Index calculus on finite fields and applications to pairing based cryptography

Sylvain Duquesne

University Rennes 1

Abidjan CIMPA-ICTP school
April 14, 2017

Institut de recherche mathématique de Rennes

IRMAR - UMR 6625 du CNRS

The discrete logarithm

Definition

Let G be a (multiplicative) group. Let g an element of G of finite order ℓ. Let $H = (1, g^1, g^2, \cdots, g^{\ell-1})$ the subgroup of G generated by g

$$\forall h \in H, \exists n \in [0, \cdots, \ell - 1] \text{ such that } h = g^n$$

n is said to be the discrete logarithm of h in base g and is denoted $\log_g(h)$.
n est determined modulo ℓ

Examples :

- The multiplicative group of a finite field : $\mathbb{F}_q^*$
- An elliptic curve
- The Jacobian of an hyperellitic curve

Goal : find a group where finding the discrete logarithm is difficult and use it in cryptography

Diffie-Hellman key exchange

Public parameters : a group G, an element g in G of order ℓ

- A picks a random number a in $[1, \ell - 1]$
- A computes g^a in G and sends it to B
- B picks a random number b in $[1, \ell - 1]$
- B computes g^b in G and sends it to A
- B gets g^a and computes $g^{ab} = (g^a)^b$
- A gets g^b and computes $g^{ab} = \left(g^b\right)^a$
- A and B share a common secret key g^{ab}.

An eavesdropper knows g and intercepts g^a, g^b but cannot deduce g^{ab} without solving a discrete logarithm problem

Computing the discrete logarithm

Definition

An algorithm to compute the discrete log is said to be generic if it uses only the following operations

- the composition of two groups elements
- the inverse of an element
- the equality test

In other words, it can be used on any group

Theorem (Shoup)

Let p be the largest prime number dividing the order ℓ of the element g. Computing a discrete logarithm using a generic algorithm requires at least $O(\sqrt{p})$ operations in the group

Combining Polhig-Hellman with BSGS or Pollard ρ method allows to compute a DL in $O(\sqrt{p})$ operations in the group.

A candidate for G : $\mathbb{F}_p^*$

p prime, $\mathbb{F}_p$ finite field
The set of non-zero elements in $\mathbb{F}_p$ is a (multiplicative) group of order $p-1 \to$ natural candidate for G

Index calculus algorithm can compute the discrete logarithm in such a group in subexponential time

Security level of 80 bits $\to p \sim 2^{1024}$
Same security as RSA

In practice, we chose p a 1024 bits prime number such that $p-1$ is divisible by a 160 bits prime number ℓ. In this case, the operations take place in $\mathbb{F}_p^*$ but the keys (the exponents) are in $\mathbb{Z}/\ell\mathbb{Z}$.

Smaller keys than RSA (160 bits instead of 1024).

Diffie-Hellman key-exchange on $\mathbb{F}_p^*$ for 80 bits security

We chose ℓ a 160 bits prime number and p a 1024 bits prime number such that $p-1=k\ell$. Let g be an element in $\mathbb{F}_p^*$ of order ℓ. Public parameters are ℓ, p and g.

- A picks a random number a in $[1, \ell-1]$
- A computes g^a modulo p and sends it to B
- B picks a random number b in $[1, \ell-1]$
- B computes g^b modulo p and sends it to A
- B gets g^a and computes $g^{ab} = (g^a)^b$ modulo p
- A gets g^b and computes $g^{ab} = \left(g^b\right)^a$ modulo p
- A and B share the common secret key g^{ab}

The standard procedure to generate ℓ, p and g is given by the NIST
http ://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf

$$\begin{aligned}
\text{for instance } \ell &= 2^{160}+7 \\
p &= 1+\left(2^{160}+7\right)\left(2^{864}+218\right) \sim 2^{1024} \\
g &= 2^{\frac{p-1}{\ell}} \bmod p
\end{aligned}$$

