DESIGNING AN ALGORITHM FOR SENDING ENCRYPTED MESSAGE BASED ON RSA

G.K. Sandhia¹, Akash S², Stephen J³

¹Computer Science and Engineering, SRM IST, Kattankulathur
²Computer Science and Engineering, SRM IST, Kattankulathur, (RA1711003010089)
³Computer Science and Engineering, SRM IST, Kattankulathur, (RA171100301010138)

ABSTRACT

The aim of our Project is to choose two random prime numbers from the sender’s message (by converting a string of characters into ASCII value and then calculating the nearest prime numbers to those values). After doing so, the message is encrypted and the encrypted message along with the public keys n(p*q), e(not a factor of those two primes) is sent to the receiver who then decrypts using these public keys by figuring out the factors using our own specialized factorization algorithm. The receiver then calculates e^d=1 mod Φ(n). This is then used to decrypt the message. In short instead of transferring the data twice in a usual RSA algorithm we transfer only once (highly efficient and faster encryption due to the distribution of load).

I. INTRODUCTION

There are two types of encryption i.e. symmetric and asymmetric. A single key is used in symmetric and that key is used for both encoding and decoding the message. In asymmetric encryption two keys are used that is public key and private key. Both the sender and the receiver will have public key but only the receiver will have the private key. One type of an algorithm that uses a symmetric encryption is RSA algorithm. Uses two prime numbers compute encryption key. It is created by three members Ron, Shamir, and Adleman in the year 1977. It is one of the toughest algorithms to crack still now it is impossible to hack into this algorithm. But it is quite a slow process because it takes time to compute the prime numbers.

II. LITERATURE SURVEY

In a paper published in 2010, the paper focuses on quickening the decryption process and signature of RSA algorithm. The performance of decryption and signature has a direct relationship with the effectiveness in implementing the modular exponentiation. This paper suggests a different variation of RSA cryptographic system (EAMRSA- Encrypt Assistant Multi Prime RSA) by decreasing the private exponents and modules in the modular exponentiation. The decryption process is significantly increased.

In a paper published in 2011, the paper states that RSA is a typical algorithm of the public key cryptographic system. Using large prime numbers is the factor of controlling this algorithm’s security, and to provide the large prime numbers. The design fitness function, crossover, and mutation strategies are the different methodologies that can be used in genetic algorithm, and finally designing an algorithm that can be able to produce large prime numbers.

In a paper published in 2019 in Cryptography and Network Security, the hash functions play a vital part. The hash algorithms are used to check the reliability and authenticity of information or data transferring between the Sender and receiver. The digital signature is also uses the hash functions and it is employed in many applications for digital signatures. Hash functions are used for generating keys in public and private key cryptographic systems. SHA-1, SHA-2, SHA3, MD4, MD5 and Whirlpool are the most familiar hash algorithms. This paper discusses the significance of hash functions and description about various familiar hash functions, and relative analysis of different hash algorithms.
III. ALGORITHMS USED:

1. Prime Number Detection Algorithm
2. Public Key and Message Encryption Algorithm
3. Public Key Decryption and Factorization Algorithm
4. Message Decryption Algorithm

PRIME NUMBER DETECTION ALGORITHM:

1. Choose a random string (str) of characters from the message
2. Convert that string(strint) into its corresponding ascii or unicode coding depending upon the programming language.
3. Calculate square root of strint i.e. \( \text{sqrtint} = \text{pow}(\text{strint}, (1/2)) \)
4. Create an array \( \text{array1} = \{2, 4, 2, 2\} \) and initialize \( k1 = 0, k2 = 0 \) and check if strint is divisible by 3 and 7.
5. For \( i = 11, i < \text{sqrtint}, i++ \)
   - If \( i \% \text{strint} == 0 \)
     - notprime;
     - break
   - Else
     - \( i = i + \text{array}[k] \)
     - \( k1++ \)
   If (notprime)
     - \( \text{strint} = \text{strint} + \text{array}[k] \)
     - \( k2++ \)
   Else
     - print(strint)

To check for two prime numbers, a random string of characters is chosen and then converted to its ascii value.

Check if strint is divisible by 3 and 7.

The square root of the ascii value is calculated and then from 11 it is incremented in the order from the array \( \{2, 4, 2, 2\} \)

This process is repeated until we get a couple of prime numbers.

Two primes are chosen with a minimum difference of about 20 to 30.

PUBLIC KEY AND MESSAGE ENCRYPTION ALGORITHM:

1. Calculate \( n = p * q \)
2. Choose \( e \) such that \( \text{gcd}(e, p-1) \neq 0 \)
3. The key is then used to encrypt the message and is then stored in \( \text{encrypt}[] \)
4. Encrypt message such that
5. \( \text{encrypt} = (\text{pow}(	ext{message}, e)) \% n \)
6. Generate two random numbers \( ra1, ra2 \) such that \( (ra2-ra1) > \text{len(public key)} \)
7. Hide the public key within these numbers. For example:
8. \( ra_1, ra_2, e \) \( \rightarrow \) *(MULTIPLY \( ra_1, ra_2 \) with \( e \))

9. \( ra_1, e, ra_2 \) \( \rightarrow \) % (MODULO OF \( ra_1, ra_2 \) with \( e \))

10. \( e, ra_1, ra_2 \) \( \rightarrow \) ^ (POWER OF \( ra_1, ra_2 \) with \( e \))

11. \( e, ra_2, ra_1 \) \( \rightarrow \) log (LOG OF \( ra_1, ra_2 \) with \( e \))

12. Here the algorithm itself can be modified based on the user’s needs. The above order is just an example.

13. Encrypt \( ra_1 \), \( ra_2 \) and then send the encrypted message (where the public key is hidden) encrypt1[, \( ra_1 \), \( ra_2 \) and \( e \) to the receiver.

