International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 2, April 2020, pp. 1469~1476
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i2.pp1469-1476

The new integer factorization algorithm based on fermat's factorization algorithm and euler's theorem

Kritsanapong Somsuk
Department of Computer and Communication Engineering, Faculty of Technology,
Udon Thani Rajabhat University, Udon Thani, 41000, Thailand

Article Info

Article history:

Received Apr 7, 2019
Revised Oct 7, 2019
Accepted Oct 20, 2019

Keywords:

Computation loops
Euler's theorem
Fermat's factorization
Integer factorization
Prime factors

ABSTRACT

Although, Integer Factorization is one of the hard problems to break RSA, many factoring techniques are still developed. Fermat's Factorization Algorithm (FFA) which has very high performance when prime factors are close to each other is a type of integer factorization algorithms. In fact, there are two ways to implement FFA. The first is called FFA-1, it is a process to find the integer from square root computing. Because this operation takes high computation cost, it consumes high computation time to find the result. The other method is called FFA-2 which is the different technique to find prime factors. Although the computation loops are quite large, there is no square root computing included into the computation. In this paper, the new efficient factorization algorithm is introduced. Euler's theorem is chosen to apply with FFA to find the addition result between two prime factors. The advantage of the proposed method is that almost of square root operations are left out from the computation while loops are not increased, they are equal to the first method. Therefore, if the proposed method is compared with the FFA-1, it implies that the computation time is decreased, because there is no the square root operation and the loops are same. On the other hand, the loops of the proposed method are less than the second method. Therefore, time is also reduced. Furthermore, the proposed method can be also selected to apply with many methods which are modified from FFA to decrease more cost.

Corresponding Author:

Kritsanapong Somsuk,
Department of Computer and Communication Engineering,
Faculty of Technology, Udon Thani Rajabhat University,
Udon Thani, 41000, Thailand
Email: kritsanapong@udru.ac.th

1. INTRODUCTION

Nowadays, the significant information is always exchanged via the communication channel connected to computer system such as internet. Generally, this channel is the insecure channel. That means attackers can access data easily by using various techniques. With this reason, security for the information becomes very important. At present, many security algorithms were introduced to protect the secret data sending over insecure channel. Cryptography is one of techniques to defend data from attackers by using encryption and decryption processes. In addition, there are two types about cryptography. The first is symmetric key cryptography using the same key which is called secret key for encryption and decryption processes. The second is asymmetric key cryptography (or public key cryptography) [1] using a pair of keys for encryption and decryption. In addition, one key which is always distributed to keep in the key center is called public key. On the other hand, the other key which is always kept secretly by owner's key is called private key. RSA [2] is the most well-known public key cryptography used for both of digital signature and data encryption. This algorithm is one-way function. That means it is very easy to compute the production of

two large prime numbers. Nevertheless, it becomes very difficult to factor the modulus. Furthermore, RSA has been improved continuously to avoid attacking from intruders [3].

In fact, if attackers can recover two large prime factors of the modulus, then RSA is broken. Although, Integer Factorization is one of the hard problems to break RSA, many integer factorization algorithms, such as [4-10] are still developing to find the weakness of RSA which is not still disclosed. In general, the efficiency of each algorithm is based on characteristic of prime factors. For example, if one of two prime factors is very small, Trial Division Algorithm (TDA) [11, 12] is the best algorithm for this situation. However, TDA becomes inefficient algorithm whenever both of prime factors are very large. The other example is that if all prime factors of p–1 or q–1 are very small, where p and q are represented as two large prime factors of the modulus, Pollard's p-1 [13] should be chosen to recover both of them. Fermat's Factorization Algorithm (FFA) [14, 15] discovered by Pierre De Fermat in 1600 is one of integer factorization algorithms. FFA can factor modulus very fast when the difference between two large prime factors is small. In addition, the other form of modulus which is equal to the difference between two perfect square numbers is considered instead of the form of the production between two prime numbers. In fact, many improvement of FFA algorithms [16-21] were presented to reduce loops. In 2017, Specific Fermat's Factorization Algorithm Considered from X (SFFA-X) [22] was proposed to reduce the time complexity. The last m digits of modulus are considered to leave many unexpected values out from the computation. Furthermore, SFFA-X can be increased performance by changing the value of m which must be larger. However, assuming only traditional FFA is considered, there are two techniques to implement FFA. Both of them have the different advantage and disadvantage. The advantage of the first technique which is called FFA-1 is small loops when it is compared with the other technique which is called FFA-2. However, every loops have to compute square root operation which is not included to FFA-2. The aim of this paper is to propose the new integer factorization algorithm. Euler's theorem is applied with FFA to get the new idea behind the proposed method. In fact, the proposed method is from the combination of the advantage from both of FFA-1 and FFA-2.

