Factoring RSA Modulus with Known Bits from Both p and q: A Lattice Method

Yao Lu[1,2], Rui Zhang[1], and Dongdai Lin[1]

[1] State Key Laboratory of Information Security (SKLOIS), Institute of Information Engineering (IIE), Chinese Academy of Sciences (CAS)
[2] University of Chinese Academy of Sciences (UCAS)
`lywhhit@gmail.com, {r-zhang,ddlin}@iie.ac.cn`

Abstract. This paper investigates the problem of factoring RSA modulus $N = pq$ with some known bits from both p and q. In Asiacrypt'08, Herrmann and May presented a heuristic algorithm to factorize N with the knowledge of a random subset of the bits (distributed over small contiguous blocks) of a factor. However, in a real attack, an adversary often obtain some bits which distributed in both primes. This paper studies this extended setting and introduces a lattice-based approach. Our strategy is an extension of Coppersmiths technique on more variables, thus it is a heuristic method, which we heuristically assumed that the polynomials resulting from the lattice basis reduction are algebraically independent. However, in our experiments, we have observed that the well-established assumption is not always true, and for these scenarios, we also propose a method to fix it.

Keywords: lattices, RSA, Coppersmith's method, factoring with known bits.

1 Introduction

Factoring large integer is an old and fascinate problem in number theory which is important for cryptographic applications, especially after the birth of the public-key cryptosystem RSA. However, until now, there is no known deterministic or randomized polynomial-time algorithm without the help of quantum computers to solve it, the best algorithm to date is Number Field Sieve (NFS), which has an expected runtime $\mathcal{O}(\exp(c(\ln N)^{1/3}(\ln \ln N)^{2/3}))$ where c is a constant.

In practice, an attacker might obtain partial information from both p and q via side-channel attacks, it is important to investigate that how these affect the hardness of factorization problem. In Eurocrypt'85, Rivest and Shamir [11] first introduced the factoring with known bits problem, they applied Integer Programme technique and factored N given two-thirds of the least significant bits (LSBs) of either p or q. In Eurocrypt'96, Coppersmith [3] improved the above result, and showed that N can be factored given half of the LSBs or most significant bits (MSBs) of a factor. He used the lattice reduction technique to output small solutions to a bivariate polynomial. Note that for the above results, the unknown bits are within one consecutive block. Then in Asiacrypt'08, Herrmann and May [6] presented a heuristic algorithm that extend to n blocks, they also used the lattice reduction technique but for a linear modular polynomial. However, the running time of this algorithm is polynomial only for $n = \mathcal{O}(\log \log N)$ blocks.

A scenario different from the above setting is based on the cold boot attack [4], where one may only recover information stored in the computer memory with certain probability less than 1. Heninger and Shacham [5] studied the problem and presented a new algorithm to factorize N given a certain fraction of the random bits of the primes. Since the known bits are randomly distributed, their algorithm cannot make use of the lattice reduction or integer programming techniques. The reconstruction method is a modified brute-force search exploiting the known bits to prune wrong branches of the search tree, thereby reduced the total search space towards possible factorization.

To summarize, in practice we prefer to use the lattice-based approach for its better performance, on the other hand, the lattice-based approach requires strigent constraints: the knowledge of contiguous blocks. We notice that the previous lattice-based methods only consider the scenario which the leaked bits lie in a single prime. While in a real attack, we may obtain known bits from both primes. This raises the question whether we have any efficient lattice-based approach to utilize such additional information?

Our Treatments. In this paper we present a new heuristic algorithm to factorize N with the knowledge of a random subset of the bits (distributed over small contiguous blocks) in both primes. Suppose that p has n_1 unknown blocks, q has n_2 unknown blocks, it leads to a multivariate polynomial equation $f(x_1, \cdots, x_{n_1}, y_1, \cdots, y_{n_2}) = N - (a_0 + a_1x_1 + \cdots + a_{n_1}x_{n_1})(b_0 + b_1y_1 + \cdots + b_{n_2}y_{n_2}) = 0$ ($a_k = 2^l$: the k-th unknown block of p starts in the l-th bit position, $b_i = 2^j$: the i-th unknown block of q starts in the j-th bit position). Then we can use Coppersmith's method to recover the small solution of f.

