New Results on Modular Inversion Hidden Number Problem and Inversive Congruential Generator

Jun Xu[1,2], Santanu Sarkar[3], Lei Hu[1,2,6], Huaxiong Wang[4], and Yanbin Pan[5]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] Data Assurance and Communications Security Research Center, Chinese Academy of Sciences, Beijing 100093, China
[3] Department of Mathematics, Indian Institute of Technology Madras, Sardar Patel Road, Chennai 600036, India
[4] Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University Singapore, Singapore
[5] Key Laboratory of Mathematics Mechanization, NCMIS, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China
[6] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
{xujun,hulei}@iie.ac.cn,sarkar.santanu.bir@gmail.com,HXWang@ntu.edu.sg

Abstract. The Modular Inversion Hidden Number Problem (MIHNP), introduced by Boneh, Halevi and Howgrave-Graham in Asiacrypt 2001, is briefly described as follows: Let $\mathrm{MSB}_\delta(z)$ refer to the δ most significant bits of z. Given many samples $\left(t_i, \mathrm{MSB}_\delta((\alpha + t_i)^{-1} \bmod p)\right)$ for random $t_i \in \mathbb{Z}_p$, the goal is to recover the hidden number $\alpha \in \mathbb{Z}_p$. MIHNP is an important class of Hidden Number Problem.

In this paper, we revisit the Coppersmith technique for solving a class of modular polynomial equations, which is respectively derived from the recovering problem of the hidden number α in MIHNP. For any positive integer constant d, let integer $n = d^{3+o(1)}$. Given a sufficiently large modulus p, $n+1$ samples of MIHNP, we present a heuristic algorithm to recover the hidden number α with a probability close to 1 when $\delta/\log_2 p > \frac{1}{d+1} + o(\frac{1}{d})$. The overall time complexity of attack is polynomial in $\log_2 p$, where the complexity of the LLL algorithm grows as $d^{\mathcal{O}(d)}$ and the complexity of the Gröbner basis computation grows as $(2d)^{\mathcal{O}(n^2)}$. When $d > 2$, this asymptotic bound outperforms $\delta/\log_2 p > \frac{1}{3}$ which is the asymptotic bound proposed by Boneh, Halevi and Howgrave-Graham in Asiacrypt 2001. It is the first time that a better bound for solving MIHNP is given, which implies that the conjecture that MIHNP is hard whenever $\delta/\log_2 p < \frac{1}{3}$ is broken. Moreover, we also get the best result for attacking the Inversive Congruential Generator (ICG) up to now.

Keywords: Modular inversion hidden number problem, inversive congruential generator, lattice, LLL algorithm, the Coppersmith technique.

1 Introduction

1.1 Background

In cryptography research, one focuses on whether a mathematical problem is computationally hard, as the hard mathematical problem is the foundation of constructing cryptographic secure schemes. In [4], Boneh, Halevi and Howgrave-Graham introduced an algebraic complexity assumption called the Modular Inversion Hidden Number Problem (MIHNP) in order to design a pseudorandom number generator and message authentication code.

Definition 1 (Modular Inversion Hidden Number Problem(MIHNP)). *For a given prime p, consider a secret $\alpha \in \mathbb{Z}_p$ and $n+1$ elements $t_0, t_1, \ldots, t_n \in \mathbb{Z}_p \setminus \{-\alpha\}$, chosen independently and uniformly at random. Given $n+1$ samples*

$$\left\{ \left(t_i, \mathrm{MSB}_\delta((\alpha + t_i)^{-1} \bmod p)\right) \right\}_{i=0}^{n}$$

where $\mathrm{MSB}_\delta(z)$ refers to the δ most significant bits of z, the goal is to recover the hidden number α.

MIHNP is closely related to Hidden Number Problem (HNP), which was introduced in [5] by Boneh and Venkatesan to prove the bit security of the Diffie-Hellman key-exchange in $\mathbb{Z}_p$. Shparlinski [28] revealed that the primary motivation of studying MIHNP is to expect the bit security result of the Elliptic Curve Diffie-Hellman key-exchange. In PKC 2017, Shani [27] used ideas [4,20] of solving MIHNP to get the first rigorous result about the bit security of the Elliptic Curve Diffie-Hellman key-exchange.

1.2 Cryptanalysis

In Asiacrypt 2001, Boneh, Halevi and Howgrave-Graham gave two heuristic lattice methods to solve MIHNP [4]. Let δ denote the given number of most significant bits of $(\alpha + t_i)^{-1} \bmod p$'s. The first method works if $\delta > \frac{2}{3}\log_2 p$. The second method solves MIHNP if $\delta > \frac{1}{3}\log_2 p$, i.e., the knowledge of significantly fewer bits is needed. Moreover, *Boneh, Halevi and Howgrave-Graham [4] conjectured that MIHNP is hard whenever $\delta < \frac{1}{3}\log_2 p$.* In 2012, Ling et al. presented a rigorous polynomial time algorithm for solving MIHNP [20]. The obtained asymptotic result is $\delta > \frac{2}{3}\log_2 p$, which is the same as that of the first method in [4]. In 2014, Xu et al. [34] gave a heuristic lattice approach based on the Coppersmith technique, which has certain advantages when the number of samples is sufficiently small. However, the corresponding asymptotic result is $\delta > \frac{1}{2}\log_2 p$ which is still weaker than the second method in [4]. On the other hand, recently Xu et al. [35] obtained the explicit lattice construction of the second method in [4] and achieved the same asymptotic result $\delta > \frac{1}{3}\log_2 p$.

1.3 Our Contribution

We revisit the Coppersmith technique to solve the following system of multivariate modular polynomial equations

$$f_{0j}(x_0, x_j) := a_{0j} + b_{0j}x_0 + c_{0j}x_j + x_0x_j = 0 \pmod{p} \text{ for } 1 \leq j \leq n,$$

which is obtained from the recovering problem of the hidden number in MIHNP [35]. In the polynomial selection strategy for the Coppersmith lattice, we use the idea on *helpful polynomials* in [21,29] (see Section 2.2). The diagonals of helpful polynomials in the basis matrix are smaller than the involved modulo. This criterion enables helpful polynomials to facilitate the solution of modular equations. Therefore, we should try our best to add helpful polynomials into the involved lattice.

Based on the lattice construction of [4,35], we find that new linearly independent polynomials can still be added into the lattice by making full use of the linear combination of multiplies of several $f_{0j}(x_0, x_j)$ with common monomials. These newly added polynomials are helpful because their diagonal elements are smaller than modulo. Because the number of newly added helpful polynomials dominates the number of all the selected polynomials, it makes the Coppersmith technique search the desired small roots much efficiently.

In this paper, we obtain the following results. For any positive integer constant d, let integer $n = d^{3+o(1)}$. For a given sufficiently large modulus $p = 2^{\omega(d^{3d+2})}$, and $n+1$ given samples of MIHNP, we present a heuristic algorithm to recover the hidden number α with a probability close to 1 when $\delta/\log_2 p > \frac{1}{d+1} + o(\frac{1}{d})$. The overall time complexity of attack is polynomial in $\log_2 p$, where the complexity of the LLL algorithm grows as $d^{\mathcal{O}(d)}$ and the complexity of the Gröbner basis computation grows as $(2d)^{\mathcal{O}(n^2)}$. When $d = 2$, our asymptotic result of $\delta/\log_2 p$ is equal to $\frac{1}{3}$, which is same as the previous best result [4,35]. When $d > 2$, the corresponding asymptotic bound is $\frac{1}{d+1} < \frac{1}{3}$. This implies that our result is beyond the bound given by the second method in [4]. Hence, we disprove the conjecture proposed by Boneh, Halevi and Howgrave-Graham in [4]. This attack also applies to ICG, as in prior work [35].

In Table 1, we compare new bound of $\delta/\log_2 p$ and the corresponding time complexity with the existing works (see Appendix A and Remark 4). Our results show that new bound of $\delta/\log_2 p$ is equal to 0 in the asymptotic sense. This is to say that MIHNP can be heuristically solved in polynomial time when δ is a constant fraction of $\log_2 p$. However, for the practical solutions, it requires a huge lattice dimension $\mathcal{O}\left((\frac{\log_2 p}{\delta})^{\mathcal{O}(\frac{\log_2 p}{\delta})}\right)$ in order to ensure that $\delta/\log_2 p$ is close to 0.

1.4 Organization of the Paper

The rest of this paper is organized as follows. In Section 2, we recall some terminologies and preliminaries. In Section 3, we present a strategy for solving modular polynomial equations and give the result for attacking MIHNP. Section 4 presents the proof of triangle basis matrix. Section 5 gives the experimental result. Section 6 concludes the paper. In Appendices, we respectively give asymptotic time complexities in previous works, the computation of lattice determinant and the analysis of bound of the desired small root.

Table 1. Comparison of lower bounds of $\delta/\log_2 p$ and corresponding time complexities, where $\rho := \delta/\log_2 p$ and $k := \log_2 p$.

MIHNP ICG	Lower Bound of $\delta/\log_2 p$	Asymptotic Time Complexities		
		LLL	Gröbner basis	SVP
[3]	**3/4**	–	–	$k^{\mathcal{O}(1)}$
[20]	**2/3**	–	–	$k^{\mathcal{O}(1)}2^{\mathcal{O}(\frac{1}{\rho-\frac{2}{3}})}$
[1]	**1/2**	$\mathcal{O}\big(k^{\mathcal{O}(1)}(\log_2 \frac{1}{\rho-\frac{1}{2}})^{\mathcal{O}(\log_2 \frac{1}{\rho-\frac{1}{2}})}\big)$	$\big(\log_2 \frac{1}{\rho-\frac{1}{2}}\big)^{\mathcal{O}\big((\log_2 \frac{1}{\rho-\frac{1}{2}})^2\big)}$	–
[34]	**1/2**	$\mathcal{O}\big(k^{\mathcal{O}(1)}(\frac{1}{\rho-\frac{1}{2}})^{\mathcal{O}(1)}\big)$	$\big(\frac{1}{\rho-\frac{1}{2}}\big)^{o\big(\log_2 \frac{1}{\rho-\frac{1}{2}}\big)}$	–
[4]	**1/3**	–	–	$k^{\mathcal{O}(1)}2^{\mathcal{O}\big((\frac{2}{3\rho-1})^{\mathcal{O}(\frac{1}{\rho-\frac{1}{3}})}\big)}$
[35]	**1/3**	$\mathcal{O}\big(k^{\mathcal{O}(1)}(\frac{2}{3\rho-1})^{\mathcal{O}(\frac{1}{\rho-\frac{1}{3}})}\big)$	$(\frac{4}{3\rho-1})^{\mathcal{O}((\frac{1}{\rho-\frac{1}{3}})^{\mathcal{O}(1)})}$	–
This paper	**0**	$\mathcal{O}\big(k^{\mathcal{O}(1)}(\frac{1}{\rho})^{\mathcal{O}(\frac{1}{\rho})}\big)$	$(\frac{2}{\rho})^{\mathcal{O}((\frac{1}{\rho})^{\mathcal{O}(1)})}$	–

2 Preliminaries

2.1 Lattices

Let vectors $\mathbf{b}_1, \ldots, \mathbf{b}_w$ be linearly independent in $\mathbb{R}^n$. The set

$$\mathcal{L} = \Big\{ \sum_{i=1}^{w} k_i \mathbf{b}_i, k_i \in \mathbb{Z} \Big\}$$

is called a lattice with basis vectors $\mathbf{b}_1, \cdots, \mathbf{b}_w$. In this paper, the basis vectors involved are row vectors. The dimension and determinant of $\mathcal{L}$ are respectively $\dim(\mathcal{L}) = w, \det(\mathcal{L}) = \sqrt{\det(BB^T)}$, where $B = [\mathbf{b}_1^T, \cdots, \mathbf{b}_w^T]^T$ is a basis matrix. If B is a square matrix, then $\det(\mathcal{L}) = |\det(B)|$.

