Relation collection for the Function Field Sieve

Jérémie Detrey, Pierrick Gaudry and Marion Videau

INRIA / CNRS / Université de Lorraine

ARITH 21 – April 10, 2013

Plan

Context

Setting of the problem

Eratosthenes

Gray codes

Benchmarks, conclusion

Plan

Hard problems in cryptography

Public key cryptography security relies on (supposedly):
Computationally hard problems

Currently **in use**:

- Integer factorization (RSA)
- Discrete logarithm in finite fields (DSA, ElGamal)
- Discrete logarithm in elliptic curves (ECDSA)

Also **widely studied**:

- Discrete log in hyperelliptic curves
- Systems based on error correcting codes
- Lattice-based systems
- Systems based on polynomial systems

Hard problems in cryptography

Public key cryptography security relies on (supposedly):

Computationally hard problems

Currently **in use**:

- Integer factorization (RSA)
- **Discrete logarithm in finite fields** (DSA, ElGamal)
- Discrete logarithm in elliptic curves (ECDSA)

Also **widely studied**:

- Discrete log in hyperelliptic curves
- Systems based on error correcting codes
- Lattice-based systems
- Systems based on polynomial systems

Basic index calculus (after Adleman)

Let $\mathbb{F}_{p^n}$ be a finite field, with small p (think $p = 2$ or 3).

Choose $\varphi(t) \in \mathbb{F}_p[t]$ irreducible of degree n, such that

$$\mathbb{F}_{p^n} = \mathbb{F}_p[t]/\varphi(t).$$

Let $g(t)$ be a generator of $\mathbb{F}_{p^n}^*$, and h any element.

Discrete log problem (DLP): find x such that $h = g^x$.

Collect relations:

- For random z, compute $g(t)^z \mod \varphi(t)$;
- Check if it is B-**smooth**, i.e. it is a product of irreducible factors π_i of degree at most B.
- If yes, $g(t)^z = \prod \pi_i(t)^{e_i}$, and taking the log yield a **linear relation**:

$$z = \sum e_i \log \pi_i(t)$$

Basic index calculus (cont'd)

Linear algebra:

- Put each relation in the row of a matrix, where columns are labelled by the π_i's.
- Get the values of $\log \pi_i$ by linear algebra.
- Having enough relations guarantees that there is a unique solution.

Individual logarithm:

- For random z, compute $h(t)g(t)^z \mod \varphi(t)$;
- Check if it is B-smooth;
- If so, $\log h = -z +$ (logs of known elements).

Analysis: Highly depends on the probability for a polynomial to be B-smooth. Get a **subexponential** complexity $\approx \exp(\sqrt{n})$.

Current situation

The best methods known are variants of the basic index calculus.

They depends on the **type of field**:

- Prime field $\mathbb{F}_p$: Number Field Sieve. Time $\approx \exp(\sqrt[3]{\log p})$.
- Field of small characteristic $\mathbb{F}_{p^n}$: Function Field Sieve. Time $\approx \exp(\sqrt[3]{n})$.
- Medium prime case: also $\approx \exp(\sqrt[3]{n \log p})$.
- Fields of tiny characteristic: Joux's algorithm (2013). Time $\approx \exp(\sqrt[4]{n})$.

Current situation

The best methods known are variants of the basic index calculus.

They depends on the **type of field**:

- Prime field $\mathbb{F}_p$: Number Field Sieve. Time $\approx \exp(\sqrt[3]{\log p})$.
- Field of small characteristic $\mathbb{F}_{p^n}$: **Function Field Sieve.** Time $\approx \exp(\sqrt[3]{n})$.
- Medium prime case: also $\approx \exp(\sqrt[3]{n \log p})$.
- Fields of tiny characteristic: **Joux's algorithm (2013).** Time $\approx \exp(\sqrt[4]{n})$.

Current situation

The best methods known are variants of the basic index calculus.

