AN ENHANCED CRYPTOGRAPHIC RSA ALGORITHM WITH TWO PRIMES, FOUR PRIMES AND CIRCULAR PRIMES

BG.Prasanthi [1], Associate professor, St. Joseph's university, Bangalore,

Siddharth Vijay Sai [2], Research scholar, Bangalore

BG Lakshmi [3], Associate professor, Surana College Autonomous, Bangalore

K.Smitha[4] , Research scholar, Bharatiyar University, Coimbatore.

ABSTRACT

In order to protect sensitive data, it is important to address data security concerns. One of the common methods for protecting data is cryptography, which is generally regarded as a vital part of data security and offers privacy, integrity, secrecy, and authentication. This paper proposes a strong data security technique that combines prime and circular primes for RSA algorithm. The integration boosts RSA's security to a higher-degree level. The message's ASCII values are encrypted using the circular primes , while the second and third levels are encrypted and decrypted using the conventional RSA. Circular primes inverse is used in the final step to re-decrypt the data. The integration aids in providing for the factorization issue with the conventional RSA is addressed by the integration. Three distinct algorithms—RSA with two primes, RSA with four primes, RSA with circular primes methods, and the suggested algorithm—were used in the comparative analysis. It has been demonstrated that the suggested algorithm takes less time to encrypt and decrypt small amounts of data than it does to encode and decrypt large amounts of data.

Keywords: Prime numbers, circular primes, ASCII values, Cryptographic algorithm RSA,

1. INTRODUCTION

Cryptography is the providing of protocol and methods required to secure a communication channel when a third party is presumed to be present .Symmetric and asymmetric keys are used in cryptographic algorithms. Only one key is needed for the encryption and decryption of data using symmetric algorithms. On the other hand, asymmetric algorithms need both public and private keys to encrypt and decrypt data. The public key is used to encrypt data while the receiver is the only one with access to the private key, which is used to decrypt the data.The RSA Algorithm is an asymmetric encryption algorithm that makesuse of prime numbers for the encryption process. Asymmetric meansthat it uses two different keys, a public key (for encryption) and a private key (for decryption). It was devised by Ron Rivest, Adi Shamirand Leonard Adleman. RSA derives its security from the difficulty of factoring large integers that are the product of two large prime numbers. Multiplying these two numbers is easy, but determining the original prime numbers from the total -- or *factoring* -- is considered infeasible due to the time it would take using even today's supercomputers.

2. METHODOLOGY

This section outlines our suggested approach for combining conventional RSA and Gaussian Interpolation formulas in order to improve data security. By tackling the RSA factorization problem, the Gaussian Forward and Backward Interpolation is combined with the standard RSA algorithm to increase security strength.

This algorithm was implemented in Python. It assumes that the original prime number is a circular prime. Once the number is provided, repeated right shifts are performed until all the circularprimes are obtained.

2.1 Proposed framework:

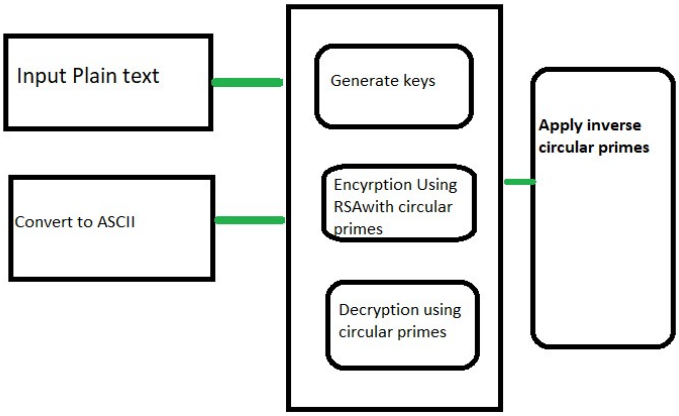


Figure 1:Flow diagram of proposed algorithm.

