Modular Inversion Hidden Number Problem- A Lattice Approach

Pranjal Dutta
pranjal@cmi.ac.in
Chennai Mathematical Institute
Chennai, India

November 23, 2015

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 Sarkar in [28] identified that the claim in Method II is actually not correct and a detailed calculation justified that this method too requires two-third of the bits of the output, contrary to the claim in BHH'01. He reconstructed the lattice and give a bound which heuristically solve with half of the output bits. Although J.Xu et al in [29] solved it with only one-third of the output bits asymptotically but that technique is difficult to understand and implement. Here we essentially use similar idea of [28] but in a clever way such that it is a better bound although we solve the problem heuristically with only half of the output bits in asymptotic sense. This is better method and a lot easier to understand and implement. Also experimental results support the claim corresponding to our heuristics. In the last section we actually talk about a variant of this which seems to be hard to solve under lattice attack.

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

1 Introduction

Background: In recent years many attempts have been taken to construct efficient cryptosystems several new complexity assumptions. One of them was The Decision Diffie-Hellman assumption (DDH) which is essentially harder than Computational DiffieHellman assumption . It was used to construct chosen cipher text secure encryption.It was also used to generate number theoretic pseudo random functions and message authentication codes. Likewise the Strong RSA assumption was used to construct efficient signature scheme. Further, security of the public key encryption schemes proposed in are based on Pailliers Decision Composite Residuosity (DCR) and Quadratic Residuosity (QR) assumptions. [22]

Hidden Number Problems(HNPs) were first introduced by Boneh and Venkatesan in 1996 [5] . Using this they actually showed that computing the most significant bits of the secret key from the public keys of participants, in a Diffe-Hellman Key exchange Protocol is as much hard as computing the key itslef. Also it opened a broader section of open problems related (and unrelated) to cryptography. In fact this leads to practically feasible side channel attack on certain implementations of DSA(e.g. OpenSSL) [17] which emphasizes the security risk which is caused by a hidden side channel in the design of Pentium 4HTT processor like SSH. The problem is as follows:

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

Modular Inversion Hidden Number Problem (MIHNP) MIHNPs are closely related to HNP. Assuming hardness of some variants of this MIHNP problems they actually constructed very efficient algebraic PRNGs and MACs. The basic step in evaluating the MAC and the PRNG is one modular inversion modulo a moderate size prime. The general idea of MIHNPs is as follows:

Let $\alpha \in \mathbb{Z}_p$ be the (hidden) secret where p is a m-bit prime. 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 $(x_i, MSB_k((\alpha + x_i)^{-1} \bmod p))$ for some $k > 0$. The question is whether we can obtain α efficiently. There is a basic assumption called δ-MIHNP assumption that is there is no polynomial time algorithm for the Basic-MIHNP problem whenever $k < \delta m$

In their paper [6] they proposed a lattice-based algorithm which eventually solves this problem when $k > \frac{m}{3}$. They also explained why this algorithm does not extend to solve it for $k < \frac{m}{3}$. Most Importantly they conjectured that such techniques cannot be used beyond the $\frac{m}{3}$ bound. More generally, they conjectured that the δ-MIHNP assumption holds for any $\delta < \frac{1}{3}$.

The basic MIHNP can also be viewed as an analog of the famous Discrete-Log Problem (DLP) which is as follows:

> Given $g^\alpha \bmod p$ find the hidden number α.

We have various MIHNP problems, we also do not have reductions between the various Discrete-Log problems, yet the only algorithms that we know for solving any of them involve solving Discrete-Log. In [6], authors presented two polynomial time heuristics to solve MIHNP, provided that k is sufficiently large. Their first heuristic was a linear approach that we refer here as Method I works only if more than $\frac{2}{3}$ portion of most significant bits of $(\alpha + x_i)^{-1} \bmod p$ are given, i.e., $k > \frac{2m}{3}$. Now most importantly, the second heuristic was

better in some sense as it claimed knowledge of significantly fewer bits which is $k > \frac{m}{3}$ only. In the second heuristic, multipliers were used which we will refer as method II. Recently, the work of probabilistic polynomial time algorithm for MIHNP provided by Ling, Shiparlinski, Steinfeld and Wang [20] could match one of the heuristics (Method I) of Boneh et al. [6], where one requires at least $(\frac{2}{3} + \epsilon)m$ for a fixed $\epsilon > 0$ to solve the problem.

