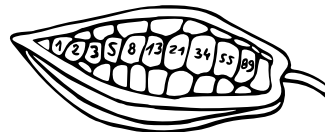

The Elliptic Curve Factorization method

Laurent Fousse



C A C A O

laurent@komite.net

LORIA – Université Henri Poincaré Nancy I

Outline

1. Introduction
2. Factorization method principle
3. Elliptic Curves
4. Fast modular multiplication
5. Fast polynomial evaluation
6. ECM in practice: GMP-ECM
7. Conclusion

Why factor?

- interesting arithmetic problems

Why factor?

- interesting arithmetic problems
- factorize numbers of a given family (Fermat, Cunningham)

Why factor?

- interesting arithmetic problems
- factorize numbers of a given family (Fermat, Cunningham)
- link to cryptography

Why factor?

- interesting arithmetic problems
- factorize numbers of a given family (Fermat, Cunningham)
- link to cryptography
- used for primality proving

Why factor?

- interesting arithmetic problems
- factorize numbers of a given family (Fermat, Cunningham)
- link to cryptography
- used for primality proving
- for fun and records breaking

Factorization method principle

Let n be the number to be factored, and p an unknown prime factor of n .

Factorization method principle

Let n be the number to be factored, and p an unknown prime factor of n .

1. Pick an abelian group G_p where operations are compatible with modular arithmetics.

Factorization method principle

Let n be the number to be factored, and p an unknown prime factor of n .

1. Pick an abelian group G_p where operations are compatible with modular arithmetics.
2. Perform all operations mod n in the structure G_n .

Factorization method principle

Let n be the number to be factored, and p an unknown prime factor of n .

1. Pick an abelian group G_p where operations are compatible with modular arithmetics.
2. Perform all operations mod n in the structure G_n .
3. Operations done mod n reduce naturally to G_p .

Factorization method principle

Let n be the number to be factored, and p an unknown prime factor of n .

1. Pick an abelian group G_p where operations are compatible with modular arithmetics.
2. Perform all operations mod n in the structure G_n .
3. Operations done mod n reduce naturally to G_p .
4. Pick an element $P_0 \in G_n$, a positive integer x and compute $x \cdot P$.

Factorization method principle

Let n be the number to be factored, and p an unknown prime factor of n .

1. Pick an abelian group G_p where operations are compatible with modular arithmetics.
2. Perform all operations mod n in the structure G_n .
3. Operations done mod n reduce naturally to G_p .
4. Pick an element $P_0 \in G_n$, a positive integer x and compute $x \cdot P$.
5. If $o_{G_p}(P_0) \mid x$ the computation of $x \cdot P_0$ will give a factor of n (probably).

Factorization method principle

Example:

Factorization method principle

Example:

- For the P-1 factorization method, we have $G_p = (\mathbb{Z}/p\mathbb{Z})^*$.

Factorization method principle

Example:

- For the P-1 factorization method, we have $G_p = (\mathbb{Z}/p\mathbb{Z})^*$.
- For the P+1 factorization method, we have $G_p = GF(p^2)^*$

Factorization method principle

Example:

- For the P-1 factorization method, we have $G_p = (\mathbb{Z}/p\mathbb{Z})^*$.
- For the P+1 factorization method, we have $G_p = GF(p^2)^*$
(more correctly $GF(p^2)^* / GF(p)^*$).

Factorization method principle

Example:

- For the P-1 factorization method, we have $G_p = (\mathbb{Z}/p\mathbb{Z})^*$.
- For the P+1 factorization method, we have $G_p = GF(p^2)^*$
(more correctly $GF(p^2)^*/GF(p)^*$).
- For ECM, we chose an elliptic curve $E_{a,b}$.

A simple P-1 example

Take $n = 10090019171$, $P_0 = 42$ and $x = 42$.

A simple P-1 example

Take $n = 10090019171$, $P_0 = 42$ and $x = 42$.

We notice that

$$\gcd(P_0^x - 1, n) = 1009$$

so we found a non trivial factor $p = 1009$ of n .

A simple P-1 example

Take $n = 10090019171$, $P_0 = 42$ and $x = 42$.

We notice that

$$\gcd(P_0^x - 1, n) = 1009$$

so we found a non trivial factor $p = 1009$ of n .

The order of 42 in $(\mathbb{Z}/p\mathbb{Z})^*$ is 21 and $21|42$.

A simple P-1 example

Take $n = 10090019171$, $P_0 = 42$ and $x = 42$.

We notice that

$$\gcd(P_0^x - 1, n) = 1009$$

so we found a non trivial factor $p = 1009$ of n .

The order of 42 in $(\mathbb{Z}/p\mathbb{Z})^*$ is 21 and $21|42$.

Moreover $p - 1 = 2^4 \cdot 3^2 \cdot 7$

A simple P-1 example

Take $n = 10090019171$, $P_0 = 42$ and $x = 42$.

We notice that

$$\gcd(P_0^x - 1, n) = 1009$$

so we found a non trivial factor $p = 1009$ of n .

The order of 42 in $(\mathbb{Z}/p\mathbb{Z})^*$ is 21 and $21|42$.

Moreover $p - 1 = 2^4 \cdot 3^2 \cdot 7$

$n = p \cdot q$ where q is prime and $q - 1 = 2 \cdot 7^2 \cdot 67 \cdot 1523$

A simple P-1 example

Take $n = 10090019171$, $P_0 = 42$ and $x = 42$.

We notice that

$$\gcd(P_0^x - 1, n) = 1009$$

so we found a non trivial factor $p = 1009$ of n .

The order of 42 in $(\mathbb{Z}/p\mathbb{Z})^*$ is 21 and $21|42$.

Moreover $p - 1 = 2^4 \cdot 3^2 \cdot 7$

$n = p \cdot q$ where q is prime and $q - 1 = 2 \cdot 7^2 \cdot 67 \cdot 1523$

and the order of 42 in $(\mathbb{Z}/q\mathbb{Z})^*$ is $102041 > 42$.

