Date of publication JUN-30, 2023, date of current version FEB-15, 2023.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.22.2.3094

Hidden Real Modulus RSA Cryptosystem

GETANEH AWULACHEW' ZIMBELE[1], SAMUEL ASFERAW DEMILEW[2]

[1]Department of Information Technology, College of Computing, Debre Berhan University, Debre Berhan, Ethiopia, PO Box: 445, e-mail: get.awulachew@gmail.com

[2]Department of Information Technology, College of Computing, Debre Berhan University, Debre Berhan, Ethiopia, PO Box: 445, e-mail: samasferaw@gmail.com

Corresponding author: Getaneh Awulachew' Zimbele (e-mail: get.awulachew@gmail.com).

ABSTRACT Cryptographic techniques in cyber security can be categorized into symmetric and asymmetric. Among asymmetric cryptographic techniques, the RSA algorithm is more popular and considered as secured. Since, RSA uses the common modulus in both encryption and decryption, this modulus is openly available for the public which makes it exposed for attack. Its security is based on the assumption of large integer factorization problem, but this could leave it open to different cryptanalysis attacks: low private exponent attack, Shor's polynomial-time quantum algorithm, quantum inverse Fourier transform and phase estimation. To address these shortcomings, this paper proposes a public-key security algorithm called Hidden Real Modulus RSA (HRM-RSA) which hides real modulus by masking it. The public mask modulus which is a pseudo random masking number is derived from real modulus. Then, this derived public mask modulus is introduced in a public key component; as a result, a real modulus is kept hidden from the public unlike the case in RSA. Encryption is done using this public mask modulus and the decryption process is done using a private hidden real modulus. For performance analysis Net bean IDE 8.2 is used, and the proposed algorithm is compared with state-of-the-art algorithms: RSA, ESRKGS, and MRSA based on security strength, time complexity, key generation time, encryption speed, and decryption speed. The performance analysis shows that HRM-RSA is less complex but highly secured than existing algorithms. It improves key generation time of ESRKGS, and MRSA by 311%, 42%; encryption time of RSA, ESRKGS, MRSA by 0.7%, 139%, 735%; decryption time of RSA, ESRKGS, MRSA by 3%, 138%, 799%, respectively.

KEYWORDS Asymmetric key; Cryptography; Data security; Hidden real modulus; Masking; RSA.

I. INTRODUCTION

TODAY, secure online communication has become an enormous concern in the untrusted world of the Internet, where there are several effective security attackers. Cyber-attacks occur quickly and unfold across the world in minutes not depending on borders, geography, or national jurisdictions [1]. There are several proposed mechanisms to ensure data security on the Internet, which includes: Intrusion Detection System (IDS), Username, Password, Intrusion Prevention System (IPS), Firewall, Biometric, Proxy, and Cryptography. Cryptography is a cyber-security mechanism which provides data Confidentiality, Integrity, and Authenticity. While traditionally cryptographic algorithms are divided into three categories Keyless, Symmetric key and Asymmetric key, modern cryptographic techniques can be categorized into asymmetric and symmetric key cryptosystems [2], [3], [4], [5].

In symmetric-key the challenge is secure key distribution because of eavesdropping during key sharing; as a result, numerous keys are needed: for n users' n*(n-1)/2 keys required, whereas in asymmetric key the challenge is on the need for third party, i.e., Certificate Authority(CA) [2], [3], [4], [5].

RSA cryptography, which is one of the most commonly used asymmetric cryptographic techniques today, was developed by Rivest, Shamir, and Adelman in 1978. RSA cryptography is based on the generation of two large-random-prime numbers p, and q of equal bit-size and the generation of random exponents d and e satisfying Euler's function as described in Equation (1) [4], [6], [7], [8].

$$d \equiv e^{-1} mod\ \phi(n); \forall (e, d, n) \in Z+ \neq \{0,1\}, \quad n = p*q \therefore \phi(n) = (p-1)*(q-1). \tag{1}$$

Although RSA is considered as a popular and secured public key cryptography technique, it could be open to different security attacks because it uses a common real modulus during the encryption-decryption process. To fill this gap, this paper proposes a new cryptographic algorithm called Hidden Real Modulus RSA (HRM-RSA) algorithm.

The rest of the paper is organized as follows: Section 2 discusses related work methods, contributions, and gaps. The proposed algorithm is introduced in Section 3. Section 4 presents the mathematical proof of the proposed algorithm. Section 5 presents a performance analysis of HRM-RSA with respect to existing work. Finally, Section 6 presents a conclusion and future work.

II. RELATED WORK

Rivest et al. (1978) proposed a novel asymmetric cryptosystem method called RSA for protecting the confidentiality of data. It is the first algorithm used for both digital signature and data encryption [4], [9]. It uses two large prime numbers p and q to generate private and public key pairs. As depicted in Algorithm 1, in the RSA cryptosystem there are three main procedures, namely: key generation, encryption, and decryption processes. The decryption key exponent is different from the encryption key exponent but has a mathematical relationship [2], [3].

The major drawback of RSA algorithm is that its security is based on the assumption of the difficulty of integer factorization which does not work in massive parallel computational quantum computers: D-Wave quantum computer attacks and Shor's quantum polynomial-time algorithms based on the quantum inverse Fourier transform, and phase estimation [10], [11], [12], [13]. The reason for the possibility of RSA factorization is that its common real modulus n property is constant, i.e., it is a product of two prime numbers. Other attacks on RSA include Wiener's continued fraction attack, lattice reduction and Coppersmith's method, weak public and private exponent attacks, large private exponent attack, combined attack by sat-approach, advanced timing attack, ion fault injection attack, common modulus attack, blind signature attack, and double encryption attack [2], [3], [5], [10], [14], [15], [16], [17], [18], [19], [20], [21].

Algorithm 1: RSA Algorithm [9]

RSA_Key_Generation ()
INPUT: Prime integers' p and q.
OUTPUT: Find public key exponent (e), private key exponent (d), and common modulus (n).
Begin
Procedure (p, q, e, d, and n)
1. Generate two random distinct prime integers' p and q
2. Calculate $n \leftarrow p * q$
3. Calculate Euler $\emptyset(n) \leftarrow (p-1) * (q-1)$
4. Generate random public key exponent e *such that*, $gcd(e, \emptyset(n))=1$, $1<e<\emptyset(n)$
5. Calculate private key exponent d, such that, $d \leftarrow e^{-1} mod \emptyset(n)$

End Procedure
End
RSA_Encryption ()
Input: Select plain text ($\mathcal{T}$), public key exponent (e), and common modulus (n).
Output: Find cipher text (C).
Begin
Procedure ($\mathcal{T}$, e, n, and C)
$C \leftarrow (\mathcal{T})^e \ mod \ n$
End Procedure
End
RSA_Decryption ()
Input: Select cipher (C), private key exponent (d), and common modulus (n).
Output: Find plain text ($\mathcal{T}$).
Begin
Procedure ($\mathcal{T}$, d, n, and C)
$\mathcal{T} \leftarrow C^d \ mod \ n$
End Procedure
End

J. Jaiswal et al. (2014) proposed an algorithm called "Reformed RSA algorithm based on Prime Number" to secure data communication over the network and to increase speed performance of the RSA algorithm [22]. This method uses the common modulus n which is a multiplication result of four prime numbers p, q, r, and s, and offline storage method. As a result, algorithm speed is increased through offline storage of public key in a database which is identical in all networks without any improvement on the security of standard RSA. The limitation of this algorithm, like standard RSA, is that its encryption and decryption keys are dependent on the common modulus; therefore, it can be easily unlocked. Other drawbacks include a distributed database (DDB) update time and DDB attack [22].

