Noname manuscript No.
(will be inserted by the editor)

Modular Inversion Hidden Number Problem – Correction and Improvements

Santanu Sarkar

Received: date / Accepted: date

Abstract The Modular Inversion Hidden Number Problem (MIHNP) was introduced by Boneh, Halevi and Howgrave-Graham in Asiacrypt 2001 (BHH'01). They provided two heuristics - in Method I, two-third of the output bits are required to solve the problem, whereas the more efficient heuristic (Method II) requires only one-third of the bits of the output. After more than a decade, here we identify that the claim in Method II is actually not correct and a detailed calculation justifies that this method too requires two-third of the bits of the output, contrary to the claim in BHH'01[1]. Further, we show that using the same relations as in Boneh et al., one can reconstruct the lattice so that the problem can be heuristically solved by the knowledge of five-eighth of the bits. Finally, we could accumulate additional relations to solve the problem heuristically with only half of the output bits in asymptotic sense. Experimental results support the claim corresponding to our heuristics.

Keywords Coppersmith's techniques · Hidden Number Problem · LLL algorithm · Modular Inversion.

1 Introduction

1.1 Background

In many instances, complexity assumptions are used to prove the security of a cryptographic protocol. For example, Chosen Ciphertext Secure Encryption [3] is based on Decision Diffie-Hellman assumption (DDH). This DDH assumption is also used to generate pseudo random functions [22]. Construction of secure hash-and-sign signature [14] relies on strong RSA assumption. Further, security of the public key encryption schemes proposed in [11] are based on Paillier's Decision Composite Residuosity (DCR) [23] and Quadratic Residuosity (QR) assumptions.

Hidden Number Problems (HNPs) were introduced in [6] by Boneh and Venkatesan. They used these to prove that computing the most significant bits of the secret key from the public keys of participants, in a Diffie-Hellman key-exchange protocol, is as hard as computing the

S. Sarkar
Department of Mathematics, Indian Institute of Technology Madras, Chennai 600 036, India
E-mail: sarkar.santanu.bir@gmail.com

[1] This work was initially submitted to Eurocrypt 2014 on 15th October, 2013.

secret key itself. The work [6] has originated a whole direction of research and HNP has since been exploited in a wide spectrum of applications like an attack on weak versions of the Digital Signature Algorithm (DSA) [18] and appeared in a number of questions, related and unrelated to cryptography (see [25] for a survey of relevant results and also [1] for some recent developments). The general idea of HNPs [6] is as follows.

Let $\alpha \in \mathbb{Z}_p$ be the (hidden) secret. Now consider n elements $x_1, \ldots, x_n \in \mathbb{Z}_p^*$, chosen independently and uniformly at random. The attacker is given n pairs $\big\{(x_i, \mathrm{MSB}_k(\alpha x_i \bmod p))\big\}_{i=1}^{n}$ for some $k > 0$ (here $\mathrm{MSB}_k(z \bmod p)$ refers to the k most significant bits of z). The target is to obtain α efficiently.

1.2 Modular Inversion Hidden Number Problem (MIHNP): Our Concern

A closely related class to HNPs known as Modular Inverse Hidden Number Problems (MIHNPs) was introduced and studied in [7] by Boneh, Halevi and Howgrave-Graham and certain important cryptographic applications in pseudo-random generator and MAC had been proposed. The exact problem (MIHNP) is as follows.

For a sufficiently large m-bit prime p, consider n elements $x_1, \ldots, x_n \in \mathbb{Z}_p^*$, chosen independently and uniformly at random and a secret $\alpha \in \mathbb{Z}_p$. The question is, given n pairs $\Big\{ \big(x_i, \mathrm{MSB}_k((\alpha + x_i)^{-1} \bmod p)\big) \Big\}_{i=1}^{n}$ for some $k > 0$, whether it is possible to recover α. The δ-MIHNP *assumption* states that there is no polynomial time algorithm for MIHNP whenever $k < \delta m$.

One of the most fundamental cryptographic primitives is the Pseudorandom Generator (PRG). It has numerous applications in cryptography. Several number theoretic PRGs have been analysed over the last few decades [27, 8, 4, 15, 26, 2]. The problem under consideration, MIHNP, can also be used to construct an efficient pseudorandom generator. Take an m bit prime p and two positive integers n and k. The input of the generator is a sequence $\{\alpha, x_1, \ldots, x_n\}$ of $n + 1$ elements in $\mathbb{Z}_p$. The output is

$$\Big\{x_1, \ldots, x_n, \mathrm{MSB}_k((\alpha + x_1)^{-1} \bmod p), \ldots, \mathrm{MSB}_k((\alpha + x_n)^{-1} \bmod p)\Big\}.$$

Note that although this is a pseudorandom number generator, this is not a pseudorandom bit generator. However, there are standard techniques [19, 28] to convert this into a pseudorandom bit generator as well.

In [7], two polynomial time heuristics were presented to solve MIHNP, provided that k is sufficiently large. Their first heuristic (a linear approach [7, Section 3.1] that we refer here as Method I) works only if more than $\frac{2}{3}$ portion of most significant bits of $(\alpha + x_i)^{-1} \bmod p$'s are given, i.e., $k > \frac{2m}{3}$. Importantly, the second heuristic (where multiples are used [7, Section 3.2] that we refer here as Method II) claimed knowledge of significantly fewer bits which is $k > \frac{m}{3}$ only.

Ling, Shparlinski, Steinfeld and Wang [21] provided a rigorous probabilistic polynomial time algorithm for MIHNP. The work [21] could match one of the heuristics (Method I) of Boneh et al. [7], where one requires two-third of the bits of the output to solve the problem. However, Ling et al. [21] 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. Recently, Xu et al. [29] revisit the work of [7]. They give involved lattice construction to obtain the same asymptotic bound of [7].

In this direction, we were motivated to look at Method II of Boneh et al. [7]. A detailed study of this pointed out an important flaw in the analysis at [7, Page 44], where it had been written that

$$\text{weight(relations)} \leq d \cdot (d+1)\binom{n}{d}(1+o(1)).$$

However, we note that this would actually be

$$\text{weight(relations)} = \frac{d\cdot(d+1)}{2}\binom{n}{d}(1+o(1)).$$

and correspondingly, the results will be modified. In particular, the performance of Method II will be same as Method I in [7] and thus, $k > \frac{2m}{3}$ output bits need to be known instead of $k > \frac{m}{3}$.

1.3 Contribution of this paper

In this paper, we study hardness of the δ-MIHNP assumption in detail. In particular, the contribution of this paper is three-fold.

- We revisit the analysis of the MIHNP as in [7] and observe a flaw. The proposed heuristic (Method II) in [7] that claims to solve the MIHNP when $k > \frac{m}{3}$ is not correct. More precisely, we show that the approach of [7] only works when $k > \frac{2m}{3}$ and does not work in the range of $\frac{m}{3} < k < \frac{2m}{3}$.
- We then use same relations as in [7] with a different lattice construction and obtain a new heuristic that solves the MIHNP when $k > \frac{5}{8}m$.
- Finally in Section 4, we present two lattice based heuristics for MIHNP using two different kinds of lattices based on same relations. We exploit additional relations over those mentioned in [7]. For both these heuristics, we require about half of the bits of the output, i.e., $k > \frac{m}{2}$.

2 Preliminaries

Consider w many linearly independent vectors $b_1, \ldots, b_w \in \mathbb{R}^n$. The set

$$L = \left\{ b : b = c_1 b_1 + \cdots + c_w b_w, c_1, \ldots, c_w \in \mathbb{Z} \right\}$$

is called an w dimensional lattice with basis $B = \{b_1, \ldots, b_w\}$. A lattice is of full rank when $w = n$ and in this paper we only use such lattices. The determinant of L is defined as $\det(L) = \det(B)$, where B is a $w \times w$ matrix. When $b_1, \ldots, b_w \in \mathbb{Z}^n$, lattice is called integer lattice.

In 1982, Lenstra, Lenstra and Lovász [20] defined LLL reduced basis of a lattice and proposed a polynomial time algorithm (famous as LLL algorithm) to obtain such a basis. Given a basis $b_1, \ldots, b_\omega$ of a lattice L, LLL algorithm can find a reduced basis $u_1, \ldots, u_\omega$ with

$$||u_1|| \leq ||u_2|| \leq \cdots \leq ||u_i|| \leq 2^{\frac{\omega(\omega-1)}{4(\omega+1-i)}} \det(L)^{\frac{1}{\omega+1-i}}, \text{ for } i = 1, \ldots, \omega.$$

In [12], Coppersmith formulated seminal ideas to find small roots of a modular polynomial in a single variable and also for polynomials in two variables over the integers. One may also refer to the works of Coron [9,10] in this context. These deterministic techniques have many important consequences in cryptography. The idea of [12] can also be extended to more than two variables, but the method becomes a heuristic in such cases. The following result due to Howgrave-Graham [17] gives a sufficient condition under which modular roots become the roots over integers for polynomials over two or more variables.

