A note on the prime factorization method by Nemec et al.

Osamu Kikuchi¹ and Shigenori Uchiyama¹

¹ Tokyo Metropolitan University, 1-1 Minami-Osawa, Hachioji, Tokyo 192-0397, Japan

E-mail kikuchi-osamu@ed.tmu.ac.jp

Received December 7, 2018, Accepted February 16, 2019

Abstract

Nemec et al. analyzed secret key structure in the RSA cryptosystem used in smartcards manufactured by Infineon Technologies. In 2017, they estimated a form of prime numbers used for secret keys and proposed the prime factorization method in the context of composite numbers used in public keys. The purpose of this paper is to implement, and explore the applicability of, the prime factorization algorithm by Nemec et al.. We provide some notes on parameters used in the algorithm and conduct a numerical experiment under the parameter values we set.

Keywords RSA, factorization, smartcard, Coppersmith's algorithm

Research Activity Group Algorithmic Number Theory and Its Applications

1. Introduction

The RSA (Rivest-Shamir-Adleman) is a widely known cryptosystem in which the encryption and decryption keys are public and private, respectively.

Nemec et al. analyzed the structure of secret keys in the context of the RSA cryptosystem used in smartcards manufactured by Infineon Technologies. In 2017, they guessed the form of prime numbers used for secret keys and proposed a prime factorization method of the composite number used for public keys [1].

Specifically, they discovered an algorithmic flaw in the construction of prime numbers for RSA key generation and found that these prime numbers suffered from a significant loss of entropy.

According to them, the prime number used in RSA has a specific structure and their method of prime factorization requires no additional information except for the value of the public modulus.

They used knowledge of the specific structure of the prime numbers and applied their derivation of Coppersmith's algorithm, which finds small solutions of univariate polynomial equations. Furthermore, they devised an alternative representation of the prime numbers to render the prime factorization computationally feasible on consumer hardware.

The purpose of this paper is to implement the prime factorization algorithm by Nemec et al. and explore the applicability of their algorithm. The results suggest that there are cases where this algorithm does not work well. Specifically, there were cases where a solution modulo a divisor of a composite number could not be obtained by Coppersmith's algorithm. We provide some notes on parameters used in the algorithm and conduct a numerical experiment under the parameter values we set.

The remainder of the paper is organized as follows. In Section 2, we describe Coppersmith's algorithm, which plays a central role in the prime factorization method by Nemec et al. In Section 3, we describe the prime fac-

torization algorithm by Nemec et al. as well as a modified version of this algorithm, with different parameters, and explain the reason for the change. In Section 4, we present a numerical experiment. Finally, conclusions are offered in Section 5.

2. The prime factorization method by Nemec et al.

2.1 Coppersmith's algorithm

We outline Coppersmith's algorithm, which plays a central role in the prime factorization method of [1]. Let N be a positive integer of unknown factorization, which has a divisor $b \geq N^\beta$, $0 < \beta \leq 1$ and $f(x)$ an integer coefficient monic polynomial. When N and $f(x)$ are given, we can provide all solutions x_0 satisfying the following polynomial equation.

$$f(x_0) \equiv 0 \pmod{b} \quad (|x_0| \leq X) \quad (1)$$

where $|x_0|$ is the absolute value of x_0 and X is an upper bound on the size of solutions. X is set to $N^{\beta^2/\delta-\epsilon}/2$, where δ is a degree of $f(x)$ and $0 < \epsilon \leq \beta/7$. Coppersmith's algorithm can be executed in polynomial time in the input size. This algorithm uses the LLL algorithm.

An outline of Coppersmith's algorithm is given as follows:

Step1. We construct a collection of polynomials with the same solutions as (1) by using $f(x)$ and N .

Step2. We form a lattice basis which is spanned by the coefficient vectors of polynomials constructed in **Step1**.

Step3. We apply the LLL algorithm to this and obtain a first basis.

Step4. We construct a polynomial from the first basis and solve it over the integers.

Step5. Let x_0 be solutions obtained in **Step4**. If $\gcd(N, f(x_0)) \geq N^\beta$, x_0 is the solution we want to obtain.

