Practical Small Private Exponent Attacks against RSA

Yansong Feng[1,2], Zhen Liu[3], Abderrahmane Nitaj[4], Yanbin Pan[1,2*]

[1*]Key Laboratory of Mathematics Mechanization, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China.
[2]School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing, China.
[3]School of Cyber Science and Technology, Hubei University, Wuhan, China.
[4]Normandie Univ, UNICAEN, CNRS, LMNO, Caen, France.

*Corresponding author(s). E-mail(s): panyanbin@amss.ac.cn;
Contributing authors: fengyansong@amss.ac.cn; liuzhen@hubu.edu.cn;
abderrahmane.nitaj@unicaen.fr;

Abstract

It is well known that the best small private exponent attack against RSA is that when the private exponent $d < N^{0.292}$, one can factor the RSA modulus $N = pq$. However, the bound $N^{0.292}$ is very difficult to achieve directly since we need to deal with some lattice with very high dimension, which seems infeasible by now. Recently, Li et al. proposed a practical attack that can solve cases when d approaches $N^{0.292}$ within a month for **1024** bit N. In this paper, we propose an improved practical small private exponent attack by enumerating the most significant bits of $p+q$. Together with some skills in implementations, we can also achieve the bound $N^{0.292}$, but with significantly less time compared to previous work.

Keywords: RSA, Small Private Exponent Attack, Lattice, Coppersmith's method

1 Introduction

In 1978, Rivest et al.[1] proposed the first public key cryptosystem RSA, which is now widely deployed in modern commercial systems for privacy and authenticity. Even after forty years, RSA continues to be a vibrant area of research, particularly in the field of cryptanalysis.

Key recovery attacks on RSA have been discussed in numerous works with various scenarios. One specific type of attack highlights the dangers of using a small private exponent and has gained a lot of attention because keeping decryption costs low is very important for RSA systems. In 1990, Wiener [2] successfully gave a key recovery attack against RSA for a small private exponent $d < \frac{1}{3}N^{1/4}$ by a continued fraction method, where $N = pq$ is the RSA modulus. Later, Coppersmith [3] proposed a lattice-based technique for RSA cryptanalysis. Coppersmith's methods open up a lot of in-depth research on lattice-based analysis of RSA. In [4], Boneh and Durfee extended the bound to $d < N^{0.292}$ for small private exponent attack via a new lattice-based approach. In 2010, Herrmann and May [5] employed a simpler and more efficient method to achieve the same bound $d < N^{0.292}$. Despite several efforts [6, 7], $d < N^{0.292}$ still remains the best bound.

However, it has been demonstrated that the bound can be improved under the relaxed condition of partial knowledge leakage. The concept of partial key exposure attacks on RSA was introduced by Boneh, Durfee, and Frankel in [8]. It addresses the scenario where an attacker has obtained some bits of the private exponent d. Ernst et al. [9] proposed a partial key exposure attack with the knowledge of the most significant bits(MSBs) of the private key d in the range of $N^{0.284} < d < N$. Later, Takayasu and Kunihiro [10] covered the range to $N^{0.292} < d < N$. The partial key exposure attack can be applied to various scenarios, including the leakage of the prime divisors p or q of the modulus N, or their sum $p + q$, etc [11–13].

1.1 Related work

Note that the bound in Coppersmith's method is typically an asymptotic theoretical bound. More precisely, the small private exponent attacks [4, 5] work if

$$d < N^{0.292-\epsilon} \text{ for some } \epsilon > 0.$$

Due to the existence of ϵ, we need to handle a lattice with a much larger dimension to approach the theoretical bound, but the corresponding lattice basis reduction algorithms cannot work well in both efficiency and quality for such lattices. Therefore, the bound that can be achieved in experiments is always inferior compared to the theoretical bound.

To achieve the theoretical bound in practice, an exhaustive search is usually applied. By guessing some bits of the prime factors of the RSA modulus, the upper bound $N^{0.292}$ on the private exponent can be slightly raised. In 2008, Sarkar et al. [14] presented their findings on the number of bits required to guess in the instances they discussed. For example, 33 bits of prime factor need to be guessed in the instances of 1000-bit modulus with private exponents up to $N^{0.3}$. However, they did not complete

the full attack, and the estimated running time for the complete attack was a cluster of 221 CPUs working for 512 days. Therefore, it is important to improve the efficiency of the attack to complete it in practice.

