HENSEL LIFTING: A TOOL FOR FAST DECRYPTION PROCESS IN RSA CRYPTOSYSTEM

By

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

[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*

ayobamimesioye@gmail.com

Abstract

RSA cryptosystem is known for its effective key exchange management, integrity of data and security. The security strength of RSA cryptosystem lies in the size (length) of the keys, which slows down the operation at the decryption stage of the cryptosystem. Chinese Remainder Theorem (CRT) has been employed in the past to increase the overall performance of the RSA cryptosystem (CRT-RSA) but stills possesses time lag in execution. In this work, a modified CRT (MCRT-RSA) is developed to further improve the speed of RSA at the decryption stage. Algorithms of the classical RSA (RSA), CRT-RSA and MCRT-RSA at the decryption stage were implemented using Visual Studio C- sharp. Analysis of algorithms was carried out by measuring the complexity of each RSA variants through the big-O test with six data characters. The three RSA variants were compared using the parameters of time (t_c) and space (S_c) complexities. Results generated by MCRT-RSA indicated $O(n^2)$ and $O(n^3)$ for t_c and S_c respectively, while t_c and S_c of CRT-RSA were $O(n^3)$ and $O(n^3)$ respectively and for RSA, t_c and S_c were $O(n^4)$ and $O(n^4)$ respectively. Conclusively, MCRT-RSA was found to be more efficient in execution time at the cost of memory usage when compared to RSA and CRT- RSA variants.

Keywords— *Cryptography; Chinese Remainder; Hensel Lifting; Homomorphic encryption; MultiPower RSA; Classical RSA*

Introduction

Basically, RSA cryptosystem implementation involves three different stages namely key generation, encryption and decryption processes. These stages depend very much on each other in order to optimize efficiency as well as computation costs. Key generation produces pairs of public and private keys from sufficiently large numbers to make the reverse operation practically impossible and thus make the system secure [1]. Its strength lies on the difficulty of factoring large numbers. Public and private keys produced at the key generation are used for encryption and decryption process respectively. Both encryption and decryption processes are based on performing exponentiation in modular arithmetic, which may be stated as follows:

$$N = a^b mod\ m$$

where a is the base, b is the exponent and m is the modulus.

The required modular exponentiation is computed by a series of modular multiplications. This is performed in two steps: first an integer multiplication is done followed by a reduction by modulo *m*. The computation and processing time for this operation is large on the account of large key size in use resulting to slow rate of the whole RSA cryptosystem. Modular exponentiation ($a^b mod\ m$) and its key constituent operation, modular multiplication ($a \cdot b\ mod\ m$), are the fundamental operations underlying RSA algorithm. Since modular multiplication accounts for most of the time spent for encryption and decryption, then the optimization of both modular exponentiation and multiplication is crucial.

To address the long computational time in RSA cryptosystem, small value for the public key e is considered. This choice of public key can only speed up the encryption operation

resulting to a more computational time at the decryption stage due to a larger decryption exponent d generated mathematically in the relationship between e and d [2]. To avoid the transfer of computational time of RSA cryptosystem between the encryption and decryption process, a mathematical concept called Chinese Remainder Theorem (CRT) has been deployed in the past. Since the complexity of RSA decryption depends on the size of the decryption exponent, d and the modulus, n, CRT is deployed to reduce the size of both d and n. [3] has proved the improvement equated the computational time of RSA cryptosystem at the decryption stage by method as 25%.

In this work, the generalized Multi-Power RSA is used as a base to suggest a variant RSA algorithm with improved speed. Before applying CRT, Hensel's lemma is used to stimulate conditions for roots of polynomials modulo powers of primes to be expressed as roots of higher powers. This lowering of modulo powers reduces the numbers of modular exponentiation and multiplication which in turn speeds up the execution time of the developed RSA variant.

The rest of the paper is organized as follows: section 2.0 reviews related works done in this area, while section 3.0 discuss the material and method of MCRT-RSA, section 4.0 presents the results and its discussion and section 5.0 presents the conclusion.

RELATED WORK

Public key cryptography such as RSA is more secured but much slower than the symmetric key cryptography. Due to the security strength of public-key cryptography, lot of research work has been done to improve on its execution speed.