In 1982, Lenstra, Lenstra and Lovász presented a deterministic polynomial-time algorithm [19] in order to find a reduced basis of the lattice.

Lemma 1 ([19]). *Let $\mathcal{L}$ be a lattice. Within polynomial time, the LLL algorithm outputs reduced basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_w$ that satisfy*

$$\|\mathbf{v}_1\| \leq \|\mathbf{v}_2\| \leq \cdots \leq \|\mathbf{v}_i\| \leq 2^{\frac{w(w-1)}{4(w+1-i)}} \det(\mathcal{L})^{\frac{1}{w+1-i}}, 1 \leq i \leq w.$$

2.2 The Coppersmith Technique

In 1996, Coppersmith proposed lattice-based techniques [7,8,9] for finding the small solution of univariate modular polynomials and bivariate integer polynomials. In 2006, May et al. presented heuristic strategies for solving multivariate polynomials [15]. The Coppersmith technique has been widely used in the field of cryptanalysis such as attacking RSA and its variants (see the survey [21]

and recent results such as [16,30,31,25]) and analyzing pseudorandom number generators as well as computationally hard mathematical problems such as [14,11,12,1,32,6,2,35].

We explain briefly how one can utilize the idea of the Coppersmith technique to solve multivariate modular polynomials.

Definition of the Problem. Let $f_1(x_0, x_1, \cdots, x_n), \cdots, f_m(x_0, x_1, \cdots, x_n)$ be m irreducible multivariate polynomials defined over $\mathbb{Z}$, which have a common root $(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n)$ modulo a known integer p such that $|\widetilde{x}_0| < X_0, \cdots, |\widetilde{x}_n| < X_n$. The question is to recover the desired root $(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n)$ in polynomial time. The analysis needs to establish bounds X_i's to ensure recovery.

Step 1: Collection of Polynomials. One generates a collection of polynomials $g_1(x_0, x_1, \cdots, x_n)$, $\cdots$, $g_w(x_0, x_1, \cdots, x_n)$ such that $(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n)$ is a common modular root. For example, g_i's can be constructed as follows: $g_i(x_0, x_1, \ldots, x_n) = p^{d-(\beta_1^i+\cdots+\beta_m^i)} x_0^{\alpha_0^i} x_1^{\alpha_1^i} \cdots x_n^{\alpha_n^i} f_1^{\beta_1^i} \cdots f_m^{\beta_m^i}$ for $i = 1, \cdots, w$, where $d \in \mathbb{Z}^+$, $\alpha_0^i, \alpha_1^i, \cdots, \alpha_n^i, \beta_1^i, \cdots, \beta_m^i$ are nonnegative integers and $0 \leq \beta_1^i + \cdots + \beta_m^i \leq d$. It is easy to see that $g_i(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n) \equiv 0 \bmod p^d$ for every $i \in [1, \cdots, w]$.

Step 2: Construction of Lattice. Let $\mathbf{b}_i$ be the coefficient vector of the polynomial $g_i(x_0X_0, x_1X_1, \ldots, x_nX_n)$ for all $1 \leq i \leq w$. Then one generates the lattice $\mathcal{L} = \big\{ \sum_{i=1}^{w} k_i \mathbf{b}_i, k_i \in \mathbb{Z} \big\}$.

Step 3: Generation of Reduced Basis. One runs a lattice reduction algorithm, such as LLL algorithm, to obtain the $n+1$ reduced basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_{n+1}$ such that the corresponding polynomials $h_1(x_0, x_1, \cdots, x_n), \cdots, h_{n+1}(x_0, x_1, \cdots, x_n)$ have the desired common root $(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n)$ over $\mathbb{Z}$, where $\mathbf{v}_i$ is the coefficient vector of the polynomial $h_i(x_0X_0, x_1X_1, \ldots, x_nX_n)$ for $i = 1, \cdots, n+1$. Note that $h_i(x_0, x_1, \ldots, x_n)$ is a linear combination of $g_1(x_0, x_1, \ldots, x_n)$, $\cdots$, $g_w(x_0, x_1, \ldots, x_n)$. Hence, we have $h_i(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n) = 0 \pmod{p^d}$ for every $i \in [1, \cdots, n+1]$. In order to obtain $h_i(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n) = 0$ for all $1 \leq i \leq n$, we need the following lemma in this process.

Lemma 2 ([13]). *Let $h(x_0, x_1, \ldots, x_n)$ be an integer polynomial that consists of at most w monomials. Let d be a positive integer and the integers X_i be the upper bound of $|\widetilde{x}_i|$ for $i = 0, 1, \cdots, n$. Let $\|h(x_0X_0, x_1X_1, \ldots, x_nX_n)\|$ be the Euclidean norm of the coefficient vector of the polynomial $h(x_0X_0, x_1X_1, \ldots, x_nX_n)$ with variables $x_0, x_1, \ldots, x_n$. Suppose that*

1. $h(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n) = 0 \pmod{p^d}$,
2. $\|h(x_0X_0, x_1X_1, \ldots, x_nX_n)\| < \frac{p^d}{\sqrt{w}}$,

then $h(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n) = 0$ holds over $\mathbb{Z}$.

To get the above $n+1$ polynomials $h_1(x_0, x_1, \cdots, x_n), \cdots h_{n+1}(x_0, x_1, \cdots, x_n)$, from Lemma 1 and Lemma 2, one needs the Euclidean lengths of the $n+1$ reduced basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_{n+1}$ satisfy the condition

$$2^{\frac{w(w-1)}{4(w-n)}} \cdot \big(\det(\mathcal{L})\big)^{\frac{1}{w-n}} < \frac{p^d}{\sqrt{w}}, \tag{1}$$

where $w = \dim(\mathcal{L})$.

Based on Condition (1), one can determine the bounds X_i for $i = 0, \cdots, n$. In order to make the bounds X_i as large as possible, the polynomials $g_1(x_0, x_1, \cdots, x_n)$, $\cdots$, $g_w(x_0, x_1, \cdots, x_n)$ in Step 1 need to be constructed carefully. It is a difficult step in the Coppersmith technique.

The strategy of choosing polynomials for our lattice construction is based on the idea of *helpful polynomials* [21,29]. Neglecting low-order terms in (1), we can rewrite condition (1) and obtain the simplified condition as follows:

$$(\det(\mathcal{L}))^{\frac{1}{w}} < p^d.$$

For a triangular basis matrix, the left side of this simplified condition is regarded as the geometric mean of all diagonals of the basis matrix. The polynomials whose diagonals are less than p^d are called *helpful polynomials*. For a polynomial, for example, $h(x_0, \cdots, x_n)$, the diagonal of $h(x_0, \cdots, x_n)$ means the leading coefficient of $h(x_0X_0, \cdots, x_nX_n)$. A helpful polynomial contributes to the determinant with a factor less than p^d. The more helpful polynomials are added to the lattice, the better the condition for solving modular equations becomes. This means that the Coppersmith technique of finding the wanted small root becomes more and more effective, and the above bounds X_i become larger and larger. Therefore, we should choose as many helpful polynomials as possible. In this paper, our method can significantly improve previous results because the number of helpful polynomials dominates the number of all selected polynomials.

Step 4: Recovering the Desired Root. We have no assurance that the $n+1$ obtained polynomials $h_1, \cdots, h_{n+1}$ are algebraically independent. Under the following heuristic assumption that the $n+1$ polynomials define an algebraic variety of dimension zero, the corresponding equations can be solved using elimination techniques such as the Gröbner basis computation, and then the desired root $(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n)$ is recovered. In this paper, we justify the validity of our heuristic attack by computer experiments.

Assumption 1. *Let $h_1, \cdots, h_{n+1} \in \mathbb{Z}[x_0, x_1, \cdots, x_n]$ be the polynomials that are found by the Coppersmith technique. Then the ideal generated by the polynomial equations $h_1(x_0, x_1, \cdots, x_n) = 0$, $\cdots$, $h_{n+1}(x_0, x_1, \cdots, x_n) = 0$ has dimension zero.*

2.3 A Class of Modular Polynomial Equations

In this subsection, we translate the problem of recovering the hidden number in MIHNP into solving modular polynomial equations with small root.

For a given prime p, consider a hidden number $\alpha \in \mathbb{Z}_p$ and $n+1$ elements $t_0, t_1, \ldots, t_n \in \mathbb{Z}_p \setminus \{-\alpha\}$, chosen independently and uniformly at random. The goal is to recover α, given $n+1$ samples $\left(t_i, \mathrm{MSB}_\delta((\alpha + t_i)^{-1} \bmod p)\right)$.

Denote $u_i = \mathrm{MSB}_\delta\big((\alpha + t_i)^{-1} \bmod p\big)$ and $\widetilde{x}_i = \big((\alpha + t_i)^{-1} \bmod p\big) - u_i$, where unknown $\widetilde{x}_i$ satisfies $0 \le \widetilde{x}_i \le \frac{p}{2^\delta}$ for all $0 \le i \le n$. Hence, we obtain $(\alpha + t_i)(u_i + \widetilde{x}_i) = 1 \bmod p$, eliminate α from these equations and get the following relations

$$a_{0i} + b_{0i}\widetilde{x}_0 + c_{0i}\widetilde{x}_i + \widetilde{x}_0\widetilde{x}_i = 0 \bmod p, 1 \le i \le n, \tag{2}$$

where

$$\begin{aligned} a_{0i} &= u_0 u_i + (u_0 - u_i)(t_0 - t_i)^{-1} \bmod p, \\ b_{0i} &= u_i + (t_0 - t_i)^{-1} \bmod p, \\ c_{0i} &= u_0 - (t_0 - t_i)^{-1} \bmod p. \end{aligned} \tag{3}$$

If the corresponding $\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n$ are found out, then the hidden number α can be recovered. Hence, our goal is to find the desired root $(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n)$ of the following modular polynomial equations

$$f_{0i}(x_0, x_i) := a_{0i} + b_{0i}x_0 + c_{0i}x_i + x_0 x_i = 0 \bmod p, 1 \leq i \leq n, \tag{4}$$

where $n < p$ and $|\widetilde{x}_0|, |\widetilde{x}_1|, \cdots, |\widetilde{x}_n|$ are bounded by X. We take $X = \frac{p}{2^\delta}$ where δ is the number of known MSBs. Moreover, in the following analysis, we need that all $c_{01}, \cdots, c_{0n} \in \mathbb{Z}_p$ are distinct.

