




# Implicit Factorization with Shared Any Bits

Chunzhi Zhao, Junqi Zhang, Jinzheng Cao , Qingfeng Cheng  and  
Fushan Wei 

Information Engineering University, Zhengzhou, China

**Abstract.** At PKC 2009, May and Ritzenhofen proposed the implicit factorization problem (IFP). They showed that it is undemanding to factor two  $h$ -bit RSA moduli  $N_1 = p_1q_1$ ,  $N_2 = p_2q_2$  where  $q_1, q_2$  are both  $\alpha h$ -bit, and  $p_1, p_2$  share  $uh > 2\alpha h$  the least significant bits (LSBs). Subsequent works mainly focused on extending the IFP to the cases where  $p_1, p_2$  share some of the most significant bits (MSBs) or the middle bits (MBs). In this paper, we propose a novel generalized IFP where  $p_1$  and  $p_2$  share an arbitrary number of bit blocks, with each block having a consistent displacement in its position between  $p_1$  and  $p_2$ , and we solve it successfully based on Coppersmith's method. Specifically, we generate a new set of shift polynomials to construct the lattice and optimize the structure of the lattice by introducing a new variable  $z = p_1$ . We derive that we can factor the two moduli in polynomial time when  $u > 2(n+1)\alpha(1 - \alpha^{\frac{1}{n+1}})$  with  $p_1, p_2$  sharing  $n$  blocks. Further, no matter how many blocks are shared, we can theoretically factor the two moduli as long as  $u > 2\alpha \ln(1/\alpha)$ . In addition, we consider two other cases where the positions of the shared blocks are arbitrary or there are  $k > 2$  known moduli. Meanwhile, we provide the corresponding solutions for the two cases. Our work is verified by experiments.

**Keywords:** RSA · Implicit factorization problem · Coppersmith's method

## 1 Introduction

In 1978, Rivest, Shamir, and Adleman [RSA78] proposed the famous public-key cryptographic algorithm RSA, whose security is based on the difficulty of factoring the integers  $N = pq$ , where  $p, q$  are two different large prime numbers. In general, to break RSA, we have to factor  $N$  directly, which is believed to be almost impossible to achieve with traditional computers for large  $N$ . However, if we are informed about the RSA system in part through other sources which are called side channels, we may crack it in an effective time. One of the most powerful tools for studying the security of RSA in side channels is lattice theory. In 1996, Coppersmith [Cop96] first proposed the idea of using lattice to attack RSA where he equates recovering the entire plaintext to solving a single-variable modular equation with some MSBs of the plaintext known. In detail, we need to construct a lattice and find a short vector in the lattice through the LLL algorithm [LLL82] for transforming the modular equation into an equation over the integers. Finally, we only need to solve this equation over the integers using numerical methods (e.g., Gröbner basis) to recover the whole information. After that, this method has gained further development

---

The work of Qingfeng Cheng (Corresponding Author) was supported by the National Natural Science Foundation of China under Grant Nos. 62472438 and 62172433, and the Natural Science Foundation of Henan Province under Grant No. 242300421414. The work of Fushan Wei was supported by the Science Foundation for the Excellent Youth Scholars of Henan Province under Grant No. 222300420099.

E-mail: [zhaochunzhi2022@126.com](mailto:zhaochunzhi2022@126.com) (Chunzhi Zhao), [zhangjunqi001@126.com](mailto:zhangjunqi001@126.com) (Junqi Zhang), [caojinzheng@126.com](mailto:caojinzheng@126.com) (Jinzheng Cao), [qingfengc2008@sina.com](mailto:qingfengc2008@sina.com) (Qingfeng Cheng), [weifs831020@163.com](mailto:weifs831020@163.com) (Fushan Wei)



[Cop97, BM03, JM06, MNS22]. We collectively refer to such methods as Coppersmith’s method.

At PKC 2009, May and Ritzenhofen [MR09] proposed the IFP for RSA. This problem is demonstrated as follows. Let  $N_1 = p_1q_1$ ,  $N_2 = p_2q_2$  be two different  $h$ -bit RSA moduli where  $q_1, q_2$  are both  $\alpha h$ -bit, and  $p_1, p_2$  share  $uh$  LSBs. Now we need to factor  $N_1, N_2$  efficiently. May et al. indicated that we can factor  $N_1, N_2$  in polynomial time when  $u > 2\alpha$ . Their work shows that the RSA system can be cracked with only an implicit hint, and this work can be naturally used to construct the backdoored RSA moduli, which is of great interest for studying the security of RSA. In 2010, Faugère et al. [FMR10] studied a variant of the IFP that  $p_1, p_2$  share some MSBs or MBs. They constructed a low-dimensional lattice based on the algebraic relation  $q_2N_1 - q_1N_2 = q_1q_2(p_1 - p_2)$  and used the LLL algorithm to reduce the lattice to obtain  $q_1, q_2$ , which in turn factors  $N_1, N_2$ . They showed that for the case of shared MSBs, we can factor  $N_1, N_2$  efficiently when  $u > 2\alpha$ , and for the case of shared MBs, we can efficiently factor  $N_1, N_2$  when  $u > 4\alpha$ . The works of May et al. and Faugère et al. are both non-Coppersmith lattice-based. At PKC 2023, Heninger et al. [HR23] gave a solution to a variant of the hidden number problem with small unknown multipliers, and they applied it to the IFP. Their approach is also non-Coppersmith lattice-based but can solve the IFP with a lower dimensional lattice than the prior non-Coppersmith approaches.

Table 1: Previous bounds and our result to the IFP

Case	LSBs	MSBs	MBs	Any bits
[MR09]	$2\alpha$	-	-	-
[FMR10]	-	$2\alpha$	$4\alpha$	-
[SM11, LZL13]	$2\alpha - \alpha^2$	$2\alpha - \alpha^2$	-	-
[SLH14]	-	-	-	$7\alpha$
[PHL <sup>+</sup> 15]	-	-	$4\alpha - 3\alpha^2$	-
[LPZ <sup>+</sup> 16]	$2\alpha - 2\alpha^2$	$2\alpha - 2\alpha^2$	-	-
[WQLF17, FNP24]	-	-	$4\alpha - 4\alpha^2$	-
Our work	$2\alpha - 2\alpha^2$	$2\alpha - 2\alpha^2$	$4\alpha - 4\alpha^2$	$2\alpha \ln \frac{1}{\alpha}$

In 2011, Sarkar and Maitra [SM11] transformed the IFP into an extended partially approximate common divisor problem (EPACDP) with  $p_1, p_2$  sharing MSBs, LSBs or both MSBs and LSBs. They solved this problem using Coppersmith’s method, and they indicated that we can factor  $N_1, N_2$  in polynomial time when  $u > 2\alpha - \alpha^2$ . Furthermore, they extended the analysis to the case of  $k > 2$  moduli. In 2013, Lu et al. [LZL13] used different shift polynomials to construct the lattice, which improves the bounds on the three cases in [SM11] to  $1 - (1 - \alpha)^{\frac{k}{k-1}}$ . Their results are the same as [SM11] when  $k = 2, 3$ , but better than [SM11] when  $k > 3$ . In 2014, with the method [HM08] for solving linear equations modulo unknown divisors, Peng et al. [PHX<sup>+</sup>14] improved the bound on the cases where  $p_1, p_2$  share LSBs or MSBs to  $4 - 4\alpha - 4(1 - \alpha)^{\frac{3}{2}}$ . In 2014, Shi et al. [SLH14] extended the IFP to the case where  $p_1, p_2$  share an arbitrary number of bit blocks which are aligned in  $p_1$  and  $p_2$ . They derived that  $N_1, N_2$  can be factored in polynomial time as long as  $u > 7\alpha$ . In 2015, Peng et al. [PHL<sup>+</sup>15] used the method in [SM11] to reanalyze the case where  $p_1, p_2$  share MBs. They showed that we can factor  $N_1, N_2$  in polynomial time when  $u > 4\alpha - 3\alpha^2$ . Meanwhile, Nitaj et al. [NA15] proposed a variant of the IFP that  $a_1p_1, a_2p_2$  share some MSBs or LSBs where  $a_1, a_2$  are both integers. Furthermore, they extended this problem to  $k > 2$  moduli and successfully solved the problem using the continued fraction algorithm. In 2016, Lu et al. [LPZ<sup>+</sup>16] reanalyzed the three cases in [SM11] by introducing a new variable  $z = p_2$  to reduce the determinant of the lattice, and they presented a better bound  $u > 2\alpha - 2\alpha^2$ . They also extended their work to  $k > 2$  moduli. In 2017, Wang et al. [WQLF17] gave a better bound on the case of shared MBs

by introducing a new variable that is similar to [LPZ<sup>+</sup>16]. They improved the bound to  $u > 4\alpha(1 - \sqrt{\alpha})$ . In 2019, Sun et al. [SZZ<sup>+</sup>19] resolved the variant of the IFP proposed by Nitaj et al. [NA15] using Coppersmith’s method. With  $k = 2$  moduli and relatively small  $a_1, a_2$ , they improved the result in [NA15]. And their result is generally better than [NA15] when  $k > 3$ . In 2023, Feng et al. [FNP24] proposed a generalized IFP in which  $p_1, p_2$  share some bits at different positions. They successfully solved the problem using Coppersmith’s method with the same technique for reducing the determinant of the lattice as [LPZ<sup>+</sup>16], which yields a bound consistent with [WQLF17]. Moreover, Zheng [Zhe23] proposed a generalized implicit-key attack on RSA which can be considered as a variant of the IFP. Table 1 briefly summarizes the pre-existing bounds and our result on the IFP for different cases.

