We Are on the Same Side
Alternative Sieving Strategies for the Number Field Sieve

Ambroise Fleury

Sorbonne Université, LIP6, Paris

February 15, 2024 - Séminaire LORIA

RSA Cryptosystem

Private key

- Used for decryption
- Generated from two random prime numbers p and q

Public key

- Used for encryption
- Product $N = pq$

Factorization

- RSA security is linked to the hardness of integer factorization
- Finding p and q from N breaks RSA

Generic factorization method

Finding a square

- $x^2 = y^2 \mod N$
- $x \neq y \mod N$

Then...

- $N = x^2 - y^2 \mod N$
- $N = (x + y)(x - y) \mod N$
- $gcd(x \pm y, N)$ gives a factor of N

Finding a congruence of squares?

Dixon's factorization method

Build a square

- Generate many y_i such that
  - $y_i = x_i^2 \mod N$
  - y_i is smooth (=only small divisors)
  - it is called a *relation*
- Build $Y^2 \mod N$ as a product of y_i's

1. Relation collection

- Generate many y_i
- Find many relations

2. Linear algebra

- Combine the relations
- $Y^2 = X^2 \mod N$

From factoring a large number...
...to factoring many small numbers

Relations

What relations look like

factor base	**2**	**3**	**5**	**7**	**11**	**13**	**17**
6468	2^2	3		7^2	11		
10210200	2^3	3	5^2	7	11	13	17
1449175			5^2	7^3		13^2	
79560	2^3	3^2	5			13	17
4004	2^2			7	11	13	
175032	2^3	3^2			11	13	17

Next step is to combine them into a square

How? Combine lines to get even exponents

CADO-NFS

- Implementation of the NFS
- Open source : https://gitlab.inria.fr/cado-nfs/cado-nfs
- Can also compute discrete logarithms
- 2019 : Factorization record RSA-240 (240 digits)
- 2020 : Factorization record RSA-250 (current record)
- Computing time is dominated by relation collection

	relation collection	linear alebra
RSA-240	800 CPU years	83 CPU years
RSA-250	2450 CPU years	250 CPU years

Relations in the NFS

Two sides

- Pairs (a, b) of coprime and "small" integers
- Two polynomials $F_i(a, b) = f_i(a/b)b^d$
- We call norms the evaluation of a polynomial with a pair (a, b)
  - $norm_0 = F_0(a, b)$
  - $norm_1 = F_1(a, b)$

Chosen f polynomials for RSA-250 record

$$
\begin{aligned}
f_0 &= 185112968818638292881913X \\
&\quad - 3256571715934047438664355774734330386901 \\
f_1 &= 86130508464000X^6 \\
&\quad - 81583513076429048837733781438376984122961112000 \\
&\quad - 66689953322631501408X^5 \\
&\quad - 1721614429538740120011760034829385792019395X \\
&\quad - 52733221034966333966198X^4 \\
&\quad - 31136272536132022651269074205506483260X^2 \\
&\quad + 46262124564021437136744523465879X^3
\end{aligned}
$$

Relation collection

$$
\begin{tikzcd}
 & (a, b) \arrow[dl, "\text{Rational side}"] \arrow[dr, "\text{Algebraic side}"'] & \\
\begin{array}{c} norm_0 = F_0(a, b) \\ \text{smooth?} \end{array} \arrow[dr] & & \begin{array}{c} norm_1 = F_1(a, b) \\ \text{smooth?} \end{array} \arrow[dl] \\
 & \text{Relation} &
\end{tikzcd}
$$

Special-$\mathfrak{q}$'s

Force a specific factor $\mathfrak{q}$ on one side

- Pick a side (algebraic)
- Pick a special-$\mathfrak{q}$ (prime or composite)
- Get (a, b) pairs from the special-$\mathfrak{q}$
- Factor norms!

Factoring norms

2 methods :

- Sieving to find small and medium factors
- Elliptic-curve factorization (ECM) to find large factors

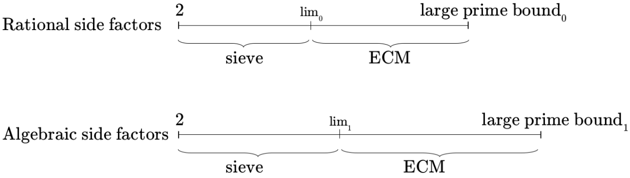
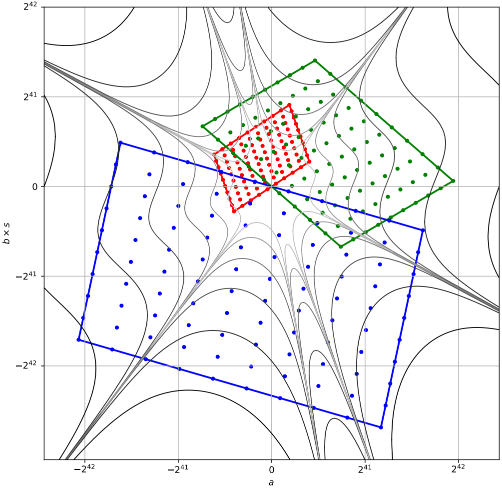


