Approximate Integer Common Divisor Problem relates to Implicit Factorization

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 analyse how to calculate the GCD of k (≥ 2) many large integers, given their approximations. Two versions of the approximate common divisor problem, presented by Howgrave-Graham in CaLC 2001, are special cases of our analysis when $k = 2$. We then relate the approximate common divisor problem to the implicit factorization problem. This has been introduced by May and Ritzenhofen in PKC 2009 and studied under the assumption that some of Least Significant Bits (LSBs) of certain primes are same. Our strategy can be applied to the implicit factorization problem considering the equality in Most Significant Bits (MSBs). We present new and improved theoretical as well as experimental results in this direction.

Keywords: Greatest Common Divisor, Factorization, Integer Approximations, Implicit Factorization, Lattice, LLL.

1 Introduction

It is known that given two large integers a, b ($a > b$), one can calculate the GCD efficiently in $O(\log^2 a)$ time [STN02, Page 169]. 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 as "approximate common divisors". Using the strategy of [HOW01], Coron and May [COR07] proved the deterministic polynomial time equivalence of computing the RSA secret key and factoring.

In a recent paper [MAY09], May and Ritzenhofen explained the problem of implicit factorization. The motivation of this problem in the context of oracle complexity has nicely been explained in [MAY09]. Thus we directly get into the problem description. Consider $N_1 = p_1q_1, N_2 = p_2q_2, \ldots, N_k = p_kq_k$, 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 $q_1, q_2, \ldots, q_k$; this is followed throughout the paper unless otherwise mentioned. 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. In [MAY09], the result was based under the assumption that some amount of LSBs are same in $p_1, p_2, \ldots, p_k$. Later, we consider in [SAR09], different cases, i.e., (i) some portions of LSBs are same and/or some portions of MSBs are same, (ii) some portions at the middle are same, have been considered. However, the technique of [SAR09] could only be applied for $k = 2$ and it has been pointed out that the extension may not be achieved for $k > 2$.

In this paper we concentrate on the implicit factorization problem considering some amount of MSBs are same in $p_1, p_2, \ldots, p_k$. This is unlike the analysis in [MAY09], where the LSBs were considered. We generalize the ideas of [HOW01] for the lattice based technique that we exploit in this paper and our strategy is different from that of [MAY09,SAR09].

First consider $k = 2$. As p_1, p_2 share certain amount of MSBs, one can write $p_1 - p_2 = x_0$, where x_0 is of lesser bit size than p_1 or p_2. Hence $N_2 = (p_1 - x_0)q_2$. Therefore, $\gcd(N_1, N_2 + x_0 q_2) = p_1$. Since, N_2 (known) is an approximation of $N_2 + x_0 q_2$ (unknown), we can use the technique of [HOW01] to solve this problem efficiently and get p_1 under certain conditions. This gives factorization of N_1. Additionally, when $p_1 > x_0 q_2$, then either $\lfloor \frac{N_2}{p_1} \rfloor$ or $\lceil \frac{N_2}{p_1} \rceil$ will provide q_2. This is explained in detail in Section 2.1.

Next we generalize the PACDP given in [HOW01]. Let $a_1, a_2, \ldots, a_k$ are integers with $\gcd(a_1, a_2, \ldots, a_k) = g$. Suppose $a_2^{(0)}, \ldots, a_k^{(0)}$ are given which are approximations of $a_2, \ldots, a_k$ respectively. We like to find g from the knowledge of $a_1, a_2^{(0)}, \ldots, a_k^{(0)}$. An immediate application of our general version towards the implicit factorization is as follows. We can write $p_1 = p_1 + x_1, \ldots, p_k = p_1 + x_k$ where $x_1 = 0$. Hence $p_1 = \gcd(N_1, N_2 - x_2 q_2, \ldots, N_k - x_k q_k)$ and consequently, we can find $x_2 q_2, \ldots, x_k q_k$ under certain conditions.

For application of the results related to approximate integer common divisor to implicit factorization, we need to generalize the PACDP [HOW01]. Let us describe the problem which we name as Extended Partially Approximate Common Divisor Problem (EPACDP).

Definition 1. EPACDP.
Let $a_1, a_2, \ldots, a_k$ be large integers of same bit size and $g = gcd(a_1, a_2, \ldots, a_k)$. Consider that $a_2^{(0)}, \ldots, a_k^{(0)}$ are the approximations of $a_2, \ldots, a_k$ respectively and $a_2^{(0)}, \ldots, a_k^{(0)}$ are of same bit size too. Let $a_2 = a_2^{(0)} + x_2^{(0)}, \ldots, a_k = a_k^{(0)} + x_k^{(0)}$. We need to find $x_2^{(0)}, \ldots, x_k^{(0)}$ from the knowledge of $a_1, a_2^{(0)}, \ldots, a_k^{(0)}$.

The PACDP of [HOW01] considers only a_1, a_2. For application to implicit factorization, we get one of the integers, denoted by a_1 in Definition 1, exactly and the rest of the integers $a_2, \ldots, a_k$ as approximations.

The problem when a_1 is not available exactly, but an approximation of a_1 is available, is referred as General Approximate Common Divisor Problem (GACDP) in [HOW01]. One can extend it as follows with the name Extended General Approximate Common Divisor Problem (EGACDP). The GACDP of [HOW01] considers only a_1, a_2.

Definition 2. EGACDP.
Let $a_1, a_2, \ldots, a_k$ be large integers of same bit size and $g = gcd(a_1, a_2, \ldots, a_k)$. Consider that $a_1^{(0)}, a_2^{(0)}, \ldots, a_k^{(0)}$ are the approximations of $a_1, a_2, \ldots, a_k$ respectively and $a_1^{(0)}, a_2^{(0)}, \ldots, a_k^{(0)}$ are of same bit size too. Let $a_1 = a_1^{(0)} + x_1^{(0)}, a_2 = a_2^{(0)} + x_2^{(0)}, \ldots, a_k = a_k^{(0)} + x_k^{(0)}$. We need to find $x_1^{(0)}, x_2^{(0)}, \ldots, x_k^{(0)}$ from the knowledge of $a_1^{(0)}, a_2^{(0)}, \ldots, a_k^{(0)}$.

Since for the implicit factorization problem we get a_1 exactly, EPACDP maps directly to it.

Contribution and Roadmap:

- We generalize the partially approximate common divisor problem (PACDP) [HOW01] in Section 2 and show how our generalized version applies to the implicit factorization problem described in [MAY09].
- In Section 2.1, we apply our result for the case $k = 2$ and show when we can achieve better theoretical result than that of [SAR09]. The implicit factorization problem while the MSBs are equal could be handled in [SAR09] only for $k = 2$. We also explain the case for $k = 3$ in Section 2.2 to detail our technique.
- The results of Section 2 are approximated in a closed form expression using a sublattice structure in Section 3. Based on this we present closed form bounds for the solution for implicit factorization when $p_1, \ldots, p_k$ share some amount of MSBs. Our results (requirements of MSBs to be equal) are compared in detail with that of [MAY09] (requirements of LSBs to be equal).
- We also exploit another recently presented technique [DJK09] for calculation of approximate common divisor. Though the theoretical bound of this technique is worse than that of our results in Section 3, we can utilize it for better experimental performance. This is presented in Section 4.
- Moreover, we study the EGACDP, the generalization of GACDP [HOW01] in Section 5.