A simple P-1 example

Take $n = 10090019171$, $P_0 = 42$ and $x = 42$.

We notice that

$$\gcd(P_0^x - 1, n) = 1009$$

so we found a non trivial factor $p = 1009$ of n .

The order of 42 in $(\mathbb{Z}/p\mathbb{Z})^*$ is 21 and $21|42$.

Moreover $p - 1 = 2^4 \cdot 3^2 \cdot 7$

$n = p \cdot q$ where q is prime and $q - 1 = 2 \cdot 7^2 \cdot 67 \cdot 1523$

and the order of 42 in $(\mathbb{Z}/q\mathbb{Z})^*$ is $102041 > 42$.

So we were able to factorize n .

Elliptic Curve

An elliptic curve over a field K of characteristic other than 2 or 3 is the set of point (X, Y) such that

$$Y^2 = X^3 + AX + B$$

where $A, B \in K$ and $4A^3 + 27B^3 \neq 0$, and a point at infinity 0_E .

Elliptic Curve

An elliptic curve over a field K of characteristic other than 2 or 3 is the set of point (X, Y) such that

$$Y^2 = X^3 + AX + B$$

where $A, B \in K$ and $4A^3 + 27B^3 \neq 0$, and a point at infinity 0_E .

This is called the Weierstrass representation of the curve.

Elliptic Curve

An elliptic curve over a field K of characteristic other than 2 or 3 is the set of point (X, Y) such that

$$Y^2 = X^3 + AX + B$$

where $A, B \in K$ and $4A^3 + 27B^3 \neq 0$, and a point at infinity 0_E .

This is called the Weierstrass representation of the curve.

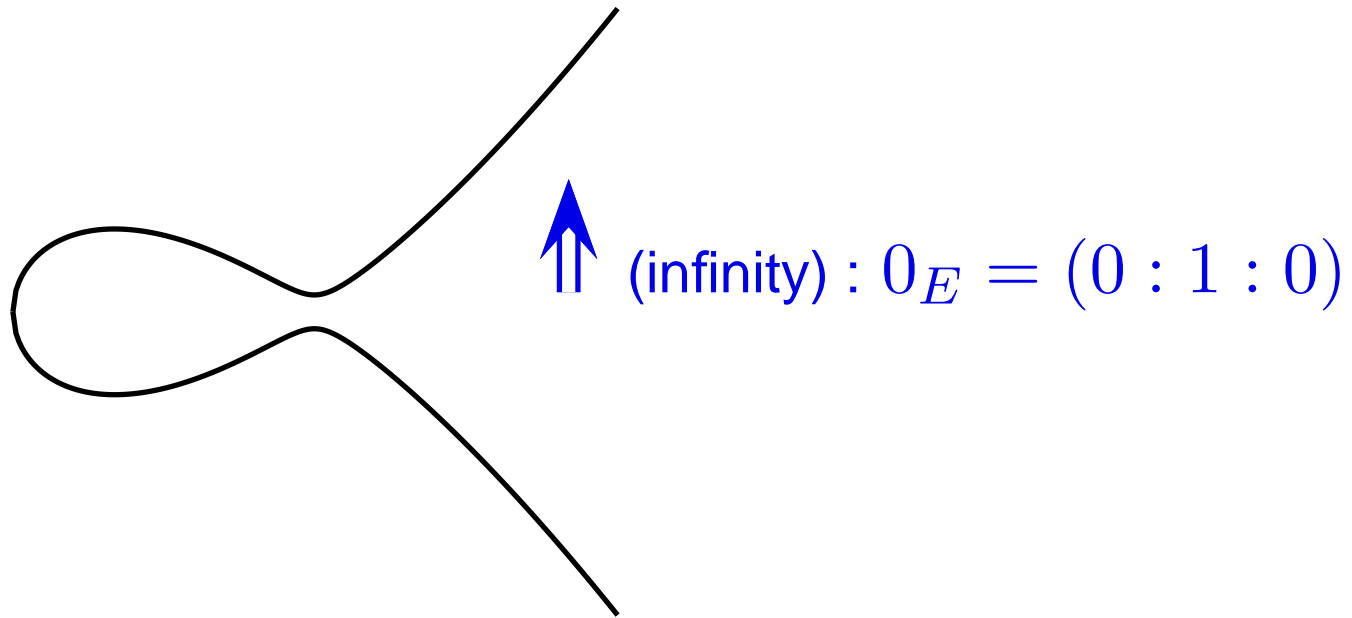
Switching to Montgomery and homogeneous form the equation of the curve becomes

$$by^2z = x^3 + ax^2z + xz^2$$

which is more suited for computations.

