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

Ambroise FLEURY

CEA LIST Saclay - LIP6

ePrint/2023/801

October 18, 2023
Factorization
  RSA Cryptosystem
  Factoring a large number

Number Field Sieve (NFS)
  Overview
  Relations
  CADO-NFS

Our contribution
  Batch factoring
  Hybrid version
  Implementation
Factorization
   RSA Cryptosystem
   Factoring a large number

Number Field Sieve (NFS)
   Overview
   Relations
   CADO-NFS

Our contribution
   Batch factoring
   Hybrid version
   Implementation
RSA Cryptosystem

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

Public key
- Used for encryption
- Generated from product $N = pq$

Factorization
- RSA security is linked to the hardness of integer factorization
- Finding $p$ and $q$ from $N$ breaks RSA
Factoring a large number

Shor’s algorithm!
Factoring a large number

Shor’s algorithm!

Classically?
Fermat’s method

▶ Try many $x$’s
▶ Is $x^2 - N$ a square?

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

Smarter way than trying $x$’s until randomly getting a square?
Fermat’s method

- Try many $x$’s
- Is $x^2 - N$ a square?

Then...

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

Smarter way than trying $x$’s until randomly getting a square?
Fermat’s method

- Try many $x$’s
- Is $x^2 - N$ a square?

Then...

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

Smarter way than trying $x$’s until randomly getting a square?
Quadratic Sieve

**Build a square**
- Generate many $y_i = x_i^2 \mod N$
- Build $Y^2 \mod N$ as a product of $y_i$’s

**Building $Y^2$**
- Factor entirely many $y_i$’s (a relation)
- Linear algebra
  - Write each relation as a list of exponents of prime factors
  - Combine to get even exponents
  - It’s a square!

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

**Build a square**

- Generate many $y_i = x_i^2 \mod N$
- Build $Y^2 \mod N$ as a product of $y_i$’s

**Building $Y^2$**

- Factor entirely many $y_i$’s (a relation)
- Linear algebra
  - Write each relation as a list of exponents of prime factors
  - Combine to get even exponents
  - It’s a square!

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

**Build a square**

- Generate many $y_i = x_i^2 \mod N$
- Build $Y^2 \mod N$ as a product of $y_i$’s

**Building $Y^2$**

- Factor entirely many $y_i$’s (a relation)
- Linear algebra
  - Write each relation as a list of exponents of prime factors
  - Combine to get even exponents
  - It’s a square!

From factoring a large number...
...to factoring many small numbers
Factorization
RSA Cryptosystem
Factoring a large number

Number Field Sieve (NFS)
Overview
Relations
CADO-NFS

Our contribution
Batch factoring
Hybrid version
Implementation
NFS: Overview

State-of-the-art algorithm

General idea

- $x^2 \equiv y^2 \pmod{N}$
- $x \pm y \neq 0 \pmod{N}$
- $\gcd(x \pm y, N)$ gives a factor of $N$

2 main parts

1. Collection of relations
   - Find many relations

2. Linear algebra
   - Combine them

Very similar to the quadratic sieve (so far...)
NFS: Overview

State-of-the-art algorithm

General idea

- \( x^2 \equiv y^2 \pmod{N} \)
- \( x \pm y \neq 0 \pmod{N} \) ?
- \( \gcd(x \pm y, N) \) gives a factor of \( N \)

2 main parts

1. Collection of relations
   - Find many relations

2. Linear algebra
   - Combine them

Very similar to the quadratic sieve (so far...)
NFS : Relations

Two sides in NFS

For each pair \((a, b)\)

- Factor rational norm
- Factor algebraic norm

Small enough factors on both norms?