2 The General Solution for EPACDP

Towards solving the EPACDP, consider the polynomials

$$\begin{aligned} h_2(x_2, \ldots, x_k) &= a_2^{(0)} + x_2, \\ &\ldots\ldots\ldots, \\ h_k(x_2, \ldots, x_k) &= a_k^{(0)} + x_k, \end{aligned} \tag{1}$$

where $x_2, \ldots, x_k$ are the variables. Clearly g (as in Definition 1) divides $h_i(x_2^{(0)}, \ldots, x_k^{(0)})$ for $2 \le i \le k$.

Now let us define the shift polynomials

$$h_{0,\ldots,0,j_2,\ldots,j_k}(x_2, \ldots, x_k) = h_2^{j_2} \ldots h_k^{j_k} a_1^{m-j_2-\ldots-j_k}, \tag{2}$$

for non-negative integers j_i, $2 \le i \le k$ such that $j_2 + \ldots + j_k \le m$ where the integer $m \ge 0$ is fixed.

Further, we define another set of shift polynomials

$$h_{0,\ldots,0,i_n,0,\ldots,0,j_2,\ldots,j_k}(x_2, \ldots, x_k) = x_n^{i_n} h_2^{j_2} \ldots h_k^{j_k}, \tag{3}$$

with the following: (i) $1 \le i_n \le t$, for $2 \le n \le k$ and a positive integer t, and (ii) $j_2+\ldots+j_k = m$, when $0 \le j_2, \ldots, j_{n-1} < i_n$, and $0 \le j_n, \ldots, j_k \le m$.

We also need the following heuristic assumption to proceed.

Assumption 1. Consider a set of polynomials $\{f_2, f_3, \ldots, f_k\}$ on $k-1$ variables having the root of the form $(x_2^{(0)}, \ldots, x_k^{(0)})$. Then we will be able to collect the root efficiently using resultants.

Let us now state the following result due to Howgrave-Graham [HOW97].

Lemma 1. *Let $h(x_2, \ldots, x_k) \in \mathbb{Z}[x]$ be the sum of at most ω monomials. Suppose that $h(x_2^{(0)}, \ldots, x_k^{(0)}) \equiv 0 \bmod N^m$ where $|x_2^{(0)}| \leq X_2, \ldots, |x_k^{(0)}| \leq X_k$ and $||h(x_2X_2, \ldots, x_kX_k)|| < \frac{N^m}{\sqrt{\omega}}$. Then $h(x_2^{(0)}, \ldots, x_k^{(0)}) = 0$.*

We also note that the basis vectors of an LLL-reduced basis fulfill the following property [LLL82,MAY03] (see also [ELN07, Theorem 3.2, Page 22]).

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.

Note that g^m divides any shift polynomial $h_{\ldots}(x_2^{(0)}, \ldots, x_k^{(0)})$. Let $X_2, \ldots, X_k$ be the upper bounds of $x_2^{(0)}, \ldots, x_k^{(0)}$ respectively. Now we define a lattice L using the coefficient vectors of $h_{\ldots}(x_2X_2, \ldots, x_kX_k)$. Let the dimension of L be ω. Under Assumption 1, one gets $x_2^{(0)}, \ldots, x_k^{(0)}$ using lattice reduction over L, if $2^{\frac{\omega(\omega-1)}{4(\omega+2-k)}} \det(L)^{\frac{1}{\omega+2-k}} < \frac{g^m}{\sqrt{\omega}}$. The result follows from Lemma 1 and putting $i = k-1$ in Lemma 2. Neglecting the small constants and considering $k << \omega$ (in fact, we will show that ω is exponential in k in our construction), we get the condition as $\det(L)^{\frac{1}{\omega}} < g^m$, i.e., $\det(L) < g^{m\omega}$. This is written formally in Theorem 1 later.

Before proceeding to the next discussion, we denote that $\binom{n}{r}$ is considered in its usual meaning when $n \geq r \geq 0$ and in all other cases we will consider the value of $\binom{n}{r}$ as 0.

Lemma 3.

$$\omega = \sum_{r=0}^{m} \binom{k+r-2}{r} + \sum_{n=2}^{k} \sum_{i_n=1}^{t} \sum_{r=0}^{n-2} (-1)^r \binom{n-2}{r} \binom{k+m-ri_n-2}{m-ri_n}.$$

Proof. Let $j_2 + \ldots + j_k = r$ where $j_i \geq 0$ for $2 \leq i \leq k$. The number of such solutions is $\binom{k+r-2}{r}$. Hence the number of shift polynomials in Equation 2 is $\omega_1 = \sum_{r=0}^{m} \binom{k+r-2}{r}$.

For fixed n, i_n, the number of shift polynomials in Equation 3 is the number of all possible solutions of $j_2 + \ldots + j_k = m$ for $0 \leq j_2, \ldots, j_{n-1} < i_n$, and $0 \leq j_n, \ldots, j_k \leq m$. Now the number of all possible solutions of $j_2 + \ldots + j_k = m$ for $0 \leq j_2, \ldots, j_{n-1} < i_n$, and $0 \leq j_n, \ldots, j_k \leq m$ is the co-efficient of x^m in $(1 + x + \ldots + x^{i_n-1})^{n-2}(1 + x + \ldots + x^m)^{k-n+1}$ and we denote the coefficient by $c(n, i_n)$, as we have fixed n, i_n. Now $(1+x+\ldots+x^{i_n-1})^{n-2}(1+x+\ldots+x^m)^{k-n+1} = (\frac{1-x^{i_n}}{1-x})^{n-2}(\frac{1-x^{m+1}}{1-x})^{k-n+1} = (1-x^{i_n})^{n-2}(1-x^{m+1})^{k-n+1}(1-$

$x)^{-k+1}$. Hence $c(n, i_n)$ will be the co-efficient of x^m in $(1-x^{i_n})^{n-2}(1-x)^{-k+1}$. We have $(1-x^{i_n})^{n-2} = \sum_{r=0}^{n-2}(-1)^r\binom{n-2}{r}x^{i_n r}$ and $(1-x)^{-k+1} = \sum_{r=0}^{\infty}\binom{k+r-2}{r}x^r$. So, $c(n, i_n) = \sum_{r=0}^{n-2}(-1)^r\binom{n-2}{r}\binom{k+m-ri_n-2}{m-ri_n}$. Hence the number of shift polynomials in Equation 3 is $\omega_2 = \sum_{n=2}^{k}\sum_{i_n=1}^{t} c(n, i_n)$. Finally, $\omega = \omega_1 + \omega_2$ provides the result. □

Lemma 4. *The determinant of L is given by $\det(L) = P_1P_2$ where*

$$P_1 = \prod X_2^{j_2}X_3^{j_3}\ldots X_k^{j_k}a_1^{m-j_2-\ldots-j_k}$$

for non-negative integers j_i, $2 \le i \le k$ such that $j_2 + \ldots + j_k \le m$, and

$P_2 = \prod X_n^{i_n}X_2^{j_2}X_3^{j_3}\ldots X_k^{j_k}$ with the following: (i) $1 \le i_n \le t$, for $2 \le n \le k$ and a positive integer t, and (ii) $j_2 + j_3 + \ldots + j_k = m$, when $0 \le j_2, \ldots, j_{n-1} < i_n$, and $0 \le j_n, \ldots, j_k \le m$.

