Some Applications of Lattice Based Root Finding Techniques

Santanu Sarkar and Subhamoy Maitra

Applied Statistics Unit, Indian Statistical Institute, 203 B. T. Road, Kolkata 700 108, India.
{santanu_r, subho}@isical.ac.in

Abstract. In this paper we present some problems and their solutions exploiting lattice based root finding techniques.
In CaLC 2001, Howgrave-Graham proposed a method to find the Greatest Common Divisor (GCD) of two large integers when one of the integers is exactly known and the other one is known approximately. In this paper, we present three applications of the technique. The first one is to show deterministic polynomial time equivalence between factoring N ($N = pq$, where $p > q$ or p, q are of same bit size) and knowledge of $q^{-1} \bmod p$. Next, we consider the problem of finding smooth integers in a short interval. The third one is to factorize N given a multiple of the decryption exponent in RSA.
In Asiacrypt 2006, Jochemsz and May presented a general strategy for finding roots of a polynomial. We apply that technique for solving the following two problems. The first one is to factorize N given an approximation of a multiple of the decryption exponent in RSA. The second one is to solve the implicit factorization problem given three RSA moduli considering certain portions of LSBs as well as MSBs of one set of three secret primes are same.

Keywords: CRT-RSA, Greatest Common Divisor, Factorization, Implicit Factorization, Lattice, LLL, RSA, Smooth Integers.

1 Introduction

It is well known that given two large integers a, b ($a > b$), one can calculate the GCD efficiently in $O(\log^2 a)$ time. In [HOW01], Howgrave-Graham has shown that it is possible to calculate the GCD efficiently when some approximations of a, b are available. This problem was referred to as Approximate Common Divisors problem. Coron and May [COR07] used the strategy of [HOW01] to proved the deterministic polynomial time equivalence of computing the RSA secret key and factoring. In this paper we present three other interesting applications of the technique presented in [HOW01].

First, we use the idea of [HOW01] to prove that factoring N is deterministic polynomial time equivalent to finding $q^{-1} \bmod p$ when $p > q$ or p, q are of same bit size. In the presentation of a recent paper [HEN09] at Crypto 2009, it has been asked how one can use $q^{-1} \bmod p$ towards factorization of N as $q^{-1} \bmod p$ is stored as a part of the secret key in PKCS #1 [PKCS].

In this direction, let us briefly explain RSA [RSA78] first. One needs to generate two large primes p, q, with (in general) $q < p < 2q$. Then we have $N = pq$ and $\phi(N) = (p-1)(q-1)$. Further, e, d are identified such that $ed = 1 + k\phi(N)$, $k \geq 1$. N, e are publicly available and the plaintext $M \in \mathbb{Z}_N$ is encrypted as $C \equiv M^e \bmod N$. The secret key d is required to decrypt the ciphertext as $M \equiv C^d \bmod N$.

To make the decryption process faster, the Chinese Remainder Theorem has been exploited and the model is well known as CRT-RSA [QUI82,WIE90]. The encryption technique is same as RSA, but the decryption process is little different. Instead of one decryption exponent as in standard RSA, two decryption exponents (d_p, d_q) are required in this case, where $d_p \equiv d \bmod (p-1)$ and $d_q \equiv d \bmod (q-1)$. To decrypt the ciphertext C, one needs to calculate both $C_p \equiv C^{d_p} \bmod p$ and $C_q \equiv C^{d_q} \bmod q$. From C_p, C_q one can get the plaintext M by the application of CRT using $q^{-1} \bmod p$. This is the reason, $q^{-1} \bmod p$ is stored in the secret key part of PKCS #1 [PKCS].

One may be tempted to consider the following method to factorize N from the knowledge of $q^{-1} \bmod p$, which does not actually work. Consider $q_1 = q^{-1} \bmod p$. Given N, one can easily calculate $q_2 = q_1^{-1} \bmod N$ using Extended Euclidean Algorithm. Now $q_2 q_1 - 1$ is divisible by N and hence $q_2 q_1 - 1$ is divisible by p. Thus, $q_2 \equiv q_1^{-1} \bmod p \equiv q \bmod p$. If q_2 would have been less than p, then $q_2 = q$ and the factorization will be immediate. However, in general, q_2 is of $O(N)$ and not less than p. Thus this method does not work and we need to look for a lattice based strategy which we explain in Section 2.1.

Next we consider the problem of finding smooth integers in a small interval [BON00]. Finding smooth numbers is important for application in the well known factorization algorithms such as quadratic sieve [POM84] and number field sieve [LEN93]. We study the results of [BON00] and show that slightly improved outcome could be achieved using a different strategy following the idea of [HOW01]. This is presented in Section 2.2.

The paper [RSA78] itself presents a probabilistic polynomial time algorithm that, on input N, e, d, provides the factorization of N. It has been proved that [MAY04,COR07] given N, e, d, one can factor N in deterministic poly$(\log N)$ time provided $ed \leq N^2$. In Section 2.3, we consider a slightly different scenario when a multiple of d say μd is known, but d is not known. Given that μ is very large, it is not possible to factorize μd easily. Thus to factorize N in such a scenario, we need to consider different approach. We exploit the idea of [HOW01] again and prove that given $\mu d, e, N$, one can factor N provided $\mu < N$. The time complexity of our deterministic algorithm is $e \cdot O(\log N)$. Indeed, the algorithm is not feasible for large e. However, the most popular mode of RSA considers small e, say of the order of $2^{16} + 1$. In such a scenario, the algorithm works in poly$(\log N)$ time.