- Relation
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 the relation collection
Relation collection in CADO-NFS

(a, b) pairs space is **large**

- No need to factor *all norms*

Objective
Finding just enough relations in the shortest time
Relation collection in CADO-NFS

(a, b) pairs space is large

- No need to factor \textit{all norms}

Objective

Finding just enough relations in the shortest time
Factoring norms

2 methods:

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

Step 1: sieve all norms
Step 2: ECM on norms most likely to become relations
Factoring norms

2 methods:

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

Rational side factors:

\[ \frac{2}{\text{sieve}} \rightarrow \frac{\text{lim}_0}{\text{ECM}} \rightarrow \text{large prime bound}_0 \]

Algebraic side factors:

\[ \frac{2}{\text{sieve}} \rightarrow \frac{\text{lim}_1}{\text{ECM}} \rightarrow \text{large prime bound}_1 \]

▶ Step 1: sieve all norms
▶ Step 2: ECM on norms most likely to become relations
The structure of norms and \((a, b)\) pairs allows sieving on a side:

- Pick a side and a prime factor \(p\)
- Find and tick a pair \((a, b)\) whose norm it divides
- Tick the next \(p\)-th pair \((a + p, b)\)
- Tick all \(p\)-th pairs
Promising pairs

- Best candidates to give a relation
- Sieving factored enough for both norms
- Only promising pairs get to the ECM step
If the bound deciding whether or not a pair is sent to ECM is...

- Too high
  - Many pairs of low quality will take too much time in ECM
- Too low
  - Few pairs of high quality will give too few relations and additional sieving will be needed
Promising bound

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

- Too high
  - Many pairs of low quality will take too much time in ECM
- Too low
  - Few pairs of high quality will give too few relations and additional sieving will be needed
Promising bound

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

- Too high
  - Many pairs of low quality will take too much time in ECM
- Too low
  - Few pairs of high quality will give too few relations and additional sieving will be needed
Factorization
  RSA Cryptosystem
  Factoring a large number

Number Field Sieve (NFS)
  Overview
  Relations
  CADO-NFS

Our contribution
  Batch factoring
  Hybrid version
  Implementation
Improving relation collection in CADO-NFS

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

Small sieve
Subroutine of CADO-NFS sieving finding small primes

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

Remove small sieve?
Improving relation collection in CADO-NFS

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

Small sieve
Subroutine of CADO-NFS sieving finding small primes

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

Remove small sieve?
Improving relation collection in CADO-NFS

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

**Small sieve**
Subroutine of CADO-NFS sieving finding small primes

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

**Remove small sieve?**
Batch factoring

How to find smooth parts of integers [Bernstein 2004]

- Input: list of integers, factor base \((b\) bits\)
- Output: list of smooth parts, meaning the product of factors from the base found in each integer
- \(O(b(\log b)^2+o(1))\)
Pick an intermediate "batch promising" bound larger than the "ECM promising" bound, then:

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

CADO-NFS
version

2

sieve

ECM

lim

large prime bound

prime factors

batch version

2

batch bound

 batch

partial sieve

ECM

lim

large prime bound

prime factors
Path to ECM

General diagram

| m |
---|
  min sieved bits  | batch  | ECM |

Filtered norm (after sieve)

| m |
---|
  partial sieve  | batch  | ECM |

Filtered norm (after batch)

| m |
---|
  partial sieve  | batch  | ECM |

Promising norm

| m |
---|
  partial sieve  | batch  | ECM |
Implementation in CADO-NFS

RSA-250’s relations
▶ Targeted number of relations
▶ Sets of parameters

Results
▶ Fewer relations are found
▶ Speedup counteracts this
▶ Better efficiency
▶ Up to 1.1 overall speedup

Benchmarks
▶ Sampled sieved regions
▶ Easy extrapolation
Implementation in CADO-NFS

RSA-250’s relations

- Targeted number of relations
- Sets of parameters

Benchmarks

- Sampled sieved regions
- Easy extrapolation

Results

- Fewer relations are found
- Speedup counteracts this
- Better efficiency
- Up to 1.1 overall speedup
Implementation in CADO-NFS

RSA-250’s relations

- Targeted number of relations
- Sets of parameters

Results

- Fewer relations are found
- Speedup counteracts this
- Better efficiency
- Up to 1.1 overall speedup

Benchmarks

- Sampled sieved regions
- Easy extrapolation
Speedup

Target: 90% of relations

![Graph showing speedup and time per relation]
Thank you!