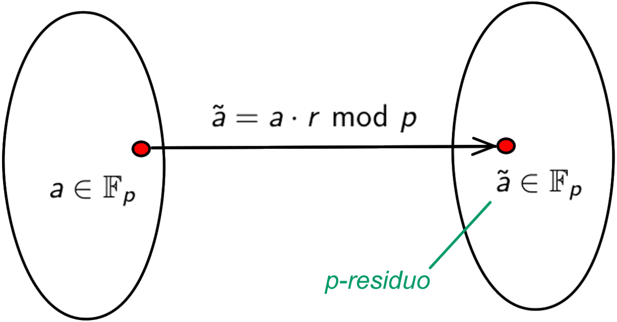


Figure: Montgomery p-Residues

Montgomery Multiplier

The montgomery product is defined as,

$$\mathrm{MontPr}(\tilde{a}, \tilde{b}) = \tilde{a} \cdot \tilde{b} \cdot r^{-1} \bmod p$$

Given its p-residue $\tilde{a}$, one can compute a by performing,

$$\mathrm{MontPr}(\tilde{a}, 1) = \tilde{a} \cdot 1 \cdot r^{-1} \bmod p = a \bmod p$$

Where p' can be obtained from Bezout's identity as,

$$r \cdot r^{-1} - p \cdot p' = 1,$$

provided that $\gcd(r, p) = 1$.

Montgomery Multiplier

Input: Prime p, p', $r = 2^k$ y $\tilde{a}$, $\tilde{b} \in \mathbb{F}_p$
Ouput: $\text{MontPr}(\tilde{a}, \tilde{b})$

1: $t \leftarrow \tilde{a} \cdot \tilde{b}$
2: $m \leftarrow t \cdot p' \bmod r$
3: $u \leftarrow (t + m \cdot p)/r$
4: **if** $u > p$ **then**
5:     **return** $u - p$
6: **else**
7:     **return** u
8: **end if**
9: **return** u

Montgomery Multiplier

Input: Prime p, p', $r = 2^k$ y $\tilde{a}$, $\tilde{b} \in \mathbb{F}_p$
Ouput: $\mathrm{MontPr}(\tilde{a}, \tilde{b})$

1: $t \leftarrow \tilde{a} \cdot \tilde{b}$
2: $m \leftarrow t \cdot p' \bmod r$ $\quad m \equiv -t \cdot p^{-1} \bmod r$
3: $u \leftarrow (t + m \cdot p)/r$ $\quad (t + m \cdot p) \equiv 0 \bmod r$
4: **if** $u > p$ **then**
5: **return** $u - p$
6: **else**
7: **return** u
8: **end if**
9: **return** u

Montgomery multiplier

Input: Prime p, p', $r = 2^k$ y $\tilde{a}$, $\tilde{b} \in \mathbb{F}_p$
Ouput: $\mathrm{MontPr}(\tilde{a}, \tilde{b})$

```
t ← ã · b̃
m ← t · p' mod r
u ← (t + m · p)/r
if u > p then
    return u − p
else
    return u
end if
return u
```

$t + m \cdot p \equiv \tilde{a} \cdot \tilde{b} \bmod p$

Montgomery multiplier variants: the SOS Separated Operand Scanning method

Computes first the product $t = a \cdot b$ and then u.

Input: $a = (a_0, a_1, \ldots, a_{n-1})$ and
$b = (b_0, b_1, \ldots, b_{n-1})$
Ouput: $t = a \cdot b$ with $t = (t_0, t_1, \ldots, t_{2n-1})$

$t \leftarrow 0$
for $i = 0 \rightarrow n-1$ **do**
    $C \leftarrow 0$
    for $j = 0 \rightarrow n-1$ **do**
        $(C, S) \leftarrow t_{i+j} + a_j \cdot b_i + C$
        $t_{i+j} = S$
    end for
    $t_{i+n} = C$
    return t
end for

				a_3	a_2	a_1	a_0
				b_3	b_2	b_1	b_0
				t_{03}	t_{02}	t_{01}	t_{00}
			t_{13}	t_{12}	t_{11}	t_{10}	
		t_{23}	t_{22}	t_{21}	t_{20}		
	t_{33}	t_{32}	t_{31}	t_{30}			
t_7	t_6	t_5	t_4	t_3	t_2	t_1	t_0

The complexity of this algorithm is $\mathcal{O}(n^2)$

Montgomery multiplier variants: the SOS Separated Operand Scanning method

Input: $t = (t_0, t_1, ..., t_{2n-1})$, $p = (p_0, p_1, ..., p_n)$ and p'_0, where $|p'_0| = \omega$
Ouput: $u \leftarrow (t + (t \cdot p' \bmod r) \cdot p)/r$

```
for i = 0 → n − 1 do
    C ← 0
    m ← t_i · p'_0 mod 2^ω
    for j = 0 → n − 1 do
        (C, S) ← t_{i+j} + m · p_j + C
        t_{i+j} = S
    end for
    ADD(t_{i+n}, C)
end for
for i = 0 → n − 1 do
    u_i = t_{i+n}
end for
return u
```

The number of products of this method is $2n^2 + n$.

Montgomery multiplier variants: the SOS Separated Operand Scanning method

Input: $t = (t_0, t_1, \ldots, t_{2n-1})$, $p = (p_0, p_1, \ldots, p_n)$ and p_0', where $|p_0'| = \omega$
Ouput: $u \leftarrow (t + (t \cdot p' \bmod r) \cdot p)/r$

```
for i = 0 → n − 1 do
    C ← 0
    m ← t_i · p'_0 mod 2^ω
    for j = 0 → n − 1 do
        (C, S) ← t_{i+j} + m · p_j + C
        t_{i+j} = S
    end for
    ADD(t_{i+n}, C)
end for
for i = 0 → n − 1 do
    u_i = t_{i+n}
end for
return u
```

The number of products of this method is $2n^2 + n$.

$\mathbb{F}_{p^2}$ Arithmetic

$$\mathbb{F}_{p^2} \cong \mathbb{F}_p[u]/(u^2 - \beta), \quad \beta \in \mathbb{F}_p$$

where β is not a square over $\mathbb{F}_p$. Hence, a field element $A \in \mathbb{F}_{p^2}$ can be seen as, $A = a_0 + a_1 u$, where $a_0, a_1 \in \mathbb{F}_p$.

Adding two elements $A, B \in \mathbb{F}_{p^2}$ is given as,

$$(a_0 + a_1 u) + (b_0 + b_1 u) = (a_0 + b_0) + (a_1 + b_1)u,$$

$\mathbb{F}_{p^2}$ Arithmetic

The multiplication of two elements $A, B \in \mathbb{F}_{p^2}$ is,

$$(a_0 + a_1 u) \cdot (b_0 + b_1 u) = (a_0 b_0 + a_1 b_1 \beta) + (a_0 b_1 + a_1 b_0) u,$$

Using karatsuba method one has,

$$(a_0 b_1 + a_1 b_0) = (a_0 + a_1) \cdot (b_0 + b_1) - a_0 b_0 - a_1 b_1.$$

Field squaring A^2 where $A \in \mathbb{F}_{p^2}$, can be done using an identity borrowed from complex theory,

$$(a_0 + a_1 u)^2 = (a_0 - \beta a_1) \cdot (a_0 - a_1) + (\beta + 1) a_0 a_1 + 2 a_0 a_1 u.$$

$\mathbb{F}_{p^6}$ Arithmetic

Arithmetic in the sextic extension corresponds to the third layer of the field towering and can be built as the cubic extension of the quadratic one as,

$$\mathbb{F}_{p^6} \cong \mathbb{F}_{p^2}[V]/(V^3 - \xi), \quad \xi \in \mathbb{F}_{p^2}$$

where $\xi = u + 1$.
An element $A \in \mathbb{F}_{p^6}$ can thus be seen as, $A = a_0 + a_1 V + a_2 V^2$ where $a_0, a_1, a_2 \in \mathbb{F}_{p^2}$.

