

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 in a general framework considering the equality of (i) Most Significant Bits (MSBs), (ii) Least Significant Bits (LSBs) and (iii) MSBs and LSBs together. We present new and improved theoretical as well as experimental results in comparison with the state of the art works in this area.

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 $N_1 = p_1q_1, N_2 = p_2q_2, \dots, N_k = p_kq_k$, where p_1, p_2, \dots, p_k and q_1, q_2, \dots, q_k are primes. It is also considered that p_1, p_2, \dots, p_k are of same bit size and so are q_1, q_2, \dots, q_k ; this is followed throughout the paper unless otherwise mentioned. Given that certain portions of bit pattern in p_1, p_2, \dots, p_k are common, the question is under what conditions it is possible to factor N_1, N_2, \dots, N_k efficiently. In [MAY09], the result was based under the assumption that some amount of LSBs are same in p_1, p_2, \dots, p_k . Later, in [SAR09], different cases has been considered, 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 $k = 2$. Recently in [FAU10], the results

of [SAR09] has been improved. Note that the work of [FAU10] and certain part of this work (Sections 2, 3, except Section 3.1) [SAR10] are done independently at the similar time.

In this paper we concentrate on the implicit factorization problem considering some amount of (i) MSBs or (ii) LSBs or (iii) MSBs as well as LSBs are same in p_1, p_2, \dots, p_k . Thus the work of [MAY09] (the LSB case) as well as that of [FAU10] (the work for MSB case) are considered in the same framework. Further, the technique takes care of the case considering some portions of MSBs and LSBs together. Our work is of the same quality (in general) and slightly improved (in certain cases) in comparison to [MAY09,FAU10]. 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,FAU10].

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, $\gcd(N_1, N_2 + x_0q_2) = p_1$. Since, N_2 (known) is an approximation of $N_2 + x_0q_2$ (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 $p_1 > x_0q_2$, 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, \dots, a_k are integers with $\gcd(a_1, a_2, \dots, a_k) = g$. Suppose $a_2^{(0)}, \dots, a_k^{(0)}$ are given which are approximations of a_2, \dots, a_k respectively. We like to find g from the knowledge of $a_1, a_2^{(0)}, \dots, a_k^{(0)}$. An immediate application of our general version towards the implicit factorization is as follows. We can write $p_1 = p_1 + x_1, \dots, p_k = p_1 + x_k$ where $x_1 = 0$. Hence $p_1 = \gcd(N_1, N_2 - x_2q_2, \dots, N_k - x_kq_k)$ and consequently, we can find x_2q_2, \dots, x_kq_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, \dots, a_k be large integers of same bit size and $g = \gcd(a_1, a_2, \dots, a_k)$. Consider that $a_2^{(0)}, \dots, a_k^{(0)}$ are the approximations of a_2, \dots, a_k respectively and $a_2^{(0)}, \dots, a_k^{(0)}$ are of same bit size too. Let $a_2 = a_2^{(0)} + x_2^{(0)}, \dots, a_k = a_k^{(0)} + x_k^{(0)}$. We need to find $x_2^{(0)}, \dots, x_k^{(0)}$ from the knowledge of $a_1, a_2^{(0)}, \dots, a_k^{(0)}$.

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, \dots, 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, \dots, a_k be large integers of same bit size and $g = \gcd(a_1, a_2, \dots, a_k)$. Consider that $a_1^{(0)}, a_2^{(0)}, \dots, a_k^{(0)}$ are the approximations of a_1, a_2, \dots, a_k respectively and $a_1^{(0)}, a_2^{(0)}, \dots, a_k^{(0)}$

are of same bit size too. Let $a_1 = a_1^{(0)} + x_1^{(0)}, a_2 = a_2^{(0)} + x_2^{(0)}, \dots, a_k = a_k^{(0)} + x_k^{(0)}$. We need to find $x_1^{(0)}, x_2^{(0)}, \dots, x_k^{(0)}$ from the knowledge of $a_1^{(0)}, a_2^{(0)}, \dots, a_k^{(0)}$.