To extend the problem little further, in Section 3.1, we assume that instead of μd, some approximation of it is available. We note that the idea of finding roots of a polynomial as described in [ELL06] can be suitably exploited here.

Finally, in Section 3.2, we study the implicit factorization problem introduced in [MAY09]. Consider $N_1 = p_1 q_1, N_2 = p_2 q_2, \ldots, N_k = p_k q_k$, where $p_1, p_2, \ldots, p_k$ and $q_1, q_2, \ldots, q_k$ are primes. It is also considered that $p_1, p_2, \ldots, p_k$ are of same bit size and so are $q_1, q_2, \ldots, q_k$. Given that certain portions of bit pattern in $p_1, p_2, \ldots, p_k$ are common, the question is under what conditions it is possible to factor $N_1, N_2, \ldots, N_k$ efficiently. Several results in this direction are presented in [MAY09,SAR09,SAR09A,FAU10]. Here we consider the case for $k = 3$, when certain portions of MSBs as well as LSBs of p_1, p_2 and p_3 are same. This case has not been covered earlier.

2 Applications of Approximate Integer Common Divisor Problem [HOW01]

For our purpose we need the following two results. We first state the following one due to Howgrave-Graham [HOW97].

Lemma 1. *Let $h(x) \in \mathbb{Z}[x]$ be the sum of at most ω monomials. Suppose that $h(x^{(0)}) \equiv 0 \bmod N^m$ where $|x^{(0)}| \leq X$ and $||h(xX)|| < \frac{N^m}{\sqrt{\omega}}$. Then $h(x^{(0)}) = 0$.*

We also note that the basis vectors of an LLL-reduced basis fulfill the following property [LLL82].

Lemma 2. *Let L be an integer lattice of dimension ω. The LLL algorithm applied on L outputs a reduced basis of L spanned by $\{v_1, \ldots, v_\omega\}$ with*

$$||v_1|| \leq ||v_2|| \leq \ldots \leq ||v_i|| \leq 2^{\frac{\omega(\omega-1)}{4(\omega+1-i)}} \det(L)^{\frac{1}{\omega+1-i}}, \text{ for } i = 1, \ldots, \omega,$$

in polynomial time of dimension ω and the bit size of the entries of L.

2.1 Equivalence of finding $q^{-1} \bmod p$ and factorization

The following theorem proves the main result towards the equivalence.

Theorem 1. *Assume $N = pq$, where p, q are primes and $p \approx N^\gamma$. Suppose an approximation p_0 of p is known such that $|p - p_0| < N^\beta$. Given $q^{-1} \bmod p$, one can factor N deterministically in poly$(\log N)$ time when $\beta - 2\gamma^2 < 0$.*

Proof. Let $q_1 = q^{-1} \bmod p$. So we can write $qq_1 = 1 + k_1 p$ for some positive integer k_1. Multiplying both sides by p, we get $q_1 N = p + k_1 p^2$. That is, we have $q_1 N - p = k_1 p^2$. Let $x_0 = p - p_0$. Thus, we have $q_1 N - p_0 - x_0 = k_1 p^2$. Also we have $N^2 = p^2 q^2$. Our goal is to recover x_0 from $q_1 N - p_0$ and N^2.

Note that p^2 is the GCD of $q_1 N - p_0 - x_0$ and N^2. In this case $q_1 N - p_0$ and N^2 is known, i.e., one term N^2 is exactly known and the other term $q_1 N - p_0 - x_0$ is approximately known. This is exactly the Partially Approximate Common Divisor Problem (PACDP) [HOW01] and we follow a similar technique to solve this as explained below. This will provide the error term $-x_0$, which added to the approximation $q_1 N - p_0$, gives the exact term $q_1 N - p_0 - x_0$.

Take $X = N^\beta$ as an upper bound of x_0. Then we consider the shift polynomials

$$\begin{aligned} g_{ij}(x) &= x^i (q_1 N - p_0 + x)^j N^{2(m-j)} \\ &\text{for } i = 0,\ 0 \leq j \leq m \text{ and } j = m,\ 1 \leq i \leq t, \end{aligned} \tag{1}$$

where m, t are fixed non-negative integers. Clearly, $g_{ij}(-x_0) \equiv 0 \bmod (p^{2m})$.

We construct the lattice L spanned by the coefficient vectors of the polynomials $g_{ij}(xX)$ in (1). One can check that the dimension of the lattice L is $\omega = m + t + 1$ and the determinant of L is

$$det(L) = X^{\frac{(m+t)(m+t+1)}{2}} N^{2\frac{m(m+1)}{2}} = X^{\frac{(m+t)(m+t+1)}{2}} N^{m(m+1)}. \tag{2}$$

Using Lattice reduction on L by LLL algorithm [LLL82], one can find a non-zero vector b whose norm $||b||$ satisfies $||b|| \leq 2^{\frac{\omega-1}{4}}(det(L))^{\frac{1}{\omega}}$. The vector b is the coefficient vector of the polynomial $h(xX)$ with $||h(xX)|| = ||b||$, where $h(x)$ is the integer linear combination of the polynomials $g_{ij}(x)$. Hence $h(-x_0) \equiv 0 \bmod (p^{2m})$. To apply Lemma 1 and Lemma 2 for finding the integer root of $h(x)$, we need

$$2^{\frac{\omega-1}{4}}(det(L))^{\frac{1}{\omega}} < \frac{p^{2m}}{\sqrt{\omega}}. \tag{3}$$

