State-of-the-art in implementing algorithms for the (ordinary) discrete logarithm problem

Antoine JOUX Reynald LERCIER

SCSSI CELAR

Issy-les-Moulineaux Bruz

Antoine.Joux@ens.fr lercier@celar.fr

FRANCE

November 1999

Plan

- 1. Introduction
- 2. Background
  - 2.a. General Methods
  - 2.b. Index Calculus Method
    - Gaussian Integers
    - Number Field Sieve
- 3. Number Field Sieve
  - 3.a. NFS Polynomials
  - 3.b. Finding Relations
  - 3.c. Linear Algebra
    - Gaussian Elimination
    - Lanczos Algorithm
  - 3.d. Individual Logarithms
- 4. Results
  - 4.a. GF(p90)
  - 4.b. GF(p100)
- 5. Conclusion

Discrete logarithm problem in $\mathbb{F}_p$

Th. : Let p be a prime, then $(\mathbb{F}_p^*, \times)$ is cyclic.

So, there exists an element $g \in \mathbb{F}_p^*$ such that,
for any element $y \in \mathbb{F}_p^*$,
there exists an integer x such that,

$$y = g^x.$$

Notation : $x = \log_g(y)$.

Algorithmical considerations :

Computing y from x is "easy" : algorithms whose complexity is polynomial in $\log(p)$.

Computing x from y is "hard" : algorithms whose complexity is sub-exponential in $\log(p)$.

Cryptographic implications

Computing discrete logarithms is of first importance in the field of cryptography.

Among others, it is :

- a tool for studying shift registers sequences (R. Lidl and H. Niederreiter. *Finite Fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Addison–Wesley, 1983).
- the basis for many public-key cryptosystems, the first one being the Diffie Hellman key exchange algorithm
  (W. Diffie and M. Hellman. *New directions in cryptography*, volume 22 of *IEEE Trans, Inform, Theory*. 1976).

General methods

Following Shoup, the best algorithm that one can think of to compute discrete logarithms in a "general group" of prime cardinality $O(p)$ is of complexity at best $O(p^{\frac{1}{2}})$
(V. Shoup, *Lower Bounds for Discrete Logarithms and Related Problems. Advances in Cryptology, EUROCRYPT'97*, volume 1223 of *Lecture Notes in Computer Science*. 1997).

Precisely for $\mathbb{F}_p^*$:

1. Thanks to the Pohlig-Hellman algorithm, computing x is equivalent to compute discrete logarithms in cyclic subgroups whose orders q are divisors of $p-1$.
2. Computing discrete logarithms in cyclic subgroups of $\mathbb{F}_p^*$ can be done in $O(q^{\frac{1}{2}})$ thanks to Baby-Giant steps methods, or better, Pollard-ρ type methods.

Index calculus method

Taking advantage of the specificity of $\mathbb{F}_p^*$, there exists much better algorithms to compute discrete logarithms.

These algorithms are so called "index-calculus" algorithms.

Three main families for $\mathbb{F}_p$.

Original Method : Due to Kraitchik ? or Western and Miller (1968) ? Complexity equal to $L_p[\frac{1}{2}, c]$.

Gaussian Integer method : Due to Coppersmith, Odlyzko and Schroeppel (1986). Complexity equal to $L_p[\frac{1}{2}, 1]$.

General Number Field Sieve : Improved version by Schirokauer (1993) of a Gordon's proposal. Complexity equal to $L_p[\frac{1}{3}, \frac{64}{9}^{\frac{1}{3}}]$.

Here,

$$L_x[\alpha, c] = O\left(e^{(c+o(1))(\log x)^{\alpha}(\log\log x)^{1-\alpha}}\right).$$

Index calculus method
An overview

Basically, two main steps.

Step 1 : One fixes a subset $S = \{p_1, \ldots, p_{|S|}\}$ of $\mathbb{F}_p^*$ called the "factor base" and tries to write elements s of $\mathbb{F}_p$ whose logarithm is known as a product of elements of $|S|$. So, we have equations of the form

$$\log_g s = \sum_{p \in S} e_p \log_g p.$$

When enough such relations are collected, one obtains the quantities $\log_g p$ via a linear inversion modulo $p - 1$.

