Public Key Cryptanalysis

Discrete Logarithms and Factorization

Christophe Petit

University of Oxford

Discrete logarithms

- Given a cyclic group $(G, \circ)$ (written multiplicatively), a generator g of G and a second element $h \in G$, compute $k \in \mathbb{Z}_{|G|}$ such that $g^k = h$

Discrete logarithms

- Given a cyclic group $(G, \circ)$ (written multiplicatively), a generator g of G and a second element $h \in G$, compute $k \in \mathbb{Z}_{|G|}$ such that $g^k = h$
- Trivial if $(G, \circ) = (\mathbb{F}_p, +)$. Why ?
- Recently broken if $(G, \circ) = (\mathbb{F}_{2^n}^*, *)$ (more generally if characteristic is small)

Discrete logarithms

- Given a cyclic group $(G, \circ)$ (written multiplicatively), a generator g of G and a second element $h \in G$, compute $k \in \mathbb{Z}_{|G|}$ such that $g^k = h$
- Trivial if $(G, \circ) = (\mathbb{F}_p, +)$. Why ?
- Recently broken if $(G, \circ) = (\mathbb{F}_{2^n}^*, *)$ (more generally if characteristic is small)
- Believed to be hard (to different extents) for $G = \mathbb{F}_p^*$ and for (well-chosen) elliptic/hyperelliptic curve groups

Integer factorization

- Given a composite number n, compute its (unique) factorization $n = \prod p_i^{e_i}$ where p_i are prime numbers

Integer factorization

- Given a composite number n, compute its (unique) factorization $n = \prod p_i^{e_i}$ where p_i are prime numbers
- Equivalently (why ?) : compute one non-trivial factor p_i
- Trivial if $n = p^e$
- Believed to be hard if $n = pq$ for well-chosen $p \neq q$

RSA and Diffie-Hellman

- DLP broken implies Diffie-Hellman broken
- Factorization broken implies RSA broken

RSA and Diffie-Hellman

- DLP broken implies Diffie-Hellman broken
- Factorization broken implies RSA broken
- We don't know whether DH broken implies DLP broken
- We don't know whether RSA broken implies factorization broken

RSA and Diffie-Hellman

- DLP broken implies Diffie-Hellman broken
- Factorization broken implies RSA broken
- We don't know whether DH broken implies DLP broken
- We don't know whether RSA broken implies factorization broken
- Nevertheless, the best attacks against DH and RSA today are discrete log and factorization attacks

Outline

Generic discrete logarithm algorithms

Discrete logarithms over finite fields

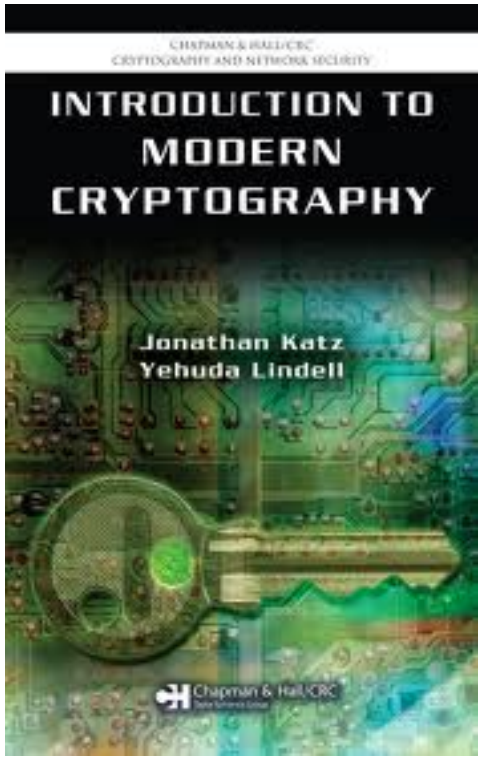
Elliptic curve discrete logarithms

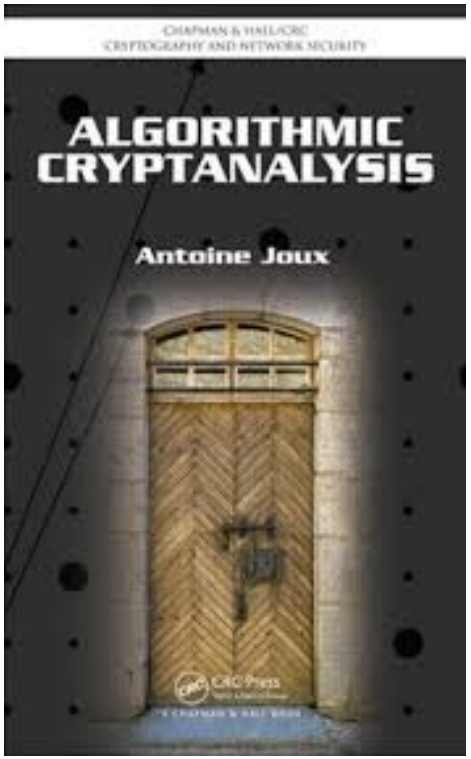
Factorization algorithms

Some side-channel attacks

Lab and tutorial content

References





- Introduction to Modern Cryptography, Chapter 8
- Algorithmic Cryptanalysis, Chapter 15

Outline

Generic discrete logarithm algorithms

Discrete logarithms over finite fields

Elliptic curve discrete logarithms

Factorization algorithms

Some side-channel attacks

Lab and tutorial content

Generic attacks

- DLP is trivial in some groups
- DLP seems harder in other groups
- Best attacks in a particular group often rely on specific properties of the group

Generic attacks

- DLP is trivial in some groups
- DLP seems harder in other groups
- Best attacks in a particular group often rely on specific properties of the group
- Can we find better groups ?
- How hard can DLP be in the best (hardest) groups ?

Group isomorphisms

- Any cyclic group $(G, \circ)$ of order n can be seen as $(\mathbb{Z}_n, +)$ in the following sense : there exists an invertible map $\varphi : G \to \mathbb{Z}_n$ such that $\forall x, y \in G$, we have

$$\varphi(x \circ y) = \varphi(x) + \varphi(y)$$

- Remark φ does not need to be efficiently computable
- Example : let g of order $p-1$ in $\mathbb{Z}_p^*$. Can define φ as sending any $h \in G$ to $\varphi(h) \in \mathbb{Z}_{p-1}$ such that $h = g^{\varphi(h)}$.
- Let $x' = \varphi(x)$ and $y' = \varphi(y)$. We have

$$\varphi^{-1}(x'+y') = \varphi^{-1}(\varphi(x)+\varphi(y)) = \varphi^{-1}(\varphi(x \circ y)) = x \circ y = \varphi^{-1}(x') \circ \varphi^{-1}(y')$$

DLP in the generic group model

- A DLP instance is generated in $(\mathbb{Z}_n, +)$, including a generator $g \in \mathbb{Z}_n$ and another element $h = kg \in \mathbb{Z}_n$
- A random invertible map $\theta : \mathbb{Z}_n \to \mathbb{Z}_n$ is chosen
- The map defines a group $(\mathbb{Z}_n, \circ)$ with

$$x \circ y = \theta\left(\theta^{-1}(x) + \theta^{-1}(y)\right)$$

- The attacker is NOT given g, h nor θ
- The attacker is given $\theta(g)$, $\theta(h)$ and access to **oracles**
  - $\mathcal{O}_1$: on input x, y, return $\theta\left(\theta^{-1}(x) + \theta^{-1}(y)\right)$
  - $\mathcal{O}_2$: on input x, return $\theta(-\theta^{-1}(x))$
- The attacker's goal is to compute k

Generic group model

- As θ is random, there is no special property of the group that can be exploited
- n itself is sometimes hidden, and the attacker just receives bitstrings instead of $\mathbb{Z}_n$ elements (the size of n cannot be hidden)
- Some attacks are generic : they work for any group
  This includes exhaustive search, BSGS, Pollard's rho
- There exist much better attacks for finite fields
- Still no better attack for (well-chosen) elliptic curves

Exhaustive search

- Given $g, h \in G$ do the following

1: $k \leftarrow 1; h' \leftarrow g$
2: **if** $h' = h$ **then**
3: **return** k
4: **else**
5: $k \leftarrow k + 1\,;\; h' \leftarrow h'g$
6: Go to Step 2
7: **end if**

- Generic algorithm
- Time complexity $|G|$ in the worst case, $|G|/2$ on average
- Can we do better ?

Baby step, giant step (BSGS)

- Let $h = g^k$. You want to compute k.

Baby step, giant step (BSGS)

- Let $h = g^k$. You want to compute k.
- Let $N' = \lceil \sqrt{|G|} \rceil$
- There exist $0 \leq i, j < N'$ such that $k = jN' + i$

Baby step, giant step (BSGS)

- Let $h = g^k$. You want to compute k.
- Let $N' = \lceil \sqrt{|G|} \rceil$
- There exist $0 \leq i, j < N'$ such that $k = jN' + i$

$$h = g^{jN'+i} \Leftrightarrow hg^{-jN'} = g^i$$

Baby step, giant step (BSGS)

- Let $h = g^k$. You want to compute k.
- Let $N' = \lceil \sqrt{|G|} \rceil$
- There exist $0 \leq i, j < N'$ such that $k = jN' + i$

$$h = g^{jN'+i} \Leftrightarrow hg^{-jN'} = g^i$$

- Compute $L_B := \{g^i | i = 0, \ldots, N' - 1\}$
- Compute $L_G := \{hg^{-jN'} | j = 0, \ldots, N' - 1\}$

Baby step, giant step (BSGS)

- Let $h = g^k$. You want to compute k.
- Let $N' = \lceil \sqrt{|G|} \rceil$
- There exist $0 \leq i, j < N'$ such that $k = jN' + i$

$$h = g^{jN'+i} \Leftrightarrow hg^{-jN'} = g^i$$

- Compute $L_B := \{g^i | i = 0, \ldots, N' - 1\}$
- Compute $L_G := \{hg^{-jN'} | j = 0, \ldots, N' - 1\}$
- Attack requires time and memory $O(\sqrt{|G|})$

Birthday paradox

- Suppose there are N_2 people in a room. What is the probability that two people have the same birthday ?
- How many people needed to have a probability larger than 50% ?

Birthday paradox

- Suppose there are N_2 people in a room. What is the probability that two people have the same birthday ?
- How many people needed to have a probability larger than 50% ?
- Answer is **23** :

$$\Pr[\text{all distinct}] = 1 \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \ldots \cdot \frac{365-22}{365} < \frac{1}{2}$$

Birthday paradox

- Suppose you choose N_2 elements randomly in a set of N elements. What is the probability that two elements are equal ?
- How should N_2 be wrt N to have a probability larger than 50% ?