2. RELATED WORKS

2.1. Fermet's factorization algorithm: FFA

FFA is the one of the methods to find two large prime factors. It suits to solve modulus which both of prime factors are close to each other. Assume p, q are odd prime numbers and n is the modulus that n=p*q. By using Fermat's technique, n must be rewritten in the other form as following:

$$n=x^2-y^2 \quad (1)$$

where, $x=\frac{p+q}{2}$ and $y=\frac{p-q}{2}$

FFA is distinguished as two methods which have different advantage and disadvantage as following:

2.2. FFA-1

Assigning, the initial value of x is $\lceil \sqrt{n} \rceil$, from (1), y is calculated from the following equation:

$$y=\sqrt{x^2-n} \quad (2)$$

Generally, if y is an integer, then p=x+y and q=x–y are two prime factors of n. On the other hand, x must be increased by 1 to find the new value of y whenever it is not the integer. In fact, the process must be repeated until the integer of y is found. Assigning l_1 is the number of loops for FFA-1, then

$$l_1=\frac{p+q}{2}-\lceil \sqrt{n} \rceil \quad (3)$$

Nowadays, many improvements of FFA-1 were proposed to reduce l_1 such as [16-18]. However, the disadvantage of FFA-1 and it's improving algorithms are about computing square root of integer.

2.3. FFA-2

FFA-2 is different from FFA-1, because there is no square root computing in the main process. From (1), the equation can be rewritten as following:

$$4n=u^2–v^2 \quad (4)$$

where, u=p+q and v=p–q. Assuming the initial value of u and v are $2\left\lceil \sqrt{n} \right\rceil$ and 0, respectively, then r is computed by following equation:

$$r=u^2–v^2–4n \quad (5)$$

From (5), if r is equal to 0, p and q are recovered from $p=\frac{u+v}{2}$ and $q=\frac{u+v}{2}$. On the othe hand, if r is not equal to 0, the process is divided as two cases:

- Case 1: r>0, v will be increased by 2 in order to reduce r until r which is equal to 0 is found.
- Case 2: r<0, u will be increased by 2 in order to enlarge r until r which is equal to 0 is found.

Assigning l_2 is represented as the number of loops computation for FFA-2, l_2 can be computed from,

$$l_2=l_u+l_v \quad (6)$$

where, l_u is loops of u and l_v is loops of v,

Considering l_u: the initial and target of u are $2\left\lceil \sqrt{n} \right\rceil$ and p+q. Moreover, it is always increased by 2 then it implied that l_u is equal to l_1. Considering l_v: the target result of v is p-q and the increment is 2, then

$$l_v=\frac{p-q}{2} \quad (7)$$

As FFA-1, there are many algorithms improved from FFA-2 such as Estimated Prime Factor (EPF) for estimating the new initial values [15] and Specific Fermat's Factorization Algorithm Considered from X (SFFA-X) [22] to leave unrelated integers from the computation. Although, integer square root is not computed, time to find prime factors is still high because of large loops. Table 1 is shown advantage and disadvantage between FFA-1 and FFA-2.

