Renovated RSA Cryptosystem for secure Data Transmission

P.Sri Ram Chandra1, Dr. G. Venkateswara Rao2 and Dr. G.V.Swamy3
1Research Scholar (Computer Science and Engineering Department), Gandhi Institute of Technology and Management (GITAM), Visakhapatnam, Andhra Pradesh, India and Faculty member in Computer Science and Engineering, Godavari Institute of Engineering and Technology (A), Rajahmundry, Andhra Pradesh, India.
2Professor and Head, Electronics and Physics Department, Godavari Institute of Engineering and Technology (A), Rajahmundry, Andhra Pradesh, India.
3Corresponding Author

Abstract
Cryptographic techniques are the principal means to provide information security, RSA (Rivest–Shamir–Adleman) is one of the first practical public-key cryptosystems and is widely used for secure data transmission. The first part of this paper describes the functionality of RSA and its attacks, identified ‘factoring the N’ is the most common one. Now authors have proposed the Renovated RSA (RRSA) cryptosystem through the use of Armstrong prime number(r) in addition to the existing two prime numbers (p, q) to generate the public (e) and private (d) keys. The theoretical proof of the newly developed cryptosystem was shown to ensure the functionality of the original RSA not to be disturbed. In the second part of this paper, strength of RRSA cryptosystem was analysed over size of the variable N, in such a way that it cannot be factored i.e., N with more than 232 decimal digits. The theoretical samples considered as per the algorithm requirements, revealed that the size of the variable N in RRSA can go beyond 232 decimal digits by using an Armstrong prime (r) even with the less number of digits in p, q when compared to RSA. A series of algorithms were also discussed to perform the calculations involved in the encryption and decryption processes at a faster rate even with several hundred digits there in e, d, N.

Keywords: Cryptosystem, Armstrong prime number, Prime number, Factorization Complexity, Encryption, Decryption, Co-Prime, Public Key, Private Key, Intruder.

INTRODUCTION
We are in the era of business on internet where the computer applications were developed to handle the data related to crucial areas like finance, banking, army and government. In transferring such significant data across the network may sometimes get into the hands of the intruders who may tamper the contents of the data [1]. In this regard the security measures are to be taken to protect the data. Cryptography is one of the best suitable platforms to ensure secure data transmission where the data encryption and decryption processes are involved with or without a secret key [1]. In cryptography, a cryptosystem is a suite of three algorithms: one for key generation, one for encryption, and the other for decryption. They are classified as Symmetric key Cryptosystem (same encryption key is used for decryption), Asymmetric key Cryptosystem (different keys for encryption and decryption are used).

Before the introduction of asymmetric-key cryptography by Diffie and Hellman [3], at first both the users has to come up with an agreement in regard of the encryption and decryption processes there after they can communicate in private with encrypted messages sent between them, said as symmetric key cryptography. An asymmetric-key cryptosystem is one in which each user places an encryption procedure E into a public file, each user has a corresponding decryption procedure D, the details of which the user does not reveal to anyone else. The key to ensure the security of asymmetric-key cryptosystem is for it to be extremely difficult to derive the decryption key from the publicly available encryption key [2].

The cryptosystem developed should ensure the few principles of security like confidentiality, authentication, integrity, non-repudiation [5]. Considering the data transmission between two entities A and B, ‘if A ensures that none expect B gets the data’ is termed to be as confidentiality. Integrity states that both A and B will undergo an agreement such that, none of them would tamper the data further [1]. ‘B assures that the data was sent by A only’ designated as authentication. Non-repudiation does not allow the sender of a message to refuse the claim of not sending the message [1].

THE RSA ALGORITHM
The vital feature of RSA public-key cryptosystem is that the encryption and decryption procedure are done with two different keys - public key and private key respectively. Its security is based on the issues like difficulty of the large number prime factorization, which is a well-known mathematical problem that has no effective solution [6]. The following algorithm1 gives the result of RSA [5].
Algorithm 1

Step1. Key Generation
Step2. Encryption
Step3. Decryption

Procedure for Step1 (Key Generation):

- Select two distinct prime numbers \( p \) and \( q \).
- Calculate \( N = p \times q \).
- Calculate \( \phi (n) = (p-1) \times (q-1) \).
- Select an integer \( e \) whose \( \gcd (\phi (n), e) = 1 \); \( 1 < e < \phi (n) \).
- Determine \( d \) (using modular arithmetic) which satisfies the congruence relation \( d \times e \equiv 1 \pmod{\phi (n)} \).
- \( d \) is kept as private key component.
- Public key = \( \{e, N\} \).
- Private Key = \( \{d, N\} \).

