Solving Modular Linear Equations via Automated Coppersmith and its Applications

Yansong Feng[1] **Zhen Liu**[2] Abderrahmane Nitaj[3] Yanbin Pan[1]

[1] Academy of Mathematics and Systems Science, Chinese Academy of Sciences

[2] School of Cyber Science and Technology, Hubei University

[3] Normandie University

December 14, Inscrypt 2024

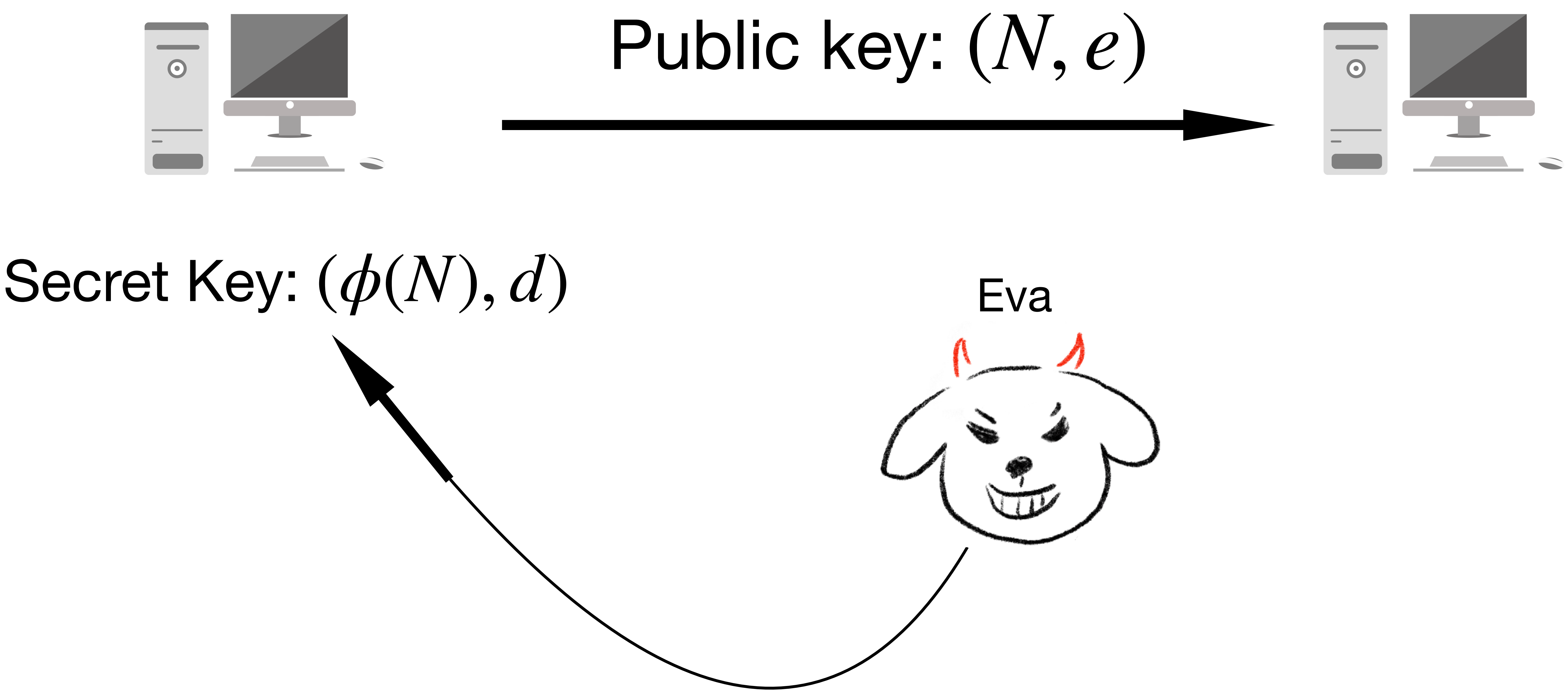
Outline

- Background
- Lattice-based Cryptanalysis: Coppersmith’s method
- Our Results
    - Implicit Factorization Problem

