Designs, Codes and Cryptography manuscript No.
(will be inserted by the editor)

Solving a Class of Modular Polynomial Equations and its Relation to Modular Inversion Hidden Number Problem and Inversive Congruential Generator

Jun Xu · Santanu Sarkar · Lei Hu · Zhangjie Huang · Liqiang Peng

Received: date / Accepted: date

Abstract In this paper we revisit the modular inversion hidden number problem (MIHNP) and the inversive congruential generator (ICG) and consider how to attack them more efficiently. We consider systems of modular polynomial equations of the form $a_{ij} + b_{ij}x_i + c_{ij}x_j + x_ix_j = 0 \pmod{p}$ and show the relation between solving such equations and attacking MIHNP and ICG. We present three heuristic strategies using Coppersmith's lattice-based root-finding technique for solving the above modular equations.

In the first strategy, we use the polynomial number of samples and get the same asymptotic bound on attacking ICG proposed in PKC 2012, which is the best result so far. However, exponential number of samples is required in the work of PKC 2012. In the second strategy, a part of polynomials chosen for the involved lattice are linear combinations of some polynomials and this enables us to achieve a larger upper bound for the desired root. Corresponding to the analysis of MIHNP we give an explicit lattice construction of the second attack method proposed by Boneh, Halevi and Howgrave-Graham in Asiacrypt 2001. We provide better bound than that in the work of PKC 2012 for attacking ICG. Moreover, we propose the third strategy in order to give a further improvement in the involved lattice construction in the sense of requiring fewer samples.

Keywords Modular inversion hidden number problem · inversive congruential generator · lattice · LLL algorithm · Coppersmith's technique

Mathematics Subject Classification (2000) 94A60

J. Xu · L. Hu · Z. Huang · L. Peng
State Key State Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing 100093, China
E-mail: {xujun,hulei,huangzhangjie,pengliqiang}@iie.ac.cn

S. Sarkar (✉)
Department of Mathematics, Indian Institute of Technology Madras, Chennai 600036, India.
E-mail: sarkar.santanu.bir@gmail.com

1 Introduction

1.1 Background

Modular Inversion Hidden Number Problem. Hidden Number Problem (HNP) was introduced in [6] by Boneh and Venkatesan. They used it to prove that computing the most significant bits (MSBs) of the secret key from the public keys of participants in a Diffie-Hellman key-exchange protocol is as hard as computing the secret key itself. The work [6] has originated a whole direction of research and HNP has been exploited in a wide spectrum of applications like an attack on weak versions of the Digital Signature Algorithm (DSA) [13] and appeared in a number of questions, related and unrelated to cryptography (see [22] for a survey of relevant results and also [1] for some developments in this direction).

A closely related class to HNP, known as Modular Inversion Hidden Number Problem (MIHNP), was introduced and studied in [5] by Boneh, Halevi and Howgrave-Graham. They utilized MIHNP to construct pseudo random number generator (PRNG) and message authentication code (MAC). The exact problem (MIHNP) is as follows.

For a given prime p, consider a secret $\alpha \in \mathbb{Z}_p$ and $n+1$ elements $t_0, t_1, \ldots, t_n \in \mathbb{Z}_p \backslash \{-\alpha\}$, chosen independently and uniformly at random. The question is, given $n+1$ samples $\left\{ \left(t_i, \mathrm{MSB}_\delta((\alpha + t_i)^{-1} \bmod p)\right) \right\}_{i=0}^{n}$ for some $\delta > 0$ (here $\mathrm{MSB}_\delta(z)$ refers to the δ most significant bits of z), whether it is possible to recover the hidden number α.

Inversive Congruential Generator. Number-theoretic PRNGs work by iterating an algebraic map f over a residue ring $\mathbb{Z}_N$ on a secret random initial seed value v_0 to compute values $v_{i+1} = f(v_i) \bmod N$. The output is some consecutive bits of the state value v_i at each iteration. The first input v_0 is called the seed. PRNGs have numerous applications in signature schemes and public key encryption schemes. When f is affine, generator is known as linear congruential generator. However this generator is not cryptographically secure [7, 23]. It was suggested to use a non-linear algebraic map f in order to avoid these attacks.

The Inversive Congruential Generator (ICG) proposed by Eichenauer and Lehn [10] is an important kind of nonlinear number-theoretic pseudo random number generator. There are extensive applications of ICG in Quasi-Monto Carlo simulation and public key schemes (see the surveys in [11, 16, 17, 19, 24] and recent results in [18, 20, 21, 25]). ICG works as follows:

For a given prime p, let $f(x) = ax^{-1} + b \pmod p$ where $a, b \in \mathbb{Z}_p$. Input a secret seed v_0 to the recursive relation $v_{i+1} = f(v_i)$ for $0 \le i \le n$. Then output a random-looking sequence $(\mathrm{MSB}_\delta(v_1), \mathrm{MSB}_\delta(v_2), \cdots, \mathrm{MSB}_\delta(v_{n+1}))$.

A very strong goal of attacking ICG is to recover the secret seed v_0 given $n+1$ outputs $\mathrm{MSB}_\delta(v_{i+1})$ for $i = 0, 1 \cdots, n$.

1.2 Previous Works

Analysis of MIHNP. Boneh et al. [5] presented two polynomial time heuristic lattice methods to solve MIHNP, provided that the number of samples is sufficiently large. Their first heuristic works only if more than $\frac{2}{3}$ portion of most significant bits of $(\alpha + t_i)^{-1} \bmod p$'s are given, i.e., $\delta > \frac{2}{3}\log_2 p$. Importantly, the second

heuristic (where multiples are used [5, Section 3.2] that we refer here as Method II) claimed knowledge of significantly fewer bits which is $\delta > \frac{1}{3}\log_2 p$ only. However, no explicit lattice construction for this case was presented.

Ling et al. [15] provided a rigorous probabilistic polynomial time algorithm for MIHNP. The work in [15] could match one of the heuristics (Method I) of Boneh et al. [5], where one requires two-third of the bits of the output to solve the problem. However, Ling et al. could not theoretically justify the more efficient heuristic (Method II) by Boneh et al., that requires only one-third of the bits of the output.

Based on the observation that the algorithm in [15] is not ideal when the number of samples is relatively small, Xu et al. [26] proposed a heuristic lattice method by combining Coppersmith's lattice technique and the priority queue technique. The corresponding result is $\delta > \frac{1}{2}\log_2 p$, which is better than that of Method I of Boneh et al in [5] and the rigorous algorithm of Ling et al. in [15], but weaker than Method II of Boneh et al.

Attack of ICG. In [3,4], Blackburn et al. pointed out that ICG can be attacked in polynomial time if sufficiently many bits of some consecutive values v_i are revealed by combining lattice method and linearization technique. In PKC 2012, Bauer et al. [2] improved the work of Blackburn et al. by utilizing Coppersmith's method. They showed that the secret seed of ICG can be recovered if more than $\frac{1}{2}$ portion of most significant bits of v_i's are given, i.e., $\delta > \frac{1}{2}\log_2 p$, provided that the number of samples is sufficiently large. In this paper, we improve the bound of δ. In Table 1, we compare new bound of δ with the existing bounds for ICG.

Table 1 Comparison of asymptotic lower bound of δ for ICG with existing methods.

	Lower bound of $\boldsymbol{\delta/\log_2 p}$
Blackburn et al. [4]	**2/3**
Bauer et al. [2]	**1/2**
Our bound	**1/3**

1.3 Our Contribution

We first translate the recovering problem of the hidden number in MIHNP and the secret seed of ICG into solving multivariate modular polynomial equations

$$a_{ij} + b_{ij}x_i + c_{ij}x_j + x_i x_j = 0 \pmod p, 0 \le i < j \le n$$

and then, we give three heuristic lattice methods to find the small solutions of the above modular equations.

In the first strategy, for given $n+1$ samples in MIHNP or $n+1$ outputs in ICG, the hidden number or the secret seed can be recovered when the number of known MSBs

$$\delta > \left(\frac{1}{2} + \frac{1}{2n+2}\right) \cdot \log_2 p.$$

The asymptotic bound when $n \to +\infty$ is better than the first result of Boneh et al. in [5] and the rigorous result of Ling et al. in [15], and same as the asymptotic results in [2, 26]. However, the dimension of the lattices in our first strategy is polynomial on n but the dimension of the involved lattices in [2, 26] is exponential on n.

In the second strategy, a part of polynomials used for the lattices are linear combinations of several polynomials, which obtains a larger upper bound for the desired root. Given $n+1$ samples in MIHNP or $n+1$ outputs in ICG, the hidden number or the secret seed can be obtained when

$$\delta > (1 - F(n, d)) \cdot \log_2 p$$

where $F(n, d)$ is defined in Section 5 and parameter d satisfies $1 \leq d \leq n$. For integers n and d, there is always $\frac{\delta}{\log_2 p} > \frac{1}{3}$. Taking $d = c \cdot n$ for $0 < c \leq \frac{1}{2}$, one can get

$$\frac{\delta}{\log_2 p} \to \frac{1}{3} \text{ when } n \to +\infty.$$

This asymptotic bound is superior than the best work on the attack of ICG [26] and is same as the result of MIHNP in Method II [5]. However, here we explicitly present lattice construction not given in Method II [5].

In the third strategy, we improve the lattice construction in the second strategy. Under the situation that $n+1$ samples in MIHNP or $n+1$ outputs in ICG are given, one can get the hidden number or the secret seed if

$$\delta > (1 - F(n, d, k)) \cdot \log_2 p$$

where $F(n, d, k)$ is described in Section 6 and $1 \leq d \leq n$ and $k \geq 1$. For integers n, d, k, there is also always $\frac{\delta}{\log_2 p} > \frac{1}{3}$. When $k = 1$, it becomes the second strategy. When $k > 1$, it is better than the second strategy as $\frac{\delta}{\log_2 p}$ is closer to $\frac{1}{3}$ in this strategy. Moreover, $\frac{\delta}{\log_2 p} < \frac{1}{2}$ for $n = 4$, which is superior to the asymptotic results in the case that sufficiently many samples are given in [2, 26].

1.4 Organization of the Paper

The rest of this paper is organized as follows. In Section 2, we recall some terminologies and preliminary knowledge. In Section 3, we transform analysis of MIHNP and attack of ICG into solving a class of modular polynomial equations. In Sections 4, 5 and 6, we respectively present three strategies for solving a class of modular polynomial equations and give the corresponding applications for MIHNP and ICG. Section 7 is the conclusion.

2 Preliminaries

Throughout the paper, the set $\{0, 1, \cdots, p^s - 1\}$ is denoted as $\mathbb{Z}_{p^s}$ where s is some positive integer, in case of need, the elements of $\mathbb{Z}_{p^s}$ are also treated as the corresponding integers.

2.1 Order of Monomials

First, we describe reverse lexicographic order and graded lexicographic reverse order respectively. For more details about the orders of monomials, please refer to [9]. Let integer vectors $I_n = (i_1, \cdots, i_n)$, $J_n = (j_1, \cdots, j_n)$.

Reverse Lexicographic Order:

$$I_n \prec_{revlex} J_n \Leftrightarrow \text{ the rightmost nonzero entry in } I_n - J_n \text{ is negative.}$$

For example,

$$(0,0) \prec_{revlex} (3,0) \prec_{revlex} (1,1) \prec_{revlex} (0,2).$$

Graded Reverse Lexicographic Order:

$$I_n \prec_{grevlex} J_n \Leftrightarrow \sum_{m=1}^{n} i_m < \sum_{m=1}^{n} j_m \text{ or } (\sum_{m=1}^{n} i_m = \sum_{m=1}^{n} j_m \text{ and } I_n \prec_{revlex} J_n).$$

For vectors in the above example, we have

$$(0,0) \prec_{grevlex} (1,1) \prec_{grevlex} (0,2) \prec_{grevlex} (3,0).$$

Next, we define an order of monomials which will be used to arrange the polynomials according to their leading monomials in the following lattice construction.

A New Defined Order:

$$x_0^{i_0} x_1^{i_1} \cdots x_n^{i_n} \prec x_0^{j_0} x_1^{j_1} \cdots x_n^{j_n} \Leftrightarrow I_n \prec_{grelex} J_n \text{ or } (I_n = J_n \text{ and } i_0 < j_0). \quad (1)$$

For example, $f(x_0, x_1) = a + bx_0 + cx_1 + x_0x_1$. According to (1), there is $1 \prec x_0 \prec x_1 \prec x_0x_1$. Thus, x_0x_1 is the leading monomial of $f(x_0, x_1)$.