M. Thangavel et al. (2015) proposed another enhanced method called "An Enhanced and Secured RSA Key Generation Scheme (ESRKGS)" based on four randomly generated prime numbers to increase the time required to factorize these primes [23]. The computation of public key and private key exponents depends on the value of n, which is the product of four prime numbers. It enhanced the security of RSA by reducing direct attack using larger exponents. The limitation of this approach includes encryption, and decryption time is higher than the original RSA, and most attacks on RSA can be applicable to this algorithm too [23]. Erkam Lüy et al. (2016) showed that ESRKGS has a similar security level as traditional RSA [24], [25].

S. Mathur et al. (2017) proposed another enhanced method called "Analysis and Design of Enhanced RSA Algorithm to Improve the Security". It uses four prime numbers and multiple public keys with the k-nearest neighbor algorithm [25]. The limitations of this approach are the following ones: it has higher key generation, encryption, and decryption time than original RSA as it encrypts and decrypts character by character and incorporates a looping process; it is compatible only for text files and special characters like @, #. $, %, &, and *.

Panda & Chattopadhyay (2017) proposed a method called "Hybrid security algorithm for RSA cryptosystem based on four random prime numbers", which is based on the ESRKGS algorithm and uses the random modulus [7]. The limitations of this algorithm are the following ones: most of the time the correct random modulus number w may not be found; it has high key generation time (due to exponentiation and modulation operations which do not add any security feature to the system); a double encryption attack (see Section 5, Subsection 5.5) and generating alternative private key exponents is possible to the system since it uses common modules.

M.A. Islam et al. (2018) proposed a modified method called a "Modified and Secured RSA Cryptosystem based on n prime numbers (MRSA)" which improves the security of the standard RSA algorithm by using four distinct prime numbers and two different encryption-decryption key pairs as shown in

Algorithm 2 [6]. In this cryptosystem, modulus n is the product of p, q, r, and s. To produce key pairs, public keys e and f have been selected randomly and private key exponents d and g are the multiplicative inverse of each public key exponents in modulo n. Key exponents are dependent on common modulus n. Since the process of encryption and decryption depends on common modulus n, it is easy to unlock the system. The limitation of this approach includes high encryption and decryption time when compared to other related work; factorization of common modulus n to unlock the system is not difficult; the cipher text size, which is a major concern in data transmission, is doubled when compared with RSA.

Algorithm 2: MRSA Algorithm [6]

MRSA_Key_Generation ()
INPUT: Prime integers' $p, q, r,$ and s.
OUTPUT: Find public key exponents ($e,$ and f), Private key exponents ($d,$ and g), and Common Modulus (n).
Begin
Procedure ($p, q, r, s, e, f, d, g,$ and n)
1. Generate four random unique prime integers' p, q, r, and s
2. Calculate $n \leftarrow p * q * r * s$
3. Calculate Euler's $\emptyset (n) \leftarrow (p-1) * (q-1) * (r-1) * (s-1)$
4. Generate distinct public key exponents e and f such that, $gcd(e, \emptyset (n)) = 1, 1 < e < \emptyset (n)$ and $gcd(f, \emptyset (n)) = 1, 1 < f < \emptyset (n)$
5. Calculate private key exponents d and g such that, $d \leftarrow e^{-1} \bmod (\emptyset (n))$ and $g \leftarrow f^{-1} \bmod (\emptyset (n))$
End Procedure
End

MRSA_Encryption ()
Input: Select Plain text (T), public key exponents (e, f), and common modulus (n).
Output: Find Cipher text C.
Begin
Procedure ($T, e, f, n,$ and C)
$C \leftarrow (T^e \bmod n)^f \bmod n$
End Procedure
End

MRSA_Decryption ()
Input: Select cipher text (C), private key exponents (d, g), and common modulus (n).
Output: Find plain text (T).
Begin
Procedure ($T, d, g, n,$ and C)
$T \leftarrow (C^g \bmod n)^d \bmod n$
End Procedure
End

A review of related work shows that RSA, ESRKGS, and MRSA are more reliable algorithms than other related work. Nevertheless, even these algorithms have security and execution performance drawbacks which could be addressed.

Hence, to address these shortcomings, this work attempts to propose a new asymmetric cryptographic algorithm called Hidden Real Modulus RSA (HRM-RSA) and its simulation result is compared with state-of-the-art related works: ESRKGS, MRSA and with common and popular algorithm RSA.

III. PROPOSED WORK

In this section, we present the proposed Hidden Real Modulus RSA (HRM-RSA) Algorithm.

Generally, all existing related works use a common real modulus for encryption and decryption which makes them unsecured. Their security strength depends on the difficulty of large integer factorization problem which will not be a problem for D-Wave quantum computer attacks, Shor's integer factorization and polynomial-time factorization algorithms [10], [11] and [12], [13]. To avoid these limitations a new security parameter called public mask modulus M which is computed from unpredictable random integer number m and a real modulus n is introduced so as to hide a real modulus n from the public.

Like RSA, the basic steps used in this proposed algorithm are key generation, encryption, and decryption processes with our new modifications in the processes. The sixth and seventh steps in our algorithm key generation process differ from RSA and other related works. Another difference is that a common real modulus n is used for both encryption and decryption in RSA and existing related works, but in HRM-RSA a real modulus n is kept private to be used only for decryption whereas encryption is done with a new parameter called public mask modulus M.

In the key generation process, steps to be computed by the receiver are as follows: First, two large prime numbers p and q are generated randomly. Second, the product of these two large prime numbers p and q generates the real modulus which in this work called hidden real modulus n. Third, Euler's Ø (n) is calculated by multiplying p-1 with q-1. Fourth, the prime encryption exponent (e) will be randomly generated between 1 and Ø (n) in which the Greatest Common Devisor (GCD) of e and Ø (n)) is 1. Fifth, the decryption exponent (d) is computed by calculating inverse of (e) mod Ø (n). Sixth, a large multiplayer number m will be randomly generated. This multiplier m can be any type of integer with any bit size. Seventh, a public mask modulus M is computed by multiplying a real modulus n by a random multiplayer number m to hide the real modulus. This masking process hides the real modulus n from the public. As a result, the real modulus n becomes private, unlike common modulus in RSA. Due to this, in this paper a real modulus n is referred as hidden real modulus n. This method of producing the public mask modulus M from any type of random number m makes it unpredictable. This unpredictable property of the public mask modulus M will challenge cryptanalysts from designing cryptanalysis algorithm to unlock HRM-RSA. Finally, the receiver makes public key components (e, and M) available to the correspondents while private key components (d, n) are kept secret.