Background

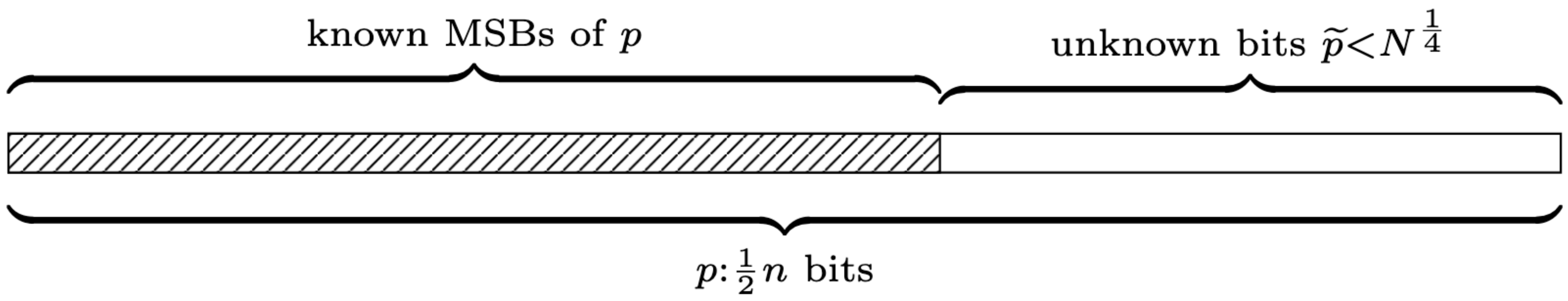
RSA Cryptosystem

$$ed \equiv 1 \mod \phi(N) \qquad \phi(N) = (p-1)(q-1)$$



Eve wants to get the SECRET KEY!!!

Lucky Eva got enough MSBs of p…



Now he just needs to solve a linear polynomial equation:

$$f(x) = x + C \equiv 0 \mod p \text{ with a small root } x_0 = \widetilde{p} < N^{\frac{1}{4}}$$

How to solve polynomials equations with small roots?

Coppersmith’s method

Coppersmith's method

Let $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n} \in \mathbb{R}^m$, the lattice $\mathscr{L}$ is

$$\mathscr{L} = \left\{ \mathbf{v} \in \mathbb{R}^m \mid \mathbf{v} = \sum_{i=1}^{n} a_i \mathbf{v_i}, a_i \in \mathbb{Z} \right\}.$$

Given bounds $X_1, \cdots, X_k$ and $f_1, \cdots, f_n \in \mathbb{Z}[x_1, \ldots, x_k]$ and modulus M, the goal is to find

the small root $\mathbf{u} = (u_1, \ldots, u_k)$ with $|u_j| < X_j$, such that $f_i(\mathbf{u}) \equiv 0 \mod M,\ 1 \leq i \leq n$.

1. Generate shift-polynomials $g_{[i_1, \cdots, i_n, j_1, \cdots, j_k]} = f_1^{i_1} \cdot \ldots \cdot f_n^{i_n} \cdot x_1^{j_1} \cdot \ldots \cdot x_k^{j_k} \cdot M^{m-(i_1+\cdots+i_n)}$ have the root $\mathbf{u}$ module M^m, for some m.

2. Use the coefficient vector of $g_{[i_1, \cdots, i_n, j_1, \cdots, j_k]}(x_1 X_1, \ldots, x_k X_k)$ to construct $\mathscr{L}$.

3. Use Lattice Reduction (LLL) to find **shorter vectors** $h_1, \ldots, h_k$

$$h_j(\mathbf{u}) \equiv 0 \mod M^m \longrightarrow h_j(\mathbf{u}) = 0 \text{ over } \mathbb{Z}$$

Crucial Condition: $\mathscr{L}$ MUST satisfied $\det(\mathscr{L}) < M^{m\dim(\mathscr{L})}$.