Procedure for Step2 (Encryption):
\[ C = M^e \mod N \]

Procedure for Step3 (Decryption):
\[ M = C^d \mod N \]

Where, \( C \) - Cipher text, \( M \) - Message, \( p \) and \( q \) - Prime Numbers, \( N \) - Common Modulus, \( e \) and \( d \) - Public and Private Keys Respectively.

The Attacks on RSA public-key Cryptosystem

The saying “A chain is no stronger than its weakest link” is very suitable for describing attacks on cryptosystems. Most of the attackers’ instinct is to go for the weakest link of the chain which includes key generation, key management, the cryptographic algorithm and the cryptographic protocol. The consequent sections briefly describe the attacks on RSA and the factored values of RSA-N.

Searching the Message Space

One of the appearing weaknesses of RSA is that one has to give away everybody the algorithm that encrypts the messages. Considering the message space is small i.e., less than 9 characters and we are using English as our plaintext that gives us \( 26^9 = 5429503678976 \) possible plaintext messages. If it takes 0.18 milliseconds to encrypt a message using RSA 1024, then it should take 434360294318.08 milliseconds, or 5027 days to cycle through all possible messages on one computer [14].

Mitigation

The simple solution to overcome this attack is to send the large messages preferably the size greater than 18 characters [14], which in turn the time to cycle all through the plaintext messages grows exponentially. Further this would become an insurmountable task for the intruder to crack the original message.

Common modulus

In common modulus attack, the intruder detects the plaintext message without factoring \( N \) or finding the secret decryption exponent \( d \) [14]. Imagine a scenario where John would like to send the message \( M \) to Alice and Bob individually. To do this Alice gives John her public key \( (N, e_1) \) and Bob gives John his public key \( (N, e_2) \) where \( e_1 \) and \( e_2 \) are relatively prime. The problem arises because Alice and Bob are both using the same modulus \( N \). John sends \( C_1 = M^{e_1} \mod N \) to Alice and \( C_2 = M^{e_2} \mod N \) to Bob. Now suppose an eavesdropper Eve intercepts \( C_1 \) and \( C_2 \). Since \( \gcd (e_1, e_2) = 1 \), Eve can use the Euclidean Algorithm to get integers \( x \) and \( y \) such that \( 1 = e_1x + e_2y \). Exactly one of \( x \) or \( y \) will be negative. Without loss of generality assume \( x \) is negative. Eve can now calculate
\[
(C_1x)^e = (C_2y) = C_1^{x}C_2^{y} = (M^{e_1})^{x}(M^{e_2})^{y} = M^{e_1x+e_2y} = M^{I} = M \pmod{N}
\]

Thus, Eve can detect the plaintext message without factoring \( N \) or uncovering the decryption exponent \( d \) [14].

Mitigation

In order to make the RSA free from this attack, two individual users in the same channel of communication should have unique modulus value \( N \).

Low exponent value for \( e \).

Most of the times RSA cryptosystem uses the lower exponent value viz., \( e = 3 \) for making the encryption faster. However, there is a vulnerability that if the same message is encrypted 3 times with different keys i.e., same exponent with different moduli then the intruder can retrieve the message [17].

Mitigation

Instead of selecting the lower exponent value for \( e \), it is preferred to use the larger exponent value and an efficient computing machine which can do encryption at a faster rate.
Man-In-The-Middle Attacks

A ‘man-in-the-middle’ attack is a kind of active attack where the attacker furtively relays and perhaps amends the conversation or keys between the two entities who believe they are directly communicating with each other [18]. The Mail-Man can listen in our conversations and collect the information we disclose. He could also fabricate any information in order to take control over the communication channel. The following figure 1 depicts this attack.

![Figure 1: Man-In-The-Middle Attack.](image)

Mitigation

To avoid Man-In-The-Middle attacks we use a trusted third party. In this scenario Alice trusts Trent as a key distributor and Trent will ensure that the key will go to Alice and not the Eve. Figure 2 clearly depicts the avoidance of this attack.