**Our contributions.** In this paper, we propose a new generalized IFP in which the two factors  $p_1, p_2$  share  $n \geq 2$  bit blocks where each block has one fixed displacement between its positions in  $p_1$  and  $p_2$ , i.e. if we denote  $pos_{i1}$  and  $pos_{i2}$  as the position of the  $i$ -th shared block in  $p_1$  and  $p_2$ , respectively, then all the  $pos_{i1} - pos_{i2}$  are equal. We solve this problem based on Coppersmith’s method. To be specific, we generate a new set of shift polynomials based on one basic algebraic relation and construct the lattice with the technique of introducing a new variable  $z = p_1$  which is proposed by Lu et al. We derived that we can factor the two moduli in polynomial time when  $u > 2(n + 1)(\alpha - \alpha^{\frac{n+2}{n+1}})$  for  $n$  shared blocks. And restricted to the three cases LSBs, MSBs, and MBs, our approach is the same as the corresponding current optimal approach. Further, no matter how many bit blocks  $p_1, p_2$  share, we can theoretically factor the two moduli in polynomial time as long as  $u > 2\alpha \ln(1/\alpha)$ .

Moreover, based on the simple situation described above, we consider two other cases where the positions of the shared blocks in  $p_1, p_2$  are arbitrary or there are  $k > 2$  moduli. For the former case, we give a solution for solving a system of modular equations that share one common variable; For the latter case, we solve the problem using a method similar to Wang et al.’s. Both methods use the technique of introducing new variables to optimize the structure of the lattice, which leads to a lower determinant.

**Motivation of our work.** Because information intercepted in the real world usually has a discrete and random structure, especially for some secret information, researchers are committed to adapting side-channel attacks to these conditions. For instance, at Asiacrypt 2008, Herrmann and May [HM08] extended the problem of *factoring with high bits known* to  $n$  ( $n \geq 1$ ) unknown blocks; at Indocrypt 2011, Sarkar [Sar11] extended partial key exposure attacks on RSA to  $n$  ( $n \geq 1$ ) leaked blocks. Certainly, the principle also applies to the IFP. Related work includes the studies of Shi et al. [SLH14] and Feng et al. [FNP24], which extended the IFP to the cases of multiple blocks and different block positions, respectively. Our work has been inspired by these prior papers.

There are several potential applications for research on generalizing the IFP. May et al. [MR09] described an attack where the attacker can fix certain registers of an RSA public key generator, allowing the attacker to gather RSA moduli whose factors share some LSBs. However, practical limitations may prevent the attacker from controlling registers at will, leading to a decentralized structure of controlled bits. Our study provides a useful framework for such scenarios. Further, when storing numerous RSA private keys, one can reduce storage requirements by fixing certain bits in the factors. By storing only the randomly generated bits (excluding the fixed ones), the storage footprint is minimized. Generally, the more blocks there are, and the more randomly they are placed, the more challenging it becomes to factor the moduli. Thus, to enhance the security, users can divide the fixed bits into several blocks, with their positions within the factors being arbitrary, rather than locking a single large block of consecutive bits. The only additional overhead is storing the positions of the blocks within the factors, which requires minimal space. Unfortunately, the attacker could still recover the factors using our method in such

a scenario. Furthermore, our work offers a novel approach to constructing backdoored RSA moduli, which, due to their decentralized structure and arbitrary bit positions, are more sophisticated than previous ones. This poses a serious risk, as attackers could create backdoors that align with our extensions, reducing the likelihood of detection.

In summary, research on generalizing the IFP will not only help mitigate security risks associated with RSA but also provide insights into the construction and analysis of backdoored RSA moduli. Therefore, it is natural and necessary to develop algorithms for analyzing the IFP in more general cases.

**Organization of the paper.** This paper is organized as follows. In Section 2, we recall some basic theories. In Section 3, we demonstrate our work in detail, including the main conclusion and its proof process. In Section 4, we experimentally verify the validity of our method. In Section 5, we give a brief overview of our work.

## 2 Preliminaries

For any integers  $a, b$  ( $a \leq b$ ), we let  $[a : b]$  denote the set  $\{a, a + 1, \dots, b\}$ . Lattice is a discrete subgroup of  $\mathbb{R}^n$ . To be specific, a lattice is the set of all integer linear combinations of basis vectors (linearly independent).

**Definition 1** (Lattice). Let  $v_1, v_2, \dots, v_m$  ( $m \leq n$ ) be  $m$  linearly independent vectors in  $\mathbb{R}^n$ , then we call  $\mathcal{L}$  the lattice spanned by  $\{v_1, v_2, \dots, v_m\}$  if

$$\mathcal{L} = \left\{ \sum_{i=1}^m z_i \cdot v_i \mid z_i \in \mathbb{Z}, \text{ for } i \in [1 : m] \right\}.$$

How to efficiently find short vectors in a lattice is what researchers have been investigating. The most widely used algorithms for finding short vectors within a lattice are known as lattice basis reduction algorithms. These algorithms involve a sequence of integer linear transformations applied to the initial lattice basis, resulting in a new basis with shorter vectors. As one of the most widely used lattice reduction algorithms, the LLL algorithm can find an approximate shortest vector in polynomial time. The following conclusion is about the length of vectors the LLL algorithm outputs.

**Theorem 1** (see [May03]). Let  $\mathcal{L}$  be an integer lattice of dimension  $\omega$ . The LLL algorithm outputs a reduced basis spanned by  $\{v_1, v_2, \dots, v_\omega\}$  with

$$\|v_1\| \leq \|v_2\| \leq \dots \leq \|v_i\| \leq 2^{\frac{\omega(\omega-1)}{4(\omega+1-i)}} \det(\mathcal{L})^{\frac{1}{\omega+1-i}},$$

for  $i \in [1 : \omega]$  in polynomial time, where  $\|\cdot\|$  denotes the  $\ell_2$  norm.

Coppersmith's method is a powerful tool for solving modular equations. The core of Coppersmith's method is transferring the modular equations into equations over the ring of integer using the LLL algorithm and finally solving the equation with numerical methods such as Newton iteration method, Gröbner basis, etc.

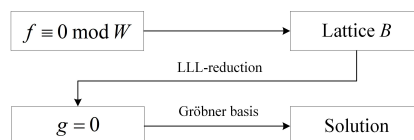


Figure 1: The main process of Coppersmith's method

Howgrave-Graham [HG97] established a sufficient condition for transforming modular roots into roots over the integers, leading to the following theorem.

**Theorem 2.** Let  $g(x_1, \dots, x_n) \in \mathbb{Z}[x_1, \dots, x_n]$  be an integer polynomial consisting of no more than  $\omega$  monomials, where  $x_1, \dots, x_n$  are algebraically independent. Assuming that  $x_i^0$  and  $X_i$  are constants for all  $i \in [1 : n]$ , if

1.  $g(x_1^0, \dots, x_n^0) = 0 \pmod{W^m}$  for  $|x_1^0| \leq X_1, |x_2^0| \leq X_2, \dots, |x_n^0| \leq X_n$ ,

2.  $\|g(x_1 X_1, \dots, x_n X_n)\| < \frac{W^m}{\sqrt{\omega}}$ ,

then  $g(x_1^0, \dots, x_n^0) = 0$  holds over the integers.

To solve for the small roots successfully, we give the following proposition.

**Proposition 1.** The lattice constructed in this paper yields algebraically independent polynomials. The common roots of these polynomials can be efficiently computed using techniques like Gröbner basis.

It is worth noting that Proposition 1 may sometimes fail, depending on the structure of the initial polynomial for generating the lattice.

### 3 Implicit Factorization Problem for $n$ blocks

In this section, we demonstrate our main work. Firstly, we introduce a new variant of the IFP in Subsection 3.1 where the two factors  $p_1, p_2$  share  $n \geq 2$  continuous bit blocks where each block has one fixed displacement between its positions in  $p_1$  and  $p_2$ . We solve this problem using Coppersmith’s method where we introduce a new variable  $z = p_1$  to optimize the structure of the lattice which is similar to [LPZ<sup>+</sup>16]. Then we extend this problem to the case where the positions of the shared bit blocks are arbitrary, and give the solution in Subsection 3.2. Further, we consider this problem in the case of  $k > 2$  moduli and give the solution in Subsection 3.3.

#### 3.1 Analysis of Ordered Position Case for Two RSA Moduli

In this subsection, we focus on a generalized IFP where  $N_1 = p_1 q_1, N_2 = p_2 q_2$  are two  $h$ -bit RSA moduli where  $q_1, q_2$  are both  $\alpha h$ -bit numbers, and  $p_1, p_2$  share  $n \geq 2$  continuous bit blocks. Each shared block has one fixed displacement between its positions in  $p_1$  and  $p_2$  (see Fig. 2). Now we try to factor  $N_1, N_2$ .