Example: $f(x) = x + C \equiv 0 \mod p$ with a small root $x_0 = \widetilde{p} < N^{\frac{1}{4}}$

$$
\begin{array}{ccccccccc}
x^0 & x^1 & x^2 & x^3 & x^4 & x^5 & x^6 & x^7 & x^8 \\
\end{array}
$$

$$
\left|\begin{array}{c|c|c|c|c|c|c|c|c}
N^4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
* & N^3X & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
* & * & N^2X^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
* & * & * & NX^3 & 0 & 0 & 0 & 0 & 0 \\
* & * & * & * & X^4 & 0 & 0 & 0 & 0 \\
* & * & * & * & * & X^5 & 0 & 0 & 0 \\
* & * & * & * & * & * & X^6 & 0 & 0 \\
* & * & * & * & * & * & * & X^7 & 0 \\
* & * & * & * & * & * & * & * & X^8
\end{array}\right|
$$

Lower triangular

$$\dim(\mathscr{L}) = m + o(m)$$

$$\det(\mathscr{L}) = N^{\frac{1}{8}m^2 + o(m^2)} X^{\frac{1}{2}m^2 + o(m^2)}$$

$X < N^{\frac{1}{4}} \rightarrow \det(\mathscr{L}) < p^{m\dim(\mathscr{L})} \rightarrow f$ can be solved with Coppersmith's method.

Using Coppersmith's method, compute $\dim(\mathscr{L})$ and $\det(\mathscr{L})$:

Manual Calculation such as calculating $\sum_{k=0}^{m}\sum_{i=k}^{m}(i - \min(s, i))$? NO!

Theorem: $\dim(\mathscr{L})$ **and** $\det(\mathscr{L})$ **are polynomials in** m**.**

Now,

~~Manual Calculation~~ $\longrightarrow$ Lagrange Interpolation

Automated Coppersmith's method

At Asiacrypt'23, Meers and Nowakowski introduced a new automated method called Automated Coppersmith.

- First determine the elements of the diagonal of the matrix (denote by $\mathscr{M}$) of the lattice $\mathscr{L}$ and then select a suitable subset $\mathscr{F}$ of shift-polynomials to construct the lattice $\mathscr{L}$
  - Shift-polynomials $\dfrac{\lambda}{LM(f_1)^{i_1} \cdot \ldots \cdot LM(f_n)^{i_n}} f_1^{i_1} \cdot \ldots \cdot f_n^{i_n} \cdot M^{m-(i_1+\cdots+i_n)}, \lambda \in \mathscr{M}$
  - An element of $\mathscr{M}$ is related to a unique element of $\mathscr{F}$
  - Given a single shift-polynomial, a locally optimal $\mathscr{F}$ can be constructed automatically
- Need to ensure:

$$\prod_{\lambda \in \mathscr{M}} \lambda(X_1, \ldots, X_k) \cdot \prod_{\lambda \in \mathscr{M}} |\mathrm{LC}(\mathscr{F}[\lambda])| \leq M^{(m-k)|\mathscr{M}|}.$$

Leading monomial of the polynomial in $\mathscr{F}$

Coefficient of Leading monomial of the polynomial in $\mathscr{F}$

Automated Coppersmith's method

Use polynomial interpolation to derive bounds $X_1, \cdots, X_k$ automatically

- A sequence of sets $\mathcal{M}_1 \subset \mathcal{M}_2 \subset \cdots$, for any fixed $\mathcal{M}_i$ and $m_i := i \cdot n$– a corresponding optimal set of shift-polynomials $\mathcal{F}_i$.

$$\mathcal{M}_i := \bigcup_{0 \le j_1, \ldots, j_n \le i} \mathrm{supp}\{f_1^{j_1} \cdot \ldots \cdot f_n^{j_n}\},\ \text{for } i \in \mathbb{N}\,.\ \text{(monomials set)}$$