During encryption step: First, the sender encrypts the plain text T using public key exponent e and public mask modulus M. The bit length of plain text T should be smaller than the bit length of n and M. Then, the encrypted text C will be sent to the receiver. Since the encryption is not done using a real modulus, this cipher text is a false cipher text. This technique challenges attackers from conducting attacks based on cipher text like brute-force attack.

During the decryption step: First, the receiver receives false cipher text C delivered from the sender and decrypts to genuine cipher C1 by removing a mask using a hidden real modulus n.

Then, the receiver decrypts this genuine cipher text C1 into the original plain text T using private key exponent d and hidden real modulus n.

Key Generation and En/Decryption architecture of the proposed algorithm is basically shown in Algorithm 3. Fig. 1 and Fig. 2 farther illustrate Key Generation and En/Decryption architecture of the proposed algorithm, respectively.

Algorithm 3: HRM-RSA Algorithm

HRM-RSA_Key_Generation ()
INPUT: Prime numbers (p, q), and multiplier number m.
OUTPUT: Find public key exponent (e), private key exponent (d), and modulus numbers (n, $\mathcal{M}$) in bit-length.
Begin
Procedure (p, q, m, e, d, n, and $\mathcal{M}$)
1. Generate two random distinct prime numbers p and q within bit-length/2
2. Compute a real modulus n such that, $n \leftarrow p * q$
3. Calculate Euler $\emptyset(n)$ such that, $\emptyset(n) \leftarrow (p-1)*(q-1)$
4. Randomly generate a prime public key exponent e, such that, $gcd(e, \emptyset(n)) = 1,\ 1 < e < \emptyset(n)$
5. Calculate private key exponent d, such that, $d \leftarrow e^{-1} \bmod \emptyset(n)$
6. Generate a random multiplier number m within any bit-length, such that, $m \leftarrow \mathcal{RNG}(m.\text{length}, rand), m>1;$
7. Compute a public mask modulus $\mathcal{M}$, such that, $\mathcal{M} \leftarrow m * n;$
End Procedure
End

HRM-RSA_Encryption ()
Input: Select plain text ($\mathcal{T}$), public key exponent (e), and public mask modulus ($\mathcal{M}$).
Output: Cipher text (C).
Begin
Procedure ($\mathcal{T}$, e, $\mathcal{M}$, and C)
$C = (\mathcal{T})^{e} \bmod \mathcal{M}$
End Procedure
End

HRM-RSA_Decryption ()
Input: Take False Cipher (C), private key exponent (d) & hidden real modulus (n).
Output: Find plain text ($\mathcal{T}$).
Begin
Procedure (C, d, n, and $\mathcal{T}$)
$C1 = C \bmod n$ *// Since (C mod M) mod n = C mod n where M is the multiple of n, This mod removes the mask and results genuine cipher text C1. This also improves speed performance as modulation is performed before exponentiation.*
$\mathcal{T} = C1^{d} \bmod n$
End Procedure
End

Fig.1 shows a flow chart for key generation steps. As shown in this figure, HRM-RSA algorithm takes two large random prime numbers p and q generated by Random Prime Number Generator function (RPNG), and random multiplier m generated by Random Number Generator function (RNG) as input. Finally, it generates the public key (K_U) = (e, M) and the private key (K_R) = (d, n) as output.

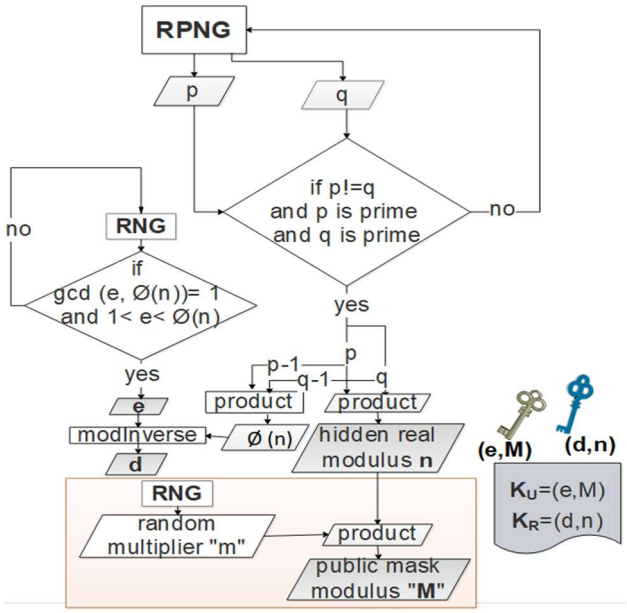


Figure 1. HRM-RSA Key Generation Architecture

Fig.2 shows a flow chart for the encryption, and decryption process of HRM-RSA. This flowchart shows that in the encryption process Alice takes the public key component of Bob which are K_U= (e, M) and her plain text T of which bit length is less than the bit length of n and M as input and produces a false cipher text C to be transmitted to the receiver Bob using encryption algorithm of HRM-RSA. Public mask modulus M is used for masking both genuine cipher text C1 (which was computed as $C1=T^{e} \bmod n$ in RSA) and a hidden real modulus n. When Alice uses a public mask modulus M to encrypt plain text T, a genuine cipher C1 becomes hidden in false cipher C.

As Bob receives false cipher text C, he starts the decryption process using our HRM-RSA decryption algorithm and his private key components K_R= (d, n). In the decryption process, first, he computes genuine cipher C1 using false cipher C received from Alice and his hidden real modulus n. Then, he uses his private key component K_R= (d, n), and genuine cipher text C1 as input and recovers the copy of original plain text T as output.

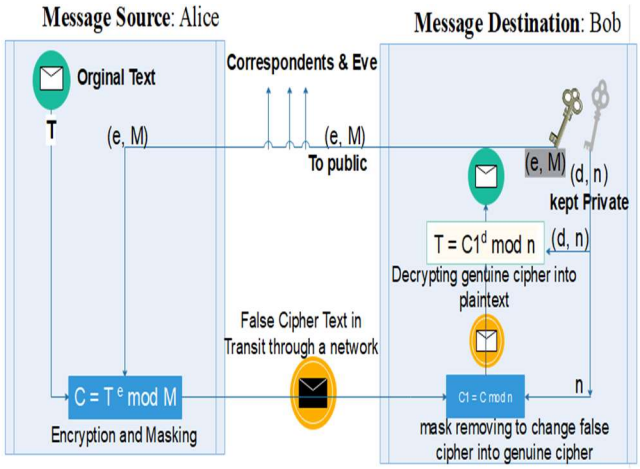


Figure 2. HRM-RSA Enc/Decryption Architecture

Let us discuss a simple example using proposed algorithm.

Key Generation:

Assume, Bob chooses p = 13, q = 11 and calculates hidden real modulus n= 143, Ø (n) = (13-1) (11-1) or 120. Bob chooses positive integer m = 5 and calculates M= n * m = 715.