Table 1. Advantage and disadvantage between FFA-1 and FFA-2

Factorization Algorithm	Square root Computing	Loops
FFA-1	Every Time	Small (Compared with FFA-2)
FFA-2	None	Large

2.4. Euler's theorem

Euler's Theorem [23] is the theorem modified from Fermat's little Theorem. Assigning $a \in \mathbb{Z}^+$ and gcd(a, n)=1, $\Phi(n)=(p–1)*(q–1)=n–(p+q)+1$ and a is relative prime to $\Phi(n)$, then

$$a^{\Phi(n)} \equiv 1 \bmod n \quad (8)$$

In fact, this theorem is one of the main cores for implementing the proposed method mentioned in the next section.

3. THE PROPOSED METHOD

In this paper, the new algorithm based on FFA is proposed. The main core is to replace square root computing with modular multiplication. In dept, modular multiplication takes low cost in comparison to square root operation. Moreover, the loops are same as FFA-1. In addition, the method is from the combination between euler's theorem and FFA. In fact, (8) is rewritten as following:

$$a^{n–(p+q)+1} \equiv 1 \bmod n \quad (9)$$

Algorithm: The Proposed Method
Input: n
Output: p, q

1 $u \leftarrow 2\lceil \sqrt{n} \rceil$
2 Select c, where gcd(c, n)=1
3 $a \leftarrow c^{-1} \bmod n$
4 $s \leftarrow c^2 \bmod n$
5 $t \leftarrow a^{n-u+1} \bmod n$
6 IF t=1 then
7 $x \leftarrow \frac{u}{2}$
8 $y \leftarrow \sqrt{x^2 - n}$
9 Else
10 y←0.1
11 End IF
12 While y is not an integer do
13 t←t*s
14 IF t>n then
15 t←t mod n
16 End IF
17 u←u+2
18 IF t=1 then
19 $x \leftarrow \frac{u}{2}$
20 $y \leftarrow \sqrt{x^2 - n}$
21 End IF
22 End While
23 p←x+y
24 q←x–y

Because $l_u = \frac{p+q}{2} - \lceil \sqrt{n} \rceil$ (x is increased by 1) or $p+q-2\lceil \sqrt{n} \rceil$ (u is increased by 2), therefore, if u increased by 2 is considered, then $p+q=l_u+2\lceil \sqrt{n} \rceil$. Assigning b, $c \in \mathbb{Z}^+$, $b= a^{n-2\lceil \sqrt{n} \rceil+1} \bmod n$ and $c=a^{-1} \bmod n$ then,

$$b*c^{lu}=(a^{n-2\lceil \sqrt{n} \rceil+1})*(a^{-1})^{lu} \bmod n$$
$$= a^{n-2\lceil \sqrt{n} \rceil+1-lu} \bmod n$$
$$= a^{n-(2\lceil \sqrt{n} \rceil+lu)+1}$$
$$=a^{n-(p+q)+1} \bmod n$$

In fact, the process begins with $b= a^{n-2\lceil \sqrt{n} \rceil+1} \bmod n$ and b is recomputed by using modular multiplication with $c^2 \bmod n$ whenever the result which is equal to 1 is not found as follow:

$$b=b*c^2 \bmod n \quad (10)$$

From (10), the result is certainly equal to 1 whenever the process is repeated l_u times.

Assigning $x=\frac{l_u}{2}$, then y can be recovered by using (2). However, it is possible that there are some integers, assigned as l_i where i=0, 1, 2, ⋯, u-1 and $l_i<l_u$, that $a^{n-(2\lceil \sqrt{n} \rceil+li)+1} \bmod n$ is equal to 1. Nevertheless, y is not certainly an integer. Therefore, the process must be repeated until the b=1 and integer of y is found.

In fact, square root operation will be processed only when b is equal to 1. Because, l_u is represented as loops of the proposed method and the increasing step is 2, it implies that loops between the method and FFA-1 are same. In addition, c^2 mod n should be calculated as the constant at first, because it is often operated in (10). Furthermore, it should be assigned as small integer to decrease the cost. Therefore, c will be assigned before finding a: $a=c^{-1}$ mod n.

