Algorithms for integer factorization and discrete logarithms computation

Cyril Bouvier

Institut de Mathématiques de Bordeaux (IMB)
Cyril.Bouvier@u-bordeaux.fr

LFANT Seminar – September, 8th, 2015

Introduction — Cryptography

- Public-key cryptography (or asymmetric cryptography):
  - It is widely used to secure internet connections, credit cards, electronic voting, ...
  - The security of many public-key cryptosystems relies on the supposed difficulty of two mathematical problems: integer factorization and discrete logarithm.
- During my thesis, I studied two algorithms for integer factorization:
  - Elliptic Curve Method (ECM): uses elliptic curves to find small- to medium-size factors of integers.
  - Number Field Sieve algorithm (NFS): best algorithm to completely factor large integers that are free of small factors.
- During my thesis, I studied two algorithms to solve the discrete logarithm problem in finite fields:
  - Number Field Sieve for Discrete Logarithm (NFS-DL) for finite fields of large characteristic ($\mathbb{F}_{p^n}$ with large p and small n).
  - Function Field Sieve (FFS) for finite fields of small characteristic ($\mathbb{F}_{p^n}$ with small p and large n).

Outline of the presentation

- ECM: Galois properties of elliptic curves and ECM-friendly curves

Joint work with J. Bos, R. Barbulescu, T. Kleinjung, and P. Montgomery

- NFS: size optimization in the polynomial selection step

Joint work with S. Bai, A. Kruppa, and P. Zimmermann

- NFS, NFS-DL, FFS: the filtering step

- Conclusion and perspectives

Outline of the presentation

- ECM: Galois properties of elliptic curves and ECM-friendly curves

Joint work with J. Bos, R. Barbulescu, T. Kleinjung, and P. Montgomery

- NFS: size optimization in the polynomial selection step

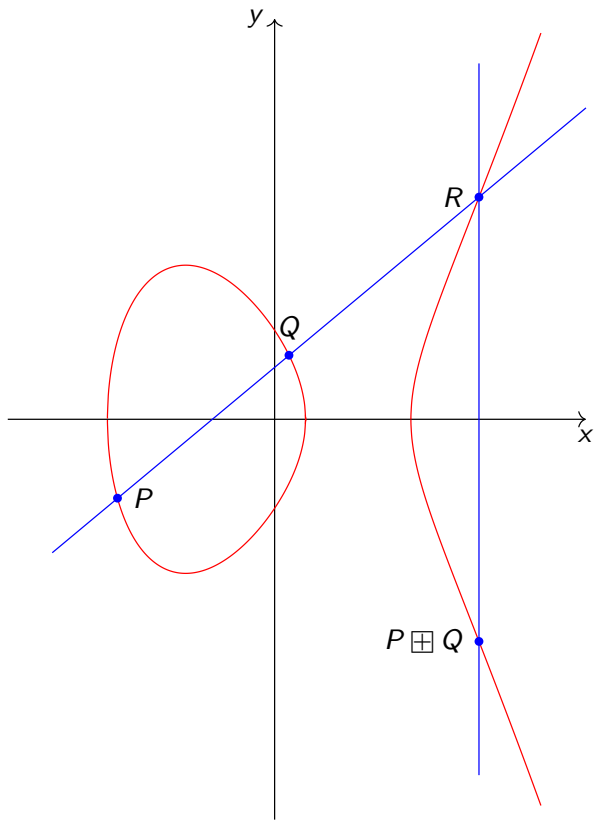
Joint work with S. Bai, A. Kruppa, and P. Zimmermann

- NFS, NFS-DL, FFS: the filtering step

- Conclusion and perspectives

Elliptic curves

- An elliptic curve E over a field K will be denoted by E/K and the point at infinity by $\mathcal{O}$.
- Given $P, Q \in E(K)$, their sum is denoted by $P \boxplus Q$.
- Given $P \in E(K)$ and $k \in \mathbb{N}$, kP is defined by $kP = P \boxplus \cdots \boxplus P$ (k times).
- Let $E/\mathbb{Q}$ be an elliptic curve. The ECM algorithm uses the following facts:
  - for almost all primes p, the curve can be reduced modulo p;
  - the set of points $E(\mathbb{F}_p)$ of the reduced curve is a finite group;
  - the group law is compatible with the reduction.



Elliptic Curve Method (ECM)

- Elliptic Curve Method (ECM): first described by H. Lenstra; best algorithm to find small- to medium-size factors of integers (largest factor found had 83 digits).
- ECM starts by choosing a positive integer B, a curve $E/\mathbb{Q}$ and a point $P \in E(\mathbb{Q})$. Then it computes $Q = sP$, where

$$s = \prod_{\substack{\pi \leqslant B \\ \pi \text{ prime}}} \pi^{\lfloor \log(B)/\log(\pi) \rfloor}$$

  and where the operations of the group law from $E/\mathbb{Q}$ are performed modulo N.
- A factor p of N can be retrieved from Q if $\#E(\mathbb{F}_p)$ is B-powersmooth, *i.e.*, if all prime powers dividing $\#E(\mathbb{F}_p)$ are at most B.
- If a curve fails to find a factor, other curves can be used.
- What curves should be used? All curves are not equivalent. For example, A. Kruppa observed that the Suyama curve $\sigma = 11$ found more factors and that the orders of the reduced curves have a higher average valuation of 2 than other Suyama curves.