Theorem 1 *Let $g(x_1, \ldots, x_v)$ be a polynomial with integer coefficients which is a sum of ω many monomials. Suppose that*

1. $g(y_1, \ldots, y_v) \equiv 0 \bmod R$ *for some positive integer* R *and* $|y_1| < Y_1, \ldots, |y_v| < Y_v$.
2. $\parallel g(x_1 Y_1, \ldots, x_v Y_v) \parallel < \frac{R}{\sqrt{\omega}}$,

Then $g(x_1, x_2, \ldots, x_v) = 0$ *holds over integers.*

Towards solving our modular polynomials, we use the idea of [17]. Our technique works in practice as noted from the experiments we perform, but this may not always happen theoretically as we are working with heuristics only. Thus our approach relies on the following heuristic assumption like many other lattice based approaches [13, 16]:

Assumption 1 *In this work, the lattice based constructions yield algebraically independent polynomials and the common roots of the polynomials involved can be computed using Gröbner basis technique.*

3 Revisiting the work on MIHNP [7]

Suppose attacker knows p, x_i and $b_i = \text{MSB}_k((\alpha + x_i)^{-1} \bmod p)$ for $0 \leq i \leq n$. Hence we can write:

$$(\alpha + x_i)(b_i + \epsilon_i) = 1 \bmod p,$$

for $0 \leq i \leq n$. It is clear that if any ϵ_i is known, one can easily find α as $\alpha = (b_i + \epsilon_i)^{-1} - x_i \bmod p$. Also note that $\epsilon_i \approx 2^{m-k}$, since p is an m-bit integer and k many MSBs of b_i and $(\alpha + x_i)^{-1} \bmod p$ are the same. Now eliminating α from $(\alpha + x_0)(b_0 + \epsilon_0) = 1 \bmod p$ and $(\alpha + x_i)(b_i + \epsilon_i) = 1 \bmod p$, for some i, we have

$$\begin{aligned}\big(x_i - x_0\big)\epsilon_0\epsilon_i + \big(b_0(x_i - x_0) + 1\big)\epsilon_i + \big(b_i(x_i - x_0) - 1\big)\epsilon_0 \\ + b_0 b_i (x_i - x_0) + b_i - b_0 \equiv 0 \bmod p,\end{aligned}$$

for $1 \leq i \leq n$. Here ϵ_0 and ϵ_i are unknowns. Hence we need to solve $f_i(\epsilon_0, \epsilon_i) = 0 \bmod p$, where $f_i(\epsilon_0, \epsilon_i) = A_i \epsilon_0 \epsilon_i + B_i \epsilon_i + C_i \epsilon_0 + D_i,\ A_i = x_i - x_0,\ B_i = b_0(x_i - x_0) + 1,\ C_i = b_i(x_i - x_0) - 1$ and $D_i = b_0 b_i (x_i - x_0) + b_i - b_0$ for $1 \leq i \leq n$. To solve these polynomials two methods namely linear and non-linear approaches were proposed in [7].

3.1 Linear Approach [7]

Since $f_i(\epsilon_0, \epsilon_i) \equiv 0 \bmod p$, we can write $A_i \epsilon_0 \epsilon_i + B_i \epsilon_i + C_i \epsilon_0 + D_i + p k_i = 0$ for $1 \leq i \leq n$. Using these relations, construct a lattice M of dimension $3n + 2$ of the form $M = \begin{pmatrix} E & R \\ 0 & P \end{pmatrix}$, where E and P are diagonal matrices of dimensions $(2n + 2) \times (2n + 2)$ and $n \times n$ respectively. Further, R is a $(2n + 2) \times n$ matrix. The first $2n + 2$ rows of M are associated with one of the terms of $f_i(\epsilon_0, \epsilon_i)$ and each of the last n columns are associated with one of the n relations. The matrix E is a diagonal matrix whose diagonal elements correspond to the upper bound of each monomial. Hence the matrix M is of the form:

$$\begin{array}{cl}
 & \underline{\text{Row corresponds to}} \\
\begin{pmatrix}
1 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & D_1 & \cdots & D_n \\
0 & 2^{k-m} & 0 & \cdots & 0 & 0 & \cdots & 0 & C_1 & \cdots & C_n \\
0 & 0 & 2^{k-m} & \cdots & 0 & 0 & \cdots & 0 & B_1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 2^{k-m} & 0 & \cdots & 0 & 0 & \cdots & B_n \\
0 & 0 & 0 & \cdots & 0 & 2^{2k-2m} & \cdots & 0 & A_1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 2^{2k-2m} & 0 & \cdots & A_n \\
0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & p & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \\
0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & p
\end{pmatrix}
&
\begin{array}{l}
\cdots\cdots\cdots\ 1 \\
\cdots\cdots\cdots\ \epsilon_0 \\
\cdots\cdots\cdots\ \epsilon_1 \\
\qquad\qquad\ \vdots \\
\cdots\cdots\cdots\ \epsilon_n \\
\cdots\cdots\cdots\ \epsilon_0\epsilon_1 \\
\qquad\qquad\ \vdots \\
\cdots\cdots\cdots\ \epsilon_0\epsilon_n \\
 \\
 \\

\end{array}
\end{array}$$

Now let us consider the vector $v = (1, \epsilon_0, \ldots, \epsilon_n, \epsilon_0\epsilon_1, \ldots, \epsilon_0\epsilon_n, k_1, \ldots, k_n)$. Note that

$$s = v \cdot M = \left(1, \frac{\epsilon_0}{2^{m-k}}, \ldots, \frac{\epsilon_n}{2^{m-k}}, \frac{\epsilon_0\epsilon_1}{2^{2(m-k)}}, \ldots, \frac{\epsilon_0\epsilon_n}{2^{2(m-k)}}, 0, \ldots, 0\right).$$

Thus s is a lattice point and $||s|| < \sqrt{2n+2}$. Also $\det(M) = \frac{p^n}{2^{(m-k)(3n+1)}}$. Hence using Gaussian heuristic, one can expect that s is the shortest vector when

$$\sqrt{2n+2} \ll \sqrt{3n+2}\left(\frac{p^n}{2^{(m-k)(3n+1)}}\right)^{\frac{1}{3n+2}}.$$

This is equivalent to $\det(M) \gg \left(\sqrt{\frac{2n+2}{3n+2}}\right)^{3n+2}$. Hence the required condition can be taken as $\det(M) \gg 1$.

Now using LLL algorithm [20], one can expect to find v whenever this condition $\det(M) \gg 1$ is satisfied. Since $p \approx 2^m$, the condition $\det(M) \gg 1$ yields $k \gg \frac{2mn+m}{3n+1}$. So asymptotically this bound becomes $k \gg \frac{2m}{3}$. Hence the linear approach of [7] works only when the number of known most significant bits is more than $\frac{2m}{3}$.

3.2 Non Linear Approach [7]: Flaw and Correction

This approach is based on the idea of Coppersmith [12]. There are n relations $f_1, \ldots, f_n$. Now we will use the products of these relations. For example, if one can use the relation $f_1f_2 = 0 \bmod p^2$ in the lattice construction, the relation $\epsilon_1 f_2 = 0 \bmod p$ can also be used for free as each term of $\epsilon_1 f_2$ is also present in f_1f_2. In this case, one can construct a lattice L from the relations $f_1f_2 = 0 \bmod p^2, \epsilon_1 f_2 = 0 \bmod p, \epsilon_0 f_2 = 0 \bmod p, f_2 = 0 \bmod p, \epsilon_0 f_1 = 0 \bmod p, \epsilon_2 f_1 = 0 \bmod p$. Let $\Delta = 2^{k-m}$. Then the matrix M is of the form:

$$\left(\begin{array}{cccccccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & 0 & 0 & - & 0 & 0 \\
0 & \Delta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & 0 & - & - & - & 0 \\
0 & 0 & \Delta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & - & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \Delta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & 0 & 0 & - & 0 & - \\
0 & 0 & 0 & 0 & \Delta^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & 0 & - & 0 & - & 0 \\
0 & 0 & 0 & 0 & 0 & \Delta^2 & 0 & 0 & 0 & 0 & 0 & 0 & - & - & 0 & 0 & - & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \Delta^2 & 0 & 0 & 0 & 0 & 0 & - & 0 & - & - & 0 & - \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta^2 & 0 & 0 & 0 & 0 & - & - & 0 & 0 & 0 & - \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta^3 & 0 & 0 & 0 & - & 0 & 0 & 0 & - & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta^3 & 0 & 0 & - & - & 0 & 0 & 0 & - \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta^3 & 0 & - & 0 & - & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta^4 & - & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p^2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p
\end{array}\right)
\begin{array}{l}
\underline{\text{Row corresponds to}} \\
\cdots\cdots\cdots\cdots \; 1 \\
\cdots\cdots\cdots\cdots \; \epsilon_0 \\
\cdots\cdots\cdots\cdots \; \epsilon_1 \\
\cdots\cdots\cdots \; \epsilon_2 \\
\cdots\cdots\cdots \; \epsilon_0^2 \\
\cdots\cdots\cdots \; \epsilon_0\epsilon_1 \\
\cdots\cdots\cdots \; \epsilon_0\epsilon_2 \\
\cdots\cdots\cdots \; \epsilon_1\epsilon_2 \\
\cdots\cdots\cdots \; \epsilon_0^2\epsilon_1 \\
\cdots\cdots\cdots \; \epsilon_0\epsilon_1\epsilon_2 \\
\cdots\cdots\cdots \; \epsilon_0^2\epsilon_2 \\
\cdots\cdots\cdots \; \epsilon_0^2\epsilon_1\epsilon_2 \\
\\ \\ \\ \\ \\ \\
\end{array}$$

In this case $\det(M) = p^7\Delta^{24}$. As in Section 3.1, using Coppersmith's idea, it is not difficult to show that one can find ϵ_i for $0 \le i \le 2$ by LLL algorithm over the matrix M if $\det(M) \gg 1$.