Our algorithm relies on a heuristic assumption that the polynomials output by the LLL algorithm are algebraically independent, which is also assumed in many works [1,8,10,6]. However, in our experiments, we met some unsuccessful instances, in particular, if the unknown blocks are significantly unbalanced in size, the polynomials output are not always algebraically independent, thus one may not find enough independent polynomials to recover all the unknown bits. Therefore, for completeness, we give a detailed report for the failure of the assumption, and also present a method to fix these "unsuccessful" situations.

The rest of the paper is organized as follows. In Section 2, we introduce some useful background on lattice basis reduction and list some previous results. In Section 3, we give the analysis of the factoring with four unknown blocks of primes p, q, and provide various data obtained through numerical experiments. In Section 4, we generalize the analysis to an arbitrary number n of unknown blocks. At last, in Section 5 we give a conclusion.

2 Preliminaries

2.1 Lattices

Consider a set of linearly independent vectors $u_1, \cdots, u_w \in \mathbb{Z}^n$, with $w \leqslant n$. The lattice L, spanned by $\{u_1, \cdots, u_w\}$, is the set of all integer linear combinations of the vectors $u_1, \cdots, u_w$. The number of vectors is the dimension of the lattice. The set $u_1, \cdots, u_w$ is called a basis of L. In lattices with arbitrary dimension, finding the shortest vector is a very hard problem, however, approximations of a shortest vector can be obtained in polynomial time by applying the well-known *LLL* basis reduction algorithm [9].

Lemma 1. (**LLL**) *Let L be a lattice of dimension w. With polynomial time, the LLL-algorithm outputs reduced basis vector v_i, $1 \leqslant i \leqslant w$ that satisfy*

$$\| v_1 \| \leqslant \| v_2 \| \leqslant \cdots \leqslant \| v_i \| \leqslant 2^{\frac{w(w-1)}{4(w+1-i)}} \det(L)^{\frac{1}{w+1-i}}$$

We state Howgrave's result [7] to find small solutions of integer equations.

Lemma 2. (**Howgrave – Graham**) *Let $g(x_1, \cdots, x_k) \in \mathbb{Z}[x_1, \cdots, x_k]$ be an integer polynomial that consists of at most w monomials. Suppose that*

1. $g(y_1, \cdots, y_k) = 0 \bmod p^m$ *for* $| y_1 | \leqslant X_1, \cdots, | y_k | \leqslant X_k$ *and*
2. $\| g(x_1 X_1, \cdots, x_k X_k) \| < \frac{p^m}{\sqrt{w}}$

Then $g(y_1, \cdots, y_k) = 0$ holds over the integers.

Let $g(x_1, \cdots, x_k) = \sum_{i_1, \cdots, i_k} a_{i_1, \cdots, i_k} x_1^{i_1} \cdots x_k^{i_k}$. We define the norm of g by the Euclidean norm of its coefficient vector: $\| g \|^2 = \sum_{i_1, \cdots, i_k} a^2_{i_1, \cdots, i_k}$.

The approach we used in the rest of the paper relies on the following heuristic assumption for computing multivariate polynomials.

Assumption 1. *The lattice-based construction yields algebraically independent polynomials, the common roots of these polynomials can be efficiently computed using techniques like calculation of the resultants or finding a Gröbner basis.*

The first part of Assumption 1 assures that the constructed polynomials allow for extracting the common root, while the second part assures that we are able to compute these common roots efficiently.

2.2 Previous Results

Let l_N denote the bit size of N, we have the following lemma [12]:

Lemma 3. (**Sarkar**) *Let $N = pq$ where p, q are of the equal bit-size. If one knows t MSBs of p: p_m, then we can compute the approximation $q_m = \lceil N/p_m \rceil$ of q, the probability that q and q_m share the first $t - t' - 1$ MSBs is at least $P_{t'} = 1 - \frac{1}{2^{t'}}$ which $0 \leq t' \leq t$. If one knows t LSBs of p: p_l, then we can compute t LSBs of q: q_l.*

In [12], the authors presented another lattice based method to handle the following situation:

Lemma 4. (**Sarkar**) *Let $N = pq$ where p, q are of equal bit-size. Suppose τl_N LSBs of p, q are unknown but the subsequence ηl_N LSBs of p, q are known. Then, under Assumption 1, one can recover the τl_N unknown LSBs of p, q in polynomial time, if $\tau < \frac{\eta}{2}$.*

3 Factoring with Four Unknown Blocks

In this section, we present an algorithm to factorize N with four unknown blocks of p and q. This attack model is illustrated in Figure 1.

