CHAPTER III

RENOVATED RSA CRYPTOSYSTEM

3.1 INTRODUCTION

As far as the information security is concerned, cryptography stood at the top position. In Cryptography, Cryptosystem is a suite of ‘3’ algorithms viz., Generation of Keys, process of Enciphering and processes of Deciphering. This chapter describes the Renovated RSA (RRSA) cryptosystem via 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. The 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 \(e, d, N\).

3.2 ARCHITECTURE OF THE ASYMMETRIC KEY CRYPTOSYSTEM

The architecture of the Cryptosystem depicts the components like Plaintext, Cipher text, Secret information stores said to be keys, procedures like Encryption and Decryption.

Plain-text: The ordinary text that is being considered as human readable format is termed to be as plain-text.

Cipher text: The text that is computationally tagged with a few mathematical operations and written in coded form is said to be cipher text.

Encryption Procedure: The procedure of converting plain content to cipher content making the use of appropriate keys is said as “Encryption”
**Decryption Procedure**: The procedure of converting cipher content to plain content making the use of appropriate keys is said as “Decryption”

**Keys**: The keys in practice can be represented as a common component between source and destination in order to maintain the private information link which is being used in the cryptographic operation.

The Asymmetric key cryptography uses two un-identical keys i.e., viz., enciphering and deciphering process respectively. Precisely speaking enciphering and deciphering are most common synonyms of encryption and decryption respectively.

![Architecture of the Asymmetric Key Cryptosystem](image)

**Fig 3.1: Architecture of the Asymmetric Key Cryptosystem**

**3.3 THE RSA ALGORITHM**

The vital element of RSA public-key cryptosystem is that the “enciphering” and “deciphering” procedures are through two unlike keys – “public key” and “private key” respectively. Its safety is based on the issues like “struggle in factorizing the large prime”, which is an eminent mathematical tricky that has no operational solution [32]. The following fig 3.2 gives the result of RSA [20, 33].
3.2 Step-by-Step execution of RSA Algorithm

3.4 ATTACKS ON RSA

The saying "A series is no tougher than its fragile connection" is exact apt for telling attacks on cryptosystems. Most of the assailants’ character is to go for the fragile link of the series which includes managing the key or its generation, the cryptographic algorithm and its set of rules. The consequent sections briefly describe the attacks on RSA and the factored values of RSA-N.

3.4.1 Probing the Message Space

One of the appearing failings of RSA is that the procedure that encrypts the messages is openly available. The maximum number of plain-text messages that can be obtained with the 9 English characters in the message space is $26^9$ i.e., 5429503678976. Let us assume that encryption using “RSA-1024” took 0.18 milliseconds and then the time to cycle
through all the possible messages on one PC is about 434360294318.08 milliseconds, or 5027 days [34].

**Resolution**

The simple resolution to overcome this attack is to send the large messages preferably the size greater than 18 characters [34], which in turn the time to cycle all through the plain-text messages grows exponentially. Further this would become an insurmountable task for the intruder to crack the original message.

### 3.4.2 Common modulus

In common modulus attack, the intruder detects the plain-text message without factoring ‘N’ or discovering the decryption component ‘d’ [34]. Visualize a situation where John would like to drive the message ‘M’ to Alex and Bobby independently. To do this Alex provides John her public key \((N, e_1)\) and Bobby contributes John his public key \((N, e_2)\) wherever \(e_1\) and \(e_2\) are relatively prime. As Alex and Bobby are using the matching modulus ‘N’ can lead to a difficulty. John pushes \(C_1 = M^{e_1} \pmod{N}\) and \(C_2 = M^{e_2} \pmod{N}\) to Alex and Bobby respectively. Now assume that \(C_1\) and \(C_2\) are being intercepted by an eavesdropper “Eve”, by means of \(\gcd (e_1, e_2) = 1\), he can come to know the integers \(x\) and \(y\) using one of the algorithm statements proposed by Euclidean i.e., \(1 = e_1x + e_2y\). Precisely one of \(x\) or \(y\) will be negative, if we assume \(x\) is negative. Eve can now calculate

\[
(C_1^{-1})^x (C_2) = C_1^x C_2^y
= (M^{e_1})^x (M^{e_2})^y
= M^{e_1x + e_2y}
= M^d
= M \pmod{N}
\]