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. For simplicity we initiate our discussion with the case (i) when the primes p_1, \dots, p_k share some amount of MSBs. However, a little modification of our strategy (presented in Section 3.1) shows that the method works as well for the cases (ii) when the primes p_1, \dots, p_k share some amount of LSBs and also (iii) when the primes p_1, \dots, p_k share some amount of MSBs as well as LSBs. Based on this we present closed form bounds for the solution for implicit factorization when p_1, \dots, p_k share some amount of bits. To show the improvements, our results are compared in detail with that of [MAY09, SAR09, FAU10].
- 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

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

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

Now let us define the shift polynomials

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

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

Further, we define another set of shift polynomials

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

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

We also need the following heuristic assumption to proceed.

Assumption 1. Consider a set of polynomials $\{f_2, f_3, \dots, f_k\}$ on $k - 1$ variables having the root of the form $(x_2^{(0)}, \dots, x_k^{(0)})$. 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 $h(x_2, \dots, x_k) \in \mathbb{Z}[x]$ be the sum of at most ω monomials. Suppose that $h(x_2^{(0)}, \dots, x_k^{(0)}) \equiv 0 \pmod{N^m}$ where $|x_2^{(0)}| \leq X_2, \dots, |x_k^{(0)}| \leq X_k$ and $||h(x_2 X_2, \dots, x_k X_k)|| < \frac{N^m}{\sqrt{\omega}}$. Then $h(x_2^{(0)}, \dots, x_k^{(0)}) = 0$.

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 $\{v_1, \dots, v_\omega\}$ with

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

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

Note that g^m divides any shift polynomial $h_{\dots}(x_2^{(0)}, \dots, x_k^{(0)})$. Let X_2, \dots, X_k be the upper bounds of $x_2^{(0)}, \dots, x_k^{(0)}$ respectively. Now we define a lattice L using the coefficient vectors of $h_{\dots}(x_2 X_2, \dots, x_k X_k)$. Let the dimension of L be ω . Under Assumption 1, one gets $x_2^{(0)}, \dots, x_k^{(0)}$ 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 $k \ll \omega$ (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.

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

Proof. Let $j_2 + \dots + j_k = r$ 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 $j_2 + \dots + j_k = m$ for $0 \leq j_2, \dots, j_{n-1} < i_n$, and $0 \leq j_n, \dots, j_k \leq m$. Now the number of all possible solutions of $j_2 + \dots + j_k = m$ for $0 \leq j_2, \dots, j_{n-1} < i_n$, and $0 \leq j_n, \dots, j_k \leq m$ is the co-efficient of x^m in $(1+x+\dots+x^{i_n-1})^{n-2}(1+x+\dots+x^m)^{k-n+1}$ and we denote the coefficient by $c(n, i_n)$, as we have fixed n, i_n . Now $(1+x+\dots+x^{i_n-1})^{n-2}(1+x+\dots+x^m)^{k-n+1} = \left(\frac{1-x^{i_n}}{1-x}\right)^{n-2} \left(\frac{1-x^{m+1}}{1-x}\right)^{k-n+1} = (1-x^{i_n})^{n-2}(1-x^{m+1})^{k-n+1}(1-x)^{-k+1}$. 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, $c(n, i_n) = \sum_{r=0}^{n-2} (-1)^r \binom{n-2}{r} \binom{k+m-r i_n-2}{m-r i_n}$. 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. \square

Lemma 4. *The determinant of L is given by $\det(L) = P_1 P_2$ where*

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

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

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

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

$$X_2^{j_2} X_3^{j_3} \dots X_k^{j_k} a_1^{m-j_2-\dots-j_k}, \quad (4)$$

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

$$X_n^{i_n} X_2^{j_2} X_3^{j_3} \dots X_k^{j_k}, \quad (5)$$

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

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

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 $\text{poly}\{\log a_1, \exp(k)\}$ 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 $x_2^{(0)}, \dots, x_k^{(0)}$. In that case we get the following result.