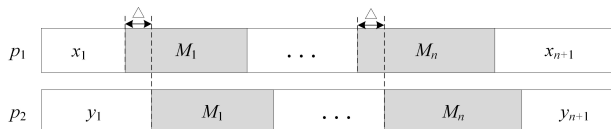


Figure 2: Shared bits of  $p_1$  and  $p_2$

**Theorem 3.** Let  $N_1 = p_1 q_1, N_2 = p_2 q_2$  be two  $h$ -bit RSA moduli where  $q_1, q_2$  are both  $\alpha h$ -bit, and  $p_1, p_2$  share  $n$  ( $n \geq 2$ ) continuous bit blocks. Each shared block has one fixed displacement between its positions in  $p_1$  and  $p_2$ . Under Proposition 1, one is able to factor  $N_1, N_2$  in polynomial time if

$$u > 2(n + 1)\alpha(1 - \alpha^{\frac{1}{n+1}}),$$

where  $uh$  is the number of all the bits shared by  $p_1$  and  $p_2$ , which is the cumulative count of bits present in either prime.

*Proof.* Under the presentation of Fig. 2, we let

$$M = M_1 \cdot 2^{a_1 h} + M_2 \cdot 2^{a_2 h} + \dots + M_n \cdot 2^{a_n h},$$

$$M' = M_1 \cdot 2^{a'_1 h} + M_2 \cdot 2^{a'_2 h} + \dots + M_n \cdot 2^{a'_n h},$$

where  $a_i h$  and  $a'_i h$  are the positions of the lowest bit of  $M_i$  in  $p_1$  and  $p_2$ , respectively, for  $i \in [1 : n]$ . Then we can write

$$\begin{aligned} p_1 &= x_{n+1} + x_n \cdot 2^{t_n h} + x_{n-1} \cdot 2^{t_{n-1} h} + \cdots + x_1 \cdot 2^{t_1 h} + M, \\ p_2 &= y_{n+1} + y_n \cdot 2^{t'_n h} + y_{n-1} \cdot 2^{t'_{n-1} h} + \cdots + y_1 \cdot 2^{t'_1 h} + M', \end{aligned}$$

where  $t_i h$  and  $t'_i h$  are the positions of the lowest bit of  $x_i$  and  $y_i$ , respectively, for  $i \in [1 : n]$ . Without loss of generality, we assume that  $t_i \geq t'_i$  for  $i \in [1 : n]$ . According to the given condition, it is clear that  $t_i - t'_i = t_j - t'_j = a_i - a'_i$  for  $i, j \in [1 : n]$ , therefore we set  $\Delta = t_i - t'_i = a_i - a'_i$  for  $i \in [1 : n]$ . Then we have

$$2^{h\Delta} p_2 - p_1 = \sum_{j=1}^n 2^{t_j h} (y_j - x_j) + 2^{h\Delta} y_{n+1} - x_{n+1}. \quad (1)$$

Multiply both sides of Eq. (1) by  $q_1$ , there is

$$2^{h\Delta} p_2 q_1 = N_1 + \left( \sum_{j=1}^n 2^{t_j h} (y_j - x_j) + 2^{h\Delta} y_{n+1} - x_{n+1} \right) q_1. \quad (2)$$

Let  $\theta_i = y_i - x_i$  ( $i \in [1 : n]$ ),  $\theta_{n+1} = 2^{h\Delta} y_{n+1} - x_{n+1}$ , then

$$N_1 + (\theta_1 \cdot 2^{t_1 h} + \theta_2 \cdot 2^{t_2 h} + \cdots + \theta_n \cdot 2^{t_n h} + \theta_{n+1}) q_1 \equiv 0 \pmod{2^{h\Delta} p_2}.$$

We define  $f(\vec{r}) = N_1 + (r_1 \cdot 2^{t_1 h} + r_2 \cdot 2^{t_2 h} + \cdots + r_n \cdot 2^{t_n h} + r_{n+1}) r_{n+2}$ , and it is clear that  $\vec{\theta}$  is a root of  $f(\vec{r}) \pmod{2^{h\Delta} p_2}$ . Here,  $\vec{r}$  and  $\vec{\theta}$  denote vectors  $(r_1, r_2, \dots, r_{n+2})$  and  $(\theta_1, \dots, \theta_{n+1}, q_1)$ , respectively. Given integer parameters  $m$  and  $t$ , we consider the following cluster of polynomials

$$g_{\vec{i},k}(\vec{r}) = 2^{(m-k)h\Delta} \prod_{j=1}^n (r_j r_{n+2})^{i_j} f^k(\vec{r}) N_2^{\max\{t-k,0\}},$$

where  $i_j, k \in [0 : m]$  such that  $k + \sum_{j=1}^n i_j \leq m$  and  $\vec{i}$  denotes the vector  $(i_1, i_2, \dots, i_n)$ . Moreover,  $t = \lfloor \tau m \rfloor$  ( $0 < \tau < 1$ ) will be optimized later. According to Eq. (2),  $f(\vec{\theta}) = 2^{h\Delta} p_2 q_1$ . Therefore, it is clear that

$$\begin{aligned} g_{\vec{i},k}(\vec{\theta}) &\equiv 2^{(m-k)h\Delta} f^k(\vec{\theta}) N_2^{\max\{t-k,0\}} \prod_{j=1}^n (\theta_j q_1)^{i_j} \\ &\equiv 2^{hm\Delta} q_1^{k + \sum_{j=1}^n i_j} p_2^{k + \max\{t-k,0\}} q_2^{\max\{t-k,0\}} \prod_{j=1}^n \theta_j^{i_j} \\ &\equiv 0 \pmod{2^{hm\Delta} p_2^t}. \end{aligned}$$

Generally, we directly use the  $g_{\vec{i},k}(\vec{r})$ 's to construct the lattice. However, we can optimize the structure of the lattice by introducing a new variable  $z = p_1$ . To be specific, we multiply each polynomial  $g_{\vec{i},k}(\vec{r})$  by  $z^l$  and replace each occurrence of the monomial  $z r_{n+2}$  by  $N_1$ , then eliminate it by multiplying  $E = N_1^{-1} \pmod{2^{hm\Delta} N_2^t}$ . As a result, we can obtain

$$g'_{\vec{i},k}(\vec{r}, z) = 2^{(m-k)h\Delta} E^T z^l \prod_{j=1}^n (r_j r_{n+2})^{i_j} f^k(\vec{r}) N_2^{\max\{t-k,0\}},$$

where  $l = \lfloor \sigma m \rfloor$  ( $0 < \sigma < 1$ ) will be optimized later, and  $T = \min\{l, k + \sum_{j=1}^n i_j\}$ . Meanwhile, there is

$$|\theta_i| < 2^{\max\{b(y_i), b(x_i)\}^1} = X_i \quad (i \in [1 : n + 1]), |q_1| < 2^{\alpha h} = X_{n+2}, |p_1| < 2^{(1-\alpha)h} = Z,$$

where  $X_i$  ( $i \in [1 : n + 1]$ ) and  $Z$  are defined.

To construct the lattice, we define the monomials' order:  $\left(\prod_{j=1}^n r_j^{i_j}\right) r_{n+1}^k r_{n+2}^{k+\sum_{j=1}^n i_j} z^l \prec \left(\prod_{j=1}^n r_j^{i'_j}\right) r_{n+1}^{k'} r_{n+2}^{k'+\sum_{j=1}^n i'_j} z^l$  if and only if  $k < k'$  or  $k = k', i_n < i'_n$  or  $\dots$  or  $k = k', i_n = i'_n, i_{n-1} = i'_{n-1}, \dots, i_2 = i'_2, i_1 < i'_1$ . Under this, the leading monomial of  $g'_{i,k}(\vec{r}, z)$  is

$$r_{n+1}^k r_{n+2}^{k+\sum_{j=1}^n i_j - \min\{l, k+\sum_{j=1}^n i_j\}} z^{l-T}.$$

Sort the leading monomials of all  $g'_{i,k}(\vec{r}, z)$  in the order specified above, yielding a list of monomials represented by the column vector  $\chi$ . We consider the polynomials  $g'_{i,k}(\vec{X})$ 's, where  $\vec{X}$  denotes the vector  $(r_1 X_1, \dots, r_{n+2} X_{n+2}, z Z)$ . These  $g'_{i,k}(\vec{X})$ 's are arranged in the order of their leading monomials. Subsequently, from each  $g'_{i,k}(\vec{X})$ , we sequentially extract the coefficients of all monomials in  $\chi$  to form row vectors (if a monomial does not appear in  $g'_{i,k}(\vec{X})$ , its coefficient is considered to be 0). We utilize these row vectors to assemble the lattice basis matrix  $B$ , meaning that the lattice is generated by this set of vectors. Table 2 demonstrates an example of the basis matrix  $B$ . The dimension of  $B$  is  $d = \binom{n+m+1}{m}$ . As shown in Table 2, the basis matrix we constructed is lower triangular. Therefore, the determinant of matrix  $B$  is equal to the product of its diagonal entries. Let  $\det(B)$  denote the determinant of matrix  $B$ , and we can write  $\det(B) = (2^{h\Delta})^{s_\Delta} \cdot \prod_{i=1}^{n+1} X_i^{s_i} \cdot X_{n+2}^{s_{n+2}} \cdot N_2^{s_{N_2}} \cdot Z^{s_z}$ . For this, we have

$$\begin{aligned} s_\Delta &= \sum_{k=0}^m (m-k) \binom{m-k+n}{m-k} = m \binom{m+n+1}{m} - \binom{m+n+1}{m-1}, \\ s_i &= \sum_{k=0}^m k \binom{m-k+n}{m-k} = \binom{n+m+1}{m-1}, \text{ for } i \in [1 : n+1], \\ s_{n+2} &= \sum_{k=l+1}^m (k-l) \binom{k+n}{k} = (m-l) \binom{m+n+1}{m} - \binom{m+n+1}{m-1} + \binom{l+n+1}{l-1}, \\ s_z &= dl - \left[ \sum_{k=0}^m k \binom{k+n}{k} - s_{n+2} \right] = \binom{l+n+1}{l-1}, \\ s_{N_2} &= \sum_{k=0}^{t-1} (t-k) \binom{m-k+n}{m-k} = t \binom{m+n+1}{m} - \binom{m+n+1}{m-1} + \binom{m-t+n+1}{m-t-1}. \end{aligned}$$