Recently, Li et al.[15] presented a practical attack on RSA with small private exponent, by detailed calculation for specific values of the dimension in Herrmann-May's attack [5] and binary search strategy for the most significant bits (MSBs) of prime divisor p. Due to a "multivalued-continuous phenomena", that is, different approximations of p lead to the same approximation of $p+q$, their attacks work well in practice. In their experiments, they can handle the case when $d \approx N^{0.292}$ for a 1024-bit RSA modulus and when $d \approx N^{0.287}$ for a 2048-bit RSA modulus in about a month. They also pointed out that when $p-q$ is relatively small, their attack will perform better. This is also validated by their experiments. All the instances for which their attack could be completed in a short time have a relatively small difference $p-q$, while for the instances with large differences their attacks still needs almost a month.

1.2 Our Contributions

In this paper, we present a new practical attack on small private exponent RSA, which works well for the case when the private exponents are close to the theoretical bound, significantly improving the practical effectiveness.

It is folklore that the complexity of the exhaustive search is a bottleneck that restricts the experimental bound. The key idea of our new attack is introducing a more efficient exhaustive search. In [15], they did an exhaustive search on the MSBs of p and for each candidate p, they tried to run Herrmann-May's attack to factor the RSA modulus. Notice that Herrmann-May's attack is launched using the MSBs of $p+q$, our attack directly goes to exhaustively search the MSBs of $p+q$. Since we know that $pq = N$, not any bit string is a possible candidate for the MSB of $p+q$. Hence, we first build a list of all the possible MSBs of $p+q$ by enumerating all the possible MSBs of p and then computing the corresponding MSBs of q and the sum. Note that we will delete the repeated items in the list to keep every elements different. Then, for each candidate in the list, we run the lattice-based attack to factor N. There are many ways to enumerate the elements in the list, such as in an ascending order, descending order or a random order. It is obvious that the location of the correct MSB of $p+q$ decides that which enumeration order is better. We think that our new exhaustive search will bring at least two advantages compared to [15].

- The whole enumeration space of the MSBs of $p+q$ is significantly less than the enumeration space of the MSBs of p. Taking Experiment 6 in [15] as an example, there are totally 8191 candidates for the 14 MSBs of p while there are only 392 candidates for the 14 MSBs of $p+q$.
- Although the multivalued-continuous phenomena narrows the search space on the MSBs of p and improves the actual performance of the attack in [15], it can be shown that the smaller $|p-q|$ is, then the smaller $p+q$ is, and the more the number of candidate MSBs of p's leading to the same MSBs of $p+q$ is, which means that if we adopt the ascending order to enumerate the elements in our list, our attack will perform better than [15]. It is shown in our experiments that our approach

can significantly reduce the cost of finding the accurate value of the MSBs of $p+q$ compared to enumerating the MSBs of p.

From a practical perspective, the effectiveness of Coppersmith-type cryptanalysis heavily relies on the performance of the lattice basis reduction algorithm, like LLL algorithm [16], and Gröbner basis method to solve a system of polynomials. In our experiments, we adopt the faster lattice reduction algorithm **flatter** [17] to lower the time reducing a lattice basis.

In addition, we propose a more efficient method for Gröbner basis computation. More precisely, we choose small finite fields and compute the Gröbner bases over these small fields. Finally, the desired roots of the system of polynomials can be obtained using the Chinese Remainder Theorem.

Overall, our strategy has changed the completion time in [15] from being measured in days to minutes. Besides, we provide more examples of successful attacks to further demonstrate the effectiveness of our attacks.

Roadmap. The remainder of the paper is organized as follows. In Section 2, we present some preliminaries and review Coppersmith's method. In Section 3, we present our analysis of attacking small private exponents based on enumerating MSBs of $p+q$. Besides, we also compare our attacks with Li et al.'s work [15], which focuses on enumerating MSBs of p. We illustrate our strategies for faster implementations in Coppersmith's method in Section 4 and describe the complete experimental results in Section 5. Finally, we give a short conclusion in Section 6.

2 Notations and Preliminaries

In this section, we state some of the notations and mathematics used in the attacks discussed in the rest of this work.

2.1 Some Notations

We use $\mathbb{Z}$ and $\mathbb{R}$ to denote the set of integers and real numbers, respectively. Let $|\cdot|$ denote the absolute value of a real number. We use the notation $\mathbb{Z}[x_1, \cdots, x_n]$ to denote the ring of polynomials in the n indeterminates $x_1, \cdots, x_n$ with coefficients from $\mathbb{Z}$. For any vector $v \in \mathbb{R}^n$, we use $\|v\|$ to denote the Euclidean norm. For any polynomial $h(x_1 \cdots, x_n) \in \mathbb{Z}[x_1, \cdots, x_n]$, we use $\|h(x_1 \cdots, x_n)\|$ to denote the Euclidean norm of the coefficient vector of $h(x_1 \cdots, x_n)$. That is, for $h(x_1 \cdots, x_n) = \sum h_{i_1,\cdots,i_n} {x_1}^{i_1} \cdots {x_n}^{i_n}$,

$$\|h(x_1 \cdots, x_n)\| = \sqrt{\sum {h_{i_1,\cdots,i_n}}^2}.$$

2.2 Lattices

This work primarily focuses on attacks that utilize lattice basis reduction techniques. We provide fundamental information about lattices and lattice basis reduction below.