Torsion and Galois representations

- Let $E/\mathbb{Q}$ be an elliptic curve and $m \geqslant 2$ be an integer.
- The set of m-torsion points:

$$E(K)[m] = \{\, P \in E(K) \mid mP = \mathcal{O} \,\}.$$

  Here, K is either a field extension of $\mathbb{Q}$ or of a finite field $\mathbb{F}_p$, for a prime p.
- An important theorem: over $\overline{K}$, the algebraic closure of K, if the characteristic of K is zero or coprime with m,

$$E(\overline{K})[m] \simeq \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}.$$

- $\mathbb{Q}(E[m])$: smallest field extension of $\mathbb{Q}$ such that all the m-torsion points are defined. It is a Galois extension.
- The Galois group $\mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q})$ acts on the m-torsion points and can be identified to a subgroup of $\mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$, *via* an injective morphism denoted by ρ_m:

$$\rho_m \colon \mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q}) \hookrightarrow \mathrm{Aut}(E(\overline{\mathbb{Q}})[m]) \simeq \mathrm{Aut}(\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}) \simeq \mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z}).$$

  The image of $\mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q})$ *via* ρ_m will be noted $G(E, m)$.

Main theorem

Theorem

Let $E/\mathbb{Q}$ be an elliptic curve, $m \geqslant 2$ be an integer and T be a subgroup of $\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$. Then,

$$\mathrm{Prob}(E(\mathbb{F}_p)[m] \simeq T) = \frac{\#\{\, g \in G(E,m) \mid \mathrm{Fix}(g) \simeq T \,\}}{\#G(E,m)}.$$

- $\mathrm{Prob}(E(\mathbb{F}_p)[m] \simeq T)$ is defined as the limit of the density of primes p satisfying this property.
- Proof: Chebotarev's density theorem applied to $G(E,m) = \mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q})$.
- Also proved a version where only primes congruent to a given a mod n are considered.

Corollary

Let $E/\mathbb{Q}$ be an elliptic curve and π be a prime number. Then,

$$\mathrm{Prob}(E(\mathbb{F}_p)[\pi] \simeq \mathbb{Z}/\pi\mathbb{Z}) = \frac{\#\{\, g \in G(E,\pi) \mid \det(g - \mathrm{Id}) = 0, g \neq \mathrm{Id} \,\}}{\#G(E,\pi)} \text{ and}$$

$$\mathrm{Prob}(E(\mathbb{F}_p)[\pi] \simeq \mathbb{Z}/\pi\mathbb{Z} \times \mathbb{Z}/\pi\mathbb{Z}) = \frac{1}{\#G(E,\pi)}.$$

Example

π	T	d_1	$\mathrm{Prob}_{\mathrm{th}}(E_1(\mathbb{F}_p)[\pi] \simeq T)$ $\mathrm{Prob}_{\mathrm{exp}}(E_1(\mathbb{F}_p)[\pi] \simeq T)$	d_2	$\mathrm{Prob}_{\mathrm{th}}(E_2(\mathbb{F}_p)[\pi] \simeq T)$ $\mathrm{Prob}_{\mathrm{exp}}(E_2(\mathbb{F}_p)[\pi] \simeq T)$
3	$\mathbb{Z}/3\mathbb{Z} \times \mathbb{Z}/3\mathbb{Z}$	48	$\frac{1}{48} \approx 0.02083$ 0.02082	16	$\frac{1}{16} = 0.06250$ 0.06245
3	$\mathbb{Z}/3\mathbb{Z}$	48	$\frac{20}{48} \approx 0.4167$ 0.4165	16	$\frac{4}{16} = 0.2500$ 0.2501
5	$\mathbb{Z}/5\mathbb{Z} \times \mathbb{Z}/5\mathbb{Z}$	480	$\frac{1}{480} \approx 0.002083$ 0.002091	32	$\frac{1}{32} = 0.03125$ 0.03123
5	$\mathbb{Z}/5\mathbb{Z}$	480	$\frac{114}{480} \approx 0.2375$ 0.2373	32	$\frac{10}{32} = 0.3125$ 0.3125

- $E_1/\mathbb{Q}\colon y^2 = x^3 + 5x + 7$ and $E_2/\mathbb{Q}\colon y^2 = x^3 - 11x + 14$.
- Theoretical values come from the previous corollary.
- For experimental values, all primes below 2^{25} were considered.
- Columns d_1 and d_2 indicate the size of $G(E_1, \pi)$ and $G(E_2, \pi)$, respectively.

Divisibility by prime powers and average valuation

- Next goal is to compute $\mathrm{Prob}(\pi^k \mid \#E(\mathbb{F}_p))$ and the average valuation defined by

$$\bar{v}_\pi = \sum_{k \geqslant 1} k \,\mathrm{Prob}(v_\pi(\#E(\mathbb{F}_p)) = k).$$

- For an elliptic curve $E/\mathbb{Q}$, a prime π and a positive integer k, $I(E, \pi, k)$ is defined by

$$I(E, \pi, k) = [\mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z}) : G(E, \pi^k)].$$

- Theorem from Serre: for all elliptic curves $E/\mathbb{Q}$ without CM and for all primes π, the sequence $(I(E, \pi, k))_{k \geqslant 1}$ is bounded and non-decreasing when k goes to infinity.