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

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

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

Proof. Let $X_2 = X_3 = \dots = X_k = X$. Then from (4), we have

$X_2^{j_2} X_3^{j_3} \dots X_k^{j_k} a_1^{m-j_2-\dots-j_k} = X^{j_2+\dots+j_k} a_1^{m-j_2-\dots-j_k}$, for non-negative integers j_i , $2 \leq i \leq k$ such that $j_2 + \dots + j_k \leq m$. Let $j_2 + \dots + j_k = r$ where $0 \leq r \leq m$. 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) $1 \leq i_n \leq t$, for $2 \leq n \leq k$ and a positive integer t , and (ii) $j_2 + j_3 + \dots + j_k = m$, when $0 \leq j_2, \dots, j_{n-1} < i_n$, and $0 \leq j_n, \dots, j_k \leq m$. Thus,

$$P_2 = \prod_{n=2}^k \prod_{i_n=1}^t X_n^{i_n} X_2^{j_2} X_3^{j_3} \dots X_k^{j_k} = \prod_{n=2}^k \prod_{i_n=1}^t X^{i_n c(n, i_n)} X^{m c(n, i_n)}$$

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

□

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 $N_1 = p_1q_1$ and $N_2 = p_2q_2$, 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 $\text{poly}(\log N)$ time when $\beta < 1 - 3\alpha + \alpha^2$, provided $2\alpha + \beta \leq 1$.*

Proof. Let $x_0 = p_1 - p_2$. We have $N_1 = p_1q_1$ and $N_2 = p_2q_2 = (p_1 - x_0)q_2$. Our goal is to recover x_0q_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_0q_2 . Now we consider the shift polynomials

$$g_{ij}(x) = x^i(N_2 + x)^j N_1^{m-j} \quad (6)$$

for $i = 0, 0 \leq j \leq m$ and $j = m, 1 \leq i \leq t$,

where m, t are fixed non-negative integers. Clearly, $g_{ij}(x_0q_2) \equiv 0 \pmod{p_1^m}$.

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

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

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 $h(x_0q_2) \equiv 0 \pmod{p_1^m}$. To apply Lemma 1 and Lemma 2 for finding the integer root of $h(x)$, we need

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

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

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

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

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

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

$$1 - \beta - 2\alpha \geq 0. \quad (11)$$

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

$$\alpha^2 - 3\alpha - \beta + 1 > 0. \quad (12)$$

Once x_0q_2 , the integer root of $h(x)$, is known, we get p_1 by calculating the GCD of $N_1, N_2 + x_0q_2$. As long as $|x_0q_2| < p_1$, we get q_2 by calculating the floor or ceiling of $\frac{N_2}{p_1}$. As $|x_0q_2| \leq N^{\alpha+\beta}$ and $p_1 \approx N^{1-\alpha}$, to satisfy $|x_0q_2| \leq p_1$ 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. \square

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].