Proof. The matrix (corresponding to the lattice L) containing the basis vectors is triangular and has the following two kinds of diagonal entries:

$$X_2^{j_2}X_3^{j_3}\ldots X_k^{j_k}a_1^{m-j_2-\ldots-j_k}, \tag{4}$$

for non-negative integers j_i, $2 \le i \le k$ such that $j_2 + \ldots + j_k \le m$ where the integer $m \ge 0$ fixed and

$$X_n^{i_n}X_2^{j_2}X_3^{j_3}\ldots X_k^{j_k}, \tag{5}$$

with the following: (i) $1 \le i_n \le t$, for $2 \le n \le k$ and a positive integer t, and (ii) $j_2 + j_3 + \ldots + j_k = m$, when $0 \le j_2, \ldots, j_{n-1} < i_n$, and $0 \le j_n, \ldots, j_k \le m$.

Clearly P_1 is the product of the elements from (4) and P_2 is the product of the elements from (5). Hence $\det(L) = P_1P_2$. □

The running time of our algorithm is dominated by the runtime of 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 k so the running time of our strategy is $\text{poly}\{\log a_1, \exp(k)\}$. Thus, for small fixed k our algorithm is polynomial in $\log a_1$. Now, we have the following main result.

Theorem 1. *Under Assumption 1, the EPACDP (as in Definition 1) can be solved in poly$\{\log a_1, \exp(k)\}$ time when $\det(L) < g^{m\omega}$, where $\det(L)$ is as in Lemma 4 and ω is as in Lemma 3.*

One may also consider the same upper bound on the errors $x_2^{(0)}, \ldots, x_k^{(0)}$. In that case we get the following result.

Corollary 1. *Considering the same upper bound X on the errors $x_2^{(0)}, \ldots, x_k^{(0)}$, we have* $\det(L) = P_1 P_2$ *where*

$$P_1 = X^{\sum_{r=0}^{m} r\binom{k+r-2}{r}} a_1^{\sum_{r=0}^{m}(m-r)\binom{k+r-2}{r}},$$
$$P_2 = X^{\sum_{n=2}^{k}\sum_{i_n=1}^{t}(i_n+m)\sum_{r=0}^{n-2}(-1)^r\binom{n-2}{r}\binom{k+m-ri_n-2}{m-ri_n}}.$$

Proof. Let $X_2 = X_3 = \ldots = X_k = X$. Then from (4), we have
$X_2^{j_2} X_3^{j_3} \ldots X_k^{j_k} a_1^{m-j_2-\ldots-j_k} = X^{j_2+\ldots+j_k} a_1^{m-j_2-\ldots-j_k}$, for non-negative integers j_i, $2 \le i \le k$ such that $j_2 + \ldots + j_k \le m$. Let $j_2 + \ldots + j_k = r$ where $0 \le r \le m$. The total number of such representation is $\binom{k+r-2}{r}$. Hence

$$P_1 = \prod_{r=0}^{m} (X^r a_1^{m-r})^{\binom{k+r-2}{r}} = X^{\sum_{r=0}^{m} r\binom{k+r-2}{r}} a_1^{\sum_{r=0}^{m}(m-r)\binom{k+r-2}{r}}.$$

For calculating P_2, we have the following constraints: (i) $1 \le i_n \le t$, for $2 \le n \le k$ and a positive integer t, and (ii) $j_2 + j_3 + \ldots + j_k = m$, when $0 \le j_2, \ldots, j_{n-1} < i_n$, and $0 \le j_n, \ldots, j_k \le m$. Thus,

$$P_2 = \prod_{n=2}^{k}\prod_{i_n=1}^{t} X_n^{i_n} X_2^{j_2} X_3^{j_3} \ldots X_k^{j_k} = \prod_{n=2}^{k}\prod_{i_n=1}^{t} X^{i_n c(n,i_n)} X^{mc(n,i_n)}$$
$$= X^{\sum_{n=2}^{k}\sum_{i_n=1}^{t}(i_n+m)\sum_{r=0}^{n-2}(-1)^r\binom{n-2}{r}\binom{k+m-ri_n-2}{m-ri_n}}.$$

□

As the results in this section are quite involved, we present below a few cases for better understanding and comparison with existing results.

2.1 Analysis for $k = 2$

We write the proof of this special case in detail as it shows that this special case is in line of the proof of [COR07, Theorem 3] where the strategy to solve the partially approximate common divisor problem (PACDP) [HOW01] has been exploited. As described in [COP97], after applying LLL, if the output polynomials are of more than one variable, then to collect the roots from these polynomials one needs, Assumption 1. However, in this case, Assumption 1 is not required since there is only one variable in the polynomial that we will consider.

Theorem 2. *Let $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$, where p_1, p_2, q_1, q_2 are primes. Let $N \approx N_1 \approx N_2$, $q_1, q_2 \approx N^\alpha$ and $|p_1 - p_2| < N^\beta$. Then one can factor N_1 and N_2 deterministically in poly*$(\log N)$ *time when $\beta < 1 - 3\alpha + \alpha^2$, provided $2\alpha + \beta \le 1$.*

Proof. Let $x_0 = p_1 - p_2$. We have $N_1 = p_1 q_1$ and $N_2 = p_2 q_2 = (p_1 - x_0) q_2$. Our goal is to recover $x_0 q_2$ from N_1 and N_2. Since $|x_0| < N^\beta$ and $q_2 = N^\alpha$, we can take $X = N^{\alpha+\beta}$ as an upper bound of $x_0 q_2$. Now we consider the shift polynomials

$$\begin{gathered} g_{ij}(x) \;=\; x^i (N_2 + x)^j N_1^{m-j} \\ \text{for } i = 0,\ 0 \leq j \leq m \text{ and } j = m,\ 1 \leq i \leq t, \end{gathered} \tag{6}$$

where m, t are fixed non-negative integers. Clearly, $g_{ij}(x_0 q_2) \equiv 0 \bmod (p_1^m)$.