Theorem

Let $E/\mathbb{Q}$ be an elliptic curve, π be a prime and n be a positive integer such that

$$\forall k \geqslant n, \quad I(E, \pi, k) = I(E, \pi, n).$$

Then, the probabilities $\mathrm{Prob}(\pi^k \mid \#E(\mathbb{F}_p))$, for all $k \geqslant 1$, and the average valuation $\bar{v}_\pi$ can be computed as linear combinations of the probabilities $\mathrm{Prob}(E(\mathbb{F}_p)[\pi^t] \simeq \mathbb{Z}/\pi^i\mathbb{Z} \times \mathbb{Z}/\pi^j\mathbb{Z})$, with $i \leqslant j \leqslant t \leqslant n$.

Example

π	$n(E_1,\pi)$	$\bar{v}_{\pi,\mathrm{th}}$ $\bar{v}_{\pi,\mathrm{exp}}$	$n(E_3,\pi)$	$\bar{v}_{\pi,\mathrm{th}}$ $\bar{v}_{\pi,\mathrm{exp}}$
2	1	$\frac{14}{9} \approx 1.556$ 1.555	3	$\frac{895}{576} \approx 1.554$ 1.554
3	1	$\frac{87}{128} \approx 0.680$ 0.679	1	$\frac{39}{32} \approx 1.219$ 1.218
5	1	$\frac{695}{2304} \approx 0.302$ 0.301	1	$\frac{155}{192} \approx 0.807$ 0.807

- $E_1/\mathbb{Q}\colon y^2 = x^3 + 5x + 7$ and $E_3/\mathbb{Q}\colon y^2 = x^3 - 10875x + 526250$.
- $n(E,\pi)$ is the smallest integer n such that $I(E,\pi,k) = I(E,\pi,n)$, for $k \geqslant n$.
- Values of $n(E_1,\pi)$ are proven, values of $n(E_3,\pi)$ are conjectured.
- Theoretical values come from the previous theorem used with $n = n(E_i,\pi)$.
- For experimental values, all primes below 2^{25} were considered.

Applications

- Apply previous results to try to identify families of elliptic curves suitable for ECM.
- The goal is to find infinite families of curves with large $\text{Prob}(\pi^k \mid \#E(\mathbb{F}_p))$, for small primes π and small prime powers π^k.
- Example: the Suyama-11 subfamily:
  - Proved that for the Suyama curve with $\sigma = 11$ the average valuation of 2 is $11/3$, instead of $10/3$ for generic Suyama curves.
  - This difference is due to a smaller Galois group for the 4-torsion that leads to better probabilities of divisibility by powers of 2 for primes congruent to 1 modulo 4.
  - Found an infinite family of Suyama curves with the same properties.
- Obtained similar results for another subfamily of Suyama curves and for subfamilies of twisted Edwards curves.

Outline of the presentation

- ECM: Galois properties of elliptic curves and ECM-friendly curves

Joint work with J. Bos, R. Barbulescu, T. Kleinjung, and P. Montgomery

- NFS: size optimization in the polynomial selection step

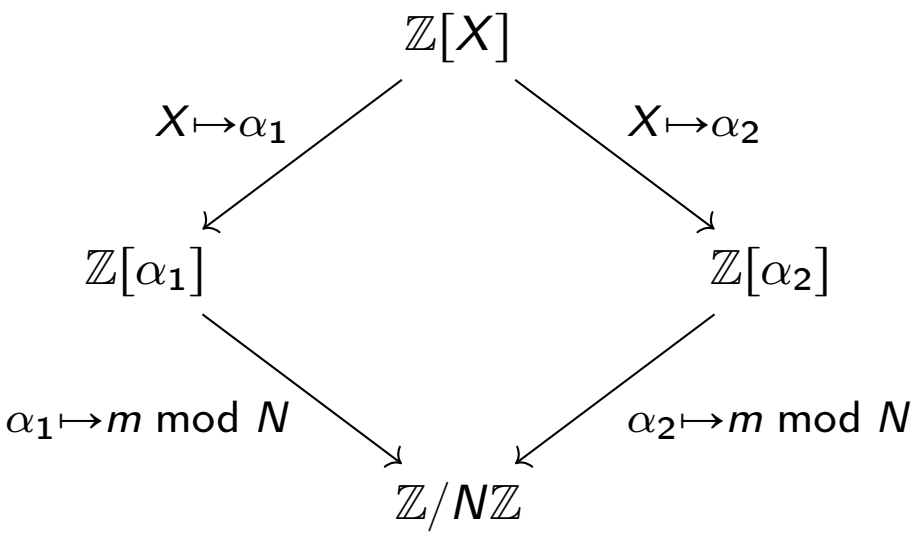
Joint work with S. Bai, A. Kruppa, and P. Zimmermann