Neglecting small constant terms, we can rewrite (3) as $det(L) < p^{2m\omega}$. Substituting the expression of $det(L)$ from (2) and using $X = N^{\beta}, p \approx N^{\gamma}$ we get

$$\frac{(m+t)(m+t+1)}{2}\beta + m(m+1) < 2m(m+t+1)\gamma. \tag{4}$$

Let $t = \tau m$. Then neglecting the terms of $o(m^2)$ we can rewrite (4) as

$$\frac{\tau^2\beta}{2} + (\beta - 2\gamma)\tau + \frac{\beta}{2} - 2\gamma + 1 < 0. \tag{5}$$

Now, the optimal value of τ to minimize the left hand side of (5) is $\frac{2\gamma-\beta}{\beta}$. Putting this optimal value in (5), we get $\beta - 2\gamma^2 < 0$.

Our strategy uses LLL [LLL82] algorithm to find $h(x)$ and then calculates the integer root of $h(x)$. Both these steps are deterministic polynomial time in $\log N$. Thus the result. □

Corollary 1. *Factoring N is deterministic polynomial time equivalent to finding $q^{-1} \bmod p$, where $N = pq$ and $p > q$.*

Proof. When no approximation of p is given, then β in the Theorem 1 is equal to γ. Putting $\beta = \gamma$ in the condition $\beta - 2\gamma^2 < 0$, we get $\gamma > \frac{1}{2}$. This requirement forces the condition that $p > q$. Also, it is trivial to note that if the factorization of N is known then one can efficiently compute $q^{-1} \bmod p$. Thus the proof. □

Corollary 2. *Factoring N is deterministic polynomial time equivalent to finding $q^{-1} \bmod p$, where $N = pq$ and p, q are of same bit size.*

Proof. The proof of the case $p > q$ is already taken care in Corollary 1. Now consider $q > p$. When p, q are of same bit size and $p < q$, then $p < q < 2p$, i.e., $\sqrt{\frac{N}{2}} < p < \sqrt{N}$ and $\sqrt{N} < q < \sqrt{2N}$.

Now if we take $p_0 = \sqrt{N}$ then $|p - p_0| < (1 - \frac{1}{\sqrt{2}})\sqrt{N} < \frac{N^{\frac{1}{2}}}{2} = N^{\frac{1}{2} - \frac{\log 2}{\log N}}$. Also $p > N^{\frac{1}{2} - \frac{\log 2}{2\log N}}$. So in this case we can take $\beta = \frac{1}{2} - \frac{\log 2}{\log N}$ and $\gamma > \frac{1}{2} - \frac{\log 2}{2\log N}$. Thus, $\beta - 2\gamma^2 < \frac{1}{2} - \frac{\log 2}{\log N} - 2(\frac{1}{2} - \frac{\log 2}{2\log N})^2 = -\frac{\log^2 2}{2\log^2 N} < 0$. Hence in this situation one can factor N following Theorem 1. □

It needs to be studied how the situation can be tackled when p is significantly smaller than q.

Now let us describe the experimental result. We have implemented the program in SAGE 4.1 over Linux Ubuntu 8.10 on a laptop with Dual CORE Intel(R) Pentium(R) D CPU 1.83 GHz, 2 GB RAM and 2 MB Cache. Note that our result in Theorem 1 holds when the lattice dimension approaches to infinity. Since in practice we use finite lattice dimension, we may not reach the bound presented in Theorem 1. For experiments, we consider that small amount of Most Significant Bits (MSBs) of p is known. In Table 1, we provide some practical results. In the first three experiments, we take N as 1000-bit integer with p, q of the same bit size and $p > q$. Then in the next three experiments, we swapped p, q, i.e., q becomes larger than p. Given $q^{-1} \bmod p$, we could successfully recover p in all the cases.

p ? q	# MSBs of p known	Lattice Parameters (m, t)	Lattice Dimension	Time (in sec.)
$p > q$	46	(5, 5)	11	1.41
$p > q$	24	(10, 10)	21	66.33
$p > q$	20	(11, 11)	23	119.72
$q > p$	47	(5, 5)	11	1.42
$q > p$	24	(10, 10)	21	66.56
$q > p$	20	(11, 11)	23	120.00

Table 1. Experimental results following Theorem 1.

2.2 Finding smooth integers in a short interval

Following [BON00], let us first formally define two notions of smoothness.

Definition 1.

- *An integer N is called B smooth if N has no prime divisor greater than B.*
- *An integer N is called strongly B smooth if N is B smooth and p^m can not divide N for any m for which $p^m > B$.*

Let us denote the n-th prime by p_n, e.g., $p_1 = 2, p_2 = 3$ and so on. Suppose we want to find a strongly B smooth integer (as written in Definition 1) N in the interval $[U, V]$.

Now let us present our result.

Theorem 2. *Let $S = \prod_{i=1}^{n} p_i^{a_i}$ where $a_i = \lfloor \frac{\log B}{\log p_i} \rfloor$ and $p_1, \ldots, p_n$ are all distinct primes not exceeding B. Let $I = [U, V]$. One can find all strongly B smooth integers $N \in I$ for which $gcd(N, S) > d$ in poly$(\log S)$ time when $|I| < 2d^{\frac{\log d}{\log S}}$ and $V < 2d$.*

Proof. We will try to find N such that $\gcd(N, S) > d$. Let us take take $a_0 = \lfloor \frac{U+V}{2} \rfloor$. We consider a_0 as an approximation of N. Thus we will try to find the GCD of S, N, by knowing exactly S and some approximation of N, which is a_0 (but N is not known). Here we follow the

idea of solving the Partially Approximate Common Divisor Problem (PACDP) as explained in [HOW01].