Given $m \leq n$ linearly independent vectors $b_1, \cdots, b_m \in \mathbb{R}^n$, the set

$$\mathcal{L} = \mathcal{L}(b_1, \cdots, b_m) = \{\sum_{i=1}^{m} z_i b_i | z_i \in \mathbb{Z}\}$$

is a lattice. The integer m is referred to as the dimension of the lattice $\mathcal{L}$, while the integer n is known as the rank of the lattice $\mathcal{L}$. Usually, we let the basis vectors $b_1, \cdots, b_m$ be the rows in a matrix $\mathbf{B}$ to represent the lattice $\mathcal{L}$. The determinant of a lattice $\mathcal{L}$, denoted by $\det(\mathcal{L})$, is defined as $\det(\mathcal{L}) = \sqrt{\det(\mathbf{B}\mathbf{B}^\mathbf{T})}$. When $m = n$, the lattice $\mathcal{L}$ is a full rank lattice and the determinant of $\mathcal{L}$ is $|\det(\mathbf{B})|$.

A lattice is a discrete additive group in $\mathbb{R}^n$, ensuring the existence of the shortest nonzero vector within the lattice. The search for this vector is a crucial aspect of lattice-based cryptography. The well-known LLL algorithm [16] efficiently computes a reduced basis of a lattice, known as an LLL-reduced basis, which satisfies the the following property:

Lemma 1 (LLL [16]). *Given a basis for a lattice $\mathcal{L}$ can in polynomial time find an LLL-reduced basis $b_1, \cdots, b_m$, satisfying*

$$||b_1|| \leq \cdots \leq ||b_i|| \leq 2^{\frac{m(m-1)}{4(m+1-i)}} \det(\mathcal{L})^{\frac{1}{m+1-i}}, i = 1, \cdots, m.$$

This result is valuable as it gives a limit on the smallest basis vector, enabling us to establish bounds for specific attacks against RSA.

2.3 Coppersmith's Method

Many cryptanalysis problems can be formulated as finding small roots of a polynomial. Howgrave-Graham's following result [18, 19] states that small modular roots of a polynomial h with small coefficients are indeed integer roots of h.

Lemma 2 (Howgrave-Graham). *Let $h(x_1, \cdots, x_n) \in \mathbb{Z}[x_1, \cdots, x_n]$ be the sum of at most w monomials and let $X_1, \cdots, X_n > 0$. For any $(y_1, \cdots, y_n) \in \mathbb{Z}^n$ that satisfies the following two conditions:*

1. $h(y_1, \cdots, y_n) \equiv 0 \mod M$ where $|y_i| < X_i$, for $1 \leq i \leq n$,
2. $\|h(x_1X_1, \cdots, x_nX_n)\| < \frac{1}{\sqrt{w}}M$,

then $h(y_1, \cdots, y_n) = 0$.

In Coppersmith's method, the first step is to create a lattice, where each row represents the coefficient vector of a polynomial sharing the same root modulo M. For finding small roots of a modular polynomial f with n variables, we need n small polynomials that satisfy Howgrave-Graham's conditions in Lemma 2. When the determinate of this lattice is small enough, we can find the n relatively short vectors satisfied Lemma 2 by computing an LLL-reduced basis for this lattice. Suppose these n vectors are corresponding to the coefficient vectors of polynomials $h_1 \ldots, h_n$, we follow a Gröbner basis based approach. Assuming the ideal I generated by $h_1, \ldots, h_n$ is zero-dimensional (which is usually the case in practice), then we can efficiently compute

the common roots of $h_1, \ldots, h_n$. However, there is no provable guarantee that I is zero-dimensional. The following Assumtion 1 is widely used in many works [11, 20–23].

Assumption 1. *The ideal generated by the polynomials obtained from Coppersmith's method is zero-dimensional.*

Therefore, it is necessary to provide experimental results to verify the heuristic Assumption 1. We verify the correctness of Assumption 1 for our algorithm for Small Private Exponent RSA in Section 5.