- $\prod_{\lambda \in \mathcal{M}_i} \lambda(X_1, \ldots, X_k) \cdot \prod_{\lambda \in \mathcal{M}_i} |\mathrm{LC}(\mathcal{F}_i[\lambda])| \le M^{(m_i - k)|\mathcal{M}|}$.

$$M^{(m_i-k)|\mathcal{M}_i|} = M^{p_{\mathcal{M}}(m_i)},$$

polynomial interpolation

$$\prod_{\lambda \in \mathcal{M}_i} \lambda(X_1, \ldots, X_k) = X_1^{p_1(m_i)} \cdot \ldots \cdot X_k^{p_k(m_i)}$$

$$\prod_{\lambda \in \mathcal{M}_i} |\mathrm{LC}(\mathcal{F}_i[\lambda])| = M^{p_{\mathcal{F}_i}(m_i)}$$ (heuristic)

$P_{\mathcal{M}}, P_1, \cdots, P_k, P_{\mathcal{F}_i}$ are polynomials of degree $k+1$.

$\Rightarrow X_1^{\alpha_1} \cdots X_k^{\alpha_k} \le M^{\delta - \epsilon}$

Our Results

Motivation

For linear equations $f_1, \cdots, f_n \in \mathbb{Z}[x_1, \ldots, x_k]$, deg($f_i$)=1.

we want to solve $f_i \equiv 0 \bmod \hat{M}_1 * M_2,\ 1 \leq i \leq n.$

where $\hat{M}_1$ is an unknown , M_2 is known, M_1 is a known multiple of $\hat{M}_1$

$$f_i \equiv 0 \bmod \hat{M}_1 * M_2 \Rightarrow f_i \equiv 0 \bmod M_1 * M_2,\ 1 \leq i \leq n?$$

Our results

- Better monomials set (better bounds on $X_1, \cdots, X_k$)

  Non-Homogenous linear equations: there exists i_0 such that $f_{i_0}(0) \neq 0$

  $$\mathscr{M}_i := \{\lambda \mid \lambda \in \mathrm{supp}\{f_1^{j_1} \cdot \ldots \cdot f_n^{j_n}\}, j_1 + \ldots + j_n \leq n \cdot i\}$$

  Homogenous linear equations:

  $$\mathscr{M}_i := \{\lambda \mid \lambda \in \mathrm{supp}\{f_1^{j_1} \cdot \ldots \cdot f_n^{j_n}\}, j_1 + \ldots + j_n = n \cdot i\}$$

- Introduce a new parameter t to consider shift polynomials

$$\frac{\lambda}{LM(f_1)^{i_1} \cdot \ldots \cdot LM(f_n)^{i_n}} f_1^{i_1} \cdot \ldots \cdot f_n^{i_n} \cdot M_1^{\max\{t-(i_1+\cdots+i_n),0\}} M_2^{m_i-(i_1+\cdots+i_n)}, \ \lambda \in \mathscr{M}_i$$

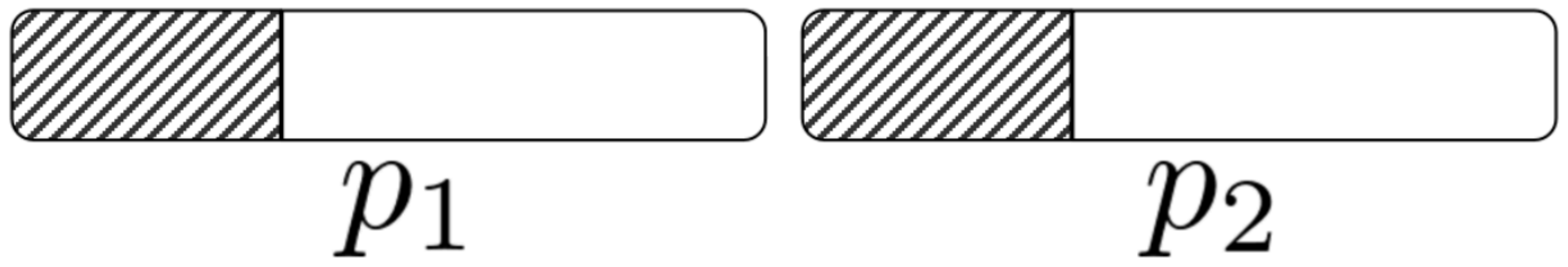
- Need to ensure: $$\prod_{\lambda \in \mathscr{M}_i} \lambda(X_1, \ldots, X_k) \cdot \prod_{\lambda \in \mathscr{M}_i} |\mathrm{LC}(\mathscr{F}_i[\lambda])| \leq M^{(m_i-k)|\mathscr{M}_i|}.$$