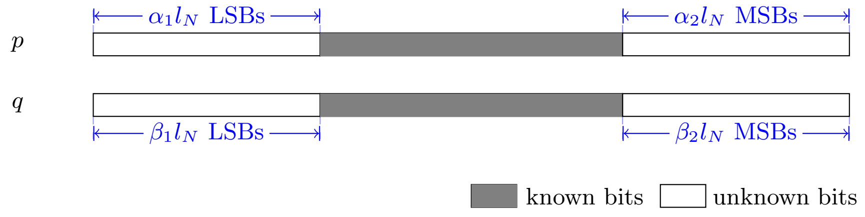

Fig. 1. Four unknown blocks of p, q

3.1 Our Algorithm

Let p_0, p_1, p_2 denote the known bits, the unknown $\alpha_1 l_N$ LSBs, the unknown $\alpha_2 l_N$ MSBs of p, let q_0, q_1, q_2 denote the known bits, the unknown $\beta_1 l_N$ LSBs, the unknown $\beta_2 l_N$ MSBs of q, respectively. Then we have

$$p = 2^{\alpha_1 l_N} p_0 + p_1 + 2^{(1/2-\alpha_2) l_N} p_2$$
$$q = 2^{\beta_1 l_N} q_0 + q_1 + 2^{(1/2-\beta_2) l_N} q_2$$

Hence we are interesting in finding the small root (p_1, p_2, q_1, q_2) of

$$f(x_1, x_2, y_1, y_2) = N - (2^{\alpha_1 l_N} p_0 + x_1 + 2^{(1/2-\alpha_2) l_N} x_2)(2^{\beta_1 l_N} q_0 + y_1 + 2^{(1/2-\beta_2) l_N} y_2)$$

Furthermore, we have the upper bounds

$$|p_i| \le X_i = N^{\alpha_i}, |q_i| \le Y_i = N^{\beta_i} \text{ for } i \in \{1, 2\}.$$

Following we use Coppersmith's method [3] to find the small integer root of polynomial f. Notice that the maximal coefficient of $f(x_1 X_1, x_2 X_2, y_1 Y_1, y_2 Y_2)$ is $N - p_1 q_1$, and the corresponding monomial is 1. Therefore, we define two sets: the set S is defined as the set of all monomials of f^{m-1} for a given positive integer m; the set M is defined as the set of all monomials that appear in

$x_1^{i_1}x_2^{i_2}y_1^{j_1}y_2^{j_2}f(x_1,x_2,y_1,y_2)$ with $x_1^{i_1}x_2^{i_2}y_1^{j_1}y_2^{j_2} \in S$. We introduce the shift polynomials

$$h_{i_1 i_2 j_1 j_2}(x_1,x_2,y_1,y_2) = x_1^{i_1}x_2^{i_2}y_1^{j_1}y_2^{j_2}f(x_1,x_2,y_1,y_2)$$

for $x_1^{i_1}x_2^{i_2}y_1^{j_1}y_2^{j_2} \in S$.

We also use the notations $s = |S|$ for the total number of shift polynomials and $d = |M| - |S|$ for the difference of the number of monomials and the number of shift polynomials. Next we build a $(d+s) \times (d+s)$ matrix L.

The upper left $d \times d$ block is diagonal, where the rows represent the monomials $x_1^{i_1}x_2^{i_2}y_1^{j_1}y_2^{j_2} \in M \backslash S$. The diagonal entry of the row corresponding to $x_1^{i_1}x_2^{i_2}y_1^{j_1}y_2^{j_2}$ is $(X_1^{i_1}X_2^{i_2}Y_1^{j_1}Y_2^{j_2})^{-1}$. The lower left $s \times d$ block contains only zeros.

The last s columns of the matrix L represent the shift polynomials $h_{i_1 i_2 j_1 j_2}$. The first d rows correspond to the monomials in $M \setminus S$, and the last s rows to the monomials of S. The entry in the column corresponding to $h_{i_1 i_2 j_1 j_2}$ is the coefficient of the monomial in $h_{i_1 i_2 j_1 j_2}$. If we sort the shift polynomials according to some ordering, the corresponding matrix defines a upper triangular lattice basis.

