Journal of the Nigerian Association of Mathematical Physics
***Volume 47 (July, 2018 Issue)*, pp187 – 192**
© J. of NAMP

ANALYSIS OF FAST DECRYPTION ALGORITHM FOR RSA CRYPTOSYSTEM

[1]Ibharalu, F.T., [1]Onashoga, S.A., [2]Olayiwola, O.M. and [1]Mesioye, A.E.

[1] Department of Computer Science, Federal University of Agriculture, Abeokuta, Ogun State, Nigeria
[2]Department of Statistics, Federal University of Agriculture, Abeokuta, Ogun State, Nigeria

Abstract

Number theory is known for its practicality in many fields including cryptosystem. RSA cryptosystem, one of the famous and widely used public key cryptosystem relies heavily on number theory for its operational and security implementations. With RSA algorithm success' story of security, more work need to be put in place to improve the computational cost in terms of speed. In this paper, a fast RSA variant that implement fast and efficient decryption algorithm is considered. In this work, an algorithm to generate low Hamming weight for prime difference generator is used to select the two large prime numbers p and q that are close and with very few numbers of 1. This variant combines both Hensel lifting and Chinese Remainder Theorem (CRT) to achieve its objective of speed enhancement. MIRACL library was compiled by C++ builders for both standard CRT-RSA decryption and the enhanced speed CRT–RSA decryption coined eCRT. This work is set to analyzed the two variants of CRT-RSA in terms of their execution time of the decryption stage as its relates to varied modulus size.

Keywords: *RSA; Decryption Algorithm; Hamming weight; Chinese Remainder Theorem; Hensel lifting; Cryptosystem*

1.0 Introduction

One of the famous and widely used public-key cryptosystem is the RSA algorithm. The system structure of RSA algorithm is based on the number theory, thus makes it the most security system in the key systems. The security of RSA algorithm as stated in [1] is based on difficulty in the factorization of the larger numbers. Research shows that large prime numbers p and q with long key sizes are necessary for safety of the cryptosystem. With a choice of large N (product of p and q) more than 100 decimal digits, the attackers cannot decompose the N in polynomial time. [2] outlines credit and debit payment smart cards, internet security, secure storage of secret keys, secure exchange of session keys and browser security as advantages of RSA algorithm that makes it suitable round the world. Due to its numerous applications as stated in [3], any leak in RSA security will lead to security breaches in the internet applications that rely on the RSA and make them a vulnerable to attacks. With the development of large number of decomposition technique, key length increases to ensure safety, which in turns increases the computation involved at both encryption and decryption stages of the algorithm. The decryption speed of the RSA algorithm is substantially slower than the encryption speed, due to the much longer decryption exponent. However, the more computationally demanding decryption is often performed by computationally-constrained devices with limited resources, such as a smart card. Different implementation and optimization methods have been suggested in [4], [5] and [6], to enhance the execution time of RSA algorithm. In practice, the Chinese Remainder Theorem (CRT) is deployed to combine the decryption computations performed in Z_p and Z_q in order to obtain the desired solution in Z_N, instead of directly computing the exponentiation in Z_N. According to [7], this method decreases the computational costs of decryption in RSA algorithm.

Corresponding Author**:** Ibharalu F.T., Email: tomibharalu@yahoo.com, Tel: +2348033447288

In this paper, we suggest a further enhanced variant of RSA coined eCRT. eCRT is designed and implemented to improve the execution time of RSA algorithm. In this work, an algorithm to generate low Hamming weight for prime difference generator is used to select the two large prime numbers p and q that are close and with very few numbers of 1. This variant combines both Hensel lifting and Chinese Remainder Theorem (CRT) to achieve its objective of speed enhancement. This work further analyse the speed (execution time) of the standard CRT-RSA and eCRT. The rest of the paper is organized as follows: section 2 gives a brief review of the RSA algorithm with few related work while section 3 introduces and analyzes theoretically the new approach. Section 4 gives the implementation, result analysis and discussion while section 5 concludes the study with an insight into future work.