2.2 Polynomial Coefficients

For positive integers k and n, the coefficient of x^s in the expansion of the polynomial $(1 + x + \cdots + x^k)^n$ is called the polynomial coefficient $\binom{n}{s}_{k+1}$, $0 \le s \le nk$. Namely, we have

$$(1 + x + \cdots + x^k)^n = \sum_{s=0}^{nk} \binom{n}{s}_{k+1} x^s,$$

where $\binom{n}{s}_{k+1} = \sum_{n_1 + \cdots + kn_k = s} \binom{n}{n - n_1 - \cdots - n_k, n_1, \cdots, n_k}$. Obviously, when $k = 1$, the polynomial coefficient $\binom{n}{s}_{k+1}$ is the binomial coefficient $\binom{n}{s}$. When $m(k+1) \le s \le (m+1)(k+1) - 1$, we have

$$\binom{n}{s}_{k+1} = \sum_{i=0}^{m} (-1)^i \binom{n + s - i(k+1) - 1}{s - i(k+1)} \binom{n}{i},$$

where $m \in \mathbb{N}$. Euler first studied this expansion. For more details on polynomial coefficients, please refer to [8].

2.3 Lattice

Let the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_\omega$ be linearly independent in $\mathbb{R}^n$, the set

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

is called a lattice with basis vectors $\mathbf{b}_1, \cdots, \mathbf{b}_\omega$. The dimension and determinant of L are respectively

$$\dim(L) = \omega, \det(L) = \sqrt{\det(BB^T)}.$$

Here, $B = [\mathbf{b}_1^T, \cdots, \mathbf{b}_\omega^T]^T$ is a basis matrix. If B is a square matrix, then $\det(L) = |\det(B)|$. In this paper all lattice basis matrices are square.

It is well known that the LLL algorithm [14] can find a reduced basis of the lattice as follows.

Lemma 1 ([14]) *Let L be a lattice. Within polynomial time, the LLL algorithm outputs reduced basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\omega$ that satisfy*

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

2.4 Coppersmith's Technique

The Coppersmith technique can be used for finding the small solution of modular polynomials. Its key step is to generate more modular polynomials with common root which is the desired solution and then use the lattice reduction algorithms to obtain integer polynomials with the desired root. In this step, the following Lemma reformulated by Howgrave-Graham is needed.

Lemma 2 ([12]) *Let $f(x_0, x_1, \ldots, x_n)$ be an integer polynomial that consists of at most ω monomials. Let d be a positive integer and the X_i be the upper bound of $|x_i|$ for $i = 0, 1, \cdots, n$. Suppose that*

1. $f(x_0, x_1, \ldots, x_n) = 0 \pmod{p^d}$,
2. $\|f(x_0X_0, x_1X_1, \ldots, x_nX_n)\| < \frac{p^d}{\sqrt{\omega}}$,

then $f(x_0, x_1, \ldots, x_n) = 0$ holds over $\mathbb{Z}$.

Now see that $\|f(x_0X_0, x_1X_1, \ldots, x_nX_n)\|$ in Lemma 2 is the Euclidean norm of the coefficient vector of the polynomial $f(x_0X_0, x_1X_1, \ldots, x_nX_n)$. This is also the norm of the corresponding row vector of the involved lattice. To obtain at least $n+1$ polynomials with the common desired root $(x_0, x_1, \ldots, x_n)$, from Lemma 1 and Lemma 2, we need

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

Since the terms $2^{\frac{\omega(\omega-1)}{4(\omega-n)}}$ and $\sqrt{\omega}$ are much smaller than p, we give the following simplified condition

$$\det(L) < p^{d \cdot (\omega - n - o(1))},$$

where $\omega = \dim(L)$. Further, we expect that the obtained integer polynomials are algebraically independent. Then, we can utilize numerical or symbolic methods such as the resultant method or the Gröbner basis technique to compute the desired root $(x_0, x_1, \ldots, x_n)$. In this process, the following assumption is used.

Assumption 1. *Let $g_1, \cdots, g_{n+1} \in \mathbb{Z}[x_0, x_1, \cdots, x_n]$ be the polynomials that are found by Coppersmith's technique. Then the ideal generated by the polynomial equations $g_1(x_0, x_1, \cdots, x_n) = 0, \cdots, g_{n+1}(x_0, x_1, \cdots, x_n) = 0$ has dimension zero.*

We consider the above assumption to be true in all the results that will be presented in this paper. In all the experiments, we observe the correctness of the assumption. So we collect the roots efficiently using Gröbner basis technique.

We implement programs in SAGE 5.13 on a Linux Mint 12 on a laptop with Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz, 3 GB RAM and 3 MB Cache.

3 A Class of Modular Polynomial Equations

In this section, we obtain a class of modular polynomial equations with small roots from the analysis of HIMNP and the attack on ICG respectively.

3.1 Analysis of MIHNP

First, we propose the following problem on the analysis of MIHNP.

Problem 1 For a sufficiently large 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

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

for some $k > 0$.

Next, we transform Problem 1 into solving simultaneous modular equations with small roots. Since the $\mathrm{MSB}_\delta((\alpha + t_i)^{-1} \bmod p))$ (i.e., δ many MSBs of the integers $(\alpha + t_i)^{-1} \bmod p$) are known, we let $u_i = \mathrm{MSB}_\delta((\alpha + t_i)^{-1} \bmod p))$ and can write $u_i + x_i = (\alpha + t_i)^{-1} \bmod p$ where u_i is known and x_i is unknown for $i = 0, 1, \cdots, n$. Rearranging the above relations, we get

$$(\alpha + t_i)(u_i + x_i) = 1 \pmod{p}.$$

Further, eliminating α from these equations, we can obtain $\binom{n+1}{2}$ modular equations as follows:

$$a_{ij} + b_{ij}x_i + c_{ij}x_j + x_i x_j = 0 \pmod{p}, 0 \le i < j \le n, \tag{3}$$

where

$$\begin{aligned} a_{ij} &= u_i u_j + (u_i - u_j)(t_i - t_j)^{-1} \bmod p, \\ b_{ij} &= u_j + (t_i - t_j)^{-1} \bmod p, \\ c_{ij} &= u_i - (t_i - t_j)^{-1} \bmod p. \end{aligned}$$

Since δ many MSBs of $(\alpha + t_i)^{-1} \bmod p$ are known, we have $0 \le x_i \le \frac{p}{2^\delta}$ for $0 \le i \le n$. It is easy to see that we can recover the hidden number α once x_i's are obtained.

3.2 Recovering Seed Attack on ICG

First, let us recall ICG. For a given prime p, the generator function of ICG is $f(x) = ax^{-1} + b \pmod{p}$ where $a, b \in \mathbb{Z}_p$. Input a secret seed v_0 to the recursive relation $v_{i+1} = f(v_i)$ for $0 \leq i \leq n$. Then outputs are $\mathrm{MSB}_\delta(v_{i+1})$ (i.e., δ many MSBs of v_{i+1}) for some $\delta > 0$. Naturally, one concerns with the attack on recovering the secret seed. In other words, one will consider the following concrete problem.

Problem 2 For a sufficiently large prime p, the goal is to recover the seed v_0 given $n+1$ ICG outputs

$$\left\{\mathrm{MSB}_\delta(v_{i+1})\right\}_{i=0}^{n}.$$

Then, let us translate Problem 2 into finding small roots of modular equations. According to the relations $v_{i+1} = av_i^{-1} + b \pmod{p}$ for $0 \leq i \leq n$, we have

$$a + bv_i - v_i v_{i+1} = 0 \bmod p, i = 0, 1, \cdots, n.$$

From these relations, we can use the resultant method and generate relations

$$a'_{ij} + b'_{ij} v_i + c'_{ij} v_j + d'_{ij} v_i v_j = 0 \bmod p \text{ for } 0 \leq i < j \leq n+1, \tag{4}$$

where $a'_{ij}, b'_{ij}, c'_{ij}, d'_{ij}$ are known. For example we have $a + bv_1 - v_1 v_2 = 0 \bmod p$ and $a + bv_2 - v_2 v_3 = 0 \bmod p$. Now we utilize the resultant method to eliminate the variable v_2 and get

$$ab + (a + b^2)v_1 - av_3 - bv_1 v_3 = 0 \bmod p.$$

Next, let $u_{i+1} = \mathrm{MSB}_\delta(v_{i+1})$, we can write

$$v_{i+1} = u_{i+1} + x_i \text{ for } i = 0, 1, \cdots, n \tag{5}$$

where u_{i+1} is known and x_i is unknown. Now plugging (5) into the relations in (4), we get the following relations

$$a_{ij} + b_{ij} x_i + c_{ij} x_j + x_i x_j = 0 \bmod p \text{ for } 0 \leq i < j \leq n \tag{6}$$

where the integers a_{ij}, b_{ij}, c_{ij} are known. Since δ many MSBs of v_i are known, we get $0 \leq x_i \leq \frac{p}{2^\delta}$ for $0 \leq i \leq n$. Obviously, if x_i's are found out, we can recover the seed v_0 from (4) and (5).

Note that the forms of equations in (3) and (6) are the same. Therefore, in order to solve Problems 1 and 2, our goal in the following sequel is to find the small roots (x_i, x_j) of the following modular polynomial equations

$$f_{ij}(x_i, x_j) := a_{ij} + b_{ij} x_i + c_{ij} x_j + x_i x_j = 0 \pmod{p}, 0 \leq i < j \leq n \tag{7}$$

where $x_0, x_1, \cdots, x_n$ are bounded by X. In the cases of MIHNP and ICG, we take $X = \frac{p}{2^\delta}$ where δ is the number of known MSBs.

For our second and third strategies on solving (7) in the sequent sections, we need to utilize the following assumption.

Assumption 2. *Assume that the $c_{0,j}$ in equation (7) are independent and uniformly random in $\mathbb{Z}_p$ for all $j = 1, \cdots, n$ and positive number $\frac{n^2}{p}$ is negligible.*

Based on Assumption 2, we can deduce that the $c_{0,j}$ are distinct in $\mathbb{Z}_p$ for all $j = 1, \cdots, n$ with overwhelming probability. Concretely speaking, from the $c_{0,j}$ are independent and uniformly random in $\mathbb{Z}_p$ for all $j = 1, \cdots, n$, we have

$$\Pr\{\textit{all } c_{0,j} \textit{ are distinct mod } p \textit{ for } j = 1, \cdots, n\} = \prod_{i=1}^{n-1}(1 - \frac{i}{p}).$$

Note that $\frac{n^2}{p}$ is negligible, we get

$$\prod_{i=1}^{n-1}(1 - \frac{i}{p}) \approx e^{-\sum\limits_{i=1}^{n-1}\frac{i}{p}} = e^{-\frac{n(n-1)}{2p}} \approx 1 - \frac{n^2 - n}{2p},$$

which is close to 1.

4 The First Strategy on Solving (7)

In this section, we propose the first strategy for solving (7) and give the corresponding application on MIHNP and ICG. First, we present the following heuristic result, which is verified by our experiments.

Result 1. *For given polynomials $f_{ij}(x_i, x_j)$ with $0 \le i < j \le n$ in (7), under Assumption 1, one can solve (7) in time polynomial in $(n, \log p)$ when the bound X of $x_0, x_1, \cdots, x_n$ satisfies*

$$X < p^{\frac{1}{2} - \frac{1}{2n+2} - \epsilon_1}$$

where ϵ_1 is some positive real number.

Proof. Consider the following set of polynomials

$$\mathcal{P} = \left\{p, px_0, px_1, \cdots, px_n, f_{0,1}, \cdots, f_{0,n}, \cdots, f_{n-1,n}\right\}.$$

Now construct a lattice L using the coefficient vectors of $h(x_0X, x_1X, \ldots, x_nX)$ for each $h \in \mathcal{P}$.

Then the matrix M, corresponding to L, is of the form

$$\begin{array}{c}
\begin{array}{ccccccccc} x_0x_1 & \ldots & x_0x_n & \ldots & x_{n-1}x_n & x_0 & \ldots & x_n & 1 \end{array} \\
\begin{pmatrix}
X^2 & \ldots & 0 & \ldots & 0 & - & \ldots & 0 & - \\
\vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \ldots & X^2 & \ldots & 0 & - & \ldots & - & - \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \ldots & 0 & \ldots & X^2 & 0 & \ldots & - & - \\
0 & \ldots & 0 & \ldots & 0 & Xp & \ldots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \ldots & 0 & \ldots & 0 & 0 & \ldots & Xp & 0 \\
0 & \ldots & 0 & \ldots & 0 & 0 & \ldots & 0 & p
\end{pmatrix}
\end{array}$$

Here '−' indicates non zero element.

Clearly, the dimension of L is $\dim(L) = \binom{n+1}{2} + n + 2$. Also

$$\det(L) = p^{n+2} \cdot \overbrace{X \cdots X}^{n+1} \cdot \overbrace{X^2 \cdots X^2}^{\binom{n+1}{2}} = p^{n+2} X^{(n+1)^2}.$$

By the property of LLL algorithm and Howgrave-Graham's lemma, if the condition (2) satisfies (here $d = 1$), i.e., there is

$$\left(\omega^{\frac{\omega-n}{2}} 2^{\frac{\omega(\omega-1)}{4}} p^n\right) \cdot \det(L) < p^{\omega} \tag{8}$$

where $\omega = \det(L)$, after reduction of lattice we get $n+1$ polynomials $g_1, \cdots, g_{n+1}$ which contain the root $(x_0, x_1, \cdots, x_n)$ over integers. Under Assumption 1, we can find $x_0, x_1, \cdots, x_n$ from $g_1, \cdots, g_{n+1}$.