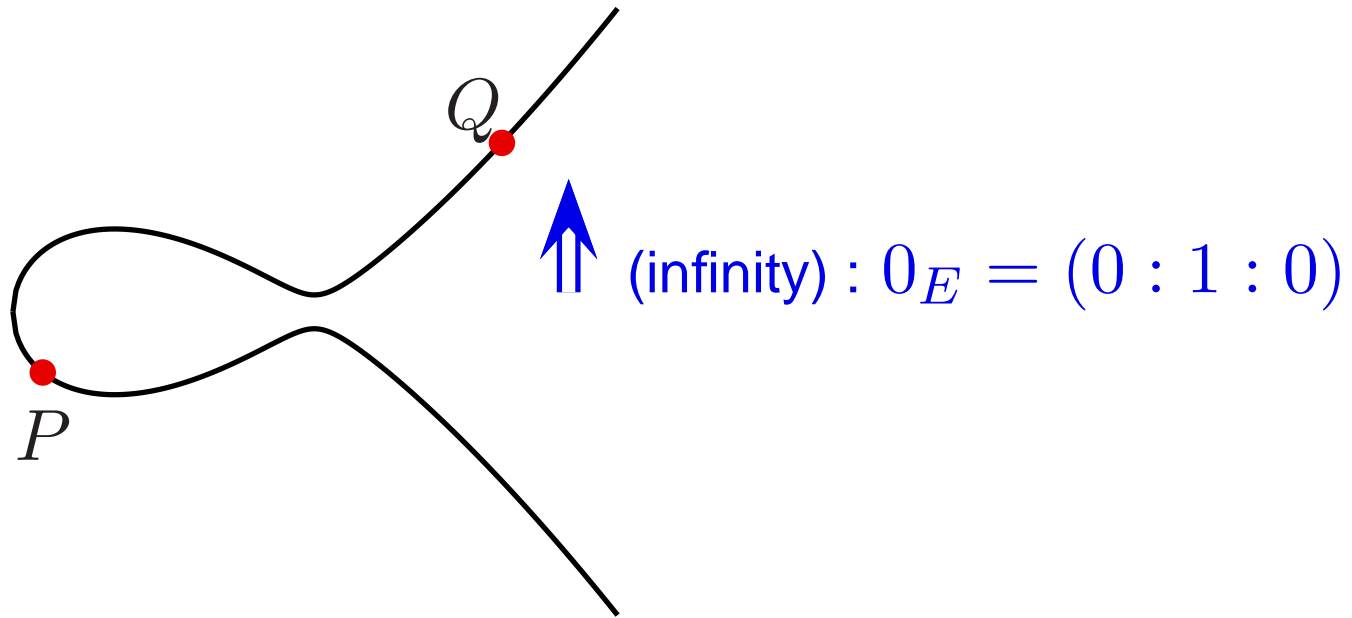
Elliptic Curve

The geometric interpretation of the group operation is that three aligned points sum to zero.



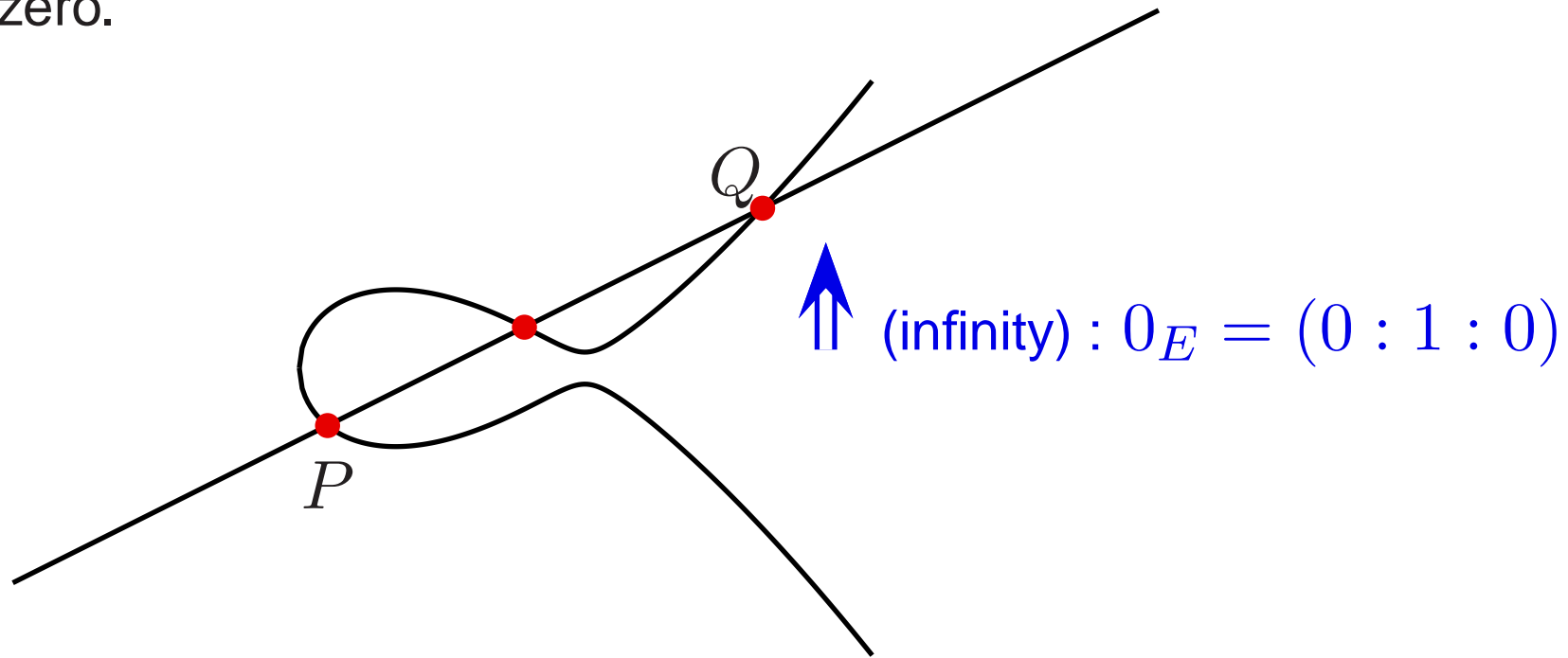
Elliptic Curve

The geometric interpretation of the group operation is that three aligned points sum to zero.