Nikita, S. and Dharmendra, M., (2014) three prime numbers were deployed in their scheme to increased difficulty in factorisation of variable n. The algorithm has slight speed improvement on the decryption side of RSA algorithm by using the concept of Chinese remainder theorem and the method also improved the security of RSA algorithm by avoiding some attacks that are possible on RSA algorithm like common modulus attack, chosen ciphertext attack, timing attack and known plaintext attack. The scheme is computationally less expensive when compared to some other RSA algorithm. The security enhancement is more pronounced than the speed enhancement due the introduction of an additional prime number.

Zhang et al (2015) introduced an asymmetric cryptosystem to improve the Montgomery algorithm. The RSA algorithm was improved by annexing large numbers as a string and by employing binary stream processing methods. The key generation algorithm was used for encryption and decryption verification. Results showed a significant improvement in encryption/decryption speed. The improved RSA algorithm used in digital signature applications results in an improved digital signature and offers evident advantages. The results have significant theoretical and practical significance in the efforts to improve large data processing encryption. The algorithm has a little significant effect on the speed improvement on large data processing. This method is slow and thus not suitable when used for large data.

Thangavel et al, (2015) proposed ESRKGS which stands for enhanced and secured key generation algorithm. The proposed algorithm makes use of four large prime numbers which increases the complexity of the system when compared to the standard RSA algorithm that is based on only two large prime numbers. The public component (n) is the product of

two large prime numbers but the values of the encryption (e) and decryption (d) keys are based on the product of four large prime number (N) making the system highly secured. With the existing factorization techniques, it is possible to find the primes p and q only. The knowledge of n alone is not sufficient to find e and d as they are based on N. This approach could withstand brute force attack better than other similar approaches but with more time complexity.

Amalarethinam et al, (2015) proposed a methodology called Message Encoding Algorithm (MsgEncA). 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. On applying the proposed algorithm MsgEncA, the speed of execution is increased without changing the key size. They proved this by comparison with existing method SRNN. The block ciphers increase the security and as well as the speed. They 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 modified algorithm helped to decompose the plain text into specific fixed size and also raises the security and processing speed. The limitation of this scheme lies in the massive use of the computer memory due to the fix block size necessary in plain text.

Karakra and Alsadeh, (2016) implemented a swift and secure variant of RSA based on Rabin and Huffman coding called Augumented RSA (A-RSA). 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.

MATERIAL AND METHODS

The generalized MultiPower RSA is used to suggest a variant of RSA algorithm with improved speed and this is deployed at the decryption stage of the cryptosystem.

Decryption: Given the private key p_k. To decrypt the ciphertext C, one uses

- Hensel lifting
- Chinese Remainder Theorem

Step 1: Compute $C_o \equiv C \ (mod\ p),\ C_p \equiv C(mod\ p^r),$

$C'_o \equiv C \ (mod\ q),\ C_q \equiv C(mod\ q^s),$

Then, compute $M_o \equiv C_o^{d_p}(mod\ p),$

$M'_o \equiv C'^{d_q}_o(mod\ q)$

Step 2: Using the Hensel lifting, construct M_P, and M_q such that

$C_p \equiv M_p^e \ (mod\ p^r),$

$C_q \equiv M_q^e \ (mod\ q^s)$

Step 3: Using Chinese Remainder, recover the plaintext $M = m_1 \times m_2 \times \ldots \times m_k$ such that

$M \equiv M_p(Mod\ p^r),$

$M \equiv M_q(Mod\ q^s)$

$M \equiv (M_p \times k_p) + (M_q \times k_q) \ \ mod\ N$

Where $p, q \ \ are\ prime\ number\ and\ r, s \ \ are\ integers \geq 2.$

Details of the decryption algorithm

By Hensel's lifting, the plaintexts $M_p \in Z_{p^r}$, $and\ M_q \in Z_{q^s}$ have $p-adic$ and $q-adic$ expansion to p^r and q^s respectively.

That is $M_p \equiv M_0 + pM_1 + p^2M_2 + \cdots + p^{r-1}M_{r-1}\ (mod\ p^r)$,

and $M_q \equiv M'_0 + qM'_1 + q^2M'_2 + \cdots + q^{s-1}M'_{s-1}\ (mod\ q^s)$

where $M_i \in Z_p$, $M'_j \in Z_q$ and $M''_l \in Z_l$ for $i = 0,1,2,\ldots,r-1$, and $j = 0,1,2,\ldots,s-1$.

From the relationship, $C_p \equiv M_p^e\ (mod\ p^r)$, the values $M_1, M_2, \ldots, M_{r-1}$ can be calculated.

Suppose the function $A_i(M_0, M_1, \ldots, M_i) = (M_0 + pM_1 + p^2M_2 + \cdots + p^iM_i)^e$,

where $i = 0, 1, \ldots, r-1$.