Step 2 : To find $\log_g y$, one tries random integers z until $g^z y$ is a product of elements of S. Then

$$\log_g y = -z + \sum_{p \in S} e_p \log_g p.$$

Gaussian Integer method

Idea : Mapping between an imaginary quadratic number field and $\mathbb{F}_p$.

Precisely : One chooses a small integer r such that $-r = m^2$ in $\mathbb{F}_p$. Then, we write $m = n/d$ in $\mathbb{F}_p$ with $n, d \simeq O(\sqrt{p})$ and we consider "the number fields" $\mathbb{Q}(\alpha)$, $\mathbb{Q}(\beta)$ respectively defined by

$$f_\alpha(X) = dX - n \text{ and } f_\beta(X) = X^2 + r.$$

Respective integer rings noted $\mathcal{O}_\alpha \simeq \mathbb{Z}$ and $\mathcal{O}_\beta$.

Mapping from the ideals of $\mathcal{O}_\alpha$ to $\mathbb{F}_p$:
Obvious, just $(a + b\alpha) \to a + bm$.

Mapping from the ideals of $\mathcal{O}_\beta$ to $\mathbb{F}_p$:
Easy when the ideal class group of $\mathcal{O}_\beta$ has cardinality 1 since

$$\forall \text{ ideal } \mathfrak{p} \text{ of } O_\beta,\ \exists (a_\mathfrak{p}, b_\mathfrak{p}) \in \mathbb{Z}^2,\ \mathfrak{p} = (a_\mathfrak{p} + b_\mathfrak{p}\beta).$$

Then, the mapping is $\mathfrak{p} \to a_\mathfrak{p} + b_\mathfrak{p} m$.

Gaussian Integer method
Algorithm

Thanks to the previous mappings, one sets

$$S = \{p \in \mathbb{Z}, p \text{ prime } < B_\alpha\} \cup$$
$$\{a_\mathfrak{p} + b_\mathfrak{p} m, \mathfrak{p} \text{ prime ideal of } O_\beta, \mathrm{Norm}(\mathfrak{p}) < B_\beta\}$$
$$\cup \{u \in \mathcal{O}_\beta, \mathrm{Norm}(u) = 1\}.$$

And step 1 is reformulated as follows :

Step 1 : Looks for couples (a, b) such that $(a + b\alpha)$ and $(a + b\beta)$ are both smooth. Then

$$v \prod_{p \in S} p^{e_p} = a + bm = u \prod_{\mathfrak{p} \in S} (a_\mathfrak{p} + b_\mathfrak{p} m)^{e_\mathfrak{p}}.$$

When enough such relations are collected, one obtains the quantities $\log_g p$, $\log_g v$, $\log_g u$ and $\log_g a_\mathfrak{p} + b_\mathfrak{p} m$ via a linear inversion done modulo $p - 1$.

Gaussian Integer method
Records

1991 : 192 bits (58 digits) by B.A. Lamacchia and A.M. Odlyzko, about 2 months on a DEC VAX 8850.

1996 : 85 digits by D. Weber, on about 120 workstations using their idle time of about 30 mips year.

1998 : 90 digits by A. Joux and R. Lercier, about 5 months on a PPro 180MHz based PC.

Number Field Sieve

Idea : Generalizing the Gaussian Integer method to arbitrary Number Fields $\mathbb{Q}(\alpha)$ or $\mathbb{Q}(\beta)$.

We still have

$$(a+b\alpha)=\prod_{\mathfrak{p}\in S}\mathfrak{p}^{e_\mathfrak{p}} \text{ and } (a+b\beta)=\prod_{\mathfrak{p}\in S}\mathfrak{p}^{e_\mathfrak{p}},$$

but ...

Problem : Mapping ideals of $\mathcal{O}_\alpha$ (resp. $\mathcal{O}_\beta$) to $\mathbb{F}_p$ is a difficult task. Mainly because the Unit and Class Groups are no more obvious.

In practice, we cannot even compute them ! This yields a so called Obstruction Group.

Moreover, we can't get rid of ideals and the inversion step done modulo $p-1$ is no more valid as it is.

Fortunately, O. Schirokauer found a way to raise these obstructions by adding to relations what he called "maps".