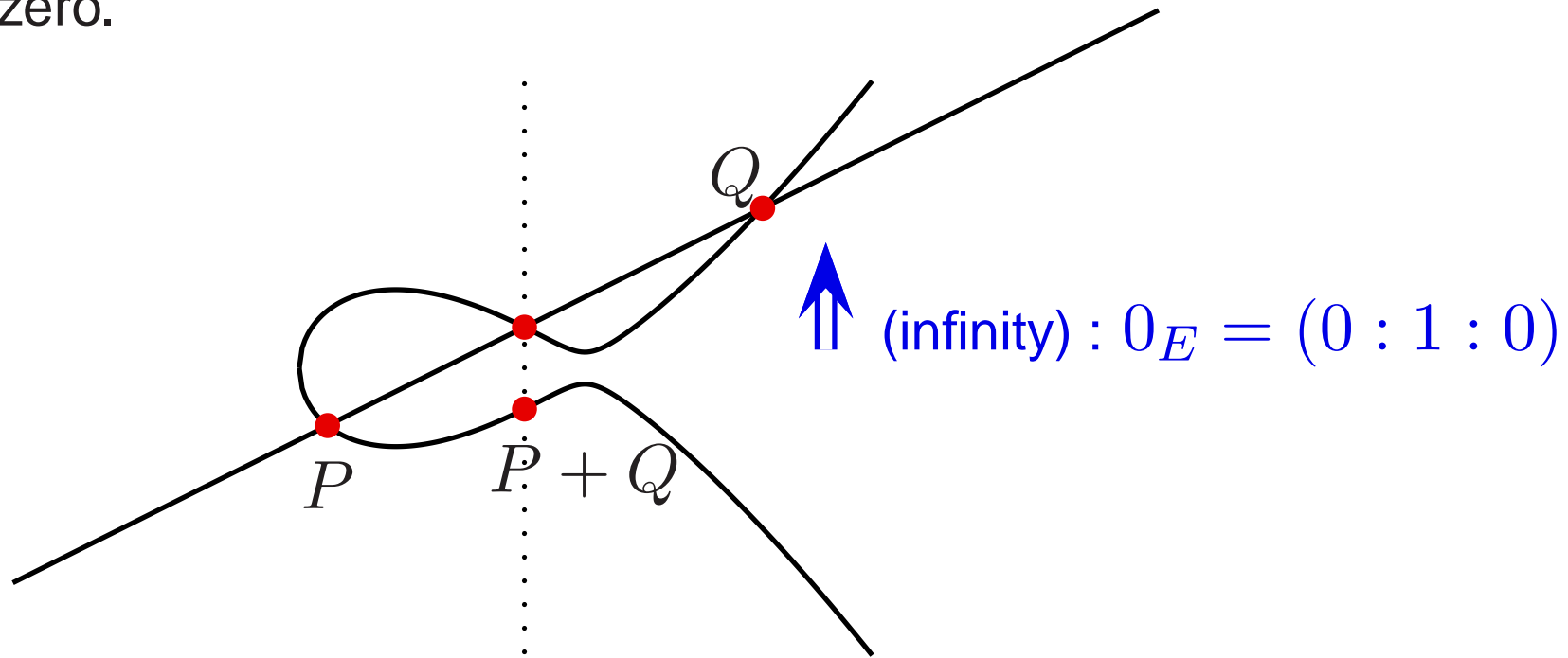
Elliptic Curve

The geometric interpretation of the group operation is that three aligned points sum to zero.



Elliptic Curve

The geometric interpretation of the group operation is that three aligned points sum to zero.



Arithmetic on the curve

- A point on the curve is a triplet $(x : y : z)$.

Arithmetic on the curve

- A point on the curve is a triplet $(x : y : z)$.
- We can forget y and identify P with $-P$ to get simpler formulas.

Arithmetic on the curve

- A point on the curve is a triplet $(x : y : z)$.
- We can forget y and identify P with $-P$ to get simpler formulas.

To compute $P + Q$:

$$x_{P+Q} = 4z_{P-Q} \cdot (x_P x_Q - z_P z_Q)^2$$

$$z_{P+Q} = 4x_{P-Q} \cdot (x_P z_Q - z_P x_Q)^2$$

so we need to have computed $P - Q$ already. Formulas for doubling omitted.

Comparison of ECM with other methods

Group	Order
$\mathbb{Z}/p\mathbb{Z}$	$p - 1 = \Pi_1(p)$
$GF(p^2)$	$p + 1 = \Pi_2(p)$
“Generic cyclotomic”	$\Pi_d(p)$
$E_{a,b} \bmod p$	$ o - (p + 1) < 2\sqrt{p}$

Complexity of ECM

Let $L_{\alpha,c}(p) = e^{c(\log p)^\alpha (\log \log p)^{1-\alpha}}$. Then the expected time complexity of ECM to find a factor p of n is

$$O(L_{\frac{1}{2},\sqrt{2}}(p)M(\log n))$$

where $M(\log n)$ is the complexity of multiplication mod n .

Complexity of ECM

Let $L_{\alpha,c}(p) = e^{c(\log p)^\alpha (\log \log p)^{1-\alpha}}$. Then the expected time complexity of ECM to find a factor p of n is

$$O(L_{\frac{1}{2},\sqrt{2}}(p)M(\log n))$$

where $M(\log n)$ is the complexity of multiplication mod n .

• for NFS $O(L_{\frac{1}{3},c}(n))$ (where $c < 2$).

Complexity of ECM

Let $L_{\alpha,c}(p) = e^{c(\log p)^\alpha (\log \log p)^{1-\alpha}}$. Then the expected time complexity of ECM to find a factor p of n is

$$O(L_{\frac{1}{2},\sqrt{2}}(p)M(\log n))$$

where $M(\log n)$ is the complexity of multiplication mod n .

- for NFS $O(L_{\frac{1}{3},c}(n))$ (where $c < 2$).
- ECM won't break RSA any time soon.

Complexity of ECM

Let $L_{\alpha,c}(p) = e^{c(\log p)^\alpha (\log \log p)^{1-\alpha}}$. Then the expected time complexity of ECM to find a factor p of n is

$$O(L_{\frac{1}{2},\sqrt{2}}(p)M(\log n))$$

where $M(\log n)$ is the complexity of multiplication mod n .

- for NFS $O(L_{\frac{1}{3},c}(n))$ (where $c < 2$).
- ECM won't break RSA any time soon.
- huge numbers with expected relatively small factors are out of reach for NSF: ECM can be used there.


How to chose x

The hope is to pick P_0 and x such that $x|_{o_{G_p}}(P_0)$.