Now take a positive integer $d < n$. In [7], the authors described the process of adding relations to the lattice in different phases. In phase d, one adds the relations that are obtained by multiplying up to d many original relations. Hence the new relations look like $f_{i_1} \cdots f_{i_d} = 0 \bmod p^d$ for some $0 < i_1, \ldots, i_d \le n$ in phase d. Now one can construct a lattice L from these polynomials. For example when $n = 3$ and $d = 2$, the relations are given as

$$\Big\{ f_1 f_2 = 0 \bmod p^2, f_1 f_3 = 0 \bmod p^2, f_2 f_3 = 0 \bmod p^2, \epsilon_0 f_1 = 0 \bmod p,$$
$$\epsilon_0 f_2 = 0 \bmod p, \epsilon_1 f_2 = 0 \bmod p, \epsilon_0 f_3 = 0 \bmod p, \epsilon_1 f_3 = 0 \bmod p,$$
$$\epsilon_2 f_3 = 0 \bmod p, f_1 = 0 \bmod p, f_2 = 0 \bmod p, f_3 = 0 \bmod p \Big\}.$$

In [7], two phrases, namely "weight of terms" and "weight of relations" are introduced. The weight of a term is its degree. For example, weight of $\epsilon_0\epsilon_1$ is 2. The weight of a relation is the number of original relations f_i that are multiplied. As for an example, $f_1 f_2 = 0 \bmod p^2$ has the weight 2. In the lattice constructions of [7, Section 3.2], all terms can be expressed as $\epsilon_0^{r_0}\epsilon_1^{r_1} \cdots \epsilon_n^{r_n}$, where $0 \le r_0 \le d$ and $0 \le r_1, \ldots, r_n \le 1$ with $r_1 + \cdots + r_n \le d$. Hence by taking $r_1 + \ldots + r_n = j$, the total weight W_T of all the terms is

$$\sum_{\substack{r_1,\ldots,r_n=0 \\ r_1+\ldots+r_n \le d}}^{1} \sum_{r_0=0}^{d} (r_0 + r_1 + \cdots + r_n) = \sum_{j=0}^{d}\sum_{r_0=0}^{d} \binom{n}{j}(j + r_0)$$
$$= \sum_{j=0}^{d} \binom{n}{j}(d+1)\left(j + \frac{d}{2}\right) = \binom{n}{d}(d+1)\frac{3d}{2}\Big(1 + o(1)\Big),$$

when $d \ll n$. This exactly matches with the calculation provided in [7].

3.2.1 The Flaw and the Correction

Next we calculate the total weight of relations. Consider the relation $F = f_1^{t_1} \cdots f_n^{t_n}$ where $0 \le t_i \le 1$ with $i \in [1, n]$ and $\sum_{i=1}^n t_i \le d$. The weight of F is $\displaystyle\sum_{i=1}^n t_i$ and the corresponding term is $\epsilon_0^{t_1+\ldots+t_n}\epsilon_1^{t_1}\cdots\epsilon_n^{t_n}$ since the term corresponding to f_i is $\epsilon_0\epsilon_i$. The weight of the relation which corresponds to the term $\epsilon_0^{r_0}\epsilon_1^{r_1}\cdots\epsilon_n^{r_n}$ is $\min(r_0, r_1 + r_2 + \ldots + r_n)$. Hence total weight W_R of all relations is

$$\sum_{\substack{r_1,\ldots,r_n=0\\ r_1+\ldots+r_n\le d}}^{1} \sum_{r_0=0}^{d} \min(r_0, r_1 + \cdots + r_n) = \sum_{j=0}^{d}\sum_{r_0=0}^{d} \binom{n}{j} \min(r_0, j)$$
$$= \sum_{j=0}^{d} \binom{n}{j} \left(\sum_{r_0=0}^{j} r_0 + \sum_{r_0=j+1}^{d} j \right) = \sum_{j=0}^{d} \binom{n}{j} \left(\frac{j(j+1)}{2} + j(d-j) \right)$$
$$= \frac{d\cdot(d+1)}{2} \binom{n}{d} (1 + o(1)),$$

when $d \ll n$. Let us refer to [7, Page 44] for the comment "In fact, it is possible to show that this bound is tight, and the total weight of the relations that we get is at least $d^2\binom{n}{d}$." That is, in [7], it has been claimed that the total weight of relations W_R is upper bounded by $d \cdot (d+1)\binom{n}{d}(1 + o(1))$.

Although there are approximately $(d + 1)\binom{n}{d}$ terms, many corresponding relations have weight less than d. For example, the relation corresponding to the term $\epsilon_1^{r_1}\cdots\epsilon_n^{r_n}$ has weight zero. Hence we cannot approximate the total weight of relations W_R by $d\cdot(d+1)\binom{n}{d}$; instead it is $\frac{d\cdot(d+1)}{2}\binom{n}{d}$. This is the reason, one should not obtain any further improvement by considering this non linear approach over the linear approach in [7].

When $p \approx 2^m$ and $\epsilon_0 \approx \cdots \approx \epsilon_n \approx 2^{m-k}$, the determinant of the lattice will be approximately $\det(L) \approx 2^{m\cdot W_R-(m-k)\cdot W_T}$. Now as in Section 3.1, using Coppersmith's technique one can find ϵ_i if $\det(L) \gg 1$ for $0 \le i \le n$. Now the condition $\det(L) \gg 1$ gives $k \gg m(1 - \frac{W_R}{W_T})$. Since $\frac{W_R}{W_T} \approx \frac{1}{3}$, the condition is $k \gg \frac{2m}{3}$.

Given that the non linear method of [7] does not solve the MIHNP when $\frac{m}{3} < k < \frac{2m}{3}$, we need to obtain improved results. Considering the same relations as in [7], but with a different lattice construction, we could show that MIHNP can be solved heuristically when $k > \frac{5m}{8}$. As we could provide further improvements over this, we put this initial improvement in Appendix A for interested readers. In the following section, we go for more involved techniques to show that MIHNP can indeed be solved heuristically when $k > \frac{m}{2}$. To the best of our knowledge, this is the best result in this area given that the claim of [7] using non linear method is flawed.

4 Generate more relations: Improved attack on MIHNP

Previously, only n many relations have been used to solve the problem and we go for more relations for an improved attack.

Consider two relations $(\alpha + x_i)(b_i + \epsilon_i) = 1 \bmod p$ and $(\alpha + x_j)(b_j + \epsilon_j) = 1 \bmod p$ for $0 \le i < j \le n$. Now eliminating α from these two relations, we obtain

$$(x_i - x_j)\epsilon_i\epsilon_j + (x_i b_j - x_j b_j + 1)\epsilon_i + (x_i b_i - x_j b_i - 1)\epsilon_j$$
$$+(x_i b_i b_j - x_j b_i b_j + b_i - b_j) \equiv 0 \bmod p.$$

Hence we have total $\binom{n+1}{2}$ relations of the form

$$f_{ij} = A_{ij}\epsilon_i\epsilon_j + B_{ij}\epsilon_i + C_{ij}\epsilon_j + D_{ij} \equiv 0 \bmod p,$$

where $A_{ij} = x_i - x_j,\ B_{ij} = x_i b_j - x_j b_j + 1,\ C_{ij} = x_i b_i - x_j b_i - 1$ and $D_{ij} = x_i b_i b_j - x_j b_i b_j + b_i - b_j$. From f_{ij} we obtain the relations of the form $g_{i,j} = A_{ij}^{-1} f_{ij} \bmod p$, where the coefficient of $\epsilon_i\epsilon_j$ in $g_{i,j}$ is 1. It is immediate that $g_{i,j}(\epsilon_i, \epsilon_j) \equiv 0 \bmod p$. We will use all these relations, along with the idea of Howgrave-Graham [17], to obtain an improved bound.

4.1 Our Method I

From the knowledge of $\{x_i, \mathrm{MSB}_k((\alpha + x_i)^{-1} \bmod p)\}$, we construct $\binom{n+1}{2}$ relations $g_{i,j}$ such that $g_{i,j}(\epsilon_i, \epsilon_j) \equiv 0 \bmod p$ for $0 \le i < j \le n$. Suppose that $\epsilon_i \le 2^{m-k}$ for $0 \le i \le n+1$ and $Z = 2^{m-k}$. Consider the following set of polynomials

$$\mathcal{P} = \left\{ g_{0,1}, \ldots, g_{0,n}, \ldots, g_{n-1,n}, \epsilon_0 p, \ldots, \epsilon_n p, p \right\}.$$

Note that $h(\epsilon_0, \ldots, \epsilon_n) = 0 \bmod p$ for each $h \in \mathcal{P}$.