We construct the lattice L spanned by the coefficient vectors of the polynomials $g_{ij}(xX)$ in (6). One can check that the dimension of the lattice L is $\omega = m+t+1$ and the determinant of L is

$$\det(L) = X^{\frac{(m+t)(m+t+1)}{2}} N_1^{\frac{m(m+1)}{2}} \approx X^{\frac{(m+t)(m+t+1)}{2}} N^{\frac{m(m+1)}{2}}. \tag{7}$$

Here, $P_1 = X^{\frac{m(m+1)}{2}} N^{\frac{m(m+1)}{2}}$ and $P_2 = X^{mt+\frac{t(t+1)}{2}}$ (the general expressions of P_1, P_2 are presented in Lemma 4). 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 $h(x_0 q_2) \equiv 0 \bmod (p_1^m)$. To apply Lemma 1 and Lemma 2 for finding the integer root of $h(x)$, we need

$$2^{\frac{\omega-1}{4}} (\det(L))^{\frac{1}{\omega}} < \frac{p_1^m}{\sqrt{\omega}}. \tag{8}$$

Neglecting small constant terms, we can rewrite (8) as $\det(L) < p_1^{m\omega}$. Substituting the expression of $\det(L)$ from (7) and using $X = N^{\alpha+\beta}, p_1 \approx N^{1-\alpha}$ we get

$$\frac{(m+t)(m+t+1)}{2}(\alpha+\beta) < m((1-\alpha)(m+t+1) - \frac{m+1}{2}). \tag{9}$$

Let $t = \tau m$. Then neglecting the terms of $o(m^2)$ we can rewrite (9) as

$$\psi(\alpha,\beta,\tau) = (-\alpha-\beta)\frac{\tau^2}{2} + (-2\alpha-\beta+1)\tau + (-\frac{3\alpha}{2} - \frac{\beta}{2} + \frac{1}{2}) > 0. \tag{10}$$

The optimal value of τ, to maximize β for a fixed α is $\tau = \frac{1-\beta-2\alpha}{\alpha+\beta}$. Since $\tau \geq 0$ we need

$$1 - \beta - 2\alpha \geq 0. \tag{11}$$

Putting the optimal value of τ in (10), we get

$$\alpha^2 - 3\alpha - \beta + 1 > 0. \tag{12}$$

Once $x_0 q_2$, the integer root of $h(x)$, is known, we get p_1 by calculating the GCD of $N_1, N_2 + x_0 q_2$. As long as $|x_0 q_2| < p_1$, we get q_2 by calculating the floor or ceiling of $\frac{N_2}{p_1}$. As $|x_0 q_2| \leq N^{\alpha+\beta}$ and $p_1 \approx N^{1-\alpha}$, to satisfy $|x_0 q_2| \leq p_1$ we need $2\alpha + \beta \leq 1$ which is incidentally same as (11).

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. □

The relation presented in (9) provides the bound when the lattice parameters m, t are specified. The asymptotic relation independent of the lattice parameters has been presented in (12). This is the theoretical bound and may not be reached in practice as we work with low lattice dimensions. Now let us compare our results with that of [SAR09].

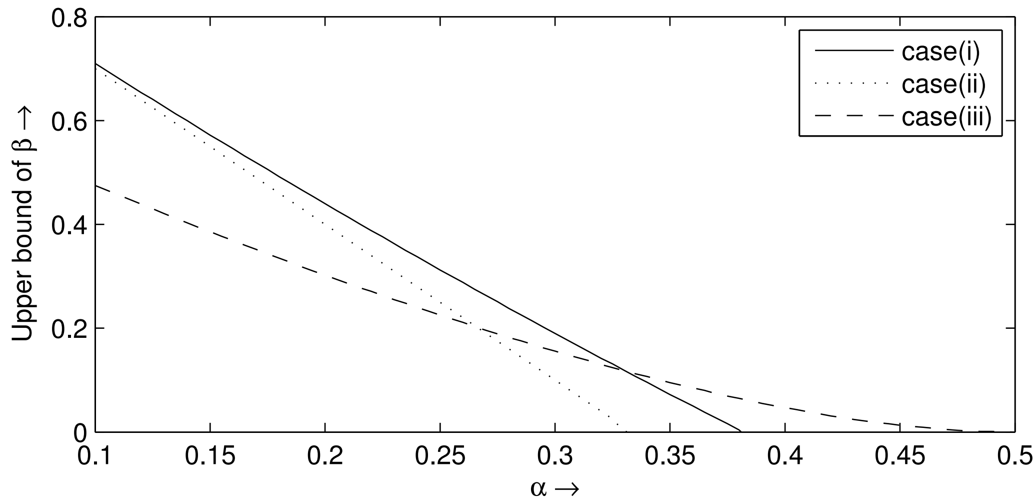

Fig. 1. Comparison of our theoretical result [case (i)] with that of [MAY09] [case (ii)] and [SAR09] [case (iii)].

1. In [SAR09, Theorem 2.1] it has been explained that factorization will be successful when $\Psi(\alpha, \beta) = 4\alpha^2 + 2\alpha\beta + \frac{1}{4}\beta^2 - 4\alpha - \frac{5}{3}\beta + 1 > 0$ provided $1 - \frac{3}{2}\beta - 2\alpha \geq 0$. In our case, the upper bound of β is $\alpha^2 - 3\alpha + 1$. Putting this upper bound of β in Ψ we get $\Psi < 0$ when $\alpha \leq 0.33$. Hence upper bound of β in our case will be greater than of [SAR09] when $\alpha \leq 0.33$.
2. The heuristic presented in [SAR09] is polynomial time in $\log N$ (based on Assumption 1 in [SAR09, Introduction]), while our algorithm is deterministic polynomial time in $\log N$.
3. The result of [SAR09] could not be extended for $k > 2$, but our result can be extended for general k.
4. We get similar quality of experimental results as in [SAR09] and both the experimental results (i.e., our and that of [SAR09]) almost coincide with our theoretical results. Our experimental results are same as our theoretical results following (9) for specific lattice dimensions, whereas the experimental results of [SAR09] are better than the theoretical results of [SAR09] as explained in [SAR09, Remark 2].

We also like to compare our result with that of [MAY09]. The strategy of [MAY09] considers equality in some LSBs of p_1, p_2 and we consider the equality in some MSBs. The strategy of [MAY09] works when $\beta \leq 1 - 3\alpha$. In our case, $\beta < 1 - 3\alpha + \alpha^2$. It is thus immediate to see that our upper bound is better than that of [MAY09]. Given α, the amount of bit sharing is $(1-\alpha-\beta)\log_2 N$. Thus, for $k=2$, we need smaller number of bit sharing in MSBs for implicit factorization than the number of bit sharing in LSBs achieved in [MAY09]. Later in Section 3, we will compare our results with that of [MAY09] for all $k \geq 2$.

One may refer to Figure 1 for the comparison of the theoretical results.

2.2 Analysis for $k = 3$

As we have already pointed out, the idea of [SAR09] could not be extended for $k > 2$. However, our technique works in the general case. We now explain the case for $k = 3$ in detail.