Use two bounds B_1, B_2

How to chose x

The hope is to pick P_0 and x such that $x|o_{G_p}(P_0)$.

Use two bounds B_1, B_2  “cover” all primes up to B_1 ,

How to chose x

The hope is to pick P_0 and x such that $x | o_{G_p}(P_0)$.

Use two bounds B_1, B_2

- “cover” all primes up to B_1 ,
- permit an additional prime factor in $[B_1, B_2]$.

How to chose x

The hope is to pick P_0 and x such that $x \mid o_{G_p}(P_0)$.

Use two bounds B_1, B_2

- “cover” all primes up to B_1 ,

- permit an additional prime factor in $[B_1, B_2]$.

This explains naturally the two stages method used in GMP-ECM.

High level algorithm description

INPUT: a number n , integer bounds $B_1 \leq B_2$.

OUTPUT: a factor p of n or FAIL.

- 1: Choose a random elliptic curve $E_{a,b} \bmod n$ and a point $P_0 = (x_0 : y_0 : z_0)$ on it.
- 2: Compute $Q = \prod_{\pi \leq B_1} \pi^{\lfloor \log B_1 / \log \pi \rfloor} P_0$.
- 3: **for** π prime, $B_1 < \pi \leq B_2$ **do**
- 4: $(x_\pi : y_\pi : z_\pi) \leftarrow \pi Q$
- 5: $g \leftarrow \gcd(n, z_\pi)$
- 6: **if** $g \neq 1$ **then**
- 7: **return** g
- 8: **end if**
- 9: **end for**
- 10: **return** FAIL

Algorithmic challenges

An implementation of ECM is faced with the following algorithmic challenges:

Algorithmic challenges

An implementation of ECM is faced with the following algorithmic challenges:

- fast modular arithmetic

Algorithmic challenges

An implementation of ECM is faced with the following algorithmic challenges:

- fast modular arithmetic
- efficient curve operations

Algorithmic challenges

An implementation of ECM is faced with the following algorithmic challenges:

- fast modular arithmetic
- efficient curve operations
- fast polynomial evaluation (stage 2)

Algorithmic challenges

An implementation of ECM is faced with the following algorithmic challenges:

- fast modular arithmetic
- efficient curve operations
- fast polynomial evaluation (stage 2)

In order to be fast, we use algorithmic improvements tailored to the size of the numbers used (with thresholds) as well as assembly code.

Modular arithmetic

Operations on the curve translate into residue arithmetic operations (addition, multiplication and division mod n).

Modular arithmetic

Operations on the curve translate into residue arithmetic operations (addition, multiplication and division mod n).

- divisions can be performed by way of multiplications

Modular arithmetic

Operations on the curve translate into residue arithmetic operations (addition, multiplication and division mod n).

- divisions can be performed by way of multiplications
- the most interesting operation to optimize is therefore multiplication.

Modular arithmetic

An example of algorithmic improvement for the operation:

Modular arithmetic

An example of algorithmic improvement for the operation:

$$c \leftarrow a \cdot b \bmod n$$

Modular arithmetic

An example of algorithmic improvement for the operation:

$$c \leftarrow a \cdot b \bmod n$$

The naive approach uses:

- a full $\log n \times \log n \rightarrow 2 \log n$ multiplication,

Modular arithmetic

An example of algorithmic improvement for the operation:

$$c \leftarrow a \cdot b \bmod n$$

The naive approach uses:

- a full $\log n \times \log n \rightarrow 2 \log n$ multiplication,
- a modular reduction $2 \log n$ by $\log n$ (as costly as a division).

Modular arithmetic

Instead we can use Montgomery representation. Let β be the integer base (usually $\beta = 2^{32}$) and l smallest integer such that

$$\beta^l > n$$

Modular arithmetic

Instead we can use Montgomery representation. Let β be the integer base (usually $\beta = 2^{32}$) and l smallest integer such that

$$\beta^l > n$$

The Montgomery representation gives:

$$a \rightarrow a' = \beta^l a \bmod n$$

Modular arithmetic

Instead we can use Montgomery representation. Let β be the integer base (usually $\beta = 2^{32}$) and l smallest integer such that

$$\beta^l > n$$

The Montgomery representation gives:

$$a \rightarrow a' = \beta^l a \bmod n$$

$$b \rightarrow b' = \beta^l b \bmod n$$

Modular arithmetic

Instead we can use Montgomery representation. Let β be the integer base (usually $\beta = 2^{32}$) and l smallest integer such that

$$\beta^l > n$$

The Montgomery representation gives:

$$a \rightarrow a' = \beta^l a \bmod n$$

$$b \rightarrow b' = \beta^l b \bmod n$$

$$a \cdot b \rightarrow \beta^l a \cdot b = (a' b' \beta^{-l}) \bmod n = \text{REDC}(a' b')$$

Algorithm REDC

INPUT: numbers T, β^l, n, n' such that $0 \leq T < \beta^l n$ and $nn' = -1 \pmod{\beta^l}$.

OUTPUT: $T\beta^{-l} \pmod{n}$.

1: $m \leftarrow Tn' \pmod{\beta^l}$

▷ Lower half multiplication

2: $t \leftarrow (T + mn)/\beta^l$

▷ Upper half multiplication

3: **if** $t \geq n$ **then**

4: **return** $t - n$

5: **else**

6: **return** t

7: **end if**

Algorithm REDC

INPUT: numbers T, β^l, n, n' such that $0 \leq T < \beta^l n$ and $nn' = -1 \pmod{\beta^l}$.

OUTPUT: $T\beta^{-l} \pmod{n}$.

1: $m \leftarrow Tn' \pmod{\beta^l}$

▷ Lower half multiplication

2: $t \leftarrow (T + mn)/\beta^l$

▷ Upper half multiplication

3: **if** $t \geq n$ **then**

4: **return** $t - n$

5: **else**

6: **return** t

7: **end if**

• $mn = Tn'n = -T \pmod{\beta^l}$

Algorithm REDC

INPUT: numbers T, β^l, n, n' such that $0 \leq T < \beta^l n$ and $nn' = -1 \pmod{\beta^l}$.