Coppersmith's approach is fundamentally a reduction of an approach to solving modular polynomial equations,

to solving univariate polynomials over the integers. (For details, see [2, Section 10].)

2.2 Outline of the prime factorization method

Nemec et al. analyzed secret keys of RSA used in smartcards manufactured by Infineon Technologies. Specifically, they investigated distributions of “ $N \bmod r$ ” and “ $p \bmod r$ ” for small prime numbers r (p is a prime factor of N). They found that these distributions were not uniform. They then assumed that the prime factor of N could be represented by the following equation.

$$p = k_p \times M + (65537^{a_p} \bmod M) \quad (2)$$

where the integers k_p and a_p are unknown, $0 \leq a_p < \text{ord}_M(65537)$, $\text{ord}_M(65537)$ is the multiplicative order of 65537 in the group $(\mathbb{Z}/M\mathbb{Z})^\times$ and $M = P_n\#$ (the product of the first n successive primes $P_n\# = \prod_{i=1}^n P_i = 2 \times 3 \times \cdots \times P_n$ and the value of P_n is determined by the number of bits of N). The value of M is related to the key size. For example, when N is 512 bits, M can be set $167\#$ ($n = 39$). Now, 65537 is the 4th Fermat prime number. They analyzed the prime factor of N using this 4th Fermat prime number.

To locate the prime factors, one has to determine the integers k_p , a_p . The most important property is that k_p is sufficiently smaller than p and M .

A basic procedure would iterate over different options of “ $65537^{a_p} \bmod M$ ” and use Coppersmith’s algorithm in an attempt to locate k_p . The prime number p is found when the parameter a_p is guessed correctly. The cost of the prime factorization method is a function of the number of guesses ($\text{ord}_M(65537)$) with respect to a_p and the complexity of Coppersmith’s algorithm. In practice, $\text{ord}_M(65537)$ determines the running time of the entire factorization. Importantly, the number of attempts is too large even for small key sizes.

Thus, Nemec et al. replaced M with an appropriate M' (divisor of M) and noted that p could be represented by the following equation.

$$p = k'_p \times M' + (65537^{a'_p} \bmod M') \quad (3)$$

where M' must satisfy $M' > N^{1/4}$ ($N^{1/4}$ is a value determined from an upper bound to the solution of Coppersmith’s algorithm) and $\text{ord}_{M'}(65537)$ is expected to be sufficiently small compared to $\text{ord}_M(65537)$.

In [1], the prime factorization algorithm was implemented using empirically derived composite numbers (actually used in Infineon Technologies) and the running time of the prime factorization algorithm was evaluated.

3. Note on algorithm parameters

Based on [1], we implemented the prime factorization algorithm with SageMath. This section provides a procedural outline.

3.1 Algorithm by Nemec et al.

The algorithm put forward by Nemec et al. is given by **Algorithm 1**. For details about choosing M' , see [1, Section 2.7].

For line 1, it is generally difficult to compute a discrete

Algorithm 1 Prime factorization of N (Original version by Nemec et al.)

Input: $N (= p \times q, p, q : \text{prime})$, M' , m , $t (= m + 1)$
 $\triangleright m, t$: dimension parameters of a lattice

Output: a prime factor of N

```

1:  $c' \leftarrow \text{Log}_{65537} N \bmod \varphi(M')$ 
    $\triangleright$  Use Pohlig-Hellman Algorithm
2:  $\text{ord}' \leftarrow \text{ord}_{M'}(65537)$ 
3: for  $a'$  in  $[c'/2, (c' + \text{ord}')/2]$  do
4:    $f(x) \leftarrow x + ((M'^{-1} \bmod N) \times (65537^{a'} \bmod M') \bmod N)$ 
5:    $(\beta, X) \leftarrow (0.5, 2 \times N^{0.5}/M')$ 
6:    $\text{Solution\_set} \leftarrow \text{Coppersmith}(f(x), N, \beta, m, t, X)$ 
7:   if  $\text{Solution\_set} \neq \emptyset$  then
8:     for  $k'$  in  $\text{Solution\_set}$  do
9:        $p' \leftarrow k' \times M' + (65537^{a'} \bmod M')$ 
10:      if  $N \bmod p' = 0$  then
11:        return  $p'$ 
```

logarithm, but M' is a divisor of M . Thus, $\varphi(M')$ is represented by a product of small prime numbers ($\varphi(M')$ is the Euler function). As such, the discrete logarithm is straightforward to compute using the Pohlig-Hellman Algorithm.

