ISSN:
Print - 2277 - 0593
Online - 2315 - 7461
© FUNAAB 2017

Journal of Natural Science, Engineering and Technology

RSA ENCRYPTION ALGORITHM AUGUMENTED WITH BIT-STUFFING TECHNIQUE FOR DATA SECURITY

IBHARALU F. T, FALOWO M. O. AND AKINWALE A. T.

Department of Computer Science, Federal University of Agriculture, Abeokuta, Nigeria
***Corresponding author:**tomibharalu@yahoo.com,, **Tel:** +2348033447288

ABSTRACT

Data transmission through the internet applications is growing very fast, and this continuous growth demands for new network bandwidth and data security. Encryption plays a major role in security of information systems and internet based applications. In this study, the RSA algorithm was modified with bit-stuffing technique to improve the protection and security of confidential data while in transits or in storage. Our modified algorithm, RSA Bit-stuffed, was implemented and compared with the modified Ron Divest Code4 and the modified RSA in MATLAB using time complexity and avalanche effect as performance metrics. The experimental results showed that our augmented bit-insertion technique increased the time complexity against different attacks, boost the randomness of encrypted messages, and also improve security of encryption keys with bit-length lower than that of the standard RSA.

Keywords: Avalanche effect, Bit-insertion, Encryption, Metrics, RSA, Time complexity.

INTRODUCTION

Cryptography is a powerful tool for protection of messages which entails many of the security mechanisms and built the science of data encryption and decryption (Singh, 2011). Cryptography refers to the ability to store data and transform such data in a particular form that only an authenticate user can access and process (Yogita and Praveen, 2015). Cryptography applications include e-shopping, banking transactions, password and etc. Encryption and decryption process are done in all branches of cryptography which includes asymmetry and symmetry key cryptography. Encryption is the cryptography process of converting plaintext (readable) to cipher text (non-readable) form so as to prevent access from unauthorized agents. Strong encryption guarantees integrity, confidentiality, privacy, access control, among others (Pradhan and Sharmal, 2013). Decryption is the reverse process of encryption, which means to convert cipher text back to plain text at the other end. Symmetric key cryptography uses a single secret key and this key is known by the message sender and the recipient. On the other hand, asymmetric key cryptography uses two different keys: one for the data encryption, known as the public key and the other for decryption, known as private key. The public key is generally known to everyone while the private key is used for decryption and it is only known by the message receiver. Asymmetric key cryptography solved the problem of key management encountered in symmetric key cryptography (William and Stalling, 2014). Nowadays, lower bits key of RSA are considered as insecure after a successful attack using an implementation of the General

Number Field Sieve (GNFS) algorithm which factors ‘N’ easily into p and q in RSA.

In this work, RSA based encryption techniques augmented with bit stuffing algorithm is proposed for securing data transmission in order to increase the complexity and security of keys with short bit lengths and at the same time produce the same security level as with keys with longer bits. The ‘bit-stuffing’ bits introduced in our algorithm totally increase the randomness of encrypted message since attacker is left confused as to how to separate the extra bits from original message.

RELATED WORKS

Several research works have been carried out in securing data transmission. Nag and Jain (2014) introduced the implementation of modified RSA in MATLAB (MRSA) by addition of appended bits per block after encryption with RSA algorithm. This appended bit is removed before decryption with private key; these appended bits boost the security of RSA and also provide good option against increasing the number of bits but lack adequate randomness against timing attacks. Balkee *et al*., (2014) focused on time complexity in key generation of RSA which involved selection of large prime. The work used the Fermat Little theorem for this purpose and for the test of primality. This approach reduces the time complexity of the key generation process so as to speed up the RSA encryption. Jindal and Singh (2014) developed Modified Ron Divest Code4 (MRC4) and study its performance. It was observed that MRC4 has fast speed of execution, which sadly, also allowed RSA security to be compromised quickly. Choudhary and Praveen (2014) developed an enhanced RSA based on three prime numbers, generated new variable apart from’ n’ for encryption and decryption which make the encrypted message unreadable for hackers. The algorithm required more computational time. Jindal and Gupta (2013) introduced modification of RSA through positive justification of random component by addition of extra alphabet to the plaintext before encryption with RSA algorithm. These appended bits boost the security of the message, but the introduction of positive justification in the work made access to the encrypted message cumbersome, even after having access to the private key.

- Alan, and Abdallah, (2012) introduced RSA algorithm using additional third prime number in composition of private and public key. This additional third prime in composition of both public and private key secures the system against brute force attack in RSA. The limitation of their research is that the complexity also increases as a result of the additional third prime number.

MATERIALS AND METHODS

The encryption algorithm enables the user A to send encrypted data to the intended recipient B by using public key generated from RSA algorithm to encrypt the message and later bit-stuffed the encrypted data E(M) (in binary form) before sending it through the communication network. The procedure for this operation is depicted in figure 1.