Very recently, Sarkar in [28] has studied MIHP and proved a flaw in the analysis of Method II of Boneh et al. He has mentioned that

> "More precisely, we show that the approach of [6] only works when $k > \frac{2m}{3}$ and does not work in the range of $\frac{m}{3} < k < \frac{2m}{3}$."

In [28], two approaches are given. In the first approach, one can find α if

Theorem 1 ([28]). *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 $x_0, x_1, \ldots, x_n \in \mathbb{Z}_p$.*

1. *One can find α in polynomial time if $k > \left(\frac{1}{2} + \frac{3n+1}{2(n+1)^2}\right)m$*
2. *One can find α in time polynomial in m but exponential in n if $k > m\left(\frac{1}{2} + \frac{1}{(n+1)(n+2)}\right)$.*

Polynomials generation in [6, 28]: Suppose attacker knows p, x_i and $b_i = \mathrm{MSB}_k((\alpha + x_i)^{-1} \bmod p)$ for $0 \le i \le n$. Hence we can write:

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

for $0 \le i \le n$. 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

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

for $1 \le i \le n$. 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 $1 \le i \le n$. Thus n many relations have been used to solve the problem. These relations were used in [6].

Sarkar generates 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

$$\begin{aligned}(x_i - x_j)\epsilon_i\epsilon_j + (x_i b_j - x_j b_j + 1)\epsilon_i + (x_i b_i - x_j b_i - 1)\epsilon_j \\ + (x_i b_i b_j - x_j b_i b_j + b_i - b_j) \equiv 0 \bmod p.\end{aligned}$$

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 $g_{i,j}(\epsilon_i, \epsilon_j) \equiv 0 \bmod p$. Thus Sarkar generates $\binom{n+1}{2}$ relations.

Our Contribution. In this paper we improve the attack bound. We prove that one can find α in time polynomial in m but exponential in n if

$$k > m\left(\frac{1}{2} + \frac{1}{(n+2)!}\right)$$

which is a slight improvement of [28] previous theorem as $(\frac{1}{2} + \frac{1}{(n+1)(n+2)}) > (\frac{1}{2} + \frac{1}{(n+2)!})$ although both asymptotically turn out to be $\frac{m}{2}$ for large enough n. We believe that improving our technique we can actually improve the bound to $\frac{1}{3}$ or close to that which will actually be the result achieved by J.Xu et al in [29]. Here the technique is lot easier to understand , verify and implement.

2 Preliminaries

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

$$L = \{z : z = a_1b_1 + ... + a_wb_w, a_1, .., a_w \in \mathbb{Z}\}$$

is called an w dimensional lattice with basis $B = \{b_1, .., b_w\}$. A full rank lattice when $w = n$. The determinant of L is defined as $\det(L) = \det(B)$ where B is a $w \times w$ matrix. When b_i's are in $\mathbb{Z}^n$, the lattice is called integer lattice.

In 1982, Lenstra, Lenstra and Lovasz 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 $b_1, ..., b_w$ of a lattice L, LLL algorithm can find a reduced basis $u_1, ..., u_w$ with

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

.

Coppersmith formulated ideas to find small roots of a modular polynomial in a single variable and also for polynomials in two variables over the integers later which was pioneered in many of the related papers leading nice results in cryptography and related fields. Not to forget, Coron's work was also seminal in the context. The idea of Coppersmith can also be extended to more than two variables, but the method becomes a heuristic in such cases. The following ground-breaking result due to Howgrave-Graham gives a sufficient condition under which modular roots become the roots over integers for polynomials over two or more variables.