Birthday paradox

- Suppose you choose N_2 elements randomly in a set of N elements. What is the probability that two elements are equal ?
- How should N_2 be wrt N to have a probability larger than 50% ?
- Answer is $O(\sqrt{N})$:

$$\begin{aligned}
\Pr[\text{all distinct}] &= 1 \cdot \frac{N-1}{N} \cdot \frac{N-2}{N} \cdot \ldots \cdot \frac{N-N_2+1}{N} \\
&\approx e^{-\frac{1}{N}} \cdot e^{-\frac{2}{N}} \cdot \ldots \cdot e^{-\frac{N_2-1}{N}} \\
&\approx e^{-\frac{N_2(N_2-1)}{N}}
\end{aligned}$$

Taking $N_2 \approx \sqrt{N}$ ensures $1 - \Pr[\text{all distinct}]$ constant

Pollard's rho (iterative function)

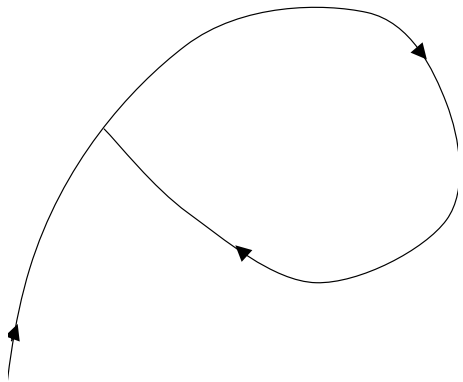
- Define G_1, G_2, G_3 of about the same size such that $G = G_1 \cup G_2 \cup G_3$ and $G_i \cap G_j = \{\}$
- Over $\mathbb{Z}_p^*$, can choose
  $G_1 = \{0, \ldots, \lfloor p/3 \rfloor\}$,
  $G_2 = \{\lfloor p/3 \rfloor + 1, \ldots, \lfloor 2p/3 \rfloor\}$,
  $G_3 = \{\lfloor 2p/3 \rfloor + 1, \ldots, p-2\}$
- Define a function $f : G \rightarrow G$ such that

$$\begin{cases} f(z) = zg & z \in G_1 \\ f(z) = z^2 & z \in G_2 \\ f(z) = zh & z \in G_3 \end{cases}$$

(original definition, other definitions possible)

Pollard's rho (intuition)

- Start from $g_0 := g$ and apply f recursively to get g_i
- By the way f is defined, we can keep track of a_i, b_i such that $g_i = g^{a_i} h^{b_i}$
- If f is "random enough", obtain random elements in G and a collision after $O(\sqrt{|G|})$ elements
- Collision gives DLP solution



Pollard's rho (simplest version)

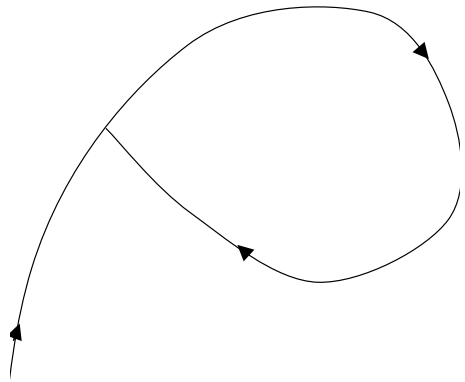
$N \leftarrow \lceil \sqrt{|G|} \rceil$
$a \leftarrow 1; b \leftarrow 0; \tilde{h} \leftarrow g; L \leftarrow \{(a, b, \tilde{h})\}$
for $k \in \{2, \ldots, N\}$ **do**
    if $\tilde{h} \in G_1$ **then** $a \leftarrow a + 1; \tilde{h} \leftarrow \tilde{h}g$
    if $\tilde{h} \in G_2$ **then** $a \leftarrow 2a; b \leftarrow 2b; \tilde{h} \leftarrow (\tilde{h})^2$
    if $\tilde{h} \in G_3$ **then** $b \leftarrow b + 1; \tilde{h} \leftarrow \tilde{h}h$
    $L \leftarrow L \cup \{(a, b, \tilde{h})\}$
end for
Find distinct $(a_i, b_i, \tilde{h}) \in L$, $i = 1, 2$
if no such elements **then abort**
return $-(a_1 - a_2)/(b_1 - b_2) \bmod |G|$

Pollard's rho analysis

- Correctness :
  - Every $(a, b, \tilde{h})$ in the list satisfies $\tilde{h} = g^a h^b$
  - $g^{a_1} h^{b_1} = g^{a_2} h^{b_2}$ implies $h = g^{-\frac{a_1 - a_2}{b_1 - b_2}}$
- Time and memory costs $N \approx \sqrt{|G|}$
- Good probability of success by birthday's paradox

Pollard's rho (improvement)

- Let $(L_1, L_1 + L_2)$ be the indices of first collision
- Then $(L_1 + j, L_1 + kL_2 + j)$ also collide
- For j, k such that $L_1 + j = kL_2$, we have $L_1 + kL_2 + j = 2(L_1 + j)$



- Now search for $(a_i, b_i, \tilde{h}_i)$ and $(a_{2i}, b_{2i}, \tilde{h}_{2i})$ such that $\tilde{h}_i = \tilde{h}_{2i}$
- Only requires constant size memory

Pohlig-Hellman

- Assume $|G| = n_1 n_2$ and let g a generator of G
- $h = g^k$ implies $h^{n_1} = (g^{n_1})^k$
  where g^{n_1} generates a subgroup of order n_2
- Solving DLP in that subgroup gives k mod n_2
- Repeating for each factor and using CRT gives k

Pohlig-Hellman (example)

- Let $G = \mathbb{Z}_{13}^*$, let $g = 2$ and let $h = 7$
- We have $|G| = 12 = 2^2 \cdot 3$
- Recover k mod 2 by solving $(2^6)^k = 7^6 \bmod 13 \Leftrightarrow (-1)^k = -1 \bmod 13 \Leftrightarrow k = 1 \bmod 2$
- Write $k = 1 + 2k'$. Recover k mod 4 by solving $(2^3)^{1+2k'} = 7^3 \bmod 13 \Leftrightarrow (-1)^{k'} = -1 \bmod 13 \Leftrightarrow k' = 1 \bmod 2 \Leftrightarrow k = 3 \bmod 4$
- Recover k mod 3 by solving $(2^4)^k = 7^4 \bmod 13 \Leftrightarrow (3)^k = 9 \bmod 13 \Leftrightarrow k = 2 \bmod 3$
- Use CRT to deduce $k = 11 \bmod 12$

Outline

Generic discrete logarithm algorithms

Discrete logarithms over finite fields

Elliptic curve discrete logarithms

Factorization algorithms

Some side-channel attacks

Lab and tutorial content

Prime fields

- $(\mathbb{Z}_p, +, *)$ is a field for any prime p
- This field is often denoted $\mathbb{F}_p$

Extension fields

- Let f be a polynomial of degree n with coefficients in $\mathbb{F}_p$, such that f has no factor of degree different than 0 or n
- Consider $(K, +, *)$ where
  - $K = \{$all polynomials of degree at most n over $\mathbb{F}_p$ $\}$
  - $+$ and $*$ are addition and multiplication **modulo the polynomial** f
- Then $(K, +, *)$ is a finite field with p^n elements
- Example : let $f(x) = x^2 + x + 1 \in \mathbb{F}_2[x]$ then $\mathbb{F}_4 = \mathbb{F}_2[x]/(f(x)\mathbb{F}_2[x])$ is a finite field with 4 elements $\{0, 1, x, x+1\}$

Extension fields

- Let f be a polynomial of degree n with coefficients in $\mathbb{F}_p$, such that f has no factor of degree different than 0 or n
- Consider $(K, +, *)$ where
  - $K = \{$all polynomials of degree at most n over $\mathbb{F}_p$ $\}$
  - $+$ and $*$ are addition and multiplication **modulo the polynomial** f
- Then $(K, +, *)$ is a finite field with p^n elements
- Example : let $f(x) = x^2 + x + 1 \in \mathbb{F}_2[x]$ then $\mathbb{F}_4 = \mathbb{F}_2[x]/(f(x)\mathbb{F}_2[x])$ is a finite field with 4 elements $\{0, 1, x, x+1\}$
- Theorem : any finite field can be constructed this way

DLP over finite fields

- In fact, DLP over the multiplicative group of finite fields (DLP over the additive group is easy)
- DLP : given p, n, given g a generator of $\mathbb{F}_{p^n}^*$, and given $h = g^k$, compute k

Fields used in cryptography

- $\mathbb{F}_p^*$ where p is prime : most used, believed to be secure
- $\mathbb{F}_{p^n}^*$ where p is prime and n is small (typically up to 12) : used in *pairing* applications
- $\mathbb{F}_{2^n}^*$ or $\mathbb{F}_{3^n}^*$ where n is a product of small primes : should be avoided (Pohlig-Hellman attack)
- $\mathbb{F}_{2^n}^*$ or $\mathbb{F}_{3^n}^*$ for arbitrary n : should now also be avoided, suggested before 2013 for efficiency reasons

Fields used in cryptography

- $\mathbb{F}_p^*$ where p is prime : most used, believed to be secure
- $\mathbb{F}_{p^n}^*$ where p is prime and n is small (typically up to 12) : used in *pairing* applications
- $\mathbb{F}_{2^n}^*$ or $\mathbb{F}_{3^n}^*$ where n is a product of small primes : should be avoided (Pohlig-Hellman attack)
- $\mathbb{F}_{2^n}^*$ or $\mathbb{F}_{3^n}^*$ for arbitrary n : should now also be avoided, suggested before 2013 for efficiency reasons
- Remark : typically work over a prime order subgroup of $\mathbb{F}_p^*$ or $\mathbb{F}_{p^n}^*$, otherwise problems such as *decisional Diffie-Hellman* are easy

L notation

$$L_Q(\alpha; c) = \exp(c(\log Q)^{\alpha}(\log\log Q)^{1-\alpha})$$

- Q is the size of the field
- $\alpha = 0 \Rightarrow L_Q(\alpha; c) = (\log Q)^c$ polynomial
- $\alpha = 1 \Rightarrow L_Q(\alpha; c) = Q^c$ exponential
- The constant c has a practical impact

Some history

- See Joux, Odlyzko, Pierrot. *The past, evolving present and future of discrete logarithms*
  `http://www-polsys.lip6.fr/~pierrot/papers/Dlog.pdf`

Index calculus

- Generic framework to solve discrete logarithm problems, but some steps are group-specific
- Let g, h a DLP problem

Index calculus

- Generic framework to solve discrete logarithm problems, but some steps are group-specific
- Let g, h a DLP problem
- Define a *factor basis* $\mathcal{F} \subset G$, ensuring $\mathcal{F}$ contains a generator (most elements in G are generators)
- Can assume $g \in \mathcal{F}$, otherwise do the following :
  - Pick a generator $g' \in \mathcal{F}$
  - Compute a such that $g = (g')^a$
  - Compute b such that $h = (g')^b$
  - Compute $k = b/a \bmod |G|$
- Remark : size of $\mathcal{F}$ will be optimized for efficiency

