Generalized Implicit Factorization Problem

Yansong Feng [1] Abderrahmane Nitaj [2] Yanbin Pan [1]

Academy of Mathematics and Systems Science, Chinese Academy of Sciences, China
ysfeng2023@163.com, panyanbin@amss.ac.cn

Normandie Univ, UNICAEN, CNRS, LMNO, 14000 Caen, France
abderrahmane.nitaj@unicaen.fr

December 24, 2023

Outline

Introduction to RSA

RSA has three steps:

Alice

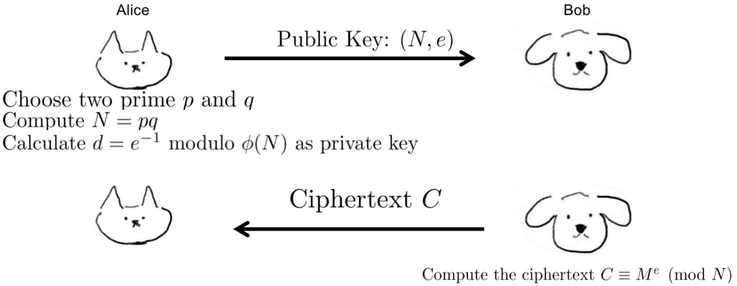
Public Key: (N, e) $\longrightarrow$

Bob

Choose two prime p and q
Compute $N = pq$
Calculate $d = e^{-1}$ modulo $\phi(N)$ as private key

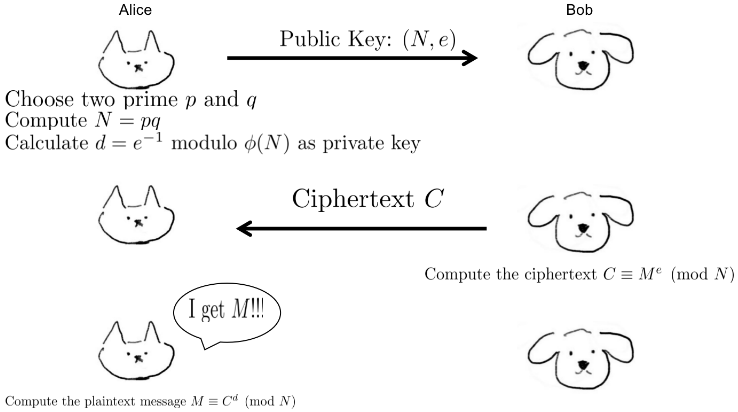
Introduction to RSA

RSA has three steps:



Introduction to RSA

RSA has three steps:



Attack on RSA

- There exist some attack on RSA, such as Side-channel attack, Winner's attack, Coppersmith's attack and so on.

Attack on RSA

- There exist some attack on RSA, such as Side-channel attack, Winner's attack, Coppersmith's attack and so on.
- Coppersmith's attack is a well-known attack on RSA.

Attack on RSA

- There exist some attack on RSA, such as Side-channel attack, Winner's attack, Coppersmith's attack and so on.
- Coppersmith's attack is a well-known attack on RSA.
- For example, by using Coppersmith's method, one can factor a RSA moduli when half of the most significant bits of p are known.

Attack on RSA

- There exist some attack on RSA, such as Side-channel attack, Winner's attack, Coppersmith's attack and so on.
- Coppersmith's attack is a well-known attack on RSA.
- For example, by using Coppersmith's method, one can factor a RSA moduli when half of the most significant bits of p are known.
- We will discuss Coppersmith's method later.

Introduction to the IFP

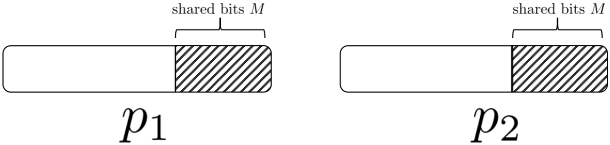
At PKC 2009, May and Ritzenhofen introduced the Implicit Factorization Problem (IFP).