Finally, we analyze the situation that the condition (8) holds. Plugging the determinant and dimension of the lattice L into (8), we get the condition

$$X < \left(\omega^{\frac{\omega-n}{2}} 2^{\frac{\omega(\omega-1)}{4}} p^n\right)^{-\frac{1}{(n+1)^2}} \cdot p^{\frac{1}{2} - \frac{1}{2n+2}}.$$

Note that $\omega = \binom{n+1}{2} + n + 2 < (n+1)^2$ for $n > 2$, we can deduce that

$$\left(\omega^{\frac{\omega-n}{2}} 2^{\frac{\omega(\omega-1)}{4}} p^n\right)^{\frac{1}{(n+1)^2}} < \omega^{\frac{1}{2}} 2^{\frac{\omega-1}{4}} p^{\frac{n}{(n+1)^2}} = p^{\epsilon_1}$$

where $\epsilon_1 = \frac{2\log_2 \omega + (\omega-1)}{4 \log_2 p} + \frac{n}{(n+1)^2} > 0$. It implies that the right side of the above condition on X can be lower bounded by $p^{\frac{1}{2} - \frac{1}{2n+2} - \epsilon_1}$. Hence, in order to let (8) be satisfy, we need

$$X < p^{\frac{1}{2} - \frac{1}{2n+2} - \epsilon_1}.$$

When $\log_2 p \gg \dim(L)$ and n is large enough, ϵ_1 is negligible. □

Note that $X = \frac{p}{2^{\delta}}$ for the situations of MIHNP and ICG. Hence, we have the following application.

Application 1. *Given $n+1$ samples in MIHNP or $n+1$ outputs in ICG, under Assumption 1, one can recover the hidden number or the secret seed in time polynomial in $(n, \log_2 p)$ if the number δ of known MSBs satisfies*

$$\frac{\delta}{\log_2 p} > \frac{1}{2} + \frac{1}{2n+2} + \epsilon_1.$$

Remark 1 It is shown that one can obtain the hidden number in MIHNP if $\frac{\delta}{\log_2 p} > \frac{2}{3}$ for $n \to \infty$ by using the first heuristic method in [5] or the rigorous technique in [15] respectively. In this section, we need $\frac{\delta}{\log_2 p} > \frac{1}{2}$ for $n \to \infty$. Hence, we get the better heuristic bound.

Remark 2 It is proved that one can heuristically recover the hidden number in MIHNP [26] or the secret seed in ICG [2] if $\frac{\delta}{\log_2 p} > \frac{1}{2}$ for $n \to \infty$ using the idea of Coppersmith. However, the involved lattices dimension in [2, 26] becomes exponential in n. We achieve the asymptotic bound of [2, 26] when the lattice dimension is polynomial in n.

Remark 3 The asymptotic result of the first strategy is better than those of the Method I in [5] or the rigorous technique in [15], which is due to that it uses all basic polynomials $f_{ij}(x_i, x_j)$ with $0 \leq i < j \leq n$. Although the first strategy gets the same asymptotic result as [2, 26] when $n \to \infty$, it obtains worse bounds for given n, which is because that it only utilizes all basic polynomials instead of their shifts or powers. We present the concrete comparison in Table 4.

Experimental results. The first strategy works successfully with low lattice dimensions and we can easily obtain the experimental results for a few values of $n \leq 14$ as given in Table 2. In our experiments, we find out that the condition $\det(L) < p^{\omega - o(1)}$ is sufficient for the attack in the first strategy.

Table 2 Experimental results of the first strategy on low bounds of $\frac{\delta}{\log_2 p}$ for 1000-bit p.

n	Low bound $\frac{1}{2} + \frac{1}{2n+2} + \epsilon_1$ (theory)	Asymptotic low bound $\frac{1}{2} + \frac{1}{2n+2}$ (theory)	Low bound (experiment)	Lattice Dimension	Time in Seconds	
					LLL	Gröbner basis
4	0.766	0.600	0.602	16	< 1	< 1
6	0.703	0.571	0.573	29	2.70	< 1
8	0.668	0.556	0.558	46	14.78	1.42
10	0.648	0.545	0.548	67	61.22	9.62
12	0.635	0.539	0.541	92	210.12	43.68
14	0.629	0.533	0.536	121	1002.52	161.62

5 The Second Strategy on Solving (7)

In this section, we give the second strategy on Solving (7) and use it to analyze MIHNP and ICG. First, let us define the notation

$$F(n,d) = \frac{2d \sum_{s=0}^{d} s\binom{n}{s}}{d(d+1)\sum_{s=0}^{d}\binom{n}{s} + 2(d+1)\sum_{s=0}^{d} s\binom{n}{s}}$$

and the set

$$I(n,d) = \{(i_0, i_1, \cdots, i_n) \mid 0 \leq i_0 \leq d, 0 \leq i_1, \cdots, i_n \leq 1, 0 \leq i_1 + \cdots + i_n \leq d\}$$

where integers n, d satisfy $0 \leq d \leq n$. Then, we propose the following heuristic result, which is also confirmed by our experiments.

Result 2. *For given polynomials $f_{0j}(x_0, x_j)$ with $1 \leq j \leq n$ in (7), under Assumptions 1 and 2, one can solve (7) in polynomial time, if the bound X of $x_0, x_1, \cdots, x_n$ satisfies*

$$X < p^{F(n,d) - \epsilon_2}$$

where ϵ_2 is some positive real number.

Proof. First, we construct the polynomial $f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ such that the leading monomial is $x_0^{i_0} x_1^{i_1} \cdots x_n^{i_n}$ in terms of the defined order (1), where the tuples $(i_0, i_1, \cdots, i_n) \in I(n, d)$.

Case 1: When $i_1 + \cdots + i_n = 0$, we have $i_1 = \cdots = i_n = 0$ and $x_0^{i_0} x_1^{i_1} \cdots x_n^{i_n} = x_0^{i_0}$. We generate the polynomial

$$f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = x_0^{i_0}. \tag{9}$$

Clearly, $x_0^{i_0} x_1^{i_1} \cdots x_n^{i_n}$ is the corresponding leading monomial.

Case 2: When $i_1 + \cdots + i_n \geq 1$, let $s = i_1 + \cdots + i_n$, we can write

$$x_0^{i_0} x_1^{i_1} \cdots x_n^{i_n} = x_0^{i_0} \cdot (x_{j_1} \cdots x_{j_s})$$

where $1 \leq s \leq d$ and $1 \leq j_1 < \cdots < j_s \leq n$. For example,

$$x_0^2 x_1^1 x_2^0 x_3^1 x_4^1 = x_0^2 \cdot (x_1 x_3 x_4).$$

Here $n = 4$, $s = 3$, and $x_{j_1} = x_1$, $x_{j_2} = x_3$, $x_{j_3} = x_4$. Then, we construct the polynomial $f_{i_0,i_1,\ldots,i_n}$ according to the following two situations:

Case 2.a: For $i_0 \geq s$, we generate the polynomial

$$f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = x_0^{i_0 - s} f_{0,j_1} \cdots f_{0,j_s}. \tag{10}$$

It is easy to see that $f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = 0 \bmod p^s$ and $x_0^{i_0} x_1^{i_1} \cdots x_n^{i_n}$ is the leading monomial in this case.

Case 2.b: For $0 \leq i_0 < s$, consider the polynomials

$$g_l(x_0, x_{j_1}, \cdots, x_{j_s}) = (f_{0,j_1} \cdots f_{0,j_{l-1}}) \cdot x_{j_l} \cdot (f_{0,j_{l+1}} \cdots f_{0,j_s}) \text{ for } l = 1, \ldots, s.$$

It is easy to see that $g_l(x_0, x_{j_1}, \cdots, x_{j_s}) = 0 \bmod p^{s-1}$ for $l = 1, \ldots, s$. Note that the polynomials $g_1, \cdots, g_s$ have common monomials

$$x_{j_1} \cdots x_{j_s},\ x_0 \cdot (x_{j_1} \cdots x_{j_s}), \cdots, x_0^{s-1} \cdot (x_{j_1} \cdots x_{j_s}).$$

We can rewrite the polynomials $g_1, \cdots, g_s$ according to the following way:

$$\begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_s \end{pmatrix} = M[j_1, \cdots, j_s] \cdot \begin{pmatrix} x_{j_1} \cdots x_{j_s} \\ x_0 x_{j_1} \cdots x_{j_s} \\ \vdots \\ x_0^{s-1} x_{j_1} \cdots x_{j_s} \end{pmatrix} + \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ \\ h_s \end{pmatrix}. \tag{11}$$

Here, the polynomial h_l $(1 \leq l \leq s)$ is composed of the terms in g_l except for the corresponding terms of the above common monomials and the matrix

$$M[j_1, \cdots, j_s] = \begin{pmatrix} \sigma_{s-1}(\wedge_1) & \cdots & \sigma_1(\wedge_1) & 1 \\ \sigma_{s-1}(\wedge_2) & \cdots & \sigma_1(\wedge_2) & 1 \\ & & \cdots & \\ \sigma_{s-1}(\wedge_s) & \cdots & \sigma_1(\wedge_s) & 1 \end{pmatrix}$$

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

$$\wedge_l := (c_{0,j_1}, \cdots, c_{0,j_{l-1}}, c_{0,j_{l+1}}, \cdots, c_{0,j_s})$$

with $1 \leq i \leq s-1$ and $1 \leq l \leq s$. Let us continue with the monomial $x_0^2 x_1^1 x_2^0 x_3^1 x_4^1$. We first construct

$$g_1 = x_1 \cdot (f_{0,3} f_{0,4}), g_2 = x_3 \cdot (f_{0,1} f_{0,4}), g_3 = x_4 \cdot (f_{0,1} f_{0,3}),$$

which have common monomials $x_1x_3x_4, x_0x_1x_3x_4, x_0^2x_1x_3x_4$. Then, we get

$$\begin{pmatrix} g_1 \\ g_2 \\ g_3 \end{pmatrix} = \begin{pmatrix} c_{0,3}c_{0,4} & c_{0,3}+c_{0,4} & 1 \\ c_{0,1}c_{0,4} & c_{0,1}+c_{0,4} & 1 \\ c_{0,1}c_{0,3} & c_{0,1}+c_{0,3} & 1 \end{pmatrix} \begin{pmatrix} x_1x_3x_4 \\ x_0x_1x_3x_4 \\ x_0^2x_1x_3x_4 \end{pmatrix} + \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix}.$$

We can compute out the determinant of the matrix $M[j_1, \cdots, j_s]$ by mathematical induction, i.e.,

$$\det(M[j_1, \cdots, j_s]) = \prod_{1 \leq l < t \leq s} (c_{0,j_l} - c_{0,j_t}).$$

Under Assumption 2, we have obtained that all $c_{0,j}$ are distinct in $\mathbb{Z}_p$ for $j = 1, \cdots, n$ with overwhelming probability. Then, $\det(M[j_1, \cdots, j_s])$ is coprime to prime p and hence any power of p, and the inverse of $M[j_1, \cdots, j_s]$ exists mod the power of p. Let integer matrix $U[j_1, \cdots, j_s]$ be the inverse of matrix $M[j_1, \cdots, j_s]$ mod p^{s-1}, i.e.,

$$U[j_1, \cdots, j_s] \cdot M[j_1, \cdots, j_s] = I_s \pmod{p^{s-1}}.$$

Multiplying (11) by $U[j_1, \cdots, j_s]$ from the left and taking modulo p^{s-1} on both sides, we get

$$U[j_1, \cdots, j_s] \cdot \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_s \end{pmatrix} \equiv \begin{pmatrix} x_{j_1} \cdots x_{j_s} \\ x_0 x_{j_1} \cdots x_{j_s} \\ \vdots \\ x_0^{s-1} x_{j_1} \cdots x_{j_s} \end{pmatrix} + U[j_1, \cdots, j_s] \cdot \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_s \end{pmatrix} \pmod{p^{s-1}}. \quad (12)$$

Let $(u_{i_0+1,1}, \cdots, u_{i_0+1,s})$ be the (i_0+1)-th row of $U[j_1, \cdots, j_s]$, where $0 \leq i_0 \leq s-1$. Then, we form the polynomial

$$f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = u_{i_0+1,1} \cdot g_1 + \cdots + u_{i_0+1,s} \cdot g_s. \quad (13)$$

Since that $g_l(x_0, x_1, \cdots, x_n) = 0 \bmod p^{s-1}$ for all $1 \leq l \leq s$, there is

$$f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \ldots, x_n) = 0 \bmod p^{s-1}.$$

Moreover, the leading monomial of $f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ in this situation is also $x_0^{i_0} x_1^{i_1} \cdots x_n^{i_n}$ which is analyzed in Appendix A.