Number Field Sieve Records

For any $\mathbb{F}_p$

1994 : 25 digits by D. Weber, about 1 day on a SPARC ELC.

1995 : 65 digits by D. Weber, on 130 workstations using their idle time of about 5 mips year.

1998 : 85 digits by D. Weber, on 100 workstations using their idle time of about 44.5 mips year.

1999 : 100 digits by A. Joux and R. Lercier, about 1 year on a Pentiun II 450MHz based PC.

For primes p with a special form

1998 : 129 digits ($(739 \cdot 7^{149} - 736)/3$) by D. Weber, about 1 month on a SPARC 20 workstation for computing *one* discrete logarithm.

Number Field Sieve
Main steps

– Finding “good” GNFS polynomials. New

– Sieving to find relations.

– Inverting the linear system. Improved

– Computing specific logarithms. New

Remark : With such a scheme, there is no more need to restart the linear algebra from scratch whenever a new logarithm must be computed.

NFS polynomials

$f_\alpha(X)$ and $f_\beta(X)$ must satisfy :

- $f_\alpha(X)$ and $f_\beta(X)$ are irreducible over $\mathbb{Z}$,
- $\mathrm{Gcd}(\mathrm{Coefficients}(f_\alpha)) = 1$,
- $\mathrm{Gcd}(\mathrm{Coefficients}(f_\beta)) = 1$,
- $f_\alpha(X) \neq \pm f_\beta(X)$,
- $\exists m \in \mathbb{F}_p^*,\ f_\alpha(m) = f_\beta(m) = 0$.

For efficiency purposes, we try some ideals $(a + b\alpha)$ or $(a + b\beta)$ of norms as small as possible $\Rightarrow f_\alpha(X)$ and $f_\beta(X)$ must have coefficients as small as possible.

But, in any case,

$$\mathrm{Resultant}(f_\alpha(X), f_\beta(X)) = k \cdot p,\ k \in \mathbb{Z}^*.$$

Additional improvements :

- polynomials with real roots ?
- polynomials with many roots modulo small primes ?

Known methods

Gaussian Integer method :

$$f_\alpha(X) = a_1X + a_0,\ f_\beta(X) = X^2 + r,$$

with $a_i \simeq p^{\frac{1}{2}}$ and $r = O(1)$.

Montgomery's method :

$$f_\alpha(X) = a_2X^2 + a_1X + a_0,$$
$$f_\beta(X) = b_2X^2 + b_1X + b_0,$$

with $a_i, b_i \simeq p^{\frac{1}{4}}$.

Base m method :

$$f_\alpha(X) = X - m,\ f_\beta(X) = \sum_{i=0}^{d} b_iX^i,$$

with $m, b_i \simeq p^{\frac{1}{d+1}}$ and $\sum_{i=0}^{d} b_im^i = p$.

Generalization of the Gaussian Integer method

- Choose a polynomial f_β of degree $d+1$ with very small coefficients and a root m in $\mathbb{F}_p$.
- Let m be a root of $f_\beta(X)$ in $\mathbb{F}_p$.
  Applying LLL algorithm to the lattice

$$\begin{pmatrix} K & Km & K(m^2 \bmod p) & \cdots & K(m^d \bmod p) \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix},$$

we obtain a vector $\begin{pmatrix} 0 \\ a_0 \\ \vdots \\ a_d \end{pmatrix}$ with $a_i \simeq p^{\frac{1}{d+1}}$.

Then,

$$f_\alpha(m) = \sum_{i=0}^{d} a_i m^i = 0.$$

NFS relations

Let S_α, S_β, be a set of "small" prime ideals in $\mathcal{O}_\alpha$, $\mathcal{O}_\beta$.