Now he chooses two exponents which are e and d from Z_{120}*. If he chooses e = 23 then d is 47. Note that e and d are inverses of each other, i.e., e * d mod 120=1.

Encryption:

Now imagine that Alice needs to transmit the plaintext T= 6 to Bob, she uses the public exponent 23 and public mask modulus 715 of Bob to encrypt the plain text 6.

$C=6^{23} \bmod 715 = 789730223053602816 \bmod 175 = 271$

Decryption:

Bob uses the hidden real modulus n = 143 to decrypt a false cipher text C = 271 received from Alice to genuine cipher C1. Then he uses the private exponent d = 47 and hidden real modulus n = 143 to decrypt a genuine cipher text C1 to plain text T.

C1= 271 mod 143=128

$T = 128^{47} \bmod 143 = 6$

IV. MATHEMATICAL PROOF OF HRM-RSA ALGORITHM

In this section, we have proved that the encryption and decryption process of HRM-RSA are inverses of each other using the 2nd version of Euler's theorem.

If $n = p*q$, $a<n$, and if k is an integer, then $a^{k*Ø(n)+1} = a \pmod n$, $Ø(n)$ is the totient function.

Additionally, we have used the congruence of modular properties.

$$a^{kr} \bmod n = (a^{k} \bmod n)^{r} \bmod n$$

Assume that the plain text delivered to Bob from Alice is T1 and prove that it is equal to plaintext T sent by Alice as encrypted cipher text C.

$C = \mathcal{T}^{e} \bmod \mathcal{M}$; where $\mathcal{M}=n*m$, $m \varepsilon Z^{+}$

$\mathcal{T}1=C^{d} \bmod n = (\mathcal{T}^{e} \bmod \mathcal{M})^{d} \bmod n$

$\mathcal{T}1 = ((\mathcal{T}^{e} \bmod \mathcal{M})^{d} \bmod \mathcal{M}) \bmod n$; where $n|\mathcal{M}$ and $n<\mathcal{M}$.

$\mathcal{T}1 = \mathcal{T}^{e\,d} (\bmod \mathcal{M} \bmod n)$ *//congruence of modular properties*

$\mathcal{T}1 = (\mathcal{T}^{e\,d} \bmod n)$ where $n|\mathcal{M}$ *and* $n<\mathcal{M}$ *therefore fields size is* n.

$\mathcal{T}1 = \mathcal{T}^{e\,d} (\bmod n)$

$e * d = k * Ø(n)+1$; *i.e.,* d *and* e *are inverse of each other in modulo* $Ø(n)$

$\mathcal{T}1 = \mathcal{T}^{e\,d} (\bmod n) = \mathcal{T}^{k*Ø(n)+1} \bmod n$

$\mathcal{T}1 = \mathcal{T}^{k*Ø(n)+1} \bmod n = \mathcal{T} \bmod n = \mathcal{P}$ *// Euler's theorem (2nd version)*

$\mathcal{P}1 = \mathcal{P}$

V. IMPLEMENTATION AND PERFORMANCE ANALYSIS

HRM-RSA is implemented using Java program on Net Beans IDE 8.2 programming environment, running on Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz (4 CPUs) and 4 GB RAM. To conduct our experiment, four distinct random prime numbers from each of six different bit sizes: 28-bit, 56-bit, 128-bit, 256-bit, 1024-bit and 2048-bit are randomly generated. Since RSA and HRM-RSA need only two prime numbers as input, two of the four distinct random prime numbers for each of the six different bit sizes are used. On the other hand, ESRKGS and MRSA use four distinct prime numbers as input from each bit size. To simulate the speed performance of the algorithms, we have used six different bit-size and distinct combinations of randomly chosen prime numbers. To make our result analysis more reliable, we have executed the algorithms five times for each input and the average execution time is considered.

A. KEY GENERATION TIME OF ALGORITHMS

Table 1 shows the average key generation time of the algorithms for each bit length. As shown in Table 1, the key generation time cost of HRM-RSA is better than other state-of-the-art algorithms, except RSA.

Table 1. Summary of Key Generation Time (in Seconds)

Bit Length	Algorithms			
	RSA	ESRKGS	MRSA	HRM -RSA
28bit	0168798005	0.17756215	0.245325576	0.184443163
56bit	0.163752084	0.169519753	0.230654326	0.184122907
128bit	0.166129807	0.170200627	0.26433909	0.182154953
256bit	0.165257958	0.18646317	0.26223776	0.180688844
1024bit	0.175072555	0.71512838	0.288558782	0.193203724
2048bit	0.185033156	3.701661608	0.334886596	0.216115685

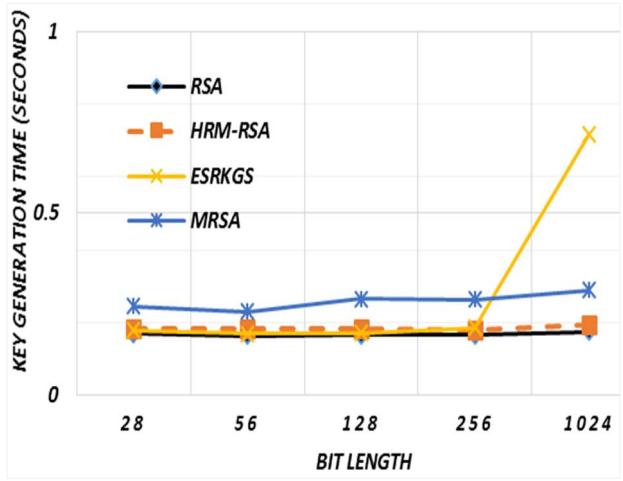
Based on key generation time due to Table 1, we have organized comparison Table 2. Table 2 shows that the key generation performance of HRM-RSA is 10% less than RSA; on the other hand, it improves key generation performance of ESRKGS and MRSA by 311% and 42%, respectively.

Table 2. Key Generation Time Comparison of HRM- RSA with RSA, ESRKGS and MRSA

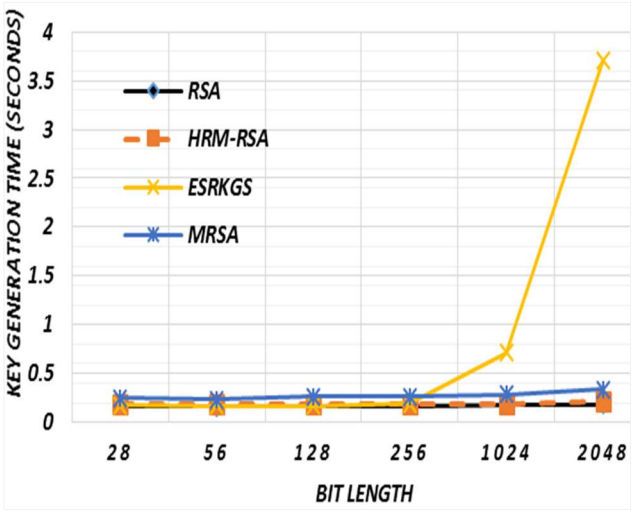
Bit Length	Algorithm		
	RSA	ESRKGS	MRSA
28bit	-8%	-4%	33%
56bit	-11%	-8%	25%
128bit	-9%	-7%	45%
256bit	-9%	3%	45%
1024bit	-9%	270%	49%
2048bit	-14%	1613%	55%
Average	-10%	311%	42%

Fig. 3(a) and Fig. 3(b) are generated from key generation time Table 1. To show the result in line with bit length increase in more detail, we have presented bit length from 28 to 1024 in Fig. 3(a) and from 28 to 2048 in Fig. 3 (b). According to Fig. 3, we can see that the key generation speed performance of HRM-RSA is slightly less than RSA. However, the key generation time performance of HRM-RSA is far better than ESRKGS and MRSA. Therefore, HRM-RSA key generation is less complex than MRSA and ESRKGS because it uses simple

computation which is real modulus hiding unlike others which use complex computation adding more exponentiation and modulation operations.