According to Theorem 2, in order to recover the small root  $\vec{\theta}$ , it needs

$$2^{\frac{d(d-1)}{4(d-n-1)}} \cdot \det(B)^{\frac{1}{d-n-1}} < \frac{2^{hm\Delta}}{\sqrt{d}}.$$

Substituting  $\det(B) = (2^{h\Delta})^{s_\Delta} \cdot \prod_{i=1}^{n+1} X_i^{s_i} \cdot X_{n+2}^{s_{n+2}} \cdot N_2^{s_{N_2}} \cdot Z^{s_z}$  into the above equation and simplifying the equation gives

$$2^{\frac{d(d-1)}{4}} d^{\frac{d-n-1}{2}} 2^{h\Delta s_\Delta} \prod_{i=1}^{n+1} X_i^{s_i} X_{n+2}^{s_{n+2}} N_2^{s_{N_2}} Z^{s_z} < 2^{hm\Delta(d-n-1)} \cdot p_2^{t(d-n-1)}. \quad (3)$$

<sup>1</sup> $b(x)$  represents the number of bits of  $x$

Table 2: The basis matrix with  $n = 2, m = 2, l = 2, t = 1$ 

$g_{i_1, i_2, k}$	$z^2$	$r_1 z$	$r_1^2$	$r_2 z$	$r_1 r_2$	$r_2^2$	$r_3 z$	$r_1 r_3$	$r_2 r_3$	$r_3^2$
$g_{0,0,0}$	$2^{2h\Delta} N_2 Z^2$									
$g_{1,0,0}$		$2^{2h\Delta} N_2 X_1 Z$								
$g_{2,0,0}$			$2^{2h\Delta} N_2 X_1^2$							
$g_{0,1,0}$				$2^{2h\Delta} N_2 X_2 Z$						
$g_{1,1,0}$					$2^{2h\Delta} N_2 X_1 X_2$					
$g_{0,2,0}$						$2^{2h\Delta} N_2 X_2^2$				
$g_{0,0,1}$	*	*	*	*			$2^{h\Delta} X_3 Z$			
$g_{1,0,1}$		*	*	*	*			$2^{h\Delta} X_1 X_3$		
$g_{0,1,1}$				*	*	*			$2^{h\Delta} X_2 X_3$	
$g_{0,0,2}$	*	*	*	*	*	*	*	*	*	$X_3^2$

“\*” denotes non-zero entries; blank elements denote zero entries.



For the convenience of analysis, we assume that  $N_2 = 2^h$ . We know that  $N_2$  is an  $h$ -bit number, which means that  $2^{h-1} \leq N_2 < 2^h$ . Therefore, if Eq. (3) holds for  $N_2 = 2^h$ , then it also holds naturally for  $N_2$  taking the real value. In other words, we assume that  $N_2 = 2^h$  will not affect the result in this proof. Meanwhile, because  $1 < 2^h/N_2 < 2$ , the error caused by this assumption is very limited. Specifically, let  $2^h/N_2 = 2$ , and at this point, the error in the power of 2 into which the left side of Eq. (3) is transformed reaches its maximum value of about  $1/h$ . Given that RSA moduli are generally set to be large, the error  $1/h$  can be neglected. Further, we can calculate that  $\prod_{i=1}^{n+1} X_i \approx 2^{[(1-\alpha)-u+\Delta]h}$ , then we simplify Eq. (3) to

$$\begin{aligned} & 2^h \left[ \binom{l+n+1}{l-1} - (2\alpha+u) \binom{m+n+1}{m-1} + \text{con}_1 \binom{m+n+1}{m} + \binom{m-t+n+1}{m-t-1} \right] \cdot 2^{\frac{(m+n+1)^2 - \binom{m+n+1}{m}}{4}} \\ & < 2^{[hm\Delta+t(1-\alpha)h] \left( \binom{m+n+1}{m} \right)^{-n-1}} / \binom{m+n+1}{m}^{\frac{(m+n+1)^{-n-1}}{2}}, \end{aligned}$$

where  $\text{con}_1 = (\Delta + \alpha)m + t - \alpha l$ . On the other hand, we know

$$\begin{aligned} \binom{m+n+1}{m} &= \frac{(m+n+1)!}{m!(n+1)!}, \quad \binom{l+n+1}{l-1} = \frac{(l+n+1)!}{(l-1)!(n+2)!}, \\ \binom{m+n+1}{m-1} &= \frac{(m+n+1)!}{(m-1)!(n+2)!}, \quad \binom{m-t+n+1}{m-t-1} = \frac{(m-t+n+1)!}{(m-t-1)!(n+2)!}, \end{aligned}$$

then after simplification, we have

$$2^{h(\text{con}_1 + \text{con}_2 + \text{con}_3) + \frac{(m+n+1)!^{-1}}{m!(n+1)!^4}} < \frac{2^{h[m\Delta+t(1-\alpha)] \left( 1 - \frac{(n+1)(n+1)!m!}{(m+n+1)!} \right)}}{\left( \frac{(m+n+1)!}{m!(n+1)!} \right)^{\frac{1}{2} - \frac{(n+1)(n+1)!m!}{2(m+n+1)!}}},$$

where  $\text{con}_2 = \frac{(l+n+1)!m! - (l-1)!(m+n+1)!(2\alpha+u)m}{(l-1)!(m+n+1)!(n+2)}$ ,  $\text{con}_3 = \frac{(m-t+n+1)!m!}{(m-t-1)!(m+n+1)!(n+2)}$ . Comparing the powers of 2 on both sides of the above equation and removing negligible small items, we can obtain

$$\frac{\left[ \frac{(l+n+1)!}{(l-1)!} + \frac{(m-t+n+1)!}{(m-t-1)!} - (2\alpha+u)m \right] (m-1)!}{(n+2)(m+n+1)!} + (1+\tau-\sigma)\alpha < 0, \quad (4)$$

after simplification. Based on the fundamental inequality  $\sqrt[n]{x_1 \cdots x_n} \leq \frac{x_1 + \cdots + x_n}{n}$ , we can rewrite

$$\frac{(l+n+1)!(m-1)!}{(l-1)!(m+n+1)!} = \sigma \prod_{s=m+1}^{m+n+1} \left( 1 + \frac{l-m}{s} \right) \leq \sigma \left( 1 + \frac{l-m}{n+1} \cdot \sum_{s=m+1}^{m+n+1} \frac{1}{s} \right)^{n+1}. \quad (5)$$

Here are two facts: Firstly, for any positive integers  $I_1 > I_2$ , the integral  $\int_{I_2}^{I_1} 1/x dx$  is greater than the sum  $\sum_{i=I_2+1}^{I_1} 1/i$ , i.e.  $\ln(I_1/I_2) = \ln(I_1) - \ln(I_2) = \int_{I_2}^{I_1} 1/x dx > \sum_{i=I_2+1}^{I_1} 1/i$ . Secondly, for any positive  $x$ , the natural logarithm satisfies  $\ln(1+x) < x$ . Therefore, we can obtain

$$\left( 1 + \frac{l-m}{n+1} \cdot \sum_{s=m+1}^{m+n+1} \frac{1}{s} \right)^{n+1} < \left( 1 + \frac{(\sigma-1)m}{n+1} \cdot \ln \left( 1 + \frac{n+1}{m} \right) \right)^{n+1} \leq \sigma^{n+1},$$

and combining Eq. (5), there is

$$\frac{(l+n+1)!(m-1)!}{(l-1)!(m+n+1)!} < \sigma^{n+2}. \quad (6)$$

Similarly, we can obtain

$$\frac{(m-t+n+1)!(m-1)!}{(m-t-1)!(m+n+1)!} < (1-\tau)^{n+2}. \quad (7)$$

Substituting Eq. (6) and Eq. (7) into Eq. (4), after simplification, we have

$$u > \sigma^{n+2} + (1-\tau)^{n+2} + [(1+\tau-\sigma)(n+2) - 2]\alpha.$$

Let  $\sigma$  and  $\tau$  take the optimized value  $\sigma_0 = \alpha^{\frac{1}{n+1}}$  and  $\tau_0 = 1 - \alpha^{\frac{1}{n+1}}$ , respectively, then we have

