Lab 3

Name: Yuvraj Singh
Roll No: 23072021
M.Tech CSE

RSA Algorithm.

In 1977, Ronald Rivest, Adi Shamir and Leonard Adleman (cf. Fig. 7.1) proposed a asymmetric cryptographic scheme RSA. There are many applications for RSA, but in practice it is most often used for: encryption of small pieces of data, especially for key transport digital signatures for digital certificates on the Internet. RSA algorithm is an 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 the Private key is kept private.

RSA Key Generation:

Output: Public key: $k_{pub} = (n, e)$ and private key: $k_{pr} = (d)$

1. Choose two large primes p and q.
2. Compute $n = p \cdot q$.
3. Compute $\Phi(n) = (p-1)(q-1)$.
4. Select the public exponent $e \in \{1, 2, \ldots, \Phi(n)-1\}$ such that $\gcd(e, \Phi(n)) = 1$.
5. Compute the private key d such that $d \cdot e \equiv 1 \pmod{\Phi(n)}$.

Encryption and Decryption: RSA encryption and decryption is done in the integer ring Zn. RSA encrypts plaintexts x, where we consider the bit string representing x to be an element in Zn = 0,1,...,n1. As a consequence the binary value of the plaintext x must be less than n. The same holds for the ciphertext. Encryption with the public key and decryption with the private key are as shown below:

RSA Encryption Given the public key $(n, e) = k_{pub}$ and the plaintext x, the encryption function is:

$$y = e_{k_{pub}}(x) \equiv x^e n$$

where $x, y \in Z_n$.

RSA Decryption Given the private key $d = k_{pr}$ and the ciphertext y, the decryption function is:

$$x = d_{k_{pr}}(y) \equiv y^d n$$

where $x, y \in Z_n$.

Alice		**Bob**
message $x = 4$		1. choose $p = 3$ and $q = 11$ 2. $n = p \cdot q = 33$ 3. $\Phi(n) = (3-1)(11-1) = 20$ 4. choose $e = 3$ 5. $d \equiv e^{-1} \equiv 7 \bmod 20$
	$\xleftarrow{k_{pub}=(33,3)}$	
$y = x^e \equiv 4^3 \equiv 31 \bmod 33$		
	$\xrightarrow{y=31}$	
		$y^d = 31^7 \equiv 4 = x \bmod 33$

Figure 1: RSA Algorithm

Example 1:
Choose $p = 61$ *and* $q = 53$.
Compute $n = p \cdot q = 61 \cdot 53 = 3233$.
Compute $\Phi(n) = (p-1)(q-1) = 3120$.
Select $e = 17$ *such that* $\gcd(e, \Phi(n)) = 1$.
Compute d *such that* $d \cdot e \equiv 1 \pmod{\Phi(n)}$.
Public key: $k_{pub} = (n, e) = (3233, 17)$.
Private key: $k_{pr} = (d) = (2753)$.
Given plaintext $x = 42$, *compute* y *using the encryption function.*
Computed ciphertext $y = 2557$, *Decrypt* y *to obtain the original plaintext* x.

Example 2:
Choose $p = 71$ *and* $q = 37$.
Compute $n = p \cdot q = 71 \cdot 37 = 2627$.
Compute $\Phi(n) = (p-1)(q-1) = 2520$.
Select $e = 11$ *such that* $\gcd(e, \Phi(n)) = 1$.
Compute d *such that* $d \cdot e \equiv 1 \pmod{\Phi(n)}$.
Public key: $k_{pub} = (n, e) = (2627, 11)$.
Private key: $k_{pr} = (d) = (2291)$.
Given plaintext $x = 128$, *compute* y *using the encryption function.*
Computed ciphertext $y = 2400$, *Decrypt* y *to obtain the original plaintext* x.

Implementation in C++

```
#include<bits/stdc++.h>
using namespace std;

bool isPrime(long long n) {
    if (n<=1) return 0;
    else if (n==2) return 1;
    else if ((n&1)==0) return 0;
    else {
        int i = 3;
```

```
        while (i*i <= n) {
            if (n % i == 0) return 0;
            i += 2;
        }
        return true;
    }
}

long long gcd(long long a, long long b, long long &x, long long &y) {
    if (a == 0) {
        x = 0;
        y = 1;
        return b;
    }

    long long x1, y1;
    long long g = gcd(b % a, a, x1, y1);

    x = y1 - (b / a) * x1;
    y = x1;

    return g;
}

long long inverse(long long e, long long phi) {
    long long x, y;
    long long g = gcd(e, phi, x, y);

    if (g != 1) {
        cerr << "Inverse doesn't exist\n";
        return -1;
    } else {
        // Ensure 'x' is positive
        return (x % phi + phi) % phi;
    }
}

long long mod_pow(long long base, long long exponent, long long modulus) {
    long long result = 1;

    while (exponent > 0) {
        if (exponent & 1)
            result = (result * base) % modulus;

        exponent >>= 1;
        base = (base * base) % modulus;
    }

    return result;
}

```

```

int main() {
    long long P, Q;
    cout<<"Please Enter the value of P: ";
    cin>>P;
    cout<<"Please Enter the value of Q: ";
    cin>>Q;
    while (!isPrime(P)) { // O(sqrt(P))
        cout<<P<<" is not prime. Please enter prime number(P): ";
        cin>>P;
    }
    while (!isPrime(Q)) { // O(sqrt(Q))
        cout<<Q<<" is not prime. Please enter prime number(Q): ";
        cin>>Q;
    }
    long long N = P * Q;
    long long phi = (P - 1) * (Q - 1);

    cout<<"Value of phi is: "<<phi<<endl;

    long long e;
    cout<<"Please Enter the value of e (2-"<<phi-1<<") that are coprime to "
    cin>>e;

    while (e < phi) {  // O(1)
        // e must be co-prime to phi and smaller than phi.
        if (gcd(e, phi) == 1)
            break;
        else {
            e = (e + 1) % phi;
            if (e == 1) e++;
        }
    }

    long long d = inverse(e, phi);

    cout<<"Value of e is : "<<e<<endl;
    cout<<"Value of d is : "<<d<<endl;

    long long m;
    cout<<"Please Enter the value of m: ";
    cin>>m;

    // Encryption c = (msg ^ e) % n
    long long c = mod_pow(m, e, N);  // O(log(e))
    cout<<"\nCipher Text = " << c;

    // Decryption m = (c ^ d) % n
    long long msg = mod_pow(c, d, N); // O(log(e))
    cout<<"\nOriginal Message Sent = " << msg;

```

```
    return 0;
}
```

```
Please Enter the value of P: 61
Please Enter the value of Q: 53
Value of phi is: 3120
Please Enter the value of e (2-3119) that are coprime to 3120 : 17
Value of e is : 17
Value of d is : 2753
Please Enter the value of m: 42
Cipher Text = 2557
Original Message Sent = 42
```

```
Please Enter the value of P: 71
Please Enter the value of Q: 37
Value of phi is: 2520
Please Enter the value of e (2-2519) that are coprime to 2520 : 11
Value of e is : 11
Value of d is : 2291
Please Enter the value of m: 128
Cipher Text = 2400
Original Message Sent = 128
```