Theorem 2 (Howgrave-Graham)**.** *Let $h(x_1, ..., x_t)$be a polynomial with integer-coefficients which is a sum of w many monomials. Suppose that*

1. *$h(y_1, .., y_t) \equiv 0 \bmod R$ for some positive integer R such that $|y_i| < Y_i$ for each $i = 1(1)t$*
2. *$\|h(y_1Y_1, ..., y_tY_t)\| < \frac{R}{\sqrt{w}}$.*

Then $h(x_1, ..., x_n) = 0$ hold over integers.

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 Grobner basis technique.

3 Algorithm and attack bound

We use same polynomials as in [28]. However our lattice construction is construction is different from [28]. Let us start with the Algorithm 1.

[H]

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

```
Sort A in the descending order ;
while A[0] > 0&A[1] > 0 do
    A[0] = A[0] − 1 ;
    A[1] = A[1] − 1 ;
    s = s + 1 ;
    Sort A in the descending order ;
end
```

Algorithm 1

Lemma 1.

$$s = \begin{cases} \dfrac{\sum_{i=0}^{n} r_i}{2}, & \text{for } \max_i r_i \leq \sum_{i=0}^{n} r_i \\ \sum_{i=0}^{n} r_i - \max_i r_i, & \text{otherwise} \end{cases}$$

Proof. Suppose initially we have $A = [r_0, \ldots, r_n]$ with $r_0 \geq r_1 \geq \ldots \geq r_n$. At each step we are decreasing the first two elements (if sorted) and suppose after k-th step our list is like $[r_0^k, \ldots, r_n^k]$ with sorted i.e. $r_0^k \geq r_1^k \geq \ldots \geq r_n^k$. Also at the last stage when the process ends A must look like $[0, 0, \ldots, r]$ where $r \geq 0$.

Now consider the first case. Initially at 0-th stage we have already $2r_0 \leq \sum_{i=0}^{n} r_i$. Here we can have possible two cases -

I)this kind of max structure like the first case is maintained through the end

II)there is a minimal N such that we have $2r_0^N \leq \sum_{i=1}^{n} r_i^N$ i.e. the first term is less than or equal to remaining sum and $2r_0^{N+1} > \sum_{i=1}^{n} r_i^{N+1}$

If I) happens then at the last stage the A must look like $[0, 0, ..., 0]$. At each step the sum in A decreases by 2 hence

$$s = \frac{\sum_{i=0}^{n} r_i}{2}$$

If II) happens then at N th stage we have the list sorted like $[r_0^N, \ldots, r_n^N]$, so at the very next step it should be like $[r_0^{N-1}, r_1^{N-1}, \ldots, r_{n-1}^N, r_n^N]$ (Unsorted).

We can easily understand the fact that in this situation r_0^{N-1} can not be maximum. Hence we must have $r_1^{N-1} \leq r_0^{N-1} < r_2^N \leq r_1^N \leq r_0^N$. From here we can conclude that $r_0^N = r_1^N = r_2^N$ at N-th stage we had the inequality $r_0^N \leq r_1^N + \ldots + r_n^N$ and the very next step we have this inequality violated. So we have $r_2^N > r_0^{N-1} + r_1^{N-1} + r_3^N + \ldots + r_n^N$. Combining this we can easily get that $r_0^N = 1$. So the list at N-th step must look like $[1, 1, 1, 0, 0, \ldots, 0]$ and the step where the inequality was violated must be $[1, 0, 0, \ldots, 0]$.

See at each step total summation in s is decreasing by 2. Total sum at the very first is $\sum_{i=0}^n r_i$ hence $s = \frac{\sum_{i=0}^n r_i - 1}{2}$. So basically we have

$$s = \left\lfloor \frac{\sum_{i=0}^n r_i}{2} \right\rfloor.$$

Like wise we can check the formula for the second case in the same manner. □

Now define $\mathcal{A} = \{[r_0, \ldots, r_n] | 0 \leq r_i \leq x, r_i' s \text{ are integers.}\}$. Let us also define $\mathcal{B} = \{[r_0, \ldots, r_n] | 0 \leq r_i \leq x, r_i' s \text{ are integers and } 2 \max r_i \leq \sum_{i=0}^n r_i\}$.