Next, we utilize the idea of Coppersmith's technique and construct the lattice $L(n,d)$ using the coefficient vectors of the polynomials $h_{i_0,i_1,\cdots,i_n}(x_0,x_1,\cdots,x_n)$

$$= \begin{cases} p^d \cdot f_{i_0,i_1,\ldots,i_n}(x_0X_0, x_1X_1, \cdots, x_nX_n) & \text{Case 1} \\ p^{d-s} \cdot f_{i_0,i_1,\ldots,i_n}(x_0X_0, x_1X_1, \cdots, x_nX_n) & \text{Case 2.a} \\ p^{d+1-s} \cdot f_{i_0,i_1,\ldots,i_n}(x_0X_0, x_1X_1, \cdots, x_nX_n) & \text{Case 2.b} \end{cases}$$

where $s = i_1 + \cdots + i_n$. Obviously, for all tuples $(i_0, i_1, \cdots, i_n) \in I(n,d)$, there are

$$h_{i_0,i_1,\cdots,i_n}(x_0,x_1,\cdots,x_n) = 0 \bmod p^d.$$

We arrange these polynomials according to the leading monomials and can make the corresponding basis matrix be a lower triangular. It is easy to see that the dimension of $L(n,d)$ is equal to the number of vectors in $I(n,d)$, i.e.,

$$\dim(L(n,d)) = (d+1)\sum_{s=0}^{d}\binom{n}{s}. \tag{14}$$

We can compute the determinant of $L(n,d)$ as

$$\det(L(n,d)) = p^{\alpha(n,d)} \cdot X^{\beta(n,d)}, \tag{15}$$

where

$$\alpha(n,d) = d(d+1)\sum_{s=0}^{d}\binom{n}{s} - d\sum_{s=0}^{d}s\binom{n}{s}$$

and

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

The detail computation is left in Appendix B. By the property of LLL algorithm and Howgrave-Graham's lemma, if the condition (2) satisfies, namely,

$$\left(\omega^{\frac{\omega-n}{2}} 2^{\frac{\omega(\omega-1)}{4}} p^{dn}\right) \cdot \det(L(n,d)) < p^{d\omega} \tag{16}$$

where $\omega = \dim L(n,d)$, after reduction of lattice we get $n+1$ polynomials which contain the root $(x_0, x_1, \cdots, x_n)$ over integers. Under Assumption 1, we can find $x_0, x_1, \cdots, x_n$.

Finally, we analyze the case that (16) holds. Plugging (14) and (15) into (16), we obtain the condition

$$X < \left(\omega^{\frac{\omega-n}{2}} 2^{\frac{\omega(\omega-1)}{4}} p^{dn}\right)^{-\frac{1}{\beta(n,d)}} \cdot p^{\frac{d\omega-\alpha(n,d)}{\beta(n,d)}}.$$

The right side of this condition can be lower bounded by $p^{F(n,d)-\epsilon_2}$ where $\epsilon_2 = \frac{2\log_2\omega+(\omega-1)}{2(d+2)\log_2 p} + \frac{dn}{\beta(n,d)} > 0$. The detail analysis is given in Appendix C. Therefore, in order to make (16) hold, we need

$$X < p^{F(n,d)-\epsilon_2}.$$

If $d\log_2 p \gg \dim(L(n,d))$ and $\beta(n,d) \gg nd$, ϵ_2 is negligible. □

Since $X = \frac{p}{2^\delta}$ about the cases of MIHNP and ICG, we give the following application.

Application 2. *Given $n+1$ samples in MIHNP or $n+1$ outputs in ICG, under Assumptions 1 and 2, we can recover the hidden number or the secret seed in polynomial time when the number δ of known MSBs satisfies*

$$\frac{\delta}{\log_2 p} \geq 1 - F(n,d) + \epsilon_2. \tag{17}$$

Remark 4 From $\sum_{s=0}^{d} s\binom{n}{s} \leq d \sum_{s=0}^{d} \binom{n}{s}$ and $1 \leq d \leq n$, we can deduce

$$F(n,d) = \frac{2d \sum_{s=0}^{d} s\binom{n}{s}}{d(d+1) \sum_{s=0}^{d} \binom{n}{s} + 2(d+1) \sum_{s=0}^{d} s\binom{n}{s}} < \frac{2d}{3d+3} \leq \frac{2n}{3n+3} < \frac{2}{3}.$$

According to (17), we get $\frac{\delta}{\log_2 p} > \frac{1}{3}$.

Remark 5 When $d \ll n$, we have

$$F(n,d) = \frac{2d \sum_{s=0}^{d} s\binom{n}{s}}{d(d+1) \sum_{s=0}^{d} \binom{n}{s} + 2(d+1) \sum_{s=0}^{d} s\binom{n}{s}} = \frac{2d^2\binom{n}{d}(1+o(1))}{3d(d+1)\binom{n}{d}(1+o(1))}.$$

Taking $d = c \cdot n$ where $0 < c \leq \frac{1}{2}$, there is $F(n,d) \to \frac{2}{3}$ when $n \to +\infty$. According to (17), we have $\frac{\delta}{\log_2 p} \to \frac{1}{3}$. This asymptotic result is better than those in [2, 15, 26], the first result of Boneh et al. in [5] and the result of the first strategy in Section 4, and same as the second result in [5], which however did not give an explicit lattice construction.

Experimental results. The second strategy works efficiently with low lattice dimensions and we present our experimental results given in Table 3. In these experiments, we observe that the condition $\det(L(n,d)) < p^{d \cdot \dim(L(n,d))}$ is sufficient to find the desired solutions with an exception where $(n,d) = (3,2)$. We can see that the experiment values are better than the corresponding theoretical results in most cases.

Table 3 Experimental results of the second strategy on low bounds of $\frac{\delta}{\log_2 p}$ for 1000-bit p.

n	d	Low bound $1 - F(n,d) + \epsilon_2$ (theory)	Asymptotic low bound $1 - F(n,d)$ (theory)	Low bound (experiment)	Lattice Dimension	Time in Seconds	
						LLL	Gröbner basis
3	2	0.754	0.625	0.630	21	< 1	< 1
4	3	0.651	0.585	0.555	60	29.57	< 1
5	3	0.613	0.562	0.525	104	1105.61	14.29
6	3	0.594	0.548	0.505	168	11742.05	149.82
6	4	0.581	0.538	0.485	285	285205.42	639.05

Comparison. In Table 4 and Table 5, we compare the result between the second strategy and the existing works. The symbols " $-$ " and " $\times$ " respectively denote that the concrete result was not given and the concrete situation can not be reached in the corresponding paper. In Table 4, we compute the needed minimum value of the ratio $\frac{\delta}{\log_2 p}$ for the fixed n. In Table 5, we present the needed smallest n for the fixed $\frac{\delta}{\log_2 p}$. The corresponding d is optimal in our second strategy. For $n \geq 9$, we can see that the ratio $\frac{\delta}{\log_2 p} < 1/2$, which is better than the results in [2,15,26], the first work in [5] and the result in the first strategy. Interestingly, the ratio $\frac{\delta}{\log_2 p}$ in our result is close to $\frac{1}{3}$ when $n \geq 100$.

Table 4 The minimum value of $\frac{\delta}{\log_2 p}$ for fixed n

Results \ n	1	2	3	4	5	6	7	8	9
[5]	-	-	-	-	-	-	-	-	-
[2]	0.6667	0.5714	0.5333	0.5161	0.5079	0.5039	0.5020	0.5010	0.5005
[15]	0.8889	0.7778	0.7407	0.7222	0.7111	0.7037	0.6984	0.6944	0.6914
[26]	0.6667	0.5417	0.5083	0.5014	0.5002	0.5000	0.5000	0.5000	0.5000
The First Strategy	0.7500	0.6667	0.6250	0.6000	0.5833	0.5714	0.5625	0.5556	0.5500
The Second Strategy	0.7500	0.6667	0.6250	0.5841	0.5611	0.5378	0.5220	0.5073	0.4953
	$d=1$	$d=2$	$d=2,3$	$d=3$	$d=3$	$d=4$	$d=4$	$d=5$	$d=5$

Table 5 The smallest n needed for fixed $\frac{\delta}{\log_2 p}$

Results \ $\frac{\delta}{\log_2 p}$	0.6678	0.5714	0.5005	0.4953	0.4276	0.3782	0.3419
[5, Method I]	$-$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
[5, Method II]	$-$	$-$	$-$	$-$	$-$	$-$	$-$
[15]	200	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
[2]	1	2	9	$\times$	$\times$	$\times$	$\times$
[26]	1	2	8	$\times$	$\times$	$\times$	$\times$
The First Strategy	2	6	999	$\times$	$\times$	$\times$	$\times$
The Second Strategy	2	3	9	9	20	50	100

6 The Third Strategy on Solving (7)

In this section, we give the third strategy for solving (7) and present further improvement on analyzing MIHNP and ICG. First, let us define notations

$$F(n,d,k) = \frac{(2dk+1)\sum_{s=0}^{dk} s\binom{n}{s}_{k+1} - n\sum_{i=0}^{k}\sum_{s=0}^{\min\{dk-i,(n-1)k\}} i^2\binom{n-1}{s}_{k+1}}{(dk+1)dk\sum_{s=0}^{dk}\binom{n}{s}_{k+1} + 2(dk+1)\sum_{s=0}^{dk} s\binom{n}{s}_{k+1}},$$

and $I(n,d,k)$

$$= \left\{(i_0,i_1,\cdots,i_n), 0 \leq i_0 \leq dk, 0 \leq i_1,\cdots,i_n \leq k, 0 \leq i_1+\cdots+i_n \leq dk\right\}$$

where integers n, d, k satisfy $0 \leq d \leq n$ and $k \geq 1$. Then, we give the following heuristic result, which is also certified by the corresponding experiments.

Result 3. *For given polynomials $f_{0j}(x_0, x_j)$ with $1 \leq j \leq n$ in (7), under Assumptions 1 and 2, one can solve (7) in polynomial time, if the bound X of $x_0, x_1, \cdots, x_n$ satisfies*

$$X < p^{F(n,d,k)-\epsilon_3}$$

where ϵ_3 is some positive real number.

Proof. First, we generate the polynomial $f_{i_0,i_1,\cdots,i_n}(x_0, x_1, \cdots, x_n)$ such that the leading monomial is $x_0^{i_0} x_1^{i_1} \cdots x_n^{i_n}$ in terms of the defined order (1), where all $(i_0, i_1, \cdots, i_n) \in I(n, d, k)$. Let $m = \max\{i_1, \cdots, i_n\}$, we discuss the following two situations in accordance with m.

Case 1: When $m = 0$, we have $i_1 = \cdots = i_n = 0$. In this case, we generate

$$f_{i_0,i_1,\cdots,i_n}(x_0, x_1, \cdots, x_n) = x_0^{i_0}.$$

Case 2: When $m > 0$, we classify variables $x_1, \cdots, x_n$ in monomial $x_0^{i_0} x_1^{i_1} \cdots x_n^{i_n}$ according to their exponents. Let $1 \leq s_1 \leq s_2 \leq \cdots \leq s_m \leq n$ and

$$\{x_{j_1}, \cdots, x_{j_{s_1}}\} \subseteq \{x_{j_1}, \cdots, x_{j_{s_2}}\} \subseteq \cdots \subseteq \{x_{j_1}, \cdots, x_{j_{s_m}}\} \subseteq \{x_1, \cdots, x_n\}$$

where the corresponding exponents of the variables in $\{x_{j_1}, \cdots, x_{j_{s_t}}\}$ are greater than or equal to $(m - t + 1)$ for $t = 1, 2, \cdots, m$. Thus, we can rewrite

$$x_0^{i_0} x_1^{i_1} \cdots x_n^{i_n} = x_0^{i_0} \cdot (x_{j_1} \cdots x_{j_{s_1}}) \cdot (x_{j_1} \cdots x_{j_{s_2}}) \cdots (x_{j_1} \cdots x_{j_{s_m}}). \tag{18}$$

It is easy to see that $i_1 + \cdots + i_n = s_1 + \cdots + s_m$. For example,

$$x_0^4 x_1^2 x_2^3 x_3^4 = x_0^4 \cdot x_3 \cdot (x_3 x_2) \cdot (x_3 x_2 x_1) \cdot (x_3 x_2 x_1).$$

Here $n = 3$, $m = 4$, $s_1 = 1$, $s_2 = 2$, $s_3 = s_4 = 3$ and $x_{j_1} = x_3$, $x_{j_2} = x_2$, $x_{j_3} = x_1$.

Case 2.a: For $i_0 \geq (s_1 + \cdots + s_m)$, we generate $f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ which is equal to

$$x_0^{i_0-(s_1+\cdots+s_m)} \cdot (f_{0,j_1} \cdots f_{0,j_{s_1}}) \cdot (f_{0,j_1} \cdots f_{0,j_{s_2}}) \cdots (f_{0,j_1} \cdots f_{0,j_{s_m}}).$$

It is easy to deduce that $f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = 0 \bmod p^{i_1+\cdots+i_n}$ due to $i_1 + \cdots + i_n = s_1 + \cdots + s_m$ and $x_0^{i_0} x_1^{i_1} \cdots x_n^{i_n}$ is the leading monomial of the polynomial $f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ from (18).