$\mathbb{F}_{p^{12}}$ Arithmetic

$\mathbb{F}_{p^{12}}$ arithmetic corresponds to the top layer in the field towering analyzed here. It can be defined as the quadratic extension of the sextic one as,

$$\mathbb{F}_{p^{12}} \cong \mathbb{F}_{p^6}[W]/(W^2 - \gamma), \quad \gamma \in \mathbb{F}_{p^6}$$

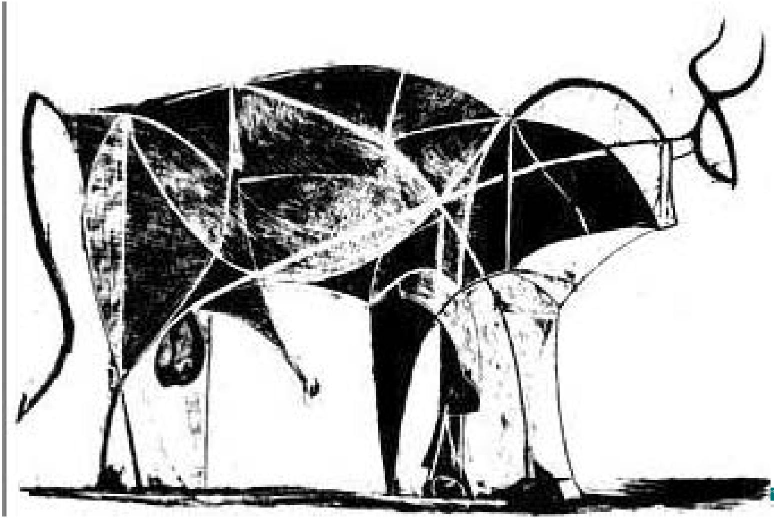
where $\gamma = V$.
Hence, an element $a \in \mathbb{F}_{p^{12}}$ can be seen as $a = a_0 + a_1 W$ where $a_0, a_1 \in \mathbb{F}_{p^6}$.

Summary of arithmetic costs

Field	Add	Mult	Square	Inverse
$\mathbb{F}_{p^2}$	$\tilde{a} = 2a$	$\tilde{m} = 3m + 5a + m_\beta$	$\tilde{s} = 2m + 3a + m_\beta$	$\tilde{i} = 4m + m_\beta + 2a + i$
$\mathbb{F}_{p^6}$	$3\tilde{a}$	$6\tilde{m} + 2m_\xi + 15\tilde{a}$	$2\tilde{m} + 3\tilde{s} + 2m_\xi + 9\tilde{a}$	$9\tilde{m} + 3\tilde{s} + 4m_\xi + 5\tilde{a} + \tilde{i}$
$\mathbb{F}_{p^{12}}$	$6\tilde{a}$	$18\tilde{m} + 6m_\xi + 60\tilde{a} + m_\gamma$	$12\tilde{m} + 4m_\xi + 45\tilde{a} + 2m_\gamma$	$25\tilde{m} + 9\tilde{s} + 12m_\xi + 61\tilde{a} + \tilde{i} + m_\gamma$

a Addition/subtraction over $\mathbb{F}_p$ | m Multiplication over $\mathbb{F}_p$
$\tilde{a}$ Addition/subtraction over $\mathbb{F}_{p^2}$ | $\tilde{m}$ Multiplication over $\mathbb{F}_{p^2}$ | $\tilde{s}$ field squaring $\mathbb{F}_{p^2}$
m_β Multiplication by β | m_ξ Multiplication by ξ | m_γ Multiplication by γ

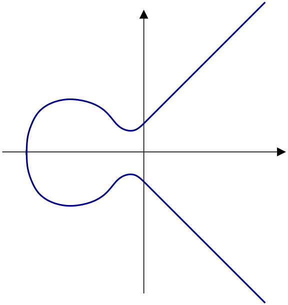
Pablo Picasso: The bull challenge (6/11)



Elliptic curves: basic definitions

- An **elliptic curve** E **over a field** $\mathbb{F}$ with field characteristic different than 2 and 3, denoted as $E/\mathbb{F}$, can be defined by the equation,

$$y^2 = x^3 + ax + b, \quad \text{where } a, b \in \mathbb{F}.$$



- $\mathcal{O}$ is the **the point at infinity**

Elliptic curves: basic definitions

- Given an elliptic curve $E/\mathbb{F}$ and a finite field $\mathbb{F}'$ such that $\mathbb{F} \subseteq \mathbb{F}'$, the set of the elliptic curve rational points $\mathbb{F}'$-**rational points** are defined as,

$$E(\mathbb{F}') = \{(x, y) \mid x, y \in \mathbb{F}',\ y^2 - x^3 - ax - b = 0\} \cup \{\mathcal{O}\}$$

- $E(\mathbb{F}')$ **is an Abelian group usually written in additive notation**, where $\mathcal{O}$ acts as the identity element.

Group law

An elliptic curve point if represented with two coordinates (x, y) is said to be in Affine coordinates. The group law of a point in such representation requires the use of inversion of elements in a finite field, which tends to be expensive.

Let $P_1 = (x_1, y_1)$, and $P_2 = (x_2, y_2)$, with $P_1, P_2 \neq \infty$. We define $P_1 + P_2 = P_3$ as follows:

Point addition

$$m = \frac{y_2 - y_1}{x_2 - x_1} \quad x_3 = m^2 - x_1 - x_2 \quad y_3 = m(x_1 - x_3) - y_1$$

Point doubling $(P_1 = P_2)$

$$m = \frac{3x_1^2 + a}{2y_1} \quad x_3 = m^2 - 2x_1 \quad y_3 = m(x_1 - x_3) - y_1$$

Elliptic curve scalar multiplication

This operation finds the k-th scalar multiple of a point $P \in E$, denoted by kP. It consists in adding k times P with itself, i.e.,

$$kP = \underbrace{P + P + \ldots + P}_{k \ times}$$

Fact: This operation can be easily computed using the binary method at a cost of $mD + \frac{m}{2}A$, where $|k| = m$.

Elliptic curves: basic definitions

- Let $\#E(\mathbb{F}_q)$ be the order of $E(\mathbb{F}_q)$, i.e. , the cardinality of the $\mathbb{F}_q$-rational points in the elliptic curve $E/\mathbb{F}_q$.

  $\#E(\mathbb{F}_p) = q + 1 - t$, where t is the Frobenius trace of E over $\mathbb{F}_q$

- Let P be a point in $E(\mathbb{F}_q)$, the order of P is defined as the smallest positive integer r, such that,

$$P + P + \ldots.. + P = rP = \mathcal{O}$$

Elliptic curves: basic definitions

- Let $\#E(\mathbb{F}_q)$ be the order of $E(\mathbb{F}_q)$, i.e. , the cardinality of the $\mathbb{F}_q$-rational points in the elliptic curve $E/\mathbb{F}_q$.