According to (3), we get $c_{0i} = u_0 - (t_0 - t_i)^{-1} \bmod p$ for $i = 1, \cdots, n$. Note that elements $t_0, t_1, \ldots, t_n \in \mathbb{Z}_p \setminus \{-\alpha\}$, chosen independently and uniformly, where α is the hidden number. The probability that all c_{0i} are distinct is equal to $\prod_{k=1}^{n-1}(1 - \frac{k}{p}) \approx e^{-\sum_{k=1}^{n-1}\frac{k}{p}} = e^{-\frac{n(n-1)}{2p}} \approx 1 - \frac{n^2-n}{2p}$, which is close to 1 for a sufficiently large p.

2.4 Order of Monomials

First, we describe reverse lexicographic order and graded lexicographic reverse order respectively. Please refer to [33, Section 21.2] for more details of these orders. Let $(i_1, \cdots, i_n)$ and $(i'_1, \cdots, i'_n)$ be integer vectors, where $i_m \geq 0, i'_m \geq 0$ for all $1 \leq m \leq n$.

Reverse Lexicographic Order: $(i'_1, \cdots, i'_n) \prec_{revlex} (i_1, \cdots, i_n) \Leftrightarrow$ the rightmost nonzero entry in $(i'_1 - i_1, \cdots, i'_n - i_n)$ is negative.

Graded Reverse Lexicographic Order: $(i'_1, \cdots, i'_n) \prec_{grevlex} (i_1, \cdots, i_n) \Leftrightarrow$
$\sum_{m=1}^{n} i'_m < \sum_{m=1}^{n} i_m$ or $\big(\sum_{m=1}^{n} i'_m = \sum_{m=1}^{n} i_m$ and $(i'_1, \cdots, i'_n) \prec_{revlex} (i_1, \cdots, i_n)\big)$.

Next, we consider the following order of monomials.

$$\begin{aligned} &x_0^{i'_0} x_1^{i'_1} \cdots x_n^{i'_n} \prec x_0^{i_0} x_1^{i_1} \cdots x_n^{i_n} \Leftrightarrow \\ &(i'_1, \cdots, i'_n) \prec_{grevlex} (i_1, \cdots, i_n) \text{ or } \big((i'_1, \cdots, i'_n) = (i_1, \cdots, i_n) \text{ and } i'_0 < i_0\big). \end{aligned} \tag{5}$$

It is worth noting that we treat i_0 differently than $i_1, \cdots, i_n$ for vector $(i_0, i_1, \cdots, i_n)$.

2.5 Elementary Symmetric Polynomials and Matrix

In this section, we first describe the definition of elementary symmetric polynomials. Please refer to [26, Section 3.1] for more details.

The elementary symmetric polynomials on m variables $\{y_1, \cdots, y_m\}$, written as $\sigma_k(y_1, \cdots, y_m)$ for $k = 0, 1, \cdots, m$, are defined by

$$\begin{cases} \sigma_0(y_1, \cdots, y_m) = 1, \\ \sigma_1(y_1, \cdots, y_m) = \sum\limits_{1 \le i \le m} y_i, \\ \sigma_2(y_1, \cdots, y_m) = \sum\limits_{1 \le i < j \le m} y_i y_j, \\ \sigma_3(y_1, \cdots, y_m) = \sum\limits_{1 \le i < j < k \le m} y_i y_j y_k, \\ \qquad\qquad \vdots \\ \sigma_m(y_1, \cdots, y_m) = \prod\limits_{1 \le i \le m} y_i. \end{cases}$$

From the above formulas, we can see that $\sigma_k(y_1, \cdots, y_m)$ is the sum of all products of exactly k distinct y_i's.

Next, we define the following $s \times s$ matrix whose entries depend on elementary symmetric polynomials, which will be used in Sections 3 and 4. Consider the matrix

$$M_{j_1, \cdots, j_s} := \begin{pmatrix} \sigma_{s-1}(\wedge_0) & \cdots & \sigma_1(\wedge_0) & \sigma_0(\wedge_0) \\ \sigma_{s-1}(\wedge_1) & \cdots & \sigma_1(\wedge_1) & \sigma_0(\wedge_1) \\ & \cdots & & \\ \sigma_{s-1}(\wedge_{s-1}) & \cdots & \sigma_1(\wedge_{s-1}) & \sigma_0(\wedge_{s-1}) \end{pmatrix},$$

where $\sigma_i(\wedge_l)$ is the i-th elementary symmetric polynomial on

$$\wedge_l := \{c_{0,j_1}, \cdots, c_{0,j_s}\} \setminus \{c_{0,j_{l+1}}\} \text{ with } 0 \le i \le s-1,\ 0 \le l \le s-1.$$

Here $1 \le j_1 < \cdots < j_s \le n$ and $c_{0,j_{l+1}}$ is the coefficient of variable $x_{j_{l+1}}$ in the polynomial $f_{0,j_{l+1}}$ in (4).

For indexes $j_1, \cdots, j_s$, row u and column v of matrix $M_{j_1, \cdots, j_s}$ is the evaluation of σ_{s-v} on all the variables c_{0,j_i} except c_{0,j_u}, where $1 \le u, v \le s$. From another perspective, we first let polynomials

$$G_u(x) := (x + c_{0,j_1}) \cdots (x + c_{0,j_{u-1}})(x + c_{0,j_{u+1}}) \cdots (x + c_{0,j_s}) \text{ for all } 1 \le u \le s.$$

For $1 \le v \le s$, we have that the coefficient of $G_u(x)$ on monomial x^{s-v} is the evaluation of σ_{s-v} on all the variables c_{0,j_i} except c_{0,j_u}. In other words, row u and column v of matrix $M_{j_1, \cdots, j_s}$ is the coefficient of $G_u(x)$ on monomial x^{s-v}.

Lemma 3. *For a given prime p and any integer $s \ge 2$, define matrix $M_{j_1, \cdots, j_s}$ as above. Then $M_{j_1, \cdots, j_s}$ is invertible over $\mathbb{Z}_{p^{s-1}}$ if $c_{0,j_1}, \cdots, c_{0,j_s}$ are distinct in $\mathbb{Z}_p$.*

Proof. Since p is a prime number, we get that $M_{j_1, \cdots, j_s}$ is invertible over $\mathbb{Z}_{p^{s-1}}$ if and only if $M_{j_1, \cdots, j_s}$ is invertible over $\mathbb{Z}_p$. Note that row u of matrix $M_{j_1, \cdots, j_s}$ is the coefficient vector of polynomial $G_u(x)$

on the basis $(1, x, \cdots, x^{s-1})$ for all $1 \leq u \leq s$. Hence, $M_{j_1,\cdots,j_s}$ is invertible over $\mathbb{Z}_p$ if and only if polynomials $G_1(x)$, $\cdots$, $G_s(x)$ are linearly independent over $\mathbb{Z}_p$.

Suppose that there exist integers $c_1, \cdots, c_s$ satisfying

$$c_1 G_1(x) + \cdots + c_s G_s(x) = 0. \tag{6}$$

Note that $G_u(x) = (x+c_{0,j_1}) \cdots (x+c_{0,j_{u-1}})(x+c_{0,j_{u+1}}) \cdots (x+c_{0,j_s})$ for $1 \leq u \leq s$. Taking modulo $x + c_{0,j_u}$ on both sides of (6), we obtain that

$$c_u G_u(x) \equiv 0 \bmod (x + c_{0,j_u}) \text{ for } u = 1, \cdots, s.$$

If $c_{0,j_1}, \cdots, c_{0,j_s}$ are distinct in $\mathbb{Z}_p$, then the polynomials $x+c_{0,j_1}, \cdots, x+c_{0,j_s}$ are pairwise coprime. Furthermore, we have $\gcd(x + c_{0,j_u}, G_u(x)) = 1$. Combining this relation with the above equations, we deduce that $c_1 = \cdots = c_s = 0$. Based on (6), we have that the polynomials $G_1(x)$, $\cdots$, $G_s(x)$ are linearly independent over $\mathbb{Z}_p$. In other words, $M_{j_1,\cdots,j_s}$ is invertible over $\mathbb{Z}_{p^{s-1}}$ if $c_{0,j_1}, \cdots, c_{0,j_s}$ are distinct in $\mathbb{Z}_p$. □

Note that the indexes $j_1, \cdots, j_s$ satisfy $1 \leq j_1 < \cdots < j_s \leq n$. We always have that elements $c_{0,j_1}, \cdots, c_{0,j_s}$ are from $c_{01}, \cdots, c_{0n}$. According to the analysis of Section 2.3, we know that $c_{01}, \cdots, c_{0n}$ are distinct with a probability close to 1 for a sufficiently large p. Hence, from Lemma 3, matrix $M_{j_1,\cdots,j_s}$ is invertible over $\mathbb{Z}_{p^{s-1}}$ with a probability close to 1 for a sufficiently large p.

3 The Strategy for Solving a Class of Modular Polynomial Equations

In this section, we first present theorems to solve the equation system (4), and then give the corresponding results for solving MIHNP.

Theorem 1. *For any given positive integer d, take positive integer $n = d^{3+o(1)}$. Given a sufficiently large prime $p = 2^{\omega(d^{3d+2})}$, and polynomials $f_{0j}(x_0, x_j)$ with $1 \leq j \leq n$ in (4), under Assumption 1, one can compute the desired root $(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n)$ with a probability close to 1, if the bound X of $|\widetilde{x}_0|, |\widetilde{x}_1|, \cdots, |\widetilde{x}_n|$ satisfies*

$$X < p^{1-\frac{1}{d+1}-o(\frac{1}{d})}. \tag{7}$$

The corresponding time complexity is polynomial in $\log_2 p$ for any constant d, where the complexity of the LLL algorithm grows as $d^{\mathcal{O}(d)}$ and the complexity of the Gröbner basis computation grows as $(2d)^{\mathcal{O}(n^2)}$.

Proof. For any given positive integer d, and integers n, t satisfying $n \geq d+1$, $0 \leq t \leq d$, we first construct the polynomials $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ for all vectors $(i_0, i_1, \cdots, i_n) \in I(n, d, t)$, where

$$\begin{aligned} I(n,d,t) = &\{(i_0, i_1, \cdots, i_n) \mid 0 \leq i_0 \leq d, 0 \leq i_1, \cdots, i_n \leq 1, 0 \leq i_1 + \cdots + i_n \leq d\} \\ &\cup \{(i_0, i_1, \cdots, i_n) \mid 0 \leq i_0 \leq t, 0 \leq i_1, \cdots, i_n \leq 1, i_1 + \cdots + i_n = d+1\}. \end{aligned}$$

We will optimize integers n, t later. Denote the level $s := i_1 + \cdots + i_n$, where $0 \leq s \leq d+1$.

When $s = 0$, we construct $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = p^d x_0^{i_0}$ for $i_0 = 0, 1, \cdots, d$.

When $s = 1$, we construct

$$F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = \begin{cases} p^d x_1^{i_1} \cdots x_n^{i_n} & \text{for } i_0 = 0, \\ p^{d-1} x_0^{i_0-1} f_{01}^{i_1} \cdots f_{0n}^{i_n} & \text{for } 1 \le i_0 \le d. \end{cases}$$

When $2 \le s \le d+1$, if $s \le i_0 \le d$, we construct the polynomials

$$F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = p^{d-s} x_0^{i_0-s} f_{01}^{i_1} \cdots f_{0n}^{i_n}.$$

If $0 \le i_0 < s$, we construct the polynomials $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ as follows.