Reduce by modulo p^{i+1}, the following relationship would be derived

$A_i(M_0, M_1, \ldots, M_i) \equiv A_{i-1} + p^i eM_0^{e-1}M_i\ (mod\ p^{i+1})$

where $A_{i-1} = M_0 + pM_1 + p^2M_2 + \cdots + p^{i-1}M_{i-1}^e$ for $i = 0,1,2,\ldots,r-1$.

Then, the values $M_1, M_2, \ldots, M_{r-1}$ are recursively calculated by the following relationship

$$M_i \equiv \frac{(C_p - A_{i-1}\ (mod\ p^{i+1})}{p^i} \times e^{-1}M_0^{1-e}\ (mod\ p)$$

Similarly, the values $M'_1, M'_2, \ldots, M'_{s-1}$ would be derived by the relationship

$$M_j \equiv \frac{(C_q - A'_{j-1}\ (mod\ q^{j+1})}{q^j} \times e^{-1}M'^{1-e}_0\ (mod\ q)$$

MCRT-RSA Decryption Algorithm

The decryption process discussed above can be summarized in the algorithm below:

Input: a private key $p_k = (N,\ p,\ q, r, s, e, d_p, d_q,\ k_p,\ k_q)$ and a ciphertext $C \in Z_N$

Output: $M \in Z_N$;

Procedure:

1: Compute $C_0 \equiv C\ (mod\ p)$, $C_p \equiv C\ (mod\ p^r)$,
$C'_0 \equiv C\ (mod\ q)$, $C_p \equiv C\ (mod\ p^s)$

2: Compute $M_0 \equiv C_0^{d_p}\ (mod\ p)$, $M_0' \equiv C'^{d_q}_0\ (mod\ q)$

3: Set $K_0 \leftarrow M_0$, $K'_0 \leftarrow M'_0$ and $K''_0 \leftarrow M''_0$

4: **for** $i = 1$ ***to*** $r-1$ **do**

5: $E_i \leftarrow K_{i-1}^e\ (mod\ p^{i+1})$

6: $F_i \leftarrow C_p - E_i\ (mod\ p^{i+1})$

7: $G_i \leftarrow F_i/p^i$

8: $M_i \leftarrow G_i \times (eK_0^{e-1})^{-1}\ (mod\ p)$

9: $K_i \leftarrow K_{i-1} + p^iM_i$

10: **end for**

11: Set $M_p \leftarrow K_{r-1}$

12: **for** $j = 1$ ***to*** $s-1$ **do**

13: $E'_j \leftarrow K'^e_{j-1}\ (mod\ q^{j+1})$

14: $F'_j \leftarrow C_q - E'_j\ (mod\ q^{j+1})$

15: $G_j \leftarrow F'_j/q^j$

16: $M'_j \leftarrow G'_j \times (eK'^{e-1}_0)^{-1}\ (mod\ q)$

17: $K'_j \leftarrow K'_{j-1} + q^jM'_j$

18: **end for**

19: Set $M_q \leftarrow K'_{s-1}$

20: Compute $M \leftarrow (M_p \times k_p) + (M_q \times k_q)\ (mod\ N)$

Correctness Proof

In this section, the correctness of the speedy RSA$^+$ algorithms is considered. Let $N = p^r q^s$ where p and q are two distinct primes and $r, s, \geq 2$. Let e, d be two integers satisfying

$ed \equiv 1 \, mod \, (lcm \, (p-1)(q-1))$ and $\gcd(e, N) = 1$.

Denote the ciphertext reduced modulo p by C_0, the ciphertext reduced modulo p^r by C_p, the ciphertext reduced modulo q by C'_0, and the ciphertext reduced modulo q^s by C'_q.

Suppose that $d_p \equiv d \, (mod \, p-1)$, and $d_q \equiv d \, (mod \, q-1)$ that is $\exists \; a_1, a_2 \in Z$ such that

$$d = d_p + a_1(p-1), \text{ and } d = d_q + a_2(q-1)$$

Recall that $C \equiv M^e \, (mod \, N)$ where $M \in Z_N$.

Decrypt C modulo p gives

$$M_0 \equiv C^d \, (mod \, p)$$
$$\equiv (C_0 + bp)^d \, (mod \, p) \quad \text{where } b \in Z.$$
$$\equiv {C_0}^{d_p + a_1(p-1)} \, (mod \, p)$$

By Fermat little theorem,

$$\equiv {C_0}^{d_p} \, (mod \, p)$$

Similarly, ${M'}_0 \equiv C'^d \, (mod \, q)$.