Index calculus

- Find about $|\mathcal{F}|$ relations between factor basis elements

$$\mathcal{R}_j : \quad \prod_{f_i \in \mathcal{F}} f_i^{a_{i,j}} = 1$$

(the algorithm to compute the relations is group-specific)

- Deduce

$$\sum_{f_i \in \mathcal{F}} a_{i,j} \log_g f_i = 0$$

or

$$\begin{pmatrix} a_{1,1} & \dots & a_{|\mathcal{F}|,1} \\ \vdots & & \vdots \\ a_{1,|\mathcal{F}|} & \dots & a_{|\mathcal{F}|,|\mathcal{F}|} \end{pmatrix} \begin{pmatrix} \log_g f_1 \\ \vdots \\ \log_g f_{|\mathcal{F}|} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

Index calculus

- Use linear algebra to compute all $\log_g f_i$,
  the discrete logarithms of factor basis elements
- Deduce the discrete logarithm of h
  (This part is group-specific and may involve several steps)
- Remarks :
  - Relations often involve few elements,
    hence linear algebra is sparse
  - In some cases, h is included in the factor basis and the last step is avoided : linear algebra produces $\log_g h$

Example : a naive index calculus for $\mathbb{F}_p^*$

- DLP : given $g, h \in \mathbb{F}_p^*$, find k such that $h = g^k$
- Factor basis made of **small primes**

$$\mathcal{F}_B := \{\text{primes } p_i \leq B\}$$

- **Relation search**
  - Compute $r_j := g^{a_j} h^{b_j}$ for random $a_j, b_j \in \{1, \ldots, p-1\}$
  - **If** all factors of r_j are $\leq B$, we have a relation

$$g^{a_j} h^{b_j} = \prod_{p_i \in \mathcal{F}} p_i^{e_{i,j}}$$

- **Linear algebra** produces $g^a h^b = 1$

Size of the factor basis

- By the prime number theorem,

$$|\{\text{primes } p_i \leq B\}| \approx \frac{B}{\ln B}$$

Smooth numbers

- A number is B-smooth if all its prime factors are smaller than B
- Define $\Psi(N, B) = \#\{B\text{-smooth numbers} \leq N\}$

Smooth numbers

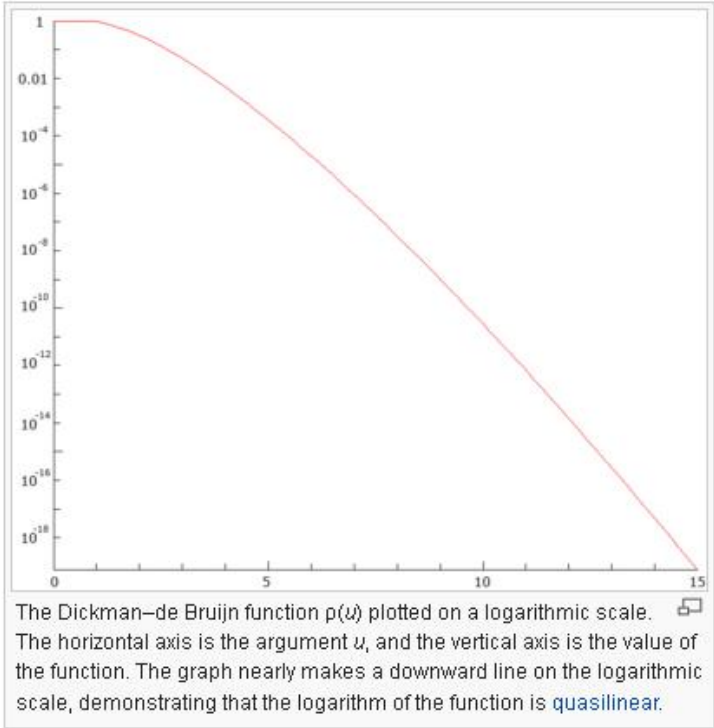
- A number is B-smooth if all its prime factors are smaller than B
- Define $\Psi(N, B) = \#\{B\text{-smooth numbers} \leq N\}$
- Let $u = \log N / \log B$. We have

$$\Psi(N, B) = N\rho(u) + O\left(\frac{N}{\log B}\right)$$

- The proportion of smooth numbers is roughly a function ρ of $u = \log N / \log B$,
- The *Dickman-de Bruijn* function ρ satisfies $\rho(u) \approx u^{-u}$

Dickman-de Bruijn function ρ

- The *Dickman-de Bruijn* function ρ satisfies $\rho(u) \approx u^{-u}$



The Dickman–de Bruijn function ρ(*u*) plotted on a logarithmic scale. The horizontal axis is the argument *u*, and the vertical axis is the value of the function. The graph nearly makes a downward line on the logarithmic scale, demonstrating that the logarithm of the function is quasilinear.

$\log \rho \approx -u \log u$

(picture source : Wikipedia)

Naive analysis of naive index calculus

- Choose $\log B \approx (\log p)^{1/2}$
- $|\mathcal{F}| \approx B/\log B \approx 2^{(\log p)^{1/2}-(\log\log p)^{-1/2}} \approx 2^{(\log p)^{1/2}}$
- $u = \log p/\log B \approx (\log p)^{1/2}$
- $\rho(u) = (\log p)^{-1/2(\log p)^{1/2}} \approx 2^{-1/2(\log p)^{1/2}(\log\log p)}$
- Number of random trials to get $|\mathcal{F}|$ relations is

$$\approx |\mathcal{F}|\rho(u)^{-1} \approx 2^{(1/2+o(1))(\log p)^{1/2}(\log\log p)}$$

- Each trial has polytime complexity in $\log p$
- Linear algebra cost is $|\mathcal{F}|^{\omega} \approx 2^{\omega(\log p)^{1/2}}$
- Total cost dominated by relation search

Naive analysis of naive index calculus

- Choose $\log B \approx (\log p)^{1/2}$
- $|\mathcal{F}| \approx B/\log B \approx 2^{(\log p)^{1/2}-(\log\log p)^{-1/2}} \approx 2^{(\log p)^{1/2}}$
- $u = \log p/\log B \approx (\log p)^{1/2}$
- $\rho(u) = (\log p)^{-1/2(\log p)^{1/2}} \approx 2^{-1/2(\log p)^{1/2}(\log\log p)}$
- Number of random trials to get $|\mathcal{F}|$ relations is

$$\approx |\mathcal{F}|\rho(u)^{-1} \approx 2^{(1/2+o(1))(\log p)^{1/2}(\log\log p)}$$

- Each trial has polytime complexity in $\log p$
- Linear algebra cost is $|\mathcal{F}|^{\omega} \approx 2^{\omega(\log p)^{1/2}}$
- Total cost dominated by relation search
- $B \approx L_p(1/2; c)$ leads to slighly better cost $L_p(1/2; c')$

Same algorithm for $\mathbb{F}_{2^n}^*$

- DLP : given $g, h \in \mathbb{F}_{2^n}^*$, find k such that $h = g^k$
- Factor basis made of **small "primes"**

$$\mathcal{F}_B := \{\text{irreducible } f(X) \in \mathbb{F}_2[X] | \deg(f) \leq B\}$$

Same algorithm for $\mathbb{F}_{2^n}^*$

- DLP : given $g, h \in \mathbb{F}_{2^n}^*$, find k such that $h = g^k$
- Factor basis made of **small "primes"**

$$\mathcal{F}_B := \{\text{irreducible } f(X) \in \mathbb{F}_2[X] | \deg(f) \leq B\}$$

- **Relation search**
  - Compute $r_j := g^{a_j} h^{b_j}$ for random $a_j, b_j \in \{1, \ldots, p-1\}$
  - Factor $r_j \in \mathbb{F}_2[X]$ with Berlekamp's algorithm
  - If all factors $\in \mathcal{F}_B$, we have a relation $g^a h^b = \prod_{f_i \in \mathcal{F}} f_i^{e_i}$
- **Linear algebra** produces $g^a h^b = 1$

Coppersmith's algorithm for $\mathbb{F}_{2^n}$

- Idea : reduce factor basis to polynomials of degree $n^{1/3}$ (vs. $n^{1/2}$) by ensuring all r_j have degree $n^{2/3}$ (vs. n)

Coppersmith's algorithm for $\mathbb{F}_{2^n}$

- Idea : reduce factor basis to polynomials of degree $n^{1/3}$ (vs. $n^{1/2}$) by ensuring all r_j have degree $n^{2/3}$ (vs. n)
- Remember $\mathbb{F}_{2^n} \approx \mathbb{F}_2[x]/(p(x))$ for any irreducible p
  Choose $p(x) = x^n + q(x)$ where $\deg q \leq n^{2/3}$
- Remember squaring is linear : $(a+b)^2 = a^2 + b^2$

Coppersmith's algorithm for $\mathbb{F}_{2^n}$

- Idea : reduce factor basis to polynomials of degree $n^{1/3}$ (vs. $n^{1/2}$) by ensuring all r_j have degree $n^{2/3}$ (vs. n)
- Remember $\mathbb{F}_{2^n} \approx \mathbb{F}_2[x]/(p(x))$ for any irreducible p
  Choose $p(x) = x^n + q(x)$ where $\deg q \leq n^{2/3}$
- Remember squaring is linear : $(a+b)^2 = a^2 + b^2$
- Let $k = 2^e \approx n^{1/3}$, let $d \approx n^{1/3}$
- Let $h \approx n^{2/3}$ least integer larger than n/k
- Let $r(x) = x^{hk} \bmod p(x) = q(x)x^{hk-n}$
  with $\deg r < k + \deg q \approx n^{2/3}$

Coppersmith's algorithm for $\mathbb{F}_{2^n}$

- Factor basis are elements with degree smaller than d, where d smallest integer $\geq n^{1/3}$
- Relations will be of the form $d(x) = (c(x))^k$ for c, d smooth, where c constructed in a special way
- Relation search
  - Take $a(x)$ and $b(x)$ coprime with degrees d
  - Take $c(x) = a(x)x^h + b(x)$ degree $O(n^{2/3})$
  - Take $d(x) = (c(x))^k \bmod p$
  - We have $d(x) = r(x)(a(x))^k + (b(x))^k$ degree $O(n^{2/3})$
  - If both c and d are smooth, we get a relation
  - Probability $O(2^{-n^{1/3}-\epsilon})$

Coppersmith's algorithm for $\mathbb{F}_{2^n}$

- Individual logarithms for polynomials of degrees $<< n$
  - Let $m(x)$ a polynomial with degree $<< n$
  - Choose $a(x)$ and $b(x)$ coprime random such that $m(x)|c(x) = a(x)x^h + b(x)$
  - Let $d(x) = (c(x))^k \bmod p(x)$ as above
  - If d and c/m smooth, we can write m in the factor basis