OUTPUT: $T\beta^{-l} \pmod{n}$.

1: $m \leftarrow Tn' \pmod{\beta^l}$

▷ Lower half multiplication

2: $t \leftarrow (T + mn)/\beta^l$

▷ Upper half multiplication

3: **if** $t \geq n$ **then**

4: **return** $t - n$

5: **else**

6: **return** t

7: **end if**

• $mn = Tn'n = -T \pmod{\beta^l}$

• $t\beta^l = T \pmod{n}$ so $t = T\beta^{-l} \pmod{n}$

Algorithm REDC

INPUT: numbers T, β^l, n, n' such that $0 \leq T < \beta^l n$ and $nn' = -1 \pmod{\beta^l}$.

OUTPUT: $T\beta^{-l} \pmod{n}$.

1: $m \leftarrow Tn' \pmod{\beta^l}$

▷ Lower half multiplication

2: $t \leftarrow (T + mn)/\beta^l$

▷ Upper half multiplication

3: **if** $t \geq n$ **then**

4: **return** $t - n$

5: **else**

6: **return** t

7: **end if**

• $mn = Tn'n = -T \pmod{\beta^l}$

• $t\beta^l = T \pmod{n}$ so $t = T\beta^{-l} \pmod{n}$

• $0 \leq T + mn < \beta^l n + \beta^l n$ so $0 \leq t < 2n$.

Fast polynomial evaluation

- Recall stage 1 computed $Q = \prod_{\pi \leq B_1} \pi^{\lfloor \log B_1 / \log \pi \rfloor} P_0$.
- We want to compute $\pi \cdot Q$ for all prime $\pi \in [B_1, B_2]$.
- Split π :

$$\pi = \sigma + \tau$$

$$\pi \cdot Q = \sigma \cdot Q + \tau \cdot Q$$

$$\sigma \cdot Q + \tau \cdot Q = 0 \Rightarrow x_\sigma = x_\tau$$

Fast polynomial evaluation

1. Choose two sets of integers S and T such that

$$S + T \supseteq \{\pi \mid B_1 \leq \pi \leq B_2, \pi \text{ prime}\}$$

$$|S|, |T| = \mathcal{O}(\sqrt{B_2})$$

2. Compute $\{x_\sigma \mid \sigma \cdot Q = (x_\sigma : y_\sigma), \sigma \in S\}$

3. Compute $\{x_\tau \mid \tau \cdot Q = (x_\tau : y_\tau), \sigma \in T\}$

4. Compute $\gcd(x_\sigma - x_\tau, n)$ for all x_σ, x_τ .

Fast polynomial evaluation

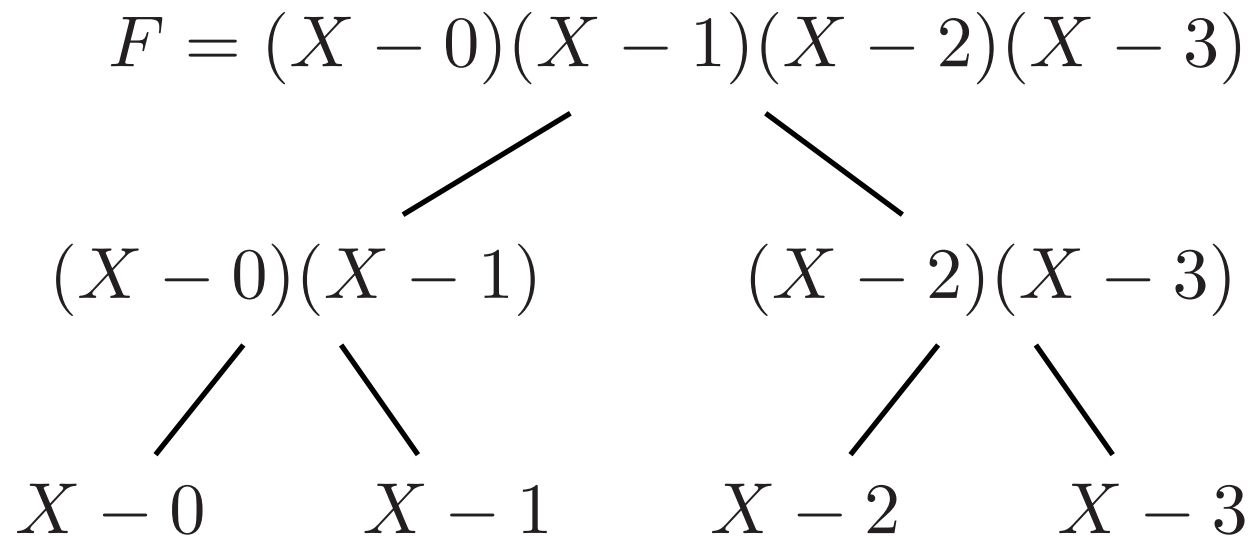
$$h = \prod_{\sigma \in S} \prod_{\tau \in T} (x_{\sigma} - x_{\tau})$$

$$F(X) = \prod_{\tau \in T} (X - x_{\tau})$$

$$G(X) = \prod_{\sigma \in S} (X - x_{\sigma})$$

Compute F at roots of G (multipoint polynomial evaluation in $\mathcal{O}(M(d) \log(d))$ where $d = \deg F$).