By Hensel lifting lemma, the relationship between the ciphertext C_p and the plaintext M_p is such that

$C_p \equiv M_p^e \, (mod \, p^r)$. Suppose M_p has $p-$adic expansion such that

$M_p \equiv M_0 + pM_1 + p^2 M_2 + \cdots + p^{r-1} M_{r-1} \, (mod \; p^r)$, where $M_0, \ldots, M_{r-1} \in Z_p$.

Let the function $A_i(M_0, M_1, \ldots, M_i)$ be defined as follows:

$A_i(M_0, M_1, \ldots, M_i) \equiv (M_0 + pM_1 + p^2 M_2 + \cdots + p^i M_i)^e$, where $0 \leq i \leq r-1$.

Then reduce C_p modulo p^{i+1} using binomial theorem,

$$A_i(M_0, M_1, \ldots, M_i) \equiv (M_0 + pM_1 + p^2 M_2 + \cdots + p^i M_i)^e \; (mod \, p^{i+1})$$
$$\sum_{k=0}^{e} C_k^e (M_0 + pM_1 + p^2 M_2 + \cdots + p^{i-1} M_{i-1})^{e-k} (p^i M_i)^k \; (mod \, p^{i+1})$$
$$\equiv (M_0 + pM_1 + \cdots + p^{i-1} M_{i-1})^e + (M_0 + pM_1 + \cdots + p^{i-1} M_{i-1})^{e-1} e p^i M_i \; (mod \, p^{i+1})$$

$$\equiv M_0 + \cdots + p^{i-1} M_{i-1})^e + e p^i M_i \times \sum_{j=0}^{e-1} C_j^{e-1} M_0^{e-1-j} \times (pM_1 + \cdots + p^{i-1} M_{i-1})^j \; (mod \, p^{i+1})$$
$$\equiv (M_0 + pM_1 + \cdots + p^{i-1} M_{i-1})^e + p^i e M_0^{e-1} M_i \; (mod \; p^{i+1})$$

We note that $A_{i-1} = (M_0 + pM_1 + p^2 M_2 + \cdots + p^{i-1} M_{i-1})^e$ for $i = 0, 1, 2, \ldots, r-1$.

It follows that, the values $M_0, M_1, \ldots, M_{r-1}$ can be recursively calculated by the relationship

$$M_i \equiv \frac{(c_p - A_{i-1} \; (mod \, p^{i+1})}{p^i} \times e^{-1} M_0^{1-e} \; (mod \, p)$$

All the previous steps can apply equally well to prove the correctness of the relation

$$M_j \equiv \frac{(c_q - A'_{j-1} \; (mod \, q^{j+1})}{q^i} \times e^{-1} {M'}_0^{1-e} \; (mod \, q)$$

Note that since e is relatively prime to $N, M_0 \in Z_p$ and ${M'}_0 \in Z_p$. It follows that e, M_0^{e-1} have reverse modulo p and e, ${M'_0}^{e-1}$ have reverse modulo q.

IMPLEMENTATION RESULTS AND DISCUSSION

This section shows execution time and memory space consumed by the three RSA variants with ten different character lengths of messages on the three RSA variants.

Table 1: Decryption time of the three RSA variants

	Character Size *(Bytes) x*	**Decryption Time (Milliseconds) *Y***		
S/N		**Classical RSA**	**CRT-RSA**	**CRT-RSA2**
1.	3	103.53	29.41	21.23

2.	15	103.30	29.45	21.34
3.	24	103.44	29.48	21.45
4.	42	103.52	29.50	21.55
5.	56	102.25	29.46	22.10
6.	73	103.17	29.35	22.23
7.	80	104.10	30.05	22.34
8.	97	104.22	30.11	22.45
9.	103	104.24	30.12	22.55
10.	120	104.30	30.15	23.13

A polynomial curve equation $Y = f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ is derived connecting x and Y.

Where: $a_0, a_1, \ldots, a_{n-1}, a_n$ are set of real numbers.

$y = f(x)$ is the result/behaviour of the function with various input variables in terms of time.

$x^n, x^{n-1}, \ldots, x^2, x$ are the input variables (character).

Using six of the ten character size in the table above, derive the values of the constants $a_0, a_1, \ldots, a_{n-1}, a_n$ with the help of Crammer online software.