The determinant of the matrix L is

$$\begin{aligned}\det(L) &= \left(\prod_{x_1^{i_1}x_2^{i_2}y_1^{j_1}y_2^{j_2} \in M \backslash S} (X_1^{i_1}X_2^{i_2}Y_1^{j_1}Y_2^{j_2})^{-1} \right) \cdot (N - p_1 q_1)^s \\ &= X_1^{s_1}X_2^{s_2}Y_1^{s_1^*}Y_2^{s_2^*} \cdot (N - p_1 q_1)^s\end{aligned}$$

For the lattice attack to work, we require the enabling condition $\det(L) > 1$ (see [3] and [8] for detail). Then after some computations, we yield the bound:

$$(X_1 X_2 Y_1 Y_2)^{\frac{5}{12}m^4 + \circ(m^4)} < N^{\frac{1}{4}m^4 + \circ(m^4)}$$

To obtain the asymptotic bound, we let m grow to infinity, and substitute the values of X_1, X_2, Y_1, Y_2. Finally we obtain

$$\alpha_1 + \alpha_2 + \beta_1 + \beta_2 < 0.6$$

Then under this condition and Assumption 1, we can compute another three polynomials that share the same root (p_1, p_2, q_1, q_2) over the integers, which finally find the desired root.

3.2 Experimental Results

Our algorithm is heuristic, therefore, we state some experimental results in Table 1 to illustrate the performance of the above algorithm. All the experiments have been performed in Magma [2] over Windows 7 on a laptop with Intel(R) Core(TM) i5-2430M CPU 2.40 GHz, 2 GB RAM. In all the cases, we suppose N is an 1000-bit RSA modulo with equal-size prime p, q.

Table 1. Experimental results for the attack in case of partial leakage of p, q

	m	p(MSBs/LSBs)	q(MSBs/LSBs)	expt(bit)	theory(bit)	dim(L)	time(sec)	result
1	2	107/107	107/107	428	428	27	3.463	success
2	2	84/130	130/84	428	428	27	4.321	success
3	2	84/130	84/130	428	428	27	3.682	x_2, y_2
4	2	90/150	90/150	480	428	27	3.479	x_2, y_2
5	3	119/119	119/119	476	473	64	1932.681	success
6	3	50/185	185/50	470	473	64	3071.519	success
7	3	50/185	50/185	470	473	64	657.638	x_2, y_2
8	3	50/250	50/250	600	473	64	2603.844	x_2, y_2

[1] The word "success" in the column "*result*" means that we can successfully recover the desired small root; whereas the symbol "x_2, y_2" means that we can only recover the values of x_2 and y_2.

For given lattice parameter m, we presented the number of bits that one should theoretically be able to recover from p and q (column *theory* of Table 1). For simplicity, we suppose that p and q have equal size of unknown bits in our experiments.

We observe that Assumption 1 does not always hold in our experiments. In our experiments, if the unknown blocks are equal in bit size ($X_1 \approx X_2 \approx Y_1 \approx Y_2$), we will recover the unknown bits of p, q just as theoretically predicted (see the second row and the sixth row of Table 1)). However in the unbalanced case, the situation is more complicated, the success of the experiment greatly depends on the location of the unknown blocks.

For instance, if the unknown blocks with smaller size are located at MSB side of p and LSB side of $q(X_1 \gg X_2, Y_1 \ll Y_2)$, we can also successfully recover the unknown bits (see the third row and the seventh row of Table 1)). Otherwise if they are both located at MSB side of p, q ($X_1 \gg X_2, Y_1 \gg Y_2$), we observe that only smaller variables x_2 and y_2 are eliminated (see the fourth row and the eighth row of Table 1)), in this case, we notice that the smaller vectors lie in a sublattice of small dimension, which may be the reason why Assumption 1 fails; on the other hand, the sublattice structure is helpful to recover the unknown blocks with smaller size, which require less exposed bits practically (see the fifth row and the ninth row of Table 1)).

3.3 Main Theorem

The method of Coppersmith is able to exploit the algebraic relation among the variables, but it completely ignores the coefficients of the polynomial. That is may be the main reason why Assumption 1 fails in many experiments of Section 3.2. However, based on the experimental results, we observe that though sometimes we may not get enough algebraic independent polynomials to recover all variables, we can still recover some smaller unknown variables with only limited number of polynomials we got. With this observation we can summarize a weaker assumption which is more close to the real fact.

Assumption 2. *The lattice-based construction of Section 3.1 at least yields two algebraically independent polynomials, and the smaller unknown variables of these polynomials can be efficiently computed using Gröbner basis technique.*

Based on Assumption 2, we can get our theorem.