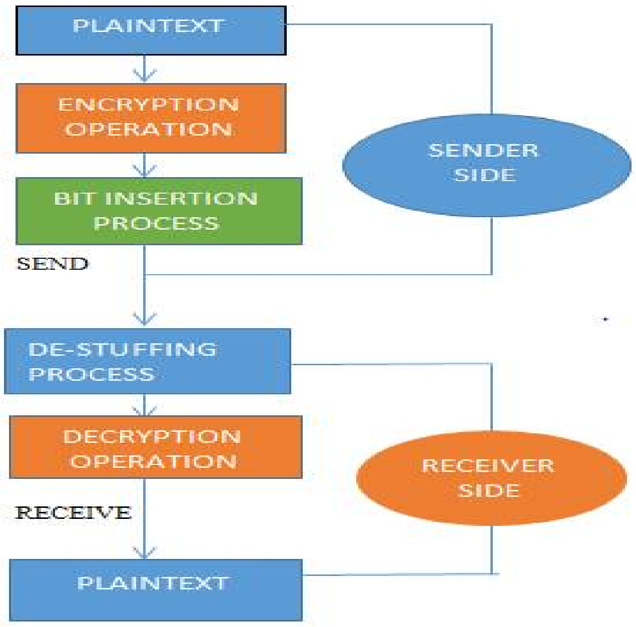

Figure 1: Bit-Stuffed RSA Encryption Procedure

Encryption operation

The first phase of any encryption operation involves the generation of public and private keys by the message recipient B. In this proposed RSA, the intended recipient B performs the following mathematical steps to calculate both public and private key with the help of Fermat little theorem. This process is depicted in the key generation algorithm1.

1. $Choose\ a\ large\ prime\ number\ r, s\ and\ t \quad r \neq s \neq t$.
2. $N = r \times s \times t$
3. $\varphi(N) = \varphi(rst) = \varphi(r) \times \varphi(s) \times \varphi(t)$
4. $\varphi(N) = (r-1)(s-1)(t-1)$
5. $Then\ we\ Choose\ the\ value\ of\ e,$

   $which\ is\ not\ factor\ of\ \varphi(N)$

   $i.e. Select\ public\ key\ (e) \ \exists$

   $1 < e < \varphi(N) and\ the\ gc\ d\ (e, \varphi) = 1$
6. $Find\ the\ value\ of\ private\ key\ (d)$

   $\exists\ e * d = 1\ mod \varphi(N)$
7. $Publish\ encryption\ Key = (e, N)$
8. $Keep\ secretly\ decryption\ key = (d, r, s, t)$

Algorithm 1: Key Generation

The private key (d) is kept secretly while the public key (e) is published on public database for easy access to any message sender A to establish a secure communication to recipient B.

Key Encryption Process

The encryption phase takes three input parameters i.e. public key (**e**), message **M** and modulus size **N**. Encryption of messages will be done at the sender side through the steps below.

(i) Sender A obtains public key 'e' from public database.

(ii) Computes $E(M) = M^e \bmod N$ *(1)*

Bit Stuffing

The output result of equation (1) will then undergo a process called bit stuffing techniques. Bit stuffing techniques is the addition of non-information bits into data. The inserted bits should not be confused with the overhead bits (Kelvin and Richard, 2012). Bit insertion is suggested as a logical means to improve the security of lower bit length of RSA by the addition of non- information bits at a reduced computational time when compared with standard RSA. The result , E (M) obtained in equation (1) is firstly converted to binary format and later scan for three conditional patterns 110,10 and 0 pattern to be bit stuffed using random binary strings 'S' of arbitrary length before it is sent out over the network. The function of these appended binary bits is to introduce random delay or noise to E (M). This increases the security provided by keys with short bit lengths and difficulty for attackers. The bit stuffing process is given in algorithm 2.

Input: encrypted message E (M) in binary format

Output: bit stuffed encrypted message BE (M)

```
Process

Step 1: I $\leftarrow$ 110, J $\leftarrow$ 10 and K $\leftarrow$ 0
Step 2: scan E (M)
Step 3: Randomly generate binary string S of arbitrary length

Step 4: for E (M) $\leftarrow$ 1 to N
Step 5: if string I $\subset$ in E (M) do
Step 6: insert random 'S' next to each string I in E (M)
Step 7: while E (M) still contain string 'I' do
Step 8: reuse binary random string 'S' and go to step 19
Step 9: else
Step 10: if string 'J' $\subset$ in E (M) do
Step 11: insert random 'S' next to each string 'J' in E (M)
Step 12: while E (M) still contain string 'J' do
Step 13: reuse random string 'S' and go to step 19
Step 14: else
Step 15: if string 'K' $\subset$ in E (M) do
Step 16: insert random ‘S’ next to each string ‘K’ in E (M)
Step 17: while E (M) still contain string K do
Step 18: reuse random string S and go to step 19
Step 19: end while
Step 20: end if
Step 21: end for
Step 22: return BE (M)
Step 23: end process
```