$a_5 = -2.92 \times 10^{-4}, a_4 = 5.92 \times 10^{-3}, a_3 = -0.41, a_2 = +315, a_1 = +26.16, a_0 = +6903$ and the polynomial curve equation for MCRT-RSA is given as:

$$y = f(x) = -2.92 \times 10^{-4}x^5 + 5.92 \times 10^{-3}x^4 - 0.41x^3 + 315x^2 + 26.16x + 6903$$

In the same view, constants for CRT-RSA algorithm are:

$a_5 = -5.32 \times 10^{-4}, a_4 = -1.92 \times 10^{-3}, a_3 = +0.41, a_2 = -315, a_1 = +26.16, a_0 = +6903$

and the mathematical model for the time analysis of the decryption module of CRT-RSA from polynomial-time function is given by

$$y = f(x) = -5.32 \times 10^{-4}x^5 - 1.92 \times 10^{-3}x^4 + 0.41x^3 - 315x^2 + 26.16x + 6903$$

While for the classical RSA, the constants are:

$a_5 = -7.37 \times 10^{-4}, a_4 = +0.13, a_3 = -84.57, a_2 = +208.18, a_1 = -1485.94, a_0 = +24002$

With the polynomial-time function given by :

$$y = f(x) = -7.37 \times 10^{-4}x^5 + 0.13x^4 - 84.57x^3 + 208.18x^2 - 1485.94x + 24002$$

Table 2: Analysis for the Big-O Complexity Test for Decryption running time of MCRT-RSA algorithm.

		$y = f(x) = -2.92 \times 10^{-4}x^5 + 5.92 \times 10^{-3}x^4 - 0.41x^3 + 315x^2 + 26.16x + 6903$						
		x						
Degree of x	Coefficient of x	1	37	40	88	90	128	230
x^5	-2.92×10^{-4}	−.000292	-20248.44	-29900.80	-1540977.20	-1724230.8	-1724230.8	-187941215.6
x^4	$+5.92 \times 10^{-3}$	+0.005.92	11095.03	15155.20	355019.65	388411.2	1589137.9	16566587.2
x^3	−0.41	−0.41	-20767.73	-26240.0	-279403.52	-298890	-859832.32	-4988470.0
x^2	+315	**315**	**431235**	**504000**	**2439360**	**2551500**	**5160960**	**16663500.0**
x	+26.16	26.16	967.92	1046.4	2302.08	2354.4	3348.48	6016.8
x^0	+6903	+6903	6903	6903	6903	6903	6903	6903

Table 2 shows the analysis of MCRT-RSA algorithm that was carried out by measuring the complexity the algorithm through the Big-O Test with a series of test input data character (byte). The result of the calculation shows that the values of x appear highest in the range of degree 2 for Big-O MCRT-RSA decryption algorithm. That is, the Big-O notation in time complexity is $O(n^2)$.

Table 3: Analysis for the Big-O Complexity Test for Decryption running time of CRT-RSA algorithm

		$y = f(x) = -5.32 \times 10^{-4}x^5 - 1.92 \times 10^{-3}x^4 + 0.41x^3 - 315x^2 + 26.16x + 6903$						
		x						
Degree of x	Coefficient of x		37	40	88	90	128	230
x^5	-5.32×10^{-4}	$-.000532$	-36891	-54477	-2807533	-3141407	-18279381	-342413448
x^4	-1.92×10^{-3}	$-.00192$	-3598	-4915	-115142	-125971	-515396	-5372947
x^3	$+0.41$	$+0.41$	**+20768**	**+26240**	**+279404**	**+298890**	**+859832**	**+4988470**
x^2	-315	-315	-431235	-504000	-2439360	-2551500	-5160960	-16663500
x	$+26.16$	$+26.16$	+968	+1046	+2302	+2354	+3348	-+6017
x^0	$+6903$	$+6903$	$+6903$	$+6903$	$+6903$	$+6903$	$+6903$	$+6903$

Table 3 shows the analysis of CRT-RSA algorithm that was carried out by measuring the complexity the algorithm through the Big-O Test with a series of test input data character (byte). The result of the calculation shows that the values of x appear highest in the range of degree 3 for Big-O CRT-RSA decryption algorithm. That is, the Big-O notation in time complexity is $O(n^3)$.