- NFS, NFS-DL, FFS: the filtering step

- Conclusion and perspectives

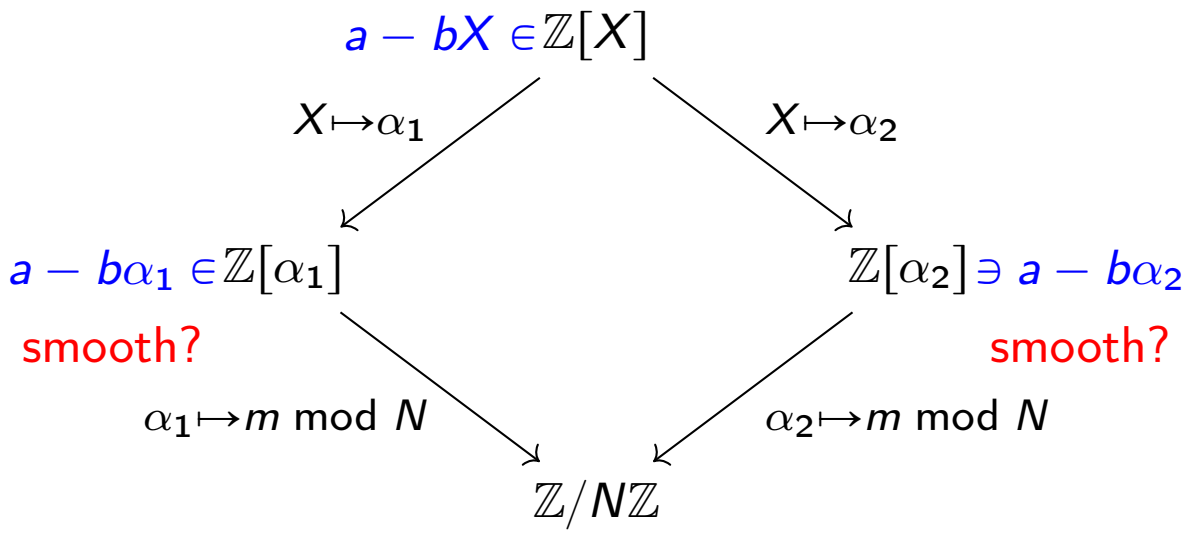
Number Field Sieve (NFS)

- Number Field Sieve (NFS): best algorithm to factor integers that are free of small factors.
- Looks for two integers x, y such that $x^2 \equiv y^2 \pmod{N}$. If $x \not\equiv \pm y \pmod{N}$, then a factor of N is found by computing $\gcd(x \pm y, N)$.
- The equality of squares is obtained by constructing squares in two number fields defined by two integer polynomials f_1 and f_2 with a common root m modulo N.



Number Field Sieve (NFS)

- Number Field Sieve (NFS): best algorithm to factor integers that are free of small factors.
- Looks for two integers x, y such that $x^2 \equiv y^2 \pmod{N}$. If $x \not\equiv \pm y \pmod{N}$, then a factor of N is found by computing $\gcd(x \pm y, N)$.
- The equality of squares is obtained by constructing squares in two number fields defined by two integer polynomials f_1 and f_2 with a common root m modulo N.



- Relation: (a, b) pair such that $f_i(a/b)b^{\deg(f_i)}$ is smooth for $i \in \{1, 2\}$.
- Combine relations to produce squares on both sides. It boils down to computing the kernel of a matrix over $\mathbb{F}_2$.

Steps of the NFS algorithm

- Polynomial selection: compute the pair of polynomials to build the two number fields.
- Relations collection (a.k.a., sieving): use sieving methods to compute many relations.
- Filtering: build the matrix from the relations.
- Linear algebra: compute the kernel of the matrix built by the filtering step.
- Square root: for each vector in the kernel, compute square roots in each number field to obtain x and y.

Polynomial selection step in NFS

- What conditions on the pair of polynomials (f_1, f_2) for polynomial selection in NFS?
  - f_1 and f_2 are primitive integer polynomials;
  - f_1 and f_2 are irreducible over $\mathbb{Q}$;
  - f_1 and f_2 are coprime over $\mathbb{Q}$;
  - f_1 and f_2 have a common root modulo N.
- Linear polynomial selection: $\deg(f_1) = 1$ and side 1 is the rational side.
  In practice $d = \deg(f_2) \in \{ 4, 5, 6 \}$ depending on the size of N.
- From a valid pair of polynomials (f_1, f_2) one can construct other valid pairs
  - by translation by any integer k:

$$\tilde{f_1}(x) = f_1(x + k) \qquad \text{and} \qquad \tilde{f_2}(x) = f_2(x + k);$$

  - by rotation by an integer polynomial $R \in \mathbb{Z}[X]$:

$$\tilde{f_1}(x) = f_1(x) \qquad \text{and} \qquad \tilde{f_2}(x) = f_2(x) + R(x)f_1(x),$$

    as long as $\tilde{f_2}$ is still an irreducible polynomial of degree d.

Size optimization problem

Given (f_1, f_2), find the translation k and rotation R that produce the "best" pair $(\tilde{f_1}, \tilde{f_2})$.

- The norm of a polynomial f is noted $\|f\|_2$.
- The norm used is not the canonical L^2 norm and takes into account the fact that the polynomials are going to be used to compute relations in the next step of the NFS algorithm.
- Linear polynomial selection: the norm of $\tilde{f_1}$ is not taken into account.
- The size optimization problem becomes:
  find the translation and the rotation that minimize $\|\tilde{f_2}\|_2$.