Let $x_0 = N - a_0$. We want to calculate x_0 from a_0, S. Assume $X = d^\beta$ is an upper bound of x_0. Let $S = d^\delta$. Using the same approach as in the proof of Theorem 1, we get the condition as

$$\frac{(m+t)(m+t+1)}{2}\beta + \frac{m(m+1)}{2}\delta < m(m+t+1). \tag{6}$$

Let $t = \tau m$. Then neglecting the terms of $o(m^2)$ we can rewrite (6) as

$$\frac{\beta}{2}\tau^2 + (\beta - 1)\tau + \frac{\beta}{2} + \frac{\delta}{2} - 1 < 0. \tag{7}$$

Now, the optimal value of τ to minimize the left hand side of (7) is $\frac{1-\beta}{\beta}$. Putting this optimal value in (7), we get $\beta < \frac{1}{\delta}$. Now $\delta = \frac{\log S}{\log d}$. So x_0 should be less than $d^{\frac{\log d}{\log S}}$.

Thus, we get x_0 and hence N in poly($\log S$) time. As, $V < 2d$, we have $N < 2d$ (since $U \leq N \leq V$). When $\gcd(N, S) > d$, then $\gcd(N, S) = N$ as $N < 2d$. Hence N divides S, i.e., N is strongly B smooth. □

B	$\log_2 d$	$\log_2(V-U)$	LD (Our), Time (sec.)	LD ([BON00]), Time (sec.)
1000	450	130	36, 15.51	32, 21.33
1000	496	156	29, 3.77	26, 8.06
1000	496	161	45, 36.88	41, 64.71

Table 2. Comparison of our experimental results with that of [BON00]. We have implemented the ideas of [BON00] for experimental comparison. LD denotes Lattice Dimension.

Asymptotically, our result is 8 times better than that of [BON00, Theorem 3.1], as that bound was $|I| < \frac{1}{4}d^{\frac{\log d}{\log S}}$. We present a few experimental results, where we find improved outcomes (in terms of execution time) using our strategy than that of [BON00]. One should also note, that the method of [BON00] requires the implementation of CRT on several primes, which is not included in the time mentioned in Table 2. Our strategy using the idea of [HOW01] does not require such computation.

2.3 Factorization of N when a multiple of d is known

In this section we analyse how N can be factorized when a large multiple of d is available.

Theorem 3. *Let d be order of N and $ed < N^2$. Suppose a multiple of the decryption exponent d, say μd is known. Then one can factor N deterministically in $e \cdot O(\log N)$ time when $\mu < N$.*

Proof. We have $ed = 1 + k(N + 1 - p - q)$. Since $k < e$, one can try every integer in $[1, e-1]$ as k. Let, $d_0 = \frac{1+kN}{e}$. Then $|d_0 - d| = \frac{k(p+q-1)}{e} < p + q$. Since $p + q = O(\sqrt{N})$, one can find the bits in the most significant half of d in e many trials.

Let $x_0 = d_0 - d$, i.e., $|x_0| < \sqrt{N}$. Further μd is known. From μd and d_0 we try to find out d following the solution strategy of the Partially Approximate Common Divisor Problem (PACDP) [HOW01].

Let $\mu \approx N^{\alpha}$. Using the same approach as in the proof of Theorem 1, we get the condition as

$$\frac{(m+t)(m+t+1)}{2}\frac{1}{2} + \frac{m(m+1)}{2}(1+\alpha) < m(m+t+1). \tag{8}$$

Let $t = \tau m$. Then neglecting the terms of $o(m^2)$ we can rewrite (8) as

$$\frac{1}{4}\tau^2 - \frac{1}{2}\tau + \frac{1}{2}\alpha - \frac{1}{4} < 0. \tag{9}$$

The optimal value of τ to minimize the left hand side of (9) is 1. Putting this optimal value in (9), we get $\alpha < 1$. After finding d one can deterministically factor N using the idea of [COR07]. □

For the experiments, we consider p, q of 500 bits each. We take $e = 2^{16} + 1$. Theoretically we should get results for $\mu < N$, i.e., μ can be of 1000 bits. In experiments, we could reach 940 bits for μ.

Bits of μ	Lattice Parameters (m, t)	Lattice Dimension	Time (in sec.)
900	(10, 10)	21	7.88
925	(14, 14)	29	44.91
940	(20, 20)	41	356.14

Table 3. Experimental results following Theorem 3.

3 Applications of finding Integer roots using the idea of [ELL06]

Next we consider a more general scenario when some approximation of a multiple of d is known. The approach presented in [ELL06] can be nicely exploited to this problem. Further, we also use the idea of [ELL06] to solve an instance of implicit factorization problem.

For the two results in this section, we need the following assumption.

Assumption 1 *Consider a set of polynomials $\{f_1, f_2, \ldots, f_i\}(i \geq n)$ on n variables having the root of the form $(x_{1,0}, x_{2,0}, \ldots, x_{n,0})$ after lattice reduction using the idea of [ELL06]. Then we can collect the root $(x_{1,0}, x_{2,0}, \ldots, x_{n,0})$ from $f_1, f_2, \ldots, f_i$.*

3.1 Factorization of N when an approximation of a multiple of d is available

We assume that an approximation of μd is known. Let, $A = \mu d - z_0$ is known, where $\mu \approx N^{\alpha}, z_0 \approx N^{\beta}$. We first state the following result due to Boneh el. al. [BON98].