Now construct a lattice L using the coefficient vectors of $h(\epsilon_0 Z, \ldots, \epsilon_n Z)$ for each $h \in \mathcal{P}$. Then the matrix M, corresponding to L, is of the form

$$\begin{array}{cccccccccc}
\epsilon_0\epsilon_1 & \ldots & \epsilon_0\epsilon_n & \ldots & \epsilon_{n-1}\epsilon_n & \epsilon_0 & \epsilon_1 & \ldots & \epsilon_n & 1 \\
\end{array}$$
$$\begin{pmatrix}
Z^2 & \ldots & 0 & \ldots & 0 & - & - & \ldots & 0 & - \\
\vdots & \ddots & \vdots & \vdots & \vdots & & \vdots & \ddots & \vdots & \vdots \\
0 & \ldots & Z^2 & \ldots & 0 & - & 0 & \ldots & - & - \\
\vdots & \vdots & \vdots & \ddots & \vdots & & \vdots & \ddots & \vdots & \vdots \\
0 & \ldots & 0 & \ldots & Z^2 & 0 & 0 & \ldots & - & - \\
0 & \ldots & 0 & \ldots & 0 & Zp & 0 & \ldots & 0 & 0 \\
0 & \vdots & 0 & \vdots & 0 & 0 & Zp & \ddots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \ldots & 0 & \ldots & 0 & 0 & \vdots & \ldots & Zp & 0 \\
0 & \ldots & 0 & \ldots & 0 & 0 & \vdots & \ldots & 0 & p
\end{pmatrix}$$

Here '$-$' indicates non zero element. The dimension of L is $\omega = 2 + n + \binom{n+1}{2}$. Also $\det(L) = Z^2 \cdots Z^2 \cdot Z \cdots Z \cdot p^{2+n} = Z^{(n+1)^2} p^{2+n}$. By the property of LLL algorithm and Howgrave-Graham's theorem (Theorem 1), if $2^{\frac{\omega(\omega-1)}{4(\omega-n)}} \left(\det(L)\right)^{\frac{1}{\omega-n}} < \frac{p}{\sqrt{\omega}}$, after lattice reduction we get $n+1$ polynomials which contain the root $(\epsilon_0, \ldots, \epsilon_n)$ over integers. Since the terms $2^{\frac{\omega(\omega-1)}{4(\omega-n)}}$ and $\sqrt{\omega}$ are much smaller than $\det(L)$ and p, the required condition reduces to $\det(L) < p^{\omega - n - o(1)}$.

Now $p \approx 2^m, \omega = 2 + n + \binom{n+1}{2}$ and $Z = 2^{m-k}$. Also

$$\det(L) = Z^{(n+1)^2} p^{2+n} \approx 2^{(m-k)(n+1)^2} 2^{m(2+n)} = 2^{(m-k)(n+1)^2 + m(2+n)}$$

and $p^{\omega-n-o(1)} = p^{2+\binom{n+1}{2}-o(1)} \approx 2^{m\left(\frac{n(n+1)}{2}+2-o(1)\right)}$.

Thus the condition $\det(L) < p^{\omega-n-o(1)}$ implies

$$(m-k)(n+1)^2 + m(2+n) < m\left(\frac{n(n+1)}{2} + 2 - o(1)\right).$$

So we get $k > \big(\frac{1}{2} + \frac{3n+1}{2(n+1)^2} + o(1)\big)m$.

So when $k > \big(\frac{1}{2} + \frac{3n+1}{2(n+1)^2} + o(1)\big)m$, after lattice reduction we have $n+1$ polynomials $f_1, \ldots, f_{n+1}$ such that $f_i(\epsilon_0, \ldots, \epsilon_n) = 0$ for $1 \le i \le n+1$. Now under Assumption 1, we can find $\epsilon_0, \ldots, \epsilon_n$ from $f_1, \ldots, f_{n+1}$. If any ϵ_i is known, one can easily find α as $\alpha = (b_i + \epsilon_i)^{-1} - x_i \bmod p$ for $i \in [0, n]$. Hence we have the following result.

Theorem 2 *Let p be an m-bit prime. Let $\alpha \in \mathbb{Z}_p$ be hidden. Consider that $n+1$ pairs $(x_i, MSB_k(\alpha + x_i)^{-1} \bmod p)$ are given for random $x_0, x_1, \ldots, x_n \in \mathbb{Z}_p$. Then one can find α under Assumption 1 in polynomial time if $k > \big(\frac{1}{2} + \frac{3n+1}{2(n+1)^2} + o(1)\big)m$.*

4.2 Our Method II

Asymptotically, when $n \to \infty$, one can solve MIHNP if $k \ge \frac{m}{2}$ using our Method I. Now we will present another method which improves the bound $k > \big(\frac{1}{2} + \frac{3n+1}{2(n+1)^2} + o(1)\big)m$ by using the broad idea of Coppersmith [12] with a set of polynomials. Solving such a system of equations using Coppersmith's method is a non-trivial task, as discussed throughly in [24]. For this purpose, we will generate more polynomials for lattice construction from the original polynomials $g_{i,j}$ discussed above. The generated polynomials satisfy certain conditions and are divisible by the polynomials of the form $\prod_{i,j} g_{i,j}^{s_{i,j}}$. The values of $s_{i,j}$ are chosen in a cleaver way to reduce the size of the determinant of the lattice constructed.

Choice of $s_{i,j}$ can be considered as the following combinatorial problem. Suppose we are given $n+1$ non-negative integers $(r_0, \ldots, r_n)$. Consider $\binom{n+1}{2}$ many monomials $\epsilon_{i,j} = \epsilon_i \epsilon_j$ for $0 \le i < j \le n$. Our problem is to find non-negative integers $s_{i,j}$ such that $\prod_{0 \le i < j \le n} \epsilon_{i,j}^{s_{i,j}}$ divides $\displaystyle\prod_{i=0}^{n} \epsilon_i^{r_i}$ and $\displaystyle\sum_{0 \le i < j \le n} s_{i,j}$ is as large as possible. This is purely an optimization problem.

Let us first discuss an algorithm for the generation of $s_{i,j}$. We will see in Theorem 3 that by choosing $s_{i,j}$ as given by this algorithm provides a better lower bound on k than in Theorem 2. We are not claiming our approach gives the optimal result but it helps to get a better lower bound on k than the existing works.

The input to our algorithm is an array $A = [r_0, \ldots, r_n]$ of length $n+1$, where $r_0, \ldots, r_n$ are non negative integers and the output is a sequence $\{s_{0,1}, \ldots, s_{n-1,n}\}$ of non negative integers of length $\binom{n+1}{2}$ and a non negative integer $s = \sum_{i,j} s_{i,j}$. We denote the number of zeros in A by $\mathcal{Z}(A)$.

For a sequence $\{r_0, r_1, \ldots, r_n\}$, let $\min_l(r_0, r_1, \ldots, r_n)$ be the l-th minimum of $\{r_0, \ldots, r_n\}$. Thus, for the sequence $\{0, 1, 0, 3\}$, we have $\min_1(0,1,0,3) = \min_2(0,1,0,3) = 0, \min_3(0,1,0,3) = 1$ and $\min_4(0,1,0,3) = 3$.

Now we will present our algorithm in Algorithm 1. In each iteration of the algorithm, if the condition of line 4 is true, then exactly two (positive) elements of the array are decreased by 1. Note that length of the array A in Algorithm 1 is $n+1$, and algorithm runs as long as it contains at least two positive integers. That is, the algorithm stops when the number $\mathcal{Z}(A)$ of zeros in A is more than $n-1$.

For a given sequence $\{r_0, r_1, \ldots, r_n\}$, we want to estimate the second output s of Algorithm 1. We have the following result in this direction. The proof of Lemma 1 is given in Appendix B.

Lemma 1 *The value of s returned by the Algorithm 1 satisfies*

$$s = \min_n(r_0, r_1, \ldots, r_n) + \frac{1}{2}\sum_{i=1}^{n-1} \min_i(r_0, r_1, \ldots, r_n) + E(n),$$

```
**Input**: $A = [r_0, \ldots, r_n]$.
**Input**: Zero sequence $\{s_{0,1}, \ldots, s_{n-1,n}\} = \{0, \ldots, 0\}$, $s = 0$.
**Output**: $\{s_{0,1}, \ldots, s_{n-1,n}\}$ and $s$
while $\Big(\mathcal{Z}(A) < n\Big)$ do
    for $i = 0$ to $n - 1$ do
        for $j = i + 1$ to $n$ do
            if $A[i] > 0$ & $A[j] > 0$ then
                $A[i] = A[i] - 1$ ;
                $A[j] = A[j] - 1$ ;
                $s_{i,j} = s_{i,j} + 1$ ;
                $s = s + 1$ ;
            end
        end
    end
end
```

Algorithm 1

with $|E(n)| < (n+1)^3$.

Let us start with the array $A = [23, 34, 65, 13]$ of length $n + 1 = 4$. Now consider the following iterations:

$$\overset{\downarrow}{[23},\ \overbrace{\overset{\downarrow}{34},\ 65,\ 13] \to [\overset{\downarrow}{22},\ 33,\ \overset{\downarrow}{65},\ 13] \to \cdots \to [\overset{\downarrow}{9},\ 21, 52,\ \overset{\downarrow}{1}]}^{27} \to [8,\ 21,\ 52,\ 0].$$

After 27 iterations, the array $[r_0, r_1, r_2, r_3] = [23,\ 34,\ 65,\ 13]$ becomes $[8,\ 21,\ 52,\ 0]$. Here $l = 27$ and $n = 3$, $n\left\lfloor \frac{l}{\binom{n+1}{2}} \right\rfloor = 3\lfloor \frac{27}{6} \rfloor = 12$. In this case, $\epsilon_0 = -3, \epsilon_1 = \epsilon_2 = \epsilon_3 = -1$. Hence $|\epsilon_i| < n + 1 = 4$ for $0 \le i \le 3$. The value of s of the algorithm for the inputs $[8, 21, 52, 0]$ and $[8, 21, 52]$ is same. Here the final value of s will be 52. In this example, $\min_1(23, 34, 65, 13) = 13, \min_2(23, 34, 65, 13) = 23, \min_3(23, 34, 65, 13) = 34$.