Algorithm 2: Bit-insertion of Encrypted Message E (M)

Receiving Phase

This phase firstly removed the inserted bits and then performs decryption which represents the inverse of both bit-insertion and encryption processes respectively. The recipient B receives bit-inserted encrypted message BE (M) from the sender A, removes these extra bits and then decrypt the message using its secret private key. Encrypting and decrypting the same message severally produces different encryption and decryption time respectively. The removal process of the bit-stuffed non information data is detailed in algorithm 3.

Input: Bit stuffed encrypted message BE (M)
Output: encrypted message E (M)

Process

```
Step 1: I $\leftarrow$ 110, J $\leftarrow$ 10 and K $\leftarrow$ 0
Step 2: scan BE (M)
Step 3: for BE (M) $\leftarrow$ 1 to R
Step 4: if string ‘I’ c in BE (M) do
Step 5: remove random string ‘S’ next to each string ‘I’ in BE (M)
Step 6: stop and go to step 14
Step 7: else
Step 8: if string ‘J’ c in BE (M) do
Step 9: remove random ‘S’ next to each string ‘J’ in BE (M)
Step 10: stop and go to step 14
Step 11: else
Step 12: if string ‘K’ c in BE (M) do
Step 13: remove random string ‘S’ next to each string ‘K’ in BE (M)
Step 14: end if
Step 15: end for
```

Step 16: return E (M)
Step 17: end process

Algorithm 3: De-stuffing process.

The output of the algorithm 3 will later undergo decryption process with private key (d) by computing.

$$Plaintext = E(M)^d ModN) \qquad (2)$$

RESULTS AND DISCUSSIONS

Implementation

The implementation of the developed algorithm was done using Java language. The results of RSA Bit-stuffed (RSAB) were compared with the results of MRC4 and MRSA on the same hardware and software platform.

Analysis Metrics

(a) The Execution time

```
startTime = System.currentTimeMillis ()
BE (M),
endTime = System.currentTimeMillis ()

Encryption time = endTime – startTime    (3)

startTime = System.currentTimeMillis ()
D (BE (M)),
endTime = System.currentTimeMillis ()

Decryption time = endTime – startTime    (4)
```

The execution time is very important for any type of encryption and decryption algorithm. This parameter decided the performance of encryption and decryption technique. Execution time or Encryption time is the time taken for encryption algorithm to produce the cipher text from the plaintext. Decryption time is also the time taken for decryption algorithm to produce plaintext from the cipher text

(b) Time complexity of the algorithm
One of the tools used to define the efficiency of an algorithm is the time used by computer to solve a problem by using particular algorithm when an input is specified. Time complexity is important when an algorithm is implemented, these are the terminology used for algorithm complexity are O (1) = constant complexity, O (n) = linear complexity, O (log n) = logarithm complexity and O (n^b) = complexity of polynomial.
With the analysis of algorithm through Big O notation we measured the performance of algorithm and also determine which algorithm is complex or not.

(c) Avalanche effect: A desirable property of good encryption algorithm is that, a slight variation in plaintext should produce significant change in cipher text. The avalanche effect is directly proportional to the security level of an algorithm.

$$\% \, Avalanche \, effect(plaintext) = \frac{Count \, of \, changed \, bits \, in \, ciphertext}{Total \, bits \, in \, ciphertext} \times 100$$

Result Comparison

Our developed encryption algorithm, RSAB, using 256-bit key was compared with both MRC4 and MRSA, each of which employed a key length of 1024 bits. This comparison was conducted on message sized in bytes for the three algorithms as shown in tables I-VII.

Table 1: The Encryption and Decryption time in Milliseconds of RSAB

Order	Size of message (bytes) (x)	Encryption time (milli seconds) (y)	Decryption time (milliseconds) (z)
1.	15	4.07	46.56
2.	18	4.19	43.22
3.	37	2.73	26.57
4.	41	3.89	18.63
5.	42	2.76	34.37
6.	44	3.81	17.96
7.	45	3.75	19.29
8.	48	3.46	40.13
9.	52	3.74	17.90
10.	57	3.62	31.03
11.	61	3.57	18.00