PUBLIC KEY DECRYPTION AND FACTORIZATION ALGORITHM:

1. Decrypt \( ra_1 \), \( ra_2 \) and get the public key (\( n \))

2. Calculate \( \text{sqrn} = \text{pow} (n, (1/2)) \) and factorize \( n \) such that

3. Create an array \( \text{array1} = \{2, 2, 4, 2\} \) and initialize \( k1 = 0 \) and check if \( n \) is divisible by 3 and 7

   for(\( i = \text{sqrn} ; i > 11, i-- \))

   if (\( i \% \text{strint} == 0 \))

   return \( i \)

   break

   else

   \( i = i - \text{array}[k] \)

   \( k1++ \)

   \( p = n/i \)

   \( q = i \)

MESSAGE DECRYPTION ALGORITHM:

1. Calculate \( r = (p-1)*(q-1) \)

2. Calculate \( e*d = 1 \mod \Phi(n) \) such that \( d \) is an integer and not float

3. For larger numbers the Chinese remainder theorem can be used. One can use the below algorithm to simplify encrypt and \( d \).

\[
\begin{align*}
\text{dd1} &= \text{int}(d \% (p1-1)) \\
\text{dd2} &= \text{int}(d \% (q1-1)) \\
\text{enc1} &= [] \\
\text{enc2} &= [] \\
\text{mm1} &= [] \\
\text{mm2} &= [] \\
\text{i1} &= 0 \\
\text{encr1} &= \text{int()} \\
\text{encr2} &= \text{int()} \\
\text{mmm1} &= \text{int()} \\
\text{mmm2} &= \text{int()} 
\end{align*}
\]
while (i1<len(wordint)):
    encr1 = encrypt[i1] % p1
    encr2 = encrypt[i1] % q1
    enc1.append(encr1)
    enc2.append(encr2)
    mmm1 = pow(enc1[i1],dd1,p1)
    mmm2 = pow(enc2[i1],dd2,q1)
    mm1.append(mmm1)
    mm2.append(mmm2)
    i1=i1+1

Decrypt:
X1≡pow(mm1[i], p1)
X2≡pow(mm2[i], q1)
Solve congruency for for all mm1 and mm2.

IV. EXISTING SYSTEM:
In the RSA algorithm two prime numbers p and q are chosen randomly by the receiver in his computer, next the product of these two prime numbers which is n is calculated. This is the first part of the public key, now a small integer or exponent e must be chosen which should not be a factor of n. The receiver then sends this to who is to the sender (sends encrypted message) who uses this public key to compute the formula encrypted message = (mess)e mod n. The sender then sends encrypted message which was calculated by the above formula to the receiver who then receives it. In this time, the receiver calculates his private key. The private key is calculated by Φ(n) = (P-1) (Q-1), e*d=1 mod Φ(n) for some integer k. Now d is the private key. The receiver uses the private key to decrypt the message. The formula is given by decrypted message = (encrypted message) d mod n.

V. PROPOSED SYSTEM:
The proposed system of our project is almost similar to RSA but with a slight modification. Instead of the computer calculating the prime numbers from the receiver’s side, in our method the sender itself calculates the prime numbers randomly by selecting a a group of Strings from the message that is to be encrypted. He then converts those substrings into their corresponding ASCII values. The prime numbers closer to these ASCII values are chosen, if the ASCII value is not a prime number. The next step in the process is encrypting the messages by calculating (n=p*q) and e (which is not a factor of n). The sender then sends the encrypted message along with the private keys n and e to the receiver. The message is encrypted using the formula encrypted message = (mess)e mod n.

The receiver gets the n and e values along with encrypted message. The receiver then finds the factors of n using sieve of Eratosthenes algorithm. He then computes his private key by using the formula Φ(n) = (P-1) (Q-1), e*d=1 mod Φ(n) for some integer k. Receiver then decrypts the message using this private key using the formula Data = cd mod n.

VI. MODULES:

SENDER SIDE:
DATA IN:
The data that is to be encrypted is typed so that a random set of strings is chosen from them which will be converted to their corresponding ASCII value.

P,Q , E value:
Two random prime numbers that are close to the chosen set of ASCII values is calculated. A random number (E) is chosen from the same set of strings such that they are not factors of p and q.

PUBLIC KEY:
Here is where the generated public key is stored and encrypted so that hackers don’t hack into the message.
ENCRIPTION PROCESS:
Use the formula $(P^e \mod n)$ to encrypt the messages.

CONTROL:
All this is done on the sender side along with the distribution of public keys $n$, $e$. This is the module that controls all these things.

RECEIVER SIDE:
The receiver now has the encrypted message along with the $(n,e)$. The factors of $N$ are calculated so that we get $p$ and $q$. This is used so that we calculate $d$ on the receiver side $D=e*d \mod \pi(n)$.

MULTIPLICATION AND PRIVATE KEY MODULE:
This is where the computation takes place and the private key is stored.

DECRYPTION PROCESS:
In this module, the decryption of data takes place and so the decrypted original message is received.

VII. CONCLUSION:
From the above process, it can be concluded that the entire process time is reduced and the level of security is increased due to the increase in randomness.
REFERENCES:


5. IJCSMC, Vol. 8, Issue. 6, June 2019, pg.147 – 152 A Comparative Study of Hash Algorithms in Cryptography Prashant P. Pittalia Department of Computer Science, S. P. University, India JOURNAL.