Definition (May, Ritzenhofen [1])

Let $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$ be two different n-bit RSA moduli with αn-bit q_i. The Implicit Factorization Problem (IFP) is to factor N_1 and N_2 with some implicit hints.

IFP in the LSBs case

They proposed their result of IFP in the LSBs case, i.e., p_1 and p_2 share γn bits least significant bits.



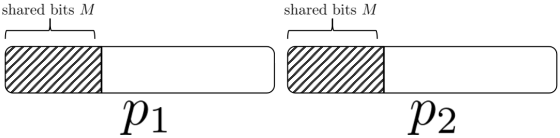
IFP in the other case

In a follow-up work, Sarkar and Maitra [2] generalized the Implicit Factorization Problem to the case where the most significant bits (MSBs) or the middle bits.

Then at PKC 2010, Faugère *et al.* [3] improved the bounds to the case where the most significant bits (MSBs) or the middle bits.

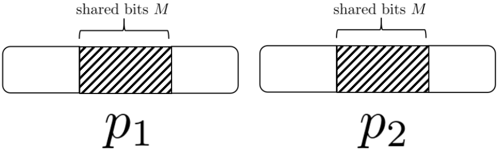
IFP in the MSBs case

The IFP in the MSBs case means factoring N_1 and N_2 with the implicit hint that p_1 and p_2 share most significant bits.



IFP in the Middle case

IFP in the Middle case means the p_i's are primes that all share γn bits from position t1 to t2 $=$ t1 $+ \gamma n$.



Faugère *et al.* [3] show that N_1 and N_2 can be factored in polynomial time when p_1 and p_2 share at least $\gamma n > 4\alpha n + 6$ bits.

IFP in the other case

In 2011, Sarkar and Maitra [4] further expanded the Implicit Factorization Problem by revealing the relations between the Approximate Common Divisor Problem (ACDP) and the Implicit Factorization Problem

1. the primes p_1, p_2 share an amount of the least significant bits (LSBs);
2. the primes p_1, p_2 share an amount of most significant bits (MSBs);
3. the primes p_1, p_2 share both an amount of least significant bits and an amount of most significant bits.

IFP in the other case

In 2011, Sarkar and Maitra [4] further expanded the Implicit Factorization Problem by revealing the relations between the Approximate Common Divisor Problem (ACDP) and the Implicit Factorization Problem

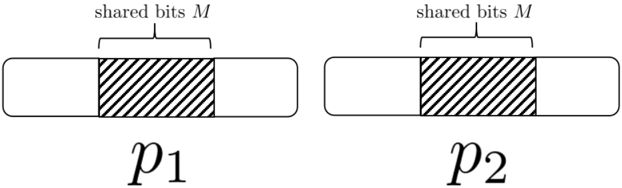
1. the primes p_1, p_2 share an amount of the least significant bits (LSBs);
2. the primes p_1, p_2 share an amount of most significant bits (MSBs);
3. the primes p_1, p_2 share both an amount of least significant bits and an amount of most significant bits.

In 2016, Lu *et al.* [5] presented a novel algorithm and improved the bounds for all the above three cases of the Implicit Factorization Problem.

Revisit the Middle case

In 2015, Peng *et al.* [6] revisited the Implicit Factorization Problem with shared **middle** bits and improved the bound.

The bound was further enhanced by Wang *et al.* [7] in 2018



Recent work on IFP

	LSBs	MSBs	both LSBs-MSBs	Middle bits	General
May, Ritzenhofen [1]	2α	-	-	-	-
Faugère, *et al.* [3]	2α	-	-	4α	-
Sarkar, Maitra [4]	$2\alpha - \alpha^2$	$2\alpha - \alpha^2$	$2\alpha - \alpha^2$	-	-
Lu, *et al.* [5]	$2\alpha - 2\alpha^2$	$2\alpha - 2\alpha^2$	$2\alpha - 2\alpha^2$	-	-
Peng, *et al.*[6]	-	-	-	$4\alpha - 3\alpha^2$	-
Wang, *et al.*[7]	-	-	-	$4\alpha(1-\sqrt{\alpha})$	-
This work	-	-	-	-	$4\alpha(1-\sqrt{\alpha})$

Table: Asymptotic lower bound of γ in the Implicit Factorization Problem for n-bit $N_1 = p_1 q_2$ and $N_2 = p_2 q_2$ where the number of shared bits is γn, q_1 and q_2 are αn-bit.