Theorem 1. *Let $N = pq$ where p, q are of equal bit-size. Let $\alpha_1, \alpha_2, \beta_1, \beta_2$ be parameters satisfy $0 < \alpha_1, \alpha_2, \beta_1, \beta_2 < 1$. Suppose $\alpha_1 l_N$ LSBs of p, $\alpha_2 l_N$ MSBs of p, $\beta_1 l_N$ LSBs of q, $\beta_2 l_N$ MSBs of q are unknown, and the rest bits of p, q are known. Then one can factorize N in polynomial time if one of the following conditions is satisfied:*

1. *$\alpha_1 + \alpha_2 < 0.207$ or $\beta_1 + \beta_2 < 0.207$ (Under Assumption 1).*
2. *$\alpha_1 + \alpha_2 + \beta_1 + \beta_2 < 0.6$ and $\alpha_i < 0.25$ or $\beta_i < 0.25$ for $i \in \{1, 2\}$ (Under Assumption 2).*

Proof. We can get Condition 1 directly from Herrmann and May's result [6], we focus on Condition 2.

In our algorithm, under the condition $\alpha_1 + \alpha_2 + \beta_1 + \beta_2 < 0.6$ and Assumption 1, we are able to recover the root efficiently. However, sometimes we only get two algebraic independent polynomials (Assumption 2), in these cases two smaller variables are eliminated, we bring the two of known variables back to the polynomial f, and construct a new polynomial with two variable. Then we can apply Coppersmith's method [3] which acts on two variables to find the desired root. There are only two cases which this method fails:

- $\alpha_1 > 0.25$ and $\beta_1 > 0.25$. In this case the unknown blocks with smaller size are both located at MSB side of p, q, we can only recover the unknown variables x_2, y_2, however, we can not get x_1, y_1 using Coppersmith's method which requires the knowledge of half of the LSBs of p or q.
- $\alpha_2 > 0.25$ and $\beta_2 > 0.25$. In this case the unknown blocks with smaller size are both located at LSB side of p, q, we can only recover the unknown variables x_1, y_1, however, we can not get x_2, y_2 using Coppersmith's method which requires the knowledge of half of the MSB of p or q.

Combining with the above discussions, we can get Condition 2.

Remark 1. Our algorithm can be improved if the the unknown blocks are significantly unbalanced (see experiment performances in Table 1), one could employ additional extra shifts in the smaller variables, which intuitively means that the smaller variable gets stronger weight since it cases smaller costs. We do not give the optimization process because of the enormous modes of the location of the unknown blocks.

4 Extension to Arbitrary Number of Unknown Blocks

In this section, we consider the scenario the number of the unknown blocks of the factors p, q is arbitrary. Suppose there are n_1 blocks unleaked whose respective length is $\alpha_i l_N$ $(1 \le i \le n_1)$, similarly, the length of unleaked blocks for q is $\beta_i l_N$ $(1 \le i \le n_2)$ respectively. Figure 2 illustrates the description of this attack model.

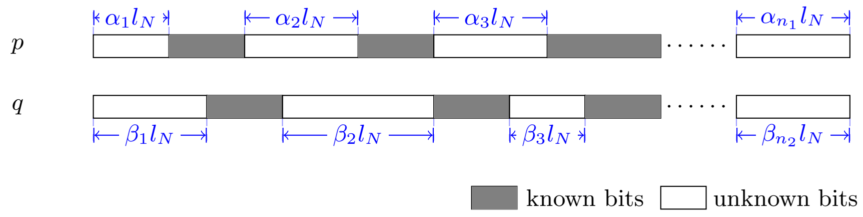

Fig. 2. Arbitrary number of unknown blocks of p and q

4.1 A General Algorithm

Since p is unknown for n_1 blocks, q is unknown for n_2 blocks, we can write $p = a_0 + a_1 p_1 + \cdots + a_{n_1} p_{n_1}$, $q = b_0 + b_1 q_1 + \cdots + b_{n_2} q_{n_2}$, where $p_i, q_j (1 \leq i \leq n_1, 1 \leq j \leq n_2)$ are unknowns, and $a_k = 2^l$ is the k-th unknown block of p starts in the l-th bit position, $b_i = 2^j$ is the i-th unknown block of q starts in the j-th bit position. This gives the following two equations:

$$p = a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_{n_1} x_{n_1}$$
$$q = b_0 + b_1 y_1 + b_2 y_2 + \cdots + b_{n_2} y_{n_2}$$

with unknown variables $x_1, \cdots, x_{n_1}, y_1, \cdots, y_{n_2}$. We multiply the two equations, then get a multivariate polynomial:

$$f'(x_1, \cdots, x_{n_1}, y_1, \cdots, y_{n_2}) = N - \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} a_i b_j x_i y_j - a_0 \sum_{j=1}^{n_2} b_j y_j - b_0 \sum_{i=1}^{n_1} a_i x_i - a_0 b_0$$