![Figure 2: Avoidance of Man-In-The-Middle Attack.](image)

Factoring Attacks

Factoring attacks on RSA are referred to as brute force attacks. These attacks rely on factoring the modulus $N$ into its distinct prime factors. If $N$ can be factored then calculating $\phi(N) = (p-1)(q-1)$ is an effortless task, further allows one to find the exponent $e$ and $d$ by solving $gcd(e, \phi(N)) = 1$ and $ed \equiv 1 \pmod{\phi(N)}$ respectively. This means as the size of $N$ increases, the time it takes for $N$ to be factored increases exponentially.

Popular Factoring algorithms

a) Fermat’s Factoring Method

Fermat’s method factors an integer $N$ by writing it as a difference of squares, $N = x^2 - y^2$. Where $x, y \in \mathbb{Z}$. Then $N = (x + y)(x - y)$. If $(x + y)$ or $(x - y)$ are not prime numbers, then Fermat’s method can be repeated with those values.

For example $N = 1121$, the Fermat’s method starts by considering $x$ as square root of $N$, i.e., $x=34$. Now construct the table 1 with two columns $x$ and $x^2 - N$ incrementing the value of $x$ by one till the $x^2 - N$ can be perfect square.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$x^2 - N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td>35</td>
<td>104</td>
</tr>
<tr>
<td>36</td>
<td>175</td>
</tr>
<tr>
<td>37</td>
<td>248</td>
</tr>
<tr>
<td>38</td>
<td>323</td>
</tr>
</tbody>
</table>

Since $400 = 20^2$ we can write $1121 = 39^2 - 20^2 = (39-20)(39+20) = 19*59$ i.e., prime factorization of $N = 1121$ is $19*59$ and the primes considered at the initial stage of the RSA cryptosystem could be $19$ and $59$.

b) Quadratic Number Field Sieve (QNFS)

The QNFS is considered the fastest algorithm for factoring numbers with approximately 50-100 decimal digits. It is an optimized version of Fermat’s factoring method. Fermat’s method starts at an initial value and increments by one until $x^2 - N$ is a perfect square. This can take a very long time. Instead, the QNFS only tries $x$ values that are considered smooth. A smooth number is one that only has "small" prime factors [15]. The steps mentioned in the algorithm 2 gives the result of QNFS.

c) General Number Field Sieve (GNFS)

The GNFS is considered the fastest algorithm for factoring numbers with approximately greater than 100 decimal digits in number. There are a lot of different implementations of the GNFS and the details are fairly advanced and beyond the scope of this paper, but the efficiency of the GNFS are that the largest RSA modulus that was successfully factored using the GNFS was 232 decimal digits [19].

Mitigation

Although being able to factor in polynomial time would be a great achievement in Mathematics and Computer Science, not being able to allow for relatively simple solutions to mitigate factoring attacks. All we have to do is increase the size or digits of $N$. 
Algorithm 2

1. Choose a smoothness bound \( B \). The number \( \pi(B) \), denoting the number of prime numbers less than \( B \), will control both the length of the vectors and the number of vectors needed.

2. Use sieving to locate \( \pi(B) + 1 \) numbers \( a_i \) such that \( bi \equiv (a_i^2 \mod n) \) is \( B \)-smooth.

3. Factor the \( bi \) and generate exponent vectors \( \mod 2 \) for each one.

4. Use linear algebra to find a subset of these vectors which add to the zero vector. Multiply the corresponding \( ai \) together naming the result \( \mod n: a \) and the \( bi \) together which yields a \( B \)-smooth square \( b^2 \).

5. We are now left with the equality \( a^2 \equiv b^2 \mod n \) from which we get two square roots of \( \phi(N) \). The analysis over cryptosystem attacks described in section 3.5. It is much easier to find the corresponding \( a \) computed in step 4.

6. We now have the desired identity: \( (a+b)(a-b) = 0 \mod n \). Compute the GCD of \( n \) with the difference (or sum) of \( a \) and \( b \). This produces a factor, although it may be a trivial factor (\( n \) or 1). If the factor is trivial, try again with a different linear dependency or different \( a \).

Factored values of RSA

As the most common attack over RSA cryptosystem is Factoring the modulus \( N \). We did a search to identify ‘\( N \) with how many digits’ are factored [20]. The following table 2 describes it with a step increment of 10.

