**PART 1:**

RSA Algorithm:

History:

RSA is one of the first practicable public-key cryptosystems and is widely used for secure data transmission. It is asymmetric key algorithm in which two different keys are used; one for encryption and the other for decryption. RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described the algorithm in 1977. Clifford Cocks, an English mathematician, had developed an equivalent system in 1973, but it wasn't declassified until 1997.

In RSA algorithm, two different keys public key and private key are mathematically linked with each other. The public key can be shared with everyone whereas private key must be kept secret. Anyone can use the one key to encrypt a message, then the other key may serve the purpose of decryption that is why it does provides a method of assuring the confidentiality, integrity, authenticity and non-reputability of electronic communications and data storage. It is secured based on the principle that it is difficult to factor the large integers that are product of two large prime numbers. [1]

Working:

1. Select two prime numbers p and q; that are random, distinct and are far away apart. The security of RSA algorithm lies on keeping these two prime numbers secret.
2. Now modulus n is generated from these two prime numbers by multiplying them with each other.
3. Then the totient of n, (n) is calculated as (n)=(p-1) \* (q-1).
4. Now the exponent e, of the public key is selected from the range [3, (n)) such that GCD(e, 𝜙(n))=1. The pair (e,n) is called the public key.
5. Now the exponent d, of the private key is multiplicative inverse of the public key exponent e, with (n). It can easily be calculated with the Extended Euclidean Algorithm. Here e\*d= (i\*(n))+1. So d= ((i\*(n))+1) / e. Now the value of the i is looped through 1 to until the remainder 0 is obtained in equation, remainder = ((i \* (n))+1) mod e. The pair (d,n) is called the private key.

* Encryption:

Encryption is done as follows: . Here the bigmod(M,e,n) is used as Matlab won’t be able to calculate for the large values of the powers.

* Decryption:

Decryption is done as follows:. Here also bigmod(M,d,n) is used as Matlab won’t be able to calculate for the large values of the powers.

* Demonstration:
* p=11 and q=13.
* n=p\*q=143.
* (n)=((p-1)\*(q-1))=120.
* e=7 because GCD[3,120)=1.
* d=103 from Extended Euclidean Algorithm.
* Encryption: ==132.
* Decryption: ==11.
* **Reference:**

[1] http://searchsecurity.techtarget.com/definition/RSA

**String to Ascii conversion function**

function ascii = str2ascii(s)

% Convert to ASCII

ascii = double(s);

end

**ASCII to String conversion function**

function str = ascii2str(ascii)

% Convert to string

str = char(ascii);

end

**Bigmod function**

function remainder = bigmod (number, power, modulo)

% modulo function for large numbers, -> number^power(mod modulo)

% by bennyboss / 2005-06-24 / Matlab 7

% I used algorithm from this webpage:

% http://www.disappearing-inc.com/ciphers/rsa.html

% binary decomposition

binary(1,1) = 1;

col = 2;

while ( binary(1, col-1) <= power-binary(1, col-1) )

binary(1, col) = 2\*binary(1, col-1);

col = col + 1;

end

% flip matrix

binary = fliplr(binary);

% extract binary decomposition from number

result = power;

cols = length(binary);

extracted\_binary = zeros(1, cols);

index = zeros(1, cols);

for ( col=1 : cols )

if( result-binary(1, col) > 0 )

result = result - binary(1, col);

extracted\_binary(1, col) = binary(1, col);

index(1, col) = col;

elseif ( result-binary(1, col) == 0 )

extracted\_binary(1, col) = binary(1, col);

index(1, col) = col;

break;

end

end

% flip matrix

binary = fliplr(binary);

% doubling the powers by squaring the numbers

cols2 = length(extracted\_binary);

rem\_sqr = zeros(1, cols);

rem\_sqr(1, 1) = mod(number^1, modulo);

if ( cols2 > 1 )

for ( col=2 : cols)

rem\_sqr(1, col) = mod(rem\_sqr(1, col-1)^2, modulo);

end

end

% flip matrix

rem\_sqr = fliplr(rem\_sqr);

% compute reminder

index = find(index);

remainder = rem\_sqr(1, index(1, 1));

cols = length(index);

for (col=2 : cols)

remainder = mod(remainder\*rem\_sqr(1, index(1, col)), modulo);

end

**Calculates the next prime number**

function a=cal\_next\_prime(p)

test=0;

%if the user by mistake entered composite number

%this function calculates the next prime number

while test == 0

%checks whether the number entered is prime or not

test=isprime(p);

if test == 0

p=p+1;

end

end

a=p;

end

**Calculates the value of e**

function e=cal\_e(p,q)

Phi\_n=((p-1)\*(q-1));

%Here t=2 because e is selected from range [3,Phi(n))

y=2;t=2;

%Calculate the value of e such that GCD(Phi,e)=1

while y > 1 && t < Phi\_n

t=t+1;