There are $(x+1)^{n+1}$ many elements in $\mathcal{A}$ and the maximum value of s can be $\lfloor \frac{(n+1)x}{2} \rfloor$. Hence maximum power of x in $\sum_{A \in \mathcal{A}} s$ can be $n+2$. We will in fact show the following lemma which will serve us sufficiently.

Lemma 2.

$$\sum_{A \in \mathcal{A}} s = \left(\frac{n+1}{4} - \frac{n+1}{2(n+2)!} \right) x^{n+2} + \epsilon(x),$$

where $\epsilon(x)$ is a polynomial of degree $n+1$.

Proof. First assume new function $g(A) = \min \left\{ \frac{\sum_{i=0}^n r_i}{2}, \sum_{i=0}^n r_i - \max r_i \right\}$. Now we have $\lfloor g(A) \rfloor = s(A)$. It is easy to see that the co-efficient of x^{n+2} in $\sum_{A \in \mathcal{A}} g(A)$ will be same as in $\sum_{A \in \mathcal{A}} s(A)$ because there are $(x+1)^{n+1}$ many elements in $\mathcal{A}$ and $s(A)$ differ from $g(A)$ by at most $\frac{1}{2}$.

Hence $\sum_{A \in \mathcal{A}} s(A)$ differ from $\sum_{A \in \mathcal{A}} g(A)$ by at most $\frac{1}{2}(x+1)^{n+1}$. Hence it suffices to find the co-efficient of x^{n+2} in $\sum_{A \in \mathcal{A}} g(A)$.

Now $2 \sum_{A \in \mathcal{A}} g(A) = \sum_{A \in \mathcal{A}} \min\{p(A), q(A)\}$, where $p(A) = \sum_{i=0}^n r_i$, $q(A) = 2 \sum_{i=0}^n r_i - 2 \max r_i$. It is also true that when $A \in \mathcal{B}$, $2g(A) = p(A)$. So we have

$$2 \sum_{A \in \mathcal{A}} g(A) = \sum_{A \in \mathcal{B}} p(A) + \sum_{A \in \mathcal{A} \setminus \mathcal{B}} q(A) = \sum_{A \in \mathcal{A}} p(A) - \sum_{A \in \mathcal{A} \setminus \mathcal{B}} (p(A) - q(A)).$$

Also

$$\begin{aligned}\sum_{A\in\mathcal{A}} p(A) &= \sum_{0\le r_i\le x} (r_0+\ldots+r_n)\\ &= (n+1)\sum_{0\le r_i\le x} r_0\\ &= (n+1)(x+1)^n\sum_{k=0}^{x} k\\ &= (n+1)(x+1)^n x(x+1)/2.\end{aligned}$$

So coefficient of x^{n+2} in the expression $\sum_{A\in\mathcal{A}} p(A) = \frac{n+1}{2}$.

Also $p(A)-q(A) = 2\max a_i - \sum_{i=0}^{n} a_i$. Fix $\max a_i = m$ with $x \ge m$ as we are interested to sum this $p(A)-q(A)$ over $\mathcal{A}\setminus\mathcal{B}$, we consider $\sum_{i=0}^{n} r_i = m+k$ with $m > k$ i.e. $k = 0\ldots, m-1$. Hence $2\max r_i - \sum_{i=0}^{n} r_i = m-k$. Now if $r_t = m$ for some $t \in \{0,\ldots,n\}$, we can easily see that no other r_i can be m as it will contradict that $A \in \mathcal{A}\setminus\mathcal{B}$. Now if $r_t = m$ we have the following equality $r_0+\ldots+r_{t-1}+r_{t+1}+\ldots+r_n = k$ with each $r_i \ge 0$. Obviously we want all $r_i's$ less than or equal to x. But the interesting part is obviously as $x \ge m > k$ all a_i's must be less than x . Hence we do not have to worry about over-counting. Number of non-negative solutions of is $\binom{n+k-1}{n-1}$. Hence the number of such lists $[r_0,\ldots,r_n]$ is nothing but $(n+1)\binom{n+k-1}{n-1}$ with $\max r_i = m$ and $[r_0,\ldots,r_n] \in \mathcal{A}\setminus\mathcal{B}$. Hence actually we have

$$\sum_{A\in\mathcal{A}\setminus\mathcal{B}} (p(A)-q(A)) = \sum_{m=1}^{x}\sum_{k=0}^{m-1}(m-k)(n+1)\binom{n+k-1}{n-1}.$$