**RRSA Cryptosystem**

The analysis over cryptosystem attacks described in section regarding the attacks over RSA helped us in identifying the most common and dangerous attack over RSA i.e., “Factoring the modulus \( N \)”, the solution provided is “Increased size or digits of \( N \)”. In this regard we proposed an enhancement, primarily focus on increasing the size of \( N \) through the use of Armstrong prime number \( r \) in addition to the existing two prime numbers \( p \) and \( q \). Thus \( e \) and \( d \) are used to generate the public and private keys. Armstrong prime number is defined such that sum of \( n^k \) powers of individual digits is equal to that number and it should be prime, where \( n \) is number of digits in given number. Prime number is a number that is divisible only by itself and 1. The key strength of the RSA depends on the two prime numbers \( p \) and \( q \). The process of factoring the \( N \) will lead to gain the values of \( p \) and \( q \) using any of the algorithms as described in section 3.5. It is much easier to find set of two numbers from factoring \( N \) than finding the set of three numbers from \( N \). In this way our proposed cryptosystem gives an insurmountable task for the intruder to find the three values \( p \), \( q \), \( r \) from factoring \( N \).

**Table 2:** A few ‘RSA-N values with their decimal digits’ that are factored

<table>
<thead>
<tr>
<th>S.No</th>
<th>RSA-N</th>
<th>No of 8 digits of extreme ends are considered</th>
<th>No of digits in RSA-N</th>
<th>No of bits to represent ( N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15226050........92006139</td>
<td>100</td>
<td>330</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>35794234.......17568667</td>
<td>110</td>
<td>364</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>22701048.......96548479</td>
<td>120</td>
<td>397</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>11438162........79543541</td>
<td>129</td>
<td>426</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>18070820........14880557</td>
<td>130</td>
<td>430</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>21290246........41936471</td>
<td>140</td>
<td>463</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>15509891........95964683</td>
<td>150</td>
<td>496</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10941738........54333897</td>
<td>155</td>
<td>512</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>21527411........70407753</td>
<td>160</td>
<td>530</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>26062623........11545759</td>
<td>170</td>
<td>563</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>18819881........50257059</td>
<td>174</td>
<td>576</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>19114792........51421041</td>
<td>180</td>
<td>596</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>19075564........01423481</td>
<td>190</td>
<td>629</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>27997833........21823983</td>
<td>200</td>
<td>640</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>24524664........70551067</td>
<td>210</td>
<td>663</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>74037563........63796359</td>
<td>212</td>
<td>704</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>22601385........96955261</td>
<td>220</td>
<td>729</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>12301866........02143413</td>
<td>232</td>
<td>768</td>
<td></td>
</tr>
</tbody>
</table>

**Key Generation**

Algorithm 3

- Choose two distinct Prime numbers \( p \), \( q \) and Armstrong Prime number \( r \).
- Find \( N \) such that \( N = p \times q \times r \). [\( n \) will be used as the modulus for both the public and private keys.]
- Find the Phi of \( N \), \( \phi (N) = (p-1) \times (q-1) \times (r-1) \).
- Choose an \( e \) such that \( 1 < e < \phi (N) \), and such that \( e \) and \( \phi (N) \) share no Divisors other than 1 (\( e \) and \( \phi (N) \) are co-prime). i.e., \( gcd (e, \phi (N)) =1 \). \( e \) is kept as the public key exponent.
- Determine \( d \) (using modular arithmetic) which satisfies the congruence relation \( d \times e \equiv 1 \mod \phi (N) \). \( d \) is kept as private key component.
Encryption
The process of encoding a message with proper keys in such a way that only authorized users can access it. The Encrypted or encoded text is also called Cipher text (C) whereas the original message is called Plaintext (M). The Encryption process involved in the algorithm is specified as
\[ C = M^e \mod N. \]

Decryption
The process of un-encrypting the message manually or using proper codes in such a way that the human/computer can understand the message termed as Decryption. The process of un-encrypting the message encoded text is also called Cipher text (C).

Number theory
As per the algorithm used in RRSA, we need to identify the series of prime numbers and Armstrong numbers among them. Accordingly we found that the least even prime is 2, least odd prime is 3 and the greatest prime as of August 2017 is 2^{24,207,281}-1,a number with 22,338,618 digits. It was found in January 2016 by the Great Internet Mersenne Prime Search (GIMPS). Further we did a search for Armstrong prime numbers and identified as follows2,3,5,7,281177399146038697307 and 35452590104031691935943. The largest Armstrong number identified is of 39 decimal digits.