Regarding algorithmic iteration, $c'/2$ is an average of the discrete logarithms of two prime factors whose base is 65537. When p and q are represented by $k'_p \times M' + (65537^{a'_p} \bmod M')$ and $k'_q \times M' + (65537^{a'_q} \bmod M')$, respectively, the following equation holds.

$$N \equiv 65537^{a'_p + a'_q} \equiv 65537^{c'} \pmod{M'}$$

Therefore, in this range, a solution modulo one of the prime factors is obtained by Coppersmith’s algorithm and the maximum number of iterations can be reduced to $\text{ord}'/2$.

3.2 Modification of algorithm parameters

As already noted, there are cases where the algorithm put forward by Nemec et al. does not work well. Accordingly, we considered a modified version of Algorithm 1 (**Algorithm 2**).

Algorithm 2 parameters are modified relative to Algorithm 1 as follows. (Note, N is 512 bits or more, p and q are the same size or $p > q$)

(a) Change β from 0.5 to 0.499:

Where $\beta = 0.5$, there are cases that a solution of modulo q cannot be obtained by Coppersmith’s algorithm because it does not satisfy $q > N^\beta$.

It satisfies $p < N^{0.501}$ if N is 512 bits or more. Therefore, $q = N/p > N^{0.499}$. As β satisfies $q > N^\beta$, this time we take it as 0.499.

(b) Change X from $(2 \times N^{0.5})/M'$ to $N^{0.501}/M'$:

The reason is that the absolute value of the solutions to be found by Coppersmith’s algorithm may become larger than $(2 \times N^{0.5})/M'$. By equation (3), $k'_p < p/M' < N^{0.501}/M'$.

(c) Change the infimum of M' from $N^{1/4}$ to $2N^{\alpha - \beta^2 + \varepsilon}$ ($\alpha = 0.501$, $\beta = 0.499$, $\varepsilon = \beta/28$):

Through change (b), take X to $N^{\beta^2 - \varepsilon}/2$ or less

Algorithm 2 Prime factorization of N (Modified version)

Input: $N (= p \times q, p, q : \text{prime}), M', m, t (= m + 1)$
 $\triangleright m, t : \text{dimension parameters of a lattice}$

Output: a prime factor of N

```

1:  $c' \leftarrow \text{Log}_{65537} N \bmod \varphi(M')$ 
    $\triangleright$  Use Pohlig-Hellman Algorithm
2:  $\text{ord}' \leftarrow \text{ord}_{M'}(65537)$ 
3: for  $a'$  in  $[c'/2, (c' + \text{ord}')/2]$  do
4:    $f(x) \leftarrow x + ((M'^{-1} \bmod N) \times (65537^{a'} \bmod M') \bmod N)$ 
5:    $(\beta, X) \leftarrow (0.499, N^{0.501}/M')$ 
6:    $\text{Solution\_set} \leftarrow \text{Coppersmith}(f(x), N, \beta, m, t, X)$ 
7:   if  $\text{Solution\_set} \neq \emptyset$  then
8:     for  $k'$  in  $\text{Solution\_set}$  do
9:        $p' \leftarrow k' \times M' + (65537^{a'} \bmod M')$ 
10:      if  $N \bmod p' = 0$  then
11:        return  $p'$ 

```

Table 1. Implementation results.

N	512bits
m	5
M'	141bits
$\text{ord}_{M'}(65537)$	1.2×10^6
Average time	3289 seconds
Longest time	7197 seconds

($N^{\beta^2-\varepsilon}/2$ is a value of an upper bound used in Coppersmith's algorithm in [2]). By $N^\alpha/M' \leq N^{\beta^2-\varepsilon}/2$, we can obtain the infimum of M' .