In particular, suppose that the leaked bits include $\gamma_m l_N$ MSBs and $\gamma_l l_N$ LSBs of p, q, we redefine the polynomial as follows:

$$f(x_1, \cdots, x_{n_1}, y_1, \cdots, y_{n_2}) = f'(x_1, \cdots, x_{n_1}, y_1, \cdots, y_{n_2}) / 2^{\gamma_l l_N}$$

Furthermore, we have the upper bounds:

$$|p_i| \leq X_i = N^{\alpha_i}, |q_j| \leq Y_j = N^{\beta_j} \text{ for } i \in \{1, 2, \ldots, n_1\} \; j \in \{1, 2, \ldots, n_2\}.$$

Following we use Coppersmith's method [3] to find the small integer root of polynomial f. Notice that the maximal coefficient of $f(x_1 X_1, x_2 X_2, y_1 Y_1, y_2 Y_2)$ is $(N - p_1 q_1)/2^{\gamma_1 l_N}$, and the corresponding monomial is 1. Therefore, we define two sets: the set S is defined as the set of all monomials of f^{m-1} for a given positive integer m; the set M is defined as the set of all monomials that appear in $x_1^{i_1} x_2^{i_2} y_1^{j_1} y_2^{j_2} f(x_1, x_2, y_1, y_2)$ with $x_1^{i_1} x_2^{i_2} y_1^{j_1} y_2^{j_2} \in S$. We introduce the shift polynomials

$$h_{i_1 i_2 j_1 j_2}(x_1, x_2, y_1, y_2) = x_1^{i_1} x_2^{i_2} y_1^{j_1} y_2^{j_2} f(x_1, x_2, y_1, y_2)$$

for $x_1^{i_1} x_2^{i_2} y_1^{j_1} y_2^{j_2} \in S$.

At first we define two sets:

$$S = \bigcup \{x_1^{i_1} \cdots x_{n_1}^{i_{n_1}} y_1^{j_1} \cdots y_{n_2}^{j_{n_2}} : x_1^{i_1} \cdots x_{n_1}^{i_{n_1}} y_1^{j_1} \cdots y_{n_2}^{j_{n_2}} \text{ is a monomial of } f^{m-1}\},$$
$$M = \{\text{monomials of } x_1^{i_1} \cdots x_{n_1}^{i_{n_1}} y_1^{j_1} \cdots y_{n_2}^{j_{n_2}} f : x_1^{i_1} \cdots x_{n_1}^{i_{n_1}} y_1^{j_1} \cdots y_{n_2}^{j_{n_2}} \in S\}$$

Next we built a matrix L to find at least $n_1 + n_2 - 1$ polynomials that share the root $(p_1, \cdots, p_{n_1}, q_1, \cdots, q_{n_2})$ over the integers. Then the matrix has triangular form if the coefficient vectors are sorted according to the order. Then we have to satisfy the following condition to get these polynomials:

$$X_1^{s_1} \cdots X_{n_1}^{s_{n_1}} Y_1^{s_1^*} \cdots Y_{n_2}^{s_{n_2}^*} < W^s$$

for $s_k = \sum_{x_1^{i_1} \cdots x_{n_1}^{i_{n_1}} y_1^{j_1} \cdots y_{n_2}^{j_{n_2}} \in M \backslash S} i_k$, $s_t^* = \sum_{x_1^{i_1} \cdots x_{n_1}^{i_{n_1}} y_1^{j_1} \cdots y_{n_2}^{j_{n_2}} \in M \backslash S} j_t$ with $k \in \{1, \cdots, n_1\}, t \in \{1, \cdots, n_2\}$, $s = |S|$ and $W = ||f(x_1 X_1, \cdots, x_{n_1} X_{n_1}, y_1 Y_1, \cdots, y_{n_2} Y_{n_2})||_\infty = N^{1-\gamma_m-\gamma_l}$.