  - **Heuristic: (proven to be true)** $$\prod_{\lambda \in \mathscr{M}_i} |\mathrm{LC}(\mathscr{F}_i[\lambda])| = M_1^{p_{\mathscr{F}_1}(t,m_i)} \cdot (M_2)^{p_{\mathscr{F}_2}(m_i)}.$$

Homogenous & $n = 1$ → G-EIFP ?

Implicit Factorization Problem

IFP (MSBs case)



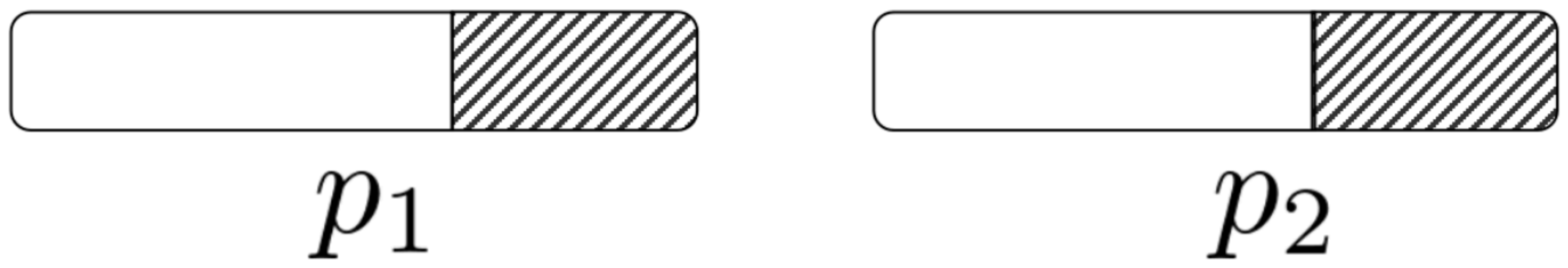
$N_1 = p_1 q_1$ and $N_2 = p_2 q_2$

p_1 share the same MSBs with p_2

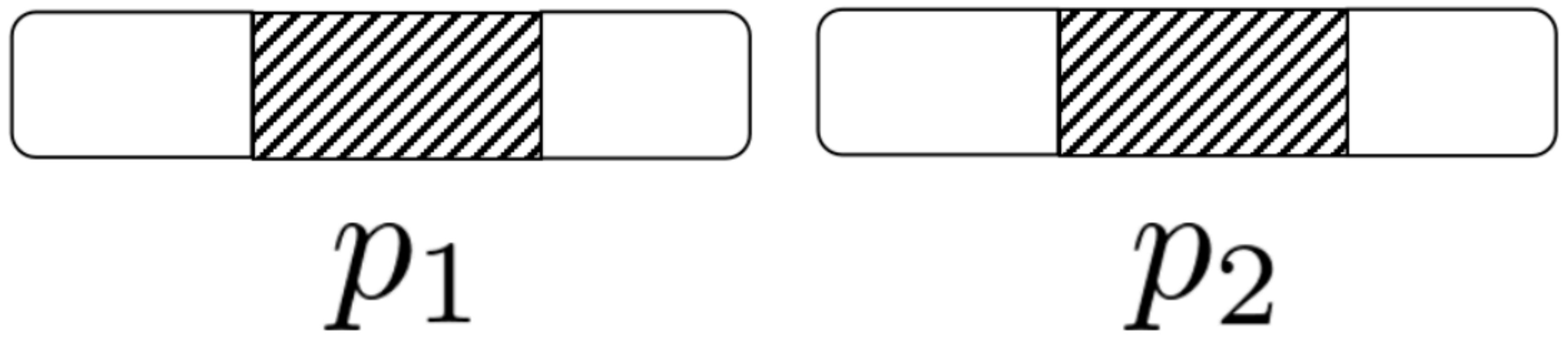
$$N_1 + (p_2 - p_1)q_1 = p_2 q_1 \equiv 0 \mod p_2$$