(a)



(b)

Figure 3. Analysis of Key Generation Performance (Seconds)

B. ENCRYPTION TIME OF ALGORITHMS

Table 3 shows the average encryption time of the algorithms for each bit. The table shows that our HRM-RSA algorithm is far better than other state-of-the-art algorithms.

Table 3. Summary of Encryption Time (in Seconds)

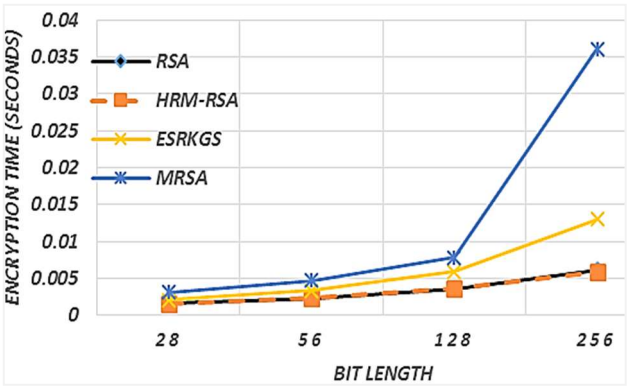
Algorithm/ Bit size	RSA	ESRKGS	MRSA	HRM-RSA
28bit	0.001664339	0.002098514	0.003099472	0.00162628
56bit	0.002234535	0.003376719	0.004664056	0.002316796
128bit	0.003467855	0.005904289	0.007825139	0.003617956
256bit	0.006172235	0.013026965	0.03611693	0.005847371
1024bit	0.11656283	0.462179563	2.010568515	0.121403782
2048bit	0.939976892	3.411535596	18.53038885	0.869988403

Based on encryption time due to Table 3, we have analyzed Table 4. Table 4 shows that HRM-RSA has improved encryption performance of traditional RSA, ESRKGS, and MRSA by 0.7%, 139%, and 735%, respectively.

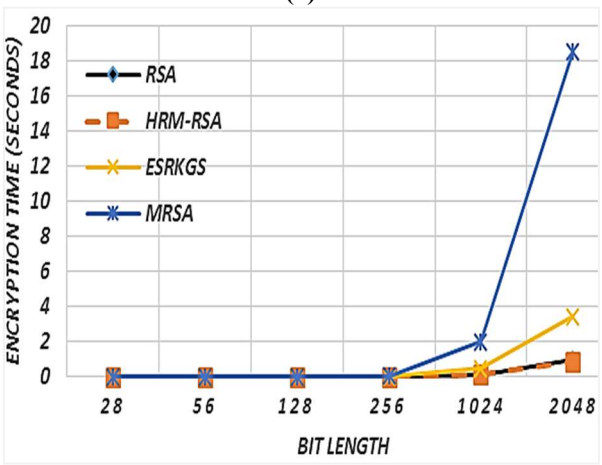
Table 4. Encryption Time Comparison of HRM-RSA with RSA, ESRKGS and MRSA

Algorithm/ Bit size	RSA	ESRKGS	MRSA
28bit	2%	29%	91%
56bit	-4%	46%	101%
128bit	-4%	63%	116%
256bit	6%	123%	518%
1024bit	-4%	281%	1556%
2048bit	8%	292%	2030%
Average	0.7%	139%	735.3%

Based on Table 3, we have analyzed Fig. 4 (a) and 4(b). These figures show that HRM-RSA performs better than other state-of-the-art algorithms; especially the performance difference from ESRKGS and MRSA is significant. Therefore, the encryption time performance of the HRM-RSA algorithm is better than the state-of-the-art algorithms.



(a)



(b)

Figure 4. Analysis of Encryption Performance (Seconds)

C. DECRYPTION TIME OF ALGORITHMS

Based on decryption time collected from the execution of each algorithm, we have summarized the average decryption time of the algorithms for each bit as shown in Table 5.

Table 5. Summary of Decryption Time (in Seconds)

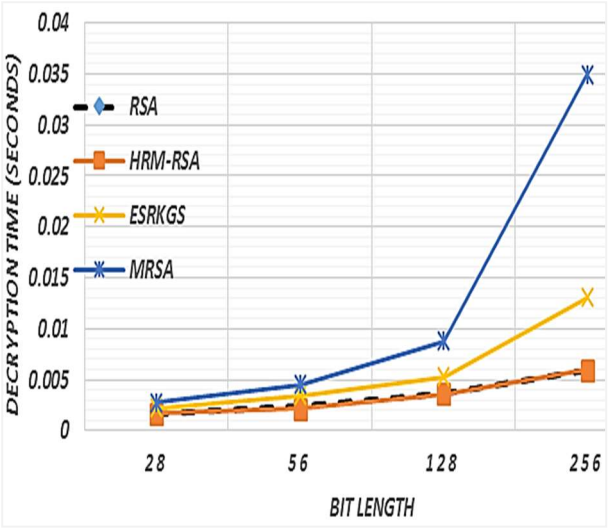
Algorithm/ Bit size	RSA	ESRKGS	MRSA	HRM-RSA
28bit	0.00168917	0.00218299	0.002762321	0.00166135
56bit	0.00238831	0.00346777	0.004531875	0.00215842
128bit	0.00368699	0.00523331	0.008751857	0.00360558
256bit	0.0059042	0.01308115	0.03492346	0.00592502
1024bit	0.1171262	0.45592054	2.215529872	0.11901915
2048bit	0.89804749	3.30941532	19.79810731	0.85252463

Based on decryption time due to Table 5, we have analyzed Table 6. Table 6 shows that in decryption speed performance, HRM-RSA outperforms the-state-of-the-art algorithms by 3%, 138%, and 799% than RSA, ESRKGS, and MRSA, respectively. This makes HRM-RSA cost-effective.