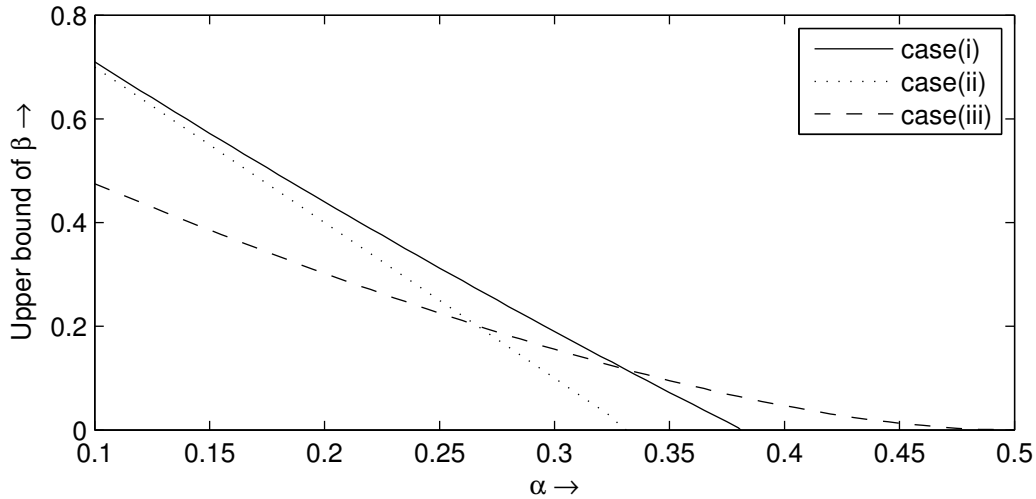


Fig. 1. Comparison of our theoretical result [case (i), works for both MSBs and LSBs] with that of [MAY09] [case (ii), works for LSBs], [FAU10] [case (ii), works for MSBs] and [SAR09] [case (iii), works for MSBs].

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,FAU10]. The strategy of [MAY09] considers equality in some LSBs of p_1, p_2 . One may note that our theoretical results for the MSB case is same as that of the LSB case (see Section 3.1 later). 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 LSBs for implicit factorization than the number of bit sharing in LSBs achieved in [MAY09]. For $k = 2$, the strategy of [FAU10] works in the MSB case when $\beta \leq 1 - 3\alpha - \frac{3}{\log_2 N}$, and thus it is clear that our theoretical result is better than [FAU10] for the MSB case. Later in Section 3, we will compare our results with that of [MAY09,FAU10] for all $k \geq 2$.

One may refer to Figure 1 for the comparison of the theoretical results. The curve for the theoretical result given in [FAU10] (MSB case) will be parallel and slightly below that of the curve for [MAY09] (LSB case) and that is why we have not drawn it separately.

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 $\text{poly}(\log N)$ 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.,

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

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. \square

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 $S = 1^r + 2^r + \dots + m^r$. 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 $S = \frac{m^{r+1}}{r+1} + o(m^{r+1})$. \square

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

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

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

for non-negative integers j_i , $2 \leq i \leq k$ such that $j_2 + \dots + j_k \leq m$ 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),

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

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

Let $X_2 = X_3 = \dots = X_k = X$, 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,

$P_2 = X^{\sum_{i_2=1}^t (i_2+m)} \frac{m^{k-2}}{(k-2)!} \approx X^{\frac{1}{(k-2)!} (\frac{t^2 m^{k-2}}{2} + t m^{k-1})}$, (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,

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

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

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

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 $4\alpha^2 k^2 + 4\alpha\beta k^2 + \beta^2 k^2 - 8\alpha^2 k - 10\alpha\beta k - 3\beta^2 k - 4\alpha k^2 - 2\beta k^2 + 2\alpha^2 + 4\alpha\beta + 2\beta^2 + 6\alpha k + 4\beta k + k^2 - 2\alpha - 2\beta - k > 0$. From which we get the required condition as $\beta < \frac{k^2+5\alpha k-2\alpha k^2-2\alpha-2k+1-\sqrt{k^2+2\alpha^2 k-\alpha^2 k^2-2k+1}}{k^2-3k+2}$ 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. \square

3.1 Implicit Factorization problem with shared MSBs and LSBs together

So far we continued our discussion for the MSB case for better understanding. Now we show that the same technique works as well when MSBs and LSBs are shared together. This also

takes care of the case when only LSBs are shared. As before, consider $N_1 = p_1q_1, N_2 = p_2q_2, \dots, N_k = p_kq_k$, where p_1, p_2, \dots, p_k and q_1, q_2, \dots, q_k are primes. It is also considered that p_1, p_2, \dots, p_k are of same bit size and so are q_1, q_2, \dots, q_k . We also assume that some amount of LSBs as well as some amount of MSBs of p_1, p_2, \dots, p_k are same.