Proof of the RRSA Cryptosystem.
The RRSA Cryptosystem has been formulated by considering an Armstrong prime in addition to the two prime numbers ensuring that the theoretical functionality of the original RSA should not be disturbed, the following proof is being considered as below.

Given positive integers \( N, e, \) and \( d \) such that [9]: \( N = p*q*r \), where \( p, q \) are distinct primes, \( r \) is an Armstrong prime, \( e \) and \( d \) represent public and private keys respectively……………… (1)
\[ \text{gcd} (e, \phi (N)) = 1 \]……………….. (2).
\[ ed \equiv 1 \mod \phi (N) \]………………………… (3).

Define the public and private key transformations of a message \( M \) to be respectively,

\[ \text{RRSA Public} (M) = M^e \mod N \]………………………… (4).
\[ \text{RRSA Private} (M) = M^d \mod N \]………………………… (5).
\[ M = \text{RRSA Private} (\text{RRSA Public} (M)) \], and that………… (6).
\[ M = \text{RRSA Public} (\text{RRSA Private} (M)) \]………………………… (7).

Considering the equations (6) and (7), if we can prove that they can be used inversely to obtain the message \( M \), we can say that the RRSA encryption is perfectly working.

By substituting equations (4) and (5) into (6) and (7) respectively, we can say that [9].

\[ \text{RRSA Private} (\text{RRSA Public} (M)) = (M^e \mod N)^d \mod N = M^{de} \mod N \] and also
\[ \text{RRSA Public} (\text{RRSA Private} (M)) = (M^d \mod N)^e \mod N = M^{de} \mod N. \]

Therefore, equations (6) and (7) are equivalent, or
RRSA Private (RRSA Public (M)) = RRSA Public (RRSA Private (M)).

If we can prove: \( M = M^{de} \mod N \), then the proof will be complete [9]. It is given that:
\[ de \equiv 1 \mod \phi (N) \]…………………………………… (3).
By the definition of mods
‘mod’ as a congruence relation: The notation ‘\( a \equiv b \mod n \)’ means \( a \) and \( b \) have the same remainder when divided by \( n \), or, equivalently,
(a) \( n \mid a - b \), or
(b) \( a - b = nk \) for some integer \( k \) hence we can write (3) as
\[ \phi (N) \mid de - 1 \]………………………… (8).

Since \( \phi (N) = \phi (p) \phi (q) \phi (r) \) only when \( p, q \) and \( r \) are relatively prime, as in this case, we have \( \phi (N) = \phi (p) \phi (q) \phi (r) \) and by substituting it into (8) we have [9].
\[ \phi (p) \phi (q) \phi (r) \mid de - 1 \]………………………… (8).

By properties of divisors, the notation \( n \mid a \) means \( a \) divides \( n \) and if \( mn \mid a \) then \( m \mid a \) and \( n \mid a \) for any integers \( m, n \), therefore we can write as [9].
\[ \phi (p) \mid de - 1 \]
\[ \phi (q) \mid de - 1 \]
\[ \phi (r) \mid de - 1 \]

Where there must be an integer \( k \) such that: \( de - 1 = k(p - 1) \)………………………… (9).

By the symmetric property of mods, we can write.
\[ M^{de} \equiv M^{de - 1 + 1} \mod p \]
Which can also be written as
\[ M^{de} \equiv (M^{de - 1}) \cdot M \mod p \]………………………… (10).
Substituting (9) into (10), we obtain
\[ M^{de} \equiv (M^{k(p - 1)}) \cdot M \mod p \]………………………… (11).

Since \( p \) is prime and for any integer \( M \), the equation (11) will be either

i. Relatively prime to \( p \)
ii. A multiple of \( p \).

When

i) \( M \) is relatively prime to \( p \), Fermat's Little Theorem states that[9]: \( M^{p - 1} \equiv 1 \mod p \).
By properties of mods, we can write:
\[ M^{k(p-l)} \equiv 1^k \pmod{p}, \]
\[ M^{k(p-l)} \equiv 1 \pmod{p} \] \hspace{1em} (12).

By combining (11) and (12), we obtain:
\[ M^{de} \equiv 1 \pmod{m}, \] \hspace{1em} (mod p), or
\[ M^{de} \equiv M \pmod{m} \] \hspace{1em} (mod p).

Hence the proof is completed.