Using this identity that $\sum_{t=0}^{s}\binom{t+r}{r} = \binom{s+r+1}{r+1}$ we get that

$$\sum_{m=1}^{x}\sum_{k=0}^{m-1} m\binom{n+k-1}{n-1} = \sum_{m=1}^{x} m\binom{m+n-1}{n} = (n+1)\binom{x+n}{n+2} + \binom{x+n-1}{n+1}$$

Contribution of x^{n+2} in $\sum_{m=1}^{x}\sum_{k=0}^{m-1} m\binom{n+k-1}{n-1}$ is $= \frac{n+1}{(n+2)!}$.

Likewise we have

$$\sum_{m=1}^{x}\sum_{k=0}^{m-1} k\binom{n+k-1}{n-1} = \sum_{m=1}^{x} n\binom{m+n-1}{n+1}$$

Now we can check that $\binom{m+n-1}{n+1} = \frac{m^{n+1}}{(n+1)!} + (\ldots)m^n + \ldots$. Hence the coefficient of x^{n+2} in the later sum is $\frac{n}{(n+2)!}$.

So co-efficient of x^{n+2} in the original sum $\sum_{m=1}^{x}\sum_{k=0}^{m-1}(m-k)(n+1)\binom{n+k-1}{n-1}$ is $(n+1)(\frac{n+1}{(n+2)!}-\frac{n}{(n+2)!})=\frac{n+1}{(n+2)!}$. Hence the co-efficient of x^{n+2} in $2\sum_{A\in\mathcal{A}}g(A)$ is $(\frac{n+1}{2}-\frac{n+1}{(n+2)!})$. So in $\sum_{A\in\mathcal{A}}g(A)$ rather in $\sum_{A\in\mathcal{A}}s(A)$ coefficient of x^{n+2} is actually $(\frac{n+1}{4}-\frac{n+1}{2(n+2)!})$. Hence finally we have

$$\sum_{A\in\mathcal{A}}s(A)=(\frac{n+1}{4}-\frac{n+1}{2(n+2)!})x^{n+2}+\epsilon(x),$$

where $\epsilon(x)$ is a polynomial of degree $n+1$. □

Theorem 3. *Let p be a m-bit prime number. Let α be in $\mathbb{Z}_p$ be hidden number. Consider $n+1$ pairs $(x_i, MSB_k(x_i+\alpha)^{-1} \bmod p)$ where x_i are are random in $\mathbb{Z}_p$. Then Then one can find α under Assumption 1 in time polynomial in m but exponential in n if $k > m(\frac{1}{2}+\frac{1}{(n+2)!}+o(1))$*

Proof. We use the same strategy used by Sarkar in [28] in his paper and construct the polynomials and the lattice . Now I will present another method which improves the bound $k > m(\frac{1}{2}+\frac{1}{(n+2)!}+o(1))$ by using the broad idea of [11] with a set of polynomials. Solving such a system of equations using Coppersmiths method is a non-trivial task, as discussed thoroughly in [23]. Hence we will somehow cleverly generate polynomials for constructing lattice from the original polynomials $g_{i,j}$ as 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 .

Now we look at one combinatorial problem. Suppose we are given $n+1$ non-negative integers $(r_0,...,r_n)$. 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

1) $\prod_{0\le i<j\le n}g_{i,j}^{s_{i,j}}$ divides $\prod_{i=0}^{n}{\epsilon_i}^{s_i}$

(2) $\sum_{0\le i<j\le n}s_{i,j}$ is maximized (to make det large as much as possible).