Local descent and initial translations

- State of the art as implemented in `cado-nfs` software: local descent algorithm.
- Works fine for $d = 4$ and $d = 5$. For larger degrees, often stuck in local minima close to the starting points.
- Apply some initial translations before calling the local descent algorithm to increase the number of starting points and to avoid being stuck in local minima too far away from the global minimum.
- How do we choose the initial translations? Choose integer approximations of roots of $\tilde{a}_{d-3}$, where the $\tilde{a}_i$'s are polynomials in k defined by

$$f_2(X + k) = \sum_{i=0}^{d} \tilde{a}_i(k) X^i.$$

For example, for $d = 6$ and $f_2 = a_6 X^6 + a_5 X^5 + \cdots + a_1 X + a_0$,

$$\tilde{a}_3(k) = 20 a_6 k^3 + 10 a_5 k^2 + 4 a_4 k + a_3.$$

New method: using LLL before local descent

- New idea: Use the LLL algorithm to search for short vectors in the lattice spanned by

$$\begin{array}{c} \begin{array}{ccccccc} X^6 & X^5 & X^4 & X^3 & X^2 & X & 1 \end{array} \\ \begin{pmatrix} a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ 0 & 0 & m_2 & -m_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_2 & -m_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_2 & -m_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & m_2 & -m_1 \end{pmatrix} \end{array} \begin{array}{r} f_2 \\ X^3 f_1 \\ X^2 f_1 \\ X f_1 \\ f_1 \end{array}$$

  where $f_1 = m_2 X - m_1$ and $f_2 = a_d X^d + \cdots + a_0$ (example for $d = 6$).

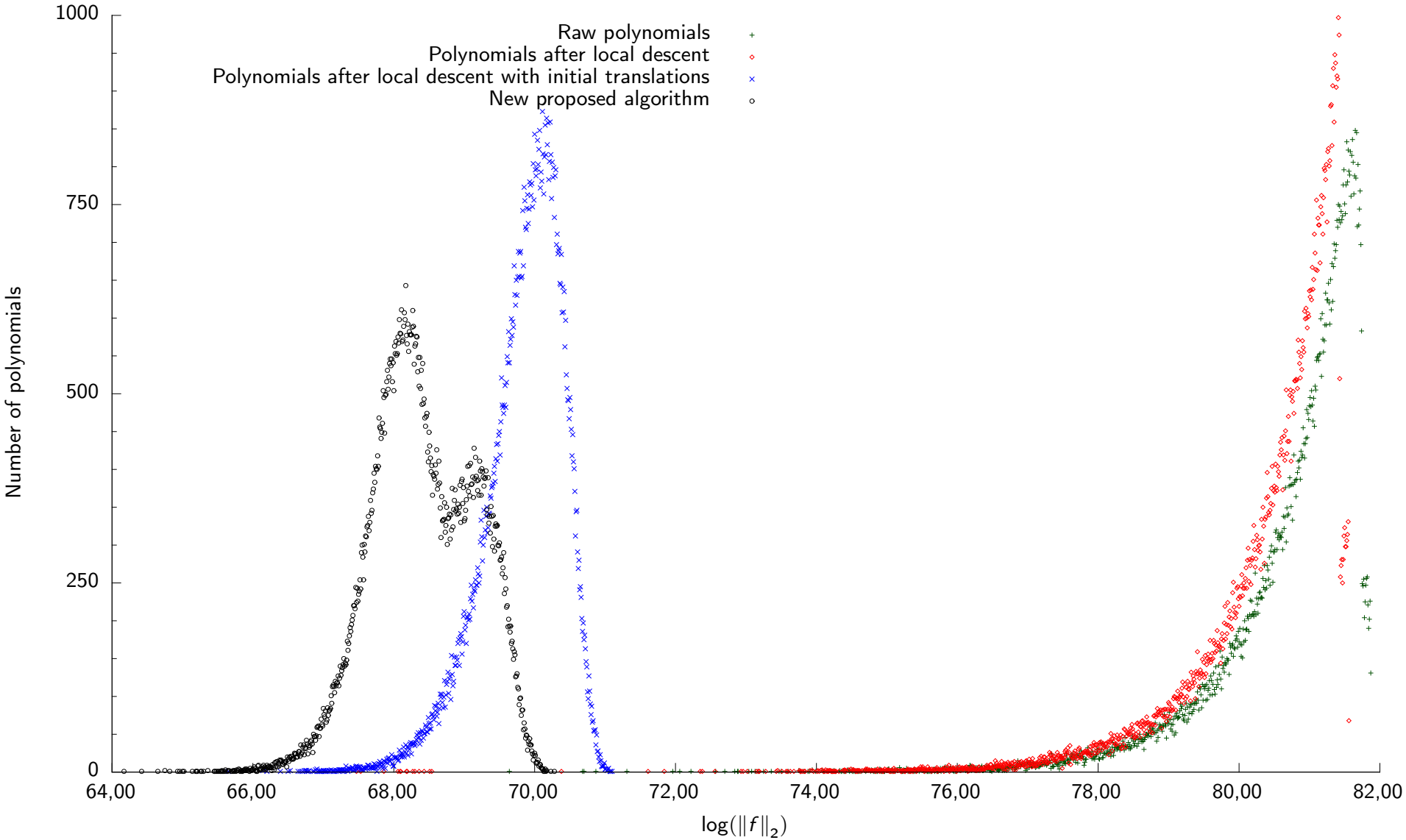
- A vector of this lattice corresponds to a polynomial of the form $cf_2 + Rf_1$, with $c \in \mathbb{Z}$ and R an integer polynomial.