4. Implementation results

We used composite numbers actually used in Infineon Technologies in the Resources section of [3]. We randomly chose 50 composite numbers N and conducted a numerical experiment to explore whether we could factorize adequately. We determined the following:

- Size of M'
- Value of the optimal dimensional parameter m
- $\text{ord}_{M'}(65537)$
 $(\text{ord}_{M'}(65537))/2$ is the maximum number of the iteration part in Algorithm 2)
- Average time
- Worst time

The computer environment is as follows:

CPU: Intel Core i7-4770 3.40 GHz
Memory: 10 GB
OS: Windows 10 64 bit
Software : SageMath

The implementation results are shown in **Table 1**.

Although the condition of M' was changed from [1], the value of $\text{ord}_{M'}(65537)$ did not change from [1]. We could factorize all N in this experiment. With $N = 512$ bits, the running time of the prime factorization

Table 2. Longest running times in Nemec et al.

N	Longest time
512bit	1.93 hours
1024bit	97.1 days

ranged from approximately 30 seconds to approximately 2 hours, and about 55 minutes on average.

On the other hand, by using the original algorithm, only 22 composite numbers were factorized in the experiment.

We will see the effect of the proposed changes **(a)**, **(b)** and **(c)** in Section 3.2 below.

Regarding **(a)**, we conducted a numerical experiment again by changing only the value of β for 28 composite numbers N that could not be factorized. As a result, we could factorize all N . Therefore, it is considered that changing the value of β is effective.

Regarding **(b)** and **(c)**, set $X_1 = (2 \times N^{0.5})/M'$ and $X_2 = N^{0.501}/M'$. If $N \geq 2^{1000}$, it holds $X_1 \leq X_2$. That is, when N is 1001 bits or more, by setting X to X_2 , all solutions of the polynomial equation (1) is expected to be found. Also, by $X_2 \leq N^{\beta^2-\varepsilon}/2$, the running time of the proposed algorithm is expected to be short. Actually, we changed the parameters and compared the running time with the original one. As a result, both of the running time were almost the same, but the proposed one was 4 % shorter in the most effective case. So, when N is a larger size, we can expect to have a further effect.

We also conducted an experiment where N is 1024 bits. However, it was difficult to factorize N because the value of $\text{ord}_{M'}(65537)$ is too large.

In [1], the running time of the prime factorization algorithm was evaluated under the following computer environment (**Table 2**):

CPU : Intel Xeon E5-2650 v3 3.40GHz
Software : SageMath

In the work of those authors, at worst, it took 97.1 days to factorize N of 1024 bits. Thus, even if prime factorization could be accomplished, it could take a very substantial amount of time to obtain the prime factor.

5. Conclusions

In this paper, we implemented the prime factorization method put forward by Nemec et al. and explored whether it could factorize adequately. This motivated us to determine whether and how improvements could be made.

Specifically, we changed the value of the parameter and the condition in Section 3. In particular, it was important to take the value of β to make good use of Coppersmith's algorithm. With those changes, we could factorize the composite number adequately when N is 512 bits.

When N is large, the running time of the algorithm is too long because the number of iterations calling Coppersmith's algorithm is too large. Indeed, it takes a very long time to factorize N of 1024 bits.

The running time of the algorithm depends mainly on the size of $\text{ord}_{M'}(65537)$. It could be fruitful for future research to explore how to best select M' , i.e., optimization of M' . We want to reduce the iteration burden by decreasing the value of $\text{ord}_{M'}(65537)$ as much as possible.

In addition, Nemec et al. guessed the form of prime numbers by (2) but there may be alternative representations of the prime numbers that can be factorized faster. Thus, this is also an issue which warrants the attention of researchers in future work.

Acknowledgments

The authors would like to thank the anonymous reviewer for his/her valuable comments. This work was partially supported by consigned research fund from NTT Secure Platform Laboratories.

References

- [1] M. Nemec et al., The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli, in: Proc. of CCS '17, Ccs'17 Conference Committee, pp. 1631–1648, ACM, New York, 2017.
- [2] Phong Q. Nguyen et al., The LLL Algorithm : Survey and Applications, Springer-Verlag, Berlin, 2010.
- [3] CRoCS wiki, Measuring Popularity of Cryptographic Libraries in Internet-Wide Scans, ACSAC 2017, <https://crocs.fi.muni.cz/public/papers/acsac2017>, 2017.