Coppersmith's algorithm for $\mathbb{F}_{2^n}$

- Individual logarithms for polynomials of degrees $<< n$
  - Let $m(x)$ a polynomial with degree $<< n$
  - Choose $a(x)$ and $b(x)$ coprime random such that $m(x)|c(x) = a(x)x^h + b(x)$
  - Let $d(x) = (c(x))^k \bmod p(x)$ as above
  - If d and c/m smooth, we can write m in the factor basis
- Individual logarithms
  - Involve several steps to write m as a product of smaller and smaller factors

Function field sieve and beyond

- Kind of generalization of Coppersmith ; complexity $L(1/3)$
- Best algorithm in all fields until 2013

Function field sieve and beyond

- Kind of generalization of Coppersmith ; complexity $L(1/3)$
- Best algorithm in all fields until 2013
- Now **quasi-polynomial algorithms** for finite fields of **small to medium characteristic**
- See Joux, Odlyzko, Pierrot for a recent survey `www-polsys.lip6.fr/~pierrot/papers/Dlog.pdf`

Outline

Generic discrete logarithm algorithms

Discrete logarithms over finite fields

Elliptic curve discrete logarithms

Factorization algorithms

Some side-channel attacks

Lab and tutorial content

Groups used in cryptography

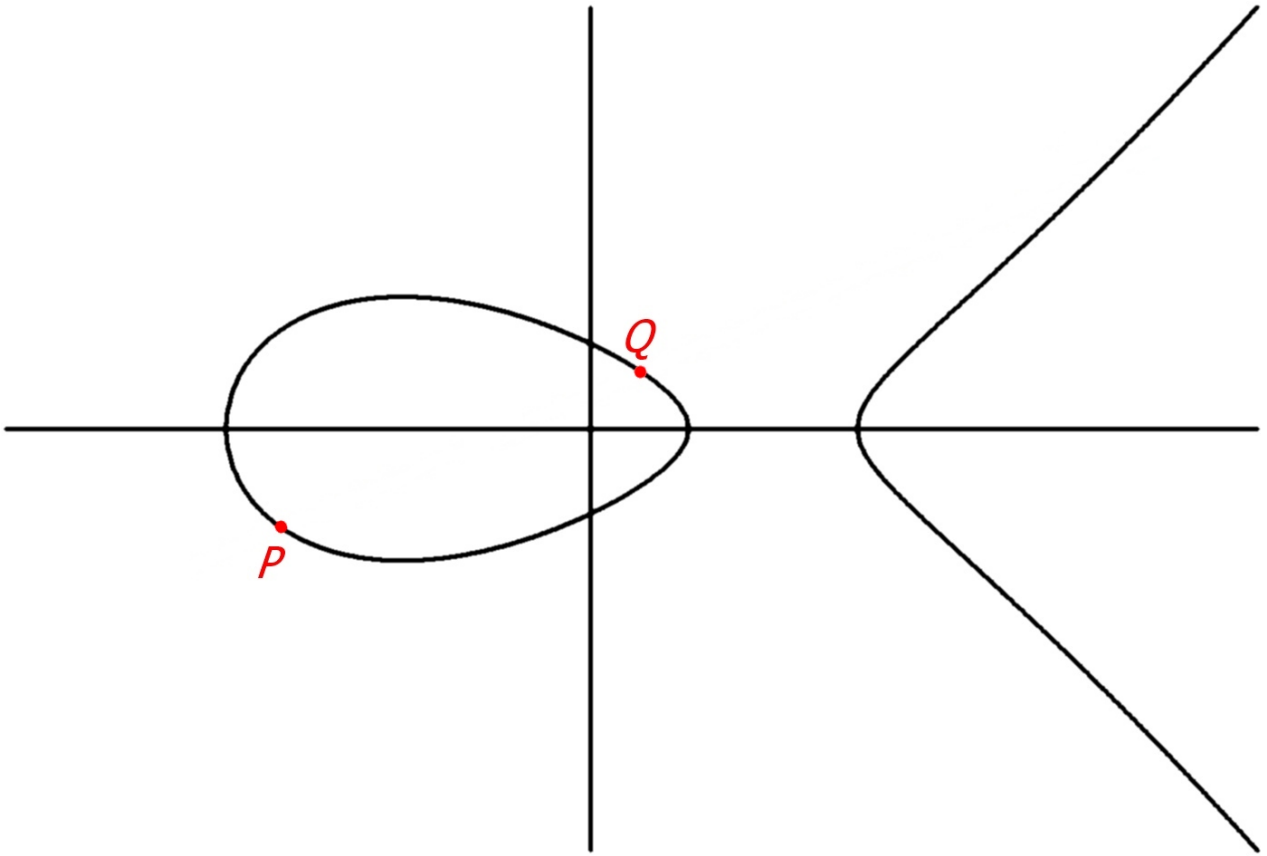
- Finite fields : avoid small characteristic since 2013, otherwise subexponential
- Elliptic curves : best attacks are generic ones for well-chosen families
- Hyperelliptic curves : subexponential for large *genus* : only genus 1 (EC) and genus 2 seriously considered

Elliptic curve cryptography

- 1985 : Koblitz and Miller independently propose to use elliptic curves in cryptography

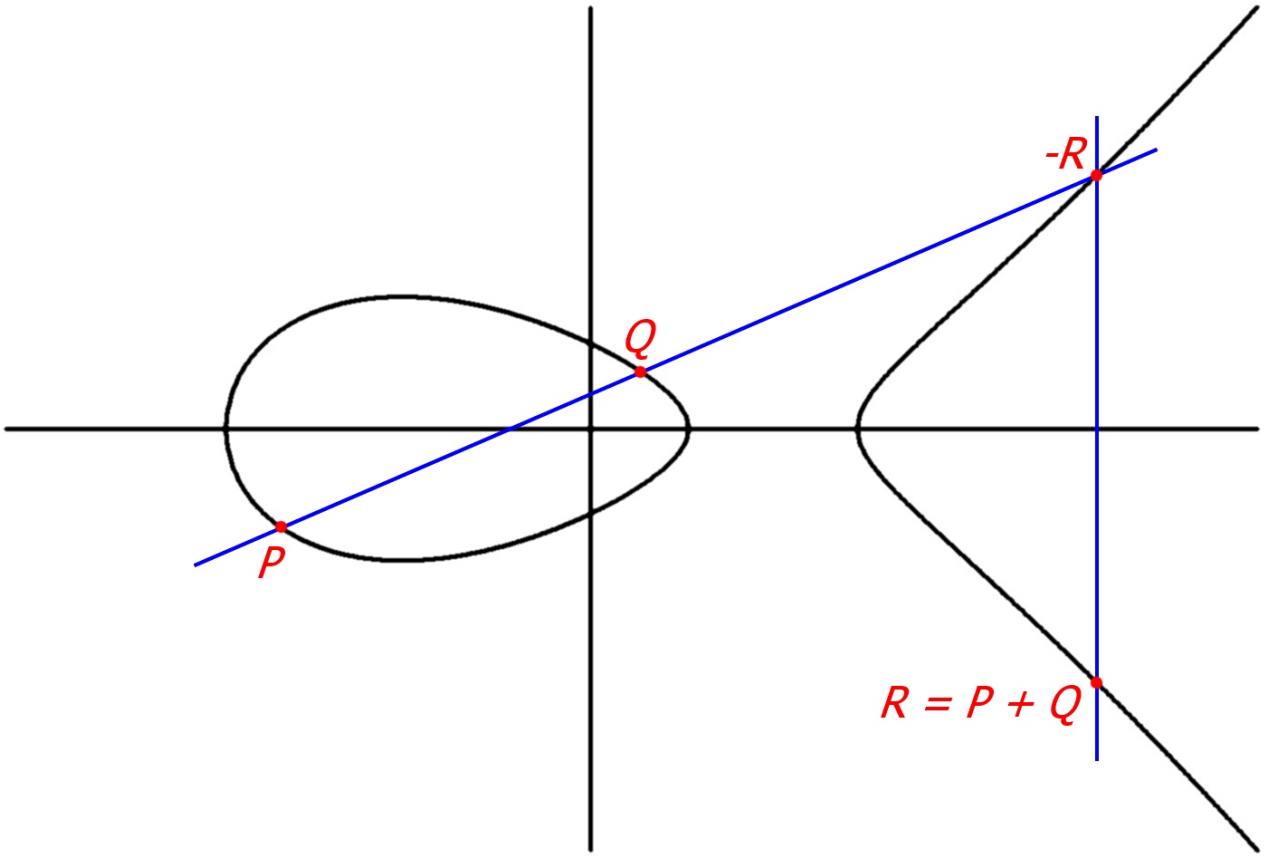
Elliptic curves

$y^2 = x^3 + Ax + B.$



Elliptic curves

$y^2 = x^3 + Ax + B.$



Elliptic curves

- Strange addition law : adding points on (special) curves

Elliptic curves

- Strange addition law : adding points on (special) curves
- Originally mathematical recreation
- Central in Wiles' proof of Fermat's last theorem
  $\forall n > 2$, $\nexists$ non trivial $x, y, z \in \mathbb{Z}$ s.t. $z^n = x^n + y^n$

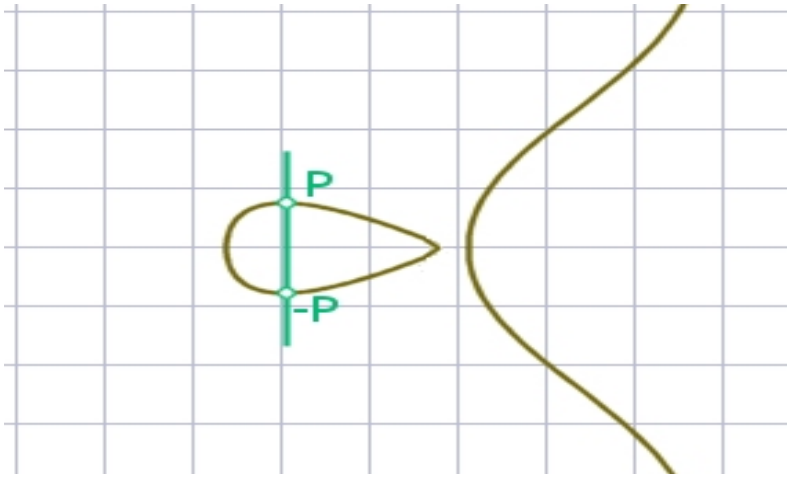
Elliptic curves

- Strange addition law : adding points on (special) curves
- Originally mathematical recreation
- Central in Wiles' proof of Fermat's last theorem
  $\forall n > 2$, $\nexists$ non trivial $x, y, z \in \mathbb{Z}$ s.t. $z^n = x^n + y^n$
- Introduced to crypto in 1985
- Now they build the strongest cryptosystems !
- Also used for factoring middle-size integers and primality proving