Lemma 3. *Let* $|p-q| > \frac{\sqrt{N}}{4}$ *and* $e < \frac{N^{\frac{1}{4}}}{8}$. *Given* $\frac{\log_2 N}{4}$ *many bits of* d *in the positions* $\frac{\log_2 N}{4}$ *to* $\frac{\log_2 N}{2}$, *one can deterministically factor* N *in* $e^2 O(\log N)$ *time.*

Based on this, we get following result.

Lemma 4. *Given* $e < \frac{N^{\frac{1}{4}}}{8}$, d *of* $O(N)$, $|p-q| > \frac{\sqrt{N}}{4}$, $\alpha < \frac{1}{2}$ *and* $\beta - \alpha < \frac{1}{4}$, *one can deterministically factor* N *in* $e^2 O(\log N)$ *time.*

Proof. Similar to the proof of Theorem 3, one can find find an integer d_0 in e many trials such that $|d - d_0| < \sqrt{N}$. Since, $\mu = N^{\alpha}$, we have $A \approx N^{1+\alpha}$, considering $d \approx N$ and $|z_0| < |\mu d|$. Let, $\mu_1 = \lfloor \frac{A}{d_0} \rfloor$. Then

$$|\mu - \mu_1| \approx |\frac{A(d_0 - d) + z_0 d_0}{d d_0}| \approx N^{\alpha - \frac{1}{2}}.$$

Thus, as long as $\alpha \leq \frac{1}{2}$, we have $\mu = \mu_1$. Now, $|\frac{z_0}{\mu}| \approx N^{\beta-\alpha} < N^{\frac{1}{4}}$ as $\beta - \alpha < \frac{1}{4}$. Then $|d - \lfloor \frac{A}{\mu} \rfloor| \approx |\frac{z_0}{\mu}| < N^{\frac{1}{4}}$. So, we find an approximation $d_2 = \lfloor \frac{A}{\mu} \rfloor$ of d such that d, d_2 matches every bits except lower $\frac{\log_2 N}{4}$ bits. In such a situation, one can factor N using the method of [BON98, Theorem 3.3] in $e^2 O(\log N)$ time. □

When $\alpha > \frac{1}{2}$, the approach of Lemma 4 cannot be used immediately. However, in such a situation, a heuristic solution is possible following the idea of [ELL06].

Theorem 4. *Given* e, N *and* $\mu d - z_0$, *one can factor* N *in* $e \cdot O(\log N)$ *time if* $\alpha^2 + 2\alpha\beta - 3\beta^2 + 2\beta - 1 < 0$ *and* $1 - \alpha - \beta \geq 0$, *under Assumption 1.*

Proof. Let, $x_0 = d - d_0$ and $y_0 = \mu - \mu_1$. Then, we have $A = (d_0 + x_0)(\mu_1 + y_0) - z_0$. Hence, we are interested to find the root of $f(x, y, z) = A - (d_0 + x)(\mu_1 + y) + z$, which is (x_0, y_0, z_0). Let, $X = \sqrt{N}, Y = N^{\alpha - \frac{1}{2}}$, and $Z = N^{\beta}$. Clearly, X, Y, Z are the upper bounds of (x_0, y_0, z_0). Following the "Extended Strategy" of [ELL06, Page 274], we get

$$S = \bigcup_{0 \leq j_1 \leq t} \{x^i y^{j+j_1} z^k : x^i y^j z^k \text{ is a monomial of } f^m\},$$

$$M = \{ \text{ monomials of } x^i y^j z^k f : x^i y^j z^k \in S\}.$$

It follows that,

$$x^i x^j z^k \in S \Leftrightarrow \begin{cases} k = 0, \ldots, m, \\ i = 0, \ldots, m - k, \\ j = 0, \ldots, m - k + t, \end{cases}$$

and

$$x^i x^j z^k \in M \Leftrightarrow \begin{cases} k = 0, \ldots, m + 1, \\ i = 0, \ldots, m + 1 - k, \\ j = 0, \ldots, m + 1 - k + t, \end{cases}$$

We exploit t many extra shifts of y where t is a non-negative integer. Our aim is to find two more polynomials f_0, f_1 that share the root (x_0, y_0, z_0) over the integers.

From [ELL06], we know that these polynomials can be found by lattice reduction if

$$X^{s_1}Y^{s_2}Z^{s_3} < W^s, \tag{10}$$

where $s = |S|$, $s_j = \sum_{x^{i_1}y^{i_2}z^{i_3} \in M \setminus S} i_j$, for $j = 1, 2, 3$, and $W = \|f(xX, yY, zZ)\|_\infty \geq \mu_1 X = N^{\alpha+\frac{1}{2}}$.