So we get $\min_3(23, 34, 65, 13) + \frac{1}{2}(\min_1(23, 34, 65, 13) + \min_2(23, 34, 65, 13)) = 34 + \frac{13+23}{2} = 52$, exactly matches with s. Now we will generate polynomials for lattice construction and use that lattice to find α. In this direction, we have the following result.

Theorem 3 *Let p be an m-bit prime. Let $\alpha \in \mathbb{Z}_p$ be hidden. Consider that $n+1$ pairs $(x_i, MSB_k(\alpha + x_i)^{-1} \bmod p)$ are given for random $x_0, x_1, \ldots, x_n \in \mathbb{Z}_p$. Then one can find α under Assumption 1 in time polynomial in m but exponential in n if $k > m\left(\frac{1}{2} + \frac{1}{(n+1)(n+2)} + o(1)\right)$.*

Proof Let $\epsilon_i \le 2^{m-k}$. Take $Z = 2^{m-k}$. Now for $0 \le r_0, \ldots, r_n \le nd$, define the polynomials

$$h_{r_0,\ldots,r_n} = g_{0,1}^{s_{0,1}} g_{0,2}^{s_{0,2}} \cdots g_{n-1,n}^{s_{n-1,n}} \prod_{k=0}^{n} \epsilon_k^{r_k - \sum_{l=0}^{k-1} s_{l,k} - \sum_{l=k+1}^{n} s_{k,l}} p^{d\binom{n+1}{2} - s},$$

where $\{s_{i,j}\}$ and s are obtained by Algorithm 1 on input $[r_0, \ldots, r_n]$ and d is a non negative integer.

In the lexicographic ordering of subscript $(r_0, \ldots, r_n)$, where each component $r_i \in [0, nd]$, the polynomial $h_{r_0,\ldots,r_n}$ introduces a new monomial $\epsilon_0^{r_0} \epsilon_1^{r_1} \cdots \epsilon_n^{r_n}$ which is not present in the previous polynomials. Thus the matrix corresponding to these polynomials will be lower-triangular and hence the matrix will be nonsingular. Note that $h_{r_0,\ldots,r_n}(\epsilon_0,\ldots,\epsilon_n) \equiv 0 \bmod p^{d\binom{n+1}{2}}$. Construct a lattice L using the coefficient vectors of $h_{r_0,\ldots,r_n}(\epsilon_0 Z,\ldots,\epsilon_n Z)$. The dimension of the lattice is $\omega = (nd+1)^{n+1}$. Now the determinant of L is

$$\det(L) = \prod_{r_0=0}^{nd} \cdots \prod_{r_n=0}^{nd} Z^{r_0}\cdots Z^{r_n} p^{\omega d\binom{n+1}{2} - \sum\limits_{r_0=0}^{nd}\cdots\sum\limits_{r_n=0}^{nd} s}.$$

Our condition is $2^{\frac{\omega(\omega-1)}{4(\omega-n)}}(\det(L))^{\frac{1}{\omega-1}} < \frac{1}{\sqrt{\omega}}p^{d\binom{n+1}{2}}$. Since the terms $2^{\frac{\omega(\omega-1)}{4(\omega-n)}}, \sqrt{\omega}$ are much smaller than $\det(L)$ and p, required condition can be written as $\det(L) < p^{d\binom{n+1}{2}(\omega-n)-o(1)}$. From this condition becomes

$$\prod_{r_0=0}^{nd} \cdots \prod_{r_n=0}^{nd} Z^{r_0}\cdots Z^{r_n} < p^{\sum_{r_0=0}^{nd}\cdots\sum_{r_n=0}^{nd} s - d\binom{n+1}{2}n - o(1)}.$$

Since $\sum_{r_0=0}^{nd}\cdots\sum_{r_n=0}^{nd}\Big(r_0+\ldots+r_n\Big) = (n+1)(nd+1)^n\sum_{r_0=0}^{nd} r_0$
$= \frac{n+1}{2}(nd)^{n+2} + o\big(d^{n+2}\big),$

we have $\prod_{r_0=0}^{nd}\cdots\prod_{r_n=0}^{nd} Z^{r_0}\cdots Z^{r_n} = Z^{\frac{n+1}{2}(nd)^{n+2}+o\left(d^{n+2}\right)}$.

Also from Lemma 1, $\displaystyle\sum_{r_0=0}^{nd}\cdots\sum_{r_n=0}^{nd} s = \sum_{r_0=0}^{nd}\cdots\sum_{r_n=0}^{nd}\Bigg(\min_n(r_0,\ldots,r_n)$
$+ \frac{\min_{n-1}(r_0,\ldots,r_n)+\cdots+\min_1(r_0,\ldots,r_n)}{2}\Bigg) + o\big(d^{n+2}\big),$

as $\sum_{r_0=0}^{nd}\cdots\sum_{r_n=0}^{nd}(n+1)^3 = (n+1)^3(nd+1)^{n+1} = o\big(d^{n+2}\big)$.

We will first calculate $\displaystyle\sum_{r_0=0}^{nd}\cdots\sum_{r_n=0}^{nd}\min_l(r_0,\ldots,r_n)$. Suppose $\min_l(r_0,\ldots,r_n) = k$, where $0 \le k \le nd$. Then we have a sequence $\{y_0,\ldots,y_n\} = \{r_0,\ldots,r_n\}$ such that

$$\underbrace{y_0 \le y_1 \le \cdots \le y_{l-2}}_{l-1} \le y_{l-1} = k \le \underbrace{y_l \le \cdots \le y_n}_{n-l+1}.$$

Total number of such sequences $\big\{r_0,\ldots,r_n\big\}$ for which $\min_l(r_0,\ldots,r_n) = k$ is equal to

$$\sum_{i=0}^{l-1}\sum_{j=0}^{n-l+1}\binom{n+1}{i+j+1}\binom{n-i-j}{l-1-i}k^{l-1-i}(nd-k)^{n-l+1-j}.$$

Here we assume $y_{l-2} = \cdots = y_{l-i-1} = k$ and $y_l = \cdots = y_{l+j-1} = k$.

Each such sequence will contribute $\min_l(r_0,\ldots,r_n) = k$ towards the sum. Hence,

$$\sum_{r_0=0}^{nd}\cdots\sum_{r_n=0}^{nd}\min_l(r_0,\ldots,r_n) = \textstyle\sum_{k=0}^{nd} k\sum_{i=0}^{l-1}\sum_{j=0}^{n-l+1}\binom{n+1}{i+j+1}\binom{n-i-j}{l-1-i}k^{l-1-i}(nd-k)^{n-l+1-j}$$
$$= \textstyle\sum_{k=0}^{nd} k\sum_{i=0}^{l-1}\sum_{j=0}^{n-l+1}\binom{n+1}{n+1-i-j}\binom{i+j}{i}k^i(nd-k)^j.$$

Now let $S(X) = \sum_{k=0}^{X} k^i (X-k)^j$ for some positive integer X. Then $\frac{S(X)}{X^{i+j+1}} = \sum_{k=0}^{X} \frac{1}{X} \left(\frac{k}{X}\right)^i \left(1 - \frac{k}{X}\right)^j$. Thus,

$$\lim_{X \to \infty} \frac{S(X)}{X^{i+j+1}} = \int_0^1 x^i (1-x)^j \, dx = B(i+1, j+1),$$

where $B(.,.)$ is the Beta function (see [5]). Hence $S(X) = \Theta\big(X^{i+j+1}\big)$.
So we have $\sum_{r_0=0}^{nd} \cdots \sum_{r_n=0}^{nd} \min_l(r_0, \ldots, r_n) = \sum_{k=0}^{nd} (n+1) \binom{n}{l-1} k^l (nd-k)^{n-l+1} + o\big(d^{n+2}\big)$.