$$u > 2(n+1)\alpha(1 - \alpha^{\frac{1}{n+1}}).$$

This terminates the proof.  $\square$

After the basis matrix  $B$  is LLL-reduced, a new basis matrix  $B'$  is obtained. Generally,  $B'[i]$  is converted into a polynomial  $\zeta_i(\vec{r}, z)$  by multiplying it with  $\chi$ , for  $i \in [1, n+3]$ , where  $B'[i]$  is the  $i$ -th row vector of  $B'$ . Then, let  $\xi_i(\vec{r}, z) = \zeta_i(r_1/X_1, \dots, r_{n+2}/X_{n+2}, z/Z)$  for  $i \in [1, n+3]$ , and  $\xi_i(\vec{\theta}, p_1) = 0$  ( $i \in [1, n+3]$ ) hold over the integers. According to Proposition 1, we can recover the small roots  $\vec{\theta}$  by solving the system of equations  $\xi_i(\vec{r}, z) = 0$  ( $i \in [1, n+3]$ ). Unfortunately, Proposition 1 usually fails for the lattice we construct. In detail, the polynomials obtained after the LLL-reduction are generally algebraically independent over  $\mathbb{Z}[r_{n+2}]$  but related over  $\mathbb{Z}$ . This may be because each occurrence of  $r_j$  is accompanied by an occurrence of  $r_{n+2}$  in the monomials so that we can consider  $r_j r_{n+2}$  as one variable for  $j \in [1 : n+1]$ . As a result, these polynomials are related to  $r_{n+2}$ . In this case, we can't obtain the desired solution with the general method. For this, we give another method to factor  $N_1, N_2$ . Specifically, we can utilize the structure of the lattice we have constructed to reduce the number of variables so that we may solve the equations successfully. Based on the structure of the lattice, we know that each monomial of  $\xi_i(\vec{r}, z)$  has a form of

$$r_{n+1}^k r_{n+2}^{k+\sum_{j=1}^n i_j - T} z^{l-T} \prod_{j=1}^n r_j^{i_j}. \quad (8)$$

Meanwhile, we know that  $r_{n+2}z = N_1$ , so multiplying Eq. (8) with  $r_{n+2}^l$ , we can get

$$\begin{aligned} r_{n+1}^k r_{n+2}^{k+\sum_{j=1}^n i_j + l - T} z^{l-T} \prod_{j=1}^n r_j^{i_j} &= r_{n+1}^k r_{n+2}^{k+\sum_{j=1}^n i_j} (r_{n+2}z)^{l-T} \prod_{j=1}^n r_j^{i_j} \\ &= r_{n+1}^k r_{n+2}^{k+\sum_{j=1}^n i_j} N_1^{l-T} \prod_{j=1}^n r_j^{i_j} \\ &= \left( \prod_{j=1}^n (r_j r_{n+2})^{i_j} \right) (r_{n+1} r_{n+2})^k N_1^{l-T}. \end{aligned}$$

Let  $c_i = r_i r_{n+2}$  for  $i \in [1 : n+1]$ , then we get new monomials with  $n+1$  variables:  $c_1^{i_1} c_2^{i_2} \dots c_n^{i_n} c_{n+1}^k N_1^{l-T}$ . Let  $\tilde{\xi}_i(\vec{c}) = r_{n+2}^l \xi_i(\vec{r}, z)$ , we can get the root  $\vec{C}$  of  $\tilde{\xi}_i(\vec{c})$  using Gröbner basis, for  $i \in [1 : n+1]$ , where  $\vec{C}$  and  $\vec{c}$  denote vectors  $(C_1, C_2, \dots, C_{n+1})$  and  $(c_1, c_2, \dots, c_{n+1})$ , respectively. In experiments, we can almost 100% obtain the small roots in this way. In reality, we obtained the small roots with appropriate parameter  $m$  for all experiments. However, during the experiments, we sometimes failed to obtain the desired roots due to  $m$  being set too small. For instance, in the experiment with  $n = 4, h = 2000, \alpha = 0.11, \Delta = 100$ , the algorithm failed when  $m$  was set to 3, but succeeded when  $m$  was increased to 4. Generally, the more unknowns and the larger their

values, the larger  $m$  must be. Consequently, careful selection of the parameter  $m$  is crucial. After getting  $\vec{C}$ , we can get  $q_1 = \text{GCD}(N_1, C_1)$ , then according to Eq. (2), there is

$$N_1 + C_1 \cdot 2^{t_1 h} + C_2 \cdot 2^{t_2 h} + \dots + C_n \cdot 2^{t_n h} + C_{n+1} = 2^{h\Delta} p_2 q_1.$$

The left side of the above equation and  $q_1$  are both known, therefore we are able to recover  $p_2$  implying that we can factor  $N_1, N_2$ .

Interestingly, as  $n$  tends to infinity, the bound in Theorem 3 tends to be  $2\alpha \ln(1/\alpha)$ . This means that we can factor  $N_1, N_2$  with an arbitrary number of shared blocks as long as  $u > 2\alpha \ln(1/\alpha)$ . The time complexity of our algorithm mainly depends on the time complexity of the LLL algorithm, and the  $L^2$  algorithm [NS05], a floating-point variant of the LLL algorithm, is widely used for lattice basis reduction in practice. The time complexity of the  $L^2$  algorithm is  $\mathcal{O}(d^5(d+B)B)$  where  $d$  is the dimension of the lattice, and  $B$  is the maximal bit-size of an entry in the lattice. In our algorithm,  $d = \binom{m+n+1}{m} = \mathcal{O}(m^{n+1}/(n+1)!) = \mathcal{O}(1/\epsilon^{n+1})$  ( $\epsilon > 0$  is a parameter to be determined), and  $B \approx d \cdot \log N_2$ . Therefore, the time complexity of our algorithm is about

$$\mathcal{O}\left(\left(\frac{1}{\epsilon}\right)^{7(n+1)} (1+h)h\right),$$

which means that our algorithm runs in polynomial time for a given  $n$ . However, it could be hard to factor  $N_1, N_2$  as  $n$  is large. Intuitively, when the total amount of information remains constant, the more the shared blocks are, the more distinct the information becomes, and at the same time, the more variables need to be solved for, thus leading to an increase in the algorithm's complexity. Specifically, consider two scenarios with different numbers of blocks,  $n_1$  and  $n_2$ , where  $n_1 > n_2$ . Moreover, with a fixed parameter  $m$ , the lattice dimension corresponding to  $n_1$  is  $d_1 = \mathcal{O}(m^{n_1+1}/(n_1+1)!)$ , and the lattice dimension corresponding to  $n_2$  is  $d_2 = \mathcal{O}(m^{n_2+1}/(n_2+1)!)$ . As the number of shared bits approaches the lower bound in Theorem 3,  $m$  must be larger. Consequently, we are typically interested in cases where the number of shared bits is close to this lower bound, leading to  $m \gg n_1 > n_2$ . In this case, the ratio  $d_1/d_2 = \mathcal{O}(m^{n_1-n_2})$ , indicating that the runtime of the algorithm for  $n_1$  blocks is about  $m^{n_1-n_2}$  times that of the algorithm for  $n_2$  blocks. This implies that the runtime of the algorithm increases exponentially with the number of shared blocks.

### 3.2 Analysis of Arbitrary Position Case for Two Moduli

In this subsection, we discuss the case where  $p_1, p_2$  share  $n \geq 2$  continuous bit blocks where each block has different displacement between its positions in  $p_1$  and  $p_2$  (see Fig. 3). Suppose that the RSA moduli  $N_1 = p_1 q_1$  and  $N_2 = p_2 q_2$  are both  $h$ -bit numbers.

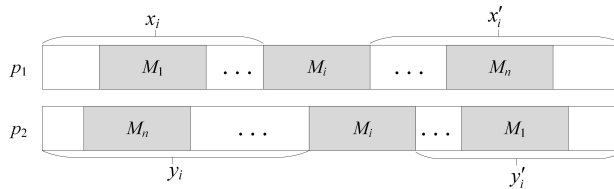


Figure 3: Shared bits of  $p_1$  and  $p_2$

In this case, we can't use one equation to represent the complete algebraic information. Accordingly, we make an algebraic equation for each shared bit block. Let  $x_i, x'_i, y_i, y'_i$  ( $i \in [1 : n]$ ) be the unknown variables presented in Fig. 3. Set  $\Delta_i = t_i - t'_i$  ( $i \in [1 : n]$ ), where  $t_i h$  and  $t'_i h$  are the positions of the lowest bit of  $x_i$  and  $y_i$ , respectively, for  $i \in [1 : n]$ .

