Rounding and Chaining LLL:
Finding Faster Small Roots of Univariate Polynomial Congruences

J. Bi, J-S. Coron, J-C. Faugère, P. Nguyen,
G. Renault, R. Zeitoun

Public Key Cryptography 2014
26-28 March, 2014 - Buenos Aires, Argentina

1. Coppersmith's Method

2. Speeding up Coppersmith's Algorithm by Rounding

3. Speeding up Exhaustive Search by Chaining

Core Ideas of Rounding and Chaining

Rounding:

- **The problem:** $a \xrightarrow{f} b$
  - ☞ Rather consider a/c instead of a.

Chaining:

- **The problem:** $a_1 \xrightarrow{f} b_1,\ a_2 \xrightarrow{f} b_2,\ a_3 \xrightarrow{f} b_3,\ \ldots$
  - ☞ Rather do $a_1 \xrightarrow{f} b_1,\ f'(b_1) \xrightarrow{f} b_2,\ f'(b_2) \xrightarrow{f} b_3,\ \ldots$

☞ Rounding and Chaining can also be combined.

The Problem (Univariate Modular Case):

- **Input:**
  - A polynomial $f(x) = x^\delta + a_{\delta-1}x^{\delta-1} + \cdots + a_1 x + a_0$.
  - N an integer of unknown factorization.
- **Find:**
  - All integers x_0 such that $f(x_0) \equiv 0 \mod N$.

Coppersmith's Theorem

The Problem (Univariate Modular Case):

- **Input:**
  - A polynomial $f(x) = x^\delta + a_{\delta-1}x^{\delta-1} + \cdots + a_1 x + a_0$.
  - N an integer of unknown factorization.
- **Find:**
  - All integers x_0 such that $f(x_0) \equiv 0 \mod N$.

Coppersmith's Method (1996)

- Find **small** integer roots.

Coppersmith's Theorem

The Problem (Univariate Modular Case):

- **Input:**
  - A polynomial $f(x) = x^\delta + a_{\delta-1}x^{\delta-1} + \cdots + a_1 x + a_0$.
  - N an integer of unknown factorization.
- **Find:**
  - All integers x_0 such that $f(x_0) \equiv 0 \mod N$.

Coppersmith's Theorem for the Univariate Modular case

- The solutions x_0 can be found in time poly $(\log N, \delta)$ if:

$$|x_0| < N^{1/\delta} .$$

Coppersmith's Theorem

The Problem (Univariate Modular Case):

- **Input:**
  - A polynomial $f(x) = x^{\delta} + a_{\delta-1}x^{\delta-1} + \cdots + a_1 x + a_0$.
  - N an integer of unknown factorization.
- **Find:**
  - All integers x_0 such that $f(x_0) \equiv 0 \mod N$.

The problem is easy without the modulo N.

☞ Find a polynomial g such that $g(x_0) = 0$ over $\mathbb{Z}$.

Applications in cryptology

Cryptanalysis of RSA

- Factoring with high bits known. *Coppersmith, 1996.*
- Security proof of RSA-OAEP. *Shoup, 2001.*
- Equivalence: factoring / computing d. *Coron, May, 2007.*
- Stereotyped messages. *Coppersmith, 1996.*
- RSA Pseudorandom Generator *Fischlin, Schnorr, 2000.*
- Affine Padding. *Coppersmith, Franklin, Patarin, Reiter, 1996.*
- Polynomially related messages (Hastad). *Coppersmith, 1997.*
- Finding smooth numbers and Factoring. *Boneh, 2001.*
- Coppersmith in the wild. *Bernstein et al., 2013.*

About Coppersmith's Method

Euclidean Lattices

Find a new **small** polynomial equation ☞ **LLL Reduction.**

A matter of Bound

Coppersmith's bound $|x_0| < N^{1/\delta}$ ☞ **Exhaustive search.**

About Coppersmith's Method

Euclidean Lattices

Find a new **small** polynomial equation ☞ **LLL Reduction.**

A matter of Bound

Coppersmith's bound $|x_0| < N^{1/\delta}$ ☞ **Exhaustive search.**

In practice

- The LLL-reduction can be costly.
- The exhaustive search can be prohibitive.

Rounding and Chaining LLL

Our Approach

- Use structure to improve Coppersmith's method.

Two Speedups: Rounding and Chaining

- Asymptotical speed-up of LLL-reduction: $\delta^{-2}\log^9 N \to \log^7 N$
- Heuristic speed-up of the exhaustive search.