Case 2.b: For $i_0 < (s_1 + \cdots + s_m)$, for the sake of analysis, define the variable set $S_t = \{x_{j_1}, \cdots, x_{j_{s_t}}\}$ and introduce the new notation $g(x_0^i; S_t)$ which is a polynomial generated by the second strategy such that $x_0^i \cdot (x_{j_1} \cdots x_{j_{s_t}})$ is the leading monomial for $t = 1, \cdots, m$.

When the integer l satisfies $(s_1 + \cdots + s_l) \leq i_0 < (s_1 + \cdots + s_l + s_{l+1})$ where $0 \leq l \leq m - 1$, we construct the polynomial $f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ as follows:

$$\big(g(x_0^{s_1}; S_1) \cdots g(x_0^{s_l}; S_l)\big) \cdot g(x_0^{i_0-(s_1+\cdots+s_l)}; S_{l+1}) \cdot \big(g(x_0^0; S_{l+2}) \cdots g(x_0^0; S_m)\big). \tag{19}$$

Let us consider the monomial $x_0^4x_1^2x_2^3x_3^4$. Then we have

$$S_1 = \{x_3\}, S_2 = \{x_2, x_3\}, S_3 = S_4 = \{x_1, x_2, x_3\}.$$

Note that $(s_1 + s_2) < i_0 = 4 < (s_1 + s_2 + s_3)$, thus $l = 2$ in this example. Then, we construct the polynomial

$$f_{3,2,3,4}(x_0, x_1, x_2, x_3) := g(x_0^1; S_1) \cdot g(x_0^2; S_2) \cdot g(x_0^1; S_3) \cdot g(x_0^0; S_4).$$

In this situation, the leading monomial of the polynomial in (19) is $x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n}$ and $f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = 0 \bmod p^{(i_1+\cdots+i_n)-(m-l)}$. The corresponding analysis is given in Appendix D.

Next, we construct a lattice $L(n, d, k)$ using coefficient vectors of the polynomials $h_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n)$ which are respectively equal to

$$\begin{cases} p^{dk} \cdot f_{i_0,i_1,\ldots,i_n}(x_0X_0, x_1X_1, \cdots, x_nX_n) & \text{Case 1} \\ p^{dk-(i_1+\cdots+i_n)} \cdot f_{i_0,i_1,\ldots,i_n}(x_0X_0, x_1X_1, \cdots, x_nX_n) & \text{Case 2.a} \\ p^{dk-(i_1+\cdots+i_n)+(m-l)} \cdot f_{i_0,i_1,\ldots,i_n}(x_0X_0, x_1X_1, \cdots, x_nX_n) & \text{Case 2.b} \end{cases}$$

where $m = \max\{i_1, \cdots, i_n\}$ and $0 \le l \le m-1$. Clearly, we have

$$h_{i_0,i_1,\cdots,i_n}(x_0, x_1, \cdots, x_n) = 0 \bmod p^{dk}$$

for all $(i_0, i_1, \ldots, i_n) \in I(n, k, d)$. We arrange all polynomials according to the order of the leading monomials and make the basis matrix of $L(n, d, k)$ lower triangular.

Then, we directly give the following formule and leave the concrete computation on the dimension and determinant of $L(n, d, k)$ in Appendix E:

$$\dim(L(n, d, k)) = (dk + 1)\sum_{s=0}^{dk}\binom{n}{s}_{k+1} \tag{20}$$

and

$$\det(L(n, d, k)) = p^{\alpha(n,d,k)} \cdot X^{\beta(n,d,k)} \tag{21}$$

where $\alpha(n, d, k)=$

$$dk(dk+1)\sum_{s=0}^{dk}\binom{n}{s}_{k+1} + \frac{n}{2}\sum_{i=0}^{k}\sum_{s=0}^{\min\{dk-i,(n-1)k\}} i^2\binom{n-1}{s}_{k+1} - \frac{2dk+1}{2}\sum_{s=0}^{dk} s\binom{n}{s}_{k+1}$$

and

$$\beta(n, d, k) = \frac{dk(dk + 1)}{2}\sum_{s=0}^{dk}\binom{n}{s}_{k+1} + (dk + 1)\sum_{s=0}^{dk} s\binom{n}{s}_{k+1}.$$

By the property of LLL algorithm and Howgrave-Graham's lemma, if the condition (2) satisfies, namely,

$$\left(\omega^{\frac{\omega-n}{2}} 2^{\frac{\omega(\omega-1)}{4}} p^{dkn}\right) \cdot \det(L(n, d, k)) < p^{dk\omega} \tag{22}$$

where $\omega = \dim L(n, d, k)$, after reduction of lattice we obtain $n + 1$ polynomials which contain the root $(x_0, x_1, \cdots, x_n)$ over integers. Under Assumption 1, we can find $x_0, x_1, \cdots, x_n$.

Finally, we analyze the situation on the condition (22) holds. Plugging (20) and (21) into (22) and rearranging this relation, we get

$$X < \left(\omega^{\frac{\omega-n}{2}} 2^{\frac{\omega(\omega-1)}{4}} p^{dkn}\right)^{-\frac{1}{\beta(n,d,k)}} \cdot p^{\frac{dk\omega-\alpha(n,d,k)}{\beta(n,d,k)}}.$$

Further, the right side of this condition can be lower bounded by $p^{F(n,d,k)-\epsilon_3}$ where $\epsilon_3 = \frac{2\log_2\omega+(\omega-1)}{2(dk+2)\log_2 p} + \frac{dkn}{\beta(n,d,k)} > 0$. Its analysis is given in Appendix F. Thus, we need

$$X < p^{F(n,d,k)-\epsilon_3}.$$

Once $dk\log_2 p \gg \dim(L(n,d,k))$ and $\beta(n,d,k) \gg dkn$, ϵ_3 is negligible. □

From $X = \frac{p}{2^\delta}$ about MIHNP and ICG, we can get the following application.

Application 3. *Given $n+1$ samples in MIHNP or $n+1$ outputs in ICG, under Assumptions 1 and 2, we can recover the hidden number or the secret seed in polynomial time when the number δ of known MSBs satisfies*

$$\frac{\delta}{\log_2 p} > 1 - F(n,d,k) + \epsilon_3. \tag{23}$$

Remark 6 For any positive integers k, n and d such that $1 \le d \le n$, we can obtain

$$F(n,d,k) < \frac{(2dk+1)\sum_{s=0}^{dk} s\binom{n}{s}_{k+1}}{(dk+1)dk\sum_{s=0}^{dk}\binom{n}{s}_{k+1} + 2(dk+1)\sum_{s=0}^{dk} s\binom{n}{s}_{k+1}} < \frac{2dk+1}{3(dk+1)} < \frac{2}{3}.$$

According to (23), there is always $\frac{\delta}{\log_2 p} > \frac{1}{3}$ even for sufficiently large positive integers n, k.

Remark 7 When $k = 1$, we can deduce that $I(n,d,k) = I(n,d)$ and $F(n,d,k) = F(n,d)$, which implies that the third strategy for the case of $k = 1$ is same as the second strategy.

Experimental results. Our results are presented in Table 6. In these experiments, we observe that the condition $\det(L(n,d,k)) < p^{dk\cdot\dim L(n,d,k))}$ is sufficient to find the desired roots. We can easily see that the experiment values are slightly better than the corresponding theoretical results.

Table 6 Experimental results of the third strategy on low bounds of $\frac{\delta}{\log_2 p}$ for 1000-bit p.

n	d	k	Low bound	Asymptotic low bound	Low bound	Lattice	Time in Seconds	
			$1 - F(n,d,k) + \epsilon_3$ (theory)	$1 - F(n,d,k)$ (theory)	(experiment)	Dimension	LLL	Gröbner basis
3	2	2	0.615	0.582	0.570	115	1794.22	139.29
4	2	2	0.590	0.555	0.545	250	46303.96	5493.06

Comparison. In order to give the explicit observation on Corollary 3, we present the ratio $\frac{\delta}{\log_2 p}$ for small n and different k in Table 7, where d is optimal for the corresponding n. We can see that the ratio $\frac{\delta}{\log_2 p}$ becomes smaller as k is bigger.

We give Table 8 and Table 9 to indicate relations between the ratio $\frac{\delta}{\log_2 p}$ and n for optimal d. We have known that the third strategy is same as the second strategy when $k = 1$. From Table 8 and Table 9, we can see that the result in the third strategy for $k = +\infty$ is more ideal than that in the second strategy.

Compared to the works in [2, 26], we can find that the asymptotic result ($k = +\infty$) of $\frac{\delta}{\log_2 p}$ in the third strategy is same as those in [2, 26] for $n = 1$. The asymptotic results in the third strategy are slightly better than that in [2] but slightly weaker than that in [26] for $n = 2$ and 3, and $\frac{\delta}{\log_2 p} < 1/2$ for $n \geq 4$ in the third strategy, which is always better than the results in [2, 26].

Table 7 The ratio $\frac{\delta}{\log_2 p}$ about small n and different k

k \ n	1	2	3	4	5	6
10	0.6818	0.5952	0.5308	0.4980	0.4780	0.4560
	d=1	d=2	d=2	d=2	d=2	d=3
20	0.6746	0.5850	0.5230	0.4893	0.4687	0.4492
	d=1	d=1	d=2	d=2	d=2	d=3
30	0.6720	0.5806	0.5203	0.4864	0.4654	0.4469
	d=1	d=1	d=2	d=2	d=2	d=3
40	0.6707	0.5784	0.5190	0.4849	0.4638	0.4458
	d=1	d=1	d=2	d=2	d=2	d=3
50	0.6699	0.5770	0.5182	0.4840	0.4628	0.4451
	d=1	d=1	d=2	d=2	d=2	d=3

Table 8 The minimum value of $\frac{\delta}{\log_2 p}$ for fixed n in the third strategy

k \ n	1	2	3	4	5	6	7	8	9
1	0.7500	0.6667	0.6250	0.5841	0.5611	0.5378	0.5220	0.5073	0.4953
	$d = 1$	$d = 2$	$d = 2, 3$	$d = 3$	$d = 3$	$d = 4$	$d = 4$	$d = 5$	$d = 5$
$+\infty$	0.6667	0.5714	0.5085	0.4748	0.4518	0.4369	0.4235	0.4141	0.4066
	$d = 1$	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 3$	$d = 4$	$d = 4$	$d = 5$

Table 9 The smallest n needed for fixed $\frac{\delta}{\log_2 p}$ in the third strategy

k \ $\delta/\log_2 p$	0.6678	0.5714	0.5005	0.4953	0.4276	0.3782	0.3419
1	2	3	9	9	20	50	100
$+\infty$	1	2	4	4	7	16	86

7 Conclusion

We revisited the modular inversion hidden number problem and the inversive congruential generator. We attacked these two problems by solving small roots of a class of modular polynomial equations. Here three heuristic strategies based on Coppersmith's technique to solve such type of equation system were presented.

For analyzing the modular inversion hidden number problem, we gave a concrete lattice for explaining the best result up to now proposed by Boneh et al., and further improved the lattice construction such that the requirement of samples are fewer. Our attacks on inversive congruential generator improve the existing works.

Acknowledgements.

The authors would like to thank anonymous reviewers for their helpful comments and suggestions.