3 Practical Attacks on Small Private Exponent RSA

3.1 Attack on Small Private Exponent RSA with MSBs of $p+q$

For the Small Private Exponent Attack of RSA, the best theoretical bound is $d < N^{0.292}$ [4, 5]. However, it is not easy to achieve this theoretical bound in practice. Previous works [11, 12, 14, 24] showed the bound of d can be improved with knowledge of partial information of p. In fact, a more direct method is improving the bound of d with knowledge of partial information of $p+q$. For completeness, we present the following result that factors N with MSBs of $p+q$, which is similar to [11].

Theorem 1. *Given a n-bit RSA moduli $N = pq$ with $d = N^{\delta}$. Let S be an approximation of $p+q$ such that $|p+q-S| < N^{\beta}$ and $\frac{1}{4} \leq \beta \leq \frac{1}{2}$. Then one can factor N in polynomial time under Assumption 1 if*

$$\delta < 1 - \sqrt{\beta}. \tag{1}$$

Proof. From the equality $ed = 1 + k(N - p - q + 1)$, we obtain the following equation:

$$f(x, y) = 1 + x(N - S + 1 + y) \mod e. \tag{2}$$

Using *unravelled linearization* proposed in [5, 25], we denote $u = xy + 1$ and $A = N - S + 1$. Hence Equation (2) can be rewritten as:

$$f(x, u) = u + Ax \mod e. \tag{3}$$

It is obvious that $(x_0, y_0) = (k, -(p+q-S))$ is a solution of Equation (2) and $(x_0, u_0) = (k, -k(p+q-S)+1)$ is a solution of Equation (3). The bound of roots can be calculated as follows:

$$\begin{aligned} |x_0| &= |k| \approx N^{\delta}, \\ |y_0| &= |p+q-S| \approx N^{\beta}, \\ |u_0| &= |-k(p+q-S)+1| \approx N^{\delta+\beta}. \end{aligned}$$

We use the following polynomials to construct lattice $\mathcal{L}$:

$$\begin{aligned} g_{i,k}(x, y, u) &= x^i f^k(x, u) e^{m-k}, \quad \text{for } k = 0, \ldots, m,\ i = 0, \ldots, m-k; \\ h_{i,j}(x, y, u) &= y^j f^k(x, u) e^{m-k}, \quad \text{for } j = 1, \ldots, t,\ k = \left\lceil \frac{m}{t} j \right\rceil, \ldots, m, \end{aligned}$$

where each monomial xy is replaced by $u-1$.

Using Lemma 1 and Lemma 2 and neglecting the lower terms $\mathcal{O}(m^3)$, we need that:

$$\det(\mathcal{L}) < e^{m \dim(\mathcal{L})}. \tag{4}$$

Using $|x_0| \approx N^{\delta}, |y_0| \approx N^{\beta}, |u_0| \approx N^{\delta+\beta}$, we can rewrite Equation (4) as following:

$$\frac{1}{6}\delta + \frac{\tau^2}{6}\beta + \left(\frac{1}{6} + \frac{\tau}{3}\right)(\delta+\beta) + \left(\frac{1}{3} + \frac{\tau}{6}\right) < \frac{1}{2} + \frac{\tau}{2}. \tag{5}$$

Simplifying the above equation, we obtain:

$$(\tau+1)\beta + \frac{1}{\tau+1} + 2\delta - 2 < 0 \tag{6}$$

Letting $\tau = \frac{1}{\sqrt{\beta}} - 1$, we have that $\delta < 1 - \sqrt{\beta}$, where $\frac{1}{4} < \beta \leq \frac{1}{2}$.

Then under Assumption 1, we can collect the roots successfully, thus concluding the proof. □

It's natural to take $2\sqrt{N}$ as an approximation of $p+q$. Obviously, when $2\sqrt{N}$ is a good approximation, $p-q$ must be small. Generally, suppose $p-q = N^{\frac{1}{2}-\theta}$, using the same notations in Theorem 1, then we directly have $p+q-2\sqrt{N} = \frac{(p-q)^2}{p+q+2\sqrt{N}} < \frac{2}{7}N^{\frac{1}{2}-2\theta}$. Using similar proof in Theorem 1, we have the following Corollary.

Corollary 1. *Suppose $p-q = N^{\frac{1}{2}-\theta}$, then we can factor N in polynomial time under Assumption 1 if*

$$\delta < 1 - \sqrt{\frac{1}{2} - 2\theta}.$$

It is obvious that larger θ yields a better bound in the Small Private Exponent Attack. Of course, if we can obtain a better approximation of $p+q$ through additional information about p and q, we can certainly further improve our bounds.

3.2 How to Enumerate MSBs of $p+q$

By Theorem 1, the MSBs of $p+q$ can help us factor N. A natural approach to get MSBs of $p+q$ is enumeration. We next show how to enumerate the s MSBs of $p+q$.