A general overview of the suggested algorithm is shown in Figure 1. The plaintext alphabets are translated into their appropriate ASCII values in the input portion. The ASCII values are subjected to Gaussian Forward Interpolation, which increases the encryption strength. The Ciphertext output is subsequently subjected to RSA. This entails the creation of keys, encryption, and decryption using the ASCII values as the basis. To get the plaintext, use Gaussian Backward Interpolation on the values from RSA that have been decoded.

2.2 Implementation

The source input is where data is accepted from the user as plain text. The individual charactersin the string are converted to ASCII and then to circular prime values. This converts the alphabets to numbers which is a form of encryption. For example, the ASCII value for 'A' is "065"

Algorithm 1: RSA algorithm using two primes

The keys for the RSA algorithm are generated in the following way:

1. Two large prime numbers, p and q are chosen.
2. The encryption modulus, n is calculated as the product of p andq.
3. The Euler's totient, ϕ(n) is calculated next as:

$$\phi(n) = (p-1) * (q-1)$$

4. An integer e is selected such that the $gcd(\phi(n), e) = 1$ where $1<e<\phi(n)$. This will be used to create the public key {e, n}.

5. Calculate d such that it satisfies the equation:
$de\ m(\phi(n)) = 1.$
This will be used to create the private key {d, n}.

6. Plaintext M is encrypted using the formula:
$$C = M^e \ mod\ n$$
M must be less than n.

7. The encrypted text can be decrypted with the help of a privatekey and the formula:
$$P = C^d \ mod\ n$$

2.3 Python code for the above Algorithm

```
# Accepting the two prime numbers, P and Q along with the plaintext M
P = int(input("Enter the first prime number: "))
```

```
Q = int(input("Enter the second prime number: "))
M = int(input("Enter the plaintext value: "))

# Calculating the modulus of encryption, n
n = P*Q

# Calculating phi_n, the Euler's totient
phi_n = (P-1)*(Q-1)

# Function to calculate gcd
def gcd(a,b):

        if (a == 0):

                return b
        if (b == 0):
                return a

        if (a == b):
                return a

        if (a > b):
                return gcd(a-b, b)
        return gcd(a, b-a)

# The public key e should be a value greater than 1 and less than phi_n. Therefore, I've initialised i to 2.
i = 2
# Running a while loop until a value for i is obtained such that the gcd of phi_n and i is 1
while True:

    # x is the gcd of phi_n and i
    x = gcd(phi_n,i)

    # if x has a value of 1, assign that value of i to the variable e.
    if x == 1:
        e = i
        break
    i+=1
# Printing the value of e
print("\ne is:",e)
# Next, we calculate the private key, d.
# For this, we choose values of j starting from 0 all the way to infinity until a value for j is obtained such
that d is a non decimal number.
# The formula used to calculate d is d = ((phi_n * j) + 1)/e
j = 0
# Running the while loop until  an appropriate value of j is obtained
while True:
    # num is the numerator
    num = ((phi_n*j)+1)
    d = num/e

    # To check whether the value of j is appropriate, if the modulus of num and e is 0, it means d is a
whole number, hence we break out of the loop
    if num%e==0:
        break

    # If the value of j isn't appropriate, incrememnt j and continue
    else:
        j+=1
```

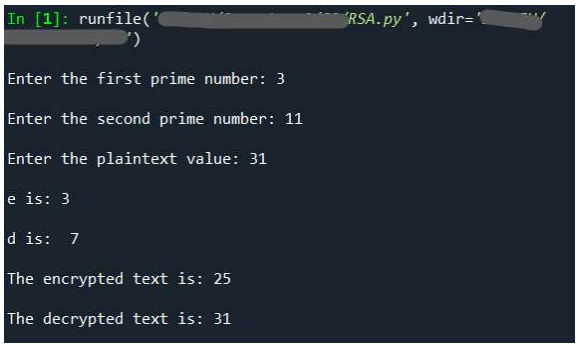
```
        continue

# d is of type float. If it isn't typecast to an integer, it will lead to an erroneous result when it is used later for decryption.
d = int(d)

# Print the value of the private key, d
print("\nd is: ",d)

# Calculate the cipher text, c and print it
c = pow(M,e)%n
print("\nThe encrypted text is:",c)

# Decipher the cipher text and print it
p = (c**d)%n
print("\nThe decrypted text is:",int(p))
```