Notice that all integers $i_1, \cdots, i_n$ are equal to 0 or 1. We can rewrite $f_{01}^{i_1} \cdots f_{0n}^{i_n} = f_{0,j_1} \cdots f_{0,j_s}$, where $j_1, \cdots, j_s$ are some integers satisfying $1 \le j_1 < \cdots < j_s \le n$. Define $M_{j_1,\cdots,j_s}$ as Section 2.5. Based on Lemma 3, we have that $M_{j_1,\cdots,j_s}$ is invertible in $\mathbb{Z}_{p^{s-1}}$ with a probability close to 1 for a sufficiently large p. Let $M^{-1}_{j_1,\cdots,j_s}$ be the inverse of $M_{j_1,\cdots,j_s}$ modulo p^{s-1}. Then we generate s polynomials $g_0(x_0, x_{j_1}, \ldots, x_{j_s})$, $g_1(x_0, x_{j_1}, \ldots, x_{j_s})$, $\cdots$, $g_{s-1}(x_0, x_{j_1}, \ldots, x_{j_s})$ according to the following way:

$$\begin{pmatrix} g_0(x_0, x_{j_1}, \ldots, x_{j_s}) \\ g_1(x_0, x_{j_1}, \ldots, x_{j_s}) \\ \vdots \\ g_{s-1}(x_0, x_{j_1}, \ldots, x_{j_s}) \end{pmatrix} = M^{-1}_{j_1,\cdots,j_s} \cdot \begin{pmatrix} x_{j_1} f_{0,j_2} \cdots f_{0,j_s} \\ f_{0,j_1} x_{j_2} \cdots f_{0,j_s} \\ \vdots \\ f_{0,j_1} \cdots f_{0,j_{s-1}} x_{j_s} \end{pmatrix} \bmod p^{s-1}. \tag{8}$$

Here, $g_0(x_0, x_{j_1}, \ldots, x_{j_s})$, $g_1(x_0, x_{j_1}, \ldots, x_{j_s})$, $\cdots$, $g_{s-1}(x_0, x_{j_1}, \ldots, x_{j_s})$ are treated as the corresponding polynomials over $\mathbb{Z}$.

Further, we define

$$F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = p^{d+1-s} \cdot g_{i_0}(x_0, x_{j_1}, \ldots, x_{j_s}), \tag{9}$$

where

$$\begin{cases} 0 \le i_0 \le s-1 & \text{for } 0 \le s \le d, \\ 0 \le i_0 \le t & \text{for } s = d+1. \end{cases}$$

Note that $g_{i_0}(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n) = 0 \bmod p^{s-1}$. The corresponding

$$F_{i_0,i_1,\cdots,i_n}(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n) = 0 \bmod p^d.$$

In fact, for all tuples $(i_0, i_1, \cdots, i_n) \in I(n, d, t)$, we always get

$$F_{i_0,i_1,\cdots,i_n}(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n) = 0 \bmod p^d.$$

Next, we present the following lemma in order to construct a triangular lattice basis matrix. The corresponding proof is given in Section 4.

Lemma 4. *Define polynomials $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ as above, where the corresponding monomials are arranged according to the order (5). Let $\mathcal{L}(n, d, t)$ be a lattice spanned by the coefficient vectors of polynomials*

$$F_{i_0,i_1,\cdots,i_n}(x_0X, x_1X, \cdots, x_nX),$$

for all $(i_0, i_1, \cdots, i_n) \in I(n, d, t)$. Then the basis matrix becomes triangular if these coefficient vectors are arranged according to the leading monomial of the corresponding $F_{i_0,i_1,\cdots,i_n}(x_0, x_1, \cdots, x_n)$ from low to high. Moreover, diagonal elements of the triangular basis matrix of $\mathcal{L}(n, d, t)$ are as follows:

$$\begin{cases} p^{d-s}X^{i_0+s} & \text{for } i_0 \geq s, \\ p^{d+1-s}X^{i_0+s} & \text{for } i_0 < s, \end{cases} \tag{10}$$

where $s = i_1 + \cdots + i_n$.

We will provide an example for lattice $\mathcal{L}(n, d, t)$ in full version. It is easy to see that the dimension of $\mathcal{L}(n, d, t)$ is equal to

$$\dim(\mathcal{L}(n, d, t)) = (d + 1)\sum_{s=0}^{d}\binom{n}{s} + (t + 1)\binom{n}{d + 1}. \tag{11}$$

We compute the determinant of $\mathcal{L}(n, d, t)$ as

$$\det(\mathcal{L}(n, d, t)) = p^{\alpha(n,d)} \cdot X^{\beta(n,d,t)}, \tag{12}$$

where

$$\begin{aligned} \alpha(n, d) &= d(d + 1)\sum_{s=0}^{d}\tbinom{n}{s} - d\sum_{s=0}^{d} s\tbinom{n}{s}, \\ \beta(n, d, t) &= \tfrac{d(d+1)}{2}\sum_{s=0}^{d}\tbinom{n}{s} + (d + 1)\sum_{s=0}^{d} s\tbinom{n}{s} + \tfrac{(2d+t+2)(t+1)}{2}\tbinom{n}{d+1}. \end{aligned}$$

The detailed computation is left in Appendix B. By the property of LLL algorithm and Howgrave-Graham's lemma, if Condition (1) is satisfied, namely,

$$2^{\frac{w(w-1)}{4(w-n)}} \cdot \det(\mathcal{L}(n, d, t))^{\frac{1}{w-n}} < \frac{p^d}{\sqrt{w}}, \tag{13}$$

where $w = \dim(\mathcal{L}(n, d, t))$, after reduction of lattice we get $n + 1$ polynomials which contain the root $(\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n)$ over integers. Under Assumption 1, we can find $\widetilde{x}_0, \widetilde{x}_1, \cdots, \widetilde{x}_n$.

Plugging (11) and (12) into (13), we obtain the condition

$$X < \left(2^{-\frac{w(w-1)}{4\beta(n,d,t)}} w^{-\frac{w-n}{2\beta(n,d,t)}}\right) \cdot p^{\lambda(n,d,t)}, \tag{14}$$

where

$$\lambda(n, d, t) := \frac{d(w - n) - \alpha(n, d)}{\beta(n, d, t)} = \frac{2d(t + 1)\binom{n}{d+1} + 2d\sum_{s=2}^{d} s\binom{n}{s}}{(2d + 2 + t)(t + 1)\binom{n}{d+1} + d(d + 1)\sum_{s=0}^{d}\binom{n}{s} + 2(d + 1)\sum_{s=0}^{d} s\binom{n}{s}}.$$

For a sufficiently large $p = 2^{\omega(d^{3d+2})}$, the above condition reduces to $X < p^{\lambda(n,d,t)}$. By taking the optimal $t = 0$ and $n = d^{3+o(1)}$, this condition further becomes

$$X < p^{1-\frac{1}{d+1}-o(\frac{1}{d})}.$$

The detailed analysis is presented in Appendix C.

Finally, we analyze the time complexity of our algorithm. Note that the running time of the LLL algorithm depends on the dimension and the maximal bit size of input basis matrix. For the optimal case that $t = 0$ and $n = d^{3+o(1)}$, the dimension of $\mathcal{L}(n, d, 0)$ is

$$(d+1)\sum_{s=0}^{d}\binom{n}{s} + \binom{n}{d+1} = \mathcal{O}(n^{d+1}) = \mathcal{O}(d^{\mathcal{O}(d)}).$$

The bit size of the entries in the basis matrix can be bounded by $2d\log_2 p$ based on (10). Hence, according to [24], the time complexity of the LLL algorithm is equal to

$$\mathrm{poly}\big(2d\log_2 p, \mathcal{O}(d^{\mathcal{O}(d)})\big) = \mathcal{O}((\log_2 p)^{\mathcal{O}(1)} d^{\mathcal{O}(d)}). \tag{15}$$

Moreover, we use the Gröbner basis computation to solve the polynomials obtained from the LLL algorithm. The running time of the Gröbner basis computation relies on degrees of the polynomials in the Gröbner basis and the number of variables in these polynomials [18,10]. Under Assumption 1, these polynomials generate a zero-dimensional ideal. Note that the maximal degree is $2d$ and the number of variables is $n + 1$. Based on [10], we get that the time complexity of the Gröbner basis computation is

$$\mathrm{poly}((2d)^{(n+1)^2}) = (2d)^{\mathcal{O}(n^2)}. \tag{16}$$

Therefore, the overall time complexity is equal to $\mathcal{O}((\log_2 p)^{\mathcal{O}(1)} d^{\mathcal{O}(d)}) + (2d)^{\mathcal{O}(n^2)}$, which is polynomial in $\log_2 p$ for any constant d, where the complexity of the LLL algorithm grows as $d^{\mathcal{O}(d)}$ and the complexity of the Gröbner basis computation grows as $(2d)^{\mathcal{O}(n^2)}$. □

Remark 1. Similar to [4,35], we choose the same polynomials $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ where any $(i_0, i_1, \cdots, i_n)$ lies in the set $\{(i_0, i_1, \cdots, i_n) \mid 0 \le i_0 \le d, 0 \le i_1, \cdots, i_n \le 1, 0 \le i_1 + \cdots + i_n \le d\}$. The difference from [4,35] is that we add new polynomials $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ where any $(i_0, i_1, \cdots, i_n)$ belongs to the set $\{(i_0, i_1, \cdots, i_n) \mid 0 \le i_0 \le t, 0 \le i_1, \cdots, i_n \le 1, i_1 + \cdots + i_n = d+1\}$, where $0 \le t \le d$. This corresponds to the case of $s = d + 1$ in the proof of Theorem 1. When $t = 0$, the involved lattice $\mathcal{L}(n, d, t)$ is optimized.

According to (8) and (9), we get that newly added polynomials $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ are linear combinations of $d+1$ polynomials $x_{j_1} f_{0,j_2} \cdots f_{0,j_{d+1}}$, $f_{0,j_1} x_{j_2} \cdots f_{0,j_{d+1}}$, $\cdots$, $f_{0,j_1} \cdots f_{0,j_d} x_{j_{d+1}}$, which have common monomials. Concretely speaking,

$$F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = g_{i_0}(x_0, x_{j_1}, \ldots, x_{j_{d+1}}),$$

where $1 \le j_1, \cdots, j_{d+1} \le n$ satisfying $x_1^{i_1} \cdots x_n^{i_n} = x_{j_1} \cdots x_{j_{d+1}}$. These newly added polynomials $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ are linearly independent of previous $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ according to the analysis of Lemma 4.

Finally, we explain why this method can work efficiently. Consider the optimized case of $t = 0$, we have $i_0 = 0$ according to $0 \leq i_0 \leq t$. Note that $i_1, \cdots, i_n$ satisfy $0 \leq i_1, \cdots, i_n \leq 1, i_1 + \cdots + i_n = d + 1$. It implies that we added $\binom{n}{d+1}$ such polynomials $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ into the involved lattice $\mathcal{L}(n, d, t)$. Based on (7) and (10), we get that every newly added polynomial $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ contributes to a diagonal element X^{d+1} ($i_1 = 0$ and $s = d + 1$ in (10)), which is smaller than modulo p^d. Such a $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ is called a *helpful polynomial* [21,29]. Hence, we have $\binom{n}{d+1}$ helpful polynomials for lattice $\mathcal{L}(n, d, t)$. Since $\dim(\mathcal{L}(n, d, t)) = \binom{n}{d+1}(1 + o(1))$ for the optimized case of $t = 0$, we get that the number of all selected polynomials for lattice $\mathcal{L}(n, d, t)$ is $\binom{n}{d+1}(1 + o(1))$. It implies that newly added $\binom{n}{d+1}$ helpful polynomials are dominant. This is the fundamental reason behind the effectiveness of our approach.