  $\#E(\mathbb{F}_p) = q + 1 - t$, where t is the Frobenius trace of E over $\mathbb{F}_q$

- Let P be a point in $E(\mathbb{F}_q)$, the order of P is defined as the smallest positive integer r, such that,

$$P + P + \ldots.. + P = rP = \mathcal{O}$$

- Fact: the order r of a point $P \in E(\mathbb{F}_q)$ always divides $\#E(\mathbb{F}_q)$.

Cinvestav
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional
50 AÑOS

Elliptic curves: basic definitions

- Let $\#E(\mathbb{F}_q)$ be the order of $E(\mathbb{F}_q)$, i.e. , the cardinality of the $\mathbb{F}_q$-rational points in the elliptic curve $E/\mathbb{F}_q$.

  $\#E(\mathbb{F}_p) = q + 1 - t$, where t is the Frobenius trace of E over $\mathbb{F}_q$

- Let P be a point in $E(\mathbb{F}_q)$, the order of P is defined as the smallest positive integer r, such that,

$$P + P + ... + P = rP = \mathcal{O}$$

- Fact: the order r of a point $P \in E(\mathbb{F}_q)$ always divides $\#E(\mathbb{F}_q)$.

Elliptic curves: basic definitions

- Given an elliptic curve $E/\mathbb{F}_p$, the set of F_q-rational points of torsion r, denoted as $E(\mathbb{F}_{p^n})[r]$, is defined as,

$$E(\mathbb{F}_{p^n})[r] = \{P \in E(\mathbb{F}_{p^n}) | rP = \mathcal{O}\}.$$

Elliptic curves: basic definitions

Embedding degree

Let $E/\mathbb{F}_p$ be an elliptic curve such that $\#E(\mathbb{F}_p) = h \cdot r$, where $h \in \mathbb{Z}^+$. and let k be a positive integer, we say that k is the embedding degree of $E/\mathbb{F}_p$ with respect to p and r, if k is the smallest positive integer such that,

$$r | p^k - 1.$$

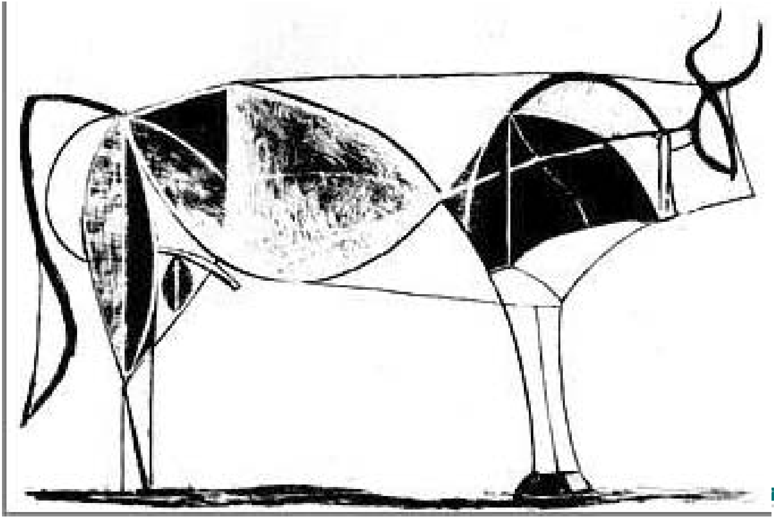
Elliptic curves: basic definitions

Twist of an elliptic curve

Given an elliptic curve $E/\mathbb{F}_p$ with embedding degree k. If the group $E(\mathbb{F}_p)$ contains a subgroup of prime order r, there exists a twist curve E' of E, defined over the field $\mathbb{F}_q$, with $q = p^{k/d}$ and $d \in \mathbb{Z}$, such that E y E' are isomorphic over $\mathbb{F}_{p^k}$, i.e,

$$\phi : E'(\mathbb{F}_{p^{k/d}}) \to E(\mathbb{F}_{p^k}),$$

Pablo Picasso: The bull challenge (7/11)



Elliptic curve families: The Barreto-Naehrig curves

- The embedding degree of a BN curve is $k = 12$, always with a prime order r, i.e., $\#E(\mathbb{F}_p) = r$.

Elliptic curve families: The Barreto-Naehrig curves

- The embedding degree of a BN curve is $k = 12$, always with a prime order r, i.e., $\#E(\mathbb{F}_p) = r$.
- Moreover, the field characteristic, group order and Frobenius trace are parameterized as,

Elliptic curve families: The Barreto-Naehrig curves

- The embedding degree of a BN curve is $k = 12$, always with a prime order r, i.e., $\#E(\mathbb{F}_p) = r$.
- Moreover, the field characteristic, group order and Frobenius trace are parameterized as,

$$
\begin{aligned}
p(z) &= 36z^4 + 36z^3 + 24z^2 + 6z + 1 \\
r(z) &= 36z^4 + 36z^3 + 18z^2 + 6z + 1 \\
t(z) &= 6z^2 + 1
\end{aligned}
$$

Elliptic curve families: The Barreto-Naehrig curves

- The embedding degree of a BN curve is $k = 12$, always with a prime order r, i.e., $\#E(\mathbb{F}_p) = r$.
- Moreover, the field characteristic, group order and Frobenius trace are parameterized as,

$$
\begin{aligned}
p(z) &= 36z^4 + 36z^3 + 24z^2 + 6z + 1 \\
r(z) &= 36z^4 + 36z^3 + 18z^2 + 6z + 1 \\
t(z) &= 6z^2 + 1
\end{aligned}
$$

- If for a given $z \in \mathbb{Z}$ $p = p(z)$ y $r = r(z)$ are prime numbers, then the BN equation is defined as, $E/\mathbb{F}_p : y^2 = x^3 + b$.

Elliptic curve families: The Barreto-Naehrig curves

- The embedding degree of a BN curve is $k = 12$, always with a prime order r, i.e., $\#E(\mathbb{F}_p) = r$.
- Moreover, the field characteristic, group order and Frobenius trace are parameterized as,

$$p(z) = 36z^4 + 36z^3 + 24z^2 + 6z + 1$$
$$r(z) = 36z^4 + 36z^3 + 18z^2 + 6z + 1$$
$$t(z) = 6z^2 + 1$$

- If for a given $z \in \mathbb{Z}$ $p = p(z)$ y $r = r(z)$ are prime numbers, then the BN equation is defined as, $E/\mathbb{F}_p : y^2 = x^3 + b$.
- $E/\mathbb{F}_p$ is isomorphic to the sextic degree twist curve denoted as $E'/\mathbb{F}_{p^2}$.

Discrete logarithm problem

Let $P = (x, y)$ be a point in $E(\mathbb{F}_p)$ of order r.

Then denote by $< P >$ the group generated by P. In other words,

$$< P >= \{\mathcal{O}, P, P + P, P + P + P, \ldots\}$$

Let $Q \in < P >$. Given Q, find n such that $Q = [n]P$. This is known as the **Elliptic Curve Discrete Logarithm Problem (ECDLP)**.

Known attacks affect some anomalous curves, P with a small prime order and some weak combinations of parameters.