The encryption and decryption time of RSAB depends not only the size of the messages but also majorly on the binay patterns *110*, *10* and *0* the encrypted message E (M) contains. If the message M, in binary format, contains large number of these binary patterns, then the more the insertions of extra bits into M. This produces variations in encryption and decryption times. The complexity of the RSAB encryption algorithm is derived from table I using *polynomial curve equation* as follows:

$$Y = F(X) = ax^6 + bx^5 + cx^4 + dx^3 + ex^2 + fx + g$$

$$Order \; 1 \; equation \; becomes \;\; 4.07 = a(15)^6 + b(15)^5 + c(15)^4 + d(15)^3 + e(15)^2 + f(15)^1 + g$$

$$Order \;\; 2, \quad 4.19 = a(18)^6 + b(18)^5 + c(18)^4 + d(18)^3 + e(18)^2 + f(18) + g$$

$$Order \; 3, \quad 2.73 = a(37)^6 + b(37)^5 + c(37)^4 + d(37)^3 + e(37)^2 + f(37) + g$$

$$Order\ 5, \quad 2.76 = a(42)^6 + b(42)^5 + c(42)^4 + d(42)^3 + e(42)^2 + f(42) + g$$

$$Order\ 8, \quad 3.46 = a(48)^6 + b(48)^5 + c(48)^4 + d(48)^3 + e(48)^2 + f(48) + g$$

$$Order\ 10, \quad 3.62 = a(57)^6 + b(57)^5 + c(57)^4 + d(57)^3 + e(57)^2 + f(57) + g$$

$$Order\ 11, \quad 3.57 = a(61)^6 + b(61)^5 + c(61)^4 + d(61)^3 + e(61)^2 + f(61) + g$$

Using Crammer's online software to calculate the values of a,b,c.d,e,f and g,
$a = 4.8E\text{-}8$, $b = -1.1E\text{-}5$, $c = 1E\text{-}3$, $d = 4.6E\text{-}2$, $e = 1.1$, $f = -13$ and $g = 64.0$
yielding $y = 4.8E\text{-}8x^6 - 1.1E\text{-}5x^5 + 1E\text{-}3x^4 - 4.6E\text{-}2x^3 + 1.1x^2 - 13x + 64$ *(6)*

This equation (6) is the encryption model of RSAB algorithm, x is the size of the message in bytes and y is the encryption time in milliseconds.
Similarly, using the *same polynomial curve equation* to determine order of complexity of decryption algorithm of RSAB from table I, yields

$$Z = F(X) = ax^6 + bx^5 + cx^4 + dx^3 + ex^2 + fx + g$$

$$Order\ 1\ equation\ becomes\ 46.56 = a(15)^6 + b(15)^5 + c(15)^4 + d(15)^3 + e(15)^2 + f(15)^1 + g$$

$$Order\ 2, \quad 43.33 = a(18)^6 + b(18)^5 + c(18)^4 + d(18)^3 + e(18)^2 + f(18) + g$$

$$Order\ 3, \quad 26.56 = a(37)^6 + b(37)^5 + c(37)^4 + d(37)^3 + e(37)^2 + f(37) + g$$

$$Order\ 5, \quad 34.37 = a(42)^6 + b(42)^5 + c(42)^4 + d(42)^3 + e(42)^2 + f(42) + g$$

$$Order\ 8, \quad 40.13 = a(48)^6 + b(48)^5 + c(48)^4 + d(48)^3 + e(48)^2 + f(48) + g$$

$$Order\ 10, \quad 31.03 = a(57)^6 + b(57)^5 + c(57)^4 + d(57)^3 + e(57)^2 + f(57) + g$$

$$Order\ 11, \quad 18.00 = a(61)^6 + b(61)^5 + c(61)^4 + d(61)^3 + e(61)^2 + f(61) + g$$

Using Crammer's online Software to calculate the value of a,b,c.d,e,f and g
$A = -1.35E\text{-}7$, $b = 3.6E\text{-}5$, $c = -3.9E\text{-}3$, $d = 2.17E\text{-}1$, $e = -6.1$, $f = 82.0$ and $g = -372$ and the equation *becomes* $Z = -1.35E\text{-}7x^6 + 3.6E\text{-}5x^5 - 3.9E\text{-}3x^4 + 2.17E\text{-}1.x^3 - 6.1x^2 + 82x - 372$ *(7)*

Equation (7) is the decryption model of RSAB algorithm, x is the size of the message in bytes and Z is the decryption time in milliseconds

The RSAB was analyzed by measuring the complexity through the Big-O test with the series of test inputs (bytes). The results of RSAB encryption and decryption were as described in Table II and III respectively.