"Inverse" of a point

$$y^2 = x^3 + Ax + B.$$

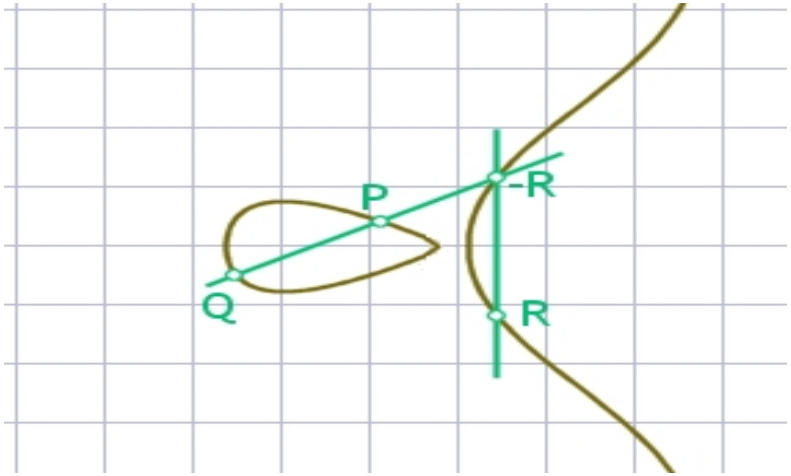
- Let $P := (x, y)$ be a point of a curve
- Define $-P$ as the symmetric of P by the x-axis, that is $-P := (x, -y)$



Adding two distinct points

$$y^2 = x^3 + Ax + B.$$

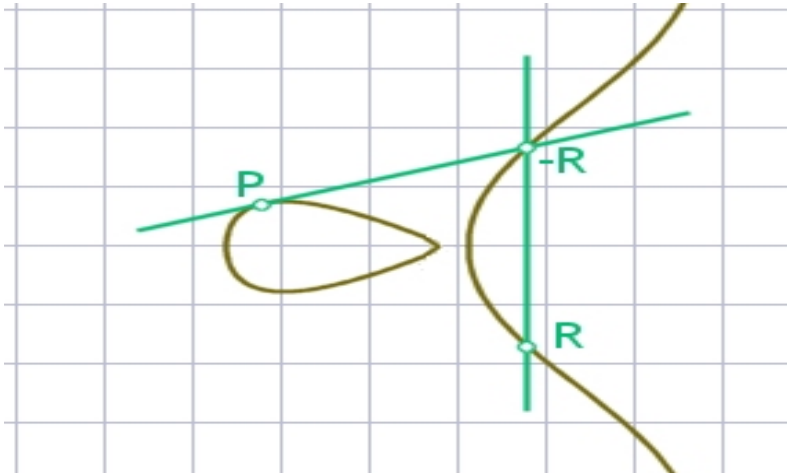
- Let $P := (x_1, y_1)$ and $Q := (x_2, y_2)$ where $x_1 \neq x_2$
- Draw the line through P and Q
- Call $-R$ the third intersection of this line with the curve
- Define $P + Q$ as the symmetric of $-R$ by the x-axis



Doubling a point

$$y^2 = x^3 + Ax + B.$$

- Let $P := (x, y)$
- Draw the tangent line through P
- Call $-R$ the second intersection of this line with the curve
- Define $P + P$ as the symmetric of $-R$ by the x-axis



Secant and tangent rules

- Any non vertical line intersects the curve at exactly three points (counted with multiplicities)

  A tangent point is counted twice

Secant and tangent rules

- Any non vertical line intersects the curve at exactly three points (counted with multiplicities)

  A tangent point is counted twice
- By convention, the *point at infinity* O intersects every vertical line

A group law

- The sum of two points of the curve is a point of the curve (including the point at infinity)
- The point at infinity is the neutral element
- Any element has an inverse
- Can prove associativity : $(P + Q) + R = P + (Q + R)$

Scalar multiplication

$$y^2 = x^3 + Ax + B.$$

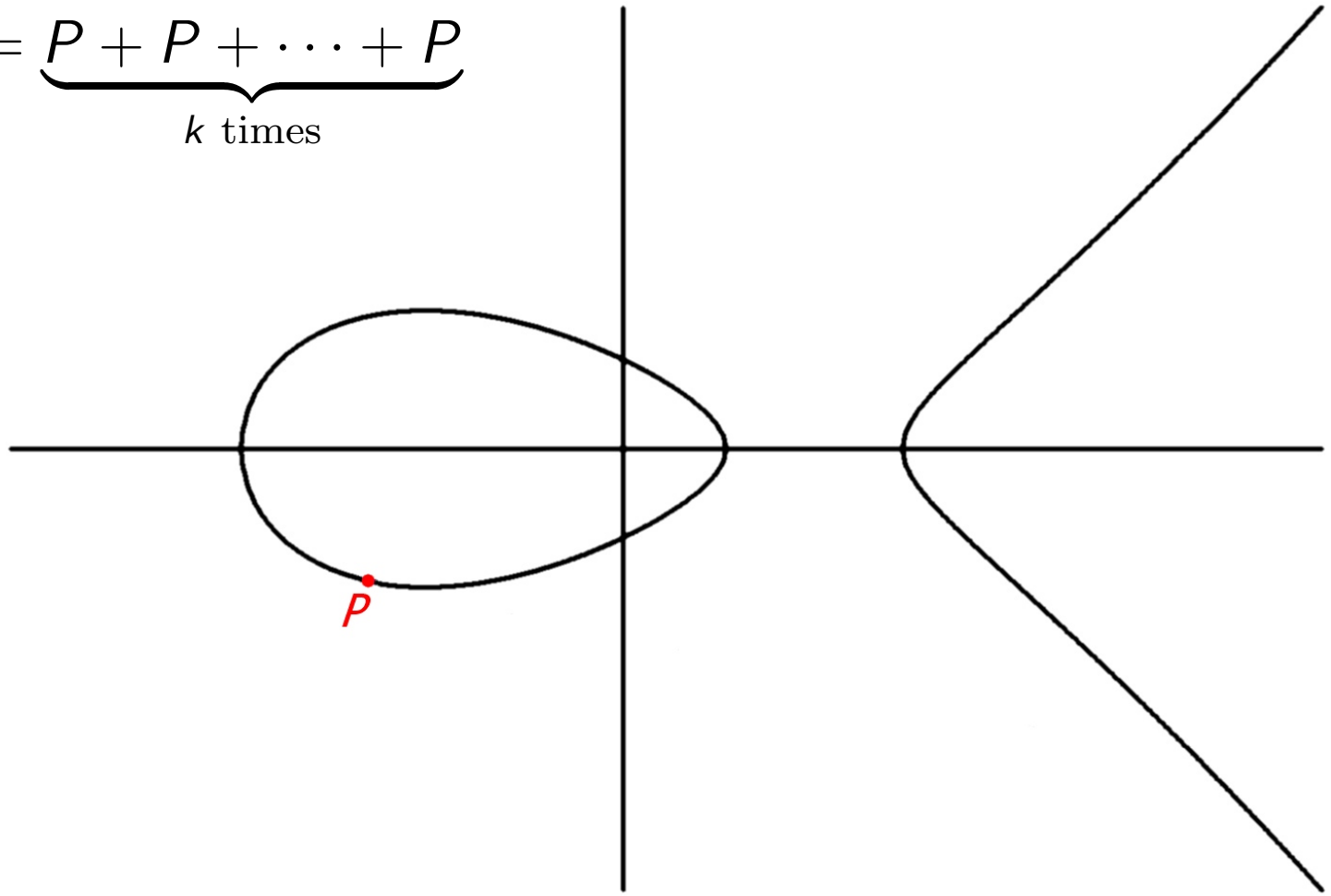
- For $k \in \mathbb{Z}$, define

$$[k](P) := \underbrace{P + P + \ldots + P}_{k \text{ times}}$$

- If K finite, then for any $P \in E(K)$, there is $m \in \mathbb{Z}$ such that $[m](P) = O$ (m is called *the order* of P)