Figure: Method used to recover factors of different sizes

Step 1 : sieving

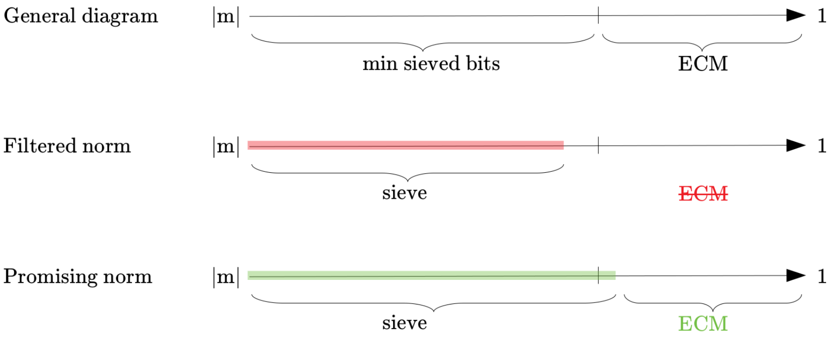
- special-$\mathfrak{q}$ sieving on one specific side (algebraic)
- Regular sieving on the other side (rational)



Step 2 : filtering

Keep only promising pairs

- Sieving factored enough for both norms
- Non-factored part is below a certain bound
- More likely to give a relation



Step 3 : ECM

Promising bound

If the bound deciding wether or not a pair is sent to ECM is...

- Too high
  - Many pairs of low quality
  - Too much time in ECM
- Too low
  - Few pairs of high quality will give too few relations
  - Additional sieving needed

Trying to improve the relation collection in CADO-NFS

Goal : almost as many promising pairs at a much lower cost

Small sieve

Subroutine of CADO-NFS sieving finding small primes ($< 2^{17}$)

- Small factors are worth few bits
- Not decisive on promising pairs

Remove small sieve?

How to find smooth parts of integers [Bernstein 2004]

Input

- Integers $n_0, \ldots, n_i$
- Factor base $P = 2 \times 3 \times \cdots \times p_k$

Output

- Smooth parts of each n_i, meaning the product of factors from the base found in each integer

Complexity

- P is b bits
- $O(b(\log b)^{2+O(1)})$

Batch factoring

- P is the product of factors from the factor base
- Find factors from P in all n's

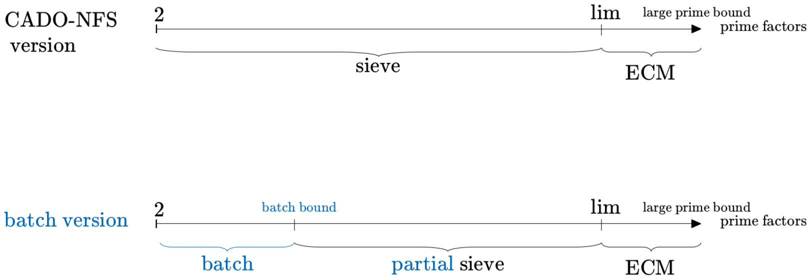
$$\begin{tikzcd}
 & & & P \bmod n_0 n_1 n_2 n_3 \arrow[dll] \arrow[drr] & & & \\
 & P \bmod n_0 n_1 \arrow[dl] \arrow[dr] & & & & P \bmod n_2 n_3 \arrow[dl] \arrow[dr] & \\
P \bmod n_0 \arrow[d] & & P \bmod n_1 \arrow[d] & & P \bmod n_2 \arrow[d] & & P \bmod n_3 \arrow[d] \\
\text{gcd with } n_0 & & \text{gcd with } n_1 & & \text{gcd with } n_2 & & \text{gcd with } n_3
\end{tikzcd}$$

Hybrid strategy

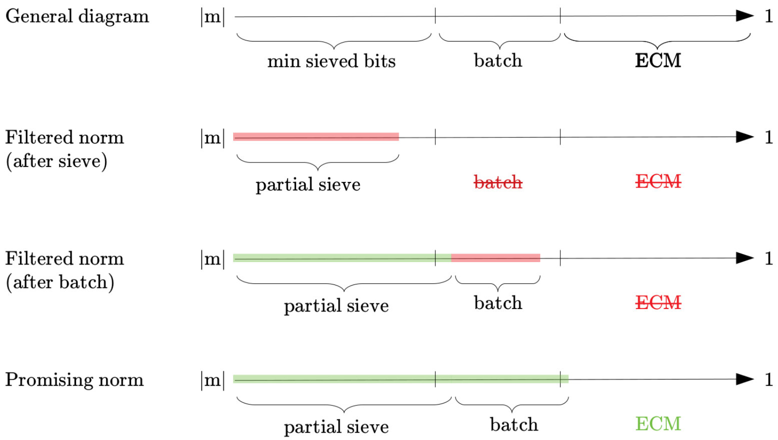
Pick an intermediate "batch promising" bound larger than the "ECM promising" bound, then :

1. Sieve only on medium primes
2. Remove non-promising pairs
3. Get small factors using batch factoring
4. Remove non-promising pairs
5. Get large factors using ECM
6. Relations!