References

1. Akavia, A.: Advances in Cryptology - CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings, chap. Solving Hidden Number Problem with One Bit Oracle and Advice, pp. 337–354. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). DOI 10.1007/978-3-642-03356-8_20. URL http://dx.doi.org/10.1007/978-3-642-03356-8_20
2. Bauer, A., Vergnaud, D., Zapalowicz, J.C.: Inferring sequences produced by nonlinear pseudorandom number generators using coppersmiths methods. In: M. Fischlin, J. Buchmann, M. Manulis (eds.) Public Key Cryptography-PKC 2012, *Lecture Notes in Computer Science*, vol. 7293, pp. 609–626. Springer Berlin Heidelberg (2012). DOI 10.1007/978-3-642-30057-8_36. URL http://dx.doi.org/10.1007/978-3-642-30057-8_36
3. Blackburn, S., Gomez-Perez, D., Gutierrez, J., Shparlinski, I.: Predicting the inversive generator. In: K. Paterson (ed.) Cryptography and Coding, *Lecture Notes in Computer Science*, vol. 2898, pp. 264–275. Springer Berlin Heidelberg (2003). DOI 10.1007/978-3-540-40974-8_21. URL http://dx.doi.org/10.1007/978-3-540-40974-8_21
4. Blackburn, S.R., Gomez-perez, D., Gutierrez, J., Shparlinski, I.E.: Predicting nonlinear pseudorandom number generators. MATH. COMPUTATION **74**, 2004 (2004)
5. Boneh, D., Halevi, S., Howgrave-Graham, N.: The modular inversion hidden number problem. In: ASIACRYPT 2001, pp. 36–51. Springer (2001)
6. Boneh, D., Venkatesan, R.: Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In: CRYPTO 1996, pp. 129–142. Springer (1996)
7. Boyar, J.: Inferring sequences produced by pseudo-random number generators. J. ACM **36**(1), 129–141 (1989). DOI 10.1145/58562.59305. URL http://doi.acm.org/10.1145/58562.59305
8. Comtet, L.: Advanced Combinatorics. D. Reidel Publishing Company (1974)
9. Cox, D.A.: Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra. Springer (2007)
10. Eichenauer, J., Lehn, J.: A non-linear congruential pseudo random number generator. Statistische Hefte **27**(1), 315–326 (1986). DOI 10.1007/BF02932576. URL http://dx.doi.org/10.1007/BF02932576
11. Eichenauer-Herrmann, J., Herrmann, E., Wegenkittl, S.: A survey of quadratic and inversive congruential pseudorandom numbers, pp. 66–97. Springer New York, New York, NY (1998). DOI 10.1007/978-1-4612-1690-2_4. URL http://dx.doi.org/10.1007/978-1-4612-1690-2_4
12. Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In: Crytography and Coding, pp. 131–142. Springer (1997)
13. Howgrave-Graham, N.A., Smart, N.P.: Lattice attacks on digital signature schemes. Designs, Codes and Cryptography **23**(3), 283–290 (2001). DOI 10.1023/A:1011214926272. URL http://dx.doi.org/10.1023/A:1011214926272
14. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**(4), 515–534 (1982)
15. Ling, S., Shparlinski, I.E., Steinfeld, R., Wang, H.: On the modular inversion hidden number problem. Journal of Symbolic Computation **47**(4), 358–367 (2012)

16. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. Society for Industrial and Applied Mathematics (1992). DOI 10.1137/1.9781611970081. URL http://epubs.siam.org/doi/abs/10.1137/1.9781611970081
17. Niederreiter, H.: New developments in uniform pseudorandom number and vector generation. In: H. Niederreiter, P.S. Shiue (eds.) Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, *Lecture Notes in Statistics*, vol. 106, pp. 87–120. Springer New York (1995). DOI 10.1007/978-1-4612-2552-2_5. URL http://dx.doi.org/10.1007/978-1-4612-2552-2_5
18. Niederreiter, H., Rivat, J.: On the correlation of pseudorandom numbers generated by inversive methods. Monatshefte für Mathematik **153**(3), 251–264 (2008). DOI 10.1007/s00605-007-0503-3. URL http://dx.doi.org/10.1007/s00605-007-0503-3
19. Niederreiter, H., Shparlinski, I.: Recent advances in the theory of nonlinear pseudorandom number generators. In: K.T. Fang, H. Niederreiter, F. Hickernell (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2000, pp. 86–102. Springer Berlin Heidelberg (2002). DOI 10.1007/978-3-642-56046-0_6. URL http://dx.doi.org/10.1007/978-3-642-56046-0_6
20. Niederreiter, H., Winterhof, A.: On the Structure of Inversive Pseudorandom Number Generators, pp. 208–216. Springer Berlin Heidelberg, Berlin, Heidelberg (2007). DOI 10.1007/978-3-540-77224-8_25. URL http://dx.doi.org/10.1007/978-3-540-77224-8_25
21. Pirsic, G., Winterhof, A.: On the structure of digital explicit nonlinear and inversive pseudorandom number generators. Journal of Complexity **26**(1), 43 – 50 (2010). DOI http://dx.doi.org/10.1016/j.jco.2009.07.001. URL http://www.sciencedirect.com/science/article/pii/S0885064X09000661
22. Shparlinski, I.E.: Playing hide-and-seek with numbers: the hidden number problem, lattices, and exponential sums. In: proceeding of symposia in applied mathematics, vol. 62, pp. 153–177 (2005)
23. Stern, J.: Secret linear congruential generators are not cryptographically secure. In: Foundations of Computer Science, 1987., 28th Annual Symposium on, pp. 421–426 (1987). DOI 10.1109/SFCS.1987.51
24. Topuzoğlu, A., Winterhof, A.: On the linear complexity profile of nonlinear congruential pseudorandom number generators of higher orders. Applicable Algebra in Engineering, Communication and Computing **16**(4), 219–228 (2005). DOI 10.1007/s00200-005-0181-0. URL http://dx.doi.org/10.1007/s00200-005-0181-0
25. Winterhof, A.: Recent Results on Recursive Nonlinear Pseudorandom Number Generators, pp. 113–124. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). DOI 10.1007/978-3-642-15874-2_9. URL http://dx.doi.org/10.1007/978-3-642-15874-2_9
26. Xu, J., Hu, L., Huang, Z., Peng, L.: Modular inversion hidden number problem revisited. In: Information Security Practice and Experience, pp. 537–551. Springer (2014)

A The leading monomial of the polynomial in (13)

From (12) and (13), we can deduce

$$f_{i_0,i_1,\ldots,i_n}(x_0,x_1,\cdots,x_n) = x_0^{i_0}\cdot(x_{j_1}\cdots x_{j_s}) + u_{i_0+1,1}\cdot h_1 + \cdots + u_{i_0+1,s}\cdot h_s.$$

Since that $x_0^{i_0}x_1^{i_1}\cdots x_n^{i_n}$ is written as $x_0^{i_0}\cdot(x_{j_1}\cdots x_{j_s})$, thus, our goal is to analyze $x_0^{i_0}\cdot(x_{j_1}\cdots x_{j_s})$ is the leading monomial of $f_{i_0,i_1,\ldots,i_n}(x_0,x_1,\cdots,x_n)$. Note that the polynomial h_l $(1\le l\le s)$ is composed of the terms in g_l except for the corresponding terms of monomials

$$x_{j_1}\cdots x_{j_s},\ x_0\cdot(x_{j_1}\cdots x_{j_s}),\cdots,x_0^{s-1}\cdot(x_{j_1}\cdots x_{j_s}).$$

Let $x_0^{l_0}x_{k_1}\cdots x_{k_t}$ be a monomial of $u_{i_0+1,1}\cdot h_1+\cdots+u_{i_0+1,s}\cdot h_s$. Hence, we can obtain $\{k_1,\cdots,k_t\}\subset\{j_1,\ldots,j_s\}$ where $t<s$. According to the defined order (1), there is

$$x_0^{l_0}\cdot(x_{k_1}\cdots x_{k_t}) \prec x_0^{i_0}\cdot(x_{j_1}\cdots x_{j_s}),$$

which implies that the leading monomial of $f_{i_0,i_1,\ldots,i_n}(x_0,x_1,\cdots,x_n)$ is $x_0^{i_0}\cdot(x_{j_1}\cdots x_{j_s})$.

B Computation of the Determinant of $L(n, d)$

Note that the determinant of $L(n, d)$ is product of the diagonal entries. For the case 1, the contribution of $h_{i_0,i_1,\cdots,i_n}(x_0, x_1, \cdots, x_n)$ to the determinant of $L(n, d)$ is

$$\prod_{i_0=0}^{d} \left(p^d \cdot X^{i_0}\right).$$

For the case 2.a, the contribution of $h_{i_0,i_1,\cdots,i_n}(x_0, x_1, \cdots, x_n)$ to the determinant of $L(n, d)$ is given by:

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

For the case 2.b, the contribution of $f_{i_0,i_1,\cdots,i_n}(x_0X, x_1X, \cdots, x_nX)$ is:

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

To sum up, we get

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

where $\alpha(n, d) = d(d + 1) \sum_{s=0}^{d} \binom{n}{s} - d \sum_{s=0}^{d} s\binom{n}{s}$ and $\beta(n, d) = \frac{d(d+1)}{2} \sum_{s=0}^{d} \binom{n}{s} + (d + 1) \sum_{s=0}^{d} s\binom{n}{s}$.

C Computation on $p^{F(n,d)-\epsilon_2}$

Our goal is to derive a lower bound of

$$\left(\omega^{\frac{\omega-n}{2}} 2^{\frac{\omega(\omega-1)}{4}} p^{dn}\right)^{-\frac{1}{\beta(n,d)}} \cdot p^{\frac{d\omega-\alpha(n,d)}{\beta(n,d)}}.$$

First, from the values $\beta(n, d) = \frac{d(d+1)}{2} \sum_{s=0}^{d} \binom{n}{s} + (d + 1) \sum_{s=0}^{d} s\binom{n}{s}$ and $\omega = \dim(L(n, d)) = (d + 1) \sum_{s=0}^{d} \binom{n}{s}$, we get $\frac{\omega}{\beta(n,d)} < \frac{2}{d+2}$. Further, there is

$$\left(\omega^{\frac{\omega-n}{2}} 2^{\frac{\omega(\omega-1)}{4}} p^{dn}\right)^{\frac{1}{\beta(n,d)}} < \omega^{\frac{1}{d+2}} 2^{\frac{\omega-1}{2(d+2)}} p^{\frac{dn}{\beta(n,d)}} = p^{\epsilon_2},$$

where $\epsilon_2 = \frac{2\log_2 \omega+\omega-1}{2(d+2)\log_2 p} + \frac{dn}{\beta(n,d)}$.

Second, plugging the values ω, $\alpha(n, d)$ and $\beta(n, d)$ into $\frac{d\omega-\alpha(n,d)}{\beta(n,d)}$, we get that

$$\frac{d\omega - \alpha(n, d)}{\beta(n, d)} = \frac{2d \sum_{s=0}^{d} s\binom{n}{s}}{d(d + 1) \sum_{s=0}^{d} \binom{n}{s} + 2(d + 1) \sum_{s=0}^{d} s\binom{n}{s}} = F(n, d).$$

Thus,

$$\left(\omega^{\frac{\omega-n}{2}} 2^{\frac{\omega(\omega-1)}{4}} p^{dn}\right)^{-\frac{1}{\beta(n,d)}} \cdot p^{\frac{d\omega-\alpha(n,d)}{\beta(n,d)}} > p^{F(n,d)-\epsilon_2}.$$

D Analysis of the polynomial in (19)

First, we analyze the leading monomial of the polynomial. We know that the leading monomial of the polynomial in (19) is the product of the leading monomials of the following polynomials:

$$g(x_0^{s_1}; S_1), \cdots, g(x_0^{s_l}; S_l), g(x_0^{i_0-(s_1+\cdots+s_l)}; S_{l+1}), g(x_0^0; S_{l+2}), \cdots, g(x_0^0; S_m).$$

Note that the leading monomials of $g(x_0^{s_1}; S_1), \cdots, g(x_0^{s_l}; S_l)$ are respectively

$$x^{s_1} \cdot (x_{j_1} \cdots x_{j_{s_1}}), \cdots, x^{s_l} \cdot (x_{j_1} \cdots x_{j_{s_l}}),$$

the leading monomial of $g(x_0^{i_0-(s_1+\cdots+s_l)}; S_{l+1})$ is $x^{i_0-(s_1+\cdots+s_l)} \cdot (x_{j_1} \cdots x_{j_{s_{l+1}}})$ and the leading monomials of $g(x_0^0; S_{l+2}), \cdots, g(x_0^0; S_m)$ are respectively

$$(x_{j_1} \cdots x_{j_{s_{l+2}}}), \cdots, (x_{j_1} \cdots x_{j_{s_m}}).$$

It is easy to compute that the leading monomial of the polynomial in (19) is

$$x^{i_0} \cdot (x_{j_1} \cdots x_{j_{s_1}}) \cdot (x_{j_1} \cdots x_{j_{s_2}}) \cdots (x_{j_1} \cdots x_{j_{s_m}})$$

which is equal to $x_0^{i_0} x_1^{i_1} \cdots x_n^{i_n}$ from (18) directly. Hence, the leading monomial of the polynomial in (19) is $x_0^{i_0} x_1^{i_1} \cdots x_n^{i_n}$.

Next, we show that the polynomial in (19) has the following relation

$$f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = 0 \bmod p^{(i_1+\cdots+i_n)-(m-l)}.$$

Note that the polynomials in (19) are composed of the polynomials generated by the second strategy. From $|S_1| = s_1, \cdots, |S_l| = s_l$, we have

$$g(x_0^{s_1}; S_1) = 0 \bmod p^{s_1}, \cdots, g(x_0^{s_l}; S_l) \bmod p^{s_l}.$$

From $s_1 + \cdots + s_l \le i_0 < (s_1 + \cdots + s_l) + s_{l+1}$, we get $0 \le i_0 - (s_1 + \cdots + s_l) < s_{l+1} = |S_{l+1}|$, thus there is

$$g(x_0^{i_0-(s_1+\cdots+s_l)}; S_{l+1}) = 0 \bmod p^{s_{l+1}-1}.$$

According to $|S_m| \ge \cdots \ge |S_{l+2}| \ge |S_{l+1}| = s_{l+1} > 0$, we obtain

$$g(x_0^0; S_{l+2}) = 0 \bmod p^{s_{l+2}-1}, \cdots, g(x_0^0; S_m) = 0 \bmod p^{s_m-1}.$$

Therefore, we get $f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = 0 \bmod p^{(s_1+\cdots+s_m)+(m-l)}$. From (18), we have $s_1 + \cdots + s_m = i_1 + \cdots + i_n$. Hence

$$f_{i_0,i_1,\ldots,i_n}(x_0, x_1, \cdots, x_n) = 0 \bmod p^{(i_1+\cdots+i_n)-(m-l)}.$$

E Computation on Dimension and Determinant of $L(n, d, k)$

Let

$$S(n, d, k) = \{(i_1, \cdots, i_n), 0 \le i_1, \cdots, i_n \le k, 0 \le i_1 + \cdots + i_n \le dk\}.$$

Denote $|S(n, d, k)|$ is the cardinality of $S(n, d, k)$. Note that $|S(n, d, k)|$ can also be regarded as the sum of coefficients of the x^s in the expansion of the polynomial $(1 + x + \cdots + x^k)^n$, $s = 0, 1, \cdots, dk$. Namely, $|S(n, d, k)| = \sum_{s=0}^{dk} \binom{n}{s}_{k+1}$.