Solving $f(x_1, x_2) = x_1 x_2 + N_1 \equiv 0 \mod p_2$

IFP (LSBs case)



IFP (Middle case)

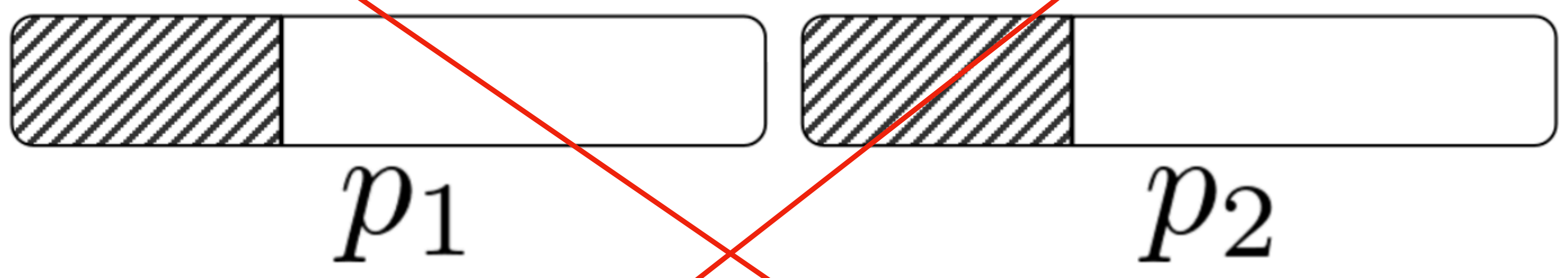


IFP (Generalized case)

(a) p_1 (b) p_2

EIFP (MSBs case)

$N_1 = p_1 q_1$ and $N_2 = p_2 q_2$



~~p_1 share the same MSBs with p_2~~ $\longrightarrow$ $a_1 p_1$ share the same MSBs with $a_2 p_2$

How about EIFP with Generalized case? G-EIFP!

$a_1 p_1$ share some continuous bits with $a_2 p_2$, which can be located in different positions.

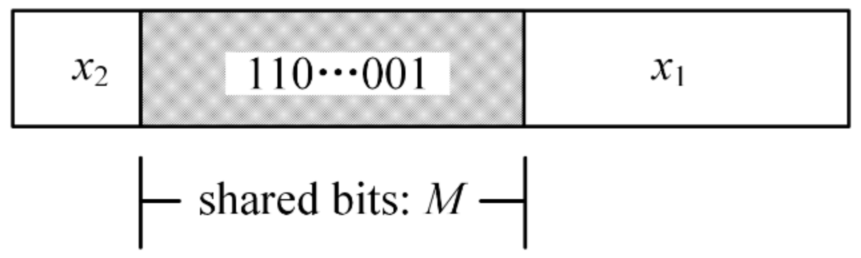
G-EIFP

Definition: Given two n-bit RSA moduli $N_1 = p_1q_1$and $N_2 = p_2q_2$, where q_1 and q_2 are αn-bit, suppose that there exist two positive integers a_1 and a_2 with $a_1, a_2 < 2^{\delta n}$ such that a_1p_1 and a_2p_2 share γn bits, where the shared bits may be located in different positions of a_1p_1 and a_2p_2 . The Generalized Extended Implicit Factorization Problem (G-EIFP) asks to factor N_1 and N_2.