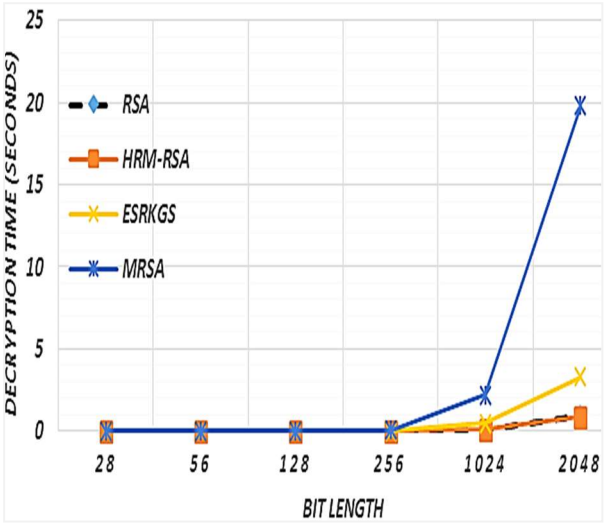
Table 6. Decryption Time Comparison of HRM- RSA with RSA, ESRKGS, and MRSA

Algorithm / Bit size	RSA	ESRKGS	MRSA
28bit	2%	31%	66%
56bit	11%	61%	110%
128bit	2%	45%	143%
256bit	0%	121%	489%
1024bit	-2%	283%	1761%
2048bit	5%	288%	2222%
Average	3%	138%	799%

Based on decryption time due to Table 5, we have analyzed Fig. 5(a) and Fig 5(b). These figures show that the decryption speed of HRM-RSA is better than other state-of-the-art algorithms. Especially, as the bit length of prime numbers increase our algorithm performance increases more significantly.



(a)



(b)

Figure 5. Analysis of Decryption Performance (Seconds)

D. TIME COMPLEXITY OF ALGORITHMS

Based on the complexity of MILLER-RABIN computed by M.A. Islam et al. (2018) [6] and Table 1 shown in this section, MRSA is more complex than RSA whereas HRM-RSA has less complexity than other related work except for traditional RSA. Therefore, HRM-RSA requires less computing resources than other existing works.

E. SECURITY ANALYSIS OF ALGORITHMS

Double Encryption Attack against RSA

In our work Double Encryption means the process of applying the public keys with the encryption algorithm both at the sender and receiver side, whereas Double Encryption Attack (DEA) means the plain text is recovered back when the receiver applies the public keys instead of private keys as described below:

Since common modulus is Public in Standard RSA, it is possible to use public key exponent both at the sender and at the receiver side to get the same result of its inverse called private key exponent by man in the middle (MITM) attack using Double Encryption Attack as follows:

Let C = cipher text, T= plain text, e = public key exponent, d = private key exponent and n = common modulus.

Sender Side: $C=T^{e} \bmod n$, Receiver Side: $T=C^{d} \bmod n$.

$T=C^{d} \bmod n$ but d is private so MITM can use $T= C^{e} \bmod n$. because 'e' and 'd' are inverses of each other within this common modulus n as shown in examples 1 and 2 below. For simplicity, we have used small integers.

Example 1: Assume: $K_u= (5, 21)$, C=16, T=?

Attacking: $T= (16)^5 \bmod 21=4$

Checking: $C=T^{e} \bmod n = (4)^5 \bmod 21=16$

Example 2: Assume: $K_u= (5, 35)$, C=17, T=?

Attacking: $T= (17)^5 \bmod 35=12$

Checking: $C=T^{e} \bmod n = (12)^5 \bmod 35=17$

Mathematical Theorems to Unlock RSA

a. Check primness of a number p

⇨ P is prime if it is not divisible by any primes $<=\sqrt{p}$

b. Find all prime numbers less than n

⇨ Write all number s b/n 2 and n

⇨ Check if any number m is divided by any prime $< \sqrt{n}$ (Sieve of Eratosthenes)

c. Euler's phi-Function; used to find co-prime numbers to n

$$\emptyset(m \times n) = \emptyset(m) \times \emptyset(n) = (m-1)(n-1)$$

Find any prime number p such that $\sqrt[3]{n}\ \sqrt[3]{n} <= p < \sqrt{n}$ and divider of n (Sieve of Eratosthenes), as one of the prime numbers lays within this range and key length of p is the key size of n–1, check 'p' from $\sqrt{n}$ -1.

Find another prime q =n/p, $\emptyset$ (n) = (p-1) (q-1). Then private key exponent (d) $=e^{-1} \bmod \emptyset$ (n), as key exponent e is known to the public.

Example 1: Let n=33, Ø (n) =?

Answer:

Find the range of one of the prime numbers let P, such that $\sqrt[3]{33} <= P<\sqrt{33} \rightarrow 3<= P <5$.

Find a prime number within a range and devisor of n

- ⇨ Check p=4 ---not prime; therefore, go to next p=p-1
- ⇨ Check p=3 ----yes (therefore 1st prime found)
- ⇨ Find q=n/p=33/3=11----- 2nd prime found
- ⇨ Find Ø(n) =(p-1)(q-1)=(3-1)(11-1)=20.

Example 2: Let n=13,221

Answer:

Find the range of one of the prime numbers let P, such that $\sqrt[3]{13221} \leq P < \sqrt{13,221}$; $23 \leq p < 114$.

Check which number in a range is a prime and devisor of 13,221

- ⇨ Check p=113---yes (1st prime found)
- ⇨ Find 2nd prime q=13,221/113=117
- ⇨ Find Ø(n) =(p-1)(q-1)=(113-1)(117-1)=12,992.

However, we have also computed a private key exponent d of RSA using Fermat's theorem as $d\equiv e^{-1} \bmod Ø(n)$. Using Fermat's theorem version 1 and 2, we have found that $d=e^{-1} \bmod Ø(n) \equiv e^{Ø(n)-1} \bmod Ø(n)$ from public key exponent e and common modulus n.

Procedures to Unlock RSA

We designed attacking procedures for RSA as shown in step 1 and 2 below based on the mathematical theorems shown in subsection 5.5.2. (Remember Ku = (e, n) is Public).

Step 1. Find Ø(n) :

(a). Find all Primes $< \sqrt{n}$ starting from $\sqrt{n} - 1$ (Sieve of Eratosthenes)

(b). Select a prime as P if divider of n i.e. GCD(P, n)=P

(c). Find q=n/p

(d). Calculate Ø(n)=(p-1)(q-1) (Euler's phi-function).