Then, sieving techniques yields $|S_\alpha| + |S_\beta| + O(1)$ equations

$$(a + b\alpha) = \prod_{\mathfrak{p}\in S_\alpha} \mathfrak{p}^{e_\mathfrak{p}},\ (a + b\beta) = \prod_{\mathfrak{p}\in S_\beta} \mathfrak{p}^{e_\mathfrak{p}}.$$

Thanks to a matrix inversion done modulo $p - 1$, we have

$$\forall \mathfrak{p} \in S_\alpha,\ \mathfrak{p} = I_\mathfrak{p}^{p-1} \prod_{(a,b)} (a + b\alpha)^{e_{a,b}} \text{ and}$$

$$\forall \mathfrak{p} \in S_\beta,\ \mathfrak{p} = I_\mathfrak{p}^{p-1} \prod_{(a,b)} (a + b\beta)^{e_{a,b}},$$

and it remains $|S_\beta| + O(1)$, $|S_\alpha| + O(1)$ supplementary equations given by

$$I^{p-1} = \prod_{(a,b)} (a + b\alpha)^{e_{a,b}}, I^{p-1} = \prod_{(a,b)} (a + b\beta)^{e_{a,b}}.$$

Schirokauer's ideas

For each equation

$$I^{p-1} = \prod_{(a,b)} (a + b\alpha)^{e_{a,b}},$$

one computes $O(1)$ number of linear "maps" $(\lambda_1, \ldots, \lambda_{O(1)})$ defined from $\mathbb{Q}(\alpha)$ to $\mathbb{Z}/(p-1)\mathbb{Z}$.

With a small linear arithmetic done modulo $(p-1)$, it is easy to compute some suitable combinations of the last $O(1)$ equations with any of the first $|S_\beta|$ equations to obtain $|S_\beta|$ equations such that their "map vectors" are equal to

$$(0, \ldots, 0).$$

Then, Schirokauer's results state that,
if Leopoldt's conjecture is true,
such equations are true for algebraic integers,

$$\exists \delta \in \mathcal{O}_\alpha,\ \delta^{p-1} = \prod_{(a,b)} (a + b\alpha)^{e_{a,b}}.$$

Sieving by vectors (1/2)

With sieving methods, instead of testing each principal ideal $(a + b\alpha)$ or $(a + b\beta)$, we mark in a sieving array multiples of small prime ideals $\mathfrak{p}$.

Since the prime ideals $\mathfrak{p}$ we consider are degree one prime ideal, there exists a root ρ of $f_\alpha(X) \bmod \pi$ where $\pi = \mathrm{Norm}(\mathfrak{p})$.

Then, multiples of $\mathfrak{p}$ define a lattice given by

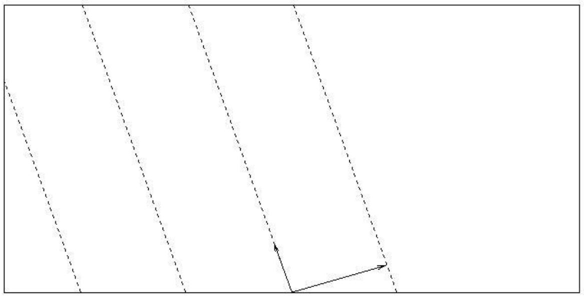
$$\begin{pmatrix} \rho & \pi \\ 1 & 0 \end{pmatrix}$$

Remark : In practice, it's important to use short vectors of this lattice to speed up the sieving.

Sieving by vectors (2/2)

To sieve larger areas, one usually prefers to work with the sub-lattice defined by an ideal $\mathfrak{q}$. This technique is due to Pollard (1993).

Precisely, we only consider algebraic integers $a + b\alpha$ which are multiples of a prime ideal $\mathfrak{q}$. Such algebraic integers are on a lattice $L_{\mathfrak{q}}$.



Then, any multiples of prime ideals $\mathfrak{p}$ are on the sub-lattice $L_{\mathfrak{q}} \cap L_{\mathfrak{p}}$.

Remark : In practice, most of the time needed for such algorithms is spent in memory accesses.

Linear Algebra

At the end of the sieving stage, we have a (huge) matrix to invert. This matrix has 2 parts :

– a large but sparse part with very small entries (mainly ± 1),

– few dense columns whose elements are integers of size p due to Schirokauer's maps.

Two classical phases.

– A Structured Gaussian Elimination (introduced by Lamachia and Odlyzko in 1991) to significantly reduce the dimension of the linear system we have to solve.

– An iterative solver based on Lanczos' algorithm which was slightly modified in order to deal with Schirokauer's maps.

Structured gaussian elimination (1/2)

We propose a refinement which enables us to reduce the dimension of our systems by a factor of 1/5 to 1/10 (instead of 1/3 as it used to be).