Table 2: Test the degree of X for Big (O) Encryption

		y=4.8E-8x6-1.1E-5x5+1E-3x4-4.6E-2x3+1.1x2-13x+64				
		X				
Degree	coefficient	37	44	45	52	61
6	4.8E-8	123.1	348.3	398.6	949.0	2473.0
5	-1.1E-5	-762.8	-1814	-2039	-4182	9291
4	1E-3	1874	3748	4101	7312	13845
3	-4.6E-2	-2330	-3918	-4192	-6468	-104411
2	1.1	1505	2129	2227	3327	4093
1	-13	-481	-572	-585	-676	-793
0	64.0	64.0	64.0	64.0	64.0	64.0

Table 3: Test the degree of X for Big (O) Decryption

		y= -1.35E-7x6+ 3.6E-5x5 -3.9E-3x4 +2.17E-1.x3-6.1x2+82x-372				
		X				
Degree	coefficient	37	44	45	52	61
6	-1.35E-7	-346.4	-976.6	-1121	-2669	-6955
5	3.6E-5	2496.4	5937.0	6643	13687	30405
4	-3.9E-3	-7309	-14617	-15992	-28515	-53998
3	2.17E-1	10991.7	18484.5	19774.1	30511.9	49254.8
2	-6.1	-8351	-11809	-123525	-16494	-22698
1	82.0	3052	3630	3713	4290	5032
0	-372	-372	-372	-372	-372	-372

The result shows that the values of X appear highest in the range of degree 4 for Big O encryption and highest in the range of degree 3 for Big O decryption as highlighted in tables II and III). The Big O notation for RSAB encryption is O (n^4) and for the decryption is O (n^3) which indicated the complexity and security level of RSA.

The encryption and decryption time of MRC4 is similarly derived from the data in table IV using polynomial curve equation.

Table 4: Encryption and Decryption time of MRC4

Order	Size of message (bytes)(x)	Encryption time (milli seconds) (y)	Decryption time (milliseconds) (z)
1.	15	0.0188	0.0217
2.	18	0.0404	0.0238
3.	37	0.0385	0.0488
4.	41	0.0370	0.0597
5.	42	0.0400	0.0529
6.	44	0.0410	0.0541
7.	45	0.0480	0.5460
8.	48	0.0597	0.0556
9.	52	0.0681	0.0609
10.	57	0.0780	0.0576
11.	61	0.0789	0.0590

$$Y = F(X) = ax^6 + bx^5 + cx^4 + dx^3 + ex^2 + fx + g$$

$Order\ 1\ equation\ becomes\ \ 0.0188 = a(15)^6 + b(15)^5 + c(15)^4 + d(15)^3 + e(15)^2 + f(15)^1 + g$

$Order\ \ 2, \quad 0.0404 = a(18)^6 + b(18)^5 + c(18)^4 + d(18)^3 + e(18)^2 + f(18) + g$

$Order\ 3, \quad 0.0385 = a(37)^6 + b(37)^5 + c(37)^4 + d(37)^3 + e(37)^2 + f(37) + g$

$Order\ 5, \quad 0.0404 = a(42)^6 + b(42)^5 + c(42)^4 + d(42)^3 + e(42)^2 + f(42) + g$

$Order\ 8, \quad 0.0481 = a(48)^6 + b(48)^5 + c(48)^4 + d(48)^3 + e(48)^2 + f(48) + g$

$Order\ 10, \quad 0.0780 = a(57)^6 + b(57)^5 + c(57)^4 + d(57)^3 + e(57)^2 + f(57) + g$

$Order\ 11, \quad 0.0789 = a(61)^6 + b(61)^5 + c(61)^4 + d(61)^3 + e(61)^2 + f(61) + g$

$Using\ Crammer's\ online\ Software\ to\ calculate\ the\ value\ of\ a, b, c.d, e, f\ and\ g$

a =-9.59E-10, b = +2.221E-7, c = -2.0806E-5, d = 1E-3, e = 2.64E-2, f = +3.5E-1 and g = -1.870 and the equation becomes

$y = -9.59E\text{-}10x^6 + 2.221E\text{-}7x^5 - 2.0806E\text{-}5x^4 + 1E\text{-}3x^3 - 2.64E\text{-}2x^2 + 3.5E\text{-}1x - 1.870$ (8)

Equation (8) represents the encryption model of MRC4 where x is the size of the message in bytes and y is the encryption time in milliseconds of MRC4.

The complexity of decryption algorithm of MRC4 from X and Z in table V using polynomial curve equation is also given below.

$$Z = F(X) = ax^6 + bx^5 + cx^4 + dx^3 + ex^2 + fx + g$$

$Order\ 1\ equation\ becomes\ \ 0.02738 = a(15)^6 + b(15)^5 + c(15)^4 + d(15)^3 + e(15)^2 + f(15)^1 + g$

$Order\ 3,\quad 0.05900 = a(37)^6 + b(37)^5 + c(37)^4 + d(37)^3 + e(37)^2 + f(37) + g$

$Order\ 5,\quad 0.06231 = a(42)^6 + b(42)^5 + c(42)^4 + d(42)^3 + e(42)^2 + f(42) + g$

$Order\ 8,\quad 0.05801 = a(48)^6 + b(48)^5 + c(48)^4 + d(48)^3 + e(48)^2 + f(48) + g$