The modular property of congruence states that when \( m \) and \( n \) are relatively prime \( a \equiv b \pmod{m} \), and \( a \equiv b \pmod{n} \), then \( a \equiv b \pmod{mn} \), we can write (14) as
\[ M^{de} \equiv M \pmod{p \cdot q \cdot r} \equiv M \pmod{N}. \]

By the modular property of symmetry, we can write
\[ M \equiv M^{de} \pmod{N} \] \hspace{1em} (mod m).

Since we have limited \( M \) to \( 0 \leq M < N \), only one integer will satisfy (15), and so
\[ M \equiv M^{de} \pmod{N} \] \hspace{1em} (mod m).

If we substitute equation (16) with our original equations
We get: RRSA Private (RRSA Public (M)) = M \( \pmod{N} \)
RRSA Public (RRSA Private (M)) = M \( \pmod{N} \)

We obtain, for \( 0 \leq M < N \),
RRSA Private (RRSA Public (M)) = M \& RRSA Public (RRSA Private (M)) = M.

Hence the proof has completed.

**Strength of the RRSA Cryptosystem – Analysis**

The strength of the cryptosystem majorly depends on its key generation process. Here \( N \) is used for generating the public and private key components, so our major motto is to protect \( N \) i.e., making it not factored or increase in the factorization time. The section 3.6 has revealed that \( N = p^q r \) with 232 decimal digits has factored till now. As our proposed system uses \( N = p^q r \), the size of \( N \) can go beyond 232 decimal digits even with the reduced size of \( p \) and \( q \), it is an insurmountable task for the intruder to find the three values \( p \), \( q \), \( r \) because \( N \) cannot be factored further. In view of increasing the number of digits or size of \( N \), we did an analysis to find the number of minimum and maximum number of digits that can be present in \( N \) when the multiplication is performed on two primes \( (p, q) \) and an Armstrong prime \( (r) \), the following table 3 describes it.

In view of intruder, he/she could majorly concentrate on prime factors of \( N \). In making use of Armstrong prime in this cryptosystem he/she has to check both the prime factor and Armstrong prime factor conditions, which could take an infinite amount of time to find factors of \( N \).

As per the user point of view calculating \( C = M^e \pmod{N} \) with more than 232 decimal digits in \( N \) is a tough task, in this regard the following algorithm 4 and algorithm 5 supports the RRSA cryptosystem to enhance the encryption by performing the ‘large number mod \( N \’ at a faster rate.

In our cryptosystem we are working toward, it is necessary to compute \( M^e \pmod{N} \) for values of \( M \), \( e \) and \( N \) that are several hundred bits long.

<table>
<thead>
<tr>
<th>( n(p) )</th>
<th>( n(q) )</th>
<th>( n(r) )</th>
<th>Min(N)</th>
<th>Max(N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>14</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>21</td>
<td>25</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>23</td>
<td>29</td>
<td>31</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>14</td>
<td>22</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>21</td>
<td>26</td>
<td>28</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>23</td>
<td>121</td>
<td>123</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>1</td>
<td>199</td>
<td>201</td>
</tr>
<tr>
<td>110</td>
<td>110</td>
<td>14</td>
<td>232</td>
<td>234</td>
</tr>
<tr>
<td>200</td>
<td>200</td>
<td>21</td>
<td>419</td>
<td>421</td>
</tr>
<tr>
<td>250</td>
<td>250</td>
<td>23</td>
<td>521</td>
<td>523</td>
</tr>
</tbody>
</table>

Generalized formula to find the Minimum and Maximum number of digits in \( N \) with \( p, q, r \):

\[
x = \frac{(a+c+d)}{3}
\]

<table>
<thead>
<tr>
<th>( n(p) ): No. of digits in ( p )</th>
<th>( n(q) ): No. of digits in ( q )</th>
<th>( n(r) ): No. of digits in ( r )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max(N): Maximum no. of digits that can be present in ( N )</td>
<td>Min(N): Minimum no. of digits that can be present in ( N )</td>
<td></td>
</tr>
</tbody>
</table>

However, the raw value of \( M^e \) could be much, much longer than this. Even when \( M \) and \( e \) are just 20-bit numbers, \( M^e \) is at least \( 2^{19} (524288) \), about 10 million bits long! Imagine what happens if \( e \) is a 500-bit number! To make sure that the operations on these numbers are to be simplified, we need to perform all intermediate computations of modulo \( N \). The idea hidden in this, is to calculate \( M^e \pmod{N} \) by repeatedly multiplying by \( M \pmod{N} \). The resulting sequence of
intermediate products consists of numbers that are smaller than \( N \), and so the individual multiplications do not take too long.