Example 1: Factoring n=340213 by using the proposed method

Sol. Each step from the algorithm is at following:

Step 1-2: u=1168, c=2

Step 3: $a=2^{-1}$ mod 340213=170107

Step 4: $s=2^2$ mod 340213=4

Step 5: $t=170107^{339046}$ mod 340213=186054

Step 6-11: Because $t \neq 1$, y=0.1

Step 12–19 (Loops)

Loop 1:

Step 13-16: t=186054*4=744216

Because t>340213 then 744216 mod 340213=63790

Step 17: u=1168+2=1170

Loop 2:

Step 13: t=63790*4 mod 340213=255160

Step 17: u=1170+2=1172

Loop 3:

Step 13-16: t=255160*4=1020640

Because t>340213 then 1020640 mod 340213=1

Step 17: u=1172+2=1174

Step 18–21: Because t=1, then

Step 19-20: $x=\frac{1174}{2}=587$ and $y=\sqrt{587^2-340213}=66$

Because t=1 and y is an integer (end of loop), then

Step 23-24: p=587+66=653, q=587-66=521

Therefore, there are only three loops to implement n=340213 by using the proposed method.

Considering FFA-1: $l_1=\frac{653+521}{2}-584=3$

Considering FFA-2: $l_v=\frac{653-521}{2}=66$, and $l_2=3+66=69$

Therefore, it implies that loops of FFA-2 are larger than the proposed method. However, FFA-1 takes time complexity higher than the proposed method, because modular multiplication takes low time complexity in comparison to square root operation [24]. Because n=340213 is too small, all mentioned algorithms can solve the problem very quick. The information in Table 2 is shown the main process and loops of FFA-1, FFA-2 and the proposed method to factor n = 340213. Assuming n=788582867650121563 which is from the production between p=1066200463 and q=739619701 (p and q have the same bits' length), Table 3 is shown the comparison during three algorithms to find p and q of n=788582867650121563.

Table 2. The comparison during three algorithms to solve problem with n = 340213

Factorization Algorithm	Loops	Main Process
FFA-1	3	Square root
FFA-2	69	Multiplication
The Proposed Method	3	Multiplication

Table 3. The comparison during three algorithms to solve problem with n=788582867650121563

Factorization Algorithm	Loops	Time (Second)
FFA-1	902910079	19.04
FFA-2	1066200460	25.75
The Proposed Method	902910079	2.82

From Table 3, although the loops between FFA-1 and the proposed method are same, the proposed method takes time lower than FFA-1. In addition, FFA-2 consumes the highest computation cost. In the example, the method is faster than FFA-1 and FFA-2 about 75% and 89%, respectively. Next, assuming n=10473296368211398l3 which is from the production between p=1971074143 and q=531349691 (p and q have the different bits' length). Table 4 is shown the comparison during three algorithms to find p and q of n=10473296368211398l3.

Table 4. The comparison during three algorithms to solve problem with n=10473296368211398l3

Factorization Algorithm	Loops	Time (Second)
FFA-1	227820673	290.02
FFA-2	947682899	142.53
The Proposed Method	227820673	41.17

In Table 4, the proposed method still takes the lowest computation time. However, FFA-2 becomes faster than FFA-1. The reason is that, although loops of FFA-2 are the highest, FFA-1 is slower than FFA-2, because square root computing becomes inefficient operation when loops are large (bits length between p and q are different). In the example, the method is faster than FFA-1 and FFA-2 about 75% and 72%.

4. COMPLEXITY ANALYSIS

4.1. The proposed method Vs. FFA-1

The multiplication of the main process from the proposed method is different from square root operation. The value of s which is the multiplication of t is selected at first. Then, the cost for the production between t and s is equal to the s-1 times of the addition between t and itself. For example, the compared process of proposed method in the example 1 is t+t+t+t = 4t. For each iteration, it implies that the complexity of the main process for the proposed method is close to the addition operation that is $\Theta(m)$ when m is represented as binary based on t. Therefore, it is less than square root cost which is $O(M(m))$, for Newton's method.