Theorem 5. *Let $q_1, q_2, \dots, q_k \approx N^\alpha$. Consider that $\gamma_1 \log_2 N$ many MSBs and $\gamma_2 \log_2 N$ many LSBs of p_1, p_2, \dots, p_k are same. Let $\beta = 1 - \alpha - \gamma_1 - \gamma_2$. Then, under Assumption 1, one can factor N_1, N_2, \dots, N_k in $\text{poly}\{\log a, \exp(k)\}$ if*

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

$$< 1 - 3\alpha + \alpha^2, \text{ for } k = 2.$$

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

Proof. It is given that $\gamma_1 \log_2 N$ many MSBs and $\gamma_2 \log_2 N$ many LSBs of p_1, p_2, \dots, p_k are same. Thus, we can write

$$\begin{aligned} p_2 &= p_1^{(0)} + N^{\gamma_2} x_2^{(0)}, \\ p_3 &= p_1^{(0)} + N^{\gamma_2} x_3^{(0)}, \\ &\dots \\ p_k &= p_1^{(0)} + N^{\gamma_2} x_k^{(0)}. \end{aligned}$$

Using the above relations, we have,

$$N_i q_1 - N_1 q_i = N^{\gamma_2} x_i^{(0)} q_i q_1 \text{ for } 1 < i \leq k. \quad (18)$$

Suppose, $aN^{\gamma_2} \equiv 1 \pmod{N_1}$. Now multiplying Equation 18 by a , we get $aN_i q_1 - x_i^{(0)} q_i q_1 \equiv 0 \pmod{N_1}$.

Let $b_i \equiv aN_i \pmod{N_1}$ for $1 < i \leq k$. Thus we have, $b_i q_1 - x_i^{(0)} q_i q_1 \equiv 0 \pmod{N_1}$. From which, we get $b_i - x_i^{(0)} q_i \equiv 0 \pmod{p_1}$ for $1 < i \leq k$. Hence our first aim is to find $x_i^{(0)} q_i$ from N_1 and b_i for $1 < i \leq k$. Then from the knowledge of $x_i^{(0)} q_i$, we want to find the factorization of N_i for $1 \leq i \leq k$. Here we have $x_i^{(0)} \approx N^{1-\alpha-\gamma_1-\gamma_2}$ for $1 \leq i \leq k$. Then following the similar technique as in the proof of Theorem 4, the desired result is achieved. \square

We extend the results related to MSBs (as in Theorem 4 and earlier) in case of LSBs or MSBs and LSBs taken together (as in Theorem 3.1). This is achieved by exploiting the idea of a where, $aN^{\gamma_2} \equiv 1 \pmod{N_1}$ and then multiplying the equations by a .

This is the first time a solution could be achieved considering sharing of MSBs and LSBs together for general k . This has been studied for $k = 2$ in [SAR09] and it has been left as an open problem in [FAU10].

As this method works for considering the MSBs and LSBs together, the idea works for only LSBs too by considering one bit is shared in the MSBs (as the MSB of each of the primes is 1).

3.2 Comparison with the work of [MAY09,FAU10]

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 same here following the result in Section 3.1.

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 $k > 2$, $\frac{k^2+5\alpha k-2\alpha k^2-2\alpha-2k+1-\sqrt{k^2+2\alpha^2 k-\alpha^2 k^2-2k+1}}{k^2-3k+2} > 1 - \frac{2k-1}{k-1}\alpha$. Thus, for any k , $k \geq 2$, we need smaller amount of bit sharing in LSBs for implicit factorization than the number of bit sharing in LSBs achieved in [MAY09]. Our upper bound on β is $\frac{k-1-\sqrt{k^2+2\alpha^2 k-\alpha^2 k^2-2k+1}}{k^2-3k+2}$ 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 LSBs 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 $\text{poly}\{\log N, \exp(k)\}$. 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.

k	Bitsize of p_i, q_i $(1-\alpha)\log_2 N, \alpha\log_2 N$	No. of shared LSBs [MAY09] in p_i				No. of shared LSBs (our) in p_i			
		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	41.92
* 3	700, 300	450	452	3	< 1	416	431	56	59.58
* 3	650, 350	525	527	3	< 1	478	499	56	74.54
# 3	600, 400	600	-	-	-	539	562	56	106.87
* 4	750, 250	334	336	4	< 1	320	334	65	32.87
* 4	700, 300	400	402	4	< 1	380	400	65	38.17
* 4	650, 350	467	469	4	< 1	439	471	65	39.18
* 4	600, 400	534	535	4	< 1	497	528	65	65.15

Table 1. For 1000 bit N , theoretical and experimental data of the number of shared LSBs in [MAY09] and shared LSBs 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.