One can check that

$$\begin{aligned}
s_1 &= \frac{m^3}{2} + \frac{5}{2}m^2 + 4m + t + \frac{1}{2}m^2t + \frac{3}{2}mt + 2,\\
s_2 &= \frac{1}{2}m^3 + \frac{5}{2}m^2 + 4m + 3t + m^2t + \frac{7}{2}mt + \frac{1}{2}mt^2 + t^2 + 2,\\
s_3 &= \frac{1}{3}m^3 + \frac{3}{2}m^2 + \frac{13}{6}m + t + \frac{3}{2}mt + \frac{1}{2}m^2t + 1,\\
s &= \frac{1}{3}m^3 + \frac{3}{2}m^2 + \frac{13}{6}m + t + \frac{3}{2}mt + \frac{1}{2}m^2t + 1.
\end{aligned}$$

Let $t = \tau m$, where τ is a non negative real number. Neglecting the lower order terms and putting the values of X, Y, Z and the lower bound of W from (10), we get the condition as

$$\frac{1}{2}\alpha\tau^2 + \frac{1}{2}\alpha\tau - \frac{1}{4}\tau^2 + \frac{1}{2}\tau\beta + \frac{1}{6}\alpha - \frac{1}{2}\tau + \frac{1}{3}\beta - \frac{1}{6} < 0. \tag{11}$$

The optimal value of τ, to minimize the left hand side of (11), is $\frac{1-\alpha-\beta}{2\alpha-1}$. Putting this optimal value, the required condition becomes $\alpha^2 + 2\alpha\beta - 3\beta^2 + 2\beta - 1 < 0$. Since $\tau \geq 0$, so we need $1 - \alpha - \beta \geq 0$.

That is, when these conditions hold, according to [ELL06], we get two polynomials f_0, f_1 such that $f_0(x_0, y_0, z_0) = f_1(x_0, y_0, z_0) = 0$. Under Assumption 1, we can extract x_0, y_0, z_0 in poly($\log N$) time. □

Now we describe a few experimental results and we consider that the primes are of 500 bits and $e = 2^{16}+1$. In all the experiments, we observe that f, f_0, f_1 are algebraically independent that support Assumption 1 and we could successfully collect the root using the method of resultants.

α	β	Lattice Parameters (m, t)	Lattice Dimension	Time (in sec.)
0.63	0.2	(1, 1)	20	2.58
0.55	0.39	(2, 1)	40	41.79
0.65	0.2	(2, 1)	40	51.89

Table 4. Experimental results following Theorem 4.

3.2 An instance of Implicit Factorization problem

Here we study a special case of the implicit factorization problem introduced in [MAY09]. Consider $N_1 = p_1q_1, N_2 = p_2q_2$, and $N_3 = p_3q_3$, where p_1, p_2, p_3 and q_1, q_2, q_3 are primes. It is also considered that p_1, p_2, p_3 are of same bit size and so are q_1, q_2, q_3. We also assume that some amount of LSBs as well as some amount of MSBs of p_1, p_2, p_3 are same. To the best of our knowledge, this is the first attempt to the solution of this instance.

Theorem 5. *Let $q_1, q_2, q_3 \approx N^\alpha$. Consider that $\gamma_1 \log_2 N$ many MSBs and $\gamma_2 \log_2 N$ many LSBs of p_1, p_2, p_3 are same. Let $\beta = 1 - \alpha - \gamma_1 - \gamma_2$. Then, under Assumption 1, one can factor N_1, N_2, N_3 in polynomial time if $10\alpha + 5\beta - 4 \leq 0$.*

Proof. It is given that $\gamma_1 \log_2 N$ many MSBs and $\gamma_2 \log_2 N$ many LSBs of p_1, p_2, p_3 are same. Thus, we can write $p_1 = N^{1-\alpha-\gamma_1}P_0 + N^{\gamma_2}P_1 + P_2$, $p_2 = N^{1-\alpha-\gamma_1}P_0 + N^{\gamma_2}P_1' + P_2$ and $p_3 = N^{1-\alpha-\gamma_1}P_0 + N^{\gamma_2}P_1'' + P_2$. Thus, $p_1 - p_2 = N^{\gamma_2}(P_1 - P_1')$. Since $N_1 = p_1q_1$ and $N_2 = p_2q_2$, putting $p_1 = \frac{N_1}{q_1}$ and $p_2 = \frac{N_2}{q_2}$, we get

$$N_1q_2 - N_2q_1 = N^{\gamma_2}(P_1 - P_1')q_1q_2. \tag{12}$$

Similarly, we have

$$N_1q_3 - N_3q_1 = N^{\gamma_2}(P_1 - P_1'')q_1q_3. \tag{13}$$

Now, multiplying Equation 12 by N_3 and Equation 13 by N_2 and then subtracting, we get $N_1N_3q_2 - N_1N_2q_3 - N^{\gamma_2}(P_1 - P_1')q_1q_2N_3 + N^{\gamma_2}(P_1 - P_1'')q_1q_3N_2 = 0$.

Thus we need to solve $f'(x, y, z, w, t) = N_1N_3y - N_1N_2z - N_3N^{\gamma_2}xyw + N_2N^{\gamma_2}xzt = 0$ whose roots corresponding to x, y, z, w, t are $q_1, q_2, q_3, P_1 - P_1', P_1 - P_1''$. Since there is no constant term in f', we define a new polynomial $f(x, y, z, w, t) = f'(x, y + 1, z, w, t) = N_1N_3 + N_1N_3y - N_1N_2z - N_3N^{\gamma_2}xyw - N_3N^{\gamma_2}xw + N_2N^{\gamma_2}xzt$. The root $(x_0, y_0, z_0, w_0, t_0)$ of f is $(q_1, q_2 - 1, q_3, P_1 - P_1', P_1 - P_1'')$. The idea of modifying the polynomial with a constant term was introduced in [COR04, Appendix A] and later used in [ELL06] which we follow here.