Without loss of generality, we assume that  $\Delta_i \geq 0$  for  $i \in [1 : n]$  (if not, the equation will be  $p_2 - 2^{-h\Delta_i}p_1 = (y_i - x_i)2^{t_i h} + (y'_i - 2^{-h\Delta_i}x'_i)$ , which exhibits the same structure to the equation  $2^{h\Delta_i}p_2 - p_1 = (y_i - x_i)2^{t_i h} + (2^{h\Delta_i}y'_i - x'_i)$  encountered when  $\Delta_i \geq 0$ , thus it has no impact on the subsequent analysis). Consequently, we can obtain the following system of equations

$$\begin{cases} 2^{h\Delta_1}p_2 - p_1 = (y_1 - x_1)2^{t_1 h} + (2^{h\Delta_1}y'_1 - x'_1), \\ 2^{h\Delta_2}p_2 - p_1 = (y_2 - x_2)2^{t_2 h} + (2^{h\Delta_2}y'_2 - x'_2), \\ \dots \\ 2^{h\Delta_n}p_2 - p_1 = (y_n - x_n)2^{t_n h} + (2^{h\Delta_n}y'_n - x'_n), \end{cases}$$

Multiplying both sides of each equation in the above system of equations by  $q_1$ , we have

$$\begin{cases} 2^{h\Delta_1}p_2q_1 = N_1 + 2^{t_1 h}r_1q_1 + r'_1q_1, \\ 2^{h\Delta_2}p_2q_1 = N_1 + 2^{t_2 h}r_2q_1 + r'_2q_1, \\ \dots \\ 2^{h\Delta_n}p_2q_1 = N_1 + 2^{t_n h}r_nq_1 + r'_nq_1. \end{cases}$$

where  $r_i = y_i - x_i$ ,  $r'_i = 2^{h\Delta_i}y'_i - x'_i$  ( $i \in [1 : n]$ ). Let  $r_i, r'_i$  ( $i \in [1 : n]$ ), and  $r = q_1$  be the variables, we can obtain a set of modular polynomials as follows

$$\begin{cases} f_1 = N_1 + 2^{t_1 h}r_1r + r'_1r \equiv 0 \pmod{2^{h\Delta_1}p_2}, \\ f_2 = N_1 + 2^{t_2 h}r_2r + r'_2r \equiv 0 \pmod{2^{h\Delta_2}p_2}, \\ \dots \\ f_n = N_1 + 2^{t_n h}r_nr + r'_nr \equiv 0 \pmod{2^{h\Delta_n}p_2}. \end{cases}$$

To solve the above system of modular equations, we can also use the technique of introducing a new variable  $z = p_1$  to reduce the determinant of the lattice. Specifically, we define a cluster of shift polynomials for  $i_j \in [0 : m_j]$  such that  $i_j \leq m_j - k_j$  for  $j \in [1 : n]$  as follows.

$$g_{\vec{i}\vec{k}} = 2^{\sum_{j=1}^n (m_j - k_j)h\Delta_j} \left( \prod_{j=1}^n (r_j r)^{i_j} f_j^{k_j} \right) z^l N_2^{T_1} E^{T_2},$$

where  $T_1 = \max\{t - \sum_{j=1}^n k_j, 0\}$ ,  $T_2 = \min\{l, \sum_{j=1}^n (i_j + k_j)\}$ , and  $t = \lfloor \tau \sum_{j=1}^n m_j \rfloor$  ( $0 < \tau < 1$ ),  $l = \lfloor \sigma \sum_{j=1}^n m_j \rfloor$  ( $0 < \sigma < 1$ ) with  $m_j$  being the parameters to be determined.

Here,  $\vec{i}\vec{k}$  denotes the vector  $(i_1, \dots, i_n, k_1, \dots, k_n)$ ,  $E = N_1^{-1} \pmod{2^{\sum_{j=1}^n hm_j \Delta_j} N_2^t}$ . It is easy to verify that

$$g_{\vec{i}\vec{k}} \equiv 0 \pmod{2^{\sum_{j=1}^n hm_j \Delta_j} \cdot p_2^t}.$$

We have defined the corresponding monomial order  $\prec$ :

$$z^l r^{\sum_{j=1}^n (i_j + k_j)} \prod_{j=1}^n r_j^{i_j} r_j'^{k_j} \prec z^l r^{\sum_{j=1}^n (i'_j + k'_j)} \prod_{j=1}^n r_j^{i'_j} r_j'^{k'_j}$$

if and only if  $k'_1 > k_1$  or  $k'_1 = k_1, k'_2 > k_2$  or  $\dots$  or  $k'_1 = k_1, k'_2 = k_2, \dots, k'_{n-1} = k_{n-1}, k'_n > k_n$  or  $k'_1 = k_1, k'_2 = k_2, \dots, k'_n = k_n, i'_1 > i_1$  or  $k'_1 = k_1, k'_2 = k_2, \dots, i'_1 = i_1, i'_2 > i_2$  or  $\dots$  or  $k'_1 = k_1, k'_2 = k_2, \dots, i'_{n-1} = i_{n-1}, i'_n > i_n$ . We define

$$\mathbf{G} = \left\{ \prod_{j=1}^n (r_j r)^{i_j} f_j^{k_j} \mid k_j + i_j \leq m_j \text{ for } j \in [1 : n] \right\}, \mathbf{F}_j = \{(r_j r)^i f_j^k \mid k + i \leq m_j\}.$$

Let  $L(\mathbf{G})$  be the lattice generated by  $\mathbf{G}$  and  $L(\mathbf{F}_j)$  be the lattice generated by  $\mathbf{F}_j$ , for  $j \in [1 : n]$ . It is clear that each polynomial in  $\mathbf{G}$  can be uniquely decomposed into one product of the polynomials in  $\mathbf{F}_j$ , for  $j \in [1 : n]$ . Therefore, we can consider that  $L(\mathbf{G})$  is the Minkowski sum lattice (introduced by [Aon13]) of  $L(\mathbf{F}_j)$ , for  $j \in [1 : n]$ . Further, we know that all  $L(\mathbf{F}_j)$  have a strictly increasing degree order, and they are all lower triangular. So  $L(\mathbf{G})$  is lower triangular due to Theorem 3 in [Aon13]. Because lattice  $L$  generated by the  $g_{i\vec{k}}$ 's has the same structure as  $L(\mathbf{G})$ ,  $L$  is lower triangular too. As a result, we can easily compute the determinant  $\det(L)$  of  $L$  by multiplying the elements on the diagonal, and compute the dimension of  $L$  as

$$\dim(L) = \prod_{j=1}^n \binom{m_j + 2}{m_j}.$$

We know that there are  $2n + 1$  variables for this case with the method in the end of Subsection 3.1. To solve the above system of modular equations, we need the condition of Coppersmith's method working successfully to hold

$$\det(L)^{\frac{1}{\dim(L)-2n+2}} < 2^{\sum_{j=1}^n hm_j \Delta_j} \cdot p_2^t.$$

We can use this condition to determine appropriate values of  $m_j$  firstly, for  $j \in [1 : n]$ , then carefully determine the optimal values  $t_0 = \lceil \tau_0 \sum_{j=1}^n m_j \rceil$  and  $l_0 = \lfloor \sigma_0 \sum_{j=1}^n m_j \rfloor$  ( $0 < \tau_0, \sigma_0 < 1$ ). After selecting the appropriate parameters, we can construct a lattice basis just as mentioned before. Next, we reduce it using the LLL algorithm. Finally, we can obtain the solutions using numerical methods, such as Gröbner basis.

### 3.3 Analysis of Ordered Position Case for $k$ RSA Moduli

In this subsection, we generalized our analysis in Subsection 3.1 from two RSA moduli to an arbitrary number of RSA moduli.

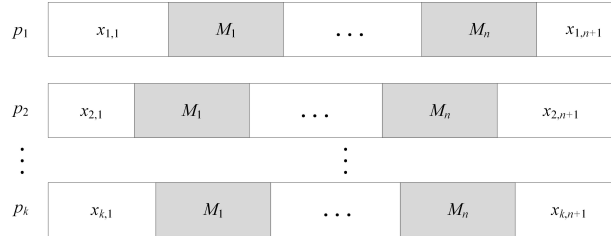


Figure 4: Shared bits of  $p_1, p_2, \dots, p_k$

Suppose that there are  $k > 2$   $h$ -bit RSA moduli  $N_1 = p_1 q_1, N_2 = p_2 q_2, \dots, N_k = p_k q_k$ , and  $p_i, p_j$  are all  $(1 - \alpha)h$ -bit prime factors which share  $n \geq 2$  continuous bit blocks where each block has the same displacement  $\Delta_{i,j}$  between its positions in  $p_i$  and  $p_j$ , for  $1 \leq i \neq j \leq k$ . Just like Subsection 3.1, we can write

$$p_j = x_{j,n+1} + x_{j,n} \cdot 2^{t_{j,n}h} + x_{j,n-1} \cdot 2^{t_{j,n-1}h} + \dots + x_{j,1} \cdot 2^{t_{j,1}h} + M_j,$$

for  $j \in [1 : k]$ , where  $M_j = M_1 \cdot 2^{a_{j,1}h} + M_2 \cdot 2^{a_{j,2}h} + \dots + M_n \cdot 2^{a_{j,n}h}$ , and  $a_{j,i}h, t_{j,i}h$  are the positions of the lowest bits of  $M_i, x_{j,i}$ , respectively, for  $i \in [1 : n]$  (see Fig. 4). Without loss of generality, we assume that  $t_{1,i} \leq t_{j,i}$  for  $j \in [2 : k]$  and  $i \in [1 : n]$ . Then there is

$$2^{h\Delta_{1,j}} p_1 - p_j = \sum_{i=1}^n (x_{1,i} - x_{j,i}) \cdot 2^{t_{j,i}h} + (2^{h\Delta_{1,j}} x_{1,n+1} - x_{j,n+1}), \quad (9)$$

for  $j \in [2 : k]$ . Multiplying both sides of Eq. (9) by  $q_j$ , we can obtain

$$\sum_{i=1}^n (x_{1,i} - x_{j,i}) q_j 2^{t_{j,i}h} + (2^{h\Delta_{1,j}} x_{1,n+1} - x_{j,n+1}) q_j + N_j \equiv 0 \pmod{2^{h\Delta_{1,j}} p_1},$$

for  $j \in [2 : k]$ . Naturally, we have the following modular polynomials

$$f_j(\vec{r}_j) := N_j + \sum_{i=1}^n r_{i,j} r_{n+2,j} \cdot 2^{t_{j,i}h} + r_{n+1,j} r_{n+2,j} \pmod{2^{h\Delta_{1,j}} p_1},$$