They depends on the **type of field**:

- Prime field $\mathbb{F}_p$: Number Field Sieve. Time $\approx \exp(\sqrt[3]{\log p})$.
- Field of small characteristic $\mathbb{F}_{p^n}$: **Function Field Sieve.** Time $\approx \exp(\sqrt[3]{n})$.
- Medium prime case: also $\approx \exp(\sqrt[3]{n \log p})$.
- Fields of tiny characteristic: **Joux's algorithm (2013).** Time $\approx \exp(\sqrt[4]{n})$.

Rem. Our paper and our software directly apply to the first phase of the descent in Joux's algorithm when applied to prime degree extensions of $\mathbb{F}_2$.

Plan

Relation collection

Given:

- Base ring $= \mathbb{F}_p[t]$, where $p = 2$ or 3;
- Two bivariate polynomials $f(x)$ and $g(x)$:

$$\begin{aligned} f(x) &= x^6 + f_5(t)x^5 + \cdots + f_1(t)x + f_0(t) \\ g(x) &= x + g_0(t), \end{aligned}$$

  where $\deg f_i(t) = O(1)$ and $\deg g_0(t) =$ large.
- A smoothness bound B.

Looking for:

- A **relation** is a pair $(a(t), b(t))$ of polynomials such that both $f(a/b)b^{\deg_x f}$ and $a(t) - g_0(t)b(t)$ are B-smooth.
- Need millions of them (well... billions).

Rem: Explaining the way the matrix is built requires a bit more theory, but the general idea is close to Adleman's basic algo.

Relation collection – example

For solving DLP in $\mathbb{F}_{2^{1039}}$, one can take

$$f = x^6 + (t^2 + t + 1)x^5 + (t^2 + t)x + (t^{12} + t^{10} + t^8 + t^5 + t^3 + t)$$

$$g = x + (t^{174}+t^{20}+t^{19}+t^{18}+t^{17}+t^{15}+t^{14}+t^{13}+t^{12}+t^{11}+t^8+t^7+t^5+t+1)$$

(This is because $\mathrm{Res}_x(f, g)$ has an irreducible factor of degree 1039.)

Let

$$a(t) = t^{23} + t^{22} + t^{20} + t^{19} + t^{16} + t^{15} + t^{14} + t^{13} + t^{12} + t^{11} + t^7 + t^2 + t,$$

$$b(t) = t^{22} + t^{21} + t^{18} + t^{16} + t^{14} + t^{10} + t^9 + t^8 + t^4 + t^3 + t + 1$$

Then both $a^6 + f_5 a^5 b + f_1 ab^5 + f_0 b^6$ and $a + g_0 b$ have irreducible factors (in t) of degree at most 33.

We write this relation in hexa:

```
d9f886,65471b
  :2,7,d,d,d,b,9d,54f41,77a48b,e88e91,1bf57123,ee2d01bb
  :7,6d,f1,a79,925,52c5,3a90a07,400004d,52811b33,db40a61b,25380517b
```

Relation collection – example (cont'd)

For this example, one need at least one billion of such a,b.

The degrees on both sides are 144 and 196.

The probability that both are 33-smooth is **very low** (one in several million).

Can not be satisfied with a trial and error search.

Sieving is the solution.

Plan

Eratosthenes' sieve

First goal: find all primes up to a certain bound N.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Eratosthenes' sieve

First goal: find all primes up to a certain bound N.

~~0~~	~~1~~	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Mark multiples of 2.

Eratosthenes' sieve

First goal: find all primes up to a certain bound N.

~~0~~	~~1~~	2	3	~~4~~	5	~~6~~	7	~~8~~	9	~~10~~	11	~~12~~	13	~~14~~	15	~~16~~	17	~~18~~	19	~~20~~	21	~~22~~	23	~~24~~

Eratosthenes' sieve

First goal: find all primes up to a certain bound N.