Discrete logarithm problem II

Similarly, let $\alpha \in \mathbb{F}^*_{p^k}$ and $k \in \mathbb{Z}$, $k > 0$. Define $\alpha^e = \alpha \cdot \alpha \ldots \alpha$, e times. Then, the **order of the element** α is the smallest n such that $\alpha^n = 1$.

Denote by $< \alpha >$ the group generated by α. In other words,

$$< \alpha >= \{1, \alpha, \alpha \cdot \alpha, \alpha \cdot \alpha \cdot \alpha, \ldots\}$$

Let $\beta \in < \alpha >$. Given β, the problem of finding s modulo $|\alpha|$ such that $\beta = \alpha^s$. is known as the **The Finite Field Discrete Logarithm Problem (DLP)**.

The most efficient methods for the finite field case are based on Index Calculus. The most efficient methods in elliptic curves are based on the Pollard's Rho attack.

Point multiplication on EC w/ efficient endomorphisms

Paper:*Faster Point Multiplication on Elliptic Curves* by Gallant, Lambert and Vanstone.

The scalar-point multiplication is the additive analogue of the exponentiation operation α^k in a general (multiplicatively-written) finite group.

In other words, we can apply the same concepts in groups defined with different operations, and referring the operation simply as exponentiation in a group.

Speeding up

Generic methods to speed up the exponentiation in any finite Abelian group includes,

- Precomputation
- Addition chains whenever the scalar is known
- Windowing techniques
- Simultaneous multiple exponentiation techniques.

Replacing the binary representation of the scalar into one with fewer non-zero terms.

Speeding up II

Elliptic curve specific methods:

- A field defined with a (pseudo-)Mersenne prime.
- Field construction using small irreducible polynomials
- Point representation with fast arithmetic
- EC with special properties.

Jacobian Coordinate System

A point can be represented in projective coordinates as (X, Y, Z), where $(X, Y, Z) = (x/z^c, y/z^d)$. If $c = 2, d = 3$, the coordinates are called Jacobian coordinates.

The traditional form of the curve is:

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

In a projective coordinate system, the equation changes. In the case of the Jacobian coordinates, the equation of the curve is now:

$$E : Y^2 = X^3 + axZ^4 + bZ^6.$$

The group law becomes...

Jacobian Coordinate System II

Point doubling:

$$X3 = (3(X1^2))^2 - 8X1Y1^2$$
$$Y3 = 3(X1^2)(4X1Y1^2 - X3) - 8(Y1^2)^2$$
$$Z3 = 2Y1Z1$$

Point addition:

$$X3 = (2(Y2Z1^3 - Y1Z2^3))^2 - (X2Z1^2 - X1Z2^2)(2(X2Z1^2 - X1Z2^2))$$
$$Y3 = 2(Y2Z1^3 - Y1Z2^3)(X1Z2^2(X2Z1^2 - X1Z2^2)^2 - X3) - 2Y1Z2^3$$
$$Z3 = (2Z1Z2)(X2Z1^2 - X1Z2^2)$$

... we better have a look at the "Explicit-Formulas Database".

w-NAF representation

A non-adjacent form (NAF) of a positive integer k is an expression: $k = \sum_{i=0}^{l-1} k_i 2^i$, where $k_i \in 0, \pm 1, k_{l-1} \neq 0$, and no two consecutive digits k_i are nonzero. The length of the NAF is l.

Let $w \geq 2$ be a positive integer. A width-w NAF of a positive integer k is also an expression $k = \sum_{i=0}^{l-1} k_i 2^i$, but where each nonzero coefficient k_i is odd, $|ki| < 2^{w-1}$, $k_{l-1} \neq 0$, and at most one of any w consecutive digits is nonzero. The length of the width-w NAF is l.

w-NAF representation

Express $k = \sum_{i=0}^{l-1} k_i 2^i$, where each coefficient k_i different than zero is odd, $2^{\omega-1} \leq k_i \leq 2^{\omega-1}$, $k_{l-1} \neq 0$

Ejemplo

Given $k = 1122334455$, the binary representation of k and the ω-NAF representations of k for $2 \leq \omega \leq 6$ are:

$$
\begin{array}{rcr}
(k)_2 & = & 1\,0\,0\,0\;\;0\,1\,0\,1\,1\;\;1\,0\,0\,1\,0\,1\;\;0\,1\,1\,1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1\,1\,1 \\
NAF_2(k) & = & 1\,0\,0\,0\;\;1\,0\,\bar{1}\,0\,0\;\;\bar{1}\,0\,1\,0\,\bar{1}\,0\;\;\bar{1}\,0\,0\,0\,\bar{1}\,0\,0\,\bar{1}\,0\,0\,0\,0\,\bar{1}\,0\,0\,\bar{1} \\
NAF_3(k) & = & 1\,0\,0\,0\;\;0\,0\,3\,0\,0\;\;\bar{1}\,0\,0\,1\,0\,0\;\;3\,0\,0\,0\,\bar{1}\,0\,0\,\bar{1}\,0\,0\,0\,0\,\bar{1}\,0\,0\,\bar{1} \\
NAF_4(k) & = & 1\,0\,0\,0\;\;0\,1\,0\,0\,0\;\;7\,0\,0\,0\,0\,5\;\;0\,0\,0\,7\,0\,0\,0\,7\,0\,0\,0\,\bar{1}\,0\,0\,0\,7 \\
NAF_5(k) & = & 1\,0\,0\,0\,0\,\bar{1}5\,0\,0\,0\,0\;\;\bar{9}\,0\,0\,0\,0\,0\,11\,0\,0\,0\,0\,0\,0\,\bar{9}\,0\,0\,0\,0\,0\,0\,0\,\bar{9} \\
NAF_6(k) & = & 1\,0\,0\,0\;\;0\,0\,0\,0\,0\,23\,0\,0\,0\,0\,0\,11\,0\,0\,0\,0\,0\,0\,\bar{9}\,0\,0\,0\,0\,0\,0\,0\,\bar{9}
\end{array}
$$

Double and add algorithm

Algorithm 1 Double-and-add scalar-point multiplication

Input: Positive integer k in base 2 representation, $P \in E(\mathbb{F}_{p^m})$
Ouput: kP

1: $Q \leftarrow \infty$
2: **for** $i = l - 1$ downto 0 **do**
3: $\quad Q \leftarrow [2]Q$
4: $\quad$ **if** $k_i = 1$ **then**
5: $\quad\quad Q \leftarrow Q + P$
6: $\quad$ **end if**
7: **end for**
8: **return** Q

Applying the algorithm

Algorithm 2 w-NAF multiplication

Input: Window width w, positive integer k, $P \in E(\mathbb{F}_{p^m})$
Ouput: kP

Compute the w-NAF expansion of k
Compute $P_i = iP$ for $i \in \{1, 3, 5, \ldots 2^{w-1} - 1\}$
$Q \leftarrow \infty$
for $i = l - 1$ downto 0 **do**
$Q \leftarrow [2]Q$
if $k_i \neq 0$ **then**
if $k_i > 0$ **then**
$Q \leftarrow Q + P_{k_i}$
else
$Q \leftarrow Q - P_{k_i}$
end if
end if
end for

The Comb method

For $k \in \mathbb{Z}^{+}$, let $t = |k|$ and $d = \lceil t/\omega \rceil$, where ω is the window size. The comb method works as follows,