- New degree of freedom: will output polynomial pair $(\tilde{f}_1, \tilde{f}_2)$ such that $\mathrm{Res}(\tilde{f}_1, \tilde{f}_2) = cN$. With previous methods, $\mathrm{Res}(\tilde{f}_1, \tilde{f}_2) = \mathrm{Res}(f_1, f_2) = f_2(m_1/m_2)m_2^d = N$.

- This new method is used before the local descent algorithm and after the computation of the initial translations.

- New initial translations that take advantage of this new degree of freedom can be computed.

Results – RSA-768 ($d = 6$)



- The best polynomial pair found with this new method would have reduced the time spent in the sieving step by 5 %.

Results – RSA-896 ($d = 6$)

- RSA-896: not yet factored but a large amount of computations for polynomial selection has been done.

#	1	2	3	4	5	6	7	8	9	10
Raw polynomial	98.28	98.11	96.89	98.00	97.84	98.53	97.18	98.37	96.97	96.63
Previous algorithm	82.88	82.74	82.30	82.03	82.37	83.33	82.12	79.36	83.79	82.45
New algorithm	80.53	80.16	79.33	79.75	79.78	79.83	80.04	80.72	79.92	79.38

Table: $\log(\left\|f_{2,i}\right\|_2)$ for $i \in [1, 10]$ from 10 polynomial pairs for RSA-896.

- The log of the norm is smaller by 2.40 on average (79.94 against 82.34) and always smaller, with the exception of #8.

Results – RSA-1024 ($d = 6$)

- Applied the new algorithm on a polynomial pair previously published: it brought down the log of the norm from 100.02 to 94.91.
- Generated other raw polynomials to find the current best polynomial pair:

$$
\begin{aligned}
f_1 &= 2387707688882042760409842 1X \\
&\quad - 3332563300755253307596506559178566254508204949738 \\
f_2 &= 492999999999872400X^6 \\
&\quad + 19986130996295579328005858 00X^5 \\
&\quad + 14776348389733418096949161617663667X^4 \\
&\quad - 17369563296702789247942467572798015432351 6X^3 \\
&\quad - 58245139481832624147323198441400656783348781896 2X^2 \\
&\quad + 29609635772301623248273428019688928620985521680508271 56X \\
&\quad - 20364558899868538420816205898474403074641452593893682451 54065
\end{aligned}
$$

  with $\log(\|f_2\|_2) = 91.90$.

- This polynomial pair was found after around 1000 core-hours of computation. A real computational effort for RSA-1024 should required a few thousand core-years.

Outline of the presentation

- ECM: Galois properties of elliptic curves and ECM-friendly curves

Joint work with J. Bos, R. Barbulescu, T. Kleinjung, and P. Montgomery

- NFS: size optimization in the polynomial selection step

Joint work with S. Bai, A. Kruppa, and P. Zimmermann

- NFS, NFS-DL, FFS: the filtering step

- Conclusion and perspectives

Filtering step of NFS, NFS-DL and FFS

- Filtering step: common to NFS, NFS-DL and FFS algorithms. Also common to other factoring algorithms and to other discrete logarithm algorithms.
- In these algorithms, a relation is the decomposition of the image of one element in two different factor bases.
- The set of relations is seen as a matrix where a row corresponds to a relation and a column to an element of one of the two factor bases.

Factorization:

- Combine relations in order to generate squares.
- Linear algebra problem:
  - Matrix over $\mathbb{F}_2$
  - Left kernel

Discrete logarithm:

- A relation is an equality between (virtual) logarithms
- Linear algebra problem:
  - Matrix over $\mathbb{F}_\ell$
  - Right kernel

Filtering step of NFS, NFS-DL and FFS

- Beginning of the filtering step: the matrix is very large but is very sparse (around 20 to 30 non-zero coefficients per row).
- Goal of the filtering step: to produce a matrix as small and as sparse as possible from the given relations in order to decrease the time spent in the linear algebra step.
- Example: data from the factorization of RSA-768:
  - input: 48 billion rows and 35 billion columns.
  - output: 193 million rows and columns with 144 non-zero coefficients per row in average.
- Excess: difference between the number of rows and the number of columns of the matrix.
- Stages of the filtering step:
  - singleton removal: remove useless rows and columns;
  - clique removal: use the excess to reduce the size of the matrix;
  - merge: beginning of a Gaussian elimination.

Singleton removal

- Weight: the weight of a row (resp. column) is the number of non-zero coefficients in this row (resp. column). The total weight of the matrix is the total number of non-zero coefficients.
- A singleton is a column of weight 1.
- Removing a singleton is the removal of the column and of the row corresponding to the non-zero coefficient.
- Implementation remarks: only need to know if a coefficient is non-zero or not, not the actual value; in the the discrete logarithm case, the deleted rows must be saved.
- Example:

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

Singleton removal

- Weight: the weight of a row (resp. column) is the number of non-zero coefficients in this row (resp. column). The total weight of the matrix is the total number of non-zero coefficients.
- A singleton is a column of weight 1.
- Removing a singleton is the removal of the column and of the row corresponding to the non-zero coefficient.
- Implementation remarks: only need to know if a coefficient is non-zero or not, not the actual value; in the the discrete logarithm case, the deleted rows must be saved.
- Example:

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

Singleton removal

- Weight: the weight of a row (resp. column) is the number of non-zero coefficients in this row (resp. column). The total weight of the matrix is the total number of non-zero coefficients.
- A singleton is a column of weight 1.
- Removing a singleton is the removal of the column and of the row corresponding to the non-zero coefficient.
- Implementation remarks: only need to know if a coefficient is non-zero or not, not the actual value; in the the discrete logarithm case, the deleted rows must be saved.
- Example:

$$\left(\begin{array}{c|ccccc} 0 & 1 & 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

Singleton removal

- Weight: the weight of a row (resp. column) is the number of non-zero coefficients in this row (resp. column). The total weight of the matrix is the total number of non-zero coefficients.
- A singleton is a column of weight 1.
- Removing a singleton is the removal of the column and of the row corresponding to the non-zero coefficient.
- Implementation remarks: only need to know if a coefficient is non-zero or not, not the actual value; in the the discrete logarithm case, the deleted rows must be saved.
- Example:

$$\left(\begin{array}{c|cccccc} 0 & 1 & 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

Singleton removal

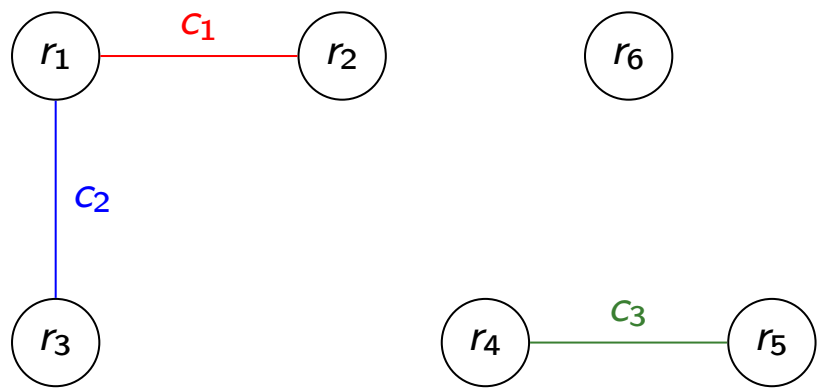
- Weight: the weight of a row (resp. column) is the number of non-zero coefficients in this row (resp. column). The total weight of the matrix is the total number of non-zero coefficients.
- A singleton is a column of weight 1.
- Removing a singleton is the removal of the column and of the row corresponding to the non-zero coefficient.
- Implementation remarks: only need to know if a coefficient is non-zero or not, not the actual value; in the the discrete logarithm case, the deleted rows must be saved.
- Example:

$$\left(\begin{array}{cc|cccc} \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{1} \\ \cancel{1} & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{0} & \cancel{1} \\ \hline 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

Clique removal

- While the excess is larger that what is needed, it is possible to remove some rows.
- If a row containing a column of weight 2 is removed, this column becomes a singleton and can be removed.
- A clique is a connected component of the graph where the nodes are the rows and the edges are the columns of weight 2.
- Example:

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



- When removing a clique, one more row than column is removed, so the excess is reduced by 1.

Merge

- Merge is the beginning of a Gaussian elimination: combinations of rows are performed to create singletons that are then removed.
- Singleton removal and clique removal reduce the size and the total weight of the matrix. Merge reduces the size of the matrix but increases the total weight of the matrix.
- Merge is performed until a given average weight per row is reached.
- In merge, the values of the non-zero coefficients matter. So there are some differences between the factorization and discrete logarithm contexts.
- Merge is the last step of the filtering step. The matrix returned by merge should be as sparse and as small as possible.

Weight functions for clique removal

- During clique removal, one can choose which cliques are removed.
- How to choose? What choice of cliques, done during clique removal, produces the smallest and sparsest matrix at the end of merge?
- Used weight functions to determine the heaviest cliques that should be removed. The weight of a clique depends on the number of rows in the clique and of the weight of the columns appearing in a row of the clique.
- Proposed 31 weight functions and tested them on data coming from actual factorization and discrete logarithm computations.
- Example of weight functions:
  - Cavallar's weight function (`Msieve`): add 1 per row and $1/2^w$ per column appearing in a row of the clique, where w is the weight of the column.
  - `GGNFS`: add 1 per row and 1 per column in a row of the clique.
  - `cado-nfs` 1.1: add 1 per row in the clique.

Experiments

- To have a fair comparison between the 31 weight functions, they were all implemented in `cado-nfs`.
- All the weight functions were benchmarked on 8 data sets: 3 coming from the factorization context (RSA-155, B200 and RSA-704) and 5 coming from the discrete logarithm context (computations in $\mathbb{F}_{2^{619}}$, $\mathbb{F}_{2^{809}}$ and $\mathbb{F}_{2^{1039}}$ with FFS and in two prime fields of size 155 digits and 180 digits with NFS-DL).
- Input: set of unique relations, the target excess and the target average number of non-zero coefficients per row.
- Output: the matrix after merge.
- How to compare the final matrices? To a first approximation, the complexity of the linear algebra step is the product of the size of the matrix by its total weight. This is the value used to compare the different weight functions.