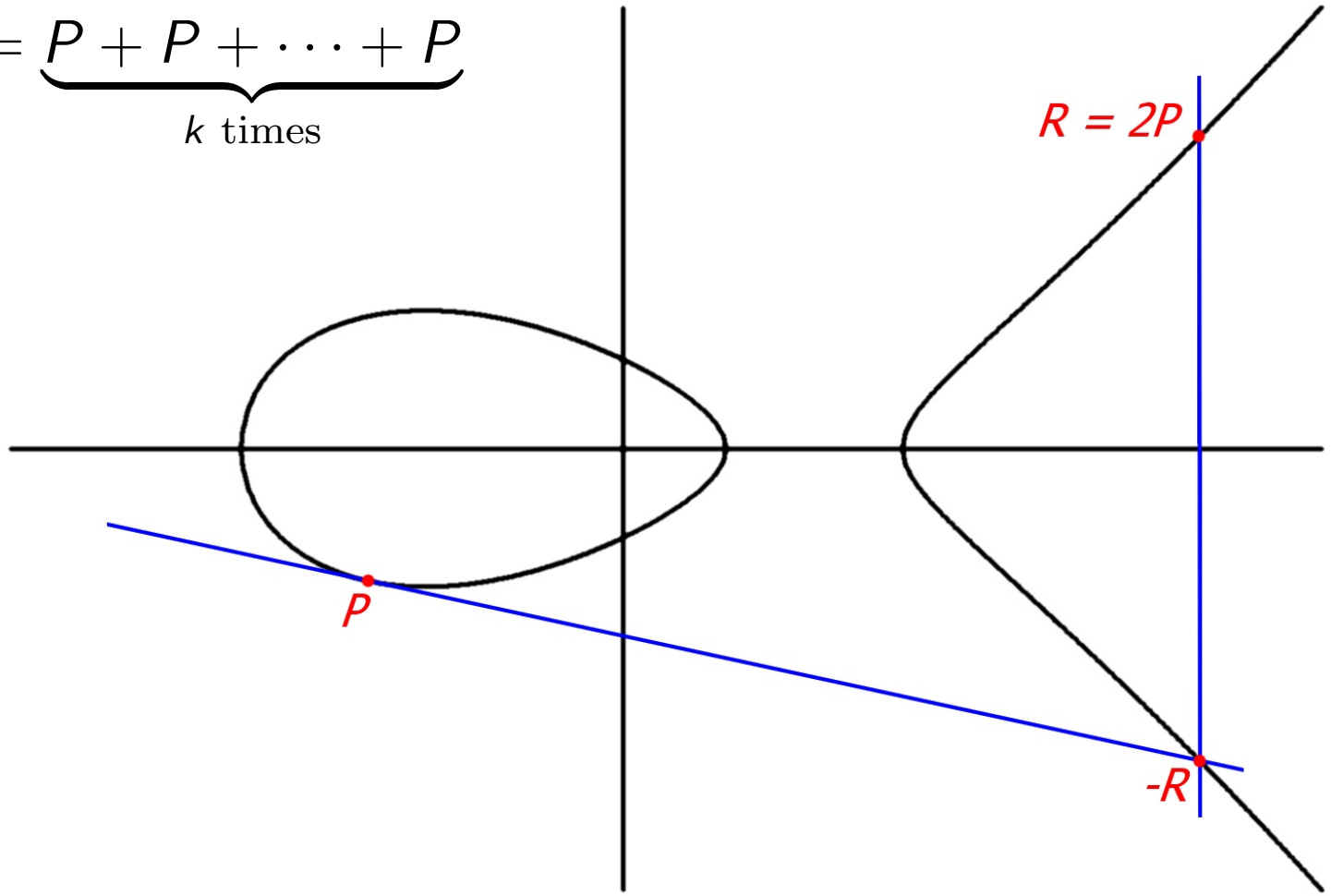
Scalar multiplication

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



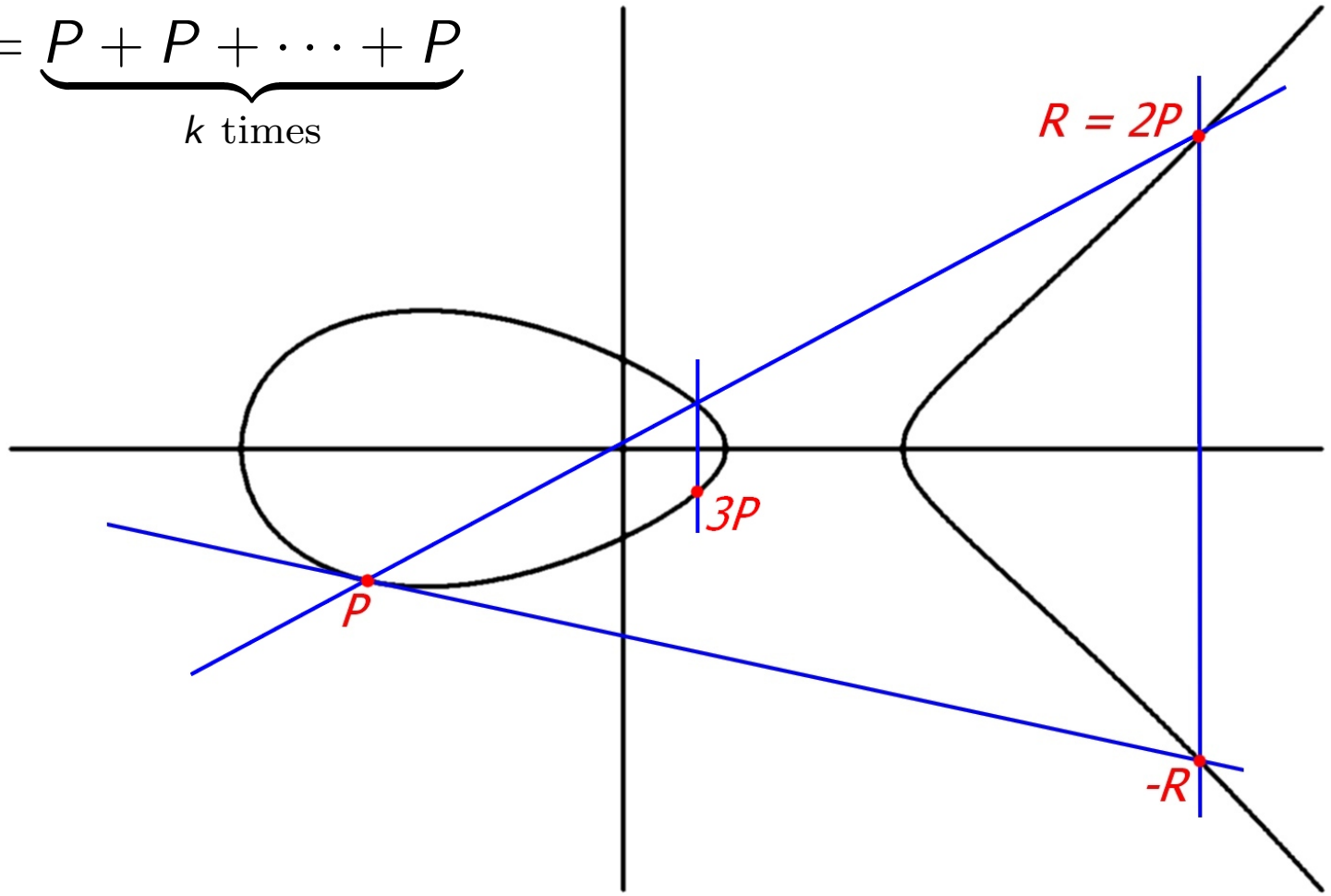
Scalar multiplication

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



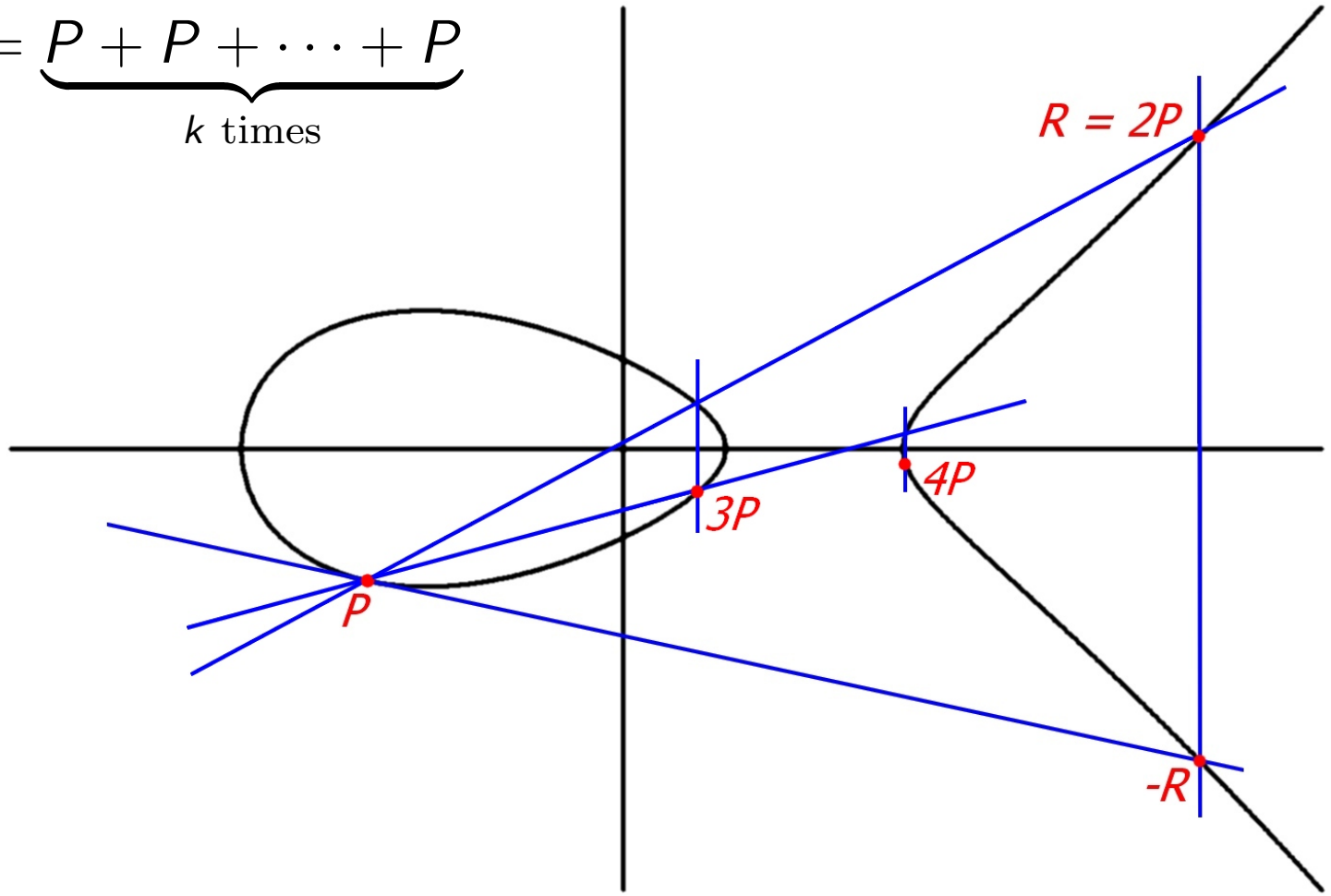
Scalar multiplication

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



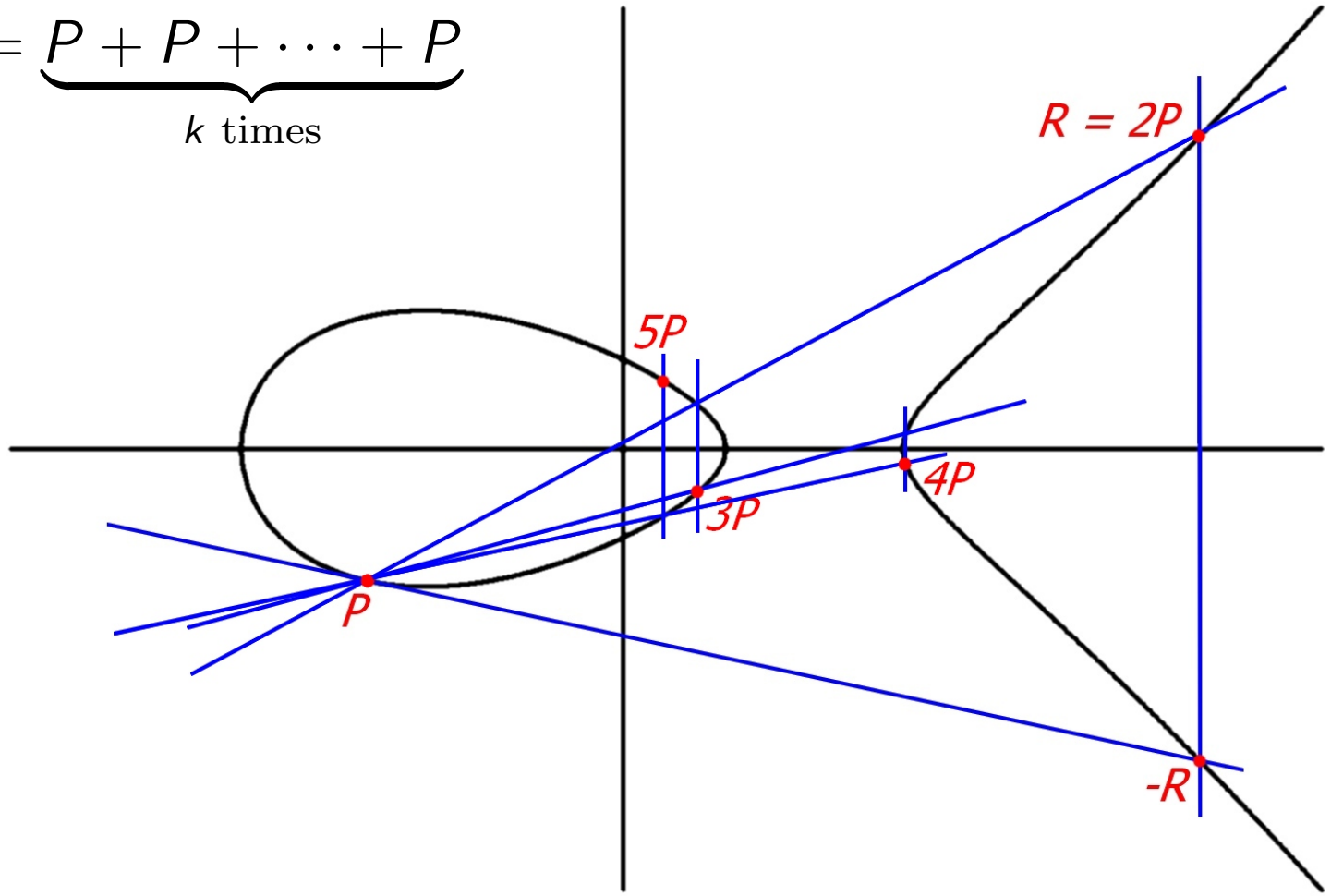
Scalar multiplication

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



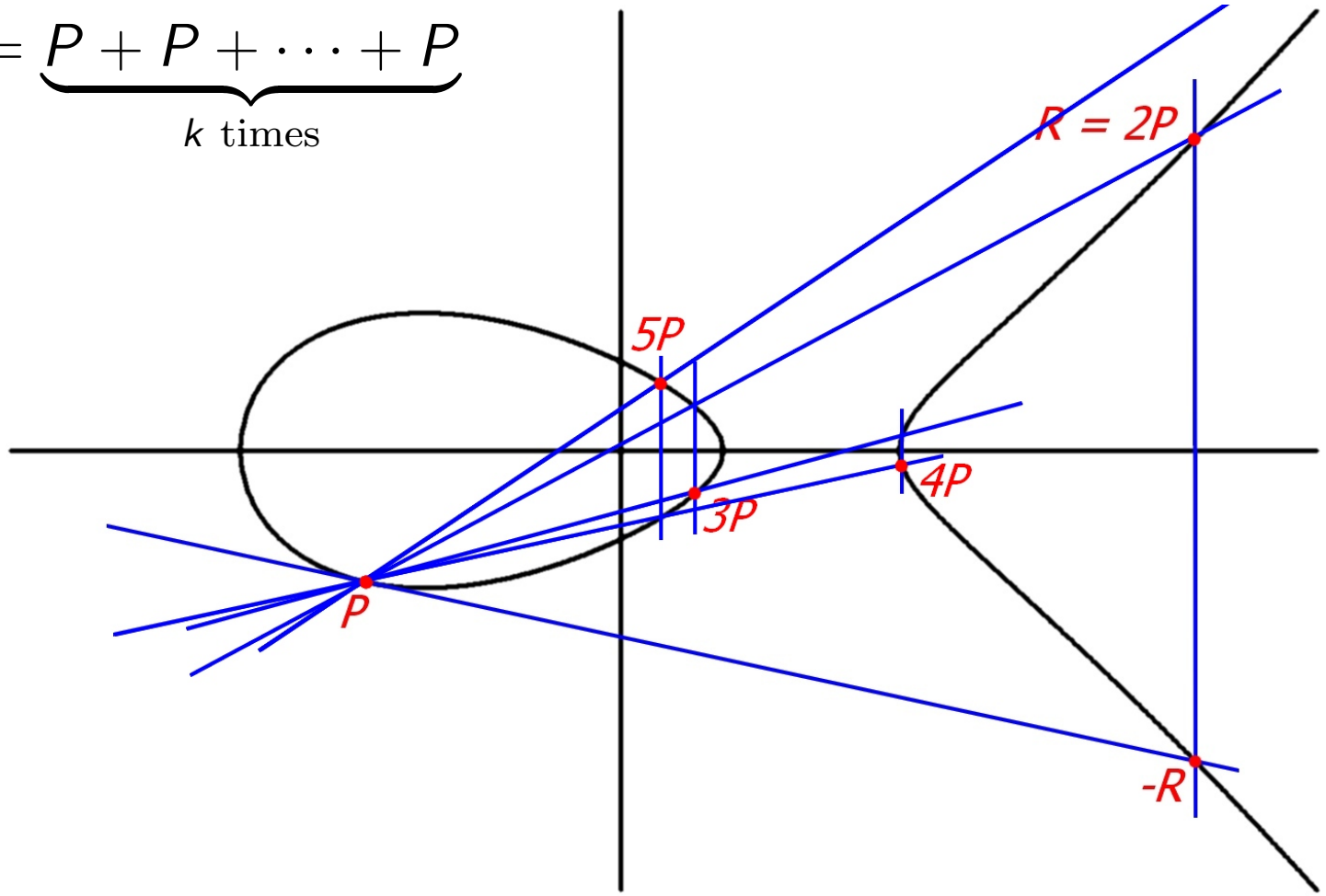
Scalar multiplication

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



Scalar multiplication

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



Elliptic curve discrete logarithm problem (ECDLP)

- Let K be a finite field and let E a curve over K
- Let $P \in E(K)$ with order m
- The function

$$\sigma : \{0, \ldots, m-1\} \rightarrow E(K) : k \rightarrow [k]P$$

is bijective

Elliptic curve discrete logarithm problem (ECDLP)

- Let K be a finite field and let E a curve over K
- Let $P \in E(K)$ with order m
- The function

$$\sigma : \{0, \ldots, m-1\} \rightarrow E(K) : k \rightarrow [k]P$$

  is bijective
- Computing σ is easy. Inverting σ is know as the *elliptic curve discrete logarithm problem* (ECDLP)

ECDLP even harder than DLP and factoring

- ECDLP is (believed to be) a very hard computational problem
- Discrete logarithm and integer factorization problems require numbers as big as 1200 bits when ECDLP is safe with only 160 bits ($\rightarrow$ performance consequences)
- On the other hand, DLP and FP better studied and understood than ECDLP
- Elliptic curve groups very far from generic ones ; we might find particular structures to exploit in future

Reductions to simpler DLP

- Idea : transfer ECDLP to a "simpler" DLP problem through a group homorphism

Reductions to simpler DLP

- Idea : transfer ECDLP to a "simpler" DLP problem through a group homorphism
- **MOV reduction** if $|G|$ divides $q^m - 1$
  Transfer ECDLP to DLP on K^m using pairings

Reductions to simpler DLP

- Idea : transfer ECDLP to a "simpler" DLP problem through a group homorphism
- **MOV reduction** if $|G|$ divides $q^m - 1$
  Transfer ECDLP to DLP on K^m using pairings
- Polynomial time for **anomalous curves**
  Transfer ECDLP to a p-adic elliptic logarithm if $|G| = |K|$

Reductions to simpler DLP

- Idea : transfer ECDLP to a "simpler" DLP problem through a group homorphism