1. Represent k in its [signed] binary form, such that $|k| = \omega d$
2. Divide the scalar k in ω-bit words, each of size d:

$$k \;=\; K^{\omega-1} \parallel \ldots \parallel K^{1} \parallel K^{0}$$

3. Write the K^j words as a matrix,

$$\begin{bmatrix} K^0 \\ \vdots \\ K^{\omega'} \\ \vdots \\ K^{\omega-1} \end{bmatrix} = \begin{bmatrix} K^0_{d-1} & \cdots & K^0_0 \\ \vdots & & \vdots \\ K^{\omega'}_{d-1} & \cdots & K^{\omega'}_0 \\ \vdots & & \vdots \\ K^{\omega-1}_{d-1} & \cdots & K^{\omega-1}_0 \end{bmatrix} = \begin{bmatrix} k_{d-1} & \cdots & k_0 \\ \vdots & & \vdots \\ k_{(\omega'+1)d-1} & \cdots & k_{\omega' d} \\ \vdots & & \vdots \\ k_{\omega d-1} & \cdots & k_{(\omega-1)d} \end{bmatrix}$$

4. Process sequentially each column of the scalar

Comb's method

Input: Window size ω, positive integer k, $P \in E(\mathbb{F}_q)$
Ouput: kP

1: Precompute Calculate $[a_{\omega-1}, \ldots, a_2, a_1, a_0]P$ for all bit combinations $(a_{\omega-1}, \ldots, a_2, a_1, a_0)$ of size ω
2: By padding k on the left with zeroes, write $k = K^{\omega-1} \parallel \ldots \parallel K^1 \parallel K^0$, where K^j is a word of length d. Represent each K_i^j as the i-th bit of the word K^j
3: $Q \leftarrow \mathcal{O}$
4: **for** $i = (d-1) \rightarrow 0$ **do**
5: $\quad Q \leftarrow 2Q$
6: $\quad Q \leftarrow Q + [K_i^{\omega-1}, \ldots, K_i^1, K_i^0]P$
7: **end for**
8: **return** Q

Variants of the Comb method

- Combs method are only useful in the context when the point P is known in advance [such as in ECDSA key generation and signature primitives]

Variants of the Comb method

- Combs method are only useful in the context when the point P is known in advance [such as in ECDSA key generation and signature primitives]
- It is possible to generalize the Comb method using two or more precomputed tables as discussed by Hankerson, Menezes and Vanstone in their famous book

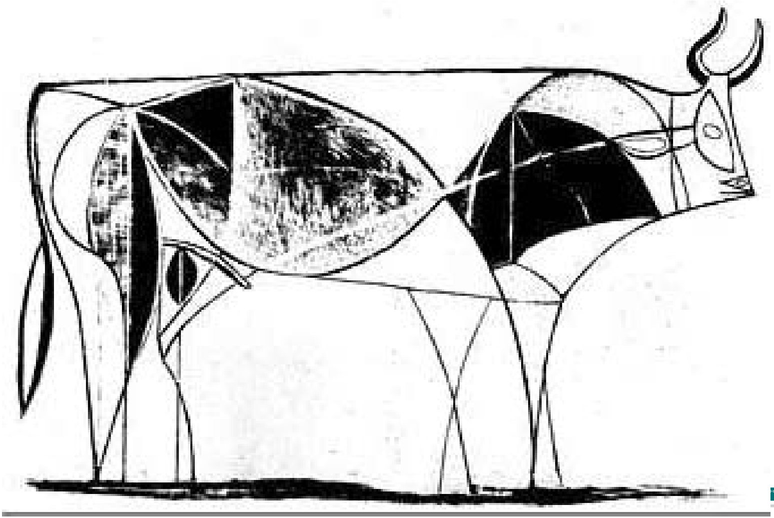
Variants of the Comb method

- Combs method are only useful in the context when the point P is known in advance [such as in ECDSA key generation and signature primitives]
- It is possible to generalize the Comb method using two or more precomputed tables as discussed by Hankerson, Menezes and Vanstone in their famous book
- Lim and Lee gave more flexible methods for performing the comb algorithm

Variants of the Comb method

- Combs method are only useful in the context when the point P is known in advance [such as in ECDSA key generation and signature primitives]
- It is possible to generalize the Comb method using two or more precomputed tables as discussed by Hankerson, Menezes and Vanstone in their famous book
- Lim and Lee gave more flexible methods for performing the comb algorithm
- In eprint 2012/309, Hamburg presented a signed multi-comb algorithm that nicely allows the saving of half of the precomputed points. Hamburg's representation writes the scalar in a signed binary representation where the bits can only get the values of ± 1

Pablo Picasso: The bull challenge (8/11)



Endomorphisms

Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$ with the point at infinity denoted by $\mathcal{O}$.

An endomorphism of E is a map $\phi : E \to E$ such that $\phi(\mathcal{O}) = \mathcal{O}$ and $\phi(P) = (g(P), h(P))$, for all P in the curve and where g, h are rational functions with coefficients in $\mathbb{F}_q$. The characteristic polynomial of an endomorphism ϕ is the monic polynomial $f(X)$ of least degree in $\mathbb{Z}[X]$ such that $f(\phi) = 0$.

Examples

Example 1. The p^{th} power map $\phi : E \to E$ defined by $(x, y) \mapsto (x^p, y^p)$ and $\mathcal{O} \mapsto \mathcal{O}$ is an endomorphism defined over $\mathbb{F}_p$, called the Frobenius endomorphism.

This endomorphism is usually denoted as π, and is normally quite fast as it can be efficiently computed

Examples II

Example 3. Let $p \equiv 1 \pmod 4$ be a prime, and consider the following elliptic curve

$$E_1 : y^2 = x^3 + ax.$$

defined over $\mathbb{F}_p$. Let $\alpha \in \mathbb{F}_p$. Then, the map $\phi : E_1 \to E_1$ defined by $(x, y) \mapsto (-x, \alpha y)$ and $\mathcal{O} \mapsto \mathcal{O}$ is an endomorphism defined over $\mathbb{F}_p$.

If $P \in E(\mathbb{F}_p)$ is a point of prime order r, then ϕ acts on $\langle P \rangle$ as a multiplication map $[\lambda]$, in essence: $\phi(Q) = \lambda Q$, $\forall A \in \langle P \rangle$, with $\lambda^2 \equiv -1 \pmod r$

Examples III

Example 3. Let $p \equiv 1 \pmod 3$ be a prime, and consider the following elliptic curve

$$E_2 : y^2 = x^3 + b.$$

defined over $\mathbb{F}_p$. Let $\beta \in \mathbb{F}_p$. Then, the map $\phi : E_2 \to E_2$ defined by $(x, y) \mapsto (\beta x, y)$ and $\mathcal{O} \mapsto \mathcal{O}$ is an endomorphism defined over $\mathbb{F}_p$.

If $P \in E(\mathbb{F}_p)$ is a point of prime order r, then ϕ acts on $\langle P \rangle$ as a multiplication map $[\lambda]$, in essence: $\phi(Q) = \lambda Q$, $\forall A \in \langle P \rangle$, with $\lambda^2 + \lambda \equiv -1 \pmod r$

GLV Method

- In 2001 Gallant, Lambert and Vanstone presented a method that allows to speedup the scalar multiplication kP in $E(\mathbb{F}_p)[r]$ by taking advantage of certain properties of some elliptic curve families. In short, the method will work whenever given a point P one can get a non-trivial multiple of it in an efficient manner

GLV Method

- This will work provided that there exists an endomorphism ψ that can be efficiently computed over $E/\mathbb{F}_p$ such that $\psi(P) = \lambda P$, where $\lambda \in \mathbb{Z}_r$.