**Fig 3.3: Common Modulus attack on RSA**

Thus, Eve can sense plain-text message neither by factoring \(N\) nor uncovering the decryption exponent \(d\) [34].
Resolution

With the aim of making the RSA free from this attack, two distinct users in the identical channel of communication ought to have unique modulus value \( N \).

3.4.3 Low exponent value for \( e \).

Most of the times RSA cryptosystem uses the lower exponent value viz., \( e=3 \) for making the encryption quicker. Still, there is a weakness that if the equivalent message is enciphered 3 times with unlike keys i.e., similar exponent with dissimilar moduli then the intruder can regain the message [35].

Resolution

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.

3.4.4 Description of “Man-In-The-Middle” Attacks

It is a kind of active attack where the assailant furtively relays and perhaps amends the conversation or keys between the two entities who believe they are directly communicating with each other [36]. The Mail-Man can hear our chats and collect the information what we reveal. With the aim of having control over the channel of communication, he could also try to fabricate any information. The following Fig 3.4 depicts this attack.

Resolution

To avoid this attack, trusted third party individual is being used as key distributor. In this picture Alex keep faith on Trent as a key supplier and Trent will safeguard that the key will be delivered to Bobby and not the Eve. Fig 3.5 clearly depicts the avoidance of this attack.
3.4.5 Factorization Attacks

“Factoring assaults” on RSA are alluded to as “brute-force” assaults. These assaults are dependent on “factorizing the modulus $N$” into its distinct “prime factors”. Figuring $\phi (N) = (p-1)(q-1)$ is a natural task, if $N$ is factored. Further, solving $gcd (e, \phi (N)) = 1$ and $ed \equiv 1 \pmod{\phi (N)}$ let the intruder to catch the exponent $e$ and $d$ respectively. It clearly depicts that the size of ‘$N$’ is directly proportional to the factorization time of ‘$N$’ [37].
Popular Factoring algorithms

a) Method of Factoring through Fermat’s theorem

Fermat’s strategy factors a number ‘$N$’ by composing it as a difference of squares, $N = x^2 - a^2$ Where $x, a \in \mathbb{Z}$. Then $N = (x + a)(x - a)$. If $(x + a)$ or $(x - a)$ are not prime values, then Fermat’s strategy can be recurrent with those values. For example $N = 1121$, the Fermat’s method starts by considering $x$ as square root of $N$ i.e., $x \approx 34$. Now construct the table 3.1 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>
<tr>
<td>39</td>
<td>400</td>
</tr>
</tbody>
</table>

TABLE 3.1: RSA-Factoring: $x$ increments till $x^2 - N$ is a perfect square.

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

b) Factorization using “Quadratic Number Field Sieve” (QNFS)

The QNFS is treated as quickest algorithm for factoring numbers around 50-100 decimal digits. It is an augmented version of Fermat’s strategy. Fermat’s method can take a very long time as per the calculations shown in table 3.1. Rather the QNFS just attempts $x$ esteems that are viewed as smooth. A number that has just little prime factors is termed as smooth number [37]. The steps mentioned in the fig 3.6 gives the result of QNFS.
c) Factorization using “General Number Field Sieve” (GNFS)

The GNFS is treated as quickest algorithm for factoring the numbers not less than 100 decimal digits. The productivity of GNFS is that the major RSA modulus that was fruitfully factored was ‘\(N\)’ with 232 decimal digits [38].