We try to minimize the weight increase during a single elimination.

For a potential pivoting entry in a line of weight l and a column of weight c, the estimated variation is equal to

$$(c-1)((l-1)-1)-l=(l-2)(c-2)-2.$$

$\Rightarrow$ One chooses the entry with $(l-2)(c-2)$ minimal as a pivot.

Structured gaussian elimination (2/2)

After each step, $(l-2)(c-2)$ must be recomputed but thanks to an adapted representation of the matrix we can do it incrementally at high speed.

Precisely :

- We store the lines and the columns of the matrix to quickly get l and c.
- For each variable (except those appearing to often), we keep a pointer to the less heavy line which contains it and the corresponding value $(l-2)(c-2)$. We store these datas in a balanced binary tree.

Lanczos algorithm

It mainly consists of the multiplication of the matrix by a vector whose elements are integers of size p.

Therefore :

– Adding few dense columns due to Schirokauer's map implies only few modifications of the algorithm,

– To parallelize it, it's enough to parallelize the product matrix by vector. Which can be done very easily on a shared memory multi-processors computer.
With n processors, we can expect to speed up the computation by a factor of n or, equivalently, to deal with matrixes $\sqrt{n}$ times larger.

– Most of the time is spent in integer additions.

Individual logarithms

To compute the discrete logarithm of any element $y \in \mathbb{F}_p$, we usually looks for two integers A and B of size $p^{\frac{1}{2}}$ such that

$$y = \frac{A}{B} \bmod p, \;\; A \text{ and } B \text{ smooth.}$$

The classical approach consists of testing many such couples (A, B) until one appears to be smooth.

$\Rightarrow$ One can significantly improves the process using sieving techniques.

Individual logarithms using sieving techniques

Once reduced the lattice $\begin{pmatrix} y & p \\ 1 & 0 \end{pmatrix}$, we have integers A_1, B_1, A_2 and B_2 such that

$$\forall(\gamma,\delta) \in \mathbb{Z}^2,\ y = \frac{A_1}{B_1} = \frac{A_2}{B_2} = \frac{\gamma A_1 + \delta A_2}{\gamma B_1 + \delta B_2} \bmod p.$$

Then, the 'sieving by vectors" method applies to find integers γ and δ such that $\gamma A_1 + \delta A_2$ and $\gamma B_1 + \delta B_2$ are both smooth.

Individual logarithms using sieving techniques

Remarks :

- We use primes which split in $\mathbb{Q}_\alpha$ for the sieving but these primes are much larger than the factor base.
- Large primes are allowed on both sides.
- Whenever we need the discrete logarithm of a prime, we split it in prime ideals which can be considered as special-$\mathfrak{q}$ in a new sieving

Consequence : A discrete logarithm computation usually turns out to be a “tree” of sieving.

Results

Two discrete logarithm computations :

- Gaussian Integer method for $\mathbb{F}_{p_{90}}$ where $p_{90} = \lfloor 10^{89}\pi \rfloor + 156137$ ($(p_{90}-1)/2$ prime).
- General Number Field Sieve method for $\mathbb{F}_{p_{100}}$ where $p_{100} = \lfloor 10^{99}\pi \rfloor + 3932$ ($(p_{100}-1)/2$ prime).

Questions :

- What is the logarithm of y_{90}, $y_{90}+1$, $y_{90}+2$ and $y_{90}+3$ where $y_{90} = \lfloor 10^{89} e \rfloor$?
- What is the logarithm of y_{100}, $y_{100}+1$, $y_{100}+2$ and $y_{100}+3$ where $y_{100} = \lfloor 10^{99} e \rfloor$?

$\mathbb{F}_{p_{90}}$ (1/2)

Polynomials :

$$\begin{aligned} f_\beta(X) &= X^2 + 2, \\ f_\alpha(X) &= -248748997517243504330966956058992943413697827X + \\ &\quad 436356663553236108573801834571320046705446681. \end{aligned}$$

$$\Rightarrow \text{Units}(Q_\beta) = \{\pm 1\},\ h_\beta = 1.$$

Smoothness bounds :

$$\begin{aligned} B_\alpha &= 1299709 \text{ (100000 primes)}, \\ B_\beta &= 350377 \text{ (30000 primes)}, \end{aligned} \qquad B = 10^9.$$

Sieving : 6663097 equations, 4 months.