for  $j \in [2 : k]$ , where  $\vec{r}_j$  denotes the vector  $(r_{1,j}, r_{2,j}, \dots, r_{n+2,j})$ . It is clear that  $(x_{1,1} - x_{j,1}, x_{1,2} - x_{j,2}, \dots, x_{1,n} - x_{j,n}, 2^{h\Delta_{1,j}} x_{1,n+1} - x_{j,n+1}, q_j)$  is a root of  $f_j(\vec{r}_j)$  for  $j \in [2 : k]$ . Same as before, we introduce a new variable  $z_j = p_j$  to reduce  $q_j$  ( $j \in [2 : k]$ ) by multiplying  $E_j = N_j^{-1} \pmod{2^{mh \cdot \min\{\Delta_{1,j}\}} N_1^t}$ . Given parameters  $m$ ,  $t = \lceil \tau m \rceil$ , and  $l = \lfloor \sigma m \rfloor$  ( $0 < \sigma, \tau < 1$ ), we generate the following family of polynomials

$$g_{\vec{v}\vec{w}} = C \prod_{j=2}^k z_j^l \prod_{j=2}^k \left( \prod_{i=1}^n (r_{i,j} r_{n+2,j})^{v_{j,i}} \right) f_j^{w_j},$$

where  $C = 2^{(m - \sum_{j=2}^k w_j)h \cdot \min\{\Delta_{1,j}\}} \cdot \prod_{j=2}^k E_j^T \cdot N_1^{\max\{t - \sum_{j=2}^k w_j, 0\}}$ . Here,  $v_{j,i}, w_j \in [0 : m]$ ,  $\sum_{j=2}^k (w_j + \sum_{i=1}^n v_{j,i}) \leq m$ , and  $\vec{v}\vec{w}$  denotes the vector  $(v_{2,1}, v_{2,2}, \dots, v_{k,n}, w_2, \dots, w_k)$ . It is easy to verify that

$$g_{\vec{v}\vec{w}} \equiv 0 \pmod{2^{hm \cdot \min\{\Delta_{1,j}\}} p_1^t}.$$

We have also defined the corresponding monomial order  $\prec$ :

$$\prod_{j=2}^k z_j^l \prod_{j=2}^k \left( \prod_{i=1}^n r_{i,j}^{v_{j,i}} \right) r_{n+1,j}^{w_j} r_{n+2,j}^{w_j + \sum_{i=1}^n v_{j,i}} \prec \prod_{j=2}^k z_j^l \prod_{j=2}^k \left( \prod_{i=1}^n r_{i,j}^{v'_{j,i}} \right) r_{n+1,j}^{w'_j} r_{n+2,j}^{w'_j + \sum_{i=1}^n v'_{j,i}},$$

if and only if  $w_k < w'_k$  or  $w_k = w'_k, w_{k-1} < w'_{k-1}$  or  $\dots$  or  $w_k = w'_k, w_{k-1} = w'_{k-1}, \dots, w_2 = w'_2, v_{k,n} < v'_{k,n}$  or  $w_k = w'_k, w_{k-1} = w'_{k-1}, \dots, w_2 = w'_2, v_{k,n} = v'_{k,n}, v_{k,n-1} < v'_{k,n-1}$  or  $\dots$  or  $w_k = w'_k, w_{k-1} = w'_{k-1}, \dots, w_2 = w'_2, v_{k,n} = v'_{k,n}, v_{k,n-1} = v'_{k,n-1}, \dots, v_{2,2} = v'_{2,2}, v_{2,1} < v'_{2,1}$ .

We can construct the lattice basis  $L$  by arranging the above-generated polynomials based on the monomial order we have defined. And there are  $(n+2)(k-1)$  variables for this case with the method in the end of Subsection 3.1. Let the determinant and dimension of  $L$  be  $\det(L)$  and  $\dim(L)$ , respectively. To recover the desired roots, based on Theorem 2, it needs

$$\det(L)^{\frac{1}{\dim(L) - (n+2)(k-1) + 1}} < 2^{mh \cdot \min\{\Delta_{1,j}\}} p_1^t.$$

Then, as  $m$  tends to be infinity, we can get optimized values  $t_0 = \lceil \tau_0 m \rceil$  and  $l_0 = \lfloor \sigma_0 m \rfloor$  ( $0 < \sigma_0, \tau_0 < 1$ ). These values can be substituted into the former equations to construct lattice basis with a smaller determinant. Next, we reduce the basis using the LLL algorithm to make the equations hold over the integers. Finally, we solve the equations using numerical methods, such as Gröbner basis.

## 4 Experiments

We implemented several experiments to verify the validity of our approach. The experiments were implemented in SAGE 9.3 over Windows 10 on a PC with 2.90 GHz Intel Core CPU and 32 GB RAM. The code is at <https://github.com/chennnstar/Codes-for-IFP>.

Table 3: The experiment results for  $n = 2$ 

$h$	$\alpha$	$\Delta$	$m$	$t$	$l$	dim	Time for LLL
500	0.16	30	6	3	3	84	235.10
500	0.14	40	4	2	2	35	1.35
1000	0.16	50	6	3	3	84	229.59
1000	0.14	90	4	2	2	35	2.97
2000	0.16	100	6	3	3	84	863.46
2000	0.14	160	4	2	2	35	8.50

The positions of the shared blocks are (190, 390, 40, 140), (300, 400, 40, 240), (460, 760, 80, 380), (480, 780, 90, 390), (920, 1520, 160, 760), (940, 1540, 160, 760).

Table 4: The experiment results for  $n = 3$ 

$h$	$\alpha$	$\Delta$	$m$	$t$	$l$	dim	Time for LLL
500	0.13	20	5	2	3	126	437.57
500	0.10	30	4	2	2	70	24.91
1000	0.13	40	5	2	3	126	1130.95
1000	0.10	20	4	2	2	70	48.47
2000	0.13	80	5	2	3	126	4193.38
2000	0.10	100	4	2	2	70	196.07

The positions of the shared blocks are (305, 405, 165, 265, 25, 125), (320, 420, 180, 280, 40, 140), (650, 850, 350, 550, 50, 250), (680, 880, 380, 580, 30, 230), (1240, 1640, 650, 1050, 100, 500), (1250, 1650, 700, 1100, 150, 550).

Table 5: The experiment results for  $n = 4$ 

$h$	$\alpha$	$\Delta$	$m$	$t$	$l$	dim	Time for LLL
500	0.10	15	3	1	2	56	3.39
500	0.11	20	4	2	3	126	1148.56
1000	0.10	40	3	1	2	56	5.93
1000	0.11	30	4	2	3	126	1093.09
2000	0.10	60	3	1	2	56	16.99
2000	0.11	100	4	2	3	126	4026.71

The positions of the shared blocks are (330, 430, 210, 310, 115, 190, 20, 95), (330, 430, 210, 310, 135, 190, 25, 125), (680, 880, 540, 640, 300, 500, 60, 260), (768, 868, 532, 732, 296, 496, 50, 260), (1360, 1760, 880, 1280, 580, 780, 100, 500), (1360, 1760, 890, 1290, 510, 810, 140, 460).

According to our work, we could factor  $N_1, N_2$  if  $u > 2(n+1)\alpha(1 - \alpha^{\frac{1}{n+1}})$  for the case in Subsection 3.1. First, we generate a family of polynomials  $g_{i,k}^l(\vec{r}, z)$ 's to construct the lattice basis we want, then reduce it using the LLL algorithm. After the LLL-reduction, we convert the first  $n+1$  row vectors of the basis matrix to polynomials  $\tilde{\xi}_1, \tilde{\xi}_2, \dots, \tilde{\xi}_{n+1}$  over the integers using the method in the end of Subsection 3.1. Finally, we can solve the system of  $\tilde{\xi}_1 = 0, \tilde{\xi}_2 = 0, \dots, \tilde{\xi}_{n+1} = 0$  over the integers using Gröbner basis to recover  $p_2, q_1$  so that we are able to factor  $N_1, N_2$ .

We implemented our experiments with different parameters including the prime factors generated randomly, and we obtained the desired roots in all experiments. The results are shown in Table 3, Table 4 and Table 5. Meanwhile, we have compared our work with the work of Wang et al. in the MBs case as an example of our approach is the same as

the corresponding current optimal approach in three cases LSBs, MSBs, and MBs. The results are shown in Table 6.

Table 6: The comparison of our method with  $n = 1$  and Wang et al.’s

	$h$	$\alpha$	$m$	$t$	$l$	dim	Time for LLL
[WQLF17]	1000	0.31	8	4	4	45	12.92
	2000	0.30	7	3	4	36	11.53
Ours	1000	0.31	7	3	4	36	3.70
	2000	0.30	7	3	4	36	11.31

The positions of the shared blocks are (10, 670), (60, 1360), (10, 670), (60, 1360).

In the tables,  $\Delta = t_i - s_i$  for  $i \in [1 : n]$ ; “dim” denotes the dimension of basis matrix  $B$ ; Time for LLL is measured in seconds. Moreover, we denote the positions of the shared blocks as  $(a_1h, t_1h, a_2h, t_2h, \dots, a_nh, t_nh)$ , and these positions are presented in the order in which the corresponding parameter setting appears in the table.