Building the list. We do not enumerate s MSBs of $p+q$ bit by bit, i.e., all 2^s values. With the constraint $N = pq$, not every value is a candidate value. Note that when s MSBs of p is known, it yields an approximation p_0 of p. A common technique is to use

$$q_0 = \frac{N}{p_0} \tag{7}$$

to obtain an approximation of q, and then obtain MSBs of $p+q$ [12, 14, 23].

Hence, to build the list of all the possible candidates of $p+q$, we first enumerate all the possible MSBs of p and then calculate and record the candidate value for the MSBs of $p+q$ as above. Notice that we will not add the repeated MSBs of $p+q$ into

Algorithm 1: Building the list of candidate MSBs of $p+q$

Input: Given a public key (N, e) and a parameter s.
Output: A set L of candidate approximation of $p+q$.

$L \leftarrow \emptyset$;
$W \leftarrow 2^{\frac{\log N}{2}-s}$;
for $i \in [\max\{2^{s-1}, \lfloor \frac{\sqrt{N}}{W} \rfloor\}, \min\{2^s-1, \lfloor \frac{\sqrt{2N}}{W} \rfloor\}]$ **do**
  $p_0 \leftarrow i * W$;
  $q_0 \leftarrow \lfloor \frac{N}{p_0} \rfloor$;
  $L \leftarrow L \cup \{\lfloor \frac{p_0+q_0}{W} \rfloor * W\}$;
end
return L;

the list, and since $p > \sqrt{N}$, we just enumerate the MSBs of p that is big enough. We summarize the whole process as Algorithm 1.

If we enumerate all the possible bit strings that consists of s bits, the complexity of this step is approximately 2^{s-1} as the first bit of p is always 1. However, with the constraint $pq = N$, the number of candidate $p+q$ will be significantly reduced.

Enumeration Order. The second step is to enumerate the candidate value of $p+q$'s MSBs in L. There are many ways to enumerate the elements of L, including but not limited to an ascending order, a descending order, a zig-zag enumerating order starting from the middle value in the list, or a random order. It is obvious that the location of the correct MSB of $p+q$ decides that which enumeration order is better.

For any $v \in L$, we can collect a set P_v of p_0's that yields v in Algorithm 1. We say that such p_0 is v-admissible and P_v is an v-admissible set. We have to point out that for the real v, the size of P_v depends on the size of an v-admissible p_0 heavily. A phenomenon already noticed in [15] is that $\#P_v$ becomes larger when p_0 gets close to $\sqrt{N}$. An intuitive understanding is to consider the function $f = x + \frac{N}{x}$, as shown in the Figure 1 below. It can be seen that when $x > \sqrt{N}$, the second derivative of f is greater than 0, meaning it becomes steeper. Therefore, this indicates that the closer p_0 is to $\sqrt{N}$, the larger the corresponding $\#P_v$. Besides, when $|p - \sqrt{N}|$ is small, i.e, $|p-q|$ is small, we have $p+q$ is small now, which implies $MSB(p+q)$ is small. Hence we adopt the most natural approach of enumerating from smallest to largest.

3.3 Comparison with Enumeration Strategy in [15]

The total number of enumerations in Li et al.'s Algorithm 1 [15] is still $2^{s-1}-1$, while the total number of our enumerations is much less. Taking Experiment 1-6 in [15] as examples, we list the number of total enumerations for $s = 14$ MSBs in Table 1.

Although the multivalued-continuous phenomena narrows the search space on the MSBs of p and improves the actual performance of the attack in [15], it has been shown that the smaller $|p-q|$ is, then the smaller $p+q$ is, and the more the number of candidate MSBs of p's leading to the same MSBs of $p+q$ is, which means that if we adopt the ascending order to enumerate the elements in our list, our attack will perform better than [15].

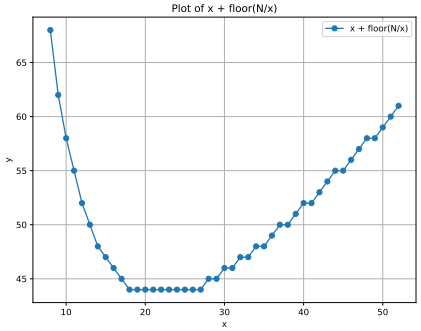


Fig. 1: Plot of $f = x + \lfloor \frac{N}{x} \rfloor$ with $N = 484$.