Very recently Faugere et. al. [FAU10] (independently at the same time of our work) presented a different lattice based approach for the problem of implicit factorization with shared MSBs of p_1, p_2, \dots, p_k . The strategy of [FAU10, Equation (10)] works when

$$\begin{aligned} \beta &\leq 1 - \alpha - \frac{k}{k-1}\alpha - \frac{1}{\log_2 N} - \frac{k}{2(k-1)\log_2 N} \left(2 + \frac{\log_2 k}{k} + \log_2(\pi e) \right) \\ &= 1 - \frac{2k-1}{k-1}\alpha - \frac{1}{\log_2 N} - \frac{k}{2(k-1)\log_2 N} \left(2 + \frac{\log_2 k}{k} + \log_2(\pi e) \right). \end{aligned}$$

Similar to our comparison with that of [MAY09] for LSB case, it can be noted that our method requires less number of MSBs to be shared compared to [FAU10] and the gap between our bound and that of [FAU10] reduces as k increases.

k	Bitsize of p_i, q_i $(1 - \alpha) \log_2 N, \alpha \log_2 N$	No. of shared MSBs [FAU10] in p_i				No. of shared MSBs (our) in p_i			
		Theory	Expt.	LD	Time (sec)	Theory	Expt.	LD	Time (sec)
2	874, 150	303	302	2	< 1	278	289	16	1.38
2	824, 200	403	402	2	< 1	361	372	16	1.51
2	774, 250	503	502	2	< 1	439	453	16	1.78
2	724, 300	603	602	2	< 1	513	527	16	2.14
3	874, 150	231	230	3	< 1	217	230	56	29.24
3	824, 200	306	304	3	< 1	286	304	56	36.28
3	774, 250	381	380	3	< 1	352	375	56	51.04
3	724, 300	456	455	3	< 1	417	441	56	70.55
3	674, 350	531	530	3	< 1	480	505	56	87.18
3	624, 400	606	604	3	< 1	540	569	56	117.14

Table 2. For 1024 bit N , theoretical and experimental data of the number of shared MSBs in [FAU10] and shared MSBs in our case.

We have explained our results for the MSB case as well as LSB case and compared with state of the art literature. The experimental results in both the cases are of similar quality using our techniques. Similar results are achieved in our case if one considers sharing of MSBs and LSBs together in the primes p_1, p_2, \dots, p_k . That is the reason we do not repeat these results.

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 6. The result in Theorem 6 below is not exactly presented in a similar form in [DJK09].

One can write,

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

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

$$\|b\| < \sqrt{k} a^{2\alpha+\beta}. \quad (19)$$

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

$$\|v\| < \sqrt{k} a^{\frac{\alpha+\beta+k-1}{k}}, \quad (20)$$

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 (19), (20), 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 6. 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 $\text{poly}\{\log a, \exp(k)\}$ time when,

$$\beta < 1 - \frac{2k-1}{k-1}\alpha. \quad (21)$$

Thus the bound in (21) works for the case when MSBs are shared and further, using the idea of Section 3.1, the bound in (21) can as well be used for the LSB case or the MSB and LSB case together. From the analysis presented in Section 3.2, 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.

4.1 Comparison with the work of [MAY09,FAU10]

The statement of Theorem 6 states that the time complexity is $\text{poly}\{\log a, \exp(k)\}$. However, under the assumption that “the shortest vector of the lattice L can be found by the LLL algorithm”, the complexity becomes $\text{poly}\{\log a, k\}$. 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 for larger k . We have implemented the method of [MAY09] for comparison as presented in Table 3.

α	k	Theoretical bound	Results of [MAY09]		Our results	
			Experiments	Time(sec.)	Experiments	Time(sec.)
0.25	3	375	377	< 1	376	< 1
0.35	10	389	391	< 1	390	< 1
0.40	100	405	408	50.36	407	28.21
0.44	50	449	452	7.09	451	4.04
0.48	100	485	492	68.88	488	36.36

Table 3. 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 LSBs in our case.