- **MOV reduction** if $|G|$ divides $q^m - 1$
  Transfer ECDLP to DLP on K^m using pairings
- Polynomial time for **anomalous curves**
  Transfer ECDLP to a p-adic elliptic logarithm if $|G| = |K|$
- **Weil descent** for some curves over $\mathbb{F}_{p^n}$
  Transfer ECDLP to the Jacobian of a hyperelliptic curve
- Only work for specific families, not the ones recommended in standards

Index calculus for ECDLP

- Long-standing challenge : how to define "small elements"
- 2005 : first answer by Semaev
  - Factor basis = elements with x-coordinate in a subset
  - Computing a relation is reduced to solving some multivariate polynomial, with additional constraints
- 2008 : attacks by Gaudry and Diem for elliptic curves over $\mathbb{F}_{p^n}$ when n is composite
- 2012 : evidence that ECDLP over $\mathbb{F}_{2^n}$ is subexponential, but in practice generic attacks are still better

Outline

Generic discrete logarithm algorithms

Discrete logarithms over finite fields

Elliptic curve discrete logarithms

Factorization algorithms

Some side-channel attacks

Lab and tutorial content

Integer factorization

- Given a composite number n, compute its (unique) factorization $n = \prod p_i^{e_i}$ where p_i are prime numbers
- Equivalently : compute one non-trivial factor p_i
- We will assume $n = pq$, where p and q are primes

Sieving

- Principle : try every prime number up to $\sqrt{n}$
- Expect to do $O(n^{1/2}/\log n)$ trials

Pollard's rho

- Idea : find x and y such that $\gcd(x-y, n) = p$
  in other words $x = y \bmod p$ but $x \neq y \bmod n$
- Define some "pseudorandom" iteration function f
- Compute iterates x_i and x_{2i}
- Simultaneously compute $\gcd(x_i - x_{2i}, n)$
- By birthday's paradox,
  $x_i = x_{2i} \bmod p$ after $O(p^{1/2})$ trials on average, and
  $x_i = x_{2i} \bmod n$ after $O(n^{1/2})$ trials on average
- Hence we succeed after $O(p^{1/2})$ trials on average

$p-1$ powersmooth

- A number $x = \prod p_i^{e_i}$ is B-powersmooth if $p_i^{e_i} < B$
- Assume $p-1$ is B-powersmooth
- If $s =$ product of all $p_i^{e_i} < B$ then $p-1|s$
  then $g^s = 1 \bmod p$
- We deduce $\gcd(g^s - 1, n) = p$
- Can be computed with square-and-multiply algorithm

Elliptic curve factorization method

- Idea : generalize previous method when neither $p-1$ nor $q-1$ are smooth
- The group order $\#E(\mathbb{F}_p)$ of an elliptic curve can be smooth even when $p-1$ is not !