Theorem 3. *Let $N_1 = p_1q_1, N_2 = p_2q_2$ and $N_3 = p_3q_3$, where p_1, p_2, p_3, and q_1, q_2, q_3 are primes. Let N, N_1, N_2, N_3 be of same bit size and $q_1, q_2, q_3 \approx N^{\alpha}$, $|p_1 - p_2| < N^{\beta}$, $|p_1 - p_3| < N^{\beta}$. Then, under Assumption 1, one can factor N_1, N_2 and N_3 in poly$(\log N)$ time when*

$$\beta < (1-\alpha)^{\frac{3}{2}} - \alpha, \text{ provided } 2\alpha + \beta \leq 1.$$

Proof. Let $x_0 = p_2 - p_1$ and $y_0 = p_3 - p_1$. We have $N_1 = p_1q_1, N_2 = p_2q_2 = (x_0 + p_1)q_2, N_3 = (y_0 + p_1)q_3$. Our goal is to recover x_0q_2, y_0q_3 from N_1, N_2 and N_3. Let $X = N^{\alpha+\beta}$. Clearly X is an upper bound of x_0q_2, y_0q_3. Also we have $p_1 \approx N^{1-\alpha}$. When $k = 3$ then $P_1 = X^{\frac{m^3}{3}+o(m^3)}N^{\frac{m^3}{6}+o(m^3)}$.

Let $t = \tau m$. To have a manageable formula for P_2, we need to assume $t \leq m + 1$. Then $P_2 = X^{m^3\tau^2+m^3\tau+\frac{m^3\tau^3}{3}+o(m^3)}$ and $\omega = \frac{m^2}{2} + m^2\tau + \frac{m^2\tau^2}{2} + o(m^2)$.

Neglecting the $o(m^3)$ terms, the required condition $\det(L) < p_1^{m\omega}$ implies $(\frac{1}{3} + \tau^2 + \tau + \frac{\tau^3}{3})(\alpha + \beta) + \frac{1}{6} < (1-\alpha)(\frac{1}{2} + \tau + \frac{\tau^2}{2})$, i.e.,

$$-\frac{1}{3}\tau^3\alpha - \frac{1}{3}\tau^3\beta - \frac{3}{2}\tau^2\alpha - \tau^2\beta + \frac{1}{2}\tau^2 - 2\tau\alpha - \tau\beta + \tau - \frac{5}{6}\alpha - \frac{1}{3}\beta + \frac{1}{3} > 0. \qquad (13)$$

To maximize β for a fixed α, the optimal value of τ is $\frac{1-2\alpha-\beta}{\alpha+\beta}$. Putting this optimal value of τ in (13), we get the required condition as $-\alpha^3 + 2\alpha^2 - 2\alpha\beta - \beta^2 - 3\alpha + 1 > 0$, i.e., $\beta < \sqrt{1 - 3\alpha + 3\alpha^2 - \alpha^3} - \alpha$. As $\tau \geq 0$, we also need the constraint $2\alpha + \beta \leq 1$. Then under Assumption 1 (as the polynomials are of two variables), we can collect the roots successfully. □

3 Sublattice and Generalized Bound

In this section, we study a sublattice L' of the lattice L explained in the previous section. This helps in two ways as follows.

- The dimension of the sublattice L' is less than that of L and this helps in actual experiments.
- The theoretical analysis helps us to get a generalized bound for β.

We need the following technical result that will be used later.

Lemma 5. *For any fixed positive integer $r \geq 1$, and a large integer m,* $\sum_{t=1}^{m} t^r = \frac{m^{r+1}}{r+1} + o(m^{r+1})$.

Proof. Let $S = 1^r + 2^r + \ldots + m^r$. Then, $\int_0^m x^r\,dx < S < \int_1^{m+1} x^r\,dx$. Thus, $\frac{m^{r+1}}{r+1} < S < \frac{(m+1)^{r+1}-1}{r+1}$, which gives, $\frac{m^{r+1}}{r+1} < S < \frac{(m+1)^{r+1}}{r+1}$. Now, $\frac{(m+1)^{r+1}}{r+1} - \frac{m^{r+1}}{r+1}$ contains the terms m^i for $i \leq r$. Thus, for a fixed r and large m, one can write $S = \frac{m^{r+1}}{r+1} + o(m^{r+1})$. □

Now we present the main result describing the bound on β.

Theorem 4. *Consider EPACDP with $g \approx a_1^{1-\alpha}$ and $x_2^{(0)} \approx \ldots \approx x_k^{(0)} \approx a_1^{\alpha+\beta}$. Then, under Assumption 1, one can solve EPACDP in poly$\{\log a, \exp(k)\}$ time when*

$$\beta < \frac{k^2 + 5\alpha k - 2\alpha k^2 - 2\alpha - 2k + 1 - \sqrt{k^2 + 2\alpha^2 k - \alpha^2 k^2 - 2k + 1}}{k^2 - 3k + 2} \text{ for } k > 2 \text{ and}$$
$$< 1 - 3\alpha + \alpha^2, \text{ for } k = 2,$$

with the constraint $2\alpha + \beta \leq 1$.

Proof. We start by explaining the shift polynomials. First we consider the following ones which are same as given in (2) in the previous section.

$$h_{0,\ldots,0,j_2,\ldots,j_k}(x_2, \ldots, x_k) = h_2^{j_2} \ldots h_k^{j_k} a_1^{m-j_2-\ldots-j_k}, \tag{14}$$

for non-negative integers j_i, $2 \leq i \leq k$ such that $j_2 + \ldots + j_k \leq m$ where the integer $m \geq 0$ fixed.

Further, we define another set of shift polynomials which is a sub-collection of the polynomials presented in the last section in (3),

$$h_{i_2,0,\ldots,0,0,\ldots,0,j_2,\ldots,j_k}(x_2, \ldots, x_k) = x_2^{i_2} h_2^{j_2} \ldots h_k^{j_k}, \tag{15}$$

with the following: (i) $1 \leq i_2 \leq t$, for a positive integer t, and (ii) $j_2 + \ldots + j_k = m$, and $j_i \geq 0$ for $2 \leq i \leq k$. Next we define a lattice L' using the coefficient vectors of $h_{\ldots}(x_2X_2, \ldots, x_kX_k)$.

Let $X_2 = X_3 = \ldots = X_k = X$, the common upper bound. The shift polynomials from (14) contribute $P_1' = \prod_{r=0}^{m} (X^r a_1^{m-r})^{\binom{k+r-2}{r}} = X^{\sum_{r=0}^{m} r\binom{k+r-2}{r}} a_1^{\sum_{r=0}^{m}(m-r)\binom{k+r-2}{r}}$ to the determinant of L'. (This P_1' is same as P_1 in Corollary 1.) The shift polynomials from (15) contribute $P_2' = \prod_{i_2=1}^{t} (X^{i_2} X^m)^{\binom{k+m-2}{m}} = X^{\sum_{i_2=1}^{t}(i_2+m)\binom{k+m-2}{m}}$ to the determinant of L'. The dimension of L' is $\omega' = \sum_{r=0}^{m} \binom{k+r-2}{r} + t\binom{m+k-2}{m}$.