4.2. The proposed method Vs. FFA-2

In fact, the multiplication with the small multiple value is the main process for FFA-2. Therefore, the cost complexity for each iteration is equal to the proposed method. However, loops of FFA-2 are very larger than the proposed method. Therefore, the proposed method's cost is less than the compared algorithm.

5. APPLIED PROPOSED METHOD WITH IMPROVED ALGORITHMS

In addition, the proposed method can be applied with most of algorithms that modified from both of FFA-1 and FFA-2. However, two algorithms are selected as the examples to combine with the proposed method. First, EPF [15] improved from FFA-2 is the method to estimate the new initial values (u and v) for unbalanced modulus. In fact, the new initial value of u is also applied with the proposed method.

Example 2: Factoring n=1783647329 (p=84449 and q=21121)

Sol. Assign u_i is the new initial value, the loop's equation is changed as: $\frac{p + q - u_i}{2}$

Because, $\left\lceil \sqrt{n} \right\rceil$=42234, then from (3) the loops are 10551. However, u_i is computed as 97430, then the new loops can be re-estimated as 4070 that is less than traditional proposed method.

Second method is Multi Forms of Modulus for Fermat Factorization Algorithm (Mn-FFA) [25]. The key of this method is to find the pattern of x which is distinguished as 16 cases from the considering forms of n mod 4, n mod 6 and n mod 20 together. In fact, this method can be applied the proposed method by converting pattern of x as pattern of u.

Example 3: Factoring n=571187 (p=941 and q=607)

Sol. Because, 571187 mod 4=3, 571187 mod 6=5 and 571187 mod 20=7, then from Table 2 in [15], u must be divided by 12 and the last digit is 2 or 8.

Because $2\left\lceil \sqrt{n} \right\rceil$=1512 that is divided by 12 and the last digit is 2, u_i is equal to 1512.

Assigning c=2, then a=285594 and $t=a^{n-u+1} \bmod n$=411191

Usually u=u+2=1514 must be computed for the proposed method. However, for applying Mn-FFA with the proposed method, u can be computed as follow: u=u+36=1548, because it is the minimum value which is larger than 1512 and still in the condition. Therefore, the multiple value is not c^2 but it is c^{36} as following: c^{36} mod n=539953 and t=t*c^{36} mod n=1, then x=774 and y=167. Because y is an integer, then p and q can be recovered. In fact, there are only 2 loops for the implementation which are less than loops of tradition proposed method which are equal to 18.

6. CONCLUSION

In this paper, the new factorization algorithm from the applied method between Fermat's factorization algorithm and euler's theorem is proposed. The key is that the almost square root operations are left from the computation while the loops are not increased. In fact, multiplication operation with small multiple value is the main process of the proposed method. Then, it implies that the complexity for each iteration is only $\Theta(m)$, where m is binary based of computed number. Moreover, the proposed method can be also chosen to apply with the other algorithms modified from FFA to decrease more costs. In addition, this paper shows the proposed method which is applied with EPF and Mn-FFA.

REFERENCES

[1] W. Diffie and M. Hellman, "New directions in cryptography," in *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644-654, November 1976.

[2] R.L.Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Communications of ACM*, vol. 21, pp. 120-126, 1978.

[3] A.E., Mexher, "Enhanced RSA Cryptosystem based on Multiplicity of Public and Private Keys," *International Journal of Electrical and Computer Engineering*, vol. 8, pp. 3949-3953, 2018.

[4] J. Pollard, "Monte Carlo methods for index computation (mod p)," *Mathematics of Computation*, vol.32, pp.918 – 924, 1978.

[5] Z.Joseph, H.B.Joseph, "Prime factorization using square root approximation," *Computers and Mathematics with Applications*, vol. 61, pp. 2463 – 2467, 2011.

[6] P. Sharma, A.K. Gupta, A.Vijay, "Modified Integer Factorization Algorithm using V-Factor Method," *Proceedings of 2nd International Conference on Advanced Computing & Communication Technologies*, India: RG Education Society, pp. 423 – 425, 2012.