First, we compute the dimension of $L(n, d, k)$. Clearly, the dimension of the lattice $L(n, d, k)$ is equal to the number of vectors in $I(n, d, k)$, which can be expressed as $(dk + 1) \cdot |S(n, d, k)|$, Therefore,

$$\dim(L(n, d, k)) = (dk + 1) \sum_{s=0}^{dk} \binom{n}{s}_{k+1}.$$

Then, we compute the determinant of $L(n,d,k)$. Since that the determinant of $L(n,d,k)$ is product of the diagonal entries where all tuples $(i_0,i_1,\cdots,i_n)\in I(n,d,k)$, we consider the following cases.

For the case 1 and case 2.a, the contribution of $h_{i_0,i_1,\cdots,i_n}(x_0,x_1,\cdots,x_n)$ to the determinant of $L(n,d)$ is

$$\prod_{(i_1,\cdots,i_n)\in S(n,d,k)}\prod_{i_0=i_1+\cdots+i_n}^{dk}\left(p^{dk-(i_1+\cdots+i_n)}\cdot X^{i_0+i_1+\cdots+i_n}\right),$$

where $i_1+\cdots+i_n=0$ in the case 1 and $i_1+\cdots+i_n>0$ in the case 2.a.

For the case 2.b, the contribution of $h_{i_0,i_1,\cdots,i_n}(x_0X,x_1X,\cdots,x_nX)$ is given as follows:

$$\prod_{(i_1,\cdots,i_n)\in S(n,d,k)}\prod_{l=0}^{m-1}\prod_{i_0=s_1+s_2+\cdots+s_l}^{s_1+s_2+\cdots+s_{l+1}-1}\left(p^{dk-(i_1+\cdots+i_n)+(m-l)}\cdot X^{i_0+i_1+\cdots+i_n}\right),$$

which can be rearranged as

$$\prod_{(i_1,\cdots,i_n)\in S(n,d,k)}\left(p^{\sum_{l=0}^{m-1}(m-l)s_{l+1}}\cdot\prod_{i_0=0}^{i_1+\cdots+i_n-1}\left(p^{dk-(i_1+\cdots+i_n)}\cdot X^{i_0+i_1+\cdots+i_n}\right)\right)$$

according to the relation $s_1+\cdots+s_m=i_1+\cdots+i_n$ in (18).

Thus, we can get that $\det(L(n,d,k))=$

$$\prod_{(i_1,\cdots,i_n)\in S(n,d,k)}\left(p^{\sum_{l=0}^{m-1}(m-l)s_{l+1}}\cdot\prod_{i_0=0}^{dk}\left(p^{dk-(i_1+\cdots+i_n)}\cdot X^{i_0+i_1+\cdots+i_n}\right)\right).$$

First, let us compute $\prod_{(i_1,\cdots,i_n)\in S(n,d,k)}\prod_{i_0=0}^{dk}p^{dk-(i_1+\cdots+i_n)}$. We can deduce that

$$\begin{aligned}&\sum_{(i_1,\cdots,i_n)\in S(n,d,k)}\sum_{i_0=0}^{dk}(dk-(i_1+\cdots+i_n))\\&=dk(dk+1)\cdot\sum_{(i_1,\cdots,i_n)\in S(n,d,k)}1-(dk+1)\cdot\sum_{(i_1,\cdots,i_n)\in S(n,d,k)}(i_1+\cdots+i_n)\\&=dk(dk+1)\sum_{s=0}^{dk}\binom{n}{s}_{k+1}-(dk+1)\sum_{s=0}^{dk}s\binom{n}{s}_{k+1}\end{aligned}$$

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

$$\prod_{(i_1,\cdots,i_n)\in S(n,d,k)}\prod_{i_0=0}^{dk}p^{dk-(i_1+\cdots+i_n)}=p^{dk(dk+1)\sum_{s=0}^{dk}\binom{n}{s}_{k+1}-(dk+1)\sum_{s=0}^{dk}s\binom{n}{s}_{k+1}}.$$

Second, let us compute $\prod_{(i_1,\cdots,i_n)\in S(n,d,k)}\prod_{i_0=0}^{dk}X^{i_0+i_1+\cdots+i_n}$. Note that

$$\begin{aligned}&\sum_{(i_1,\cdots,i_n)\in S(n,d,k)}\sum_{i_0=0}^{dk}(i_0+i_1+\cdots+i_n)\\&=\tfrac{dk(dk+1)}{2}\cdot\sum_{(i_1,\cdots,i_n)\in S(n,d,k)}1-(dk+1)\cdot\sum_{(i_1,\cdots,i_n)\in S(n,d,k)}(i_1+\cdots+i_n)\\&=\tfrac{dk(dk+1)}{2}\sum_{s=0}^{dk}\binom{n}{s}_{k+1}-(dk+1)\sum_{s=0}^{dk}s\binom{n}{s}_{k+1}.\end{aligned}$$

Therefore,

$$\prod_{(i_1,\cdots,i_n)\in S(n,d,k)}\prod_{i_0=0}^{dk} X^{i_0+i_1+\cdots+i_n} = X^{\frac{dk(dk+1)}{2}\sum_{s=0}^{dk}\binom{n}{s}_{k+1}-(dk+1)\sum_{s=0}^{dk}s\binom{n}{s}_{k+1}}.$$

Third, let us compute $\prod_{(i_1,\cdots,i_n)\in S(n,d,k)} p^{\sum_{l=0}^{m-1}(m-l)s_{l+1}}$. We can rewrite

$$\sum_{l=0}^{m-1}(m-l)s_{l+1} = \sum_{l=0}^{m-1}(s_{l+1}-s_l)(1+\cdots+m-l)$$

where $s_0=0$. Note that there are exactly $(s_{l+1}-s_l)$ entries that are equal to $(m-l)$ in the exponent set $\{i_1,\cdots,i_n\}$ for $l=0,1,\cdots,m-1$. Hence, $\sum_{l=0}^{m-1}(s_{l+1}-s_l)(1+\cdots+m-l)$ can be regarded as a rearrangement of $(1+\cdots+i_1)+\cdots+(1+\cdots+i_n)$, which is computed as $\frac{i_1+\cdots+i_n+i_1^2+\cdots+i_n^2}{2}$. Therefore,

$$\sum_{(i_1,\cdots,i_n)\in S(n,d,k)}\sum_{l=0}^{m-1}(t-l)s_{l+1} = \sum_{(i_1,\cdots,i_n)\in S(n,d,k)}\frac{i_1+\cdots+i_n+i_1^2+\cdots+i_n^2}{2}.$$

We have known $\sum_{(i_1,\cdots,i_n)\in S(n,d,k)}(i_1+\cdots+i_n) = \sum_{s=0}^{dk}s\binom{n}{s}_{k+1}$. Next, we analyze

$$\sum_{(i_1,\cdots,i_n)\in S(n,d,k)}\left(i_1^2+\cdots+i_n^2\right).$$

Since

$$\sum_{(i_1,\cdots,i_n)\in S(n,d,k)}\left(i_1^2+\cdots+i_n^2\right) = n\cdot\sum_{(i_1,\cdots,i_n)\in S(n,d,k)}i_n^2$$

Here, $0\le i_n\le k$ and $0\le(i_1+\cdots+i_{n-1})\le\min\{dk-i_n,(n-1)k\}$ as $(i_1,\cdots,i_n)\in S(n,d,k)$. Thus, we can rewrite the above relation using polynomial coefficients:

$$\sum_{(i_1,\cdots,i_n)\in S(n,d,k)}\left(i_1^2+\cdots+i_n^2\right) = n\cdot\sum_{i=0}^{k}\sum_{s=0}^{\min\{dk-i,(n-1)k\}}i^2\binom{n-1}{s}_{k+1}.$$

Further, we obtain

$$\prod_{(i_1,\cdots,i_n)\in S(n,d,k)} p^{\sum_{l=0}^{m-1}(m-l)s_{l+1}} = p^{\frac{n}{2}\cdot\sum_{i=0}^{k}\sum_{s=0}^{\min\{dk-i,(n-1)k\}}i^2\binom{n-1}{s}_{k+1}+\frac{1}{2}\sum_{s=0}^{dk}s\binom{n}{s}_{k+1}}.$$

According to the above analysis, we get $\det(n,d,k)=p^{\alpha(n,d,k)}\cdot X^{\beta(n,d,k)}$, where $\alpha(n,d,k)=$

$$dk(dk+1)\sum_{s=0}^{dk}\binom{n}{s}_{k+1}+\frac{n}{2}\sum_{i=0}^{k}\sum_{s=0}^{\min\{dk-i,(n-1)k\}}i^2\binom{n-1}{s}_{k+1}-\frac{2dk+1}{2}\sum_{s=0}^{dk}s\binom{n}{s}_{k+1}$$

and

$$\beta(n,d,k)=\frac{dk(dk+1)}{2}\sum_{s=0}^{dk}\binom{n}{s}_{k+1}+(dk+1)\sum_{s=0}^{dk}s\binom{n}{s}_{k+1}.$$

F Computation on $p^{F(n,d,k)-\epsilon_3}$

Our goal is to show a lower bound of

$$\left(\omega^{\frac{\omega-n}{2}}2^{\frac{\omega(\omega-1)}{4}}p^{ndk}\right)^{-\frac{1}{\beta(n,d,k)}}\cdot p^{\frac{dk\omega-\alpha(n,d,k)}{\beta(n,d,k)}}$$

where $\omega=\dim(L(n,d,k))$.

First, from the values $\dim(L(n,d,k))$ and $\beta(n,d,k)$, we get $\frac{\omega}{\beta(n,d,k)}<\frac{2}{dk+2}$, furthermore,

$$\left(\omega^{\frac{\omega-n}{2}}2^{\frac{\omega(\omega-1)}{4}}p^{ndk}\right)^{\frac{1}{\beta(n,d,k)}}<\omega^{\frac{1}{dk+2}}2^{\frac{\omega-1}{2(dk+2)}}p^{\frac{ndk}{\beta(n,d,k)}}=p^{\epsilon_3}$$

where $\epsilon_3=\frac{2\log_2\omega+(\omega-1)}{2(dk+2)\log_2 p}+\frac{ndk}{\beta(n,d,k)}$.

Second, according to the values $\alpha(n,d,k)$, $\beta(n,d,k)$, ω and $F(n,d,k)$, we can compute

$$\frac{dk\omega-\alpha(n,d,k)}{\beta(n,d,k)}=F(n,d,k).$$

Therefore, there is the following relation

$$\left(\omega^{\frac{\omega-n}{2}}2^{\frac{\omega(\omega-1)}{4}}p^{ndk}\right)^{-\frac{1}{\beta(n,d,k)}}\cdot p^{\frac{d\omega-\alpha(n,d,k)}{\beta(n,d,k)}}>p^{F(n,d,k)-\epsilon_3}.$$

G Sage Code for the first strategy

```
n=14


P=ZZ.random_element(2^999,2^1000)
P=next_prime(P)
a=ZZ.random_element(P)
X=[]
B=[]
E=[]
bb=1000-536
for i in range(n+1):
    xi=ZZ.random_element(P)
    X.append(xi)
    yi=(a+xi).inverse_mod(P)
    bi=yi-yi%2^bb
    ei=yi%2^bb
    B.append(bi)
    E.append(ei)

R.<z0,z1,z2,z3,z4,z5,z6,z7,z8,z9,z10,z11,z12,z13,z14>=QQ[]
Z=[z0,z1,z2,z3,z4,z5,z6,z7,z8,z9,z10,z11,z12,z13,z14]
# U is the upper bound of the root
U=ZZ.random_element(2^(bb-1),2^(bb))
G=[]
H=[]
for i in range(n+1):
    for j in range(i+1,n+1):
        Aij=X[i]-X[j]
        Bij=X[i]*B[j]-X[j]*B[j]+1
        Cij=X[i]*B[i]-X[j]*B[i]-1
        Dij=(X[i]-X[j])*B[i]*B[j]+B[i]-B[j]
```

```
        f=Aij*Z[i]*Z[j]+Bij*Z[i]+Cij*Z[j]+Dij
        bn=(Aij).inverse_mod(P)

        g=Z[i]*Z[j]+(Bij*bn)%P*Z[i]+(Cij*bn)%P*Z[j]+(Dij*bn)%P
        G.append(g)
        H=union(H,g.monomials())


for i in range(n+1):
   G.append(Z[i]*P)
G.append(1*P)
cc=Set(H).cardinality()
print 'Dimension of the Lattice=', cc
M=matrix(ZZ,cc,cc,range(cc*cc))
for i in range(cc):
    g=R(G[i])(z0*U,z1*U,z2*U,z3*U,z4*U,z5*U,z6*U,z7*U,z8*U,z9*U,z10*U,z11*U,z12*U,z13*U,z14*U)
    for j in range(cc):
        cij=g.coefficient(H[j])(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
        M[i,j]=cij
tt=cputime()
M=M.LLL()
print 'Time of LLL algorithm',cputime(tt)
F=[]

M3=[]
for j in range(116):
  f1=0
  for i in range(cc):
    f1=f1+(M[j][i]/H[i](U,U,U,U,U,U,U,U,U,U,U,U,U,U,U))*H[i]

  M3.append(f1)


I=(M3)*R
tt=cputime()
B=I.groebner_basis()
print 'Time of Groebner basis', cputime(tt)
print B[0],E[0]
```