GLV Method

- This will work provided that there exists an endomorphism ψ that can be efficiently computed over $E/\mathbb{F}_p$ such that $\psi(P) = \lambda P$, where $\lambda \in \mathbb{Z}_r$.
- In the case of BN curves (an in general, all elliptic curves with j-invariant zero), $\psi : E_1 \to E_1$ defined as, $(x, y) \to (\beta x, y)$, where $\beta \in \mathbb{F}_p$ is an element of order three and it can be easily checked that λ satisfies, $\lambda^2 + \lambda \equiv -1 \pmod{r}$.

GLV Method

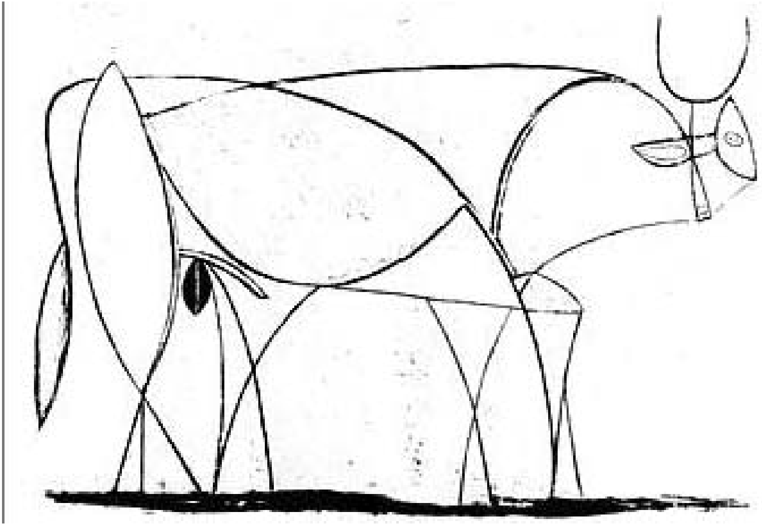
- This will work provided that there exists an endomorphism ψ that can be efficiently computed over $E/\mathbb{F}_p$ such that $\psi(P) = \lambda P$, where $\lambda \in \mathbb{Z}_r$.
- In the case of BN curves (an in general, all elliptic curves with j-invariant zero), $\psi : E_1 \to E_1$ defined as, $(x, y) \to (\beta x, y)$, where $\beta \in \mathbb{F}_p$ is an element of order three and it can be easily checked that λ satisfies, $\lambda^2 + \lambda \equiv -1 \pmod{r}$.
- hence, it is possible to speedup the computation of kP by writing $k \equiv k_0 + k_1\lambda \pmod{r}$ with $|k_i| < \sqrt{r}$ followed by a simultaneous multiplication $k_0P + k_1\psi(P)$

GLV Method

Input: Positive integer k, $P \in E(\mathbb{F}_p)$, endomorphism ψ over $E(\mathbb{F}_p)$.
Ouput: kP

$Q \leftarrow \psi(P)$ $(= \lambda P)$
Decompose k as, $k = u + v\lambda$ where $|u| = |v| = l$
[using egcd or lattice methods]
Obtain the w-NAF representation of u and v
$R \leftarrow \mathcal{O}$
for $i = l - 1 \to 0$ **do**
  $R \leftarrow 2R$
  if $u_i \neq 0$ **then**
    $R \leftarrow R + P$
  end if
  if $v_i \neq 0$ **then**
    $R \leftarrow R + Q$
  end if
end for
return R

Pablo Picasso: The bull challenge (9/11)



Introduction

Galbraith and Scott, and Galbraith, Linn and Scott in showed a technique for generalizing the GLV method for higher powers of the endomorphism for the groups $\mathbb{G}_2$ and $\mathbb{G}_T$ [to be defined next!].

To get an m-dimensional expansion

$$n \equiv n_0 + n_1\lambda + \cdots + n_{m-1}\lambda^{m-1} \pmod{r}$$

of $[n]P$, one must decompose n with powers of λ sufficiently different and modulo r.

The method then solves a closest vector problem in a lattice using Babai's rounding off method. A reduced lattice basis, however, must be precomputed in order to get an efficient implementation.

Decomposition

For a pairing friendly elliptic curve family, it is possible to get a "natural" m-dimensional expansion with $m = \varphi(k)$, where $\varphi(k)$ is the Euler totient function on k, the embedding degree of the family.

The modular lattice basis is defined as, by:

$$L = \left\{ x \in \mathbb{Z}^m : \sum_{i=0}^{m-1} x_i \lambda^i \equiv 0 \pmod{r} \right\}$$

where $\lambda = T = t - 1$. This m-dimensional modular lattice L will be used to construct a $m \times m$ matrix. Then, one can fill the matrix with any linear combination of $\lambda : L_{i,j} \equiv 0 \pmod{r}$.

Summary of the GLS Method [EuroCrypt'09]

The GLS method can be seen as a version of the GLV method, where the endomorphism $\psi = \phi^{-1}\pi_p\phi$ de E' such that $\psi : E'(\mathbb{F}_{p^{k/d}}) \rightarrow E'(\mathbb{F}_{p^{k/d}})$, where π_p is the Frobenius operator defined as,

$$\pi_p : E(\mathbb{F}_{p^k}) \rightarrow E(\mathbb{F}_{p^k}) : (X, Y) \mapsto (X^p, Y^p) \in E(\mathbb{F}_{p^k})$$

In the case of BN curves one has that the following identity holds, $\psi^4 - \psi^2 + 1 = 0$, which can be seen as a scalar multiplication by p. Since $p \equiv t - 1 \pmod{r}$ and $|t - 1| \approx \frac{1}{4}|r|$, the scalar k can be decomposed as $k \equiv k_0 + k_1\lambda + k_2\lambda^2 + k_3\lambda^3 \pmod{r}$, para $\lambda = t - 1$.

GLS Method [EuroCrypt'09]

Input: A positive integer k, $Q \in E(\mathbb{F}_{p^2})$, endomorphism $\psi = \phi^{-1}\pi_p\phi$ over $E(\mathbb{F}_{p^2})$.
Ouput: kQ

```
R_0 ← Q, R_1 ← ψ(Q), R_2 ← ψ^2(Q), R_3 ← ψ^3(Q)
Decompose k = k_0 + k_1 λ + k_2 λ^2 + k_3 λ^3 where |k_i| = l
Represent k_i = Σ_{j=0}^{l-1} k_ij 2^j
R ← O
for i = l − 1 → 0 do
    R ← 2R
    if k_0i ≠ 0 then
        R ← R + R_0
    if k_1i ≠ 0 then
        R ← R + R_1
    if k_2i ≠ 0 then
        R ← R + R_2
    if k_3i ≠ 0 then
        R ← R + R_3
end for
return R
```

Definition of a pairing

Here, we define a **pairing** as a map: $G_2 \times G_1 \to G_T$.

These groups are finite and cyclic. G_1 and G_2 are additively-written and both of them are of prime order r, $G_1 \subseteq E(\mathbb{F}_p)$, and $G_2 \subseteq E(\mathbb{F}_{p^d})$.