Resolution: The modest solution to overwhelm this attack is “upturn the size or digits of \(N\)”, which showed us a path to renovate the existing RSA.

3.4.6 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 [39]. The following table 3.2 describes it with a step increment of 10.
<table>
<thead>
<tr>
<th>S.No</th>
<th>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>
</tr>
<tr>
<td>2</td>
<td>35794234..........17568667</td>
<td>110</td>
<td>364</td>
</tr>
<tr>
<td>3</td>
<td>22701048...........96548479</td>
<td>120</td>
<td>397</td>
</tr>
<tr>
<td>4</td>
<td>11438162...........79543541</td>
<td>129</td>
<td>426</td>
</tr>
<tr>
<td>5</td>
<td>18070820...........14880557</td>
<td>130</td>
<td>430</td>
</tr>
<tr>
<td>6</td>
<td>21290246...........41936471</td>
<td>140</td>
<td>463</td>
</tr>
<tr>
<td>7</td>
<td>15508981...........95964683</td>
<td>150</td>
<td>496</td>
</tr>
<tr>
<td>8</td>
<td>10941738...........54333897</td>
<td>155</td>
<td>512</td>
</tr>
<tr>
<td>9</td>
<td>21527411...........70407753</td>
<td>160</td>
<td>530</td>
</tr>
<tr>
<td>10</td>
<td>26062623...........11545759</td>
<td>170</td>
<td>563</td>
</tr>
<tr>
<td>11</td>
<td>18819881...........50257059</td>
<td>174</td>
<td>576</td>
</tr>
<tr>
<td>12</td>
<td>19114792...........51421041</td>
<td>180</td>
<td>596</td>
</tr>
<tr>
<td>13</td>
<td>19075564...........01423481</td>
<td>190</td>
<td>629</td>
</tr>
<tr>
<td>14</td>
<td>27997833...........21823983</td>
<td>200</td>
<td>640</td>
</tr>
<tr>
<td>15</td>
<td>24524664...........70551067</td>
<td>210</td>
<td>663</td>
</tr>
<tr>
<td>16</td>
<td>74037563...........63796359</td>
<td>212</td>
<td>704</td>
</tr>
<tr>
<td>17</td>
<td>22601385...........96955261</td>
<td>220</td>
<td>729</td>
</tr>
<tr>
<td>18</td>
<td>12301866...........02143413</td>
<td>232</td>
<td>768</td>
</tr>
</tbody>
</table>

**TABLE 3.2: A few ‘RSA-N values with their decimal digits’ that are factored.**
3.5 THE 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 $N$ via using “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 $n^{th}$ powers of individual digits is equal to that number and it should be prime, where $n$ is number of digits in given number. The number that is divisible only by itself and 1 is named as “prime number”. The primary asset of the “RSA” relies on the two “prime values” viz., ‘$p$’ and ‘$q$’. The algorithms regarding factorization of ‘$N$’ viz., Fermat’s, QNFS, GNFS may give clue to achieve initial values like ‘$p$’ and ‘$q$’. It is considerably less demanding to discover set of two values from factoring ‘$N$’ than calculating the sequence of ‘3’ numbers from ‘$N$’. By the way our proposed cryptosystem gives an insurmountable task for the trespasser to find the series of ‘3’ values $p, q, r$ from factoring $N$.

3.5.1 Generation of Keys

- Choose two distinct Prime numbers $p, q$ and Armstrong Prime number $r$.
- Find $N$ such that $N = p*q*r$. [$n$ will be used as the modulus for both the public and private keys.]
- Find the Phi of $N$, $\phi(N) = (p-1)*(q-1)*(r-1)$.
- 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 $d*e \equiv 1 (mod \ \phi(N))$. $d$ is kept as private key component.