Thus,

$$\begin{aligned}
\sum_{r_0=0}^{nd} \cdots \sum_{r_n=0}^{nd} s =& \frac{1}{2} \sum_{l=1}^{n-1} \sum_{k=0}^{nd} (n+1) \binom{n}{l-1} k^l (nd-k)^{n-l+1} + \sum_{k=0}^{nd} (n+1) \binom{n}{n-1} k^n (nd-k) + o\big(d^{n+2}\big) \\
=& \frac{n+1}{2} \sum_{k=0}^{nd} \sum_{l=0}^{n-2} \binom{n}{l} k^{l+1} (nd-k)^{n-l} + \frac{n}{n+2} (nd)^{n+2} + o\big(d^{n+2}\big) \\
=& \frac{n+1}{2} \sum_{k=0}^{nd} k \cdot \Big(\big(k + (nd-k)\big)^n - nk^{n-1}(nd-k) - k^n \Big) + \frac{n}{n+2} (nd)^{n+2} + o\big(d^{n+2}\big) \\
=& \frac{n+1}{2} (nd)^{n+2} \left(\frac{1}{2} - \frac{n}{n+1} + \frac{n-1}{n+2} \right) + \frac{n}{n+2} (nd)^{n+2} + o\big(d^{n+2}\big).
\end{aligned}$$

Now putting $Z = 2^{m-k}$ and $p \approx 2^m$ in $Z^{\frac{n+1}{2}(nd)^{n+2}} < p^{\sum_{r_0=0}^{nd} \cdots \sum_{r_n=0}^{nd} s - d\binom{n+1}{2} n - o(1)}$, required condition is $k > m\left(\frac{1}{2} + \frac{1}{(n+1)(n+2)} + o(1)\right)$. So when $k > \big(\frac{1}{2} + \frac{3n+1}{2(n+1)^2} + o(1)\big)m$, after lattice reduction we have $n+1$ polynomials $f_1, \ldots, f_{n+1}$ such that $f_i(\epsilon_0, \ldots, \epsilon_n) = 0$ for $1 \leq i \leq n+1$. Now under Assumption 1, we can find $\epsilon_0, \ldots, \epsilon_n$ from $f_1, \ldots, f_{n+1}$.
The running time of our algorithm is dominated by the runtime of the LLL algorithm, which is polynomial in the dimension of the lattice and in the bitsize of the entries. Since the lattice dimension in our case is exponential in n, the running time of our strategy is polynomial in m but exponential in n. □

As $\frac{1}{2} + \frac{1}{(n+1)(n+2)} < \frac{1}{2} + \frac{3n+1}{2(n+1)^2}$, our Method II provides better result than our Method I. Asymptotically both our methods approach the same limit $\frac{1}{2}$.

In Figure 1, we present the new attack region pictorially.

As we work with low lattice dimensions, the theoretical bounds presented in Theorem 3 may not be achieved. In Table 1, we present few numerical values of δ for different values of d, n, where $k \geq \delta m$.

4.3 Comparisons of our two methods and experimental results

We have implemented the code 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. In all our experiments, the polynomials obtained (after lattice reduction) satisfy the desired root over integers, we could successfully collect the root using Gröbner basis technique.

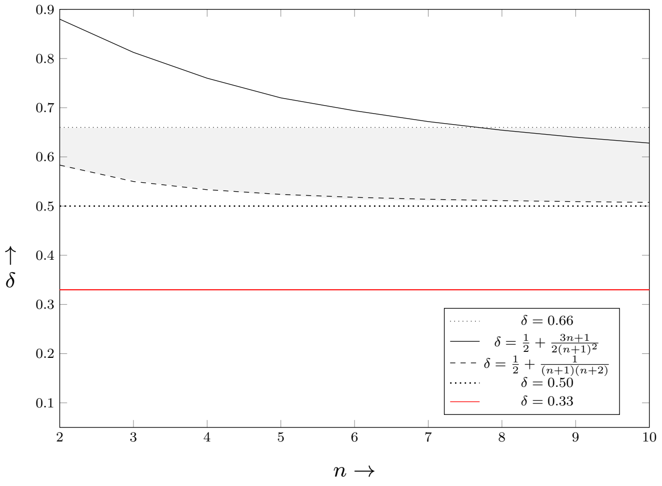

Fig. 1 New attack region. We need $k \geq \delta m$.

n	d	δ	Lattice Dimension	Asymptotic lower Bound of δ
2	8	0.588	4913	0.583
3	3	0.558	10000	0.550
4	3	0.539	371293	0.533
5	2	0.532	46656	0.524

Table 1 Numerical values of lower bound δm of k for different parameters.

First let us consider the case when $n = 2$. This case is interesting as our Method I does require $k > \left(\frac{1}{2} + \frac{3n+1}{2(n+1)^2}\right)m = \frac{8m}{9}$, whereas our Method II requires $k > m\left(\frac{1}{2} + \frac{1}{(n+1)(n+2)}\right) = \frac{7m}{12}$. However, in experiments we observe for $k > \frac{2m}{3}$ is sufficient for Method I. Gröbner basis means the time to extract the roots.

Method	Parameters	Theory	Experiment	Time in Seconds	
				LLL Algorithm	Gröbner basis
Our Method I	$n = 2, LD = 7, m = 1000$	$k \geq 889$	667	< 1	< 1
Our Method II	$n = 2, LD = 27, m = 1000$	$k \geq 583$	630	2.57	< 1

Table 2 Comparison of our Method I and II for $n = 2$; LD means Lattice Dimension.

Method I 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 3. In fact although theoretically we need $\det(L) < p^{\omega - n - o(1)}$, experimentally we observe that condition $\det(L) < p^{\omega - o(1)}$ is sufficient to find the roots. Thus for Method I, experimental result is much better than the theoretical prediction.

Experimental results of Method II are presented in tabular form in Table 4. We cannot perform the experiments for our Method II for large values of n due to very high lattice dimensions.

n	Lower bound of k (theory)	Lower bound of k (expt.)	Lattice Dimension	Time in Seconds	
				LLL Algorithm	Gröbner basis
4	760	602	16	< 1	< 1
6	694	573	29	3.15	< 1
8	654	558	46	17.94	2.83
10	628	548	67	74.17	18.65
12	609	540	92	239.60	88.55
14	596	535	121	651.81	318.49

Table 3 Method I: Experimental results for 1000-bit p.

n	d	Lower bound of k (theory)	Lower bound of k (expt.)	Lattice Dimension	Time in Seconds	
					LLL Algorithm	Gröbner basis
2	1	583	630	27	2.57	< 1
2	2	583	608	125	3371.41	6111.55
3	1	550	582	256	29581.79	28305.81

Table 4 Method II: Experimental results for 1000-bit p.

5 Conclusion

In this paper, we have studied the Modular Inversion Hidden Number Problem (MIHNP). This problem was studied in [7] and two heuristics were presented. We note that there is a flaw in the improved heuristic proposed in [7], and as a result, it requires two-third (instead of claimed one-third) of the bits of the outputs to find the hidden number.

Then, using the same relations as in [7], we could reconstruct the lattice, so that the problem can be heuristically solved by the knowledge of five-eighth of the bits. Finally, we explored additional relations to solve the problem heuristically with only half of the output bits in asymptotic sense. In terms of theoretical results, our Method II is better than our Method I. The experimental results are also close to what we obtain in theory. We also note that our Method I is better in terms of actual implementation as the lattice dimension is much less that what is required for Method II.

References

1. A. Akavia. Solving Hidden Number Problem with One Bit Oracle and Advice. Crypto 2009, LNCS 5677, pp. 337–354, 2009.
2. A. Bauer, D. Vergnaud and J-C Zapalowicz. Inferring Sequences Produced by Nonlinear Pseudorandom Number Generators Using Coppersmith's Methods. PKC 2012, LNCS 7293, pp. 609–626, 2012.
3. M. Bellare, A. Desai, D. Pointcheval and P. Rogaway. Relations among notions of security for public-key encryption schemes. Crypto 1998, LNCS 1462, pp. 26–46, 1998.
4. S. R. Blackburn, D. Gómez-Pérez, J. Gutierrez and I. Shparlinski. Reconstructing noisy polynomial evaluation in residue rings. J. Algorithms, 61(2):47–59, 2006.
5. R. Courant and F. John. Introduction to Calculus and Analysis. Springer, 1989.
6. D. Boneh and R. Venkatesan. Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes. Crypto 1996, LNCS 1109, pp. 129–142, 1996.
7. D. Boneh, S. Halevi and N. Howgrave-Graham. The Modular Inversion Hidden Number Problem. Asiacrypt 2001, LNCS 2248, pp. 36–51, 2001.
8. J. Boyar. Inferring sequences produced by pseudo-random number generators. J. ACM, 36(1):129–141, 1989.
9. J-S Coron. Finding Small Roots of Bivariate Integer Polynomial Equations Revisited. Eurocrypt 2004, LNCS 3027, pp. 492–505, 2004.
10. J-S Coron. Finding Small Roots of Bivariate Integer Polynomial Equations: A Direct Approach. Crypto 2007, LNCS 4622, pp. 379–394, 2007.
11. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. Eurocrypt 2002, LNCS 2332, pp. 45–64, 2002.