As we have already discussed, in the approach of [FAU10], the number of shared MSBs should be greater than or equal to $\frac{k}{k-1}\alpha \log_2 N + 6$, for $k \geq 3$. In our case, number of shared bit should be greater than $(1 - \alpha - (1 - \frac{2k-1}{k-1}\alpha)) \log_2 N = \frac{k}{k-1}\alpha \log_2 N$, putting the upper bound of β .

k	Bitsize of p_i, q_i $(1 - \alpha) \log_2 N, \alpha \log_2 N$	No. of shared MSBs [FAU10] in p_i				No. of shared MSBs (our) in p_i			
		Theory	Expt.	LD	Time (sec)	Theory	Expt.	LD	Time (sec)
10	874, 150	171	170	10	< 1	166	170	10	< 1
10	824, 200	227	225	10	< 1	220	225	10	< 1
10	774, 250	282	280	10	< 1	274	280	10	< 1
10	724, 300	338	334	10	< 1	328	332	10	< 1
10	674, 350	393	390	10	< 1	382	388	10	< 1
10	624, 400	449	446	10	< 1	435	444	10	< 1
40	874, 150	158	157	40	12.74	154	157	40	< 1
40	824, 200	209	206	40	17.42	205	206	40	< 1
40	774, 250	261	258	40	21.64	256	258	40	1.13
40	724, 300	312	309	40	24.17	307	308	40	1.26
40	674, 350	363	361	40	29.87	358	360	40	1.48
40	624, 400	414	412	40	34.69	409	410	40	1.75
100	874, 150	155	154	100	299.64	152	153	100	5.63
100	824, 200	206	205	100	525.67	202	204	100	9.36
100	774, 250	257	257	100	781.42	253	255	100	14.11
100	724, 300	307	307	100	1053.66	303	305	100	18.61
100	674, 350	358	357	100	1415.02	353	355	100	24.16
100	624, 400	408	408	100	2967.75	404	406	100	29.95

Table 4. For 1024 bit N , theoretical (bound for [FAU10] and in our case) and experimental data of the number of shared MSBs in [FAU10] and shared MSBs in our case.

Advantages of our approach over [FAU10] are as follows.

1. Our theoretical result in this section is slightly better than that of [FAU10] in terms of number of shared MSBs.
2. In the method in this section, the matrix corresponding to the lattice is a square one, but that is rectangular in the method of [FAU10]. Hence, determinant calculation is easier in

our method. In the presence of k RSA modulus, in this case we have to reduce a $k \times k$ matrix. Whereas, the size of the matrix in [FAU10] is $k \times \frac{k(k+1)}{2}$. Hence, in practice, the matrix reduction in case of [FAU10] takes more time than ours. From the experimental results presented in Table 4, it is clear that our strategy requires much less time than the method of [FAU10].

We have implemented the method of [FAU10] (this matches with the results presented in [FAU10, Tables 4, 5]) for comparison as presented in Table 4.

Above, we have present our results for the MSB case as well as LSB case and compared with [MAY09,FAU10]. The experimental results in both the cases are of similar quality using our techniques. Similar results are achieved in our case if one considers sharing of MSBs and LSBs together, and thus we do not repeat these results.

4.2 Comparison of the methods with respect to EPACDP

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

$$\begin{aligned} \beta &< 1 - 2\alpha + \alpha^2 \text{ for } k = 2 \text{ and} \\ &< \frac{1 - 2k + 2\alpha k + k^2 - \alpha k^2 - \sqrt{k^2 + 2\alpha^2 k - \alpha^2 k^2 - 2k + 1}}{k^2 - 3k + 2} \text{ for } k > 2. \end{aligned} \quad (22)$$

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 5, we present few such examples.

k	$\alpha = \frac{k-1}{k}$	Bound of β		LD	Time
		Inequality (22)	Experimental		
3	$\frac{2}{3}$	0.1835	0.135	25	10.72
4	$\frac{3}{4}$	0.1464	0.09	28	28.75