Full-Full	Full-Partial	Partial-Full	Partial-P
83519	235921	1377187	4966470

Linear algebra : Removing useless variables and equations $\Rightarrow$ 976062×674564 matrix.

- Structured Gaussian Elimination : 61136×61036, 5683954 entries, 1 day.
- Lanczos Algorithm : 3 weeks.

$\mathbb{F}_{p_{90}}$ (2/2)

<u>Logarithms of small primes :</u>

$$\log_2(3)=197483009611671477932484091097772889165637441883405533161063698387711654453959032453019844,$$
$$\log_2(5)=149119915642401871166239900822961048984976067993569475849547058535625211933991916749097892,$$
$$\vdots$$

<u>Individual logarithms :</u> 9 hours.

$$1299709^{92} y_{90} = A/B \bmod p,$$

with

$$A = 7 \cdot 167 \cdot 347 \cdot 421 \cdot 1049 \cdot 3067 \cdot 3079 \cdot 3319174077 \cdot 293263 \cdot 38174761 \cdot 82044163 \cdot,$$
$$B = 5^2 \cdot 17 \cdot 1187 \cdot 1877 \cdot 2039 \cdot 3019 \cdot 110863 \cdot 207589 \cdot 2903639 \cdot 5397737 \cdot 1064068253.$$

Then,

$$\log_2(y_{90})=1767138072114216962732048234071620272302057952449914157493844716677918658538374188101093.$$

<u>Total time :</u> about 5 months.

$\mathbb{F}_{p_{100}}$ (1/2)

Polynomials :

$$f_\beta(X) = X^3 + 2,$$
$$f_\alpha(X) = 28905469752538627713929962366361 2X^2 + 34750596397982099950362005023925 0X + 13257068516300237300787244033305 83$$

$$\Rightarrow \begin{cases} \text{2 maps for } \mathbb{Q}(\alpha), \\ \text{Units}(Q_\beta) = \pm < \beta + 1 >, \ h_\beta = 1. \end{cases}$$

Smoothness bounds :

$$\left.\begin{array}{l} B_\alpha = 8960467 \text{ (600000 primes)}, \\ B_\beta = 2750161 \text{ (200000 primes)}, \end{array}\right. \text{ no large prime.}$$

Sieving : 2900000 equations, 8 months.

Linear algebra : Removing useless variables and equations $\Rightarrow$ 2592659×816761 matrix.

- Structured Gaussian Elimination : 172049×171061, 21747682 entries, 1 day.
- Lanczos Algorithm : 3 weeks (quadri-processors DEC alpha 500MHz).

$\mathbb{F}_{p_{100}}$ **(2/2)**

Logarithms of small primes :

$$\begin{aligned}\log_{67}(3) &= 15231699951693893676192894808039900459829546515301\\ &\ 17535489480512018192128533920122634532977309738960,\\ \log_{67}(5) &= 19377553728171823105413539921382679272133888091020\\ &\ 12017702986213046313481150182975740143065448645802,\\ &\ \ \vdots\end{aligned}$$

Individual logarithms : 1 day.

$$8960453^{48} y_{100} = A/B \bmod p,$$

with

$$\begin{aligned}A &= 2^4 \cdot 3361 \cdot 27107 \cdot 3182657 \cdot 4467119 \cdot\\ &\ 11770463 \cdot 32918983 \cdot 6211385489629777,\\ B &= 613 \cdot 11317 \cdot 16427 \cdot 257473 \cdot 1188587 \cdot\\ &\ 95157851 \cdot 124086068723 \cdot 293844018211.\end{aligned}$$

Then,

$$\begin{aligned}\log_{67}(y_{100}) &= 12067222770482298529755135197701403621450790321155\\ &\ 79252791899015195684040750444570690623611190553071.\end{aligned}$$

Total time : about 1 year.

Conclusion

- GNFS algorithm seems to be better than the Gaussian Integer Method for computing discrete logarithms modulo primes over 100 digits.
- Main differences with integer factorization :
  - Choosing polynomials.
  - Choice of parameters.
  - Linear algebra.
  - Individual Logarithms.
- The future ?