$Order\ 10,\quad 0.0630 = a(57)^6 + b(57)^5 + c(57)^4 + d(57)^3 + e(57)^2 + f(57) + g$

$Order\ 11,\quad 0.0650 = a(61)^6 + b(61)^5 + c(61)^4 + d(61)^3 + e(61)^2 + f(61) + g$

$Using\ Crammer's\ online\ Software\ to\ calculate\ the\ value\ of\ a, b, c.d, e, f\ and\ g$

a =-2.065E-12, b = 1.440E-10, c = 5.452E-8, d = -8.145E-6, e = 4.048E-4, f = -7.045E-3 and g = 0.0610 and the equation becomes

$$Z = -2.065E\text{-}12x^6 + 1.440E\text{-}10x^5 + 5.452E\text{-}8x^4 - 8.145E\text{-}6x^3 + 4.048E\text{-}4x^2 - 7.045E\text{-}3x + 0.0610 \qquad (9)$$

Equation (9) represents decryption model of MRC4, x is the size of the message in bytes and Z is the decryption time in milliseconds

Table 5: Test the degree of X for Big (O) Encryption of MRC4

		y=-9.59E-10x6+2.22E-7x5-2.081E-5x4+1E-3x3-2.64E-2x2+3.5E-1x-1.8707				
				X		
Degree	coefficient	37	44	45	52	61
6	-9.59E-10	-2.460	-6.95	-7.96	-18.96	-49.41
5	+2.22E-7	15.39	36.61	40.97	84.41	114.38
4	-2.081E-5	-38.98	-77.96	-85.29	-152.08	-287.99
3	+1E-3	50.95	85.18	91.13	140.61	226.98
2	-2.6E-2	-35.59	-50.33	-50.63	70.30	96.75
1	3.5E-1	12.95	15.4	15.75	18.2	21.35
0	-1.8707	-1.8707	-1.8707	-1.8707	-1.8707	-1.8707

The MRC4 was also analyzed by measuring the complexity through the Big-O test with the series of test inputs (bytes) under the same parameter settings with MRSA. The results for MRC4 encryption and decryption were shown in Table V and VI respectively. The values of X appear highest in the range of degree 2 for Big O encryption and highest in the range of degree 2 for Big O decryption. This is values were highlighted in table V and VI. The Big O notation for MRC4 encryption is O (n^3) and the Big O notation for MRC4 encryption is O (n^2) which indicated the complexity and security level of MRC4.

Table 6: Test the degree of X for Big (O) Decryption of MRC4

		Z=-2.065E-12x6+1.440E-10x5+5.452E-8x4-8.145E-6x3+4.048E-4x2-7.045E-3x+0.0610				
				X		
Degree	coefficient	37	44	45	52	61
6	-2.065E-12	-0.00529	-0.01498	-0.01715	0.0408	0.1064
5	1.440E-10	0.0099	0.0237	0.0266	0.0547	0.1216
4	5.452E-8	0.1022	0.2044	0.2236	0.3987	0.7549
3	-8.145E-6	-0.4126	-0.6939	-0.7423	-1.1453	-1.8489
2	4.048E-4	0.5542	0.7838	0.8148	1.0947	1.5064
1	-7.045E-3	-0.260	-0.3099	-0.3170	-0.3663	-0.4297
0	0.0610	0.0610	0.0610	0.0610	0.0610	0.0610

The derivation of equation connecting the X and Y in table VII using polynomial curve to determine order of complexity of the encryption algorithm of MRSA resulted into equation (10).

Table 7: Encryption and Decryption time of MRSA

Order	Size of message (bytes)(x)	Encryption time (milli seconds) (y)	Decryption time (milliseconds) (z)
1.	15	1.418	5.560
2.	18	2.665	8.805
3.	37	3.001	11.064
4.	41	3.3915	11.932
5.	42	3.4012	12.015
6.	44	3.6502	12.357
7.	45	3.7711	12.567
8.	48	3.9550	13.001
9.	52	4.0250	13.832
10.	57	4.512	14.852
11.	61	4.8215	15.067

$$Y = F(X) = ax^6 + bx^5 + cx^4 + dx^3 + ex^2 + fx + g$$

$$\text{Order 1 equation becomes } 1.418 = a(15)^6 + b(15)^5 + c(15)^4 + d(15)^3 + e(15)^2 + f(15)^1 + g$$

$$\text{Order 2,} \quad 2.665 = a(18)^6 + b(18)^5 + c(18)^4 + d(18)^3 + e(18)^2 + f(18) + g$$

$$\text{Order 3,} \quad 3.001 = a(37)^6 + b(37)^5 + c(37)^4 + d(37)^3 + e(37)^2 + f(37) + g$$

$$\text{Order 5,} \quad 3.4012 = a(42)^6 + b(42)^5 + c(42)^4 + d(42)^3 + e(42)^2 + f(42) + g$$

$$\text{Order 8,} \quad 3.9550 = a(48)^6 + b(48)^5 + c(48)^4 + d(48)^3 + e(48)^2 + f(48) + g$$

$$\text{Order 10,} \quad 4.5120 = a(57)^6 + b(57)^5 + c(57)^4 + d(57)^3 + e(57)^2 + f(57) + g$$

$$\text{Order 11,} \quad 4.8215 = a(61)^6 + b(61)^5 + c(61)^4 + d(61)^3 + e(61)^2 + f(61) + g$$