Since $X = \frac{p}{2^\delta}$ in the case of MIHNP, we give the following result.

Corollary 1. *For any given positive integer d, let positive integer $n = d^{3+o(1)}$. For a given sufficiently large prime $p = 2^{\omega(d^{3d+2})}$ and $n + 1$ given samples in MIHNP, the hidden number can be recovered with a probability close to 1 under Assumption 1 if the number δ of known MSBs satisfies*

$$\frac{\delta}{\log_2 p} \geq \frac{1}{d+1} + o(\frac{1}{d}). \tag{17}$$

The involved time complexity is polynomial in $\log_2 p$ for any constant d, where the complexity of the LLL algorithm grows as $d^{\mathcal{O}(d)}$ and the complexity of the Gröbner basis computation grows as $(2d)^{\mathcal{O}(n^2)}$.

Remark 2. The algorithm in Theorem 1 can be also applied to the attack case of ICG, as described in prior work [35].

Remark 3. Taking $d = 2$, the asymptotic result of $\delta/\log_2 p$ is equal to $\frac{1}{3}$, which is the same as the previous best result [4,35]. When $d > 2$, our asymptotic bound is $\frac{1}{d+1} < \frac{1}{3}$, resulting in the best asymptotic result known so far.

Remark 4. Similar to Appendix A, we also use notations $\rho = \delta/\log_2 p$ and $k = \log_2 p$, where $0 < \rho < 1$. According to (17), namely, $\rho \geq \frac{1}{d+1} + o(\frac{1}{d})$. In the asymptotic sense, we have $\rho > \frac{1}{d+1}$, i.e., $d > \frac{1}{\rho} - 1$. Hence, we can take $d = 1/\rho$ asymptotically. Then, (15) and (16) respectively become $\mathcal{O}\big(k^{\mathcal{O}(1)}(\frac{1}{\rho})^{\mathcal{O}(\frac{1}{\rho})}\big)$ and $\big(\frac{2}{\rho}\big)^{\mathcal{O}((\frac{1}{\rho})^{\mathcal{O}(1)})}$. Hence, the overall time complexity is polynomial in $\log_2 p$ for any constant ρ (i.e., δ is a constant fraction of $\log_2 p$). Note that $\rho = \delta/\log_2 p$ tends to 0 as d becomes large. It implies that the asymptotic lower bound of $\delta/\log_2 p$ is equal to 0. However, in order to ensure that $\delta/\log_2 p$ is close to 0, a huge lattice dimension is required, which is because that the dimension of the involved lattice is equal to $\mathcal{O}(d^{\mathcal{O}(d)}) = \mathcal{O}\big((\frac{1}{\rho})^{\mathcal{O}(\frac{1}{\rho})}\big) = \mathcal{O}\big((\frac{\log_2 p}{\delta})^{\mathcal{O}(\frac{\log_2 p}{\delta})}\big)$.

Figure 1 shows that the theoretical values of $\delta/\log_2 p$ for different lattice dimension, where the smallest dimension is calculated among different n, d, t for the fixed $\delta/\log_2 p$. One can achieve $\delta/\log_2 p < \frac{1}{3}$ by using a lattice of dimension 209899. Theoretical value of the involved $\lambda(n, d, t)$ in this case is 0.671. Corresponding parameters are $n = 45, d = 3, t = 0$.

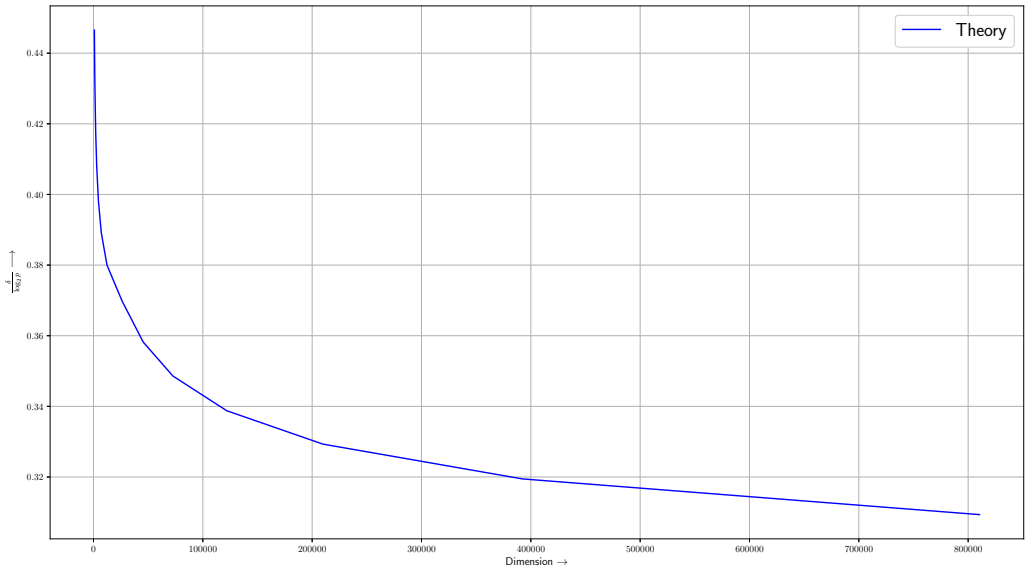

Fig. 1. Theoretical bound of $\delta/\log_2 p$ for different dimensions.

4 Proof of Lemma 4

Proof. First, we will show that the leading term of $F_{i_0,i_1,\cdots,i_n}(x_0, x_1, \cdots, x_n)$ according to the order (5) is as follows:

- $p^{d-s}x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n}$ $\quad i_0 \geq s,$
- $p^{d+1-s}x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n}$ $\quad i_0 < s,$

where $s = i_1 + \cdots + i_n$.

For the case of $s = 0$, $F_{i_0,i_1,\cdots,i_n}(x_0, x_1, \cdots, x_n) = p^d x_0^{i_0}$ for $i_0 \geq 0$. Obviously, the corresponding leading term is $p^{d-s}x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n}$ where $i_0 \geq s = 0$.

For the case of $s = 1$, we have

$$F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = \begin{cases} p^d x_1^{i_1}\cdots x_n^{i_n} & \text{for } i_0 = 0, \\ p^{d-1}x_0^{i_0-1}f_{01}^{i_1}\cdots f_{0n}^{i_n} & \text{for } 1 \leq i_0 \leq d. \end{cases}$$

For $i_0 = 0$, the leading term of $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = p^d x_1^{i_1}\cdots x_n^{i_n}$ can be rewritten as $p^{d+1-s}x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n}$ since $s = 1$. For $i_0 \geq 1$, $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = p^{d-1}x_0^{i_0-1}f_{01}^{i_1}\cdots f_{0n}^{i_n}$. We

analyze its leading term as follows. Note that $f_{0j} = a_{0j} + b_{0j}x_0 + c_{0j}x_j + x_0x_j$ for $1 \le j \le n$. Based on the order (5), we get

$$1 \prec x_0 \prec x_j \prec x_0x_j \text{ for } j = 1, \cdots, n.$$

So the leading term of f_{0j} is x_0x_j. Furthermore, the leading term of $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = p^{d-1}x_0^{i_0-1}f_{01}^{i_1}\cdots f_{0n}^{i_n}$ is equal to

$$p^{d-1}x_0^{i_0-1}(x_0x_1)^{i_1}\cdots(x_0x_n)^{i_n} = p^{d-s}x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n},$$

where $i_0 \ge s = 1$.

For the case of $2 \le s \le d+1$, if $s \le i_0 \le d$, we define

$$F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = p^{d-s}x_0^{i_0-s}f_{01}^{i_1}\cdots f_{0n}^{i_n}.$$

In this situation, the leading term of $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ is

$$p^{d-s}x_0^{i_0-s}(x_0x_1)^{i_1}\cdots(x_0x_n)^{i_n} = p^{d-s}x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n}.$$

For the following situations

$$\begin{cases} 0 \le i_0 \le s-1 & \text{for } 0 \le s \le d, \\ 0 \le i_0 \le t & \text{for } s = d+1, \end{cases}$$

where $0 \le t \le d$, we define

$$F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = p^{d+1-s} \cdot g_{i_0}(x_0, x_{j_1}, \ldots, x_{j_s}).$$

Our goal is to show that $p^{d+1-s}x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n}$ is the leading term of the corresponding polynomial $F_{i_0,i_1,\ldots,i_n}(x_0,\ x_1, \cdots, x_n)$. Note that $f_{01}^{i_1}\cdots f_{0n}^{i_n}$ is expressed as $f_{0,j_1}\cdots f_{0,j_s}$ in this situation. It is easy to deduce that $x_1^{i_1}\cdots x_n^{i_n} = x_{j_1}\cdots x_{j_s}$ by comparing terms of $f_{01}^{i_1}\cdots f_{0n}^{i_n}$ and $f_{0,j_1}\cdots f_{0,j_s}$. Hence we aim to prove that $x_0^{i_0}x_{j_1}\cdots x_{j_s}$ is the leading term of $g_{i_0}(x_0, x_{j_1}, \ldots, x_{j_s})$.

According to (8), i.e.,

$$\begin{pmatrix} g_0(x_0, x_{j_1}, \ldots, x_{j_s}) \\ g_1(x_0, x_{j_1}, \ldots, x_{j_s}) \\ \vdots \\ g_{s-1}(x_0, x_{j_1}, \ldots, x_{j_s}) \end{pmatrix} = M_{j_1,\cdots,j_s}^{-1} \cdot \begin{pmatrix} x_{j_1}f_{0,j_2}\cdots f_{0,j_s} \\ f_{0,j_1}x_{j_2}\cdots f_{0,j_s} \\ \vdots \\ f_{0,j_1}\cdots f_{0,j_{s-1}}x_{j_s} \end{pmatrix} \bmod p^{s-1},$$

we get that $g_{i_0}(x_0, x_{j_1}, \ldots, x_{j_s})$ is some linear combination of the following polynomials

$$x_{j_1}f_{0,j_2}\cdots f_{0,j_s}, f_{0,j_1}x_{j_2}\cdots f_{0,j_s}, \cdots, f_{0,j_1}\cdots f_{0,j_{s-1}}x_{j_s}.$$

Note that these polynomials have common monomials

$$x_{j_1}\cdots x_{j_s}, x_0x_{j_1}\cdots x_{j_s}, \cdots, x_0^{s-1}x_{j_1}\cdots x_{j_s}. \tag{18}$$