GIFP

It can be seen in Table 1 that the positions for the sharing bits are located similarly. So we consider a general case that the positions for the sharing bits are located differently.

GIFP

It can be seen in Table 1 that the positions for the sharing bits are located similarly. So we consider a general case that the positions for the sharing bits are located differently.

Definition (GIFP(n, α, γ))

Given two n-bit RSA moduli $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$, where q_1 and q_2 are αn-bit, assume that p_1 and p_2 share γn consecutive bits, where the shared bits may be located in different positions of p_1 and p_2. The Generalized Implicit Factorization Problem (GIFP) asks to factor N_1 and N_2.

GIFP

Theorem

GIFP(n, α, γ) can be solved in polynomial time when

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

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

Figure: Shared bits M for p_1 and p_2

Preliminaries

The proof of this theorem needs some knowledge of Lattice and Coppersmith's theory.

Let $m \geq 2$ be an integer. A lattice is a discrete additive subgroup of $\mathbb{R}^m$. A more explicit definition is presented as follows.

Preliminaries

The proof of this theorem needs some knowledge of Lattice and Coppersmith's theory.

Let $m \geq 2$ be an integer. A lattice is a discrete additive subgroup of $\mathbb{R}^m$. A more explicit definition is presented as follows.

Definition (Lattice)

Let $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n} \in \mathbb{R}^m$ be n linearly independent vectors with $n \leq m$. The lattice $\mathcal{L}$ spanned by $\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}\}$ is the set of all integer linear combinations of $\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}\}$, i.e.,

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

Lattice

The Shortest Vector Problem (SVP) is one of the famous computational problems in lattices.

Definition (Shortest Vector Problem (SVP))

Given a lattice $\mathcal{L}$, the Shortest Vector Problem (SVP) asks to find a non-zero lattice vector $\mathbf{v} \in \mathcal{L}$ of minimum Euclidean norm, i.e., find $\mathbf{v} \in \mathcal{L} \backslash \{\mathbf{0}\}$ such that $\|\mathbf{v}\| \leq \|\mathbf{w}\|$ for all non-zero $\mathbf{w} \in \mathcal{L}$.

LLL Algorithm

Although SVP is NP-hard under randomized reductions [8], there exist algorithms that can find a relatively short vector, instead of the exactly shortest vector, in polynomial time, such as the famous LLL algorithm proposed by Lenstra, Lenstra, and Lovasz [9] in 1982. The following result is useful for our analysis[10].

LLL Algorithm

Although SVP is NP-hard under randomized reductions [8], there exist algorithms that can find a relatively short vector, instead of the exactly shortest vector, in polynomial time, such as the famous LLL algorithm proposed by Lenstra, Lenstra, and Lovasz [9] in 1982. The following result is useful for our analysis[10].

Theorem (LLL Algorithm [9])

Given an n-dimensional lattice $\mathcal{L}$, we can find an LLL-reduced basis $\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}\}$ of $\mathcal{L}$ in polynomial time, which satisfies

$$\|\mathbf{v_i}\| \leq 2^{\frac{n(n-1)}{4(n+1-i)}} \det(\mathcal{L})^{\frac{1}{n+1-i}}, \quad \textit{for} \quad i = 1, \ldots, n.$$

Coppersmith's method

Theorem

Let M be a positive integer, and $f(x_1, \ldots, x_k)$ be a polynomial with integer coefficients. Coppersmith's method give us a way to find a small solution $(y_1, \ldots, y_k)$ of the modular equation $f(x_1, \ldots, x_k) \equiv 0 \pmod{M}$ with the bounds $y_i < X_i$ for $i = 1, \ldots, k$.

Algorithm Overview

The algorithm to find small integer roots using Coppersmith's Theorem involves lattice reduction techniques.