Example

1. Consider p = 3 and q = 11
2. n = p x q = 33.
3. $\phi(n)$ = (p-1) x (q-1) = 20.
4. e = 3 using the formula gcd(e, $\phi(n)$) = 1(1<e< $\phi(n)$).

5.The private key, d = 7 using the formula d = [($\phi(n)$ x i) + 1] where $0 \leq i < \infty$ and the equation de mod($\phi(n)$) = 1 shouldbe satisfied.

6. The public key: KU = {e,n} = {3, 33}.
7. The private key: KR = {d,n} = {7, 33}.
8. Consider the plain text, M = 31. It is then encrypted usingthe formula: $C = M^e \bmod n = 31^3 \bmod 15 = 25$.
9.The text is decrypted using the formula: $P = C^d \bmod n = 25^7 \bmod 33 = 31$.



2.4 Python code for RSA using 4 prime numbers

```
# Accepting for prime numbers, P,Q,R,S along with the plaintext M
P = int(input("Enter the first prime number: "))
Q = int(input("Enter the second prime number: "))
R = int(input("Enter the third prime number: "))
S = int(input("Enter the fourth prime number: "))
M = int(input("Enter the plaintext value: "))

# Calculating the modulus of encryption, n
n = P*Q*R*S

# Calculating phi_n, the Euler's quotient
phi_n = (P-1)*(Q-1)*(R-1)*(S-1)

# Function to calculate gcd
def gcd(a,b):
        if (a == 0):
                return b
        if (b == 0):
                return a

        if (a == b):
                return a

        if (a > b):
                return gcd(a-b, b)
        return gcd(a, b-a)


# The public key e should be a value greater than 1 and less than phi_n. Therefore, I've initialised i to 2.
i = 2

# Running a while loop until a value for i is obtained such that the gcd of phi_n and i is 1
while True:
```

```
    # x is the gcd of phi_n and i
    x = gcd(phi_n,i)

    # if x has a value of 1, assign that value of i to the variable e.
    if x == 1:
        e = i
        break
    i+=1

# Printing the value of e
print("\ne is:",e)

# Next, we calculate the private key, d.
# For this, we choose values of j starting from 0 all the way to infinity until a value for j is obtained such that d is a non decimal number.
# The formula used to calculate d is d = ((phi_n * j) + 1)/e

j = 0

# Running the while loop until  an appropriate value of j is obtained

while True:

    # num is the numerator
    num = ((phi_n*j)+1)
    d = num/e

    # To check whether the value of j is appropriate, if the modulus of num and e is 0, it means d is a whole number, hence we break out of the loop
    if num%e==0:
        break

    # If the value of j isn't appropriate, incrememnt j and continue
    else:
        j+=1
        continue

# d is of type float. If it isn't typecast to an integer, it will lead to an erroneous result when it is used later for decryption.
d = int(d)

# Print the value of the private key, d
print("\nd is: ",d)

# Calculate the cipher text, c and print it
c = pow(M,e)%n
print("\nThe encrypted text is:",c)

# Decipher the cipher text and print it
p = (c**d)%n
print("\nThe decrypted text is:",int(p))
```

2.5 Algorithm 3: Proposed RSA algorithm using circular primes

RSA Algorithm wherein circular primes are used. The number of primes depends on the number of digits of the given circular prime. A circular prime is a prime number wherein performing a right shifton each digit of the number creates a new prime number. This is repeated until the initial number is obtained.

1:Take any two large prime numbers ,convert to circular prime.

2: Perform right shifts depending on the length of the numberand generate all the circular primes.