\[
M \mod N \rightarrow M^2 \mod N \rightarrow M^4 \mod N \rightarrow \cdots \rightarrow M^e \mod N
\]

But the problem arises when the value of \( e \) is 500 bits long and more, there it is necessary to compute \( e-1=2^{500} \) multiplications [21]. This reveals that algorithm4 is clearly exponential in the size of \( e \). With the slight modification of (i), the better step we can define is, starting with \( M \) and squaring repeatedly modulo \( N \), we get

\[
M \mod N \rightarrow M^2 \mod N \rightarrow M^4 \mod N \rightarrow M^8 \mod N \rightarrow \cdots \rightarrow M^{2^{\log e}} \mod N
\]

Where each steps holds just \( O(\log^2 N) \) time to compute, and in this case there are only \( \log e \) multiplications. To determine \( M^e \mod N \), we simply multiply together an appropriate subset of these powers, those corresponding to 1’s in the binary representation of \( e \). For instance [21],

\[
\begin{align*}
M^3 & = M^{10012} = M^{00002} \cdot M^{00002} \cdot M^{12} \\
& = M^{16} \cdot M^8 \cdot M^1 \quad \cdots \cdots (iii)
\end{align*}
\]

In doing so, it closely parallels our recursive Multiplication algorithm (Algorithm 5) [21]. For instance, the algorithm would compute the product \( M \cdot 25 \) by an analogous decomposition. According to the equation (iii), \( M \cdot 25 \) can be written as \( M \cdot 16 + M \cdot 8 + M \cdot 1 \) and whereas for the terms of kind \( M \cdot 2^i \), multiplication come from repeated doubling, for exponentiation the corresponding terms \((M^2)^y \) are generated by repeated squaring. Let \( n \) be the size in bits of \( M, e, \) and \( N \) (whichever is largest of the three). As with multiplication, the algorithm will halt after at most \( n \) recursive calls, and during each call it multiplies \( n \)-bit numbers (doing computation modulo \( N \) saves us here), for a total running time of \( O(n^2) \) [21].

**RESULTS AND DISCUSSION**

As far as the results section is concerned, it deals with the Comparisons between the RSA and RRSA (Proposed one) particularly in terms of increased size of \( N \), the Factoring modulus. The analysis made in the table3 describes that, if the number of digits in \( N \) is 234 then \( p \) and \( q \) can have 117 decimal digits each in RSA. On the other hand \( N \) in RRSA can have 110 decimal digits for \( p \) and \( q \) each and 14 decimal digits for \( r \) which could go beyond 322 decimal digits. The encryption and decryption times would be comparatively less in RRSA because of the reduced strength multiplication and modular operations involved in calculating \( M^e \mod N \) as per algorithms 4 and 5. It clearly reveals that the size of \( N \) in RRSA is being enhanced by using the Armstrong prime \( r \) even with lesser digits of prime numbers and factoring \( N \) become impractical.

**CONCLUSION**

The concept of cryptography is playing a vital role in the secure data transmission, we have gone through the RSA cryptosystem along with its strengths and weakness. In this paper authors have proposed RRSA which primarily focusses not only in analyzing the variable \( N \) in making the factorization process more complex or making it not factored but also speed up the encryption process. The values considered for the variables \( p, q, r \) helped in increasing the size of \( N \), so the time taken to factor the value \( N \) had been increased. The algorithms 4 and 5 have supported in calculating \( M^e \mod N \) (even with larger number of digits in \( M \) and \( N \)) at a faster rate, so that the time taken for encryption is comparatively less with respect to RSA. The key advantage characteristics considered for this cryptosystem are extreme secure, energy efficient, more power and relatively fast.

**REFERENCES**


[7]. Amar Kumar Mohapatra, Dr.Nupur Prakash, ‘Step-wise Calculation of Performance and Complexity Analysis of Safer with RSA Algorithm’ University School of Information Technology.


[10]. Prof. Dr. Alaa Hussein Al-Hamami and Ibrahim Abdallah Al-Dairish, ‘Enhanced Method for RSACryptosystem Algorithm’, Department of Computer Science College of Computer Sciences and Informatics Amman Arab University.


[20]. RSA Numbers factored were identified through the web link https://en.wikipedia.org/wiki/RSA_numbers#RSA-232 accessed on 06 December 2017.