Fig: 3.6 Step-by-Step execution of Key Generation-RRSA
3.5.2 Encryption

The procedure involved in encoding a message with appropriate set of keys in such a method that only approved users can have right to use it. The Encrypted or encoded text is also called Cipher text \((C)\) whereas the original message is called Plain-text \((M)\). The Encryption process involved in the algorithm is specified as follows.

\[ C = M^e \mod N. \]

3.5.3 Decoding

The process of un-encrypting the message manually or using proper codes in such a way that the human/computer can understood the message termed as Decryption. The Decryption process involved in the algorithm is specified as

\[ M = C^d \mod N. \]

3.5.4 Number theory

As per the algorithm 3.3 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 \(2^{74,207,281}-1\), a number having “22,338,618” digits. Further we did a search for Armstrong prime numbers and identified are as follows 2, 3, 5,7,28116440335967,449177399146038697307 and 35452590104031691935943. The largest Armstrong number identified is of 39 decimal digits.

3.6 THEORETICAL 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 considered for justification.
Given positive numbers $N$, $e$, and $d$ with the end goal that [40]: $N = p*q*r$, where $p$, $q$ are distinctive primes, $r$ is an Armstrong prime, $e$ and $d$ speaks to public and private keys individually………………………………………………..(i)

\[ \text{“gcd}(e, \varphi(N))=\]………………………………………………………………………………..(ii).

\[ ed\equiv 1(\text{mod}\varphi(N)) \]……………………………………………………………………..(iii).

Characterize the general shared and secret key changes of a message $S$ to be individually, is as follows:

RRSA Public($S$)$= S^e \text{ mod } N$ ………………………………………………………(iv).

RRSA Private ($S$) $= S^d \text{ mod } N$……………………………………………………………………..(v).

$S$ = RRSA Private (RRSA Public ($S$)), and that …………………………………………..(vi).

$S$ = RRSA Public (RRSA Private ($S$))………………… ………………………………..(vii).

Thinking about the conditions (vi) and (vii), in the event that we can demonstrate that they can be utilized conversely to acquire the message ‘$S$’, we can state that the RRSA encryption is consummately working.

Having the equations (iv) and (v) used in (vi) and (vii) respectively, the following equations holds good [40].

RRSA Private (RRSA Public ($S$)) = ($S^e \text{ mod } N$)$^d \text{ mod } N = S^{de \text{ mod } N}$ and further more

RRSA Public (RRSA Private ($S$)) = ($S^d \text{ mod } N$)$^e \text{ mod } N = S^{de \text{ mod } N}$.

Accordingly, equations (vi) and (vii) are equivalent, or

RRSA Private (RRSA Public ($S$)) = RRSA Public (RRSA Private ($S$)).

On the off chance that we can demonstrate: $S = S^{de \text{ mod } N}$, at that point the evidence will be finished [40]. It is mentioned that:
\[ de \equiv 1 \pmod{\phi(N)} \] ................................................................. (iii).

**By the meaning of mods**

*mod* as a congruence relation: The symbolization ‘\( a \equiv b \pmod{n} \)’ means \( a \) and \( b \) have the same residue when divided by \( n \), or, equivalently,

(a) \( n \mid a - b \), or

(b) \( a - b = nk \) designed for certain integer \( k \) hence we can mark (3) as \( \phi(N) \mid de - 1 \) (viii).

From the time when \( \phi(N) = \phi(p) \phi(q) \phi(r) \) happens, only when \( p, q \) and \( r \) are relatively prime, as for this situation, we have been using \( \phi(N) = \phi(p) \phi(q) \phi(r) \) since longer times and now by having it into (viii) we claim \( \phi(p) \phi(q) \phi(r) \mid de - 1 \) [40].

By characteristics of divisors, the notation \( n \mid a \) means \( n \) divides \( a \) and if \( sn \mid a \) then \( s \mid a \) and \( n \mid a \) for any integers \( s, n \), therefore we claim as [40].

\[ \phi(p) \mid de - 1 \]

\[ \phi(q) \mid de - 1 \]

\[ \phi(r) \mid de - 1 \]

Here we need to ensure that ‘\( k \)’ must be an integer \( k \) with an end goal that: \( de - 1 = k \phi(p) \).