1. Formulate the problem as a lattice problem.

Algorithm Overview

The algorithm to find small integer roots using Coppersmith's Theorem involves lattice reduction techniques.

1. Formulate the problem as a lattice problem.
2. Apply lattice reduction algorithms to find short lattice vectors.

Algorithm Overview

The algorithm to find small integer roots using Coppersmith's Theorem involves lattice reduction techniques.

1. Formulate the problem as a lattice problem.
2. Apply lattice reduction algorithms to find short lattice vectors.
3. Recover integer solutions from the lattice basis.

Coppersmith's method

More precisely,the steps are as follows:

- Construct a set G of k-variate polynomial equations such that $g_i(y_1,\ldots,y_k) \equiv 0 \pmod{M}$;

Coppersmith's method

More precisely,the steps are as follows:

- Construct a set G of k-variate polynomial equations such that $g_i(y_1, \ldots, y_k) \equiv 0 \pmod{M}$;
- use the coefficient vectors of $g_i(x_1X_1, \ldots, x_kX_k)$, $i = 1, \ldots, k$, to construct a k-dimensional lattice $\mathcal{L}$;

Coppersmith's method

More precisely,the steps are as follows:

- Construct a set G of k-variate polynomial equations such that $g_i(y_1, \ldots, y_k) \equiv 0 \pmod{M}$;
- use the coefficient vectors of $g_i(x_1X_1, \ldots, x_kX_k)$, $i = 1, \ldots, k$, to construct a k-dimensional lattice $\mathcal{L}$;
- Applying the LLL algorithm to $\mathcal{L}$, we get a new set H of k polynomial equations $h_i(x_1, \ldots, x_k)$, $i = 1, \ldots, k$, with integer coefficients such that $h_i(y_1, \ldots, y_k) \equiv 0 \pmod{M}$;

Coppersmith's method

More precisely,the steps are as follows:

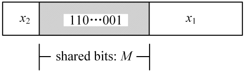
- Construct a set G of k-variate polynomial equations such that $g_i(y_1, \ldots, y_k) \equiv 0 \pmod{M}$;
- use the coefficient vectors of $g_i(x_1X_1, \ldots, x_kX_k)$, $i = 1, \ldots, k$, to construct a k-dimensional lattice $\mathcal{L}$;
- Applying the LLL algorithm to $\mathcal{L}$, we get a new set H of k polynomial equations $h_i(x_1, \ldots, x_k)$, $i = 1, \ldots, k$, with integer coefficients such that $h_i(y_1, \ldots, y_k) \equiv 0 \pmod{M}$;
- One can get $h_i(y_1, \ldots, y_k) = 0$ over the integers in some cases, where for $h(x_1, \ldots, x_k) = \sum_{i_1 \ldots i_k} a_{i_1 \ldots i_k} x_1^{i_1} \cdots x_1^{i_k}$

Proof of GIFP

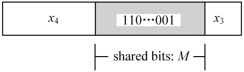
Proof.

Hence, we suppose that p_1 shares γn-bits from the $\beta_1 n$-th bit to $(\beta_1+\gamma)n$-th bit, and p_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 $\beta_1 \le \beta_2$ (see Fig. 1). Then we can write

$$p_1 = x_1 + M2^{\beta_1 n} + x_2 2^{(\beta_1+\gamma)n}, \quad p_2 = x_3 + M2^{\beta_2 n} + x_4 2^{(\beta_2+\gamma)n},$$



(a) p_1



(b) p_2

Figure: Shared bits M for p_1 and p_2

Proof of GIFP

Proof.

Next, we define the polynomial

$$f(x,y,z) = xz + 2^{(\beta_2+\gamma)n} yz + N_2,$$

which shows that $(x_1 2^{(\beta_2-\beta_1)n} - x_3, x_2 - x_4, q_2)$ is a solutions of

$$f(x,y,z) \equiv 0 \pmod{2^{(\beta_2-\beta_1)n} p_1}.$$

Proof of GIFP

Proof.