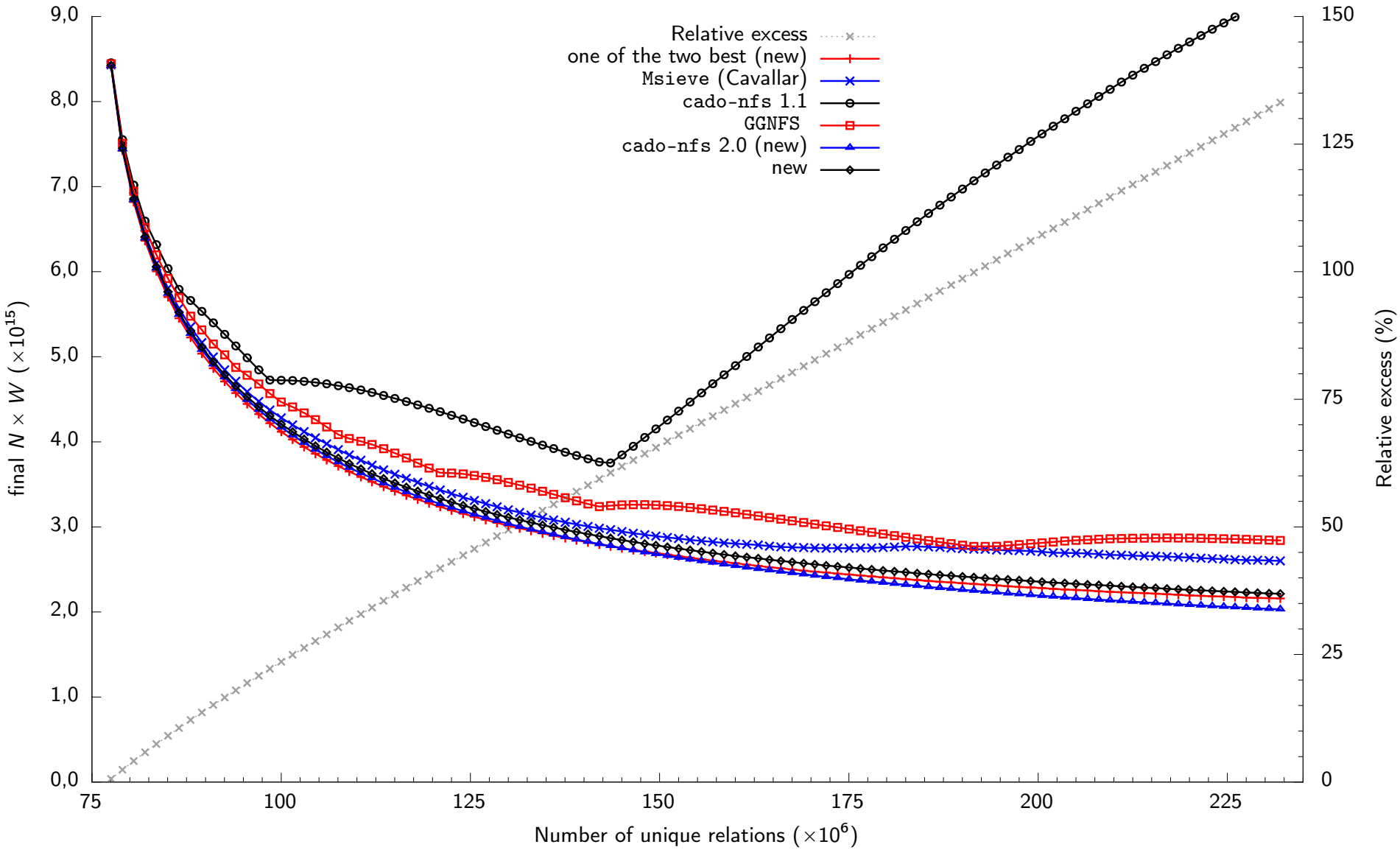
Results

	After clique removal	At the end of the filtering step		
	N	N	$N \times W$	
`cado-nfs` 2.0 (new)	188 580 425	65 138 845	4.24×10^{17}	
`Msieve`	182 939 672	67 603 362	4.57×10^{17}	+7.71 %
`GGNFS`	197 703 703	74 570 015	5.56×10^{17}	+31.05 %
`cado-nfs` 1.1	203 255 785	78 239 129	6.12×10^{17}	+44.27 %

Table: Partial results for the experiment with the data of the discrete logarithm computation in $\mathbb{F}_{2^{1039}}$

- Found two new weight functions that outperformed the others in all experiments.
- The best weight functions after clique removal are not the best at the end of the filtering step.
- The best weight functions are the ones that have few or no contribution from the number of rows in the clique.
- The larger the initial excess, the larger the differences between the weight functions.

Results — Influence of the excess



Outline of the presentation

- ECM: Galois properties of elliptic curves and ECM-friendly curves

Joint work with J. Bos, R. Barbulescu, T. Kleinjung, and P. Montgomery

- NFS: size optimization in the polynomial selection step

Joint work with S. Bai, A. Kruppa, and P. Zimmermann

- NFS, NFS-DL, FFS: the filtering step

- Conclusion and perspectives

Conclusion

- Galois properties of elliptic curves: probabilities of divisibility by prime powers and families of elliptic curves better suited for ECM.
- Polynomial selection step of the NFS algorithm: new method for the size optimization problem that produced better polynomial pairs.
- Filtering step of the NFS, NFS-DL and FFS algorithms: new weight functions for clique removal that produced smaller matrices.
- Software: contributed to GMP-ECM and `cado-nfs`.
- Other works not presented:
  - Record computations of discrete logarithms in finite fields with NFS-DL and FFS.
  - "Division-Free Binary-to-Decimal Conversion".