Now, $\binom{k+r-2}{r} = \frac{(r+1)\dots(r+k-2)}{(k-2)!} = \frac{r^{k-2}}{(k-2)!} + o(r^{k-2})$. Then,

$P_1 \approx X^{\sum_{r=0}^{m} r \frac{r^{k-2}}{(k-2)!}} a_1^{\sum_{r=0}^{m}(m-r)\frac{r^{k-2}}{(k-2)!}} \approx X^{\frac{1}{(k-2)!}\frac{m^k}{k}} a_1^{\frac{1}{(k-2)!}\frac{m^k}{k-1} - \frac{1}{(k-2)!}\frac{m^k}{k}}$, using Lemma 5 and neglecting the lower order terms. Moreover,

$P_2 = X^{\sum_{i_2=1}^{t}(i_2+m)\frac{m^{k-2}}{(k-2)!}} \approx X^{\frac{1}{(k-2)!}(\frac{t^2 m^{k-2}}{2} + tm^{k-1})}$, (neglecting the lower order terms). Further,

$\omega' \approx \sum_{r=0}^{m} \frac{r^{k-2}}{(k-2)!} + t\frac{m^{k-2}}{(k-2)!} \approx \frac{m^{k-1}}{(k-1)(k-2)!} + t\frac{m^{k-2}}{(k-2)!}$, (using Lemma 5 and neglecting the lower order terms). Following Lemma 1, the required condition is $\det(L') = P_1' P_2' < g^{m\omega'}$, where g is the common divisor. Let $g = a_1^{1-\alpha}, X = a_1^{\alpha+\beta}$. Then putting the values of g, X in $\det(L') = P_1' P_2' < g^{m\omega'}$, we get,

$$(\frac{m^k}{k} + \frac{m^{k-2}t^2}{2} + m^{k-1}t)(\alpha+\beta) + \frac{m^k}{k-1} - \frac{m^k}{k} < (1-\alpha)(m^{k-1}t + \frac{m^k}{k-1}). \tag{16}$$

Now putting $t = \tau m$, ($\tau \geq 0$ is a real number) in (16), we get the condition as

$$(\frac{1}{k} + \frac{\tau^2}{2} + \tau)(\alpha+\beta) + \frac{1}{(k-1)k} < (1-\alpha)(\tau + \frac{1}{k-1}). \tag{17}$$

To maximize β for a fixed α, the optimal value of τ is $\tau = \frac{1-2\alpha-\beta}{\alpha+\beta}$. Putting this optimal value in (17), we get the condition as $4\alpha^2k^2 + 4\alpha\beta k^2 + \beta^2k^2 - 8\alpha^2k - 10\alpha\beta k - 3\beta^2k - 4\alpha k^2 - 2\beta k^2 + 2\alpha^2 + 4\alpha\beta + 2\beta^2 + 6\alpha k + 4\beta k + k^2 - 2\alpha - 2\beta - k > 0$. From which we get the required condition as $\beta < \frac{k^2+5\alpha k-2\alpha k^2-2\alpha-2k+1-\sqrt{k^2+2\alpha^2k-\alpha^2k^2-2k+1}}{k^2-3k+2}$ when $k > 2$ and $\beta < 1 - 3\alpha + \alpha^2$ when $k = 2$. Since $\tau \geq 0$, we also need the constraint $1 - 2\alpha - \beta \geq 0$. Then under Assumption 1 (as the polynomials are of more than one variable), we can collect the roots successfully. □

3.1 Comparison with the work of [MAY09]

Let us now compare our result with that of [MAY09] for the general case. The strategy of [MAY09] considers equality in some LSBs of $p_1, p_2, \dots, p_k$ and we consider the equality in some MSBs. Still, we present the comparison as our problem is not studied for general k earlier and the only related work has been presented in [MAY09].

The strategy of [MAY09] works when $\beta \leq 1 - \alpha - \frac{k}{k-1}\alpha = 1 - \frac{2k-1}{k-1}\alpha$. As $\beta > 0$, one may note $\alpha < \frac{k-1}{2k-1}$, i.e, $\alpha < \frac{1}{2}$.

We have already discussed in Section 2.1 that for the case $k = 2$ our result is better than that of [MAY09]. The results of Theorem 4 for $k = 2$ and Theorem 2 are same, since L and L' are same for $k = 2$. However, L and L' become different for $k > 2$.

For $k > 2$, $\frac{k^2+5\alpha k-2\alpha k^2-2\alpha-2k+1-\sqrt{k^2+2\alpha^2k-\alpha^2k^2-2k+1}}{k^2-3k+2} > 1 - \frac{2k-1}{k-1}\alpha$. Thus, for any k, $k \geq 2$, we need smaller amount of bit sharing in MSBs for implicit factorization than the number of bit sharing in LSBs achieved in [MAY09]. Our upper bound on β is $\frac{k-1-\sqrt{k^2+2\alpha^2k-\alpha^2k^2-2k+1}}{k^2-3k+2}$ more than the upper bound on β in [MAY09]. Thus, the gap between our bound and that of [MAY09] reduces as k increases.

In summary, we have the following observations.

1. Our theoretical result is better than that of [MAY09] from the point that it requires less MSBs to be equal than the number of LSBs in case of [MAY09].
2. Both our result as well as that of [MAY09, Theorem 7] are of time complexity poly$\{\log N, exp(k)\}$. However, the lattice dimension in the formulation of [MAY09] is much smaller (exactly k) than the lattice dimension following our approach (exponential in k). Experimentally our results provide superior outcome for $k = 3$ and similar kind of outcome for $k = 4$, though we need more time than that of [MAY09]. Experiments for large k is not possible with our strategy in this section. To overcome this, later in Section 4, we present a technique that provides results for larger values of k.
3. The strategy of [MAY09] could be extended for balanced RSA moduli, which we could not achieve in our case.

		No. of shared LSBs [MAY09] in p_i				No. of shared MSBs (our) in p_i			
k	Bitsize of p_i, q_i $(1-\alpha)\log_2 N, \alpha \log_2 N$	Theory $\frac{k}{k-1}\alpha \log_2 N$	Expt.	LD	Time (sec)	Theory	Expt.	LD	Time (sec)
3	750, 250	375	378	3	< 10	352	367	56	48.63
* 3	700, 300	450	452	3	< 1	416	431	56	69.48
* 3	650, 350	525	527	3	< 1	478	499	56	87.51
# 3	600, 400	600	-	-	-	539	562	56	116.77
* 4	750, 250	334	336	4	< 1	320	334	65	34.94
* 4	700, 300	400	402	4	< 1	380	400	65	38.01
* 4	650, 350	467	469	4	< 1	439	471	65	52.75
* 4	600, 400	534	535	4	< 1	497	528	65	84.15

Table 1. For 1000 bit N, theoretical and experimental data of the number of shared LSBs in [MAY09] and shared MSBs in our case. LD means Lattice Dimension.

Let us now present some numerical values (both theoretical as well as experimental) for comparison with [MAY09] in Table 1. We have implemented the programs 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. In the * marked rows, experimental data is not available from [MAY09], and we perform the experiments following the method of [MAY09]. In the # marked row, the method of [MAY09] does not work as all the bits of the primes p_1, p_2, p_3 need to be same.

4 Method for Improved Results for Larger Values of k

In [DJK09, Section 5.2], the authors studied the EPACDP for analysing the security of their scheme. Based on the idea presented in [DJK09], we get Theorem 5. The result in Theorem 5 below is not exactly presented in a similar form in [DJK09].