Let X, Y, Z, W, T be the upper bounds of $q_1, q_2 - 1, q_3, P_1 - P_1', P_1 - P_1''$ respectively. As given in the statement of this theorem, one can take $X = N^\alpha, Y = N^\alpha, Z = N^\alpha, W = N^\beta, T = N^\beta$. Following the "Basic Strategy" of [ELL06, Page 274],

$$S = \bigcup \{x^{i_1}y^{i_2}z^{i_3}w^{i_4}t^{i_5} : x^{i_1}y^{i_2}z^{i_3}w^{i_4}t^{i_5} \text{ is a monomial of } f^m\},$$

$$M = \{ \text{ monomials of } x^{i_1}y^{i_2}z^{i_3}w^{i_4}t^{i_5}f : x^{i_1}y^{i_2}z^{i_3}w^{i_4}t^{i_5} \in S\}.$$

It follows that,

$$x^{i_1}y^{i_2}z^{i_3}w^{i_4}t^{i_5} \in S \Leftrightarrow \begin{cases} i_3 = 0, \ldots, m, \\ i_5 = 0, \ldots, i_3, \\ i_4 = 0, \ldots, m - i_3, \\ i_2 = 0, \ldots, m - i_3, \\ i_1 = i_4 + i_5, \end{cases}$$

and

$$x^{i_1}y^{i_2}z^{i_3}w^{i_4}t^{i_5} \in M \Leftrightarrow \begin{cases} i_3 = 0, \ldots, m+1, \\ i_5 = 0, \ldots, i_3 + 1, \\ i_4 = 0, \ldots, m+1-i_3, \\ i_2 = 0, \ldots, m+1-i_3, \\ i_1 = i_4 + i_5. \end{cases}$$

From [ELL06], we know that these polynomials can be found by lattice reduction if

$$X^{s_1}Y^{s_2}Z^{s_3}W^{s_4}T^{s_5} < W_1^s, \tag{14}$$

where $s = |S|$, $s_j = \sum_{x^{i_1}y^{i_2}z^{i_3}w^{i_4}t^{i_5} \in M \setminus S} i_j$,
for $j = 1, 2, 3, 4, 5$, and $W_1 = \|f(xX, yY, zZ, wW, tT)\|_\infty \geq N_1 N_3 Y \approx N^{2+\alpha}$.

One can check that

$$\begin{aligned} s_1 &= \frac{5}{24}m^4 + \frac{7}{4}m^3 + \frac{127}{24}m^2 + \frac{27}{4}m + 3, \\ s_2 &= \frac{1}{8}m^4 + \frac{13}{12}m^3 + \frac{27}{8}m^2 + \frac{53}{12}m + 2, \\ s_3 &= \frac{1}{6}m^4 + \frac{4}{3}m^3 + \frac{23}{6}m^2 + \frac{14}{3}m + 2, \\ s_4 &= \frac{1}{8}m^4 + \frac{13}{12}m^3 + \frac{27}{8}m^2 + \frac{53}{12}m + 2, \\ s_5 &= \frac{1}{12}m^4 + \frac{2}{3}m^3 + \frac{23}{12}m^2 + \frac{7}{3}m + 1, \\ s &= \frac{1}{12}m^4 + \frac{2}{3}m^3 + \frac{23}{12}m^2 + \frac{7}{3}m + 1. \end{aligned}$$

Neglecting the lower order terms and putting the values of X, Y, Z, W, T as well as the lower bound of W_1, from (14), we get the condition as

$$\frac{5}{12}\alpha + \frac{5}{24}\beta - \frac{1}{6} < 0 \text{ i.e., } 10\alpha + 5\beta - 4 < 0. \tag{15}$$

That is, when this condition holds, according to [ELL06], we get four polynomials f_0, f_1, f_2, f_3 such that
$f_0(x_0, y_0, z_0, w_0, t_0) = f_1(x_0, y_0, z_0, w_0, t_0) = f_2(x_0, y_0, z_0, w_0, t_0) = f_3(x_0, y_0, z_0, w_0, t_0) = 0$.
Under Assumption 1, we can extract x_0, y_0, z_0, w_0, t_0 in poly$(\log N)$ time. □

Now we describe a few experimental results.

In Theorem 5, we consider the Assumption 1. Let us now clarify how it actually works. In the proof of Theorem 5, we consider that we will be able to get at least four polynomials f_0, f_1, f_2, f_3 along with f, that share the integer root. In experiments we found more than 4 polynomials (other than f) after the LLL algorithm that share the same root, and let us name them f_0, f_1, f_2, f_3. Let $R(f, f_0)$ be the resultant of f, f_0 and so on. We calculate

α	γ_1	γ_2	β (Theory)	Time	β (Experimental Values)			
			our	(in sec.)	our	[MAY09]	[SAR09A]	[FAU10]
0.25	0.2	0.175	0.3	1.64	0.375	0.372	0.383	0.372
0.3	0.225	0.225	0.2	1.55	0.25	0.248	0.269	0.246
0.35	0.28	0.26	0.1	1.52	0.11	0.125	0.151	0.123

Table 5. Theoretical and experimental values of α, β for which N_1, N_2, N_3 can be factored efficiently. Results using our technique are obtained with the lattice dimension 20.

$f_4 = R(f, f_0), f_5 = R(f, f_1)$ and then $f_6 = R(f_4, f_5)$. We always find a factor $-\frac{t_0}{\gcd(t_0,w_0)}w + \frac{w_0}{\gcd(t_0,w_0)}t$ of f_6. In all the cases, we find $\gcd(t_0, w_0) \leq 2$. After getting t_0, w_0, we define a new polynomial $f_7(y, z) = f_4(y, z, w_0, t_0)$. We always find a factor $q_3y - q_2z + q_3$ of f_7. From this we can find $(y_0, z_0) = (q_2 - 1, q_3)$. Finally, putting the values of y_0, z_0, w_0, t_0 in f, we obtain $x_0 = q_1$.