To apply Coppersmith's method, we consider a family of polynomials $g_{i,j}(x, y, z)$ for $0 \le i \le m$ and $0 \le j \le m - i$:

$$g_{i,j}(x, y, z) = (yz)^j f(x, y, z)^i \left(2^{(\beta_2 - \beta_1)n}\right)^{m-i} N_1^{\max(t-i,0)}.$$

Proof of GIFP

Proof.

These polynomials satisfy

$$
\begin{aligned}
& g_{i,j}\left(x_1 2^{(\beta_2-\beta_1)n} - x_3, x_2 - x_4, q_2\right) \\
& = (x_2 - x_4)^j q_2^j \left(2^{(\beta_2-\beta_1)n} p_1 q_2\right)^i \left(2^{(\beta_2-\beta_1)n}\right)^{m-i} N_1^{\max(t-i,0)} \\
& \equiv 0 \pmod{\left(2^{(\beta_2-\beta_1)n}\right)^m p_1^t}.
\end{aligned}
$$

Trick

Proof.

To reduce the determinant of the lattice, we introduce a new variable w for p_2, and multiply the polynomials $g_{i,j}(x, y, z)$ by a power w^s for some s that will be optimized later.
Similar to t, we also require $0 \leq s \leq m$

Trick

Proof.

Note that we can replace zw in $g_{i,j}(x,y,z)w^s$ by N_2.
We then eliminate $(zw)^i$ from the original polynomial by multiplying it by N_2^{-i}, while ensuring that the resulting polynomial evaluation is still a multiple of $\left(2^{(\beta_2-\beta_1)n}\right)^m p_1^t$.
By selecting the appropriate parameter s, we aim to reduce the determinant of the lattice.

Trick

Proof.

For example, suppose $m = 5$ and $t = 2$, then

$$\begin{aligned} g_{1,2}(x,y,z) =& (yz)^j f(x,y,z)^i \left(2^{(\beta_2-\beta_1)n}\right)^{m-i} N_1^{\max(t-i,0)} \\ =& (yz)^2 f(x,y,z)^1 \left(2^{(\beta_2-\beta_1)n}\right)^{5-1} N_1^{\max(2-1,0)} \\ =& (yz)^2 f(x,y,z) \left(2^{(\beta_2-\beta_1)n}\right)^{4} N_1 \end{aligned}$$

Trick

Proof.

For example, suppose $m = 5$ and $t = 2$, then

$$\begin{aligned} g_{1,2}(x,y,z) =& (yz)^j f(x,y,z)^i \left(2^{(\beta_2-\beta_1)n}\right)^{m-i} N_1^{\max(t-i,0)} \\ =& (yz)^2 f(x,y,z)^1 \left(2^{(\beta_2-\beta_1)n}\right)^{5-1} N_1^{\max(2-1,0)} \\ =& (yz)^2 f(x,y,z) \left(2^{(\beta_2-\beta_1)n}\right)^{4} N_1 \end{aligned}$$

Suppose $s = 2$, we multiply the polynomials $g_{1,2}(x,y,z)$ by a power $w^s = w^2$, then

$$\widetilde{g}_{1,2}(x,y,z,w) = (yz)^2 f(x,y,z) \left(2^{(\beta_2-\beta_1)n}\right)^{4} N_1 w^2$$

Trick

Proof.

See that

$$\begin{aligned}\widetilde{g}_{1,2}(x,y,z,w) =&(yz)^2 f(x,y,z)\left(2^{(\beta_2-\beta_1)n}\right)^4 N_1 w^2\\ =&(zw)^2 y^2 f(x,y,z)\left(2^{(\beta_2-\beta_1)n}\right)^4 N_1\end{aligned}$$

Trick

Proof.