Using Crammer's online Software to calculate the value of a, b, c.d, e, f and g

A = -2.858E-9, b = +1.1415E-6, c =1.62E-4, d = 1.05E-2, e = 3.8E-1, f = +6.674 and g = -41.49 and the equation becomes

$$Z = -2.85E\text{-}9x^6+1.1415E\text{-}6x^5+1.62E\text{-}4x^4+1.05E\text{-}2x^3+3.8E\text{-}1x^2+6.674x-41.49 \qquad (10)$$

The equation (10) represents encryption model of MRSA, where x is the size of the message in bytes and y is the encryption time in milliseconds of MRSA.

The derivation of equation connecting X and Z in Table VII using the same polynomial curve to determine the order of complexity of decryption algorithm of MRSA resulted into equation (11).

$$Z = F(X) = ax^6 + bx^5 + cx^4 + dx^3 + ex^2 + fx + g$$

$$\text{Order 1 equation becomes } 5.560 = a(15)^6 + b(15)^5 + c(15)^4 + d(15)^3 + e(15)^2 + f(15)^1 + g$$

$$\text{Order 2,} \quad 8.805 = a(18)^6 + b(18)^5 + c(18)^4 + d(18)^3 + e(18)^2 + f(18) + g$$

$$\text{Order 3,} \quad 11.064 = a(37)^6 + b(37)^5 + c(37)^4 + d(37)^3 + e(37)^2 + f(37) + g$$

$$\text{Order 5,} \quad 12.015 = a(42)^6 + b(42)^5 + c(42)^4 + d(42)^3 + e(42)^2 + f(42) + g$$

$$\text{Order 8,} \quad 12.357 = a(48)^6 + b(48)^5 + c(48)^4 + d(48)^3 + e(48)^2 + f(48) + g$$

$$\text{Order 10,} \quad 14.852 = a(57)^6 + b(57)^5 + c(57)^4 + d(57)^3 + e(57)^2 + f(57) + g$$

$$\text{Order 11,} \quad 15.067 = a(61)^6 + b(61)^5 + c(61)^4 + d(61)^3 + e(61)^2 + f(61) + g$$

Using Crammer's online Software to calculate the value of a, b, c.d, e, f and g

a =-6.931E-8, b = 1.68E-5, c = -1.6459E-3, d = 8.427E-2, e =-2.33207, f = 33.18 and g =-180.07 and the equation becomes

$$Z = -6.931E\text{-}8x^6+1.68E\text{-}5x^5-1.6459E - 3x^4+ 8.427E\text{-}2x^3- 2.33207x^2+ 33.18x- 180.07 \qquad (11)$$

Equation (11) represents decryption model of MRSA, where x is the size of the message in bytes and Z is the decryption time in milliseconds.

The complexity of MRSA was also analyzed with the Big-O test using the series of test inputs (bytes) under the same parameter settings with RSAB and MRC4. The encryption and decryption models in equations (10) and (11) respectively show that the values of x appear highest in the range of degree 3 for Big O encryption and highest in the range of degree 3 for Big O decryption. The Big O notation for MRSA encryption is O (n^3) and the Big O notation for MRSA encryption is O (n^3), which indicated the complexity and security level of MRSA.

Avalanche Effect Analysis

The output of avalanche effect by flipping one bit in plaintext of RSAB, MRC4 and MRSA generated 45.82%, 21.31% and 9.80% change in cipher text respectively. Figure 5 shows the graph of avalanche effect in (%) of the three algorithms RSAB, MRC4 and MRSA. The higher the value of avalanche effect of the encryption algorithm, the greater the randomness and the better the security of such encryption algorithm against different attacks.