%gcd function calculates the gcd of two numbers

y=gcd(Phi\_n,t);

end

e=t;

end

**RSA function performs Encryption as well as Decryption**

function mc = rsa(M,N,key)

x=length(M);

c=zeros;

for i=1:x

%Encrypt or Decrypt using the bigmod function

c(i)=bigmod(M(i),key,N);

end

%Return the Encrypted or Decrypted array to rsa\_script.m

mc=c;

end

**genPrivKey function calculates modulus,public key and private key exponent and totient.**

function [n, d] = genPrivKey(p,q,e)

%cal\_next\_prime function calculates the next prime number

%if user by mistake inputs composite number

p=cal\_next\_prime(p);

q=cal\_next\_prime(q);

n=p\*q;

phi\_n=((p-1)\*(q-1));

%This function calculates the value of d using

%Extended Euclidean Algorithm

i=1;

r=1;

while r > 0

k=(phi\_n\*i)+1;

r=rem(k,e);

i=i+1;

end

d=k/e;

%Display the modulus,public key and private key exponent

%and totient

disp(['The value of modulus (n) is: ' num2str(n)]);

disp(['The exponent of public key (e) is: ' num2str(e)]);

disp(['The value of totient (Phi) is: ' num2str(phi\_n)]);

disp(['The exponent of private key (d)is: ' num2str(d)]);

end

**rsa\_script**

%PART 2 Decryption

p=11;

q=13;

%calculate the value of e

e=cal\_e(p,q);

%calculates private key exponent and modulus

[n,d]=genPrivKey(p,q,e);

%Code to be Decrypted

str=[89,59,33,98,67,40,39,98,69,62,59,29,98,77,108,110];

%Performs decryption by calling rsa function

M=rsa(str,n,d);

%displays decrypted ASCII

disp(['Decrypted ASCII of Message is: ' num2str(M)]);

%Convert to string from ASCII

str=ascii2str(M);

%Displays the decrypted String

disp(['Decrypted Message is: ' num2str(str)]);

%Code to be Decrypted

str=[124,45,45,100,98,35,40,66,98,20,24];

%Performs decryption by calling rsa function

M=rsa(str,n,d);

%displays decrypted ASCII

disp(['Decrypted ASCII of Message is: ' num2str(M)]);

%Convert to string from ASCII

str=ascii2str(M);

%Displays the decrypted String

disp(['Decrypted Message is: ' num2str(str)]);

%PART 3 Encryption

p=19;

q=17;

e=7;

%calculates private key exponent and modulus

[n,d]=genPrivKey(p,q,e);

%String to be encrypted

str='the rain in spain falls mainly on the plain';

%Convert string to ASCII

M=str2ascii(str);

%Displays ASCII of entered message

disp(['ASCII code of entered Message is: ' num2str(M)]);

%Encrypt using rsa function

M=rsa(M,n,e);

%Displays the Cipher text

disp(['Cipher text of entered Message is: ' num2str(M)]);

%String to be encrypted

str='Supercalifragilisticexpialidocious';

%Convert string to ASCII

M=str2ascii(str);

%Displays ASCII of entered message

disp(['ASCII code of entered Message is: ' num2str(M)]);

%Encrypt using rsa function

M=rsa(M,n,e);

%Displays the Cipher text

disp(['Cipher text of entered Message is: ' num2str(M)]);

**PART 2 output:**

The value of modulus (n) is: 143

The exponent of public key (e) is: 7

The value of totient (Phi) is: 120

The exponent of private key (d)is: 103

[1]

Decrypted ASCII of Message is: 67 97 110 32 89 79 117 32 82 101 97 68 32 77 69 33

Decrypted Message is: Can YOu ReaD ME!

[2]

Decrypted ASCII of Message is: 71 111 111 100 32 74 79 66 32 58 41

Decrypted Message is: Good JOB :)

**PART 3 output:**

The value of modulus (n) is: 323

The exponent of public key (e) is: 7

The value of totient (Phi) is: 288

The exponent of private key (d)is: 247

[1]

ASCII code of entered Message is: 116 104 101 32 114 97 105 110 32 105 110 32 115 112 97 105 110 32 102 97 108 108 115 32 109 97 105 110 108 121 32 111 110 32 116 104 101 32 112 108 97 105 110

Cipher text of entered Message is: 261 213 237 314 228 109 300 32 314 300 32 314 191 5 109 300 32 314 102 109 48 48 191 314 250 109 300 32 48 26 314 36 32 314 261 213 237 314 5 48 109 300 32

[2]

ASCII code of entered Message is: 83 117 112 101 114 99 97 108 105 102 114 97 103 105 108 105 115 116 105 99 101 120 112 105 97 108 105 100 111 99 105 111 117 115

Cipher text of entered Message is: 178 59 5 237 228 6 109 48 300 102 228 109 103 300 48 300 191 261 300 6 237 256 5 300 109 48 300 263 36 6 300 36 59 191