12. D. Coppersmith. Small Solutions to Polynomial Equations and Low Exponent Vulnerabilities. Journal of Cryptology, 10(4):223–260, 1997.
13. M. Ernst, E. Jochemsz, A. May, and B. de Weger. Partial key exposure attacks on RSA up to full size exponents. Eurocrypt 2005, LNCS 3494, pp. 371–386, 2005.
14. R. Gennaro, S. Halevi and T. Rabin. Secure Hash-and-Sign Signature Without the Random Oracle. Eurocrypt 1999, LNCS 1592, pp. 123–139, 1999.
15. D. Gómez-Pérez, J. Gutierrez and A. Ibeas. Attacking the Pollard Generator. IEEE Transactions on Information Theory, 52(12):5518–5523,2006.
16. M. Herrmann and A. May. Attacking Power Generators Using Unravelled Linearization: When Do We Output Too Much? Asiacrypt 2009, LNCS 5912, pp. 487–504, 2008.
17. N. Howgrave-Graham. Finding small roots of univariate modular equations revisited. In Proceedings of IMA International Conference on Cryptography and Coding, LNCS 1355, pp. 131–142, 1997.
18. N. Howgrave-Graham and N. P. Smart. Lattice Attacks on Digital Signature Schemes. Des. Codes Cryptography, vol. 23(3), pp. 283-290, 2001.
19. R. Impagliazzo and D. Zuckerman. How to Recycle Random Bits. FOCS 1989, pp. 248–253, 1989.
20. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. Mathematische Annalen, 261:515–534, 1982.
21. S. Ling, I.E. Shparlinski, R. Steinfeld, and H. Wang. On the Modular Inversion Hidden Number Problem. Journal of Symbolic Computation, 47(4):358–367, 2012.
22. M. Naor and O. Reingold. Number theoretic constructions of efficient pseudo random functions. FOCS 1997, pp. 458–467, 1997.
23. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Eurocrypt 1999. pp. 223238, LNCS 1592, 1999.
24. M. Ritzenhofen. On Efficiently Calculating Small Solutions of Systems of Polynomial Equations. PhD thesis, Ruhr-University of Bochum, Germany, 2010. Available at `http://www.cits.rub.de/personen/maike.html`.
25. I. E. Shparlinski. Playing Hide-and-Seek with Numbers: the Hidden Number Problem, Lattices and Exponential Sums. Proc. Symp. in Appl. Math., vol. 62. Amer. Math. Soc., pp. 153-177, 2005.
26. R. Steinfeld, J. Pieprzyk and H. Wang. On the Provable Security of an Efficient RSA-Based Pseudorandom Generator. Asiacrypt 2006. pp. 194-209, LNCS 4284, 2006.
27. J. Stren. Secret Linear Congruential Generators Are Not Cryptographically Secure. FOCS 1987, pp. 421–426, 1987.
28. A. Ta-Shma, D. Zuckerman and S. Safra. Extractors from Reed-Muller Codes. FOCS 2001, pp. 638–647, 2001.
29. J. Xu, L. Hu, Z. Huang and L. Peng. Modular Inversion Hidden Number Problem Revisited ISPEC 2014, pp. 537–551, LNCS 8434, 2014.

Appendix A: An Intermediate Improvement on MIHNP

Here we consider same n many relations f_i as in [7], as described in our Section 3. Using these relations we prove that one can solve MIHNP when $k > \frac{5m}{8}$.

Theorem 4 *Let p be an m-bit prime. Let $\alpha \in \mathbb{Z}_p$ be hidden. Consider that $n+1$ pairs $(x_i, MSB_k(\alpha + x_i)^{-1} \bmod p)$ are given for random $x_0, x_1, \ldots, x_n \in \mathbb{Z}_p$. Then one can find α heuristically if $k > \left(\frac{5}{8} + \frac{1}{24n} + o(1)\right)m$.*

Proof We have a system of n equations $f_i = A_i\epsilon_0\epsilon_i + B_i\epsilon_i + C_i\epsilon_0 + D_i$, for $1 \le i \le n$ over the variables $\epsilon_0, \epsilon_1, \ldots, \epsilon_n$. From f_i, we generate another system of n equations $g_i = A_i^{-1} f_i \bmod p$ for $1 \le i \le n$. For a given sequence of $n+1$ non negative integers $r_0, r_1, \ldots, r_n$, let us define a new sequence $\{j_1, \ldots, j_n\}$ where

$$j_l = \begin{cases} \min(r_0, r_1), & \text{for } l = 1, \\ \min(r_0 - \sum_{k=1}^{l-1} j_k, r_l), & \text{for } l > 1. \end{cases}$$

Further, for a positive integer d, consider the following shift polynomials

$$h_{r_0,\cdots,r_n} = \left(\prod_{k=1}^{n} g_k^{j_k}\right)\epsilon_0^{r_0 - \sum_{k=1}^{n} j_k}\left(\prod_{k=1}^{n} \epsilon_k^{r_k - j_k}\right)p^{nd - \sum_{k=1}^{n} j_k},$$

for $0 \le r_0 \le nd$ and $0 \le r_k \le d$ with $1 \le k \le n$.

Note that $h_{r_0,\cdots,r_n}(\epsilon_0, \epsilon_1, \ldots, \epsilon_n) \equiv 0 \bmod p^{nd}$. Let $Z = 2^{m-k}$. Clearly $|\epsilon_i| < Z$. Now construct a lattice L using the coefficient vectors of the polynomials $h_{r_0,\cdots,r_n}(\epsilon_0 Z, \ldots, \epsilon_n Z)$. The dimension ω of the lattice L is

$$\omega = \sum_{r_0=0}^{nd} \sum_{r_1,\cdots,r_n=0}^{d} 1 = (nd+1)(d+1)^n.$$

Now, let us calculate the determinant of the lattice L. We have

$$\det(L) = \Bigg(\prod_{r_0=0}^{nd} \prod_{r_1,\cdots,r_n=0}^{d} Z^{r_0} \cdots Z^{r_n} \Bigg) \Bigg(\prod_{r_0=0}^{nd} \prod_{r_1,\cdots,r_n=0}^{d} p^{nd - \sum_{k=1}^{n} j_k} \Bigg)$$

Now $\prod_{r_0=0}^{nd} \prod_{r_1,\cdots,r_n=0}^{d} Z^{r_0} \cdots Z^{r_n} = Z^{\sum_{r_0=0}^{nd} \sum_{r_1,\cdots,r_n=0}^{d} (r_0 + \cdots + r_n)} = Z^{n^2 d^{n+2} + o(d^{n+2})}$.

Also,

$$\begin{aligned} \prod_{r_0=0}^{nd} \prod_{r_1,\cdots,r_n=0}^{d} p^{nd - \sum_{k=1}^{n} j_k} &= p^{\sum_{r_0=0}^{nd} \sum_{r_1,\cdots,r_n=0}^{d} \left(nd - \min(r_0, r_1 + \cdots + r_n) \right)} \\ &= p^{nd\omega - \sum_{r_0=0}^{nd} \sum_{r_1,\cdots,r_n=0}^{d} \min(r_0, r_1 + \cdots + r_n)}. \end{aligned}$$

Let us now calculate $\sum_{r_0=0}^{nd} \sum_{r_1,\cdots,r_n=0}^{d} \min(r_0, r_1 + \cdots + r_n)$. We have

$$\begin{aligned} \sum_{r_0=0}^{nd} \sum_{r_1,\ldots,r_n=0}^{d} \min(r_0, r_1 + \cdots + r_n) &= \sum_{r_1,\ldots,r_n=0}^{d} \sum_{r_0=0}^{r_1+\cdots+r_n} r_0 \\ &+ \sum_{r_1,\ldots,r_n=0}^{d} \sum_{r_0=r_1+\cdots+r_n+1}^{nd} (r_1 + \cdots + r_n) \\ &= \sum_{r_1,\cdots,r_n=0}^{d} \frac{(r_1 + \cdots + r_n)^2}{2} + \\ &\sum_{r_1,\cdots,r_n=0}^{d} (nd - r_1 - \cdots - r_n)(r_1 + \cdots + r_n) + o(d^{n+2}) \\ &= \sum_{r_1,\cdots,r_n=0}^{d} \left(nd(r_1 + \cdots + r_n) - \frac{(r_1 + \cdots + r_n)^2}{2} \right) \\ &= \frac{n^2 d^{n+2}}{2} - \frac{1}{2} \sum_{r_1,\cdots,r_n=0}^{d} (r_1 + \cdots + r_n)^2 + o(d^{n+2}) \\ &= \frac{n^2 d^{n+2}}{2} - \frac{1}{2} \left(\frac{n d^{n+2}}{12} + \frac{n^2 d^{n+2}}{4} \right) + o(d^{n+2}) \\ &= \left(\frac{3}{8} n^2 - \frac{n}{24} \right) d^{n+2} + o(d^{n+2}) \end{aligned}$$

Hence, we have $\det(L) = Z^{n^2 d^{n+2} + o(d^{n+2})} p^{nd\omega - \left(\frac{3}{8} n^2 - \frac{n}{24} \right) d^{n+2} - o(d^{n+2})}$. So, from the condition $\det(L) < p^{nd(\omega - n) - o(1)}$, we have $Z^{n^2 d^{n+2}} < p^{\left(\frac{3}{8} n^2 - \frac{n}{24} \right) d^{n+2} - o(1)}$. Since $Z = 2^{m-k}$ and $p \approx 2^m$, we finally obtain $k > \left(\frac{5}{8} + \frac{1}{24n} + o(1) \right) m$. $\square$

Hence, as n approaches to infinity, we need $k \ge \frac{5}{8} m$ to solve the problem.

Appendix B: Proof of Lemma 1

Proof Let us prove the result by induction on n and first consider $n = 2$, i.e, $A = [r_0, r_1, r_2]$ with $r_i \geq 0$ for $0 \leq i \leq 2$. Suppose that one of r_0, r_1, r_2 is zero, then $\min_1(r_0, r_1, r_2) = 0$. In this case the value of $s = \min_2(r_0, r_1, r_2)$, and so $s = \min_2(r_0, r_1, r_2) + \frac{1}{2}\min_1(r_0, r_1, r_2)$. We can take $E(2) = 0 < 3^3$ and thus the result is true in this case.

Now assume that $r_i > 0$ for all $i \in [0, 2]$. Recall that in each successful iteration (the condition of line 4 is true), two elements of A are decremented by 1. Suppose that for the first time one of the element of A becomes zero after l successful iterations.