One can write,

$$\begin{aligned} a_1 &= gq_1, \\ a_2^{(0)} &= gq_2 - x_2^{(0)}, \\ &\ldots, \\ a_k^{(0)} &= gq_k - x_k^{(0)}. \end{aligned}$$

Let $M = \begin{pmatrix} 2^{\rho} & a_2^{(0)} & a_3^{(0)} & \dots & a_k^{(0)} \\ 0 & -a_1 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & -a_1 \end{pmatrix}$, where $2^{\rho} \approx x_2^{(0)}$. One can note that $(q_1, q_2, \dots, q_k) \cdot M = (2^{\rho} q_1, -q_1 x_2^{(0)}, \dots, q_1 x_k^{(0)}) = b$, say. It can be checked that

$$||b|| < \sqrt{k} a^{2\alpha+\beta}. \tag{18}$$

Moreover, $|\det(M)| = 2^{\rho} a_1^{k-1} \approx a^{\alpha+\beta+k-1}$. We know that there is a vector v in the lattice L corresponding to M such that

$$||v|| < \sqrt{k} a^{\frac{\alpha+\beta+k-1}{k}}, \tag{19}$$

following Minkowski's theorem (see [RGV04] for more details). Now we consider the following assumption.

Assumption 2. The vector b is a shortest vector and the next shortest vector is significantly larger than $||b||$.

Under this assumption, and from (18), (19), we get b from L if

$$a^{2\alpha+\beta} < a^{\frac{\alpha+\beta+k-1}{k}}.$$

From which we get $\beta < \frac{k-1+\alpha-2\alpha k}{k-1}$. The running time is determined by the time to calculate a shortest vector in L which is polynomial in $\log a$ but exponential in k.

Thus, we get the following result.

Theorem 5. *Consider EPACDP with $g \approx a_1^{1-\alpha}$ and $x_2^{(0)} \approx x_k^{(0)} \approx a_1^{\alpha+\beta}$. Then, under Assumption 2, one can solve EPACDP in poly$\{\log a, \exp(k)\}$ time when,*

$$\beta < 1 - \frac{2k-1}{k-1}\alpha. \tag{20}$$

The formula presented in (20) for the MSB case is exactly same as that of [MAY09, Theorem 7] in LSB case. Further, from the analysis presented in Section 3.1, it is clear that this bound is worse than the bound presented in Theorem 4 in Section 3. However, this result helps us to provide much better experimental performance for larger values of k, that could not be achieved by the method in Section 3.

The statement of Theorem 5 states that the time complexity is poly$\{\log a, \exp(k)\}$. However, under the assumption that "the shortest vector of the lattice L can be found by the LLL algorithm", the complexity becomes poly$\{\log a, k\}$. This happens in practice as observed in [MAY09] too. Below we present the experimental results and compare that with the results presented in [MAY09, Table 1, Section 6.2]. One may note that both our results and the results of [MAY09] are of similar quality. This we have lacked with our earlier method presented in Section 3. The running time of all our experiments are less than 50 seconds in our platform described in the previous section.

α	k	Theoretical bound	Experiments ([MAY09])	Experiments (our)
0.25	3	375	378	377
0.35	10	389	391	390
0.40	100	405	410	408
0.44	50	449	453	452
0.48	100	485	492	489

Table 2. For 1000 bit N, theoretical (same bound for [MAY09] and in our case) and experimental data of the number of shared LSBs in [MAY09] and shared MSBs in our case.

So far we have discussed the methods towards applying them in implicit factorization. Now, let us compare the method presented in this section with that of Section 3 for EPACDP itself.

Let $g \approx a^{1-\alpha}$ and $x_i \approx a^{\beta}$ for $i = 2$ to k. Following similar kind of calculations as in the proof of Theorem 4, we get

$$\begin{aligned} \beta &< 1 - 2\alpha + \alpha^2 \text{ for } k = 2 \text{ and} \\ &< \frac{1 - 2k + 2\alpha k + k^2 - \alpha k^2 - \sqrt{k^2 + 2\alpha^2 k - \alpha^2 k^2 - 2k + 1}}{k^2 - 3k + 2} \text{ for } k > 2. \end{aligned} \tag{21}$$

The approach in this section provides the bound as

$$\beta < 1 - \frac{k}{k-1}\alpha.$$

When $\alpha \geq \frac{k-1}{k}$, we can not get the common divisor by the proposed method in this section. However, we get results in such situations using the results of Section 3. In Table 3, we present few such examples.

k	$\alpha = \frac{k-1}{k}$	Bound of β		LD	Time
		Inequality (21)	Experimental		
3	$\frac{2}{3}$	0.1835	0.135	25	10.72
4	$\frac{3}{4}$	0.1464	0.09	28	28.75

Table 3. Experimental results for 1000-bit $a_i^{(0)}$.

To conclude this section, we like to point out the following issues.

- Theoretically the idea of Section 3 is always better than that of this section.
- The method of this section works better than the idea of Section 3 experimentally for larger values of k.
- There are some situations for small values of k, where the idea of Section 3 works better than the strategy in this section.

5 EGACDP

So far we have concentrated on EPACDP, i.e., we considered that the first item is exactly known. However, the more general problem is when the first item is also not known exactly and some approximation is available. This is what we study in this section and we like to refer to EGACDP (see Definition 2). Towards solving EPACDP, we presented two different techniques, one in Section 2 and another in Section 4. We will try similar methods in this section as Method I and Method II respectively. In EGACDP, we have,

$$\begin{aligned} a_1^{(0)} &= gq_1 - x_1^{(0)}, \\ a_2^{(0)} &= gq_2 - x_2^{(0)}, \\ &\ldots, \\ a_k^{(0)} &= gq_k - x_k^{(0)}, \end{aligned}$$

where $a_1^{(0)}, \ldots, a_k^{(0)}$ are known. We want to find g from $a_1^{(0)}, \ldots, a_k^{(0)}$.

5.1 Method I

Towards solving the EGACDP in a manner similar to Section 2, consider the polynomials

$$\begin{aligned} h_1(x_1, x_2, \ldots, x_k) &= a_1^{(0)} + x_1, \\ &\ldots\ldots\ldots, \\ h_k(x_1, x_2, \ldots, x_k) &= a_k^{(0)} + x_k, \end{aligned} \tag{22}$$

where $x_1, x_2, \ldots, x_k$ are the variables. Clearly g (of Definition 2) divides $h_i(x_1^{(0)}, x_2^{(0)}, \ldots, x_k^{(0)})$ for $1 \leq i \leq k$.

Now let us define the shift polynomials

$$h_{s_1,\ldots,s_k}(x_1, x_2, \ldots, x_k) = h_1^{s_1} \ldots h_k^{s_k}, \tag{23}$$

for non-negative integers u, m such that $u \leq s_1 + \ldots + s_k \leq m$ where the integers $u, m \geq 0$ are fixed.

Let $X_1, \ldots, X_k$ be the upper bounds of $x_1^{(0)}, \ldots, x_k^{(0)}$ respectively. Now we define a lattice L using the coefficient vectors of $h_{\ldots}(x_1X_1, \ldots, x_kX_k)$. Let the dimension of L be ω. One gets $x_1^{(0)}, \ldots, x_k^{(0)}$ (under Assumption 1 and following Lemma 1 and Lemma 2) using lattice reduction over L, if $\det(L)^{\frac{1}{\omega}} < g^m$, i.e., when $\det(L) < g^{m\omega}$ (neglecting the lower order terms).