Elliptic curve addition law

- Let $E : y^2 = x^3 + a_4x + a_6$
- Let $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ two points on the curve
- The chord-and-tangent rules lead to addition law formulae : for example we have $P_1 + P_2 = (x_3, y_3)$ where
  $\lambda = \frac{y_2 - y_1}{x_2 - x_1}, \quad \nu = \frac{y_1 x_2 - y_2 x_1}{x_2 - x_1},$
  $x_3 = \lambda^2 - x_1 - x_2, \qquad y_3 = -\lambda x_3 - \nu$

Elliptic curve addition law

- Let $E : y^2 = x^3 + a_4x + a_6$
- Let $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ two points on the curve
- The chord-and-tangent rules lead to addition law formulae : for example we have $P_1 + P_2 = (x_3, y_3)$ where
  $\lambda = \frac{y_2 - y_1}{x_2 - x_1}, \quad \nu = \frac{y_1 x_2 - y_2 x_1}{x_2 - x_1},$
  $x_3 = \lambda^2 - x_1 - x_2, \qquad y_3 = -\lambda x_3 - \nu$
- These formulae involve divisions
- Over $\mathbb{F}_p$, a division by 0 means P_3 is point at infinity
- Over $\mathbb{Z}_n$, a division fails if $(x_2 - x_1)$ is not invertible
- A failure reveals a factor of n !

Elliptic curve factorization method

1. Choose E and $P = (x, y) \in E(\mathbb{Z}_n)$
2. Let B be a smoothness bound on $\#E(\mathbb{Z}_p)$ for $p|n$
3. Compute $s = \prod p_i^{e_i}$ where $p_i^{e_i} \leq B$
4. We have $[s]P = 0 =$ "point at infinity" modulo p but $[s]P \neq 0$ in $\mathbb{Z}_n$
5. Try to compute $[s](P)$: a division by p must occur and produce an error
6. When a division by some d fails, compute

$$\gcd(d, n) \neq 1$$

Elliptic curve factorization method

1. For a random curve, we expect $\#E(\mathbb{F}_p)$ to be $\pm$ uniformly distributed in

$$\#E(\mathbb{F}_p) \in [(p+1) - 2\sqrt{p}, (p+1) + 2\sqrt{p}]$$

2. Powersmooth probabilities can be estimated
3. In practice : choose the best bound B and choose a random curve until the method works
4. In practice, the method is used as subroutine to factor middle-size integers when $\log_2 n \approx 60 - 80$ bits
5. Remark : runtime depends on the smallest factor

Sieving algorithms

- Goal : find $x \neq \pm 1 \bmod n$ with $x^2 = 1 \bmod n$

Sieving algorithms

- Goal : find $x \neq \pm 1 \bmod n$ with $x^2 = 1 \bmod n$
- Idea : index calculus
  - Search for many relations $\prod p_i^{e_i} = 1 \bmod n$
  - Do linear algebra over $\mathbb{Z}_2$ to deduce a relation $\left(\prod p_i^{f_i}\right)^2 = 1 \bmod n$

Sieving algorithms

- Goal : find $x \neq \pm 1 \bmod n$ with $x^2 = 1 \bmod n$
- Idea : index calculus
  - Search for many relations $\prod p_i^{e_i} = 1 \bmod n$
  - Do linear algebra over $\mathbb{Z}_2$ to deduce a relation $\left(\prod p_i^{f_i}\right)^2 = 1 \bmod n$
- To obtain relations
  - Linear sieve : look for a and $a + n$ both smooth

Sieving algorithms

- Goal : find $x \neq \pm 1 \bmod n$ with $x^2 = 1 \bmod n$
- Idea : index calculus
  - Search for many relations $\prod p_i^{e_i} = 1 \bmod n$
  - Do linear algebra over $\mathbb{Z}_2$ to deduce a relation $\left(\prod p_i^{f_i}\right)^2 = 1 \bmod n$
- To obtain relations
  - Linear sieve : look for a and $a + n$ both smooth
  - Quadratic sieve : let $r = \lceil \sqrt{n} \rceil$, hence $r^2 - n < 2\sqrt{n} + 1$. Look for $(r + x)^2 - n$ smooth

General number field sieve (GNFS)

- Best algorithm to date
- Involves smaller factorization problems, usually solved with other sieves and the elliptic curve method
- Involves large, sparse linear algebra over $\mathbb{F}_2$

General number field sieve (GNFS)

- Best algorithm to date
- Involves smaller factorization problems, usually solved with other sieves and the elliptic curve method
- Involves large, sparse linear algebra over $\mathbb{F}_2$
- Factorization record : 768 bits
  Several research teams and a large computing effort
- "1024-bit RSA about 1000 times more difficult"
  `http://eprint.iacr.org/2010/006.pdf`

Outline

Generic discrete logarithm algorithms

Discrete logarithms over finite fields

Elliptic curve discrete logarithms

Factorization algorithms

Some side-channel attacks

Lab and tutorial content

Side-channel attacks

- So far we have assumed the attacker had access to some public data, and was trying to deduce private data using mathematical algorithms
- Sometimes, the attacker also got access to some oracle answering queries
- In practice, the secret data may be on a smart card, and the attacker may observe the smart card when the computation is done
- Does this help ?

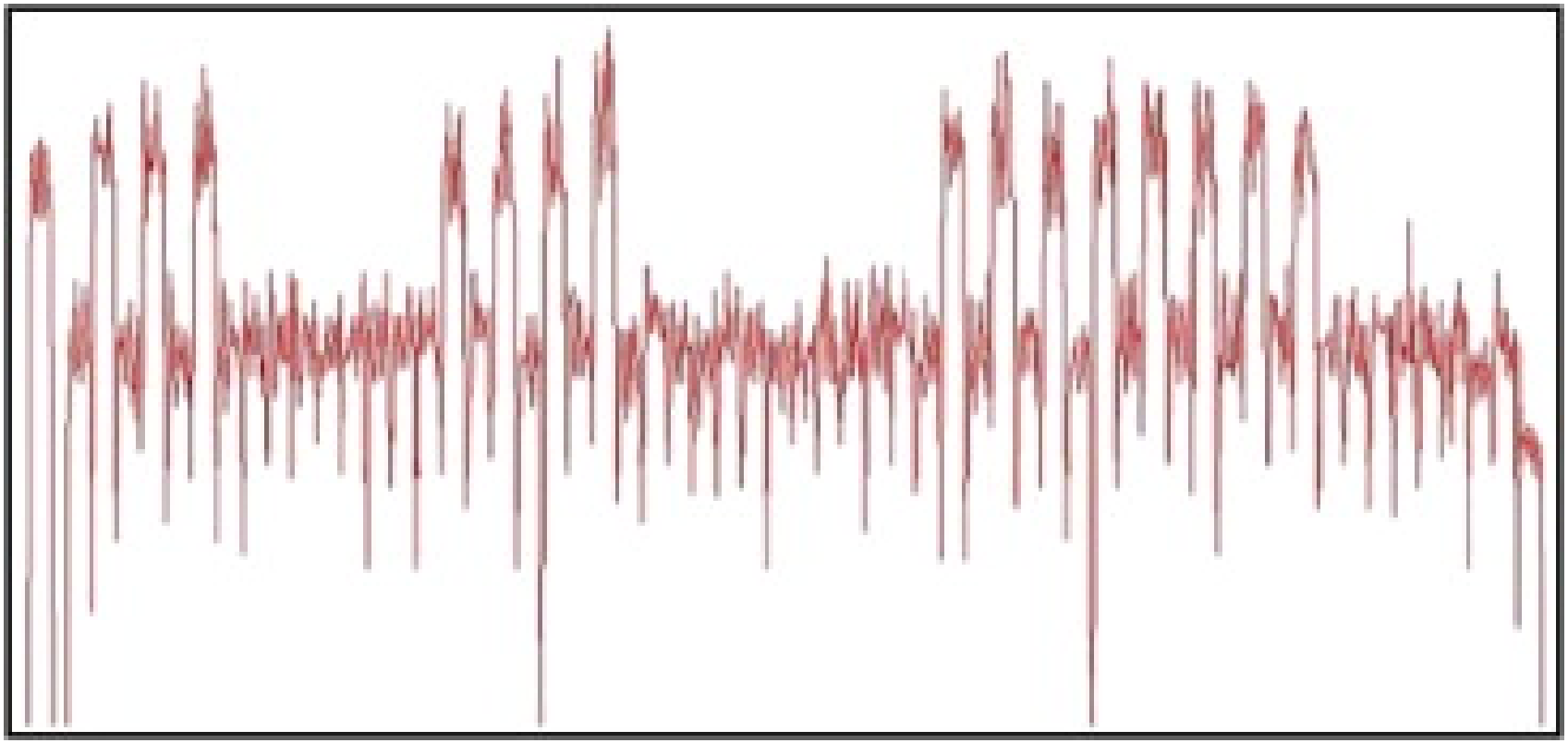
Reminder : Square-and-Multiply

1: Let $x = \sum_{i=0}^{n} x_i 2^i$
2: $a' \leftarrow a$; $c \leftarrow a^{x_0}$;
3: **for** i=1 **to** n **do**
4: $\quad a' \leftarrow a'^2 \bmod p$
5: $\quad$ **if** $x_i = 1$ **then**
6: $\quad\quad c \leftarrow ca' \bmod p$
7: $\quad$ **end if**
8: **end for**
9: **return** c

Power consumption

- Let x be some secret
- Suppose the attacker observes the power consumption of the smart card during the computation g^x mod p
- Suppose the smart card uses the square-and-multiply algorithm
- How does this help ?

Power consumption



Power consumption

- A squaring is done at each step, a multiplication occurs only for odd bits
- The bits of x can be read directly from the power consumption !
- Could be an RSA private key, or a DH random value, or...

Countermeasure

- Add "dummy" multiplications to the algorithm

1: Let $x = \sum_{i=0}^{n} x_i 2^i$
2: $a' \leftarrow a$; $c \leftarrow a^{x_0}$; $d \leftarrow a^{1-x_0}$
3: **for** i=1 **to** n **do**
4: $\quad a' \leftarrow a'^2 \bmod p$
5: $\quad c \leftarrow c(a')^{x_i} \bmod p$
6: $\quad d \leftarrow d(a')^{1-x_i} \bmod p$
7: **end for**
8: **return** c

- Additional operations do not change the result but they will make power consumption look more uniform

Side-channel attacks

- Example of succesfully exploited side-channels (in academic contexts) : time, power consumption, electromagnetic radiations, ...
- Do not require to break the maths, but do require some physical access to the computing device

Outline

Generic discrete logarithm algorithms

Discrete logarithms over finite fields

Elliptic curve discrete logarithms

Factorization algorithms

Some side-channel attacks

Lab and tutorial content

Lab and tutorial content

- `www.keylength.com`
- Discrete log and factorization algorithms
- Implementation of BSGS, Pollard's rho, index calculus (in pairs, each pair focusing on a different algorithm)
- Experimentation on your implementations and comparison with Sage's routines
- Variants of birthday's paradox

Possible related projects

- Elliptic curve primality test
- Index calculus for elliptic curves
- MOV reduction
- Quasi-polynomial time algorithm of Barbulescu-Gaudry-Joux-Thomé