~~0~~	~~1~~	2	3	~~4~~	5	~~6~~	7	~~8~~	9	~~10~~	11	~~12~~	13	~~14~~	15	~~16~~	17	~~18~~	19	~~20~~	21	~~22~~	23	~~24~~

Mark multiples of 3.

Eratosthenes' sieve

First goal: find all primes up to a certain bound N.

~~0~~	~~1~~	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~	11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23	~~24~~

Eratosthenes' sieve

First goal: find all primes up to a certain bound N.

Mark multiples of 5.

Eratosthenes' sieve

First goal: find all primes up to a certain bound N.

~~0~~	~~1~~	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~	11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23	~~24~~

Remaining positions are prime.

Eratosthenes' sieve

Memory requirement. Array of N bits with random access.

Time complexity.

The integer P in the loop takes all prime values up to $\sqrt{N}$.

For each P, we visit $\lfloor N/P \rfloor$ positions.

So the total number of operations is $\sum_{P<\sqrt{N} \text{ prime}} \lfloor N/P \rfloor$, which is essentially

$$N \sum_{P<\sqrt{N} \text{ prime}} \frac{1}{P}.$$

By Mertens' theorem, this gives a cost of $O(N \log \log N)$.

Sieving for factoring integers

Instead of putting a zero in the array, one can keep **further information**.

Depending on the information stored, one can get more or less data on the factorization of the integers, at a cost of **higher memory**.

Variants of Eratosthenes:

- Add one at each sieving step: get number of distinct prime factors.
- With only 2 bits per position, one can get the numbers that contain exactly two distinct primes.

Sieving for factoring integers

Variants of Eratosthenes (cont'd):

- Initialize position n with integer n. When sieving, divide the value by P as long as we can. Keep the divided values. This gives the **full factorization** of all numbers up to N.
- Initialize position n with approximation of $\log n$. When sieving, subtract $\log P$. In the end, positions with a small remaining value are **likely to be smooth** (not exact, due to powers).

Rem. Applies to various sets of inputs:

- Need the property $T[i+P] \equiv T[i] \mod P$, for all P that we want to sieve.
- True if $T[i]$ is any rational fraction in i.
- For a given P, the initial position to mark might be difficult to compute.

2D sieving

In our case, the input is bi-dimensional (indexed by (a, b) pairs).

Various complications:

- Small primes have several hits per row: can sieve row by row (similar to 1D sieving).
- Primes larger than row length hit only a fraction of the rows. Theory of **lattices** to the rescue.
  Need to find an appropriate basis, not really reduced in the classical sense.
- Computation of the initial position can be a bit more difficult.

Sieving polynomials

Things to change when working over $\mathbb{F}_p[t]$ instead of $\mathbb{Z}$:

- Prime numbers replaced by (monic) irreducible polynomials.
- Need conversion $\mathbb{F}_p[t] \longleftrightarrow \mathbb{Z}$, because positions are indexed by polynomials. Usually use **Lex-order**.
- The set of multiples of a prime $p(t)$ is an $\mathbb{F}_p$-**vector space**.

Need to enumerate quickly elements of a vector space:
Gray codes

Plan

Basic Gray code

Binary Gray code of length 3 over $\mathbb{F}_2$:

0	0	0
1	0	0
1	1	0
0	1	0
0	1	1
1	1	1
1	0	1
0	0	1

Only one bit-flip between two lines.

Enumerating an $\mathbb{F}_2$-vector space of dim k with basis $\{e_1, \ldots, e_k\}$:
Add successively vectors corresponding to sequence of bit-flips.

Gray sequence for p-ary Gray codes

- 2-ary sequence:
  $(0, 1, 0, 2, 0, 1, 0, 3, \ldots)$ = 2-adic valuations of $1, 2, 3, 4, 5, \ldots$
- 3-ary sequence:
  $(0, 0, 1, 0, 0, 1, 0, 0, 2, 0, \ldots)$ = 3-adic val. of $1, 2, 3, 4, 5, \ldots$
  = t-adic valuations of polys in $\mathbb{F}_3[t]$ in Lex order.
- p-ary sequence: t-adic val. of polys in Lex order.
  Can be defined recursively by

$$\Delta_0 = (\ ), \qquad \Delta_{i+1} = (\Delta_i, i, \Delta_i, \ldots, i, \Delta_i),$$

with Δ_i repeated p times.