From Table 5 it may be noted that we get much better results in the experiments than the theoretical bounds. This is because, for the parameters we consider here, the shortest vectors may belong to some sub-lattice. However, the theoretical calculation in Theorem 5 cannot capture that and further, identifying such optimal sub-lattice seems to be difficult. This kind of scenario, where experimental results perform better than theoretical estimates, has earlier been observed in [ELL06, Section 7.1], too.

Our experimental results show that the total number of bits to be shared in the primes for implicit factorization is of similar order to the requirements in [MAY09,SAR09A,FAU10] for $k = 3$.

4 Conclusion

In this paper we use the method of finding approximate common divisor, as proposed in [HOW01], for approaching three problems. The first one is to show the deterministic polynomial time equivalence between factorization of the RSA moduli and finding $q^{-1} \bmod p$. To the best of our knowledge, this equivalence has not been studied earlier. We also do not find any trivial method to prove it. Next, we revisit the problem of finding smooth integers in an interval as explained in [BON00]. We find slightly improved results than that of [BON00] using the technique presented by [HOW01]. Further, using the technique of [HOW01], we identify how N can be factored when a multiple of the RSA secret exponent d is available.

We also exploit the technique of [ELL06] and solve two more problems in related areas. The first one is to show how N can be factored when some approximation of a multiple of the RSA secret exponent d is available. Next we consider an instance of the implicit factorization problem which has not been solved earlier.

References

[BON98] D. Boneh, G. Durfee and Y. Frankel. Exposing an RSA Private Key Given a Small Fraction of its Bits. Available at http://crypto.stanford.edu/∼dabo/abstracts/bits_of_d.html. [last accessed 25 March, 2010].

[BON00] D. Boneh. Finding smooth integers in short intervals using CRT decoding. Proceedings of STOC 2000, pages 265–272, 2000.

[COP97] D. Coppersmith. Small Solutions to Polynomial Equations and Low Exponent Vulnerabilities. Journal of Cryptology, 10(4):223–260, 1997.

[COR04] J.-S. Coron. Finding small roots of bivariate integer equations revisited, Eurocrypt 2004, LNCS 3027, pp. 492–505, 2004.

[COR07] J. -S. Coron and A. May. Deterministic polynomial-time equivalence of computing the RSA secret key and factoring. Journal of Cryptology, 20(1):39–50, 2007.

[FAU10] J-C. Faugere, R. Marinier and G. Renault. Implicit Factoring with Shared Most Significant and Middle Bits. Accepted in PKC 2010.

[HEN09] N. Heninger and H. Shacham. Reconstructing RSA Private Keys from Random Key Bits. Proceedings of Crypto 2009, Lecture Notes in Computer Science, Volume 5677, pages 1–17, Springer, 2009. The presentation is available at http://www.iacr.org/conferences/crypto2009/slides/p001-rsa-keys.pdf

[HOW97] N. Howgrave-Graham. Finding Small Roots of Univariate Modular Equations Revisited. Proceedings of Cryptography and Coding, Lecture Notes in Computer Science, Volume 1355, pages 131–142, Springer, 1997.

[HOW01] N. Howgrave-Graham. Approximate integer common divisors. Proceedings of CALC 2001, Lecture Notes in Computer Science, Volume 2146, pages 51–66, Springer, 2001.

[ELL06] E. Jochemsz and A. May. A strategy for finding roots of multivariate polynomials with new applications in attacking RSA variants, Asiacrypt 2006, LNCS 4284, pp. 267–282, 2006.

[LLL82] A. K. Lenstra, H. W. Lenstra and L. Lovász. Factoring Polynomials with Rational Coefficients. Mathematische Annalen, 261:513–534, 1982.

[LEN93] A. K. Lenstra and H. W. Jr. Lenstra. The Development of the Number Field Sieve. Springer-Verlag, 1993.

[MAY04] A. May. Computing the RSA secret key is deterministic polynomial time equivalent to factoring. Crypto 2004, LNCS 3152, pp. 213–219, 2004.

[MAY09] A. May and M. Ritzenhofen. Implicit factoring: on polynomial time factoring given only an implicit hint, PKC 2009, LNCS 5443, pp. 1–14,2009.

[PKCS] http://www.rsa.com/rsalabs/node.asp?id=2125

[POM84] C. Pomerance. The Quadratic Sieve Factoring Algorithm. Proceedings of Eurocrypt 1984, Lecture Notes in Computer Science, Volume 209, pages 169–182, 1985.

[QUI82] J. -J. Quisquater and C. Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. Electronic Letters, volume 18, pages 905–907, 1982.

[RSA78] R. L. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of ACM, 21(2):158–164, February 1978.

[SAR09] S. Sarkar and S. Maitra. Further Results on Implicit Factoring in Polynomial Time. Advances in Mathematics of Communications, 3(2):205–217, 2009.

[SAR09A] S. Sarkar and S. Maitra. Approximate Integer Common Divisor Problem relates to Implicit Factorization. Available at http://eprint.iacr.org/2009/626.

[WIE90] M. Wiener. Cryptanalysis of Short RSA Secret Exponents. IEEE Transactions on Information Theory, 36(3):553–558, 1990.