See that

$$\begin{aligned}\widetilde{g}_{1,2}(x,y,z,w) =&(yz)^2 f(x,y,z)\left(2^{(\beta_2-\beta_1)n}\right)^4 N_1 w^2\\ =&(zw)^2 y^2 f(x,y,z)\left(2^{(\beta_2-\beta_1)n}\right)^4 N_1\end{aligned}$$

We then eliminate $(zw)^2$ from the original polynomial by multiplying it by N_2^{-2}, i.e.,

$$\begin{aligned}\overline{g}_{1,2}(x,y,z,w) =&\widetilde{g}_{1,2}(x,y,z,w) * N_2^{-2}\\ =&(zw)^2 y^2 f(x,y,z)\left(2^{(\beta_2-\beta_1)n}\right)^4 N_1 * N_2^{-2}\end{aligned}$$

Trick

Proof.

See that

$$\begin{aligned}\widetilde{g}_{1,2}(x,y,z,w) =&(yz)^2 f(x,y,z)\left(2^{(\beta_2-\beta_1)n}\right)^4 N_1 w^2\\ =&(zw)^2 y^2 f(x,y,z)\left(2^{(\beta_2-\beta_1)n}\right)^4 N_1\end{aligned}$$

We then eliminate $(zw)^2$ from the original polynomial by multiplying it by N_2^{-2}, i.e.,

$$\begin{aligned}\overline{g}_{1,2}(x,y,z,w) =&\widetilde{g}_{1,2}(x,y,z,w) * N_2^{-2}\\ =&(zw)^2 y^2 f(x,y,z)\left(2^{(\beta_2-\beta_1)n}\right)^4 N_1 * N_2^{-2}\end{aligned}$$

For simplicity, the results $\overline{g}_{1,2}(x,y,z,w)$ are denoted as $g_{1,2}(x,y,z,w)$.

Proof of GIFP

Proof.

Consider the lattice $\mathcal{L}$ spanned by the matrix $\mathbf{B}$ whose rows are the coefficients of the polynomials $g_{i,j}(x, y, z, w)$ for $0 \leq i \leq m$, $0 \leq j \leq m - i$.

Proof of GIFP

Proof.

Then

$$\det(\mathcal{L}) < \frac{1}{2^{\frac{\omega-1}{4}}\sqrt{\omega}} \left(2^{(\beta_2-\beta_1)n}\right)^{\omega m} p_1^{t\omega},$$

The inequality implies

$$\tau^2(3-\tau) - 3(1-\alpha)\tau + \sigma^3 - 3\alpha\sigma + 1 - \gamma + \alpha < 0.$$

The left side is optimized for $\tau_0 = 1 - \sqrt{\alpha}$ and $\sigma_0 = \sqrt{\alpha}$, which gives

$$\gamma > 4\alpha\left(1 - \sqrt{\alpha}\right).$$

Proof of GIFP

Proof.

By Assumption 1, we can get $(x_0, y_0, z_0) = (x_1 2^{(\beta_2-\beta_1)n} - x_3, x_2 - x_4, q_2)$, so we have $q_2 = z_0$, and we calculate

$$p_2 = \frac{N_2}{q_2}.$$

Proof of GIFP

Proof.

Next, we have

$$2^{(\beta_2-\beta_1)n}p_1 = p_2+(x_1 2^{(\beta_2-\beta_1)n}-x_3)+(x_2-x_4)2^{(\beta_2+\gamma)n} = p_2+y_0+z_0 2^{(\beta_2+\gamma)n}.$$

Therefore, we can calculate p_1 and $q_1 = \frac{N_1}{p_1}$. This terminates the proof. $\square$

Assumption

We used a famous assumption that has been mentioned in all previous work. In order to make our results more convincing, we also conducted some experiments

Assumption

We used a famous assumption that has been mentioned in all previous work. In order to make our results more convincing, we also conducted some experiments

Assumption

The k polynomials $h_i(x_1, \cdots, x_k)$, $i = 1, \cdots, k$, that are derived from the reduced basis of the lattice in the Coppersmith method are algebraically independent. Equivalently, the common root of the polynomials $h_i(x_1, \cdots, x_k)$ can be found by computing the resultant or computing the Gröbner basis.