Core Ideas of Rounding and Chaining

- **Rounding:** Apply LLL on a matrix with smaller coefficients
  ☞ Divide all coefficients in Coppersmith's matrix.
- **Chaining:** Reuse previous computation
  ☞ Apply a small transformation on the last reduced matrix.

Rounding and Chaining LLL

Our Approach

- Use structure to improve Coppersmith's method.

Two Speedups: Rounding and Chaining

- Asymptotical speed-up of LLL-reduction: $\delta^{-2} \log^9 N \rightarrow \log^7 N$
- Heuristic speed-up of the exhaustive search.

Timings for a typical instance ($\lceil \log_2(N) \rceil = 2048$ and $\delta = 3$)

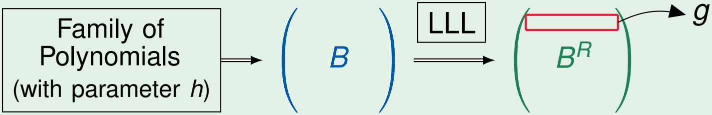
- Original method: 4 years.
- Our new method: 2.6 days.

Coppersmith's Method (Howgrave-Graham)

The problem: find all small integers x_0 s.t. $f(x_0) \equiv 0 \mod N$.

The idea: find a small polynomial g s.t. $g(x_0) = 0$ over $\mathbb{Z}$.

How to find the polynomial g:



$$\left.\begin{array}{l} \bullet\ g(x_0) \equiv 0 \mod N^{h-1} \\ \bullet\ g(x_0) < N^{h-1} \end{array}\right\} \Rightarrow g(x_0) = 0 \text{ over } \mathbb{Z}.$$

Complexity / Practical Results of Coppersmith's Method

State-of-the-art Analysis

- Complexity using L^2: $O(\log^9(N)/\delta^2)$.

In practice, for $\lceil \log_2(N) \rceil = 1024$ and $\delta = 2$

Upper bound for x_0	2^{492}	2^{496}	2^{500}	2^{503}	2^{504}	2^{505}	...	2^{512}
Lattice Dimension $\mathbf{n = h\delta + 1}$	29	35	51	71	77	87	...	NA
Size of elements in B (bits)	15360	18432	26624	36864	39936	45056	...	NA
Time for LLL (seconds)	10.6	35.2	355	2338	4432	11426	...	NA

Remark: All tests were performed using Magma V2.19-5.

[L^2] *An LLL Algorithm with Quadratic Complexity.* P. Q. Nguyen and D. Stehlé, *SIAM J. of Computing, 2009*.

Using Structure: A First Result

State-of-the-art Analysis

- Complexity using L^2: $O(\log^9(N)/\delta^2)$.

New Preliminary Result Using Structure [1]

- Complexity using L^2: $O(\log^8(N)/\delta)$.

[1] *An Upper Bound on the Average Number of Iterations of the LLL Algorithm.* Hervé Daudé, Brigitte Vallée, 1994.

Speeding up Coppersmith’s Algorithm by Rounding

☞ Use Coppersmith’s matrix structure.

The idea: Perform computations with most significant bits

$$\begin{pmatrix} & & & & \\ A & \leqslant & & \leqslant & B \\ & & & & \end{pmatrix} \Rightarrow \begin{pmatrix} & & & & \\ \frac{A}{c} & \leqslant & & \leqslant & \frac{B}{c} \\ & & & & \end{pmatrix}$$

Speeding up Coppersmith's Algorithm by Rounding

$$B = \begin{pmatrix}
N^{h-1} &&&&&&&&&&&& \\
& XN^{h-1} &&&&&&&&&&& \\
&& \ddots &&&&&&&&&& \\
&&& X^{\delta-1}N^{h-1} &&&&&&&&& \\
a_0N^{h-2} && \cdots && X^{\delta}N^{h-2} &&&&&&&& \\
& a_0XN^{h-2} && \cdots && X^{\delta+1}N^{h-2} &&&&&&& \\
&& \ddots &&&& \ddots &&&&&& \\
&&& a_0X^{\delta-1}N^{h-2} && \cdots && X^{2\delta-1}N^{h-2} &&&&& \\
&&& \cdots && \cdots &&& \ddots &&&& \\
a_0^{\delta} &&& \cdots &&& \cdots &&& X^{\delta(h-1)} &&& \\
& a_0^{\delta}X &&& \cdots &&& \cdots &&& X^{\delta(h-1)+1} && \\
&& \ddots &&&&&&&&& \ddots & \\
&&& a_0^{\delta}X^{\delta-1} && \cdots &&& \cdots &&&& X^{\delta h-1}
\end{pmatrix}$$

Largest

Smallest

☞ Since $X < N^{\frac{1}{\delta}}$, all diagonal elements lie between N^{h-2} and N^{h}.