Since the lattice dimension $\omega = \sum_{s=u}^{m} \binom{k+s-1}{s}$ is exponential in k, the running time of this strategy will be poly$\{\log a_1, \exp(k)\}$. Thus, for small fixed k this algorithm is polynomial in $\log a_1$. To summarize we get the following result.

Theorem 6. *Under Assumption 1, the EGACDP can be solved in poly*$\{\log a_1, \exp(k)\}$ *time when* $\det(L) < g^{m\omega}$.

Since in this case matrix corresponding to the Lattice L is not square, finding $\det(L)$ may not be easy for general k. Further, for large k, dimension of L will be very large. Experimental results corresponding to this idea are presented in Table 4.

5.2 Method II

Here we follow the idea of Section 4. We have

$$\begin{aligned} a_1^{(0)} &= gq_1 - x_1^{(0)}, \\ a_2^{(0)} &= gq_2 - x_2^{(0)}, \\ &\ldots, \\ a_k^{(0)} &= gq_k - x_k^{(0)}, \end{aligned}$$

where $a_1^{(0)}, \ldots, a_k^{(0)}$ are known and $a_i^{(0)} \approx a$ for $1 \leq i \leq k$. Suppose, $x_i^{(0)} \approx a^\beta$ for $1 \leq i \leq k$ and $g \approx a^{1-\alpha}$. Then $q_i \approx a^\alpha$ for $i \in [1, k]$.

Let $M = \begin{pmatrix} 2^\rho & a_2^{(0)} & a_3^{(0)} & \ldots & a_k^{(0)} \\ 0 & -a_1^{(0)} & 0 & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & -a_1^{(0)} \end{pmatrix}$, where $2^\rho \approx 2x_1^{(0)}$. One can note that $(q_1, q_2, \ldots, q_k) \cdot M = (2^\rho q_1, x_1^{(0)} q_2 - q_1 x_2^{(0)}, \ldots, x_1^{(0)} q_k - q_1 x_k^{(0)}) = b$, say.

It can be checked that

$$||b|| < 2\sqrt{k} a^{\alpha+\beta}. \tag{24}$$

Moreover, $|\det(M)| = 2^\rho (a_1^{(0)})^{k-1} \approx 2a^{\beta+k-1}$. Following Minkowski's theorem, there is a vector v in the lattice L corresponding to M such that

$$||v|| < \sqrt{k} 2^{\frac{1}{k}} a^{\frac{\beta+k-1}{k}}. \tag{25}$$

Under Assumption 2 (Section 4), and from (24), (25), one can obtain b from L if

$$a^{\alpha+\beta} < a^{\frac{\beta+k-1}{k}},$$

neglecting the terms 2 and $2^{\frac{1}{k}}$; from which we get $\beta < 1 - \frac{k}{k-1}\alpha$. With the knowledge of b, one can find q_1, from which g is obtained if $x_1^{(0)} \leq q_1$. So we need $1 - \frac{k}{k-1}\alpha \leq \alpha$, i.e., $\alpha \geq \frac{k-1}{2k-1}$. The running time is determined by the time to calculate a shortest vector in L which is polynomial in $\log a$ but exponential in k.

Thus, we get the following result.

Theorem 7. *Consider EGACDP with $g \approx a_1^{1-\alpha}$ and $x_1^{(0)} \approx x_2^{(0)} \approx \ldots \approx x_k^{(0)} \approx a_1^\beta$. Then, under Assumption 2, one can solve EGACDP in poly$\{\log a, \exp(k)\}$ time when,*

$$\beta < 1 - \frac{k}{k-1}\alpha, \tag{26}$$

provided $\alpha \geq \frac{k-1}{2k-1}$.

5.3 Experimental Results

In this section we present a few experimental results for both the methods. When $k = 3$ and $1 - \alpha = 0.25$, we have $1 - \frac{k}{k-1}\alpha < 0$. In such a situation one can not get results using Method II, but the Method I will succeed. As example, Method I succeeds given 250-bit g for $k = 3, 4$, whereas Method II does not provide results in such a scenario.

For large values of k, we can not perform experiments corresponding to Method I due to high lattice dimensions. The statement of Theorem 7 states that the time complexity is poly$\{\log a, \exp(k)\}$. However, under the assumption that "the shortest vector of the lattice L can be found by the LLL algorithm", the complexity becomes poly$\{\log a, k\}$. Thus, Method II will work successfully and we can easily obtain the experimental results using LLL up to $k \leq 100$.

k	g	error	LD	Time (in sec.)
3	250-bit	36-bit	31	11.72
3	500-bit	245-bit	31	2.85
4	250-bit	82-bit	65	210.91
4	500-bit	320-bit	65	63.04

Method I.

k	g	error	LD	Time (in sec.)
3	500-bit	249-bit	3	< 1
4	500-bit	331-bit	4	< 1
10	500-bit	441-bit	10	< 1
50	500-bit	487-bit	50	4.62
100	500-bit	490-bit	100	40.17

Method II.

Table 4. Experimental results for 1000-bit $a_i^{(0)}$.

6 Conclusion

In this paper we present a generalization of the partially approximate common divisor problem (PACDP) [HOW01] which we term as Extended Partially Approximate Common Divisor Problem (EPACDP). This problem immediately relates to the implicit factorization problem introduced in [MAY09]. We consider the case when some MSBs of the primes $p_1, p_2, \ldots, p_k$ are equal (but unknown) as opposed to the case when the LSBs are equal (but unknown) in [MAY09]. Our strategy provides new and improved theoretical as well as experimental results. We also study the extension of GACDP (General Approximate Common Divisor Problem) [HOW01].

References

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

[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.

[DJK09] M. v. Dijk, C. Gentry, S. Halevi and V. Vaikuntanathan. Fully Homomorphic Encryption over the Integers. Cryptology ePrint Archive, Report 2009/616, Available at http://eprint.iacr.org/2009/616.

[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.

[ELN07] E. Jochemsz. Cryptanalysis of RSA Variants Using Small Roots of Polynomials. Ph. D. thesis, Technische Universiteit Eindhoven, 2007.

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

[MAY03] A. May. New RSA Vulnerabilities Using Lattice Reduction Methods. PhD thesis, University of Paderborn, 2003.

[MAY09] A. May and M. Ritzenhofen. Implicit factoring: on polynomial time factoring given only an implicit hint. Proceedings of PKC 2009, Lecture Notes in Computer Science, Volume 5443, pages 1–14, Springer, 2009.

[RGV04] O. Regev. Lattices in Computer Science (Lecture Notes), 2004. Available at: http://www.cs.tau.ac.il/˜ odedr/teaching/lattices_fall_2004/index.html [last accessed December 19, 2009].

[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.

[STN02] D. R. Stinson. Cryptography - Theory and Practice. 2nd Edition, Chapman & Hall/CRC, 2002.