[7] K. Omar, L. Szalay, "Sufficient conditions for factoring a class of large integers," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 13, pp. 95-103, 2010.

[8] J. Jormakka, "On finding Fermat's pairs," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 10(3), pp. 401-413, 2007.

[9] H. W. Lenstra, "Factoring integers with elliptic curves," *Annals of Mathematics*, vol. 126 (2) , pp. 649–673, 1987.

[10] K. Somsuk, S. Kasemvilas, "MVFactor: A method to decrease processing time for factorization algorithm," *Proceedings of 17th International Computer Science and Engineering Conference*, Thailand, pp. 339-342, 2013.

[11] L.Nidhi, P. Anurag, K.Shishupal, "Modified Trial Division Algorithm Using KNJ-Factorization Method To Factorize RSA Public Key Encryption," *Proceedings of the International Conference on Contemporary Computing and Informatics*, India, pp. 992-995, 2014.

[12] S.Murat, "Generalized Trial Division," *International Journal of Contemporary Mathematical Science*, vol. 6(2), pp. 59-64, 2011.

[13] D.Bishop, "Introduction to Cryptography with java Applets," London: Jonesand Bartlett Publisher, 2003.

[14] B.R.Ambedkar, A. Gupta, P. Gautam, S.S. Bedi, "An Efficient Method to Factorize the RSA Public Key Encryption," *Proceedings of the International Conference on Communication Systems and Network Technologies*, Katra, pp. 108-111, 2011.

[15] M.E.Wu, R.Tso, H.M. Sun, "On the improvement of Fermat factorization using a continued fraction technique", *Future Generation Computer Systems*, vol. 30(1), pp.162 – 168, 2014.

[16] G.Xiang, "Fermat's Method of Factorization," *Applied Probability Trust*, vol. 36(2), pp.34-35, 2004.

[17] K.Somsuk, "A New Modified Integer Factorization Algorithm Using Integer Mod 20's Technique," *Proceedings of the 18 International Computer Science and Engineering Conference*, Thailand, pp. 312-316, 2014.

[18] K. Somsuk, S. Kasemvilas, "Possible Prime Modified Fermat Factorization New Improved Integer Factorization to Decrease Computation Time for Breaking RSA," *Proceedings of the 10 International Conference on Computing and Information Technology*, Thailand, pp. 325-334, 2014.

[19] K. Omar, "Algorithm for factoring some RSA and Rabin moduli," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 11(5), pp. 537–543, 2008.

[20] B. Randall, "Fingers find Fermat's factorization most probable," *The Mathematical Gazette*, vol.99(544), pp. 452-458, 2014.

[21] J. McKee, "Speeding Fermat's Factoring method," *Mathematics of Computation*, vol. 68, pp. 1729-1738, 1999.

[22] K. Somsuk, K. Tientanopajai, "An Improvement of Fermat's Factorization by Considering the Last m Digits of Modulus to Decrease Computation Time," *International Journal of Network Security*, vol.19, pp. 99-111, 2017.

[23] J.A. Buchmann, 'Introduction to Cryptography', USA: *Springer*, 2000.

[24] Computational complexity of mathematical operations, https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations (accessed: 2018)
[25] K. Somsuk, K. Tientanopajai, "Improving fermat factorization algorithm by dividing modulus into three forms," *KKU Engineering Journal*, vol. 43, pp. 350-353, 2016.

BIOGRAPHY OF AUTHOR

Kritsanapong Somsuk is an assistant professor of the department of Computer and Communication Engineering in Faculty of Technology, Udon Thani Rajabhat University, Udon Thani, Thailand. He received his M.Eng. (Computer Engineering) from department of Computer Engineering in Faculty of Engineering, Khonkaen University, M.Sc. (Computer Science) from department of Computer Science in Faculty of Science, Khonkaen University and his Ph.D. (Computer Engineering) from department of Computer Engineering in Faculty of Engineering, Khonkaen University. His research interests include computer security, cryptography and integer factorization algorithms.