3: Use these primes to calculate n and ϕ(n)

4: An integer e is selected such that the $g(\phi(n), e) = 1$
where 1<e<ϕ(n). This will be used to create the public key {e,n}.

5:Calculate d such that it satisfies the equation:
$de\ m(\phi(n)) = 1.$
This will be used to create the private key {d, n}.

6:Plaintext M is encrypted using the formula:

$$C = M^e \ mod \ n$$

M must be less than n.

7:The encrypted text can be decrypted with the help of a privatekey and the formula:

$$P = C^d \ mod \ n$$

Taken the prime number converted to circular prime and the encryption and decryption are done.

Python code for the proposed enhanced RSA

```
# import math as m

n1 = input("Enter a prime number: ") #'31'
M = int(input("Enter the plaintext value: "))

l = []
for i in range(len(n1)):
   l.append(n1[i])
nums = [n1]
for i in range(len(n1)-1):
   l.append(l[0])
   l.pop(0)

   new_num = ""
   for i in l:
      new_num +=i
   print(new_num)
   nums.append(new_num)

n = 1
phi_n = 1
for i in nums:

   n*=int(i)
   phi_n *= int(i)-1

# Function to calculate gcd
def gcd(a,b):

            if (a == 0):

                        return b
            if (b == 0):
                        return a

            if (a == b):
                        return a
```

```
        if (a > b):
                return gcd(a-b, b)
        return gcd(a, b-a)


# The public key e should be a value greater than 1 and less than phi_n. Therefore, I've initialised i to 2.
i = 2

# Running a while loop until a value for i is obtained such that the gcd of phi_n and i is 1
while True:

    # x is the gcd of phi_n and i
    x = gcd(phi_n,i)
    # x = m.gcd(phi_n, i)

    # if x has a value of 1, assign that value of i to the variable e.
    if x == 1:
        e = i
        break
    i+=1

# Printing the value of e
print("\ne is:",e)
# Next, we calculate the private key, d.
# For this, we choose values of j starting from 0 all the way to infinity until a value for j is obtained such that d is a non decimal number.
# The formula used to calculate d is d = ((phi_n * j) + 1)/e
j = 0
# Running the while loop until  an appropriate value of j is obtained
while True:

    # num is the numerator
    num = ((phi_n*j)+1)
    d = num/e

    # To check whether the value of j is appropriate, if the modulus of num and e is 0, it means d is a whole number, hence we break out of the loop
    if num%e==0:
        break

    # If the value of j isn't appropriate, incrememnt j and continue
    else:
        j+=1
        continue

# d is of type float. If it isn't typecast to an integer, it will lead to an erroneous result when it is used later for decryption.
d = int(d)

# Print the value of the private key, d
print("\nd is: ",d)

# Calculate the cipher text, c and print it
c = pow(M,e)%n
print("\nThe encrypted text is:",c)

# Decipher the cipher text and print it
p = (c**d)%n
```

```
print("\nThe decrypted text is:",int(p))
```

Example:

1.Consider p = 13

2 . n = p x q = 33.

3. $\phi(n)$ = (p-1) x (q-1) = 20.

4.e = 3 using the formula gcd(e, $\phi(n)$) = 1(1<e< $\phi(n)$).