Let the polynomial $g_l^*(x_0, x_{j_1}, \ldots, x_{j_s})$ $(0 \le l \le s-1)$ be composed of the terms in the polynomial $f_{0,j_1} \cdots f_{0,j_l} x_{j_{l+1}} f_{0,j_{l+2}} \cdots f_{0,j_s}$ except the corresponding terms of common monomials in (18). Then we can rewrite

$$\begin{aligned} f_{0,j_1} \cdots f_{0,j_l} x_{j_{l+1}} f_{0,j_{l+2}} \cdots f_{0,j_s} &= x_{j_{l+1}} \cdot \prod_{k \neq l+1} (c_{0,j_k} x_{j_k} + x_0 x_{j_k}) + g_l^*(x_0, x_{j_1}, \ldots, x_{j_s}) \\ &= (x_{j_1} \cdots x_{j_s}) \cdot \prod_{k \neq l+1} (x_0 + c_{0,j_k}) + g_l^*(x_0, x_{j_1}, \ldots, x_{j_s}) \\ &= (x_{j_1} \cdots x_{j_s}) \cdot \sum_{i=0}^{s-1} \left(\sigma_i(\wedge_l) \cdot x_0^{s-1-i} \right) + g_l^*(x_0, x_{j_1}, \ldots, x_{j_s}), \end{aligned}$$

where $\wedge_l = \{c_{0,j_1}, \cdots, c_{0,j_s}\} \setminus \{c_{0,j_{l+1}}\}$ and σ_i is the i-th elementary symmetric polynomial. Furthermore, we express the above equalities for all $0 \le l \le s-1$ by using the matrix equation as follows:

$$\begin{pmatrix} x_{j_1} f_{0,j_2} \cdots f_{0,j_s} \\ f_{0,j_1} x_{j_2} \cdots f_{0,j_s} \\ \vdots \\ f_{0,j_1} \cdots f_{0,j_{s-1}} x_{j_s} \end{pmatrix} = M_{j_1,\cdots,j_s} \cdot \begin{pmatrix} x_{j_1} \cdots x_{j_s} \\ x_0 x_{j_1} \cdots x_{j_s} \\ \vdots \\ x_0^{s-1} x_{j_1} \cdots x_{j_s} \end{pmatrix} + \begin{pmatrix} g_0^*(x_0, x_{j_1}, \ldots, x_{j_s}) \\ g_1^*(x_0, x_{j_1}, \ldots, x_{j_s}) \\ \vdots \\ g_{s-1}^*(x_0, x_{j_1}, \ldots, x_{j_s}) \end{pmatrix}. \tag{19}$$

Plugging (19) into (8), we obtain

$$\begin{pmatrix} g_0(x_0, x_{j_1}, \ldots, x_{j_s}) \\ \vdots \\ g_{i_0}(x_0, x_{j_1}, \ldots, x_{j_s}) \\ \vdots \\ g_{s-1}(x_0, x_{j_1}, \ldots, x_{j_s}) \end{pmatrix} = \begin{pmatrix} x_{j_1} \cdots x_{j_s} \\ \vdots \\ x_0^{i_0} x_{j_1} \cdots x_{j_s} \\ \vdots \\ x_0^{s-1} x_{j_1} \cdots x_{j_s} \end{pmatrix} + M_{j_1,\cdots,j_s}^{-1} \begin{pmatrix} g_0^*(x_0, x_{j_1}, \ldots, x_{j_s}) \\ \vdots \\ g_{i_0}^*(x_0, x_{j_1}, \ldots, x_{j_s}) \\ \vdots \\ g_{s-1}^*(x_0, x_{j_1}, \ldots, x_{j_s}) \end{pmatrix} \tag{20}$$

in the sense of modulo p^{s-1}. According to (20), in order to prove that $x_0^{i_0} x_{j_1} \cdots x_{j_s}$ is the leading monomial of $g_{i_0}(x_0, x_{j_1}, \ldots, x_{j_s})$, we need to analyze that all monomials from the following polynomials

$$g_0^*(x_0, x_{j_1}, \ldots, x_{j_s}), g_1^*(x_0, x_{j_1}, \ldots, x_{j_s}), \cdots, g_{s-1}^*(x_0, x_{j_1}, \ldots, x_{j_s})$$

are lower than $x_0^{i_0} x_{j_1} \cdots x_{j_s}$ based on the order (5).

From (19), we can deduce that the monomial set from these polynomials $g_0^*, g_1^*, \cdots, g_{s-1}^*$ is equal to

$$\left\{ x_0^{r_0} x_{k_1} \cdots x_{k_m} \mid 0 \le r_0 \le d, \{k_1, \cdots, k_m\} \subsetneqq \{j_1, \cdots, j_s\} \right\}.$$

It implies that for any monomial $x_0^{r_0} x_{k_1} \cdots x_{k_m}$ from the above monomial set, we have $m < s$. Therefore, we get $x_0^{r_0} x_{k_1} \cdots x_{k_m} \prec x_0^{i_0} x_{j_1} \cdots x_{j_s}$ according to the order (5). In other words, $x_0^{i_0} x_{j_1} \cdots x_{j_s}$

is the leading monomial of $g_{i_0}(x_0, x_{j_1}, \ldots, x_{j_s})$. Hence, $p^{d+1-s}x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n}$ is the leading term of $F_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ in this situation.

Next, we will show that the basis matrix of $\mathcal{L}(n,d,t)$ is triangular based on the leading monomials of the polynomials $F_{i_0,i_1,\cdots,i_n}(x_0,x_1,\cdots,x_n)$ from low to high. Note that the basis matrix of $\mathcal{L}(n,d,t)$ consists of the coefficient vectors of the polynomials $F_{i_0,i_1,\cdots,i_n}(x_0X,x_1X,\cdots,x_nX)$. It is easy to see that there is a one-to-one correspondence between the polynomials $F_{i_0,i_1,\cdots,i_n}(x_0,x_1,\cdots,x_n)$ and $F_{i_0,i_1,\cdots,i_n}(x_0X,x_1X,\cdots,x_nX)$. So, our goal is to prove that all polynomials $F_{i_0,i_1,\cdots,i_n}(x_0, x_1,\cdots,x_n)$ form a triangular matrix according to the corresponding leading monomials from low to high.

For the case of $s=0$, the corresponding polynomial $F_{i_0,i_1,\cdots,i_n}(x_0,x_1,\cdots,x_n)$ is equal to $p^d x_0^{i_0}$, where $i_0 = 0,1,\cdots,d$. According to the order (5), we have $p^d \prec p^d x_0 \prec \cdots \prec p^d x_0^d$. It is obvious that all polynomials $F_{i_0,i_1,\cdots,i_n}(x_0,x_1,\cdots,x_n)$ for the case of $s=0$ generate a triangular matrix. The remaining proof is inductive. Suppose that all $F_{i'_0,i'_1,\cdots,i'_n}(x_0,x_1,\cdots,x_n)$ satisfying $x_0^{i'_0}x_1^{i'_1}\cdots x_n^{i'_n} \prec x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n}$ produce a triangular matrix as stated in Lemma 4. Then, we show that a matrix is still triangular with a new polynomial $F_{i_0,i_1,\cdots,i_n}(x_0,x_1,\cdots,x_n)$. According to the above analysis, we get that $x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n}$ is the leading monomial of $F_{i_0,i_1,\cdots,i_n}(x_0,x_1,\cdots,x_n)$. Without loss of generality, let $x_0^{k_0}x_1^{k_1}\cdots x_n^{k_n}$ be any monomial of the polynomial $F_{i_0,i_1,\cdots,i_n}(x_0,x_1,\cdots,x_n)$ except its leading monomial $x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n}$. Clearly, we have $x_0^{k_0}x_1^{k_1}\cdots x_n^{k_n} \prec x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n}$. Note that $x_0^{k_0}x_1^{k_1}\cdots x_n^{k_n}$ is the leading monomial of the polynomial $F_{k_0,k_1,\cdots,k_n}(x_0,x_1,\cdots,x_n)$. It implies that these monomials except $x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n}$ already appeared in the diagonals of a basis matrix. Hence, all polynomials $F_{i_0,i_1,\cdots,i_n}(x_0,x_1,\cdots,x_n)$ can produce a triangular matrix. In other words, the corresponding basis matrix of $\mathcal{L}(n,d,t)$ is triangular.

Since the leading term of $F_{i_0,i_1,\cdots,i_n}(x_0,x_1,\cdots,x_n)$ is as follows:

$$\begin{cases} p^{d-s}x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n} & \text{for } i_0 \geq s, \\ p^{d+1-s}x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n} & \text{for } i_0 < s, \end{cases}$$

where $s = i_1 + \cdots + i_n$, the diagonal elements of the triangular basis matrix of $\mathcal{L}(n,d,t)$ are as follows:

$$\begin{cases} p^{d-s}X^{i_0+i_1+\cdots+i_n} = p^{d-s}X^{i_0+s} & \text{for } i_0 \geq s, \\ p^{d+1-s}X^{i_0+i_1+\cdots+i_n} = p^{d+1-s}X^{i_0+s} & \text{for } i_0 < s. \end{cases}$$

□

5 Experimental Results

We implemented our lattice-based algorithm in SAGE 8.2 on a desktop with Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz, 3 GB RAM and 3 MB Cache using the L^2 reduction algorithm [23] from Nguyen and Stehlé. We tested the algorithm up to lattice dimension 291. Table 2 shows the experimental results for MIHNP with 1000 bit prime p. To confirm the claim that experimental outcome is better than theoretical bound based on (14), 100 experiments each time have been

Table 2. Experimental results on low bounds of $\delta/\log_2 p$ for 1000-bit p

n	d	t	Dimension	Low bounds of $\delta/\log_2 p$			Total Time (sec.)	
				Theory	Exp.	Success	LLL	Gröbner
3	2	1	23	0.712	0.595	100	11.01	1.22
6	1	0	29	0.699	0.575	100	29.42	13.36
4	2	0	37	0.660	0.560	100	190.41	10.65
4	2	1	41	0.638	0.550	99	636.12	54.36
5	2	0	58	0.614	0.530	100	2555.91	182.38
5	2	1	68	0.592	0.525	100	7889.82	809.49
6	2	0	86	0.582	0.505	100	18896.34	2185.73
6	2	1	106	0.564	0.505	100	33276.93	4974.75
7	2	0	122	0.558	0.495	100	175276.85	29248.29
7	2	1	157	0.546	0.490	100	312450.32	23893.45
8	2	0	167	0.540	0.475	100	872078.62	128818.90
6	3	0	183	0.547	0.485	100	897793.07	14371.18
9	2	0	222	0.528	0.460	100	5440027.10	858799.13
10	2	0	288	0.519	0.450	87	18250890.61	3415266.53
7	3	0	291	0.521	0.465	100	9223260.81	287510.60

carried out. We see that the success rate of each time is 100 percent for most cases. Total time means that sum of time for 100 experiments of LLL algorithm and Gröbner basis computation respectively.

We also perform one experiment for $n = 11, d = 2$ and $t = 0$. Corresponding lattice dimension is 366. Here theoretical bound of $\frac{\delta}{\log_2 p}$ is 0.514. As for other parameters, in this case also we get better experimental bound 0.445. Lattice reduction takes 336895.32 seconds and Gröbner computation takes 191821.33 seconds.

One may see Figure 2 for a comparison between theoretical and experimental values of $\delta/\log_2 p$ for different dimensions. One can see from the figure that for smaller lattice dimensions, experimental values substantially outperform their theoretical values.