2.0 THE RSA ALGORITHM AND RELATED WORK

2.1 The basic operation of RSA is represented as follows:

If Alice wishes to send a secret message M to Bob; Alice must use the Bob public key (KU_B) to encrypt the message, and Bob uses his private key (KR_B) to decrypt the encrypted message C, as the following steps.

Step1: Bob generates the key pair (KU_B, KR_B) by using Algorithm

Algorithm 1 Keys Generation Process

```
Procedure
  1: Choose two distinct large prime numbers p, q.
  2: Compute N as N = p x q.
  3: Compute Euler's totient function (N) as $\emptyset$(N) = (p x 1)(q x 1).
  4: Choose a random integer $e$ (public exponent), such that 1< $e$< $\emptyset$(N) and gcd($e$, $\emptyset$(N)) = 1.
  5: Calculate the private exponent d such that $ed$ = 1 mod $\emptyset$(N).
```

Step 2: Bob keeps the private key KR_B = (N,d) secret, while he publishes the public key KU_B =(N, e).

Step 3: Encryption process is done by the sender. In this scenario, the sender is Alice, such as Algorithm 2.

Algorithm 2 Encryption Process

```
Procedure
  1: Alice writes the message M she wishes to send to Bob, where M between 0 and N - 1.
  2: She uses a Bob public key (KUB) to encrypt the message M by computes C, such a
         C = $M^e$ mod N.
  3: She sends the ciphertext C to Bob.
```

Step 4: Decryption process is done by the receiver. In this scenario, the receiver is Bob, as Algorithm 3.

Algorithm 3 Decryption Process

```
Procedure
  1: Bob receives the ciphertext C from Alice.
  2:   He uses his private key (KRB) to decrypt the ciphertext C, to recover the original message M as
         M = $C^d$ mod N.
```

As shown in the steps before, the secret key remains with Bob, not given to Alice nor to any one from the public.

Illustration of RSA algorithm an example:

Example: (RSA encryption and decryption process)

Key pair generation**:** Bob does the following:

1. Selects p =11 and q = 13
2. Computes N as N = 11 X 13 = 143
3. Computes (N) as $\emptyset$(N) = 10 X 12 = 120
4. Selects e = 13
5. Computes d as d = 37
6. The public key is KU = (143,13)
7. The private key is KR= (143,37)

The private key (143, 37) is kept secret, also, Bob hides p, q and $\emptyset$ (N) while **the public key (143, 13)** is publish.

Encryption Process**:** Alice does the following

1. Selects the message M = 15
2. By using Bob public key, she calculates the ciphertext C = 15^{13}mod 143=119
3. Sends the ciphertext C to Bob

Decryption Process: Bob does the following

By using his private key, he computes the message as M= 119^{37}mod 143 = 15

Figure 1: Classical RSA Algorithm

Figure 1 gives an illustrative classical RSA algorithm showing the decryption process with a large number 119 raised to power of 37 (119^{37}mod 143). This mathematical operation involves exponentiation modular or 119 multiplied in 37 times before finding mod 143 of the result.

2.2 Related Work

Asymmetric key cryptography such as RSA is much slower in execution speed but more secure than any symmetric key cryptography and because of this, lot of research work has been done to improve the speed of RSA cryptosystem.

A technique to increase the speed of RSA decryption algorithm was developed in [8] by Couvreur and Quisquater. They use the concept of Chinese Remainder Theorem (CRT), which was called QCRSA that improves the basic RSA decryption performance. Furthermore, the concept of Batch RSA was introduced in 1989 to improve the speed of decryption. Batch RSA makes it possible to decrypt two cipher texts with small public exponent e at the cost of one. This technique is only valuable when the public key exponents e_1 and e_2 have small values.

Pradhan and Sharma in [9] proposed an efficient algorithm named BM-Prime method. The word BM came from two variants of RSA algorithm namely Batch RSA and Multi-Prime RSA. In order to speed up the decryption process, this approach combined both variants of RSA. This work produced an algorithm that was less vulnerable to various attacks on RSA. The results showed an improvement in decryption process time when compare to the classical RSA algorithm. BM-Prime method brought the significant advantage (which can be approved) in terms of speed when compare to the standard / classical RSA.