G_T, is multiplicatively-written and of order r, $G_T \subseteq \mu_r$ or just $\mathbb{F}_{p^k}^*$

Properties:

- Bilinearity
- Non-degeneracy
- Efficiently computable

Scalar-point multiplication and exponentiation in pairings

The most important property of a pairing is the bilinearity, denoted as:

$$e([a]Q,[b]P) = e([b]Q,[a]P) = e(Q,[ab]P) = e(Q,P)^{ab}$$

where $Q \in G_2$, $P \in G_1$, and the result is in G_T.

A multiplication in G_2 is much more expensive than in G_1, it is wise to place such operation in the smaller group.

It is also know that an exponentiation in G_T is cheaper than a pairing computation, some protocol designers try to exploit this too.

Security in pairings (1/3)

- A pairing-based cryptosystem is considered secure if the discrete log problem is computationally intractable:
  - In the subgroup $\mathbb{F}^*_{p^k}$, finding the solution to $g^x = h \in \mathbb{F}^*_{p^k}$

Security in pairings (1/3)

- A pairing-based cryptosystem is considered secure if the discrete log problem is computationally intractable:
  - In the subgroup $\mathbb{F}^*_{p^k}$, finding the solution to $g^x = h \in \mathbb{F}^*_{p^k}$
  - In the group $E[r]$, given xP y Q, find the integer x such that $xP = Q \in E[r]$.

Security in pairings (1/3)

- A pairing-based cryptosystem is considered secure if the discrete log problem is computationally intractable:
  - In the subgroup $\mathbb{F}_{p^k}^*$, finding the solution to $g^x = h \in \mathbb{F}_{p^k}^*$
  - In the group $E[r]$, given xP y Q, find the integer x such that $xP = Q \in E[r]$.
- hence, the security guarantees of a pairing are measured with respect to $\log_2(r)$ y $\log_2(p^k)$.

Security in pairings (1/3)

- A pairing-based cryptosystem is considered secure if the discrete log problem is computationally intractable:
  - In the subgroup $\mathbb{F}^*_{p^k}$, finding the solution to $g^x = h \in \mathbb{F}^*_{p^k}$
  - In the group $E[r]$, given xP y Q, find the integer x such that $xP = Q \in E[r]$.
- hence, the security guarantees of a pairing are measured with respect to $\log_2(r)$ y $\log_2(p^k)$.
- the ratio between this two parameters is captured by $k \cdot \rho$, where $\rho = \log_2(p)/\log_2(r)$.

Cinvestav
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional
50 AÑOS

Security in pairings (1/3)

- A pairing-based cryptosystem is considered secure if the discrete log problem is computationally intractable:
  - In the subgroup $\mathbb{F}^*_{p^k}$, finding the solution to $g^x = h \in \mathbb{F}^*_{p^k}$
  - In the group $E[r]$, given xP y Q, find the integer x such that $xP = Q \in E[r]$.
- hence, the security guarantees of a pairing are measured with respect to $\log_2(r)$ y $\log_2(p^k)$.
- the ratio between this two parameters is captured by $k \cdot \rho$, where $\rho = \log_2(p)/\log_2(r)$.

Security in pairings (2/3)

- In the following it is shown an estimation of the required embedding degree for different lengths in bits of p and r, whic are required to obtain the level of security achieved by the pairing

Security level (bits)	r bitlength $\log_2(r)$	p^k bitlength $\log_2(p^k)$	embedding degree k	
			$\rho \approx 1$	$\rho \approx 2$
80	160	960 - 1280	6 - 8	3 - 4
112	224	2200 - 3600	10 - 16	5 - 8
128	256	3000 - 5000	12 - 20	6 - 10
192	384	8000 - 10000	20 - 26	10 - 13
256	512	14000 - 18000	28 - 36	14 - 18

Security in pairings (2/3)

- In the following it is shown an estimation of the required embedding degree for different lengths in bits of p and r, whic are required to obtain the level of security achieved by the pairing

Security level (bits)	r bitlength $\log_2(r)$	p^k bitlength $\log_2(p^k)$	embedding degree k $\rho \approx 1$	$\rho \approx 2$
80	160	960 - 1280	6 - 8	3 - 4
112	224	2200 - 3600	10 - 16	5 - 8
128	256	3000 - 5000	12 - 20	6 - 10
192	384	8000 - 10000	20 - 26	10 - 13
256	512	14000 - 18000	28 - 36	14 - 18

Optimal ate pairing

Given the elliptic curve $E/\mathbb{F}_p$ with embedding degree k and order $\#E(\mathbb{F}_p) = p + 1 - t$, where t is the Frobenius trace of E over $\mathbb{F}_p$. given the points $P \in E(\mathbb{F}_p)[r]$ and $Q \in E(\mathbb{F}_{p^2})[r]$, the optimal ate pairing $\hat{a}$ is defined as,

$$\hat{a}(Q, P) = f_{t-1,Q}(P)^{(p^k-1)/r}$$

where $f_{t-1,Q}$ can be recursively computed using the doubling and add method for computing lines:

$$f_{a+1,R} = f_{a,R} \cdot \ell_{aR,R} \quad \text{and} \quad f_{2a,R} = f_{a,R}^2 \cdot \ell_{aR,aR}$$

Optimal ate pairing

Given the elliptic curve $E/\mathbb{F}_p$ with embedding degree k and order $\#E(\mathbb{F}_p) = p + 1 - t$, where t is the Frobenius trace of E over $\mathbb{F}_p$. given the points $P \in E(\mathbb{F}_p)[r]$ and $Q \in E(\mathbb{F}_{p^2})[r]$, the optimal ate pairing $\hat{a}$ is defined as,

$$\hat{a}(Q, P) = f_{t-1,Q}(P)^{(p^k-1)/r}$$

where $f_{t-1,Q}$ can be recursively computed using the doubling and add method for computing lines:

$$f_{a+1,R} = f_{a,R} \cdot \ell_{aR,R} \quad \text{and} \quad f_{2a,R} = f_{a,R}^2 \cdot \ell_{aR,aR}$$

Miller's loop

Input: $r = (r_{l-1}, \ldots, r_1, r_0)_2,\ Q, P \in E(\bar{\mathbb{F}}_p)$ such that $P \neq Q$
Ouput: $f_{r,Q}(P)$

1: $T \leftarrow Q,\ f \leftarrow 1$
2: **for** $i = l-2 \rightarrow 0$ **do**
3: $\quad f \leftarrow f^2 \cdot \ell_{T,T}(P),\ T \leftarrow 2T$
4: $\quad$ **if** $r_i = 1$ **then**
5: $\quad\quad f \leftarrow f \cdot \ell_{T,Q}(P),\ T \leftarrow T + Q$
6: $\quad$ **end if**
7: **end for**
8: **return** f