Step 2. Find $d=e^{-1} \bmod Ø(n)$ using

(a). $e^{Ø(n)}=1 \bmod Ø(n)$ (Euler's theorem)

(b). $e^{Ø(n)-1}=e^{-1} \bmod Ø(n)$ (divide both sides by e)

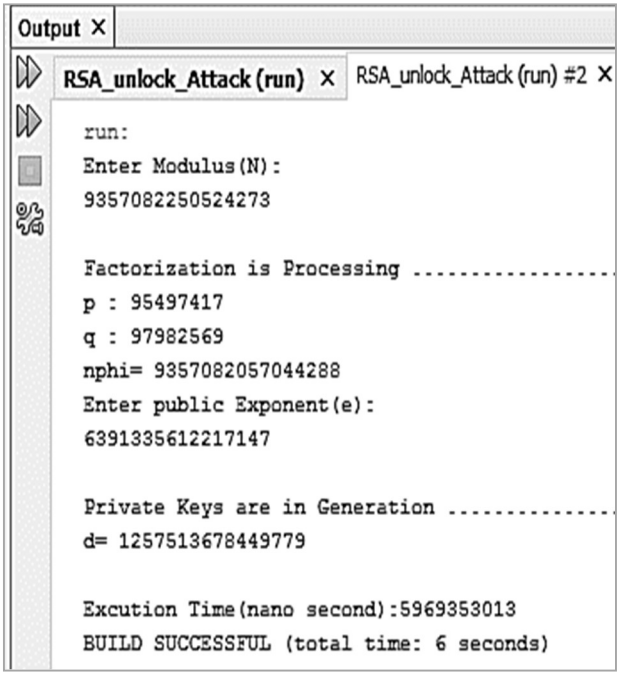
(c). $e^{-1}= e^{Ø(n)-1} \bmod Ø(n)$ (Fermat's little theorem)

(d). $de= 1 \bmod Ø(n)$ (private exponent generation algorithm)

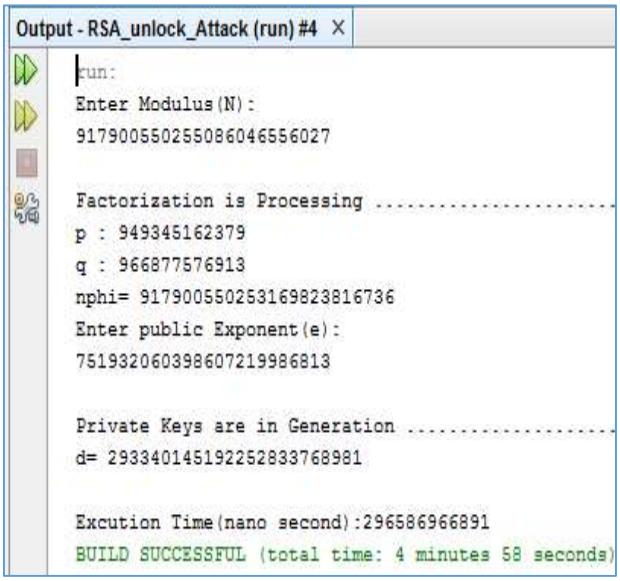
(e). $d= e^{Ø(n)-1} \bmod Ø(n)$ (substitute e^{-1} with Fermat's little theorem).

Experimental Output

We have implemented java program to unlock RSA based on unlocking procedures depicted in subsection 5.5.3. Experimental sample Java outputs for 56 bits and 80 bits are shown in Fig. 6 (a) and (b), respectively.



(a)



(b)

Figure 6. Experimental sample Java outputs

HRM-RSA Resistance against Double Encryption Attack

Let C=Cipher Text, T=Plain Text, e= public key exponent, d=private key exponent, n= a hidden real modulus, M= a public masking modulus, and K_U=Public key Components.

$C=T^e \bmod M$

$T=C^{d*e} \bmod M$

$T= (C^d \bmod M)^e \bmod M$

Example: Assume: correspondents know K_U= (291, 20723), C=5978, T=?

Primes= 17, 23 and Plain Text=63 (not public)

Attacking: $T= (5978)^{291} \bmod 20723=175 \neq 63$

***Checking*:** If we can or cannot compute back to its known cipher text using a known public key?

$C=T^e \bmod M=175^{291} \bmod 20723=1349 \neq 5978$.

This shows that HRM-RSA is resistant against a Double Encryption Attack.

HRM-RSA Resistance against Mathematical Theorems

To unlock existing work, we can find prime numbers P1, P2 … Pn between $\sqrt[3]{n}$ and $\sqrt{n}$ which are divisors of n (Sieve of Eratosthenes). Then we can compute Ø (n) and d $=e^{-1}$mod Ø (n), as e and n are public. However, in our HRM-RSA algorithm mask M is public and real modulus n is hidden, attackers cannot have any clue to unlock our cryptosystem.

Private Key exponent (d) $=e^{-1}$ mod Ø (n); but in our algorithm d and n are private key components; therefore, attackers cannot get n to factorize it and to drive d.

HRM-RSA Resistance against Multiple Private Exponent Attack

Due to keys generated between 1 and Ø (n), its private and public key exponents' size cannot be larger than a hidden real modulus n. It results in HRM-RSA to have a unique key pairs. Hence, a hidden real modulus n is kept private and a public mask modulus M has no relationship to prime numbers p and q, it is difficult for attackers to make factorization and multiple key generation attack.

Security Strength of Algorithms

We have summarized algorithms security strength analysis discussed in section "5" using Table 7.

Table 7. Comparison based on the Security Strength of the Algorithms

Algorithm	Strength	Reason
RSA	low	It uses direct mathematical relationship between common modulus and prime numbers. Factorization and a number of other attacks.
ESRKGS	Medium	Use more primes, indirect mathematical relationships. But alternative private key exponent, factorization, and other attacks are possible.
MRSA	High	Use more primes and double keys. But Factorization is possible.
HRM-RSA	Very High	Difficult to factorize and cryptanalysis because public mask modulus keeps real modulus hidden from the public in HRM-RSA algorithm unlike in RAS and other related work.

VI. CONCLUSIONS

Hidden Real Modulus based RSA cryptosystem has been proposed in this paper. The existing cryptosystem security algorithms are based on common modulus n which makes them vulnerable to different types of cryptanalysis attacks like factorization, polynomial-time quantum algorithms, multiple exponents, double encryption attacks, etc. Since our novel algorithm hides the real modulus n from attackers using random mask multiplier m by converting it into public mask modulus M, it leaves no clue for attackers. As a result, we have found that it is highly secured than existing systems. Hence, as the real modulus n is hidden from the public, key exponents' (e, d) and prime numbers (p, q) are not dependent on a public mask modulus M, and public modulus M has unpredictable property, it is difficult to unlock our cryptosystem. As a result, the false cipher is transported over a network medium that makes it more difficult for attackers to conduct attacks based on cipher text.

Generally, our proposed, HRM-RSA algorithm, has improved "security strength", "key generation speed", "encryption speed," and "decryption speed". This makes it more ideal to be implemented in very security demanding environments like Banks, e-commerce, etc. applications.

As future work, this algorithm can be extended by considering higher key lengths; real world implementation of the algorithm in different applications: pretty-good-privacy (PGP), cryptocurrency transaction, mobile communication, wireless communication, and others.

References

[1] S. Kumar, et al., "Comparative study on AES and RSA," *Proceedings of the International Conference on Communication and Signal Processing*, India, April 6-8'2018, pp. 0501-0504.

[2] U. Thirupalu and E. K. Reddy, "Performance analysis of cryptographic algorithms in the information security," no. March, *International Journal of Engineering Research & Technology (IJERT)*, vol. 8, issue 2, 2019.

[3] D. Mahto and D. K. Yadav, "Performance analysis of RSA and elliptic curve cryptography," vol. 20, no. 4, pp. 625–635, 2018, doi: 10.6633/IJNS.201807.