A modified approach to existing RSA algorithm to decrease its time complexity during communication over the network was introduced in [10] by Patidar and Bhartiya. The modification was done by the use of additional prime number to make it three instead of two as in the case of a standard RSA algorithm so that modulus n is not easily gotten by intruders. In addition, key values such as the encryption 'e' and decryption 'd' exponents of the system are stored in a database before use. Thus, increases the speed of encryption and decryption processes when compared to the traditional RSA method.

Amalarethinam et al, in [11] developed a methodology called Message Encoding Algorithm (MsgEncA) to increase the execution speed without changing the key size. This algorithm was designed to precisely fix block size of plain text that has been replaced by the binary form of ASCII code. The fix block size of plain text aids in achieving better performance in terms of speed. This algorithm performance of MsgEncA was compared to the existing Short Range of Natural Numbers (SRNN) method. The block ciphers increase the security and as well as the speed. It was discovered that when block size is fixed to less than the twice of key size, then the execution speed is at the best. Thus, the MsgEncA helps to decompose the plain text into specific fixed size and also raises the security and processing speed.

A swift and secure variant of RSA based on Rabin and Huffman coding called Augumented RSA (A-RSA) was designed and implemented in [3]. In their work, a new additional randomization component was provided and encryption done by Rabin algorithm to improve the security level of RSA against the indirect attacks and make RSA semantically secure. A-RSA makes the factorization problem harder, since the attackers need to break the factorization of large numbers for both RSA and Rabin. Beside this, Huffman coding compression in A-RSA prevents frequency of blocks (FOB) attacks and speeds up the execution time. With the introduction of an additional component, the execution is increased.

3.0 METHODOLOGY

In this approach, two large prime numbers p and q with low Hamming weight prime difference are generated after which a fast decryption algorithm is deployed to decrypt any encrypted data.

3.1 Algorithm for low Hamming Weight Prime Difference Generator

```
1.    Generate a prime p
2.    Generate the difference r
a.  Set $l = k - n,$ where $k$ is the required Hamming weight of the prime
    difference $r,\ l$ is a new Hamming weight to enable faster computation of $p$
    and $q, n\ (n \neq 0)$ is a positive integer ($n = 6$ is sufficient)
b.  Set $f = \frac{c}{l},\ r = 0,\ r_0 = 0,\ r_{c-1} = 1,\ i = 1.$
    Where $|r| = c,$ and $r_x$ is the $x^{th}$ bit position of c.
    While ($|r| > i$ and $H(r) < l$)
i.      $i = i + R(f),$ where $R(f)$generates a random positive integer greater or
        equal to 0 and smaller than $f$.
ii.     Set $r_i = 1.$
```

iii. $i = i + (f - R(f))$, the same $R(f)$ as in c(i)
3. Calculate $q = p + r$.
4. While q is not prime $q = q + 2$.
5. Set $r = q - p$. (Now r has become a prime difference)
6. If $H(r) \neq k$, go back to step 1 (generate a prime p) else return p and q.
7. Output two primes p and q with difference r. The prime difference has a size c and Hamming weight h.

3.2 The Fast Decryption Algorithm

i. With the output p and q (primes) from the algorithm above where $|r| \geq c$, $H(r) > h$, c and h are specific constants.

ii. Calculate the decryption exponent d, using

$$d \equiv (p^2q) \bmod (p-1)(q-1) \quad (1)$$

iii. Calculate the encryption exponent e, using the Extended Euclidean algorithm

$$e \equiv d^{-1} \bmod (p-1)(q-1) \quad (2)$$

iv. Using the Chinese Remainder theorem to deduce d_p and d_q from the choice of d in (1).