6 Conclusion

We presented a heuristic polynomial time algorithm to find the hidden number in the modular inversion hidden number problem. After more than 15 years, we improved the bound for solving modular inversion hidden number problem for the first time. We also obtained the best attack result

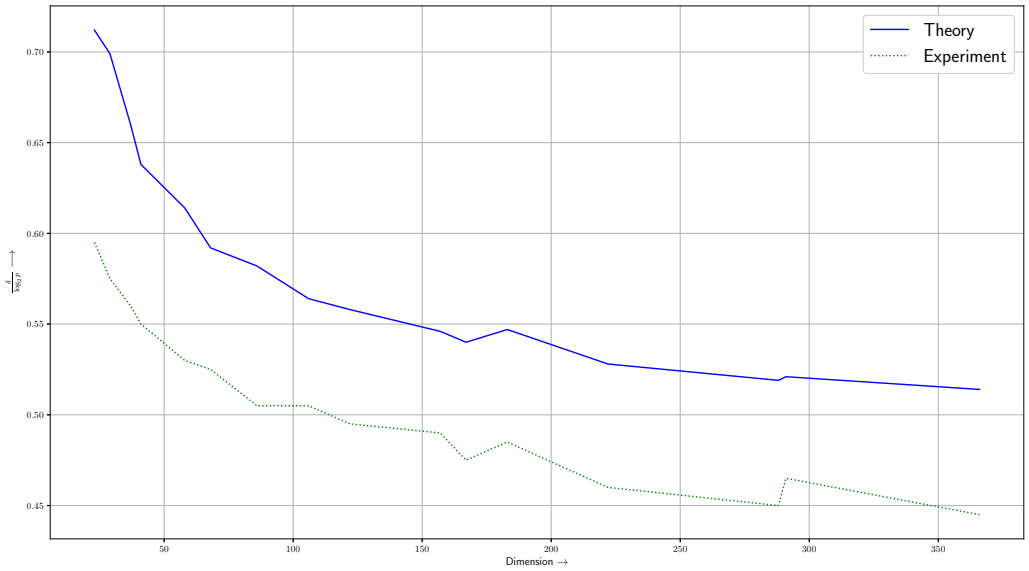

Fig. 2. Theoretical vs Experimental values of $\frac{\delta}{\log_2 p}$ for different dimensions.

on the inversive congruential generator till now.

Acknowledgments The authors would like to thank the reviewers of Eurocrypt 2019 and Crypto 2019 for their helpful comments and suggestions. The work of this paper was supported the National Natural Science Foundation of China (Grants 61732021, 61502488, 61572490 and 61702505). J. Xu is supported by China Scholarship Council (No. 201804910206). H. Wang is supported by the National Research Foundation, Prime Ministers Office, Singapore under its Strategic Capability Research Centres Funding Initiative and Singapore Ministry of Education under Research Grant MOE2016-T2-2-014(S). Y. Pan is supported by the National Center for Mathematics and Interdisciplinary Sciences, CAS.

References

1. Bauer, A., Vergnaud, D., Zapalowicz, J.C.: Inferring sequences produced by nonlinear pseudorandom number generators using coppersmith's methods. In Fischlin, M., Buchmann, J., Manulis, M., eds.: Public Key Cryptography-PKC 2012. Volume 7293 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 609–626

2. Bi, J., Coron, J., Faugère, J., Nguyen, P.Q., Renault, G., Zeitoun, R.: Rounding and chaining LLL: finding faster small roots of univariate polynomial congruences. In: Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings. (2014) 185–202
3. Blackburn, S.R., Gomez-perez, D., Gutierrez, J., Shparlinski, I.E.: Predicting nonlinear pseudorandom number generators. MATH. COMPUTATION **74** (2004) 2004
4. Boneh, D., Halevi, S., Howgrave-Graham, N.: The modular inversion hidden number problem. In: ASIACRYPT 2001, Springer (2001) 36–51
5. Boneh, D., Venkatesan, R.: Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In: CRYPTO 1996, Springer (1996) 129–142
6. Cohn, H., Heninger, N.: Approximate common divisors via lattices. The Open Book Series **1**(1) (2013) 271–293
7. Coppersmith, D.: Finding a small root of a bivariate integer equation; factoring with high bits known. In: EUROCRYPT 1996, Springer (1996) 178–189
8. Coppersmith, D.: Finding a small root of a univariate modular equation. In: EUROCRYPT 1996, Springer (1996) 155–165
9. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. Journal of Cryptology **10**(4) (1997) 233–260
10. Faugère, J., Gianni, P.M., Lazard, D., Mora, T.: Efficient computation of zero-dimensional Gröbner bases by change of ordering. J. Symb. Comput. **16**(4) (1993) 329–344
11. Herrmann, M., May, A.: Solving linear equations modulo divisors: On factoring given any bits. In: Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings. (2008) 406–424
12. Herrmann, M., May, A.: Attacking power generators using unravelled linearization: When do we output too much? In: Advances in Cryptology–ASIACRYPT 2009. Springer (2009) 487–504
13. Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In: Crytography and Coding. Springer (1997) 131–142
14. Howgrave-Graham, N.: Approximate integer common divisors. In Silverman, J., ed.: Cryptography and Lattices. Volume 2146 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2001) 51–66
15. Jochemsz, E., May, A.: A strategy for finding roots of multivariate polynomials with new applications in attacking RSA variants. In: ASIACRYPT 2006. Springer (2006) 267–282
16. Kakvi, S.A., Kiltz, E., May, A.: Certifying RSA. In: Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings. (2012) 404–414
17. Kannan, R.: Minkowski's convex body theorem and integer programming. Math. Oper. Res. **12**(3) (1987) 415–440
18. Lazard, D.: Gröbner-bases, gaussian elimination and resolution of systems of algebraic equations. In: Computer Algebra, EUROCAL '83, European Computer Algebra Conference, London, England, March 28-30, 1983, Proceedings. (1983) 146–156
19. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**(4) (1982) 515–534
20. Ling, S., Shparlinski, I.E., Steinfeld, R., Wang, H.: On the modular inversion hidden number problem. Journal of Symbolic Computation **47**(4) (2012) 358–367
21. May, A.: Using LLL-reduction for solving RSA and factorization problems. In: The LLL algorithm. Springer (2010) 315–348

22. Micciancio, D., Voulgaris, P.: A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. Electronic Colloquium on Computational Complexity (ECCC) **17** (2010) 14
23. Nguyen, P.Q., Stehlé, D.: An LLL algorithm with quadratic complexity. SIAM J. Comput. **39**(3) (2009) 874–903
24. Novocin, A., Stehlé, D., Villard, G.: An LLL-reduction algorithm with quasi-linear time complexity: Extended abstract. In: Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing. STOC '11, New York, NY, USA, ACM (2011) 403–412
25. Peng, L., Hu, L., Lu, Y., Xu, J., Huang, Z.: Cryptanalysis of dual RSA. Des. Codes Cryptography **83**(1) (2017) 1–21
26. Prasolov, V.V.: Polynomials, volume 11 of algorithms and computation in mathematics (2004)
27. Shani, B.: On the bit security of elliptic curve Diffie-Hellman. In: Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I. (2017) 361–387
28. Shparlinski, I.E.: Playing hide-and-seek with numbers: the hidden number problem, lattices, and exponential sums. In: proceeding of symposia in applied mathematics. Volume 62. (2005) 153–177
29. Takayasu, A., Kunihiro, N.: Better lattice constructions for solving multivariate linear equations modulo unknown divisors. In: Information Security and Privacy - 18th Australasian Conference, ACISP 2013, Brisbane, Australia, July 1-3, 2013. Proceedings. (2013) 118–135
30. Takayasu, A., Kunihiro, N.: How to generalize RSA cryptanalyses. In: Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part II. (2016) 67–97
31. Takayasu, A., Lu, Y., Peng, L.: Small CRT-exponent RSA revisited. In: Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II. (2017) 130–159
32. Tosu, K., Kunihiro, N.: Optimal bounds for multi-prime Φ-hiding assumption. In: Information Security and Privacy - 17th Australasian Conference, ACISP 2012, Wollongong, NSW, Australia, July 9-11, 2012. Proceedings. (2012) 1–14
33. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra (3. ed.). Cambridge University Press (2013)
34. Xu, J., Hu, L., Huang, Z., Peng, L.: Modular inversion hidden number problem revisited. In: Information Security Practice and Experience. Springer (2014) 537–551
35. Xu, J., Sarkar, S., Hu, L., Huang, Z., Peng, L.: Solving a class of modular polynomial equations and its relation to modular inversion hidden number problem and inversive congruential generator. Designs, Codes and Cryptography **86**(9) (Sep 2018) 1997–2033

A Asymptotic Time Complexities in Previous Works

The running time functions for solving MIHNP or ICG are not fully presented explicitly in previous works. For the sake of comparison, we analyze the corresponding running time functions according to the following way. Let $\rho = \delta/\log_2 p$ and $k = \log_2 p$, where $0 < \rho < 1$.

In [3, Theorem 1], the bound $\rho > \frac{3}{4}$ is shown for solving ICG with known $\mathcal{F}$ based on the SVP assumption. Since the involved lattice is 4-dimensional, the time complexity of the SVP algorithm is $k^{\mathcal{O}(1)}$, which is deterministic polynomial in the bit size of a given basis of the lattice for the fixed dimension [17].

In [20, Corollary 1], the bound $\rho \geq \frac{2}{3} + \varepsilon$ is presented to solve MIHNP based on the SVP assumption. By taking $\varepsilon = \rho - \frac{2}{3}$, the time complexity using SVP algorithm becomes $k^{\mathcal{O}(1)} 2^{\mathcal{O}(\frac{1}{\rho - \frac{2}{3}})}$ [22].

In [1, Section 3.4 and Theorem 2], the asymptotic bound $\rho \geq \frac{1}{2} + \frac{1}{2^{n+3}}$ is obtained to solve ICG with known $\mathcal{F}$ based on the Coppersmith technique, where $n+2$ denotes the number of unknown variables. Let $m = n^{\mathcal{O}(1)}$. The involved lattice dimension can be expressed as $\mathcal{O}(m^n)$, and the bit size of lattice basis matrix is at most km. Hence, the time complexity of the LLL algorithm is $(\mathcal{O}(m^n))^{\mathcal{O}(1)} \cdot (km)^{\mathcal{O}(1)} = \mathcal{O}\big(k^{\mathcal{O}(1)} n^{\mathcal{O}(n)}\big)$. For the Gröbner basis, the maximal degree of input polynomials is $2m$, and the number of unknown variables of input polynomials is $n+2$. Under Assumption 1, these polynomials generate a zero-dimensional Gröbner basis. We have that the time complexity of Gröbner basis computation is $(n+2)^{\mathcal{O}((2m)^2)} = n^{\mathcal{O}(n^2)}$ [10]. Based on the above bound $\rho \geq \frac{1}{2} + \frac{1}{2^{n+3}}$, we can take $n \approx \log_2(\frac{1}{\rho - \frac{1}{2}})$. Hence, time complexities of the LLL algorithm and the Gröbner basis computation are reduced to $\mathcal{O}\big(k^{\mathcal{O}(1)} \big(\log_2 \frac{1}{\rho - \frac{1}{2}}\big)^{\mathcal{O}(\log_2 \frac{1}{\rho - \frac{1}{2}})}\big)$ and $\big(\log_2 \frac{1}{\rho - \frac{1}{2}}\big)^{\mathcal{O}\big((\log_2 \frac{1}{\rho - \frac{1}{2}})^2\big)}$ respectively.