## 5 Conclusion

In this paper, we present a new generalized implicit factorization problem that  $p_1, p_2$  share  $n \geq 2$  continuous bit blocks where each block has one fixed displacement between its positions in  $p_1$  and  $p_2$ . We give a polynomial-time algorithm for the problem when  $p_1, p_2$  share more than  $(2\alpha \ln(1/\alpha))h$  bits. That is an exciting result because it seems that we can factor  $N_1, N_2$  in polynomial time as long as  $u > 2\alpha \ln(1/\alpha)$ , no matter how many blocks  $p_1, p_2$  share. We have verified the correctness of our approach by experiments. In addition, we extend the problem to arbitrary position and  $k > 2$  moduli, respectively, and we theoretically give the corresponding solutions based on Coppersmith’s method. However, factoring  $N_1, N_2, \dots, N_k$  with the positions of the shared bits arbitrary still remains unsolved. This is the direction of our future work. In summary, our work is valuable for studying the security of RSA with various implicit hints and can support users to avoid the corresponding security risks.

## References

- [Aon13] Yoshinori Aono. Minkowski sum based lattice construction for multivariate simultaneous Coppersmith’s technique and applications to RSA. In Colin Boyd and Leonie Simpson, editors, *Information Security and Privacy*, pages 88–103, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-39059-3\_7.
- [BM03] Johannes Blömer and Alexander May. New partial key exposure attacks on RSA. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 27–43, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. doi:10.1007/978-3-540-45146-4\_2.
- [Cop96] Don Coppersmith. Finding a small root of a univariate modular equation. In Ueli Maurer, editor, *Advances in Cryptology - EUROCRYPT 1996*, pages 155–165, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. doi:10.1007/3-540-68339-9\_14.



- [Cop97] Don Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10:233–260, 1997. doi:[10.1007/s001459900030](https://doi.org/10.1007/s001459900030).
- [FMR10] Jean-Charles Faugère, Raphaël Marinier, and Guénaél Renault. Implicit factoring with shared most significant and middle bits. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography - PKC 2010*, pages 70–87, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. doi:[10.1007/978-3-642-13013-7\\_5](https://doi.org/10.1007/978-3-642-13013-7_5).
- [FNP24] Yansong Feng, Abderrahmane Nitaj, and Yanbin Pan. Generalized implicit factorization problem. In Claude Carlet, Kalikinkar Mandal, and Vincent Rijmen, editors, *Selected Areas in Cryptography - SAC 2023*, pages 369–384, Cham, 2024. Springer Nature Switzerland. doi:[10.1007/978-3-031-53368-6\\_18](https://doi.org/10.1007/978-3-031-53368-6_18).
- [HG97] Nicholas Howgrave-Graham. Finding small roots of univariate modular equations revisited. In Michael Darnell, editor, *Cryptography and Coding*, pages 131–142, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. doi:[10.1007/BFb0024458](https://doi.org/10.1007/BFb0024458).
- [HM08] Mathias Herrmann and Alexander May. Solving linear equations modulo divisors: On factoring given any bits. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, pages 406–424, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. doi:[10.1007/978-3-540-89255-7\\_25](https://doi.org/10.1007/978-3-540-89255-7_25).
- [HR23] Nadia Heninger and Keegan Ryan. The hidden number problem with small unknown multipliers: Cryptanalyzing MEGA in six queries and other applications. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *Public-Key Cryptography - PKC 2023*, pages 147–176, Cham, 2023. Springer Nature Switzerland. doi:[10.1007/978-3-031-31368-4\\_6](https://doi.org/10.1007/978-3-031-31368-4_6).
- [JM06] Ellen Jochemsz and Alexander May. A strategy for finding roots of multivariate polynomials with new applications in attacking RSA variants. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology - ASIACRYPT 2006*, pages 267–282, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. doi:[10.1007/11935230\\_18](https://doi.org/10.1007/11935230_18).
- [LLL82] W Lenstra, H, K Lenstra, A, and L Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982. doi:[10.1007/BF01457454](https://doi.org/10.1007/BF01457454).
- [LPZ<sup>+</sup>16] Yao Lu, Liqiang Peng, Rui Zhang, Lei Hu, and Dongdai Lin. Towards optimal bounds for implicit factorization problem. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography - SAC 2015*, pages 462–476, Cham, 2016. Springer International Publishing. doi:[10.1007/978-3-319-31301-6\\_26](https://doi.org/10.1007/978-3-319-31301-6_26).
- [LZL13] Yao Lu, Rui Zhang, and Dongdai Lin. Improved bounds for the implicit factorization problem. *Adv. Math. Commun.*, 7:243–251, 2013. doi:[10.3934/amc.2013.7.243](https://doi.org/10.3934/amc.2013.7.243).
- [May03] Alexander May. *New RSA vulnerabilities using lattice reduction methods*. PhD thesis, Paderborn University, 2003.
- [MNS22] Alexander May, Julian Nowakowski, and Santanu Sarkar. Approximate divisor multiples – factoring with only a third of the secret CRT-exponents. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology*

- *EUROCRYPT 2022*, pages 147–167, Cham, 2022. Springer International Publishing. doi:[10.1007/978-3-031-07082-2\\_6](https://doi.org/10.1007/978-3-031-07082-2_6).
- [MR09] Alexander May and Maike Ritzenhofen. Implicit factoring: On polynomial time factoring given only an implicit hint. In Stanisław Jarecki and Gene Tsudik, editors, *Public Key Cryptography - PKC 2009*, pages 1–14, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:[10.1007/978-3-642-00468-1\\_1](https://doi.org/10.1007/978-3-642-00468-1_1).
- [NA15] Abderrahmane Nitaj and Muhammad Reza Kamel Ariffin. Implicit factorization of unbalanced RSA moduli. *Journal of Applied Mathematics and Computing*, 48:349–363, 2015. doi:[10.1007/s12190-014-0806-1](https://doi.org/10.1007/s12190-014-0806-1).
- [NS05] Phong Q. Nguên and Damien Stehlé. Floating-point LLL revisited. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, pages 215–233, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:[10.1007/11426639\\_13](https://doi.org/10.1007/11426639_13).
- [PHL<sup>+</sup>15] Liqiang Peng, Lei Hu, Yao Lu, Zhangjie Huang, and Jun Xu. Implicit factorization of RSA moduli revisited (short paper). In Keisuke Tanaka and Yuji Suga, editors, *Advances in Information and Computer Security*, pages 67–76, Cham, 2015. Springer International Publishing. doi:[10.1007/978-3-319-22425-1\\_5](https://doi.org/10.1007/978-3-319-22425-1_5).
- [PHX<sup>+</sup>14] Liqiang Peng, Lei Hu, Jun Xu, Zhangjie Huang, and Yonghong Xie. Further improvement of factoring RSA moduli with implicit hint. In David Pointcheval and Damien Vergnaud, editors, *Progress in Cryptology - AFRICACRYPT 2014*, pages 165–177, Cham, 2014. Springer International Publishing. doi:[10.1007/978-3-319-06734-6\\_11](https://doi.org/10.1007/978-3-319-06734-6_11).
- [RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978. doi:[10.1145/359340.359342](https://doi.org/10.1145/359340.359342).
- [Sar11] Santanu Sarkar. Partial key exposure: Generalized framework to attack RSA. In Daniel J. Bernstein and Sanjit Chatterjee, editors, *Progress in Cryptology - INDOCRYPT 2011*, pages 76–92, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:[10.1007/978-3-642-25578-6\\_7](https://doi.org/10.1007/978-3-642-25578-6_7).
- [SLH14] Meng Shi, Xianghui Liu, and Wenbao Han. Implicit factoring with shared middle discrete bits. In W. Eric Wong and Tingshao Zhu, editors, *Computer Engineering and Networking*, pages 255–263, Cham, 2014. Springer International Publishing. doi:[10.1007/978-3-319-01766-2\\_30](https://doi.org/10.1007/978-3-319-01766-2_30).
- [SM11] S. Sarkar and S. Maitra. Approximate integer common divisor problem relates to implicit factorization. *IEEE Trans. Inf. Theor.*, 57(6):4002–4013, jun 2011. doi:[10.1109/TIT.2011.2137270](https://doi.org/10.1109/TIT.2011.2137270).
- [SZZ<sup>+</sup>19] Zhelei Sun, Tianwei Zhang, Xiaoxia Zheng, Liuqing Yang, and Liqiang Peng. A method for solving generalized implicit factorization problem. In Songlin Sun, Meixia Fu, and Lexi Xu, editors, *Signal and Information Processing, Networking and Computers*, pages 284–290, Singapore, 2019. Springer Singapore. doi:[10.1007/978-981-13-7123-3\\_34](https://doi.org/10.1007/978-981-13-7123-3_34).
- [WQLF17] Shixiong Wang, Longjiang Qu, Chao Li, and Shaojing Fu. A better bound for implicit factorization problem with shared middle bits. *Science China Information Sciences*, 61, 2017. doi:[10.1007/s11432-017-9176-5](https://doi.org/10.1007/s11432-017-9176-5).

- 
- [Zhe23] Mengce Zheng. Generalized implicit-key attacks on RSA. *Journal of Information Security and Applications*, 77:103562, 2023. doi:[10.1016/j.jisa.2023.103562](https://doi.org/10.1016/j.jisa.2023.103562).