RSA Cryptography

Submitted by

Arnish Satasiya (202001031)
Jay Kuvadiya (202001042)
Meet Sutariya (202001046)
Deep Kanani (202001454)

Submitted to

Prof. Manish K. Gupta

Prof. Rahul Muthu

Course: Discrete Mathematics

Course code: SC205

1 Introduction

- RSA is used for transmission of data (messages) securely .
- The RSA algorithm, is an algorithm for Public Key cryptography .
- In public-key cryptography, users reveal a public encryption key so that other users in the system are able to send private messages to them, but each user has their own private decryption key .
- The key to ensuring privacy in a public-key cryptosystem is for it to be extremely difficult to derive the decryption key from the publicly available encryption key .
- The algorithm works by exploiting concepts from Number Theory , including **Fermat's Little Theorem , Sieve Algorithm and Extended Euclid Algorithm**

2 Mathematical Problem

Nowadays Countries or any company need a way to communicate without anyone except receiver being able to decipher what exactly they are communicating. How the communication worked ?

The ans is that they started out by establishing a cipher that both parties would use, and from there one ally would use the numbers correlating to a certain number to create a message. Once the message was received from the other ally that group would use the cipher established to decode the message.

3 Solution

- Here, we have try to solve this problem using our Discrete Mathematics knowledge.
- Using the concept of RSA cryptography we can solve this real world problem.

How RSA works ?

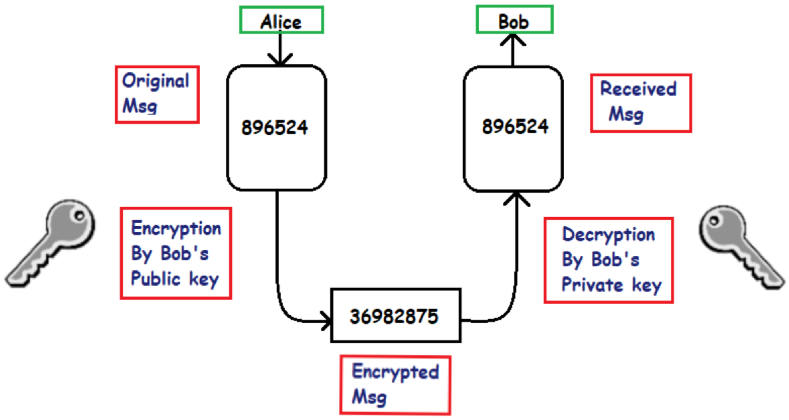
- RSA works on two different keys i.e. Public Key and Private Key.
- The Public Key is accessible by everyone and Private key is kept private.
- Here, we describe two methods to perform RSA .

3.1 Method 1 :

- If you want to send the message to one particular person , For that you have to encrypt your message using that person's public key n and e .

{ Here, n is multiplication of two Prime numbers (p and q) and e must be selected by satisfying this condition —> (gcd of (p - 1)*(q - 1) and e = 1) }

- After that , encrypted message will be sent to that particular person and that person will decrypt the message using his/her private key d . Person will get the exact message that you have sent.
- By using this method the message will be shown to the only one person whom you want to send the message.
- we can understand this method by the example given below , In this example... Alice want to send a number 896524 to bob .
- for that Alice have used the public key n and e of Bob and encrypted the message , encrypted message 36982875 is sent to Bob.
- Now , Bob will decrypt the message using his private key d and he'll get the original message 896524 that Alice has sent.



- To encrypt a message using the RSA algorithm, the general method of the encryption procedure is as follows:

Simple RSA

First of all select message (**M**) that you (here Alice) want to send someone (here Bob).
To encrypt the message , compute M^e mod n .

Cipher text **C** = M^e mod n   (where n and e is Bob's Public Key)

n = p*q (p and q are large Prime numbers)
gcd (n , (p-1)*(q-1)) = 1 .

–> Now , C will be sent to Bob .

To decrypt the C (encrypted message) Bob will do calculation that is given below :

D = (C^d) mod n (where d is Bob's Private Key)

where D is decrypted message .

$d{\cdot}e \equiv 1 \pmod{n}$

So, by this Bob will get message that Alice want to sent him .

Code (in c++) : (given in RSA1.cpp)

Limitations of this Code :

- This code is valid only if we want to encrypt integers , but we can't use this for the characters .
- We can encrypt or decrypt the number in a limited range , if the number is to large (more than 8 digits) this code we'll not work .
- In the real world scenario , RSA only based on numbers won't so helpful so this is the main limitations of this code.

Updated RSA

- For solving the limitations of Simple RSA code, we have written this code.
- In this code , We have take input as string . By using string , we can also encrypt and decrypt very large length of message .
- we can also encrypt or decrypt the characters , symbols , numbers etc.
- Like the previous code there are same two methods for encrypt the message , This code is very similar to the previous code .
- In this code , we'll encrypt the message by their ASCII value , we'll first make 2-2 pairs of the characters in the message , then we'll write their ASCII value and we'll take the public key n and e of the person whom you want to send the message for encrypting.
- For that first you should understand how we have used vector , we'll push back all the pairs in the vector .
- If the length of the string is odd then the last pair won't be made so we'll push back only that perticular character's ASCII value in the vector.
- Now , one by one we'll encrypt every numbers of the vector using n and e as shown in previous code.
- After encrypting the message , we will print all the elements in the vectors by ASCII values converting to the characters . By that we'll get the encrypted message.
- Now for decrypt this encrypted message the receiver have to insert the decryption private key d .
- By Inserting the correct decryption key , Message will be decrypted successfully and the only person whom you have sent the message able to see the correct message .

Code (in c++) : (refer RSAfinal.cpp)

Some Algorithms used in RSA

1. Sieve Algorithm

- We can get Prime numbers (p & q) from most powerful Sieve Algorithms : (refer Sieve.cpp)

> - The sieve of Eratosthenes is one of the most Powerful Algorithm to find all prime numbers smaller than N, even when N is very large number.

2. power function

- For understanding of pow() function : (refer power.cpp)

> - This algorithm is use for find the modulo of big numbers .
> - (N^M modulo P) , where N, M and P are large numbers .

3. Extended Euclidean Algorithm

- For Find Modular Multiplicative Inverse (for finding d) : (refer MMI.cpp)

- We have two integers a and m and we want to find inverse of a mod m .

- The multiplicative inverse of a modulo m exists if and only if a and m are relatively prime (gcd (a, m) = 1)

and satisfy this equation : $a \cdot x \equiv 1 \pmod{m}$

- { where , x is inverse of a mod m }
- x should be in range { 1, 2, ...upto m-1 }
- By Extended Euclidean Algorithm, For any two integers A and B, We can find integers s and t, such that:

  $s \cdot A + t \cdot B = \gcd(A, B)$

(s and t are know as Bezout Coefficients)

$s \cdot a + t \cdot m = 1$ { gcd (a , m) = 1 }
$(s \cdot a + t \cdot m) \bmod m = 1 \bmod m$
$s \cdot a \bmod m = 1 \bmod m$
$a \cdot s \equiv 1 \bmod m$

So, inverse of **a mod m** is **s.**