Monic Gray codes

Why? Expressions to test for smoothness are homogeneous. Hence, can force $b(t)$ to be monic. Reduces the search space.

How? Take a basis $\{e_1, \ldots, e_k\}$ that is monic and **echelonized**: $\deg e_i < \deg e_{i+1}$.

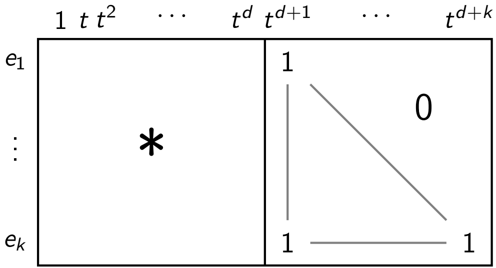
To ensure that the most-significant coeff is one, the recursive definition becomes:

$$\Delta'_0 = (\ ), \qquad \Delta'_{i+1} = (\Delta'_i, i, \Delta_i).$$

E.g. The 3-ary monic Gray sequence is $(0, 1, 0, 0, 2, 0, 0, 1, 0, 0, 1, 0, 0, 3, \ldots)$.

Enumerating in Lex order (non-monic)

Take an "all-one-triangular" basis:



Fact. p-ary Gray code enumerates $\mathrm{Span}(e_1, \ldots, e_k)$ in Lex order.
"Proof": The part $\sum t^j$ of e_i emulates the carry propagation.

*(**Rem**. Yes, we can do it for monic as well.)*

Plan

Benchmark for $\mathbb{F}_{2^{1039}}$

Parameters:

Max. deg. of sieved primes (factor base bound)	25
Max. deg. of large primes (smoothness bound)	33
Threshold degree for starting cofactorization	99

Sieving time, per position:

Step	Cycles/pos	Percentage
Initialize norms	1.10	2.04 %
Sieve by rows	9.73	18.15 %
Fill buckets	31.73	59.21 %
Apply buckets	2.74	5.12 %
Cofactorization	7.43	13.87 %
Total	**53.59**	100.00 %

In that case, proba of being smooth $\approx$ 2e−8. Hence about 1 s/rel.

Benchmark for $\mathbb{F}_{2^{1039}}$

The computation is not finished.

Currently, the matrix is a bit too big, so we do a lot of **oversieving**.

Still no clear idea of the total running time for discrete log in this field.

New record: $\mathbb{F}_{2^{809}}$

Joint work with: R. Barbulescu, C. Bouvier, H. Jeljeli, E. Thomé, P. Zimmermann. `http://eprint.iacr.org/2013/197`

Note: 809 is **prime**. Previsous was 613 (Joux-Lercier, 2005). All recent records (in particular based on the $L(1/4)$ algorithm by Joux) are for composite degree extensions, which are much easier.

Running time:

- Relations: 32M rels collected in 18,000 hours on one core of Intel Core i5-2500.
- Filtering: reduce to a matrix of size 4.46M, with 100 coeffs per row. Negligible time (quality of output is important).
- Linear algebra: 1,300 hours on an Nvidia GTX 680 GPU.
- Individual logs: around 1 hour.

Conclusion

- **Sieving** is very efficient. Many funny complications when switching from integers to polynomials.
- Crossover point between FFS and Joux's new algorithm still to be determined for prime degree extensions.
- Our relation collection implementation can be used for both.
- It is available under LGPL: feel free to play with it!

  `http://cado-nfs.gforge.inria.fr/`

  (In the `ffs/` subdirectory of the git repo.)