Table 1: Total Number of Enumerations with $s = 14$

	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6
Enumerate MSBs of p	8191	8191	8191	8191	8191	8191
Our method	1007	1274	997	1266	906	392

Furthermore, the performance difference between our attacks and [15] lies in the different weighting of the candidate values. Taking $N = 17 \times 19$ and $s = 4$ as an example, the set of all possible values of p's MSBs is $\{8, 9, 10, 11, 12\}$ (We ignore the possible MSBs that makes $p < \sqrt{N}$ although [15] does not rule them out). Then we can calculate the corresponding MSBs of $p + q$, which is exactly $\{17, 18\}$ by using Equation (7). Among $\{8, 9, 10, 11, 12\}$, only 9 can lead to the correct MSBs 18 of $p+q$. Therefore, for randomly enumerating p's MSBs, the correct probability is $1/5$. But for randomly enumerating $p + q$'s MSBs, the correct probability is $1/2$.

3.4 The Full Attack

Finally, our complete algorithm is presented in Algorithm 2.

Algorithm 2: Factor N with s MSBs Enumeration

Input: Given a public key (N, e) and a parameter s.
Output: Factorization $N = pq$

```
Run Algorithm 1 and get a list L;
for S ∈ L do
    Run Coppersmith's method by Theorem 1;
end
return p and q;
```

4 A Faster Implementation of Coppersmith Algorithm

4.1 Employing Faster Lattice Basis Reduction Algorithm

The LLL lattice reduction was carried out using the recently introduced **flatter** algorithm [17], which significantly surpasses the performance of SageMath's native LLL implementation (which internally utilizes FPLLL).

4.2 A More Efficient Gröbner Basis Computation

The Gröbner basis computation built into SageMath is somewhat slow. Meers et al. [26] choose to solve the common roots of $\mathcal{F} = \{f_1, \ldots, f_n\}$ over $\mathbb{Z}_{p_j}$ in their code implementation and use the Chinese Remainder Theorem to lift the results to $\mathbb{Z}$.

More precisely, if the upper bound of desired roots is B, we need a set of modulus $M = \{p_j\}$ such that $\prod p_j > B$. We denote this set as *modulus set*. In Meers et al.'s implementation [26], the *modulus set* is just chosen as $\{2, 3, 5, \cdots, p_k\}$ where p_k is the smallest prime such that the product of the elements in the *modulus set* is greater than B. However, this is not an optimal choice, we next show how to select a better *modulus set* to cost less time.

Select better *modulus set*. Suppose $\mathcal{F} = \{f_1, \ldots, f_n\}$ has a common root $(x_1, ...x_k)$ and a bound B satisfied $|x_j| < B$, we need a set of modulus $M = \{p_j\}$ such that $\prod p_j > B$ as above. Suppose the computation time of $\mathcal{F}$ over $\mathbb{F}_{p_j}$ is $T(p_j)$, then the total time should be

$$T = \sum_{p_j \in M} T(p_j), \tag{8}$$

with

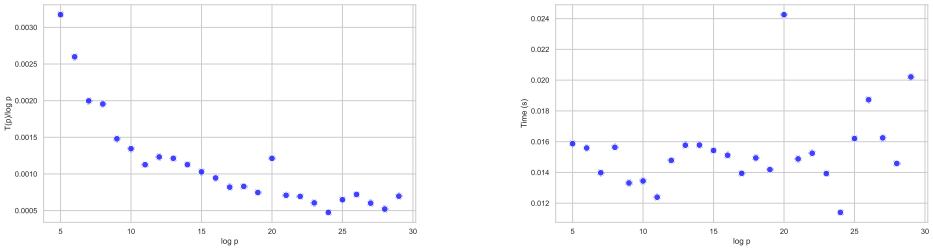
$$\sum_{p_j \in M} \log p_j > \log B. \tag{9}$$

If p_j is small, the time to compute $\mathcal{F}$ on $\mathbb{F}_{p_j}$ will be less, but the module's number of p_j, i.e., $|M|$ required will be more. If each p_j is very large, the time for a single computation will increase, but $|M|$ will also decrease.

We firstly define $\frac{T(p_j)}{\log p_j}$ to determine whether to add p_j in the *modulus set* M. The larger $\frac{T(p_j)}{\log p_j}$ is, the more likely p_j will be added to M. The following figures show how $\frac{T(p)}{\log p}$ changes with p.

From Figure 2, we can see that as $\log p$ increases, $y = \frac{T(p)}{\log p}$ becomes smaller and smaller. This is because as $\log p$ increases, $T(p)$ does not vary much. Therefore, when selecting the modulus set M, we should choose relatively large p as much as possible. Hence we choose p with 29 bits, which yields a significant acceleration. See Table 2 for details.

Additionally, if we can choose special p which has less $\frac{T(p)}{\log p}$, then we can further reduce the time of mod.



(a) $y = \frac{T(p)}{\log p}$ (b) $y = T(p)$

Fig. 2: y varies with $x = \log p$.