Numerical results

The experiments were run on a computer configured with AMD Ryzen 5 2500U with Radeon Vega Mobile Gfx (2.00 GHz).

n	αn	βn	$\beta_1 n$	$\beta_2 n$	γn	m	$\dim(\mathcal{L})$	Time for LLL (s)	Time for Gröbner Basis (s)
200	20	40	20	30	140	6	28	1.8620	0.0033
200	20	60	20	30	140	6	28	1.8046	0.0034
500	50	100	50	75	350	6	28	3.1158	0.0043
500	50	150	50	75	300	6	28	4.23898	0.0048
1000	100	200	100	150	700	6	28	8.2277	0.0147

Table: Some experimental results for the GIFP.

Summary

In this paper, we considered the Generalized Implicit Factoring Problem (GIFP), where the shared bits are not necessarily required to be located at the same positions.

We proposed a lattice-based algorithm for this problem.

Open problem

Can we improve the bound $4\alpha\left(1-\sqrt{\alpha}\right)$ to $2\alpha\left(1-\alpha\right)$?

Reference I

[1] Alexander May and Maike Ritzenhofen. “Implicit Factoring: On Polynomial Time Factoring Given Only an Implicit Hint”. In: *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*. Ed. by Stanislaw Jarecki and Gene Tsudik. Vol. 5443. Lecture Notes in Computer Science. Springer, 2009, pp. 1–14. DOI: 10.1007/978-3-642-00468-1_1. URL: https://doi.org/10.1007/978-3-642-00468-1%5C_1.

[2] Santanu Sarkar and Subhamoy Maitra. “Further results on implicit factoring in polynomial time”. In: *Adv. Math. Commun.* 3.2 (2009), pp. 205–217. DOI: 10.3934/amc.2009.3.205. URL: https://doi.org/10.3934/amc.2009.3.205.

Reference II

[3] Jean-Charles Faugère, Raphaël Marinier, and Guénaël Renault. "Implicit Factoring with Shared Most Significant and Middle Bits". In: *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*. Ed. by Phong Q. Nguyen and David Pointcheval. Vol. 6056. Lecture Notes in Computer Science. Springer, 2010, pp. 70–87. DOI: 10.1007/978-3-642-13013-7_5. URL: https://doi.org/10.1007/978-3-642-13013-7%5C_5.

[4] Santanu Sarkar and Subhamoy Maitra. "Approximate Integer Common Divisor Problem Relates to Implicit Factorization". In: *IEEE Trans. Inf. Theory* 57.6 (2011), pp. 4002–4013. DOI: 10.1109/TIT.2011.2137270. URL: https://doi.org/10.1109/TIT.2011.2137270.

Reference III

[5] Yao Lu et al. "Towards optimal bounds for implicit factorization problem". In: *International Conference on Selected Areas in Cryptography*. Springer. 2016, pp. 462–476.

[6] Liqiang Peng et al. "Implicit Factorization of RSA Moduli Revisited (Short Paper)". In: *Advances in Information and Computer Security - 10th International Workshop on Security, IWSEC 2015, Nara, Japan, August 26-28, 2015, Proceedings*. Ed. by Keisuke Tanaka and Yuji Suga. Vol. 9241. Lecture Notes in Computer Science. Springer, 2015, pp. 67–76. DOI: 10.1007/978-3-319-22425-1_5. URL: https://doi.org/10.1007/978-3-319-22425-1%5C_5.

Reference IV

[7] Shixiong Wang et al. "A better bound for implicit factorization problem with shared middle bits". In: *Sci. China Inf. Sci.* 61.3 (2018), 032109:1–032109:10. DOI: 10.1007/s11432-017-9176-5. URL: https://doi.org/10.1007/s11432-017-9176-5.

[8] Miklós Ajtai. "The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract)". In: *Symposium on the Theory of Computing*. 1998.

[9] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. "Factoring polynomials with rational coefficients". In: *Mathematische annalen* 261.ARTICLE (1982), pp. 515–534.

Reference V

[10] Alexander May. "New RSA vulnerabilities using lattice reduction methods". PhD thesis. University of Paderborn, 2003. URL: http://ubdata.uni-paderborn.de/ediss/17/2003/may/disserta.pdf.

Thank you!