Speeding up Coppersmith's Algorithm by Rounding

First step of rounding method

- Size-reduce B so that subdiagonal coefficients are smaller than diagonal coefficients.

$$B = \text{Size-Reduce}(B) = \begin{pmatrix} \mathbf{b_1} & & & & \\ < b_1 & \mathbf{b_2} & & & \\ < b_1 & < b_2 & \mathbf{b_3} & & \\ < b_1 & < b_2 & < b_3 & & \\ & \cdots & & \ddots & \\ < b_1 & < b_2 & < b_3 & \cdots & \mathbf{b_n} \end{pmatrix}$$

Second step of the rounding method

- Create a new rounded matrix $\lfloor B/c \rfloor$.
- Apply LLL on $\lfloor B/c \rfloor$

$$\begin{pmatrix} & B & \end{pmatrix} \xrightarrow{\boxed{/c}} \begin{pmatrix} & \lfloor B/c \rfloor & \end{pmatrix} \xrightarrow{\boxed{\text{LLL}}} \left[\begin{pmatrix} & T & \end{pmatrix}, \begin{pmatrix} & \lfloor B/c \rfloor^R & \end{pmatrix} \right]$$

Second step of the rounding method

- Create a new rounded matrix $\lfloor B/c \rfloor$.
- Apply LLL on $\lfloor B/c \rfloor$

$$\begin{pmatrix} & & \\ & B & \\ & & \end{pmatrix} \xrightarrow{\boxed{/c}} \begin{pmatrix} & & \\ & \lfloor B/c \rfloor & \\ & & \end{pmatrix} \xrightarrow{\boxed{\text{LLL}}} \left[\begin{pmatrix} & & \\ & T & \\ & & \end{pmatrix} \times \begin{pmatrix} & & \\ & \lfloor B/c \rfloor & \\ & & \end{pmatrix} \right]$$

Speeding up Coppersmith's Algorithm by Rounding

Second step of the rounding method

- Create a new rounded matrix $\lfloor B/c \rfloor$.
- Apply LLL on $\lfloor B/c \rfloor$: first vector of unimodular matrix is $\mathbf{x}$.
- Compute $\mathbf{v} = \mathbf{x}B$ and solve $\mathbf{v}$ over $\mathbb{Z}$.

$$\begin{pmatrix} B \end{pmatrix} \xrightarrow{\boxed{/c}} \begin{pmatrix} \lfloor B/c \rfloor \end{pmatrix} \xrightarrow{\boxed{\text{LLL}}} \left[\begin{pmatrix} \boxed{\mathbf{x}} \\ T \end{pmatrix} \times \begin{pmatrix} \lfloor B/c \rfloor \end{pmatrix} \right]$$

$$\begin{pmatrix} \mathbf{x} \end{pmatrix} \times \begin{pmatrix} B \end{pmatrix} = \begin{pmatrix} \mathbf{v} \end{pmatrix}$$

Complexity of Rounding Method

Theorem: Rounding Method

- Complexity using L^2: $O(\log^7 N)$.

Remainder on Coppersmith's method complexity:

- State-of-the-art complexity: $O(\log^9(N)/\delta^2)$.
- New preliminary complexity: $O(\log^8(N)/\delta)$.

Timings with Rounding Improvement

In practice, for $\lceil \log_2(N) \rceil = 1024$ and $\delta = 2$

Upper bound for x_0	2^{492}	2^{496}	2^{500}	2^{503}	$\mathbf{2^{504}}$	2^{505}	...	2^{512}
Lattice Dimension	29	35	51	71	**77**	87	...	NA
Size of elements in B (bits)	15360	18432	26624	36864	**39936**	45056	...	NA
Size of elements in $\lfloor B/c \rfloor$	2131	2127	2119	2119	**2120**	2123	...	NA
Original LLL (seconds)	10.6	35.2	355	2338	**4432**	11426	...	NA
Rounding LLL (seconds)	1.6	3.5	18.8	94	**150**	436	...	NA

☞ Dim 77: Speed-up of ≈ 30.

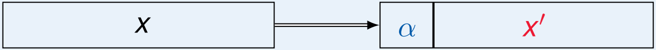
Speeding up Exhaustive Search by Chaining

☞ Use hidden algebraic structure.