Table 2: Comparision with Trivial Strategy ($p_j = 2, 3, 5, \ldots$).

$\log_2 B$	Time for Trivial Strategy	Our Time	Improvement
512	16.59 s	5.7 s	2.9x
1024	21.0 s	8.2 s	2.6x

5 Experiments

Our experimental environment is Ubuntu 22.04 (WSL) on a 12th Gen Intel(R) Core(TM) i7-12700 2.10 GHz 8 GB RAM with Sagemath 10.3.

5.1 Comparision with Time in [15]

Li et al. [15] showed that enumerating 18 MSBs can factor 1024-bit N with $d \approx N^{0.292}$ practically. They also run their algorithm over 2048-bit N. We compare our time consumption with theirs in Table 3.

Table 3: Compared with running time in [15]

	$\log_2 N$	δ	$\log_2(\lvert p - q \rvert)$	Time in [15]	Our Step 1	Our Step 2
Exp. 1	1024	0.292	501	2.3 h	<0.1 s	107.6 s≈ 1.8 min
Exp. 2	1024	0.292	508	4.1 h	<0.1 s	891.5 s ≈ 14.9 min
Exp. 3	1024	0.292	511	21.6 days	<0.1 s	22.7 h
Exp. 4	2048	0.287	1021	7.1 days	<0.1 s	197.4 s≈ 3.3 min
Exp. 5	2048	0.287	1023	35.8 days	<0.1 s	9487.8 s≈ 2.6 h
Exp. 6	2048	0.292	974	11.2 days	335.3 s≈ 5.5 min	141.3 s≈ 2.4 min

Regarding parameter selection, for enumerating the number of bits s, for 1024-bit N with $d \approx N^{0.292}$, we choose $s = 14$. For 2048-bit N, we choose $s = 10$ for $d \approx N^{0.287}$ and $s = 28$ for $d \approx N^{0.292}$. In Coppersmith's method, we choose $m = 17$, which corresponds to a lattice dimension of approximately 240.

From Table 3, it can be seen that compared to the running time in [15], our improvement is very significant and successful, reducing the time counted by days to the minute level. This greatly enhances the deterrent effect in such actual attacks.

5.2 More Experiments

In addition, we also randomly selected some other RSA modulus. We also conducted experiments on some larger RSA modulus to demonstrate the effectiveness of our method.

Table 4: Experimental results on RSA

$\log_2 N$	δ	$\log_2(\lvert p-q\rvert)$	Our Step 1	Our Step 2	Total Time
1024	0.292	507	<0.1 s	103.9 s	1.7 min
		509	<0.1 s	1.8 h	1.8 h
		511	<0.1 s	14.1 h	14.1 h
2048	0.287	1022	<0.1 s	599.7 s	10.0 min
		1023	<0.1 s	1.4 h	1.4 h
2048	0.288	1022	<0.1 s	5.4 h	5.4 h
		1023	<0.1 s	37.2 h	37.2 h
4096	0.285	2041	<0.1 s	433.6 s	7.2 min
		2046	<0.1 s	451.3 s	7.5 min
4096	0.286	2047	<0.1 s	11.2 h	11.2 h
		2047	<0.1 s	21.8 h	21.8 h

Specifically, for enumerating the number of bits s, for 1024-bit N with $d \approx N^{0.292}$, we choose $s = 14$. For 2048-bit N, we choose $s = 10$ for $d \approx N^{0.287}$ and $s = 14$ for $d \approx N^{0.288}$. For 4096-bit N, we choose $s = 6$ for $d \approx N^{0.285}$ and $s = 12$ for $d \approx N^{0.286}$. In Coppersmith's method, we choose $m = 17$, which corresponds to a lattice dimension of approximately 240.

The algorithm's efficiency depends on factors such as the size of the enumeration list and the efficiency of the lattice reduction algorithm. For a larger δ, we can enumerate more MSBs or increase the lattice dimension in Coppersmith's method. Given the decent performance of the **flatter** algorithm [17] in handling high-dimensional lattices, we believe that choosing a larger lattice dimension will be more efficient, provided that the RAM is enough to allow for it.

6 Conclusion

We introduce a novel attack to solve the Small Private Exponent Attack practically, focusing on the enumeration of the MSBs of $p+q$. Drawing inspiration from the work of Meers and Nowakowski, we propose a selection strategy related to Grobner's basis in fields with small characteristics, which facilitates a more efficient implementation

of Coppersmith's algorithm. Our approach demonstrates considerable improvement in running time compared to the previous results.