G-EIFP

suppose that a_1p_1 shares γn-bits from the $\beta_1 n$ bit to $(\beta_1+\gamma)n$-th bit, and a_2p_2 shares bits from $\beta_2 n$-th bit to $(\beta_2+\gamma)n$-th bit, where β_1 and β_2 are known with $\beta_1 \le \beta_2$.

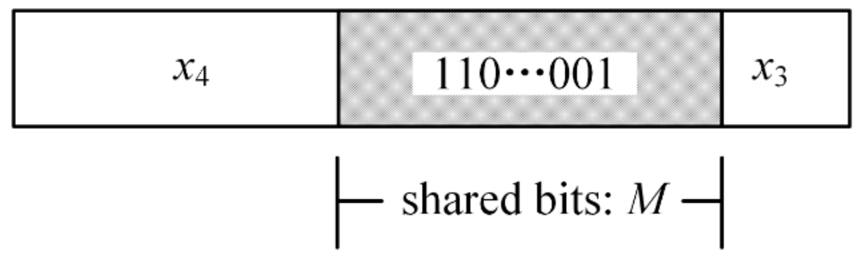
a_1p_1



$$a_1p_1 = x_1 + 2^{\beta_1 n}R + x_2 2^{(\beta_1+\gamma)n},$$

$$a_2p_2 = x_3 + 2^{\beta_2 n}R + x_4 2^{(\beta_2+\gamma)n},$$

a_2p_2



$f(x,y,z) = x + ay + N_2 z$ with $a = 2^{(\beta_2+\gamma)n}$,

where $(x_0, y_0, z_0) = \left(2^{(\beta_2-\beta_1)n}x_1q_2 - x_3q_2,\ x_2q_2 - x_4q_2,\ a_2\right)$ is a solution of the modular equation $f(x,y,z) \equiv 0 \pmod{2^{(\beta_2-\beta_1)n}p_1}$.

Since $\gcd(x_0, y_0, N_2) = q_2$, *introduce w as a new variable* and denote

$$f(x, y, z, w) = x + ay + wz \ ,$$

where $(x_0, y_0, z_0, w_0) = \left(2^{(\beta_2-\beta_1)n}x_1 - x_3, x_2 - x_4, a_2, p_2\right)$ is a solution of the modular equation $f(x, y, z, w) \equiv 0 \pmod{2^{(\beta_2-\beta_1)n}p_1}$.

We want to solve $f \equiv 0 \bmod 2^{(\beta_2-\beta_1)n}p_1$, where p_1 is an unknown divisor of N_1.

- Choose $\mathscr{M}_i := \{\lambda \mid \lambda \in \mathrm{supp}\{f^{j_1}\}, j_1 = m_i\}$
- Consider shift polynomials: $\dfrac{\lambda}{\mathrm{LM}(f)^{i_1}} \cdot f^{i_1} \cdot N_1^{\max\{t-i_1,0\}}(2^{(\beta_2-\beta_1)n})^{m_i-i_1}$
    - Choose $\mathscr{F}_i$: introduce s that will be optimized later to reduce determinant. $(w_0q_2 = N_2, v = q_2)$

$$N_2^{-min\{s,i_1\}}v^s\frac{\lambda}{\mathrm{LM}(f)^{i_1}} \cdot f^{i_1} \cdot N_1^{\max\{t-i_1,0\}}(2^{(\beta_2-\beta_1)n})^{m_i-i_1}$$