In this sense it is a purely optimization problem and solved in a combinatorial way . So actually we generate r_i's according to the algorithm I so that it is guaranteed to be maximized . Here we have $\epsilon_i\le 2^{m-k}$ and take $Z=2^{m-k}$. Now for $0\le r_0\le ..\le r_n\le X$ (X a large positive integer we can assume $X=nd$ for some positive integer d too) define the polynomials

$$h_{r_0,r_1,....,r_n}=g_{0,1}^{s_{0,1}}g_{1,2}^{s_{1,2}}....g_{n-1,n}^{s_{n-1,n}}\prod_{k=0}^{n}\epsilon_k^{\,r_k-\sum_{l=0}^{k-1}s_{l,k}-\sum_{l=k+1}^{n}s_{k,l}}p^{d\binom{n+1}{2}-s}$$

where $s_{i,j}$ and s are obtained by Algorithm 1 on input $[r_0,...,r_n]$ In the lexicographic ordering of subscript $(r_0,...,r_n)$, where each component $r_i\in[0,X]$. The matrix corresponding to

these polynomials will be lower-triangular and hence the matrix will be non-singular. Note that

$$h_{r_0,...,r_n}(\epsilon_0, ..., \epsilon_n) \equiv 0 \bmod p^{d\binom{n+1}{2}}.$$

Construct a lattice L using the coefficient vectors of $h_{r_0,...,r_n}(\epsilon_0 Z, ..., \epsilon_n Z)$. The dimension of the lattice is $w = (X+1)^{n+1}$. Now the determinant of L is

$$\det(\mathrm{L}) = \prod_{r_0=0}^{X} ... \prod_{r_n=0}^{X} Z^{r_0+....+r_n} p^{wd\binom{n+1}{2} - \sum_{r_0=0}^{X} ... \sum_{r_n=0}^{X} s}$$

So we need to satisfy the following $2^{\frac{w(w-1)}{4(w-n)}} \det(\mathrm{L})^{\frac{1}{w-n}} < \frac{1}{\sqrt{w}} p^{d\binom{n+1}{2}}$. Ignoring $\frac{w(w-1)}{4(w-n)}$ and $\frac{1}{\sqrt{w}}$ and simplifying we have the result that

$$k > m(\frac{1}{2} + \frac{1}{(n+2)!})$$

.

So when $k > m(\frac{1}{2} + \frac{1}{(n+2)!} + o(1))$ after lattice reduction we have $n+1$ polynomials $f_1, ..., f_{n+1}$ such that $f_i(0, ..., n) = 0$ for $1 \leq i \leq n+1$. Now under Assumption 1, we can find $\epsilon_0, ..., \epsilon_n$ from $f_1, ..., f_{n+1}$.

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. Also as previously said this is a better bound than [28] although asymptotically it is ($\approx \frac{m}{2}$) .

□

4 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 Grobner basis technique.

Experimental results of our method is presented in tabular form in Table 1. We cannot perform the experiments for our Method for large values of n due to very high lattice dimensions. Anyhow it shows slight improvement than experimental results in [28].

n	d	Lower bound of k (theory)	Lower bound of k (expt.)	Lattice Dimension	Time in Seconds	
					LLL Algorithm	Gröbner basis
2	1	541	620	27	3.28	< 1
2	2	541	600	125	9392.584	6803.95

Table 1: Experimental results for 1000-bit p.

5 Conclusion and A variant of MIHNP

In this paper, we have studied the Modular Inversion Hidden Number Problem (MIHNP). This problem was studied in [6] and two heuristics were presented. In [28] additional relations to solve the problem heuristically with only half of the output bits in asymptotic sense were explored. Here we extend the idea to give a better bound although asymptotically it is half of the bits . In [29] J. Xu et al actually showed that it can be done heuristically with only one-third of the output bits . But here the technique is easier to understand , implement and believed to achieve the same bound if applied more cleverly. Still there is scope to work with the problem to find whether the bound can be further improved or not .