Exhaustive Search

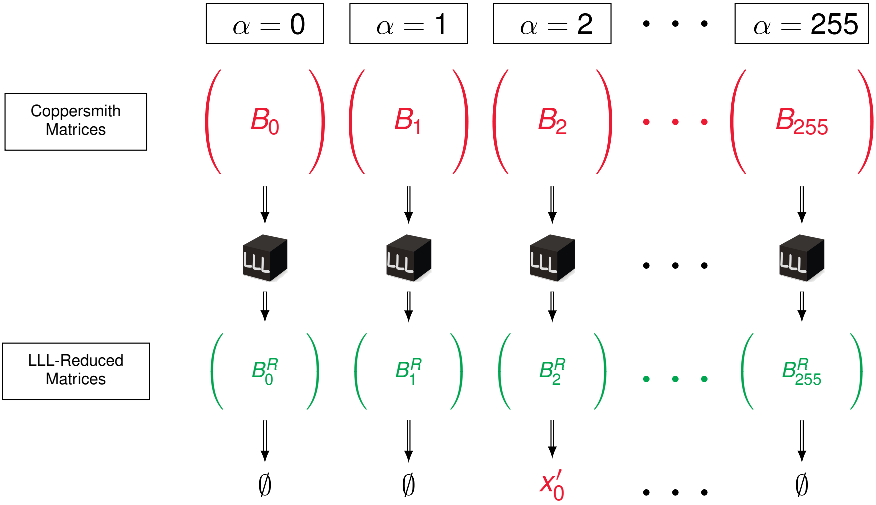
Performing exhaustive search

- Split the variable x into α and x'.

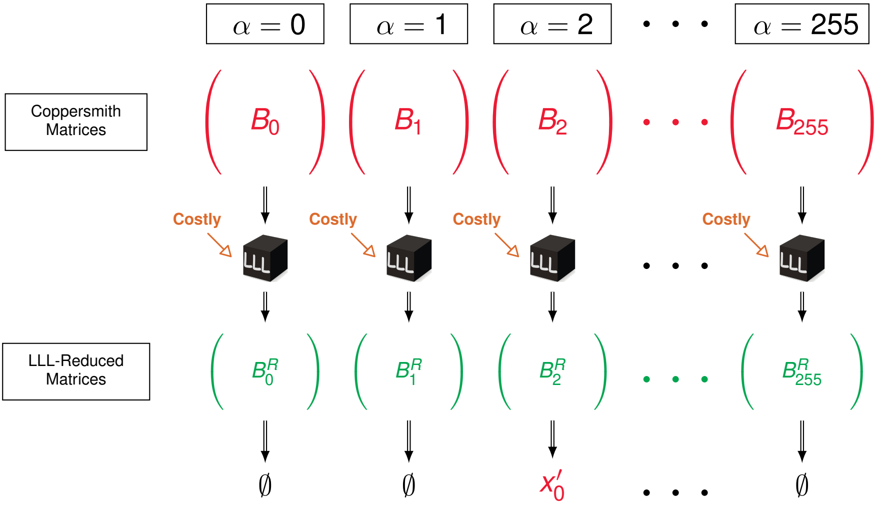


- The new variable is x'.
- Perform an exhaustive search on α.