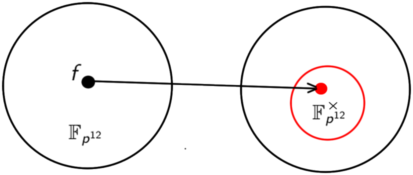
Final Exponentiation

One can represent $(p^k - 1)/r$ as the product of two exponents,

$$\frac{p^k - 1}{r} = \frac{p^k - 1}{\Phi_k(p)} \cdot \frac{\Phi_k(p)}{r}$$

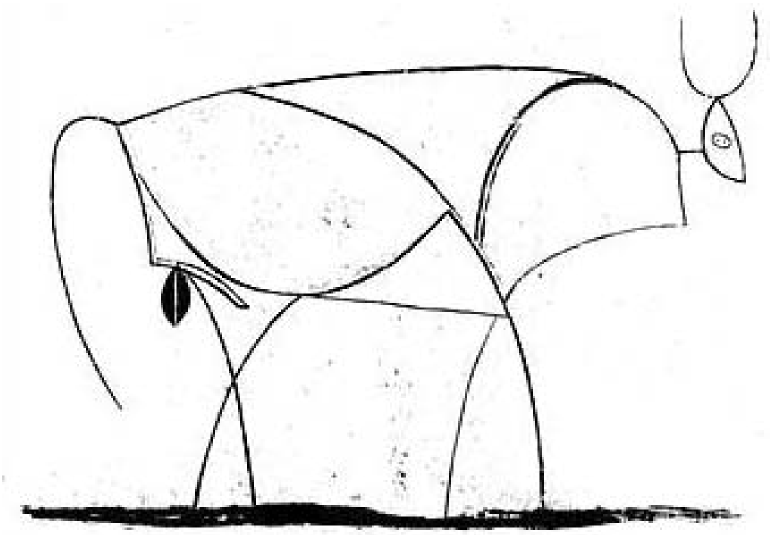
where $\Phi_k(p)$ is the k-th cyclotomic polynomial evaluated in p.In the case of BN curves where $k = 12$ one has,

$$\frac{p^{12} - 1}{r} = (p^6 - 1) \cdot (p^2 + 1) \cdot \frac{p^4 - p^2 + 1}{r}$$



$\mathbb{F}_{p^{12}}^{\times} \subset \mathbb{F}_{p^{12}}$

Pablo Picasso: The bull challenge (10/11)



Pairing algorithm/Multipairing

Basic Miller loop + final exponentiation

Input: $P \in G_1,\ Q \in G_2$
Ouput: $f \in G_T$

$f \leftarrow 1,\ T \leftarrow P,\ i \leftarrow \lfloor \mathrm{Log}_2(r) \rfloor - 1$
while $i \geq 0$ **do**
  $f \leftarrow f^2 \cdot L_{T,T}(Q)$
  $T \leftarrow 2T$
  if $s_i[i+1] = 1$ **then**
    $f \leftarrow f \cdot L_{T,P}(Q)$
    $T \leftarrow T + P$
  end if
  $i \leftarrow i - 1$
end while
$f^{(p^k-1)/r}$

Pairing algorithm/Multipairing

Basic Miller loop + final exponentiation

Input: $P \in G_1,\ Q \in G_2$
Ouput: $f \in G_T$

$f \leftarrow 1,\ T \leftarrow P,\ i \leftarrow \lfloor \mathrm{Log}_2(r) \rfloor - 1$
while $i \geq 0$ **do**
  $f \leftarrow f^2 \cdot L_{T,T}(Q)$
  $T \leftarrow 2T$
  if $s_i[i+1] = 1$ **then**
    $f \leftarrow f \cdot L_{T,P}(Q)$
    $T \leftarrow T + P$
  end if
  $i \leftarrow i - 1$
end while
$f^{(p^k-1)/r}$

Pairing algorithm/Multipairing

Basic Miller loop + final exponentiation

Input: $P \in G_1,\ Q \in G_2$
Ouput: $f \in G_T$

$f \leftarrow 1,\ T_j \leftarrow P_j,\ i \leftarrow \lfloor \mathrm{Log}_2(r) \rfloor - 1$
while $i \geq 0$ **do**
  $f \leftarrow f^2 \cdot L_{T_j,T_j}(Q)$
  $T_j \leftarrow 2T_j$
  if $s_i[i+1] = 1$ **then**
    $f \leftarrow f \cdot L_{T_j,P_j}(Q)$
    $T_j \leftarrow T_j + P_j$
  end if
  $i \leftarrow i - 1$
end while
$f^{(p^k-1)/r}$

Pairing algorithm/Multipairing

Basic Miller loop + final exponentiation

Input: $P \in G_1,\ Q \in G_2$
Ouput: $f \in G_T$

$f \leftarrow 1,\ T_j \leftarrow P_j,\ i \leftarrow \lfloor \mathrm{Log}_2(r) \rfloor - 1$
while $i \geq 0$ **do**
  $f \leftarrow f^2 \cdot L_{T_j,T_j}(Q)$
  $T_j \leftarrow 2T_j$
  if $s_i[i+1] = 1$ **then**
    $f \leftarrow f \cdot L_{T_j,P_j}(Q)$
    $T_j \leftarrow T_j + P_j$
  end if
  $i \leftarrow i - 1$
end while
$f^{(p^k-1)/r}$

Pairing algorithm/Multipairing

Basic Miller loop + final exponentiation

Input: $P \in G_1$, $Q \in G_2$
Ouput: $f \in G_T$

$f \leftarrow 1,\ T_j \leftarrow P_j,\ i \leftarrow \lfloor \mathrm{Log}_2(r) \rfloor - 1$
while $i \geq 0$ **do**
  $f \leftarrow f^2 \cdot L_{T_j,T_j}(Q)$
  $T_j \leftarrow 2T_j$
  if $s_i[i+1] = 1$ **then**
    $f \leftarrow f \cdot L_{T_j,P_j}(Q)$
    $T_j \leftarrow T_j + P_j$
  end if
  $i \leftarrow i - 1$
end while
$f^{(p^k-1)/r}$

GLS on G_T

In the case of this method, the efficiently computable endomorphism is the Frobenius endomorphism, this is because:

$$p \equiv t - 1 \bmod r$$

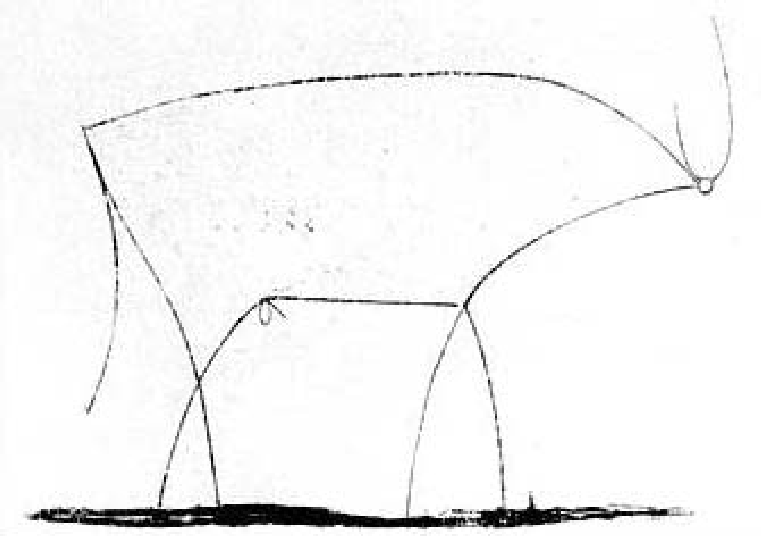
Hence,

$$e^k = e^{k_0} \cdot e^{k_1^p} \cdot e^{k_2^{p^2}} \cdots e^{k_1^{p^{m-1}}}$$

where $e \in G_T$, $k \in \mathbb{Z}_r$, m is the degree of the decomposition, and the exponentiation to the p is done using the Frobenius endomorphism.

We can use the same method for decomposing the exponent, and applying the corresponding endomorphism.

Pablo Picasso: The bull challenge (11/11)



Thank you for your attention

Questions?