After l successful iterations, array A is of the form $A = \Big[r_0 - 2\lfloor\frac{l}{3}\rfloor + e_0, r_1 - 2\lfloor\frac{l}{3}\rfloor + e_1, r_2 - 2\lfloor\frac{l}{3}\rfloor + e_2\Big]$, where $|e_i| < 3$ for $0 \leq i \leq 2$.

Now as after l iterations, one of the elements becomes zero, we have either $r_0 - 2\lfloor\frac{l}{3}\rfloor + e_0 = 0$ or $r_1 - 2\lfloor\frac{l}{3}\rfloor + e_1 = 0$ or $r_2 - 2\lfloor\frac{l}{3}\rfloor + e_2 = 0$.

Since $r_0 - \frac{2l}{3} + 2 + e_0 > r_0 - 2\left\lfloor\frac{l}{3}\right\rfloor + e_0 \geq r_0 - \frac{2l}{3} + e_0$, if $r_0 - 2\left[\frac{l}{3}\right] + e_0 = 0$, we must have $r_0 - \frac{2l}{3} + e_0 + \eta_0 = 0$ for some $\eta_0 \in [0, 2)$. That is $\frac{2l}{3} = r_0 + e$ for some $e = |e_0| + \eta_0$. Here $|e| < 3 + 2 = 5$. Similarly in general we have $\frac{2l}{3} = \min_1(r_0, r_1, r_2) + e$ for some $|e| < 5$.

After l successful iterations, algorithm gives $\min_2(r_0 - 2\lfloor\frac{l}{3}\rfloor + e_0, r_1 - 2\lfloor\frac{l}{3}\rfloor + e_1, r_2 - 2[\frac{l}{3}] + e_2)$ more successful iterations before terminate.

Thus

$$\begin{aligned}
s &= l + \min_2\left(r_0 - 2\lfloor\frac{l}{3}\rfloor + e_0, r_1 - 2\lfloor\frac{l}{3}\rfloor + e_1, r_2 - 2\lfloor\frac{l}{3}\rfloor + e_2\right)\\
&= l + \min_2(r_0 + e_0, r_1 + e_1, r_2 + e_2) - 2\lfloor\frac{l}{3}\rfloor\\
&= l - \frac{2l}{3} + \min_2(r_0, r_1, r_2) + e'\\
&= \frac{l}{3} + \min_2(r_0, r_1, r_2) + e'\\
&= \frac{\min_1(r_0, r_1, r_2)}{2} + \min_2(r_0, r_1, r_2) + \frac{1}{2}e + e'
\end{aligned}$$

where $|e'| < 2 + \max\{e_0, e_1, e_2\} < 6$. Thus we can take $E(2) = \frac{1}{2}e + e' < \frac{1}{2} \cdot 5 + 6 < (2 + 1)^3$. Hence result is true for $n = 2$.

Suppose the result is true for $n = m$. We have to prove that the result holds for $n = m + 1$. Now consider $A = [r_0, \ldots, r_m, r_{m+1}]$.

First suppose one of $r_0, \ldots, r_m, r_{m+1}$ is zero. Without loss of generality, let $r_{m+1} = 0$. In this case $\min_1(r_0, \ldots, r_m, r_{m+1}) = 0$. Also $\min_i(r_0, \ldots, r_m) = \min_{i+1}(r_0, \ldots, r_{m+1})$ for $i \geq 1$. Also value of s of the algorithm for inputs $(r_0, \ldots, r_m, r_{m+1})$ and $(r_0, \ldots, r_m)$ is same. Thus by induction hypothesis,

$$\begin{aligned}
s &= \min_m(r_0, r_1, \ldots, r_m) + \frac{1}{2}\sum_{i=1}^{m-1}\min_i(r_0, r_1, \ldots, r_m) + E(m)\\
&= \min_1(r_0, r_1, \ldots, r_{m+1}) + \min_{m+1}(r_0, r_1, \ldots, r_{m+1}) + \frac{1}{2}\sum_{i=2}^{m}\min_i(r_0, r_1, \ldots, r_{m+1}) + E(m)\\
&= \min_{m+1}(r_0, r_1, \ldots, r_{m+1}) + \frac{1}{2}\sum_{i=1}^{m}\min_i(r_0, r_1, \ldots, r_{m+1}) + E(m)
\end{aligned}$$

Hence if we take $E(m + 1) = E(m)$, $|E(m + 1)| < (m + 2)^3$ as $|E(m)| < (m + 1)^3$. Thus the result is true for $n = m + 1$.

Finally assume $r_i > 0$ for all $0 \le i \le m+1$. Suppose that for the first time one of the element of A becomes zero after l successful iterations. After l successful iterations, array A is of the form $A = \left[r_0 - (m+1)\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + e_0, \ldots, r_{m+1} - (m+1)\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + e_{m+1}\right]$ with $|e_i| < m+2$ for $i \in [0, m+1]$.
Since one of the elements of A is zero, $\frac{m+1}{\binom{m+2}{2}} l = \min_1(r_0, \ldots, r_{m+1}) + e$, where $|e| < (m+2) + (m+1) = 2m+3$. So we have $l = \frac{m+2}{2} \min_1(r_0, \ldots, r_{m+1}) + \frac{m+2}{2} e$.
Also since after l successful iterations, one of the elements in A becomes 0, without loss of generality, suppose last element $r_{m+1} - (m+1)\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + e_{m+1} = 0$. By induction hypothesis, if the input of the algorithm is

$$\left(r_0 - (m+1)\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + e_0, \ldots, r_m - (m+1)\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + e_m\right),$$

the total number of successful iterations is

$$\begin{aligned}
&\min_m(r_0 - (m+1)\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + e_0, \ldots, r_m - (m+1)\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + e_m) + \\
&\frac{1}{2}\sum_{i=1}^{m-1} \min_i(r_0 - (m+1)\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + e_0, \ldots, r_m - (m+1)\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + e_m) + E(m) \\
=\,& \min_{m+1}(r_0 - (m+1)\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + e_0, \ldots, r_{m+1} - (m+1)\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + e_{m+1}) \\
&+ \frac{1}{2}\sum_{i=2}^{m} \min_i(r_0 - (m+1)\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + e_0, \ldots, r_{m+1} - (m+1)\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + e_{m+1}) + E(m) \\
=\,& \min_{m+1}(r_0 + e_0, \ldots, r_{m+1} + e_{m+1}) - (m+1)\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + \\
&\frac{1}{2}\sum_{i=2}^{m} \min_i(r_0 + e_0, \ldots, r_{m+1} + e_{m+1}) - (m-1)\frac{m+1}{2}\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + E(m) \\
=\,& \min_{m+1}(r_0 + e_0, \ldots, r_{m+1} + e_{m+1}) - \frac{(m+1)^2}{2}\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + \\
&\frac{1}{2}\sum_{i=2}^{m} \min_i(r_0 + e_0, \ldots, r_{m+1} + e_{m+1}) + E(m).
\end{aligned}$$

Thus the value of s will be

$$\begin{aligned}
s \,=\,& l + \min_{m+1}\left(r_0 + e_0, \ldots, r_{m+1} + e_{m+1}\right) - \frac{(m+1)^2}{2}\lfloor \frac{l}{\binom{m+2}{2}} \rfloor + \\
&\frac{1}{2}\sum_{i=2}^{m} \min_i(r_0 + e_0, \ldots, r_{m+1} + e_{m+1}) + E(m) \\
=\,& \left(\frac{m+2}{2} - \frac{(m+1)^2}{2\binom{m+2}{2}} \cdot \frac{m+2}{2}\right) \min_1(r_0, \ldots, r_{m+1}) + \min_{m+1}(r_0, \ldots, r_{m+1}) + \\
&\frac{1}{2}\sum_{i=2}^{m} \min_i\left(r_0, \ldots, r_{m+1}\right) + E(m) + \epsilon' \\
=\,& \min_{m+1}(r_0, \ldots, r_{m+1}) + \frac{1}{2}\sum_{i=1}^{m} \min_i\left(r_0, \ldots, r_{m+1}\right) + E(m) + e',
\end{aligned}$$

where $|e'| < \frac{m+2}{2}(2m+3) + (m+2) + \frac{(m+1)^2}{2}\left(1 + \frac{1}{\binom{m+2}{2}}\frac{m+2}{2}(2m+3)\right)$
$+\frac{m-1}{2}(m+2) = 3m^2 + \frac{17}{2}m + 6.$ $\Bigg($Here we are using
$l = \frac{m+2}{2}\min_1(r_0, \ldots, r_{m+1}) + \frac{m+2}{2}e, |e| < 2m+3, |e_i| < m+2 \ \ \forall i \in [0, m+1].$ $\Bigg)$
Thus if we take $E(m+1) = E(m) + e'$,
$|E(m+1)| < 3m^2 + \frac{17}{2}m + 6 + |E(m)| < 3m^2 + \frac{17}{2}m + 6 + (m+1)^3 < (m+2)^3$. Hence by induction we have for all n,

$$s = \min_n(r_0, r_1, \ldots, r_n) + \frac{1}{2}\sum_{i=1}^{n-1}\min_i(r_0, r_1, \ldots, r_n) + E(n),$$

with $|E(n)| < (n+1)^3$. □