Table 4: Analysis for the Big-O Complexity Test for Decryption running time of Classical RSA algorithm

		$y = f(x) = -7.37 \times 10^{-4}x^5 + 0.13x^4 - 84.57x^3 + 208.18x^2 - 1485.94x + 24002$						
		x						
Degree of x	Coefficient of x	1	37	40	88	90	128	230
x^5	-7.37×10^{-4}	$-.000737$	-51106.50	-75468.8	-3889384.23	-4351911.3	-25323127.18	-474358479.1
x^4	$+0.13$	$+0.13$	**243640.93**	**332800**	**7796039.68**	**8529300**	**34896609.28**	**363793300**
x^3	-84.57	-84.57	-4283724.2	-5412480	-57632087.04	-61651530	-177356144.6	-1028963190
x^2	$+208.18$	$+208.18$	284998.42	333088	1612145.92	1686258	3410821.12	11012722
x	-1485.94	-1485.94	- 54979.78	-59437.6	-130762.72	-133734.6	-190200.32	-190200.32
x^0	$+24002$	$+24002$	24002	24002	24002	24002	24002	24002

Table 4 above shows the analysis of classical RSA algorithm for decryption module, it is noted that the highest value of the function $y = f(x)$ is at the degree of $x = 4$ The Big-O notation in testing the complexity is $O(n^4)$. This reflects a longer execution at the decryption stage of the classical RSA algorithm than CRT-RSA.

In summary, the computational complexity of MCRT-RSA, CRT-RSA and classical RSA variants of order $O(n^2)$, $O(n^3)$ and $O(n^4)$ respectively. This shows that MCRT-RSA algorithm is fast at decrypting ciphertexts to plaintext.

CONCLUSION

In this work, MCRT-RSA algorithm, a fast RSA decryption process was developed. This approach was implemented using C♯ in visual studio environment. The complexity measure of this algorithm helps in its performance evaluation; this shows that computational complexity of MCRT-RSA and its comparison to classical RSA and CRT-RSA. The results show that the computational complexity of MCRT-RSA in terms of speed is lower than the two other RSA variants compared. It is due to the execution time incurred in performing the decryption process of MCRT-RSA variant.

Referees

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.

Hailiza, K.H. and Norfadhilah, B. (2009). RSA Decryption Techniques and the Underlying Mathematical Concepts. International Journal of Cryptology Research. Volume 1, Issue 2. pp.165 – 177.

Grobler, T.L. and Penzhorn, W.T. (2014). Fast Decryption Methods for the RSA Cryptosystem.

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.

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

Delfs, Hans and Knebl, Helmut (2007); Introduction to Cryptography; Springer-Verlag Berlin Heidelberg.

Diffie, W. and Hellman, M. (1976). New Direction in Cryptography. IEEE Transaction on Information Theory, vol. 22, pp. 644-654, 1976.

Hardy, G.H. and Wright, E.M. (1979). An Introduction to the Theory of Numbers. Oxford University Press.

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

Kelly, M.D. (2009). A Mathematical History of the Ubiquitous Cryptological Algorithm. pp.1 -14.

Koblitz, N. (1994). A Course in Number Theory and Cryptography. Graduate Texts in Mathematics 114. New York: Springer-Verlag.

Lim, S., Kim, S., Yie, I., and Lee, H. (2000). A generalized takagi-cryptosystem with a modulus of the form $p^r q^s$. In Progress in Cryptology-INDOCRYPT 2000, pages 283-294, Springer.

Menezes, A., Van Oorschot, P. and Vanstone, S. (2001). Handbook of Applie Cryptography. Discrete Mathematics and its Applications Vol. 6. CRC Press.

Nikita, S. and Dharmendra, M. (2014). An Improved RSA Cryptgraphic System. International Journal of Computer Applications (IJCA). Volume 105, Number 16.

Penzhorn, W.T. (2004). Fast decryption Algorithms for the RSA cryptosystem. *Proceedings to AFRICON 2004 Conference at Gabarone, Botswana, 16-19 September 2004*, vol. 1, pp. 361–364.

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.

Stallings, W. (2011). Cryptography and Network Security: Principles and Practice. 5th edition. Pearson Education; pp. 121 – 144, 253 – 297.

Takagi, T. (1998). Fast RSA-Type Cryptosystem Modulo $p^k q$. Proceedings of Crypto98. Vol.1462, pp. 318 – 326.

Thangavel, M., Varalakshmi, P., Mukund Murrali, Nithya, K., (2014) . An Enhanced and Secured RSA Key Generation Scheme. Department of Information Technology, Anna University, Chennai, 2214-2126 Elsevier.

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.