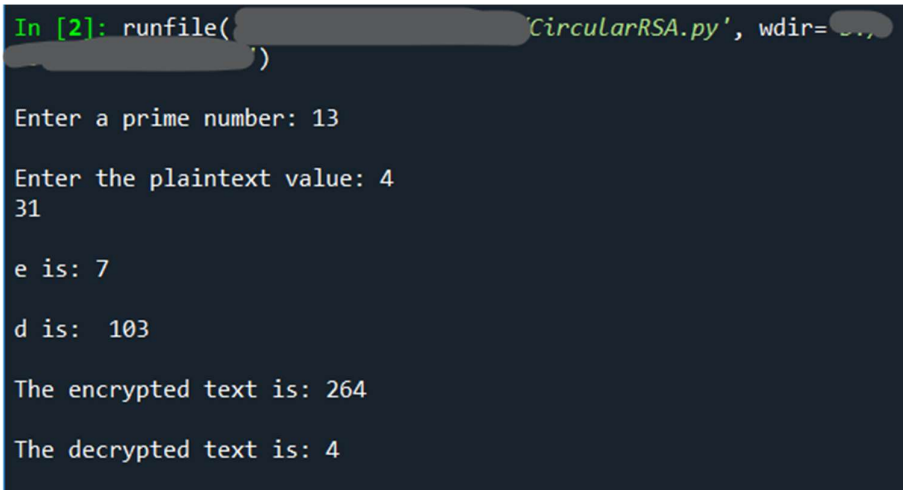
5.The private key, d = 7 using the formula d = [($\phi(n)$ x i) + 1] where $0 \leq i < \infty$ and the equation de mod($\phi(n)$) = 1 shouldbe satisfied.

6.The public key: KU = {e,n} = {3, 33}.

7.The private key: KR = {d,n} = {7, 33}.

8.Consider the plain text, M = 31. It is then encrypted usingthe formula: $C = M^e \bmod n = 31^3 \bmod 15 = 25$.

9.The text is decrypted using the formula: $P = C^d \bmod n = 25^7 \bmod 33 = 31$.



Circular prime RSA

```
In [3]: runfile( /RSA.py', wdir=

Enter the first prime number: 13

Enter the second prime number: 31

Enter the plaintext value: 4

e is: 7

d is:  103

The encrypted text is: 264

The decrypted text is: 4
```

Verification using original RSA

THE PROPOSED ALGORITHM'S RESULTS

The python programming language is used to implement the algorithm. The execution metrics and security strength are contrasted . The proposed algorithm combines traditional RSA with the circular primes . Because of this, the encryption and to the fifth level of decryption. Table 1 compares the time taken by three algorithms—RSA with two primes, RSA with four primes and the proposed algorithm. The comparison's encryption and decryption values are shown in the figure.

Table 1: Comparing the encryption with three algorithms.

Encryption			
Text Size	RSA with two prime	RSA with four primes	Proposed
5KB	72	105	72
10KB	148	209	110
15KB	217	314	200

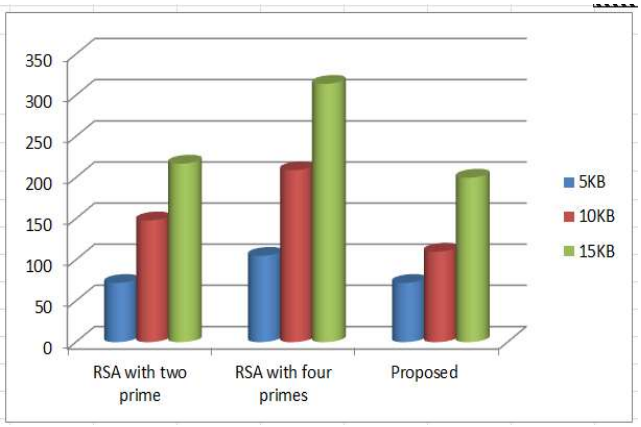


Table 2: Comparing the decryption with four algorithms.