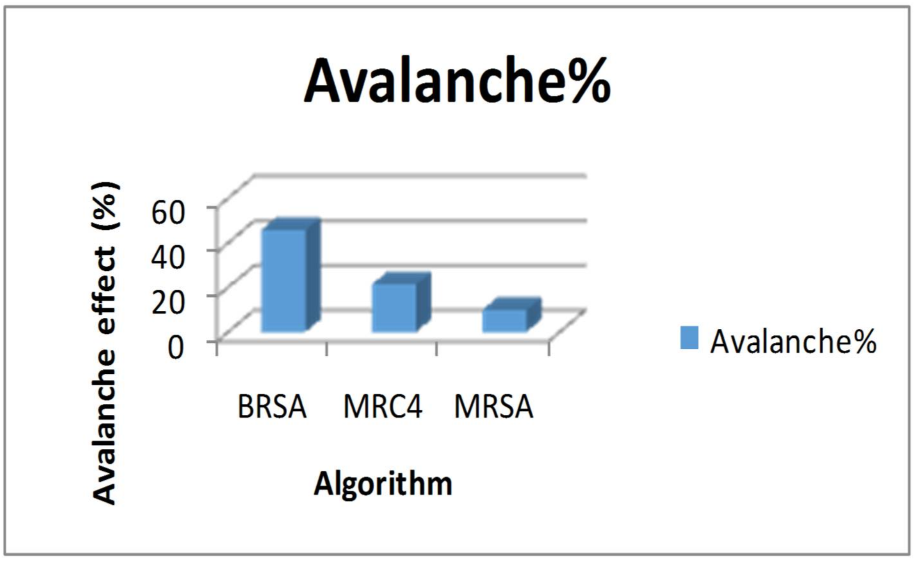

Figure 2: Effect of a bit variation of plaintext on Cipher text

Result Comparison Analysis

The result evaluation of RSAB, MRC4 and MRSA is shown in Table VIII. This consists of order of complexity of encryption (C_E), order of complexity of decryption (C_D) and avalanche effects (A_E).

Table 8: Comparisons of RSAMRC4 and MRC4B,

Encryption Algorithm	CE	CD	AE
RSAB	O(n2)	O(n2)	45.82%
MRC4	O(n3)	O(n2)	21.31%
MRSA	O(n3)	O(n3)	9.80%

The results in table VIII show that RSAB has avalanche effect value of 45.82% while MRC4 and MRSA have 21.31% and 9.80% respectively. The higher avalanche effect value in RSAB than in both MRC4 and MRSA confirms better secure level.

CONCLUSION

The RSA based encryption with bit insertion technique (RSAB) provided in this work guarantees adequate security mechanism for data in transmission and in storage. The algorithm performance analysis was conducted between proposed RSAB with key size of 256 bits, MRC4 (1024-bit key size) and MRSA (also 1024-bit key size). The time complexity factor prove that the order of complexity of proposed RSAB encryption is Big O notation O (n^2) and decryption is O (n^2), and order of complexity of MRC4 is O (n^3) for encryption and O(n^2) for decryption while that of MRSA is O (n^3) for both encryption and decryption which indicates better security performance. The higher the avalanche effects value the better the security level which implies that the result of RSAB is better than that of MRC4 and MRCA.

REFERENCES

Singh, S. 2011. *The Code Book: the science of secrecy from ancient Egypt in quantum cryptography. pp 1-23*

Yogita, G., Praveen, S. 2015**.** *Modified RSA Algorithm used for Cloud Computing for Data Security. International Journal of advanced technology in engineering and science. 3(6): 207-212.*

Pradhan, S., Sharmal, B.K. 2013. *Efficient RSA Cryptosystem with BM Prime Method. International Journal of information and Network Security. 2(1):103-107.*

William, S., Stalling, W. 2014. *Cryptography and network security principle and practice. (4th Edition) Pearson Education India. pp 591-615.*

Nag, A. and Jain, V.K. 2014. *Implementation of modified RSA in MATLAB. International journal of innovatives research and development. 3 (13):382-384.*

Balkees, M. S., Zaiton, M., Sharifah, Y. 2014. *Improve Cloud Computing Security using RSA Encryption with Fermat's Little Theorem. International Organisation of Scientific Research Journal of Engineering. 4 (2):1-8.*

Jindal, P., Singh, B. 2014. *Performance Anal-*

ysis of Modified RC4 Encryption Algorithm. IEEE International Conference on Recent Advances and Innovation in Engineering. Pp 1-5

Choudhary, V., Praveen, N. 2014. RSA Cryptosystem Based on three Prime Numbers. *International Journal of Innovative Research on Computer and Communication Engineering.* Vol. 1(10). pp. 753-757.

Jindal, P., Gupta, V. 2013. *Modification in RSA through positive justification of random component. International Journal of Computer Application. 75(17):12-16.*

Alan, H.H., Abdallah, A. 2012. *Enhanced Method for RSA Cryptosystem Algorithm. International Conference on Advanced Computer Science Applications and Technologies, pp. 402-408.*

Kelvin, R. F., Richard, W. S. 2012. *What is bit-stuffing? Article retrieve on 19th January, 2016. Available at http//en.wilkipedia.org/wiki/Bit-stuffing. Last edited on 26th October, 2015.*

(Manuscript received: 22nd May, 2017; accepted: 29th June, 2017).