In fact if we actually work with two or more hidden elements then it seems to be harder to break. The problem can be stated as follows :

Suppose we have an oracle and two hidden numbers $\alpha, \beta \in \mathbb{Z}_p$. x_i's are chosen uniformly from the same field . Each time we send one x_i to the oracle it sends back either $MSB_k((\alpha + x_i)^{-1} \bmod p)$ or $MSB_k((\beta + x_i)^{-1} \bmod p)$ non-deterministically. Suppose some one knows p and he knows n pairs of (x_i, X) where X is either $MSB_k((\alpha + x_i)^{-1} \bmod p)$ or $MSB_k((\beta + x_i)^{-1} \bmod p)$ whcih he does not know . What should be the least number of bits to be output by the oracle to recover α and β efficiently ?

One important point is that it seems that usual lattice approach will not work because we don't know for some x_i what the oracle returns $MSB_k((\alpha + x_i)^{-1} \bmod p)$ or $MSB_k((\beta + x_i)^{-1} \bmod p)$. In fact experimentally we try to randomly choose α and β for that of MSB and try lattice attack . But it seems to deviate a lot (Which seems to be fair enough) from the actual numbers α and β . So it might be an interesting problem to solve ! . In fact we can try with $l \geq 2$ number of hidden numbers and extend the same idea . This seems to be harder to break as well as secure enough to use in many cryptographic fields.

Acknowledgments: I am a sincerely thankful to my Prof. **Santanu Sarkar**, Indian Institute of Technology, Madras for introducing me the problem and invaluable feedback, kind suggestions who constantly encouraged me to solve this problem.

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] 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.

[6] D. Boneh, S. Halevi and N. Howgrave-Graham. The Modular Inversion Hidden Number Problem. Asiacrypt 2001, LNCS 2248, pp. 36–51, 2001.

[7] J. Boyar. Inferring sequences produced by pseudo-random number generators. J. ACM, 36(1):129–141, 1989.

[8] J-S Coron. Finding Small Roots of Bivariate Integer Polynomial Equations Revisited. Eurocrypt 2004, LNCS 3027, pp. 492–505, 2004.

[9] J-S Coron. Finding Small Roots of Bivariate Integer Polynomial Equations: A Direct Approach.
Crypto 2007, LNCS 4622, pp. 379–394, 2007.

[10] 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.

[11] D. Coppersmith. Small Solutions to Polynomial Equations and Low Exponent Vulnerabilities. Journal of Cryptology, 10(4):223–260, 1997.

[12] 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.

[13] R. Gennaro, S. Halevi and T. Rabin. Secure Hash-and-Sign Signature Without the Random Oracle. Eurocrypt 1999, LNCS 1592, pp. 123–139, 1999.

[14] D. Gómez-Pérez, J. Gutierrez and A. Ibeas. Attacking the Pollard Generator. IEEE Transactions on Information Theory, 52(12):5518–5523,2006.

[15] 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.

[16] 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.

[17] N. Howgrave-Graham and N. P. Smart. Lattice Attacks on Digital Signature Schemes. Des. Codes Cryptography, vol. 23(3), pp. 283-290, 2001.

[18] R. Impagliazzo and D. Zuckerman. How to Recycle Random Bits. FOCS 1989, pp. 248–253, 1989.

[19] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. Mathematische Annalen, 261:515–534, 1982.

[20] 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.

[21] M. Naor and O. Reingold. Number theoretic constructions of efficient pseudo random functions. FOCS 1997, pp. 458–467, 1997.

[22] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Eurocrypt 1999. pp. 223-238, LNCS 1592, 1999.

[23] 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`.

[24] 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.

[25] 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.

[26] J. Stren. Secret Linear Congruential Generators Are Not Cryptographically Secure. FOCS 1987, pp. 421–426, 1987.

[27] A. Ta-Shma, D. Zuckerman and S. Safra. Extractors from Reed-Muller Codes. FOCS 2001, pp. 638–647, 2001.

[28] S. Sarkar. Modular Inversion Hidden Number Problem Correction and Improvements, Cryptology ePrint Archive: Report 2015/778

[29] J. Xu, L. Hu, Z. Huang and L. Peng. Modular Inversion Hidden Number Problem Revisited. ISPEC 2014, pp. 537–551, LNCS 8434, 2014.