Fast polynomial evaluation – Product tree



Fast polynomial evaluation – Remainder tree

$$G(X) = X^4 - 22X^3 + 179X^2 - 638x + 840$$

$$-16X^3 + 168X^2 - 632X + 840 \pmod{(X - 0)(X - 1)(X - 2)(X - 3)}$$

Fast polynomial evaluation – Remainder tree

$$G(X) = X^4 - 22X^3 + 179X^2 - 638x + 840$$

$$-16X^3 + 168X^2 - 632X + 840 \pmod{(X - 0)(X - 1)(X - 2)(X - 3)}$$


$$-480X + 840 \pmod{(X - 0)(X - 1)}$$

Fast polynomial evaluation – Remainder tree

$$G(X) = X^4 - 22X^3 + 179X^2 - 638x + 840$$

$$-16X^3 + 168X^2 - 632X + 840 \pmod{(X-0)(X-1)(X-2)(X-3)}$$

$$-480X + 840 \pmod{(X-0)(X-1)}$$

$$\underbrace{840}_{G(0)} \pmod{X-0}$$

Fast polynomial evaluation – Remainder tree

$$G(X) = X^4 - 22X^3 + 179X^2 - 638x + 840$$

$$-16X^3 + 168X^2 - 632X + 840 \pmod{(X-0)(X-1)(X-2)(X-3)}$$

$$-480X + 840 \pmod{(X-0)(X-1)}$$

$$\underbrace{840}_{G(0)} \pmod{X-0}$$

$$\underbrace{360}_{G(1)} \pmod{X-1}$$

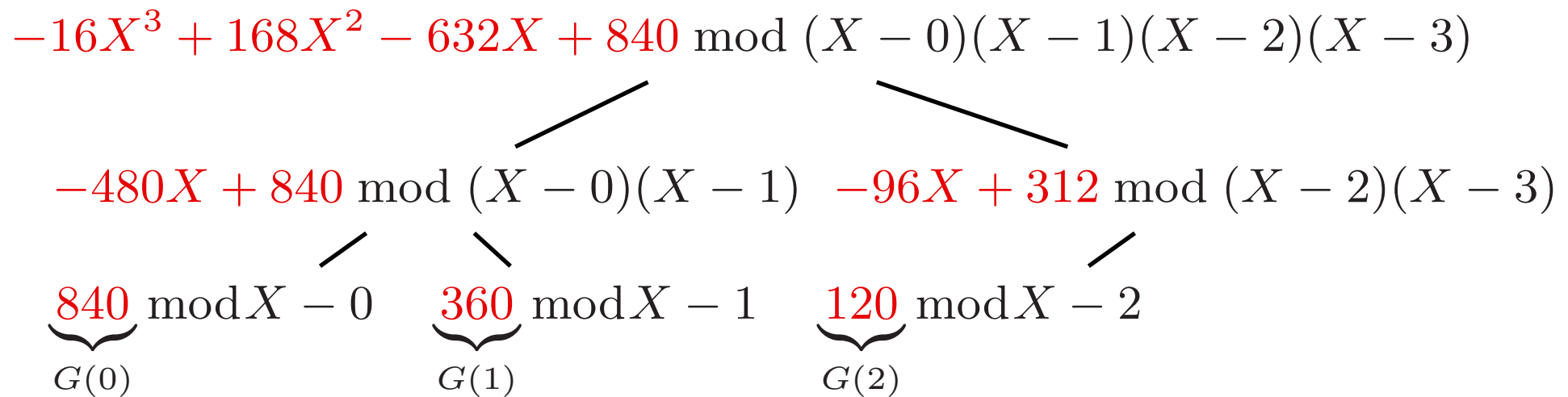
Fast polynomial evaluation – Remainder tree

$$G(X) = X^4 - 22X^3 + 179X^2 - 638x + 840$$

$$\begin{array}{c} -16X^3 + 168X^2 - 632X + 840 \pmod{(X-0)(X-1)(X-2)(X-3)} \\ \swarrow \quad \searrow \\ -480X + 840 \pmod{(X-0)(X-1)} \quad -96X + 312 \pmod{(X-2)(X-3)} \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ \underbrace{840}_{G(0)} \pmod{X-0} \quad \underbrace{360}_{G(1)} \pmod{X-1} \end{array}$$

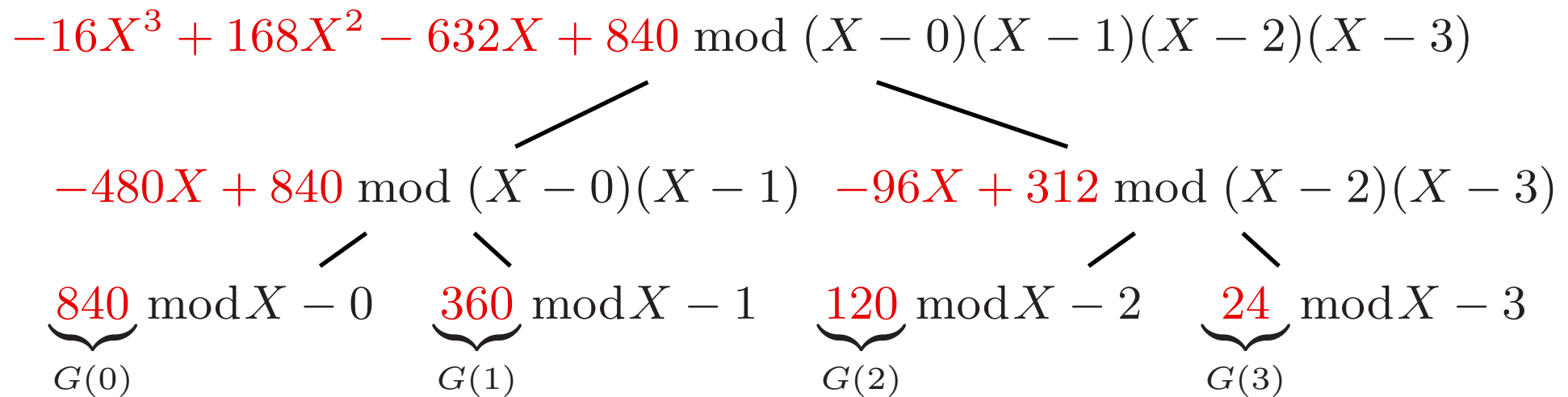
Fast polynomial evaluation – Remainder tree