Decryption			
Text Size	RSA	RSA using 4 prime	Proposed

5KB	260	490	442
10KB	578	989	588
15KB	728	1298	1116

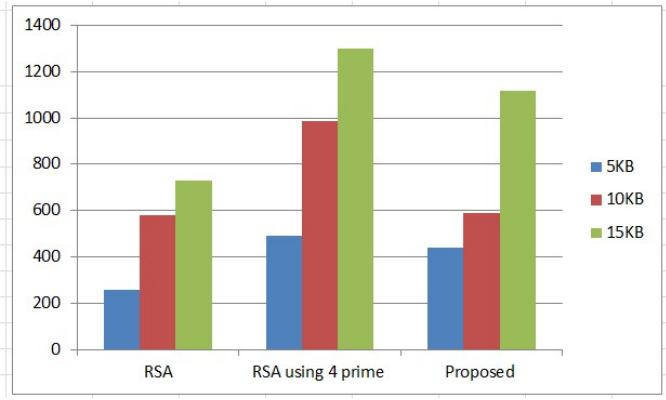


Table 1 shows that the suggested approach has the fastest encryption and decryption times for 5KB-sized data. However, the encryption time is longer than the 2-key size and SRNN but shorter than the conventional RSA techniques as the text size increased to 10 KB. Again, the suggested approach takes longer to decrypt data than RSA and SRNN, but less time than the 2-Key size algorithm. This is the result of collecting the ASCII values for each alphabet in the string and then interpolating the values using the circular primes. As a result, both the encryption and decryption operations take longer to compute.

4. CONCLUSION

The integration of the circular primes with the conventional RSA in this paper's upgraded RSA scheme increased the security of the latter. Additionally, it has raised the level of both encryption and decryption respectively. According to the study of the simulation, the execution time was less when the text size was small but longer when the text size was larger. This proposed algorithm has a stronger security strength than the traditional RSA and the time taken is also less compared to the conventional RSA algorithms with two and four primes.

REFERENCES

[1] P. Rewagad and Y. Pawar, "Use of Digital Signature with Diffie Hellman Key Exchange and AES Encryption Algorithm to Enhance Data Security in Cloud Computing," *2013 International Conference on Communication Systems and Network Technologies*, 2013, pp. 437-439, doi: 10.1109/CSNT.2013.97.

[2]A. Rawat, K. Sehgal, A. Tiwari, A. Sharma, and A. Joshi, "A novel accelerated implementationof RSA using parallel processing", *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 22, no. 2, pp. 309-322, 2019. Available: 10.1080/09720529.2019.1582864

[3] Dr.B.G.Prasanthi , " Enhanced Network Security Algorithm with Advanced key generation for RSA and Hashing" on 9/8/2019 at International conference on trends in Engineerin

[4] Dr.B.G.Prasanthi , " Contraptions to avoid object obstacles and increase the efficiency using Quad copters Drone Technology: How to Build Your Own Drone" @ IJIRSET,vol9,issue 4,

[5] Dr.B.G.Prasanthi ." Security analysis and portfolio management with effect of various market fluctuations " @ Jijnasa, **UGC Care**. ISSN : 0337-743X

[6] Dr.B.G.Prasanthi , " Energy efficient secure hybrid Bio inspired congestion control mechanism in wireless sensor networks @ International journal of computer science and engineering

[7] Dr.B.G.Prasanthi , " Design of Bio based approach in wireless sensor networks " @ International journal of computer science and engineering.

[8] Dr.B.G.Prasanthi . "Workload prediction in cloud environment during seasonal trend @ Gernze

International Journal of Engineering and Technology co-authored Smitha Krishnan,2021.

[9] Dr.B.G.Prasanthi , "The Importance of Training and Development in Organization :What matters in Practise" in International journal of Mechanical Engineering with ISSN: 0794-5823 on 4th March 2022,Published in IJCSE

[10] Dr.B.G.Prasanthi , "Intelligent parking system using Lifi technology and convolutional neural networks" with DOI number 10.37896/JSBV22.10/1490,vol22,Issue 10 2022 in "Journal of basic sciences" pageno196-200

[11] V. Rayward-Smith, T. Cormen, C. Leiserson, and R. Rivest, "Introduction to Algorithms", *TheJournal of the Operational Research Society*, vol. 42, no. 9, p. 816, 1991. Available: 10.2307/2583667.

[12]N. Kartha, "Review of the algorithm design manual, second edition by Steven S. Skiena", *ACMSIGACT News*, vol. 42, no. 4, pp. 29-31, 2011. Available: 10.1145/2078162.2078169.