The explicit computation of s, $s_1, s_2, \cdots, s_{n_1}$, $s_1^*, s_2^*, \cdots, s_{n_2}^*$ is given in Appendix A, while we only state the results here.

$$\begin{aligned}
\dim(L) = |M| &= \binom{m+n_1}{m}\binom{m+n_2}{m} \\
s &= \binom{m+n_1-1}{m-1}\binom{m+n_2-1}{m-1} \\
s_1 = \cdots = s_{n_1} &= \binom{m+n_2-1}{m}\binom{m+n_1-1}{m-2} + \binom{m+n_2}{m}\binom{m+n_1-1}{m-1} \\
s_1^* = \cdots = s_{n_2}^* &= \binom{m+n_1-1}{m}\binom{m+n_2-1}{m-2} + \binom{m+n_1}{m}\binom{m+n_2-1}{m-1}
\end{aligned}$$

Put the above values to the condition, we can get

$$\frac{\sum_{i=1}^{n_1} \alpha_i}{n_2+1} + \frac{\sum_{j=1}^{n_2} \beta_j}{n_1+1} < \frac{1-\gamma_m-\gamma_l}{n_1+n_2+1}$$

The runtime of our algorithm is dominated by the time to run LLL reduction algorithm on the lattice L, which takes polynomial time in the dimension of the lattice and in the bit-size of the entries. Thus the total time complexity of our algorithm is polynomial in $\log N$ but exponential in $n_1 + n_2$.

Let $n_1 = n_2 = 1$, after some calculations, we can get $\gamma_m + \gamma_l > 0.25$. It means that we can factorize N given $\gamma_m l_N$ MSBs and $\gamma_l l_N$ LSBs of a prime p if $\gamma_m + \gamma_l > 0.25$. If we assume $\gamma_l = 0$, then $\gamma_m > 0.25$, that is exactly Coppersmith's result on the problem of factoring with high bits known. Note that our result can be regard as an extension of Coppersmith's result.

This seems a perfect solution to the problem we posed at the beginning: Check whether or not the bit-size of unknown blocks satisfies the above conditions, if so, applies the above lattice method to recover the unknowns. However, in practice it not always works because of the failure of Assumption 1. Therefore, a natural problem is asked how we can repair this flaw.

4.2 A Combined Algorithm

In this section we present a combined algorithm to fix it. The main idea behind is as follows: Apply the lattice method to the original polynomial, though we may not recover all the unknown variables once, we still can get a part of them, then we reconstruct a new polynomial with the variables we recovered, and repeat the process until the lattice method fails. Now we give the detail.

Step 1. In this routine, we try to recover the MSBs and LSBs of p, q as much as possible. First check whether or not it satisfies the conditions of 4, if so, apply it. Secondly try to recover LSBs and MSBs of p, q using Lemma 3.

Step 2. Construct the polynomial with the known blocks of p, q, and apply the lattice method to this attack scenario.

Step 3. Check whether or not the algorithm of Step 2 recovers all the unknown variables of the polynomial, if so, terminate and return p, q; if not, test whether or not the algorithm recovers a part of variables, if that happens, go back Step 1 with the information of bits we have recovered, but if not, terminate and return fail.

This combined algorithm is a complement for the general algorithm of Section 4, it can not fully resolve the problem of the failure of Assumption 1, but it works in practice (see the discussions of Section 3).

5 Conclusion

In this paper we propose a lattice-based approach to factorize N with partial known bits of factors. Unlike previous works, we focus on the setting of the known bits from both primes. We give the detailed analysis for this extend setting, and provide the numerical experiments to support our theoretical bounds.

Acknowledgments. We would like to thank the anonymous reviewers for helpful comments. This work is supported by the National 973 Program of China under Grant No. 2011CB302400, IIEs Research Project on Cryptography under Grant No. Y3Z001C102, One Hundred Person Project of the Chinese Academy of Sciences under Grant No. NSFC61100225, the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDA06010701.

References

1. Boneh, D., Durfee, G.: Cryptanalysis of RSA with private key d less than $n^{0.292}$. IEEE Transactions on Information Theory 46(4), 1339–1349 (2000) 394
2. Cannon, J., et al.: Magma computional algebraic sydstem (version: V2. 12-16) (2012), `http://magma.maths.usyd.edu.au/magma/` 397

3. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. Journal of Cryptology 10(4), 233–260 (1997) 393, 396, 397, 399, 400
4. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: cold-boot attacks on encryption keys. Communications of the ACM 52(5), 91–98 (2009) 394
5. Heninger, N., Shacham, H.: Reconstructing RSA private keys from random key bits. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 1–17. Springer, Heidelberg (2009) 394
6. Herrmann, M., May, A.: Solving linear equations modulo divisors: On factoring given any bits. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 406–424. Springer, Heidelberg (2008) 394, 399
7. Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In: Darnell, M. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 131–142. Springer, Heidelberg (1997) 395
8. Jochemsz, E., May, A.: A polynomial time attack on RSA with private CRT-exponents smaller than $n^{0.073}$. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 395–411. Springer, Heidelberg (2007) 394, 397
9. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261(4), 515–534 (1982) 395
10. May, A.: New RSA vulnerabilities using lattice reduction methods. PhD thesis (2003) 394
11. Rivest, R.L., Shamir, A.: Efficient factoring based on partial information. In: Pichler, F. (ed.) EUROCRYPT 1985. LNCS, vol. 219, pp. 31–34. Springer, Heidelberg (1986) 393
12. Sarkar, S.: Partial key exposure: Generalized framework to attack RSA. In: Bernstein, D.J., Chatterjee, S. (eds.) INDOCRYPT 2011. LNCS, vol. 7107, pp. 76–92. Springer, Heidelberg (2011) 395, 396

A Counting $s,\ s_1, s_2, \cdots, s_{n_1},\ s_1^*, s_2^*, \cdots, s_{n_2}^*$

Note that s is the number of solutions of $0 \le i_1 + i_2 + \cdots + i_{n_1} \le m-1$, $0 \le j_1 + j_2 + \cdots + j_{n_2} \le m-1$. Thus

$$\begin{aligned} s &= \left(\sum_{i_1=0}^{m-1} \sum_{i_2=0}^{m-1-i_1} \cdots \sum_{i_{n_1}=0}^{m-1-i_1-\cdots-i_{n_1-1}} 1\right)\left(\sum_{j_1=0}^{m-1} \sum_{j_2=0}^{m-1-j_1} \cdots \sum_{j_{n_2}=0}^{m-1-j_1-\cdots-j_{n_2-1}} 1\right) \\ &= \left(\sum_{t=0}^{m-1} \binom{t+n_1-1}{t}\right)\left(\sum_{t=0}^{m-1} \binom{t+n_2-1}{t}\right) \\ &= \binom{m+n_1-1}{m-1}\binom{m+n_2-1}{m-1} \end{aligned}$$

Next we consider s_1, we have

$$s = \sum_{i_1=0}^{m} \sum_{i_2=0}^{m-i_1} \cdots \sum_{i_{n_1}=0}^{m-i_1-\cdots-i_{n_1-1}} \sum_{j_1=0}^{m} \sum_{j_2=0}^{m-j_1} \cdots \sum_{j_{n_2}=0}^{m-j_1-\cdots-j_{n_2-1}} i_1$$

$$
\begin{aligned}
&- \sum_{i_1=0}^{m-1} \sum_{i_2=0}^{m-1-i_1} \cdots \sum_{i_{n_1}=0}^{m-1-i_1-\cdots-i_{n_1-1}} \sum_{j_1=0}^{m-1} \sum_{j_2=0}^{m-1-j_1} \cdots \sum_{j_{n_2}=0}^{m-1-j_1-\cdots-j_{n_2-1}} i_1 \\
&= \binom{m+n_2}{m} \sum_{i_1=0}^{m} i_1 \binom{m-i_1+n_1-1}{m-i_1} - \binom{m+n_2-1}{m-1} \sum_{i_1=0}^{m-1} i_1 \binom{m-i_1+n_1-2}{m-i_1-1} \\
&= \binom{m+n_2}{m} \sum_{T=0}^{m} (m-T) \binom{T+n_1-1}{T} - \binom{m+n_2-1}{m-1} \sum_{T=0}^{m-1} (m-1-T) \binom{T+n_1-1}{T} \\
&= \binom{m+n_2}{m}\binom{m+n_1}{m-1} - \binom{m+n_2-1}{m-1}\binom{m+n_1-1}{m-2} \\
&= \binom{m+n_2-1}{m}\binom{m+n_1-1}{m-2} + \binom{m+n_2}{m}\binom{m+n_1-1}{m-1}
\end{aligned}
$$

According to the structure of f, we have $s_1 = \cdots = s_{n_1}$.

Because of the symmetric characteristic of x and y in f, we have

$$s_1^* = \cdots = s_{n_2}^* = \binom{m+n_1-1}{m}\binom{m+n_2-1}{m-2} + \binom{m+n_1}{m}\binom{m+n_2-1}{m-1}$$