In [34, Theorem 1], the asymptotic bound $\rho \geq \frac{1}{2} + \frac{1}{(n+1)!}$ is obtained to solve MIHNP according to the Coppersmith technique, where n denotes the number of unknown variables. Similar to the above analysis, we can also get that time complexities of the LLL algorithm and Gröbner basis computation are $\mathcal{O}\big(k^{\mathcal{O}(1)} n^{\mathcal{O}(n)}\big)$ and $n^{\mathcal{O}(n^2)}$ respectively. Further, from the above bound $\rho \geq \frac{1}{2} + \frac{1}{(n+1)!}$, we can take $n \log_2 n \approx \log_2(\frac{1}{\rho - \frac{1}{2}})$ by the Stirling formula. Therefore, time complexities of the LLL algorithm and the Gröbner basis computation are reduced to $\mathcal{O}\big(k^{\mathcal{O}(1)} \big(\frac{1}{\rho - \frac{1}{2}}\big)^{\mathcal{O}(1)}\big)$ and $\big(\frac{1}{\rho - \frac{1}{2}}\big)^{o\big(\log_2 \frac{1}{\rho - \frac{1}{2}}\big)}$ respectively.

In [4, Section 3.2], the asymptotic bound $\rho \geq \frac{1}{3} + \frac{2}{3d+3}$ is obtained to solve MIHNP based on the SVP assumption, where d is an integer satisfying some requirement. Note that the dimension of the involved lattice is equal to $\mathcal{O}(d^{\mathcal{O}(d)})$. Thus, the time complexity to solve MIHNP is $k^{\mathcal{O}(1)} 2^{\mathcal{O}(d^{\mathcal{O}(d)})}$ using the SVP algorithm, such as [22]. According to the above bound $\rho \geq \frac{1}{3} + \frac{2}{3d+3}$, we can take $d \approx \frac{2}{3\rho - 1}$. Then the above time complexity is reduced to $k^{\mathcal{O}(1)} 2^{\mathcal{O}\big((\frac{2}{3\rho-1})^{\mathcal{O}(\frac{1}{\rho - \frac{1}{3}})}\big)}$.

In [35, Remark 4], the asymptotic bound $\rho \geq \frac{1}{3} + \frac{2}{3d+3}$ is given for solving MIHNP and ICG based on the Coppersmith technique, where d is the same as that in [4]. Note that the dimension of the involved lattice is equal to $\mathcal{O}(d^{\mathcal{O}(d)})$ and the maximal bit size of lattice basis matrix is at most $2dk$. Hence, the time complexity of the LLL algorithm is $(\mathcal{O}(d^{\mathcal{O}(d)}))^{\mathcal{O}(1)} \cdot (2dk)^{\mathcal{O}(1)} = \mathcal{O}(k^{\mathcal{O}(1)} d^{\mathcal{O}(d)})$. For the Gröbner basis, the maximal degree of input polynomials is $2d$ and the number of variables is equal to $d^{\mathcal{O}(1)}$. Thus, under Assumption 1, the time complexity of the Gröbner basis computation is $(2d)^{\mathcal{O}(d^{\mathcal{O}(1)})}$ [10]. Based on the above bound $\rho \geq \frac{1}{3} + \frac{2}{3d+3}$, we can take $d \approx \frac{2}{3\rho - 1}$. Then, time complexities of the LLL algorithm and Gröbner basis computation are reduced to $\mathcal{O}(k^{\mathcal{O}(1)} (\frac{2}{3\rho-1})^{\mathcal{O}(\frac{1}{\rho - \frac{1}{3}})})$ and $(\frac{4}{3\rho-1})^{\mathcal{O}((\frac{1}{\rho - \frac{1}{3}})^{\mathcal{O}(1)})}$ respectively.

B Computation of the Determinant of $\mathcal{L}(n, d, t)$

Note that the determinant of $\mathcal{L}(n, d, t)$ is the product of the diagonal entries. We will consider the following two cases.

For the case of $i_0 \geq s$, the contribution of $F_{i_0,i_1,\cdots,i_n}(x_0X, x_1X, \cdots, x_nX)$ to the determinant of $\mathcal{L}(n, d, t)$ is

$$\prod_{s=0}^{d} \prod_{i_0=s}^{d} \left(p^{(d-s)\binom{n}{s}} \cdot X^{(i_0+s)\binom{n}{s}} \right).$$

For the case of $i_0 < s$, the contribution of $F_{i_0,i_1,\cdots,i_n}(x_0X, x_1X, \cdots, x_nX)$ is

$$\prod_{s=1}^{d} \prod_{i_0=0}^{s-1} \left(p^{(d+1-s)\binom{n}{s}} \cdot X^{(i_0+s)\binom{n}{s}} \right) \cdot \prod_{i_0=0}^{t} X^{(i_0+d+1)\binom{n}{d+1}}.$$

To sum up, we get

$$\det(\mathcal{L}(n, d, t)) = p^{\alpha(n,d)} \cdot X^{\beta(n,d,t)},$$

where

$$\begin{aligned}
\alpha(n, d) &= d(d+1) \sum_{s=0}^{d} \tbinom{n}{s} - d \sum_{s=0}^{d} s\tbinom{n}{s}, \\
\beta(n, d, t) &= \tfrac{d(d+1)}{2} \sum_{s=0}^{d} \tbinom{n}{s} + (d+1) \sum_{s=0}^{d} s\tbinom{n}{s} + \tfrac{(2d+t+2)(t+1)}{2} \tbinom{n}{d+1}.
\end{aligned}$$

C Lower Bound in Theorem 1

Our goal is to derive a lower bound of

$$2^{-\frac{w(w-1)}{4\beta(n,d,t)}} w^{-\frac{w-n}{2\beta(n,d,t)}} p^{\lambda(n,d,t)},$$

where w is the dimension of $\mathcal{L}(n, d, t)$. We now analyze its first two terms. According to the expressions of w and $\beta(n, d, t)$, i.e.,

$$\begin{aligned}
w &= (t+1)\tbinom{n}{d+1} + (d+1) \sum_{s=0}^{d} \tbinom{n}{s}, \\
\beta(n, d, t) &= \tfrac{(2d+t+2)(t+1)}{2} \tbinom{n}{d+1} + \tfrac{d(d+1)}{2} \sum_{s=0}^{d} \tbinom{n}{s} + (d+1) \sum_{s=0}^{d} s\tbinom{n}{s},
\end{aligned}$$

it is easy to deduce $\frac{\beta(n,d,t)}{w} > \frac{d+2}{2}$. Then we have $2^{-\frac{w(w-1)}{4\beta(n,d,t)}} \geq 2^{-\frac{w}{2(d+2)}}$ and $w^{-\frac{w-n}{2\beta(n,d,t)}} \geq w^{-\frac{1}{d+2}}$. Furthermore, we obtain

$$2^{-\frac{w(w-1)}{4\beta(n,d,t)}} w^{-\frac{w-n}{2\beta(n,d,t)}} p^{\lambda(n,d,t)} \geq p^{\lambda(n,d,t) - \frac{w+2\log w}{2(d+2)\log_2 p}}.$$

Note that d and w are independent of the modulus p. For a sufficiently large p, the exponent term $-\frac{w+2\log w}{2(d+2)\log_2 p}$ is negligible. In this case, we only consider the exponent term $\lambda(n,d,t)$. In other words, the right-hand side of the above condition can be simplified as $p^{\lambda(n,d,t)}$ for a sufficiently large p.

Next, we further analyze the lower bound of $\lambda(n,d,t)$. We rewrite

$$\begin{aligned}\lambda(n,d,t) &= \frac{2d(t+1)\binom{n}{d+1}+2d\sum_{s=2}^{d} s\binom{n}{s}}{(2d+2+t)(t+1)\binom{n}{d+1}+d(d+1)\sum_{s=0}^{d}\binom{n}{s}+2(d+1)\sum_{s=0}^{d} s\binom{n}{s}} \\ &= \frac{2d}{2d+2+t}(1-\epsilon(n,d,t)),\end{aligned}$$

where

$$\epsilon(n,d,t) = \frac{d(d+1)\sum_{s=0}^{d}\binom{n}{s} - t\sum_{s=2}^{d} s\binom{n}{s} + 2(d+1)\binom{n}{1}}{(2d+2+t)(t+1)\binom{n}{d+1}+d(d+1)\sum_{s=0}^{d}\binom{n}{s}+2(d+1)\sum_{s=0}^{d} s\binom{n}{s}}.$$

Note that we have

$$\epsilon(n,d,t) < \frac{d(d+1)}{(2d+2+t)(t+1)}\cdot\frac{\sum_{s=0}^{d}\binom{n}{s}}{\binom{n}{d+1}} + \frac{2(d+1)}{(2d+2+t)(t+1)}\cdot\frac{\binom{n}{1}}{\binom{n}{d+1}} < \frac{d}{2}\sum_{s=0}^{d}\frac{\binom{n}{s}}{\binom{n}{d+1}} + \frac{\binom{n}{1}}{\binom{n}{d+1}}.$$

For any $0 \le s \le d$, according to

$$\frac{\binom{n}{s}}{\binom{n}{d+1}} = \frac{(d+1)!(n-d-1)!}{s!(n-s)!} = \frac{d+1}{n-d}\cdot\frac{d}{n-d+1}\cdots\frac{s+1}{n-s} \le (\frac{d+1}{n-d})^{d-s+1},$$

we deduce that

$$\epsilon(n,d,t) < \left(\frac{d}{2}\sum_{s=0}^{d}(\frac{d+1}{n-d})^{d-s+1}\right) + (\frac{d+1}{n-d})^{d} = \frac{d(d+1)}{2(n-2d-1)}\big(1-(\frac{d+1}{n-d})^{d+1}\big) + (\frac{d+1}{n-d})^{d}.$$

Then we obtain that

$$\lambda(n,d,t) = \frac{2d}{2d+2+t}(1-\epsilon(n,d,t)) > \frac{2d}{2d+2+t}\left(1-\frac{d(d+1)}{2(n-2d-1)}\big(1-(\frac{d+1}{n-d})^{d+1}\big)-(\frac{d+1}{n-d})^{d}\right).$$

By taking the parameter $t=0$, $\lambda(n,d,t)$ is optimized as

$$\lambda(n,d,0) > 1-\frac{1}{d+1}-\left(\frac{d^2}{2(n-2d-1)}\big(1-(\frac{d+1}{n-d})^{d+1}\big)+\frac{d}{d+1}(\frac{d+1}{n-d})^{d}\right).$$

Further, by taking the parameter $n = d^{3+o(1)}$, the above relation is expressed as

$$\lambda(n,d,0) > 1-\frac{1}{d+1}-o(\frac{1}{d}).$$

Finally, we explicitly present how big the modulus p is in the asymptotic sense. Based on the above analysis, we need that the term $-\frac{w+2\log w}{2(d+2)\log_2 p}$ is negligible. For the case of $t=0$ and $n=d^{3+o(1)}$, we have that the dimension of $L(n,d,t)$ is equal to $w=(d+1)\sum_{s=0}^{d}\binom{n}{s}+\binom{n}{d+1}=d^{3d+3}(1+o(1))$. Hence, when $\log_2 p=\omega(d^{3d+2})$, i.e., $p=2^{\omega(d^{3d+2})}$, the term $-\frac{w+2\log w}{2(d+2)\log_2 p}$ is negligible.