H Sage Code for the second strategy

```
P=ZZ.random_element(2^999,2^1000)
P=next_prime(P)

n=4
d=3
a=ZZ.random_element(P)
X=[]
B=[]
E=[]
bb=1000-555
for i in range(n+1):
    xi=ZZ.random_element(P)
    X.append(xi)
    yi=(a+xi).inverse_mod(P)
    bi=yi-yi%2^bb
```

```
    ei=yi%2^bb
    B.append(bi)

    E.append(ei)



R.<z0,z1,z2,z3,z4>=QQ[]
Z=[z0,z1,z2,z3,z4]

G=[]
H=[]
a=0
G2=[]
i=0
for j in range(1,n+1):
        Aij=X[0]-X[j]
        Bij=X[0]*B[j]-X[j]*B[j]+1
        Cij=X[0]*B[0]-X[j]*B[0]-1
        Dij=(X[0]-X[j])*B[0]*B[j]+B[0]-B[j]
        f=Aij*Z[0]*Z[j]+Bij*Z[0]+Cij*Z[j]+Dij
        bn=(Aij).inverse_mod(P)

        g=Z[0]*Z[j]+(Bij*bn)%P*Z[0]+(Cij*bn)%P*Z[j]+(Dij*bn)%P
        G.append(g)
        G2.append([i,j])
        H=union(H,g.monomials())

# U is the upper bound of the root
U=ZZ.random_element(2^(bb-1), 2^bb)


f=0
for i1 in range(n):
    for i2 in range(i1+1,n):
        for i3 in range(i2+1,n):
                  f=f+G[i1]*G[i2]*G[i3]



M=(f).monomials()
ll=len(M)


print 'Dimension of the Lattice=', ll


F=[]
for i in range(len(M)):
    ss=M[i]
    s0=diff(M[i],z0)(1,1,1,1,1)
    s1=diff(M[i],z1)(1,1,1,1,1)
    s2=diff(M[i],z2)(1,1,1,1,1)
    s3=diff(M[i],z3)(1,1,1,1,1)
    s4=diff(M[i],z4)(1,1,1,1,1)

    if(s1+s2+s3+s4==0):
        F.append(M[i]*P^d)
```

```
else:
    s=s1+s2+s3+s4
    if(s0>=s):
       f=z0^(s0-s)*(G[0])^s1*(G[1])^s2*(G[2])^s3*(G[3])^s4
       F.append(f*P^(d-s))

    else:
        F2=[]
        F3=[]
        F4=[]
        if(s1==1):
            F2.append(z1)
            F3.append(G[0])

        if(s2==1):
            F2.append(z2)
            F3.append(G[1])

        if(s3==1):
            F2.append(z3)
            F3.append(G[2])

        if(s4==1):
            F2.append(z4)
            F3.append(G[3])

        for i in range(len(F2)):
            f=1
            for j in range(len(F2)):
                if(j==i):
                    f=f*F2[i]

                else:
                    f=f*F3[j]

            F4.append(f)

        F1=[]
        for j in range(s):
            F1.append(z0^j*z1^s1*z2^s2*z3^s3*z4^s4)
        if(s>1):
          R1=IntegerModRing(P^(s-1))
          MS=Matrix(R1,s,s,range(s*s))

          for j in range(s):
            for l in range(s):
                MS[j,l]=0

          for j in range(s):
            for l in range(s):
                cjl=(F4[j]).coefficient(F1[l])
                cjl=cjl(0,0,0,0,0)
                MS[j,l]=cjl

          MSIN=MS.inverse()

          F5=[]
          for l  in range(s0,s0+1):
            f=0
```

```
                for j in range(s):
                    f=f+ZZ(MSIN[l][j])*F4[j]
                S2=f.monomials()
                g=0
                for j in range(len(S2)):
                    cj=f.coefficient(S2[j])
                    cj=cj(0,0,0,0,0)
                    cj=cj%P^(s-1)
                    cj=ZZ(cj)
                    g=g+cj*S2[j]
                f=g
                F.append(f*P^(d+1-s))
            else:
              F.append(F4[j]*P^d)

rr=len(F)
M2=Matrix(ZZ,rr,rr,range(rr*rr))
for i in range(rr):
    f=F[i]
    f=f(z0*U,z1*U,z2*U,z3*U,z4*U)
    for j in range(len(M)):
        cij=f.coefficient(M[j])
        cij=cij(0,0,0,0,0)
        M2[i,j]=cij
tt=cputime()
M2=M2.LLL()
print 'LLL Time',cputime(tt)
M3=[]
for j in range(25):
  f1=0
  for i in range(rr):
    f1=f1+(M2[j][i]/M[i](U,U,U,U,U))*M[i]
  M3.append(f1)

I=(M3)*R
tt=cputime()
B=I.groebner_basis()
print 'Groebner Basis Time', cputime(tt)
print B[0],E[0]
```

I Sage Code for the third strategy

```
P=ZZ.random_element(2^999,2^1000)
P=next_prime(P)

n=3
d=2
k=2
a=ZZ.random_element(P)
X=[]
B=[]
E=[]
bb=1000-570
for i in range(n+1):
    xi=ZZ.random_element(P)
    X.append(xi)
    yi=(a+xi).inverse_mod(P)
```

```
    bi=yi-yi%2^bb
    ei=yi%2^bb
    B.append(bi)

    E.append(ei)



R.<z0,z1,z2,z3>=QQ[]
Z=[z0,z1,z2,z3]
G=[]
H=[]

G2=[]
for j in range(1,n+1):
        Aij=X[0]-X[j]
        Bij=X[0]*B[j]-X[j]*B[j]+1
        Cij=X[0]*B[0]-X[j]*B[0]-1
        Dij=(X[0]-X[j])*B[0]*B[j]+B[0]-B[j]
        f=Aij*Z[0]*Z[j]+Bij*Z[0]+Cij*Z[j]+Dij
        bn=(Aij).inverse_mod(P)

        g=Z[0]*Z[j]+(Bij*bn)%P*Z[0]+(Cij*bn)%P*Z[j]+(Dij*bn)%P
        print g(E[0],E[1],E[2],E[3])%P
        G.append(g)
        G2.append([i,j])
        H=union(H,g.monomials())


#U is the upper bound of the root
U=ZZ.random_element(2^(bb-1), 2^bb)

#This is the starting position for polynomial generation  for Case 2.b of the paper


M=[]
for i0 in range(n):
    for i1 in range(1+1):
        for i2 in range(1+1):
            for i3 in range(1+1):
                if(i1+i2+i3<=n):
                    M.append(z0^i0*z1^i1*z2^i2*z3^i3)

F33=[]
F44=[]
for i in range(len(M)):
    ss=M[i]
    s0=diff(M[i],z0)(1,1,1,1)
    s1=diff(M[i],z1)(1,1,1,1)
    s2=diff(M[i],z2)(1,1,1,1)
    s3=diff(M[i],z3)(1,1,1,1)


    if(s1+s2+s3==0):
        F33.append(M[i])
        F44.append(0)


    else:
```

```
s=s1+s2+s3
if(s0>=s):
   f=z0^(s0-s)*(G[0])^s1*(G[1])^s2*(G[2])^s3
   F33.append(f)
   F44.append(s)


else:
    F2=[]
    F3=[]
    F4=[]
    if(s1==1):
        F2.append(z1)
        F3.append(G[0])

    if(s2==1):
        F2.append(z2)
        F3.append(G[1])

    if(s3==1):
        F2.append(z3)
        F3.append(G[2])


    for i in range(len(F2)):
        f=1
        for j in range(len(F2)):
            if(j==i):
                f=f*F2[i]

            else:
                f=f*F3[j]

        F4.append(f)

    F1=[]
    for j in range(s):
        F1.append(z0^j*z1^s1*z2^s2*z3^s3)

    if(s>1):
      R1=IntegerModRing(P^(s-1))
      MS=Matrix(R1,s,s,range(s*s))

      for j in range(s):
        for l in range(s):
            MS[j,l]=0

      for j in range(s):
        for l in range(s):
            cjl=(F4[j]).coefficient(F1[l])
            cjl=cjl(0,0,0,0)
            MS[j,l]=cjl

      MSIN=MS.inverse()

      F5=[]
      for l  in range(s0,s0+1):
        f=0
        for j in range(s):
            f=f+ZZ(MSIN[l][j])*F4[j]
```

```
                S2=f.monomials()
                g=0
                for j in range(len(S2)):
                    cj=f.coefficient(S2[j])
                    cj=cj(0,0,0,0)
                    cj=cj%P^(s-1)
                    cj=ZZ(cj)
                    g=g+cj*S2[j]
                f=g
                F33.append(f)
                F44.append(s-1)
            else:
              F33.append(F4[j])
              F44.append(0)

#This is the ending position for polynomial generation for Case 2.b of the paper

I=[]
M2=[]

for i0 in range(d*k+1):
    for i1 in range(k+1):
        for i2 in range(k+1):
            for i3 in range(k+1):
                if(i1+i2+i3<=d*k):
                    I.append([i0,i1,i2,i3])
                    M2.append(z0^i0*z1^i1*z2^i2*z3^i3)

F=[]


for i in range(len(I)):
    i0=I[i][0]
    i1=I[i][1]
    i2=I[i][2]
    i3=I[i][3]

    m=max(i1,i2)
    m=max(m,i3)
    if(m==0):
        f=z0^i0*P^(d*k)
        F.append(f)
        continue

    B=[]
    C=[z1,z2,z3]
    V=[i1,i2,i3]
    S=[]
    for l in range(m):
     B1=[]
     for j in range(n):
       if(V[j]>=(m-l)):
           B1.append(j)
     S.append(len(B1))
     B.append(B1)
    S3=[]
    for l in range(len(B)):
        ss=1
        for j in range(len(B[l])):
            ss=ss*C[B[l][j]]
```

```
        S3.append(ss)


    if(i0>=(i1+i2+i3)):
        f=z0^(i0-i1-i2-i3)
        for l in range(len(B)):
          for j in range(len(B[l])):
            f=f*G[B[l][j]]

        g=0
        S1=f.monomials()
        for j in range(len(S1)):
            aj=f.coefficient(S1[j])
            aj=aj(0,0,0,0)
            aj=ZZ(aj)
            aj=ZZ(aj%P^(i1+i2+i3))
            g=g+aj*S1[j]
        F.append(f*P^(d*k-i1-i2-i3))



    else:
        s1=0
        S2=[]
        G1=[]
        for j in range(len(S)):
            s1=s1+S[j]
            if(s1>i0):
                s1=s1-S[j]
                break
            S2.append(S[j])
        l=len(S2)
        for j in range(l):
            G1.append(z0^S2[j]*S3[j])
        G1.append(z0^(i0-s1)*S3[l])
        for j in range(l+1, len(S3)):
            G1.append(S3[j])

        f=1
        ff=0
        for j in range(len(G1)):
            for j1 in range(len(M)):
                if(G1[j]==M[j1]):
                    f=f*F33[j1]
                    ff=ff+F44[j1]


        g=0
        S1=f.monomials()
        for j in range(len(S1)):
            aj=f.coefficient(S1[j])
            aj=aj(0,0,0,0)
            aj=ZZ(aj)
            aj=ZZ(aj%P^ff)
            g=g+aj*S1[j]


        F.append(f*P^(d*k-ff))
```

```
rr=len(F)

print 'Dimension of the Lattice=', rr
MM=Matrix(ZZ,rr,rr,range(rr*rr))
for i in range(rr):
    f=F[i]
    f=f(z0*U,z1*U,z2*U,z3*U)
    for j in range(len(M2)):
        cij=f.coefficient(M2[j])
        cij=cij(0,0,0,0)
        MM[i,j]=cij
tt=cputime()



MM=MM.LLL()
print 'LLL Time', cputime(tt)
M3=[]
for j in range(60):
  f1=0
  for i in range(rr):
    f1=f1+(MM[j][i]/M2[i](U,U,U,U))*M2[i]

  M3.append(f1)


I=(M3)*R
tt=cputime()
B=I.groebner_basis()
print 'Groebner Basis time', cputime(tt)
print B[0],E[0]
```