Table 5. 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,

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

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

5.1 Method I

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

$$\begin{aligned} h_1(x_1, x_2, \dots, x_k) &= a_1^{(0)} + x_1, \\ &\dots, \\ h_k(x_1, x_2, \dots, x_k) &= a_k^{(0)} + x_k, \end{aligned} \tag{23}$$

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

Now let us define the shift polynomials

$$h_{s_1, \dots, s_k}(x_1, x_2, \dots, x_k) = h_1^{s_1} \dots h_k^{s_k}, \tag{24}$$

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

Let X_1, \dots, X_k be the upper bounds of $x_1^{(0)}, \dots, x_k^{(0)}$ respectively. Now we define a lattice L using the coefficient vectors of $h_{\dots}(x_1 X_1, \dots, x_k X_k)$. Let the dimension of L be ω . One gets $x_1^{(0)}, \dots, x_k^{(0)}$ (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 $\text{poly}\{\log a_1, \exp(k)\}$. Thus, for small fixed k this algorithm is polynomial in $\log a_1$. To summarize we get the following result.

Theorem 7. *Under Assumption 1, the EGACDP can be solved in $\text{poly}\{\log a_1, \exp(k)\}$ 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 6.

5.2 Method II

Here we follow the idea of Section 4. We have

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

where $a_1^{(0)}, \dots, a_k^{(0)}$ 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 $i \in [1, k]$.

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

$$M = (2^\rho q_1, x_1^{(0)} q_2 - q_1 x_2^{(0)}, \dots, x_1^{(0)} q_k - q_1 x_k^{(0)}) = b, \text{ say.}$$

It can be checked that

$$\|b\| < 2\sqrt{k}a^{\alpha+\beta}. \quad (25)$$

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

$$\|v\| < \sqrt{k} 2^{\frac{1}{k}} a^{\frac{\beta+k-1}{k}}. \quad (26)$$

Under Assumption 2 (Section 4), and from (25), (26), 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 8. Consider EGACDP with $g \approx a_1^{1-\alpha}$ and $x_1^{(0)} \approx x_2^{(0)} \approx \dots \approx x_k^{(0)} \approx a_1^\beta$. Then, under Assumption 2, one can solve EGACDP in $\text{poly}\{\log a, \exp(k)\}$ time when,

$$\beta < 1 - \frac{k}{k-1}\alpha, \tag{27}$$

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 8 states that the time complexity is $\text{poly}\{\log a, \exp(k)\}$. However, under the assumption that “the shortest vector of the lattice L can be found by the LLL algorithm”, the complexity becomes $\text{poly}\{\log a, k\}$. 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 6. 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 or LSBs or MSBs and LSBs together of the primes p_1, p_2, \dots, p_k are equal (but unknown). This covers the case when the LSBs are equal (but unknown) in [MAY09] and MSBs are equal (but unknown) in [FAU10]. 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].

Here we did not concentrate on the case when some portions of the bits at the middle of p_1, p_2, \dots, p_k are same. This problem has been discussed in [SAR09,FAU10]. However, it seems that our techniques in this paper cannot be immediately applied to this problem.

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. *Proceedings of Eurocrypt 2010*, To be published in *Lecture Notes in Computer Science*, Springer, 2010. *Cryptology ePrint Archive*, Report 2009/616, Available at <http://eprint.iacr.org/2009/616>.
- [FAU10] J. -C. Faugere, R. Marinier and G. Renault. Implicit Factoring with Shared Most Significant and Middle Bits. *Proceedings of PKC 2010*, To be published in *Lecture Notes in Computer Science*, Springer, 2010.
- [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.
- [SAR10] S. Sarkar and S. Maitra. Approximate Integer Common Divisor Problem relates to Implicit Factorization. All the versions available at <http://eprint.iacr.org/2009/626>. Submitted: 18 Dec 2009, 1st revision: 11 Feb 2010, 2nd revision: 11 May 2010.
- [STN02] D. R. Stinson. *Cryptography - Theory and Practice*. 2nd Edition, Chapman & Hall/CRC, 2002.