From the time when \( p \) is prime value, the “*Euler phi function*” states that [40],

\[ \phi(p) = p - 1 \], therefore \( de - 1 = k (p - 1) \) ................................................................. (ix)

Observing the modular symmetric properties, we can compose

\[ S^{de} = S^{de \pmod{p}} = S^{de - 1 + 1 \pmod{p}} \]

On the other side, it can be claimed as

\[ S^{de} = (S^{de - 1}) * S \pmod{p} \] ................................................................. (x).
Having the equation (ix) used in (x), we acquire

\[ S^{de} \equiv (S^{k(p-1)}) \ast S \pmod{p} \]  

(xi).

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

i. Relatively prime to \( p \)

ii. A multiple of \( p \).

As soon as

i) \( S \) is relatively prime to \( p \), “Fermat’s Little Theorem” claims that \([40]: S^{p-1} \equiv 1 \pmod{p} \).

The mod property says that:

\[ S^{k(p-1)} \equiv 1 \pmod{p} \], or

\[ S^{k(p-1)} \equiv 1 \pmod{p} \]  

(xii).

By merging (xi) and (xii), we achieve:

\[ S^{de} \equiv 1 \ast s \pmod{p} \], or

\[ S^{de} \equiv S \pmod{p} \]  

(xiii).

In the subsequent situation where

\( S \) is a multiple of \( p \), if \( p \mid S \), then for any integer \( kp \mid S^k \)

According to the property of mod we can mark: \( S^{de} \equiv 0 \pmod{p} \), \( S \equiv 0 \pmod{p} \).

Thus we can carve: \( S^{de} \equiv S \pmod{p} \). Hence, for all that \( S \), \( S^{de} \equiv S \pmod{p} \) and put on the same procedure for \( q \) and \( r \) we can carve:

\[ S^{de} \equiv S \pmod{q} \] and

\[ S^{de} \equiv S \pmod{r} \]  

(xiv).
As soon as \( s \) and \( n \) are relatively prime \( a \equiv b \ (mod \ s) \), and \( a \equiv b \ (mod \ n) \), then \( a \equiv b \ (mod \ sn) \), we can carve (xiv) as \( S^{de} \equiv S \ (mod \ pq) \equiv S \ (mod \ N) \) as per the modular characteristic of congruence.

Through the modular characteristic of symmetry, we can transcribe

\[
S \equiv S^{de} \ (mod \ N) \quad \text{……………………………………………………………………………… } (xv)
\]

From the time when we have limited \( S \) as \( 0 \leq S < N \), only one integer will gratify (xv), and therefore \( S = S^{de \ mod\ N} \quad \text{……………………………………………………………………………… } (xvi) \)

Having equation (xvi) be used in our initial equations

We acquire: RRSA Private (RRSA Public (\( S \))) = \( S^{de \ mod\ N} \) and RRSA Public (RRSA Private (\( S \))) = \( S^{de \ mod\ N} \)

We attain, for \( 0 \leq S < N \),

RRSA Private (RRSA Public (\( S \))) = \( S \) & RRSA Public (RRSA Private (\( S \))) = \( S'' \).

Henceforth the verification has finished.

3.7 STRENGTH OF THE RRSA CRYPTOSYSTEM – ANALYSIS

The strength of the cryptosystem majorly relies on procedure involved in generating the keys. 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 3.4.6 has revealed that \( N=p\times q \) with 232 decimal digits has factored till now. As our proposed system uses \( N = p\times q\times r \), 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 interloper to discover 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 3.3 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 takes an infinite amount of time to find factors of $N$.