- New $\mathscr{M}_i = \{x^{\alpha_1}y^{\alpha_2}z^{\alpha_3}w^{\alpha_3-\min\{\alpha_3,s\}}v^{s-\min\{\alpha_3,s\}} \mid \alpha_1 + \alpha_2 + \alpha_3 = m_i\}$.

$$M^{(m_i-k)|\mathscr{M}_i|} = M^{p_{\mathscr{M}}(m_i)}$$

$$\prod_{\lambda \in \mathscr{M}_i} |\mathrm{LC}(\mathscr{F}_i[\lambda])| = M_1^{p_{\mathscr{F}_1}(t,m_i)} \cdot M_2^{p_{\mathscr{F}_2}(m_i)}$$ (proven to be true)

$$\prod_{\lambda \in \mathscr{M}_i} \lambda(X_1, \ldots, X_k, A, V) = X_1^{p_1(m_i)} \cdot \ldots \cdot X_k^{p_k(m_i)} \cdot W^{p_w(s,m_i)} \cdot V^{p_v(s,m_i)}$$

Where $M_1 = N_1$, $M_2 = 2^{(\beta_2-\beta_1)n}$, *W* is an upper bound of N_2, and *V* is an upper bound of q_2.

(s, m_i)	(0, 0)	(0, 1)	(1, 1)	(0, 2)	(1, 2)	(2, 2)	(0, 3)	(1, 3)	(2, 3)	(3, 3)
$p_w(s, m_i)$	0	1	0	4	1	0	10	4	1	0
$p_v(s, m_i)$	0	0	2	0	3	8	0	4	11	20

Lagrange Interpolation $\longrightarrow$

$$
\begin{aligned}
p_x &= \frac{1}{6}m^3 + o\left(m^3\right), \\
p_y &= \frac{1}{6}m^3 + o\left(m^3\right), \\
p_z &= \frac{1}{6}m^3 + o\left(m^3\right), \\
p_w &= \frac{1}{6}(1-\tau_2)^3 m^3 + o\left(m^3\right), \\
p_v &= \frac{1}{6}\left(-\tau_2^3 + 3\tau_2^2\right) m^3 + o\left(m^3\right), \\
p_{\mathcal{F}_1} &= \frac{1}{6}\tau_1^2(3-\tau_1) m^3 + o\left(m^3\right), \\
p_{\mathcal{F}_2} &= \frac{1}{3}m^3 + o\left(m^3\right), \\
p_{\mathcal{M}} &= m|\mathcal{M}| = \frac{1}{2}m^3 + o\left(m^3\right).
\end{aligned}
$$

$$t = m\tau_1$$

$$s = m\tau_2$$

$\dim(\mathscr{L})$ and $\det(\mathscr{L})$

use Grobner basis method or resultant computations to find $(x_0, y_0, z_0, w_0, v_0) = \left(2^{(\beta_2-\beta_1)n}x_1 - x_3, x_2 - x_4, a_2, p_2, q_2\right)$.

Summary

$$n : \log_2 N_i, \quad \alpha : \log_2 q_i, \quad \delta : \log_2 a_i, \quad \gamma : \frac{\text{shared bits}}{n}$$

Theorem: G-EIFP$(n, \alpha, \gamma, \delta)$ can be solved in polynomial time when

$$\gamma > 4\alpha\left(1 - \sqrt{\alpha}\right) + 2\delta,$$

provided that $\alpha + \gamma \leq 1$.

n	δn	αn	βn	$\beta_1 n$	$\beta_2 n$	γn	m	dim($\mathcal{L}$)	Time for LLL(s)	Time for Gröbner Basis(s)
200	10	20	40	20	30	140	4	15	0.3638	0.0094
400	20	40	80	40	60	280	6	28	1.0674	2.4525
500	25	50	100	50	75	350	6	28	1.3903	3.3241
1000	50	100	200	100	150	700	8	45	40.0927	1184.4979

Table 1: Some experimental results for G-EIFP.

Thanks for listening!

Code: https://github.com/fffmath/CombeeIFP