$d_p \equiv d \bmod (p-1)$
$\equiv (p^2q) \bmod (p-1)$
where $p^2 \bmod (p-1) = 1$
$\equiv q \bmod (p-1)$
where $q = r + p$,
$\equiv (r+p) \bmod (p-1)$
$\equiv (r+1) \bmod (p-1)$ (3)

Likewise,
$d_q \equiv d \bmod (q-1)$
$\equiv (p^2q) \bmod (q-1)$
$\equiv p^2 \bmod (q-1)$
where $p = q - r$,
$\equiv (q-r)^2 \bmod (q-1)$
$\equiv (r-1)^2 \bmod (q-1)$ (4)

This method is efficient because p^2q and $(p-1)(q-1)$ will always be relatively prime and no need for looping in the calculation of the encryption exponent e. It is important to note from (1) that the decryption exponent as well as the encryption exponent both will be the same bit size as the modulus and not half the bit size as other decryption algorithms.

The suggested values for c and h for standard RSA modulus sizes are given in the Table 1 below.

Table 1: The constants c and h for standard RSA modulus sizes.

Modulus size	c	h
1024	412	24
1536	618	21
2048	824	20

3.3 Theoretical Analysis of the New Approach and its Key Generators

The general equation in terms of operation cost (small multiplications) for decrypting cipher text is given by:

$$\frac{3}{2}(|n|)(|n|^2) = \frac{3}{2}|n|^3$$

where n represents the modulus RSA encryption and is equal to pq.

The factor $\frac{3}{2}$ comes from the standard square-and-multiply algorithm [12].

When CRT is employed, both the modulus and exponent are reduced by a factor of 2.

$$2\left(\frac{3}{2}\right)\left(\frac{|n|}{2}\right)\left(\frac{|n|}{2}\right)^2 = \frac{3}{8}|n|^3 \quad (5)$$

As discovered from equations (3) and (4), only d_p will be of size c and have hamming weight approximately equal to h. From (4), d_q will be half the modulus and have no special hamming weight characteristics.

These findings make it possible to write (5) as

$$\left(\left(\frac{3}{2}\right)\left(\frac{|n|}{2}\right)+c+h\right)\left(\frac{|n|}{2}\right)^2 \tag{6}$$

Assuming $h = 0$ due to the fact that $h \ll c$ and therefore has little effect on the amount of long multiplications generated. Substituting for $c = \frac{103|n|}{256}$ as appropriate value in (6) leads to

$$\frac{295}{1024}|n|^3 \tag{7}$$

Dividing (5) by (7) gives the appropriate improvement factor of the fast decryption algorithm on standard CRT decryption. The factor equals $1\frac{89}{295} \approx 1.301695\ldots$. This implies fast decryption algorithm improves standard CRT decryption by about 23%.

4.0 IMPLEMENTATION, RESULT ANALYSIS AND DISCUSSION

The standard CRT decryption and fast decryption algorithms were implemented on an Intel Core i5, 2.20GHz with 2GB RAM, running the Microsoft Windows 7 version. The application used the MIRACL library and was compiled by C++ Builder 6. Four different security level n, where $(n = p \times q)$ were used to determine execution times for the two decryption algorithms and the results of the simulation are summarized in Table II.

Table II: Execution Times for Decryption Algorithms

Decryption Algorithms	**Security level**			
	512 bits	**1024 bits**	**1535 bits**	**2048 bits**
Standard RSA without CRT	4.1ms	25.6ms	77.3ms	166.9ms
Standard RSA with CRT	1.9ms	8.8ms	23.1ms	49.1ms
Fast (New Approach)	Not Applicable	6.58ms	17.9ms	36.9ms

From Table II, it is clear that to improve the execution times of RSA cryptosystem at the decryption stage, CRT can be implemented on the decryption algorithm.

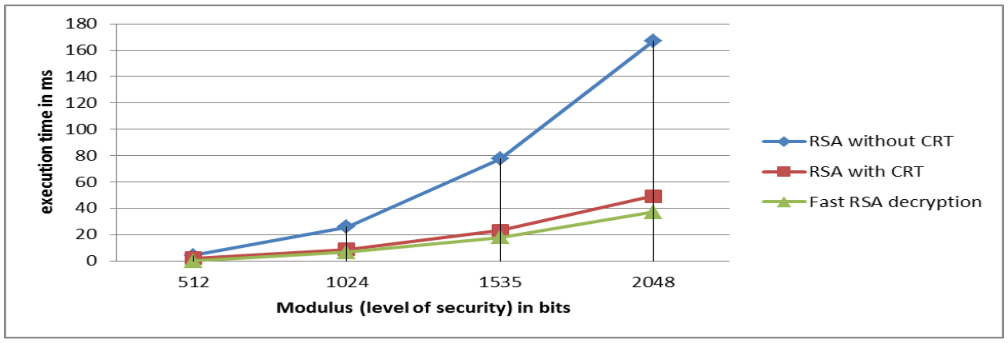

Figure 2: The Execution times of three variants of RSA decryption.