Other candidates

- Other finite fields. In particular those of the form $\mathbb{F}_{2^n}$. Index calculus works much better (since 2013) : avoid
- Elliptic curves and genus 2 (hyperelliptic) curves for which nobody knows better attacks than generic ones : 160 bits are sufficient for 80 bits of security
- Curves of larger genus but the Index calculus algorithm can be adapted

Advantages and Drawbacks compared to RSA

- Smaller key size
- Faster decryption (eg 160 bits exponent instead of 1024)
- Slower encryption (if small e is used in RSA)
- Trivial key generation

Principle of Index calculus (Western-Miller, Kraitchik)

We assume, to simplify, that $\# G = \ell$ (ie all elements of G are a power of g). We want to compute the discrete log of h

1. Construct a "factor basis" made of some particular elements of G, $(g_i)_{i=1..c}$. By definition, we have $g_i = g^{\log_g(g_i)}$
2. Find relations between these elements of the form

$$g^{\alpha_g} h^{\alpha_h} = g_1^{\alpha_1} g_2^{\alpha_2} \cdots g_c^{\alpha_c}$$

This give relations of the form

$$g^{\alpha_g} g^{\log_g(h)\alpha_h} = g^{\log_g(g_1)\alpha_1} g^{\log_g(g_2)\alpha_2} \dots g^{\log_g(g_c)\alpha_c}$$

and then

$$\alpha_g = -\log_g(h)\alpha_h + \log_g(g_1)\alpha_1 + \log_g(g_2)\alpha_2 + \cdots + \log_g(g_c)\alpha_c$$

which is a linear equation between $\log_g(h)$ and the $\log_g(g_i)$.

Principle of Index calculus (Western-Miller, Kraitchik)

3. When you have $c+1$ independent relations of this form, solve the system (standard linear algebra) assuming that $\log_g(h)$ and the $\log_g(g_i)$ are the unknowns. The solution then gives $\log_g(h)$

For efficiency, must find a balance between step 2 and step 3 (which are contradictory)

This algorithm is generic but is efficient only if a good factor basis can be used

- on $\mathbb{F}_p^*$, we choose the small prime numbers
- on $\mathbb{F}_{2^n}^*$, we choose the polynomials of small degrees
- on large genus curves, we choose elements of small degrees

Pairings in cryptography

Definition

In cryptography, a pairing is a map

$$e : (G_1, +) \times (G_2, +) \to (G_3, x)$$

- bilinear, ie $e(g_1 + g_1', g_2) = e(g_1, g_2)e(g_1', g_2)$
- non degenerate, ie $\forall g_1 \in G_1, \exists g_2 \in G_2$ tq $e(g_1, g_2) \neq 1$
- easy to compute

Applications

- Transfert of discret log.
- tri-partite key-exchange.
- identity based cryptography.
- Short signatures
- Broadcast encryption

The Tate pairing

Let E be an elliptic curve defined over $\mathbb{F}_q$ and containing a subgroup of prime order ℓ. Let k be the embedding degree relatively to ℓ

$$e_T : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_{q^k})[\ell] \rightarrow \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^\ell$$

The embedding degree

- It is the smallest extension of $\mathbb{F}_q$ that contains all the ℓ-torsion points.
- It is usually very large (same size as q).
- It is small for supersingular curves
  ($k \leq 6$ in char. 3, $k \leq 4$ in char. 2, $k \leq 2$ in char. ≥ 5)

Security issue

The DL should be hard to solve in the 3 groups involved

ℓ should be large enough to avoid generic attacks

q^k should be large enough to avoid index calculus

The Barreto-Naehrig (BN) curves

Prime order curves given by an equation $y^2 = x^3 + b$ satisfying

- $p = 36u^4 + 36u^3 + 24u^2 + 6u + 1$
- $\ell = 36u^4 + 36u^3 + 18u^2 + 6u + 1$

Properties

- $k = 12$, **"optimal"** for 128 bits security level.
- Many implementation tricks available.

$\Rightarrow$ Massively used since for 10 years

But...

The polynomial form of p can be used to significantly improved index calculus

- $u = -2^{62} - 2^{55} - 1$ provides only 100 bits of security
- BN curves no more optimal
- Others families must be considered (BLS, KSS, ...)