

Experiment 06

Write a Program to Implement and analyze RSA cryptosystem

Roll No.	08
Name	Akash
Class	D15-A
Subject	Security Lab
LO Mapped	LO1: To apply the knowledge of symmetric cryptography to implement classical ciphers. LO2 : Demonstrate Key management, distribution and user authentication.

Aim: Write a Program to Implement and analyze RSA cryptosystem

Introduction:

RSA algorithm is asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key**. As the name describes that the Public Key is given to everyone and Private key is kept private.

An example of asymmetric cryptography :

1. A client (for example browser) sends its public key to the server and requests for some data.
2. The server encrypts the data using client's public key and sends the encrypted data.
3. Client receives this data and decrypts it.

Since this is asymmetric, nobody else except browser can decrypt the data even if a third party has public key of browser.

The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024 bit keys could be broken in the near future. But till now it seems to be an infeasible task.

Let us learn the mechanism behind RSA algorithm :

>> Generating Public Key :

Select two prime no's. Suppose $P = 53$ and $Q = 59$.

Now First part of the Public key : $n = P * Q = 3127$.

We also need a small exponent say e :

But e Must be

An integer.

Not be a factor of n .

$1 < e < \Phi(n)$ [$\Phi(n)$ is discussed below],

Let us now consider it to be equal to 3.

Our Public Key is made of n and e

>> Generating Private Key :

We need to calculate $\Phi(n)$:

Such that $\Phi(n) = (P-1)(Q-1)$

so, $\Phi(n) = 3016$

Now calculate Private Key, d :

$d = (k * \Phi(n) + 1) / e$ for some integer k

For $k = 2$, value of d is 2011.

Now we are ready with our - Public Key ($n = 3127$ and $e = 3$) and Private Key($d = 2011$)

Now we will encrypt "**HI**" :

Convert letters to numbers : $H = 8$ and $I = 9$

Thus **Encrypted Data** $c = 89e \bmod n$.

Thus our Encrypted Data comes out to be 1394

Now we will decrypt **1394** :

Decrypted Data = $cd \bmod n$.

Thus our Encrypted Data comes out to be 89

8 = H and I = 9 i.e. "HI".

Code :

```
# Python for RSA asymmetric cryptographic algorithm.  
# For demonstration, values are  
# relatively small compared to practical application
```

```
import math
```

```
def gcd(a, h):  
    temp = 0  
    while(1):  
        temp = a % h
```

```
if (temp == 0):  
    return h  
a = h  
h = temp
```

```
p = 3  
q = 7  
n = p*q  
e = 2  
phi = (p-1)*(q-1)
```

```
while (e < phi):
```

```
    # e must be co-prime to phi and  
    # smaller than phi.  
    if(gcd(e, phi) == 1):  
        break  
    else:  
        e = e+1
```

```
# Private key (d stands for decrypt)  
# choosing d such that it satisfies  
#  $d * e = 1 + k * \text{totient}$ 
```

```
k = 2  
d = (1 + (k*phi))/e
```

```
# Message to be encrypted  
msg = 12.0
```

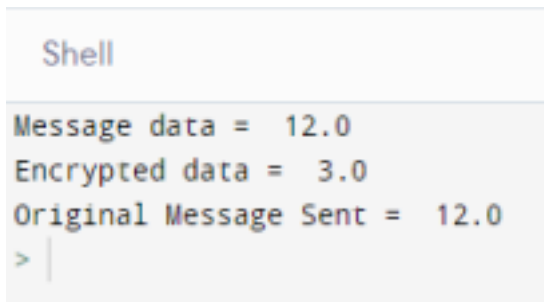
```
print("Message data = ", msg)
```

```
# Encryption  $c = (msg ^ e) \% n$ 
```

```
c = pow(msg, e)  
c = math.fmod(c, n)  
print("Encrypted data = ", c)
```

```
# Decryption  $m = (c ^ d) \% n$   
m = pow(c, d)  
m = math.fmod(m, n)  
print("Original Message Sent = ", m)
```

Results:



```
Shell  
Message data = 12.0  
Encrypted data = 3.0  
Original Message Sent = 12.0  
> |
```

Conclusion: We Implemented and analyzed RSA Algorithm in Cryptography.