References

[1] Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**(2), 120–126 (1978)

[2] Wiener, M.J.: Cryptanalysis of short rsa secret exponents. IEEE Transactions on Information theory **36**(3), 553–558 (1990)

[3] Coppersmith, D.: Small solutions to polynomial equations, and low exponent rsa vulnerabilities. Journal of cryptology **10**(4), 233–260 (1997)

[4] Boneh, D., Durfee, G.: Cryptanalysis of rsa with private key d less than n 0.292. In: Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18, pp. 1–11 (1999). Springer

[5] Herrmann, M., May, A.: Maximizing small root bounds by linearization and applications to small secret exponent rsa. In: Public Key Cryptography–PKC 2010: 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings 13, pp. 53–69 (2010). Springer

[6] Kunihiro, N., Shinohara, N., Izu, T.: A unified framework for small secret exponent attack on rsa. In: Selected Areas in Cryptography: 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers 18, pp. 260–277 (2012). Springer

[7] Takayasu, A., Kunihiro, N.: How to generalize rsa cryptanalyses. In: Public-Key Cryptography–PKC 2016, pp. 67–97. Springer, ??? (2016)

[8] Boneh, D., Durfee, G., Frankel, Y.: An attack on rsa given a small fraction of the private key bits. In: Advances in Cryptology—ASIACRYPT'98: International Conference on the Theory and Application of Cryptology and Information Security Beijing, China, October 18–22, 1998 Proceedings, pp. 25–34 (1998). Springer

[9] Ernst, M., Jochemsz, E., May, A., De Weger, B.: Partial key exposure attacks on rsa up to full size exponents. In: Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24, pp. 371–386 (2005). Springer

[10] Takayasu, A., Kunihiro, N.: Partial key exposure attacks on rsa: Achieving the

boneh–durfee bound. Theoretical Computer Science **761**, 51–77 (2019)

[11] Peng, L., Hu, L., Huang, Z., Xu, J.: Partial prime factor exposure attacks on rsa and its takagi's variant. In: Information Security Practice and Experience: 11th International Conference, ISPEC 2015, Beijing, China, May 5-8, 2015, Proceedings, pp. 96–108 (2015). Springer

[12] Sarkar, S., Maitra, S.: Improved partial key exposure attacks on rsa by guessing a few bits of one of the prime factors. In: Information Security and Cryptology–ICISC 2008: 11th International Conference, Seoul, Korea, December 3-5, 2008, Revised Selected Papers 11, pp. 37–51 (2009). Springer

[13] Wu, M.-E., Tso, R., Sun, H.-M.: Cryptanalysis of exhaustive search on attacking rsa. In: Network and System Security: 6th International Conference, NSS 2012, Wuyishan, Fujian, China, November 21-23, 2012. Proceedings 6, pp. 373–379 (2012). Springer

[14] Sarkar, S., Maitra, S., Sarkar, S.: Rsa cryptanalysis with increased bounds on the secret exponent using less lattice dimension. Cryptology ePrint Archive (2008)

[15] Li, Q., Zheng, Q.-x., Qi, W.-f.: Practical attacks on small private exponent rsa: new records and new insights. Designs, Codes and Cryptography **91**(12), 4107–4142 (2023)

[16] Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients (1982)

[17] Ryan, K., Heninger, N.: Fast practical lattice reduction through iterated compression. In: Annual International Cryptology Conference, pp. 3–36 (2023). Springer

[18] Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In: IMA International Conference on Cryptography and Coding, pp. 131–142 (1997). Springer

[19] Howgrave-Graham, N.: Approximate integer common divisors. In: International Cryptography and Lattices Conference, pp. 51–66 (2001). Springer

[20] May, A.: New rsa vulnerabilities using lattice reduction methods. PhD thesis, Citeseer (2003)

[21] May, A., Ritzenhofen, M.: Implicit factoring: On polynomial time factoring given only an implicit hint. In: International Workshop on Public Key Cryptography, pp. 1–14 (2009). Springer

[22] May, A., Nowakowski, J., Sarkar, S.: Partial key exposure attack on short secret exponent crt-rsa. In: International Conference on the Theory and Application of

Cryptology and Information Security, pp. 99–129 (2021). Springer

[23] Feng, Y., Nitaj, A., Pan, Y.: Partial prime factor exposure attacks on some rsa variants. Theoretical Computer Science **999**, 114549 (2024)

[24] Suk, A.H.: Cryptanalysis of rsa with lattice attacks. PhD thesis, University of Illinois at Urbana-Champaign (2003)

[25] Herrmann, M., May, A.: Attacking power generators using unravelled linearization: When do we output too much? In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 487–504 (2009). Springer

[26] Meers, J., Nowakowski, J.: Solving the hidden number problem for csidh and csurf via automated coppersmith. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 39–71 (2023). Springer