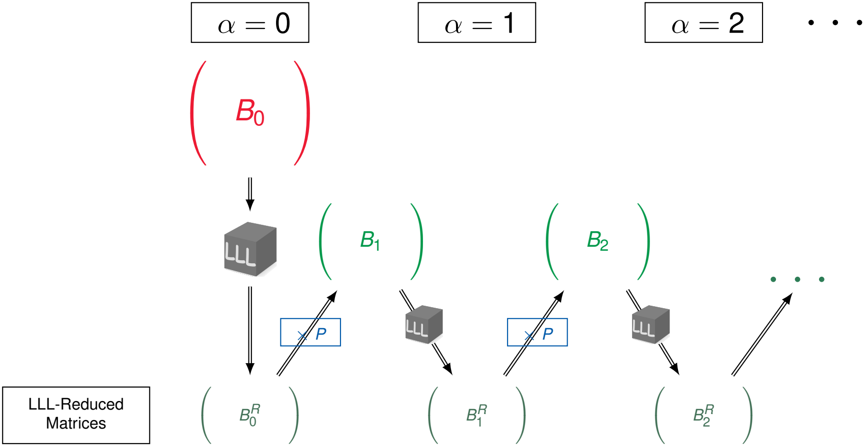
Exhaustive Search



Exhaustive Search



A New Exhaustive Search Scheme



Transformation P is the Pascal Matrix

Proposition

The matrix $B_i^R \cdot P$ is a basis for the case $\alpha = i + 1$, where

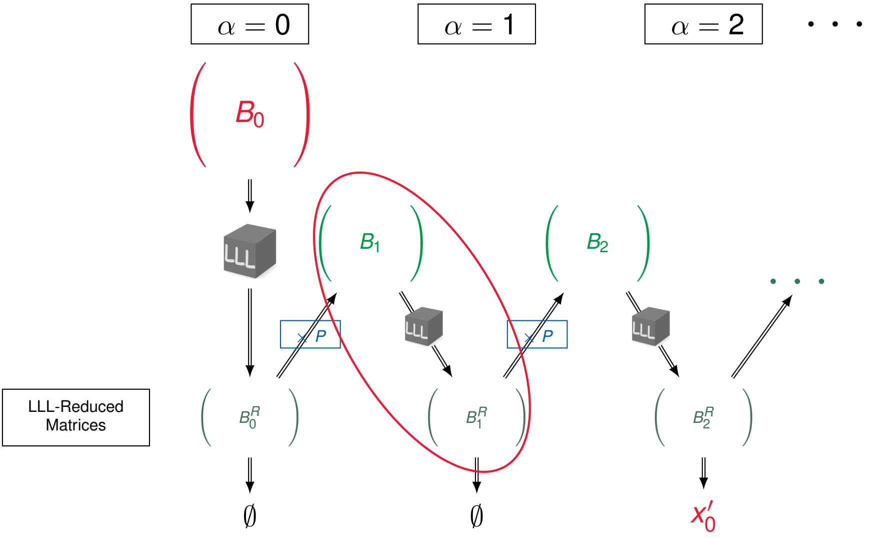
$$P = \begin{pmatrix} 1 & & & & \\ 1 & 1 & & & \\ 1 & 2 & 1 & & \\ 1 & 3 & 3 & 1 & \\ 1 & 4 & 6 & 4 & 1 \\ \vdots & \ldots & & \ldots & \ddots \end{pmatrix}$$

is the Lower Triangular Pascal Matrix.

Consequence on $B_i^R \cdot P$

- Vectors in $B_i^R \cdot P$ are close to the ones of B_i^R.

Combining Chaining and Rounding



Combining Chaining and Rounding

Chaining and Rounding Method

- Create a new rounded matrix $\lfloor B_1/c \rfloor$.
- Apply LLL on matrix $\lfloor B_1/c \rfloor$: Get T_1 and $\lfloor B_1/c \rfloor^R$.
- Compute $B_1^R = T_1 \times B_1$.

$$\left(B_1 \right) \xrightarrow{\boxed{/c}} \left(\lfloor B_1/c \rfloor \right) \xrightarrow{\boxed{\text{LLL}}} \left[\left(T_1 \right) \times \left(\lfloor B_1/c \rfloor \right) \right]$$

$$\left(T_1 \right) \times \left(B_1 \right) = \left(B_1^R \right)$$

Complexity of Rounding+Chaining Method

Heuristic: Rounding+Chaining Method

- Complexity using L^2: $O(\log^7 N)$.

Remark: Same complexity as for Rounding Method alone.

Timings with Rounding and Chaining Improvements

In practice, for $\lceil \log_2(N) \rceil = 1024$ and $\delta = 2$

Upper bound for x_0	2^{492}	2^{496}	2^{500}	2^{503}	$\mathbf{2^{504}}$	2^{505}	...	2^{512}
Lattice Dimension	29	35	51	71	**77**	87	...	NA
Original LLL (sec.)	10.6	35.2	355	2338	**4432**	11426	...	NA
Rounding LLL (sec.)	1.6	3.5	18.8	94	**150**	436	...	NA
Rounding + Chaining (sec.)	0.04	0.12	1.4	9.9	**15.1**	46.5	...	NA

☞ Dim 77: Speed-up of $\approx$ 300.

Conclusion/Perspectives

Conclusion

- This work reduces:
  - the complexity of performing LLL on Coppersmith matrix,
  - the time of exhaustive search to reach Coppersmith bound.
- It allows to reach Coppersmith's bound.
- It is easy to implement.

Conclusion/Perspectives

Conclusion

- This work reduces:
  - the complexity of performing LLL on Coppersmith matrix,
  - the time of exhaustive search to reach Coppersmith bound.
- It allows to reach Coppersmith's bound.
- It is easy to implement.

Perspectives

- Generalization to the multivariate case (approximate gcd).
- Refine complexity for Chaining + Rounding method.