Method for each prime factors interval

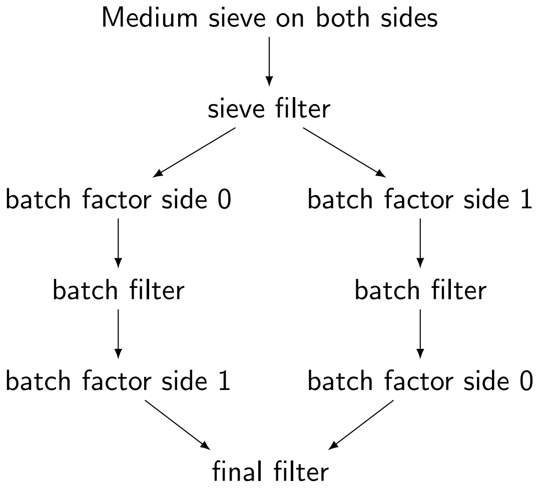


Path to ECM



General diagram
|m|
1
min sieved bits
batch
ECM
Filtered norm (after sieve)
|m|
1
partial sieve
~~batch~~
~~ECM~~
Filtered norm (after batch)
|m|
1
partial sieve
batch
~~ECM~~
Promising norm
|m|
1
partial sieve
batch
ECM

Batch factoring order



RSA-250 relations

- Around 8.4e9 relations were found
- 786 GB gzipped, 1.5 TB uncompressed

Average norm size

- 152 bits on the rational side
- 285 bits on the algebraic side

Example of an actual relation

```
308756823364,858059:
80f,bcd79,2605774d,2dadd6bb,41647363,c29c8ab9:
2,2,3,3,b,13,13,1d,53,6c5,eb9,3afd,33b5cd,2d8f009,
2439f085,3b9add75,1b0218b0d,19daa7f693,1cdbf87c21
```

- $(a, b) = (308756823364, 858059)$
- Rational norm factors :
  $2063, 773497, 637892429, 766367419, 1097102179, 3265039033$
- Algebraic norm factors :
  $2, 2, 3, 3, 11, 19, 19, 29, 83, 1733, 3769, 15101, 3388877,$
  $47771657, 607776901, 1000004981, 7249955597, 111042623123,$
  123949579297

Implementation in CADO-NFS

Parameters introduced

- `batch_first_side` : batch on this side first
- $\texttt{mfbb}_{[0|1]}$ is the bound for sieve survivors
- $\texttt{sbmp}_{[0|1]}$ is the biggest prime in the batch factor base

RSA-250's relations

- Data to target a specific number of relations
- Allow us to pick parameters
- Benchmark baseline

Benchmarks

- Massively parallel
- Pick a (random) special-$\mathfrak{q}$ range
- Sampled sieved regions
- Compare hybrid and regular version
- time vs #relations

Results

Results for a few example sieving areas picked randomly
Multiple values of `sbmp`

Example A, with `mfbb0` = 89 bits and `mfbb1` = 137 bits

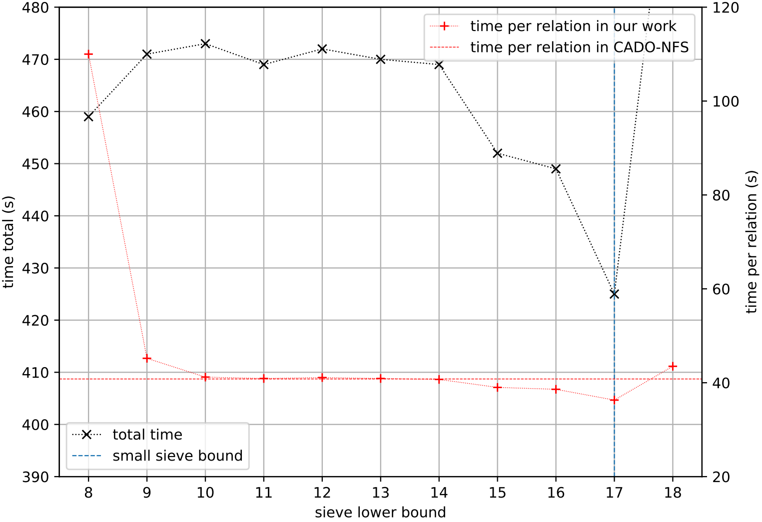
Version	# relations	ratio	Time (s)	ratio	local speed-up
Original	390	-	8619	-	-
Hybrid	347	0.89	6940	0.81	1.10

Example B, with `mfbb0` = 117 bits and `mfbb1` = 167 bits

Version	# relations	ratio	Time (s)	ratio	local speed-up
Original	674	-	6942	-	-
Hybrid	606	0.90	5684	0.82	1.10

Results

Testing multiple values of `sbmp` (sieve lower bound)



Conclusion

Results

- Fewer relations are found
- Speedup counteracts this
- Better efficiency

To come

- Use CADO-NFS tasks to fill up batches
- How much more sieving is needed to counterbalance?
- Public integration in CADO-NFS (as an option?)
- Explore sieving only small primes