[4] W. Stallings, *Cryptography and Network Security: Principles and Practice*, eight edition, Pearson Education, 2020, ISBN 978-0-13-670722-6.

[5] A. Hamza and B. Kumar, "A review paper on DES, AES, RSA encryption standards," *Proceedings of the SMART–2020, IEEE Conference ID: 50582 9th International Conference on System Modeling & Advancement in Research Trends*, 2020, pp. 333–338. https://doi.org/10.1109/SMART50582.2020.9336800.

[6] M. A. Islam, et al., "A modified and secured RSA public key cryptosystem based on "n" prime numbers," *Journal of Computer and Communications*, vol. 6, issue 3, pp. 78–90 2018. https://doi.org/10.4236/jcc.2018.63006.

[7] P. K. Panda, and S. Chattopadhyay, "A hybrid security algorithm for RSA cryptosystem," *Proceedings of the 2017 4th International Conference on Advanced Computing and Communication Systems, ICACCS'2017*, 2017, pp. 1-6. https://doi.org/10.1109/ICACCS.2017.8014644.

[8] B. S. Mathematics, D. S. B. S, and S. Barbara, "Basic application of mathematics in cryptography," *Proceedings of the 2020 IEEE International Conference on Modem Education and Information Management (ICMEIM)*, 2020, pp. 871–875.

[9] L. K. Galla, V. S. Koganti, and N. Nuthalapati, "Implementation of RSA," *Proceedings of the 2016 Int. Conf. Control Instrum. Commun. Comput. Technol. ICCICCT 2016*, 2017, pp. 81–87, https://doi.org/10.1109/ICCICCT.2016.7987922.

[10] F. Shahid, et al., "PSDS–proficient security over distributed storage: A method for data transmission in cloud," *IEEE Access*, vol. 8, pp. 118285-118298, 2020. https://doi.org/10.1109/ACCESS.2020.3004433.

[11] Y. Wang, S. Yan, & H. Zhang, "A new quantum algorithm for computing RSA cipher text period," *Wuhan Univ. J. Nat. Sci*. vol. 22, pp. 68–72, 2017. https://doi.org/10.1007/s11859-017-1218-5.

[12] B. Wang, X. Yang, and D. Zhang, "Research on quantum annealing integer factorization based on different columns," *Frontiers in Physics*, vol. 10, no. June, pp. 1–10, 2022, https://doi.org/10.3389/fphy.2022.914578.

[13] Y. Wang, H. Zhang, and H. Wang, "Quantum polynomial-time fixed-point attack for RSA," *China Communications*, pp. 25–32, 2018. https://doi.org/10.1109/CC.2018.8300269.

[14] M. Bunder, A. Nitaj, W. Susilo, and J. Tonien, "A generalized attack on RSA type cryptosystems," *Theor. Comput. Sci.*, vol. 1, pp. 1–8, 2017, https://doi.org/10.1016/j.tcs.2017.09.009.

[15] W. Susilo, W. Susilo, J. Tonien, and G. Yang, "Institutional knowledge at Singapore Management University – A generalised bound for the Wiener attack on RSA," vol. 2020, pp. 1–4, 2020. https://doi.org/10.1016/j.jisa.2020.102531.

[16] M. Mumtaz, and L. Ping, "Cryptanalysis of a special case of RSA large decryption exponent using lattice basis reduction method," *Proceedings of the* IEEE 6th International Conference on Computer and Communication Systems (ICCCS), Chengdu, China, 2021, pp. 714-720, https://doi.org/10.1109/ICCCS52626.2021.9449268.

[17] J. Mittmann and W. Schindler, "Timing attacks and local timing attacks against Barrett's modular multiplication algorithm," *J. Cryptogr. Eng.*, vol. 11, no. 4, pp. 369–397, https://doi.org/10.1007/s13389-020-00254-3.

[18] C. Shao, H. Li and X. Zhang, "Cryptographic implementation of RSA for ion fault injection attack," Proceedings of the *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, 2014, pp. 791–796. https://doi.org/10.1109/CCNC.2014.6994410.

[19] M. Mumtaz and L. Ping, "Forty years of attacks on the RSA cryptosystem: A brief survey," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 22, issue 1, pp. 9-29, 2019, https://doi.org/10.1080/09720529.2018.1564201.

[20] Y. Y. Ogorodnikov, "A combined attack on RSA algorithm by SAT-approach," *Proceedings of the 2016 Dynamics of Systems, Mechanisms and Machines, Dynamics 2016*. 2017, pp. 1-6. https://doi.org/10.1109/Dynamics.2016.7819055.

[21] R. Shamir, A. Public, and K. Cryptosystem, "A study and implementation of RSA cryptosystem," Computer Science and Engineering Department, Jadavpur University, arXiv: 1506.04265v1 [cs.CR] 13 Jun 2015.

[22] R. Jaiswal, et al., "Reformed RSA algorithm based on prime number," *International Journal of Computer Applications*, pp. 23-26, 2014.

[23] M. Thangavel, et al., "An enhanced and secured RSA key generation scheme (ESRKGS)," *J. Inf. Secur. Appl.*, vol. 20, pp. 3–10, 2015. https://doi.org/10.1016/j.jisa.2014.10.004.

[24] E. Lüy, et al., "Comment on 'an enhanced and secured RSA key generation scheme (ESRKGS)," *J. Inf. Secur. Appl.*, vol. 30, pp. 1–2, 2016. https://doi.org/10.1016/j.jisa.2016.03.006.

[25] S. Mathur, et al., "Analysis and design of enhanced RSA algorithm to improve the security," *Proceedings of the 3rd IEEE International Conference on Computational Intelligence & Communication Technology (CICT)*, Ghaziabad, 2017, pp. 3–7. https://doi.org/10.1109/CIACT.2017.7977330.

GETANEH AWULACHEW ZIMBELE (MSc) *was born in Majete Town, North Shoa Zone, Amhara Region, Ethiopia in 1989. He received the B.S. degree in Information Technology from Adama Science and Technology University in 2013 and M.S. degree in Computer Networks and Security from Debre Berhan University, Ethiopia, in 2019. Since 2017, he has been working as a Lecturer with the Information Technology Department, Debre Berhan University. He has two local research presented for conference proceedings and one project at National level. His research interests are in the area of Information Security and resource efficient routing algorithms in MANET.*

SAMUEL ASFERAW DEMILEW (PhD) *was born in Chagni Town, Gojjam, Amhara Region, Ethiopia in 1980. He received the B.A. degree in English from Bahir Dar University in 2000; M.S. degree in Information Science from Addis Ababa University in 2007 and the Ph.D. degree in Information Technology from Addis Ababa University, Ethiopia in 2017. From 2008 to 2009 he used to work as Dean of Engineering Faculty at Debre Berhan University. Since 2018, he has been working as an Assistant Professor with the Information Technology Department, Debre Berhan University. He has publications on IEEE proceedings and journals. His research interests include network security, information security, and wireless network, node geo-localization, energy-efficient routing in MANET and sensor networks.*