Furthermore, the speed of the standard RSA-CRT can be improved using the eCRT decryption algorithm. This method brings about 23% improvement over the standard RSA-CRT in terms of execution speed of the system.

Table III: Practically obtainable speed increase on RSA- CRT Decryption algorithm

Algorithm	**Modulus Size**	**% improvement over the Standard RSA-CRT**
Standard RSA-CRT	$\frac{\lvert n\rvert}{2}$	
Fast (eCRT) RSA-CRT	n	23%

5.0 CONCLUSION

Table III shows the improvement of the decryption process of RSA-CRT decryption algorithm. This was done because it has become standard to implement RSA using CRT at the decryption stage, so any algorithm that is design must be set out to improve on CRT decryption to be of use in the industry. The standard RSA-CRT algorithm has been proved to increase the speed of classical RSA by 4 times (25%), this method further increase the RSA-CRT by another 23% thus make the new approach better in terms of speed without compromising the security of RSA.

REFERENCES

[1] **Zhanga, M., Chanle Wub and Gang Ye** (2015). A New RSA Algorithm of Data Security by Large Numbers and Binary Stream Processing Methods. Journal of Information & Computational Science. Available at http://www.joics.com. pp. 5411–5418.

[2] **Rivest, R., Shamir, A. and Adleman, L.** (1978). A Method for Obtaining Digital Signature and Public-key Cryptosystems. Communications of the ACM, vol. 21, no. 2, pp. 120-126.

[3] **Karakra, A. abd Alsadeh A.** (2016). A-RSA: Augmented RSA. SAI Computing Conference 2016, London UK.

[4] **Boneh, D and Shacham, H**. (2002). Fast variants of RSA. CryptoBytes, vol. 5, no. 1, pp.1-9.

[5] **Takagi, T.** (1998). Fast RSA-Type Cryptosystem modulo $p^k q$. Advanced in Cryptology – CRYPTO'98. Springer, pp. 318 – 326.

[6] **Hwang, R.J., Su, F.F., Yeh, Y.S., and Chen, C.Y.** (2005). An efficient decryption method for RSA Cryptosystem. 19th International Conference on Advanced Information Networking and Applications. IEEE, pp. 585 – 590.

[7] **Sun, H.-M, Wu, M.-E. and Ting, W.-C.** (2007). Dual RSA and its security analysis. IEEE Transactions on Information Theory. pp. 2922 -2932.

[8] **Couvreur, C and Quisquater, J.J.** (1982). Fast Decipherment Algorithm for RSA Public-Key Cryptosystem. Electronic Letter, vol. 18.

[9] **Pradhan, S. and Sharma, B.,** (2013). An Efficient RSA Cryptosystem with BM-PRIME Method. International Journal of Information and Network Security (IJINS). Volume 2, Number 1. Page 103 -108.

[10] **Patidar, R. and Bhartiya, R.** (2014). Implementation of Modified RSA Cryptosystem Based on Offline Storage and Prime Number. International Journal of Computing and Technology (IJCAT). Volume 1, Issue 2. ISSN: 2348-6090, pp. 205 – 209.

[11] **Amalarethinam, D. I. G., Sai Geetha, J. and Mani, K.** (2015). Analysis and Enhancement of Speed in Public Key Cryptography using Message Encoding Algorithm. Indian Journal of Science and Technology, www.indjst.org, Vol. 8(16), pp. 1 – 7.

[12] **Menezes, A., Van Oorschot, P., and Vanstone, S.** (1997). Handbook of Applied Cryptography. 1st Edition, CRC Press.