As per the user point of view calculating $C=M^e \mod N$ with more than 232 decimal digits in $N$ is a tough task, in this regard the following algorithm 3.4 and 3.5 supports the RRSA cryptosystem to enhance the encryption by performing the ‘large number mod $N$’ at a faster rate. In our cryptosystem, it is essential to work out $M^e \mod N$ for values of $M$, $e$ and $N$ that are numerous hundred bits long.

The notations used in the table 3.3 are as follows

\[
\begin{align*}
n(p) & : \text{No. of digits in } p \\
n(q) & : \text{No. of digits in } q \\
n(r) & : \text{No. of digits in } r \\
Max(N) & : \text{Maximum no. of digits that can be present in } N. \\
Min(N) & : \text{Minimum no. of digits that can be present in } N
\end{align*}
\]
Generalized formula to find the Minimum and Maximum number digits in \( N \) with \( p, q, r \).

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

<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>

**TABLE 3.3:** Minimum and Maximum No. of digits that can be present in \( N \) with \( p, q, \) and \( r \)
Nevertheless, the fresh approximate value of $M^e$ could be a whole lot longer than the above mentioned values. Although when $M$ and $e$ are only 20-bit numbers, $M^e$ is somewhere around $2^{19}$ (524288), about 10 million bits in length! Envision what occurs if $e$ is a 500-bit number! To ensure that the tasks on these numbers are to be disentangled, we have to accomplish every halfway calculation of modulo $N$ [41]. The thought covered up is to compute $M^e \mod N$ by repeatedly multiplying with $M \mod N$. The subsequent arrangement of moderate items comprises of numbers that are littler than $N$, thus the individual augmentations don't take too long.

\[ M \mod N \rightarrow M^2 \mod N \rightarrow M^3 \mod N \rightarrow \cdots \rightarrow M^e \mod N \] \hspace{1cm} \text{(i)}

But the problem arises when the value of $e$ is 500 bits long and more, there it is necessary to compute $e-1 \approx 2^{500}$ multiplications [32]. This reveals that algorithm 6.4 is clearly exponential in the size of $e$. With the slight modification of (i), the better step we can define is, opening with $M$ and squaring recurrently modulo $N$, we acquire

\[ M \mod N \rightarrow M^2 \mod N \rightarrow M^4 \mod N \rightarrow M^5 \mod N \rightarrow \cdots \rightarrow M^{2^{\log e}} \mod N \] \hspace{1cm} \text{(ii)}

Where every progression holds just $O(\log^2 N)$ time to compute, and for this situation there are just $\log e$ product calculations. To decide $M^e \mod N$, we just multiply together a suitable subset of these powers, those comparing to 1’s in the binary portrayal of $e$. For example [41]

\[
M^{25} = M^{10012} = M^{100002} \cdot M^{1002} \cdot M^{12} = M^{16} \cdot M^8 \cdot M^1 \]

\hspace{1cm} \text{................................................................. (iii)}

In doing as such, it intently parallels our recursive Multiplication calculation (Fig 3.7) [41]. For example, the calculation would process the item $M \cdot 25$ by an analogous decomposition. As indicated in the equation (iii), $M \cdot 25$ can be composed as $M \cdot 16 + M \cdot 8 + M \cdot 1$ and while for the terms of kind $M \cdot 2^i$, augmentation originate from continued multiplying, for exponentiation the relating terms ($(M^2)^i$) are produced by continued squaring. Given ‘$n$’ is the size in bits of $M$, $e$, and $N$ (whichever is biggest of the three). Similarly as with augmentation, the calculation will stop after at most ‘$n$’ recursive calls, and amid each call it multiplies $n$-bit numbers (doing calculation modulo $N$ set aside here), for an aggregate running time of $O(n^3)$ [41].
3.8 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 table 3.3 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 232 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 shown in Fig 3.7. 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.
3.9 SUMMARY AND CONCLUSIONS

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 chapter, RRSA primarily focuses 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 3.4 and 3.5 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 period engaged for enciphering is comparatively less with respect to RSA.