$$G(X) = X^4 - 22X^3 + 179X^2 - 638x + 840$$



Fast polynomial evaluation – Remainder tree

$$G(X) = X^4 - 22X^3 + 179X^2 - 638x + 840$$



Fast polynomial evaluation – Scaled Remainder t

$$U = \frac{G}{F}$$

$$U = -16X^{-1} + 72X^{-2} - 24X^{-3} - 192X^{-4} + \dots \text{ mod } 1$$

$$U(X-0)(X-1) = -96X^{-1} - 168X^{-2} + \dots \text{ mod } 1$$

$$U(X-2)(X-3) = -480X^{-1} + 360X^{-2} + \dots \text{ mod } 1$$

$$U(X-0)(X-1)(X-2) = 24X^{-1} + \dots \text{ mod } 1$$

$$U(X-0)(X-1)(X-3) = 120X^{-1} + \dots \text{ mod } 1$$

$$U(X-2)(X-3)(X-0) = 360X^{-1} + \dots \text{ mod } 1$$


$$U(X-2)(X-3)(X-1) = 840X^{-1} + \dots \text{ mod } 1$$

Fast polynomial evaluation – Scaled Remainder t

$$-16X^{-1} + 72X^{-2} - 24X^{-3} - 192X^{-4}$$

Fast polynomial evaluation – Scaled Remainder t

$$-16X^{-1} + 72X^{-2} - 24X^{-3} - 192X^{-4}$$


$$-480X^{-1} + 360X^{-2}$$

Fast polynomial evaluation – Scaled Remainder t

$$-16X^{-1} + 72X^{-2} - 24X^{-3} - 192X^{-4}$$

$$-480X^{-1} + 360X^{-2}$$

$$\underbrace{840}_{G(0)} X^{-1}$$

Fast polynomial evaluation – Scaled Remainder t

$$-16X^{-1} + 72X^{-2} - 24X^{-3} - 192X^{-4}$$

$$-480X^{-1} + 360X^{-2}$$

$$\underbrace{840}_{G(0)} X^{-1}$$

$$\underbrace{360}_{G(1)} X^{-1}$$

Fast polynomial evaluation – Scaled Remainder t

$$\begin{array}{r} -16X^{-1} + 72X^{-2} - 24X^{-3} - 192X^{-4} \\ \swarrow \quad \searrow \\ -480X^{-1} + 360X^{-2} \qquad -96X^{-1} - 168X^{-2} \\ \swarrow \quad \searrow \\ \underbrace{840}_{G(0)} X^{-1} \qquad \underbrace{360}_{G(1)} X^{-1} \end{array}$$

Fast polynomial evaluation – Scaled Remainder t

$$\begin{array}{c} -16X^{-1} + 72X^{-2} - 24X^{-3} - 192X^{-4} \\ \swarrow \quad \searrow \\ -480X^{-1} + 360X^{-2} \quad -96X^{-1} - 168X^{-2} \\ \swarrow \quad \searrow \quad \swarrow \\ \underbrace{840}_{G(0)} X^{-1} \quad \underbrace{360}_{G(1)} X^{-1} \quad \underbrace{120}_{G(2)} X^{-1} \end{array}$$

Fast polynomial evaluation – Scaled Remainder t

$$\begin{array}{ccccccc} & & -16X^{-1} + 72X^{-2} - 24X^{-3} - 192X^{-4} & & & & \\ & & \swarrow & & \searrow & & \\ -480X^{-1} + 360X^{-2} & & & & & & -96X^{-1} - 168X^{-2} \\ \swarrow & & \searrow & & \swarrow & & \searrow \\ \underbrace{840}_{G(0)} X^{-1} & & \underbrace{360}_{G(1)} X^{-1} & & \underbrace{120}_{G(2)} X^{-1} & & \underbrace{24}_{G(3)} X^{-1} \end{array}$$

An example ECM factorization with GMP-ECM

Let

$$\begin{aligned} N = & 14421499473850156964908748714733530694523632245 \\ & 48199639497786788488662355550896284406444518825 \\ & 58206114371958127980496626323628808250533258921 \\ & 4668868374102124472638792350353190035972075401 \end{aligned}$$

of 187 digits. We chose $B_1 = 433993$, the curve parameterized by $\sigma = 550048451$ (Suyama).

An example ECM factorization with GMP-ECM

```
$ ecm -v -v -sigma 550048451 433993 < composite
GMP-ECM 6.1.1 [powered by GMP 4.2.1] [ECM]
Input number is [...] (187 digits)
Using MODMULN
Using B1=433993, B2=347971482, polynomial Dickson(3), sigma=550048451
Expected number of curves to find a factor of n digits:
20      25      30      35      40      45      50      55      60      65
6       33      241     2322    27856   401319  6767926  1.3e+08  2.9e+09  7.1e+10
Step 1 took 39656ms
Step 2 took 12204ms
***** Factor found in step 2: 344518986834068356794510012742065462371
Found probable prime factor of 39 digits: 344518986834068356794510012742065462371
Composite cofactor 41...1 has 148 digits
```

Conclusion

- ECM has seen numerous improvements, both mathematical and in the implementations since its discovery by H. W. Lenstra, Jr in 1985.

Conclusion

- ECM has seen numerous improvements, both mathematical and in the implementations since its discovery by H. W. Lenstra, Jr in 1985.
- easy to adapt to distributed computing (each computer runs a curve)

Conclusion

- ECM has seen numerous improvements, both mathematical and in the implementations since its discovery by H. W. Lenstra, Jr in 1985.
- easy to adapt to distributed computing (each computer runs a curve)
- packaged in Debian



Conclusion

- ECM has seen numerous improvements, both mathematical and in the implementations since its discovery by H. W. Lenstra, Jr in 1985.
- easy to adapt to distributed computing (each computer runs a curve)



- packaged in Debian
- why not try it today?

`http://gforge.inria.fr/projects/ecm/`