Assignment 6

Name: Shreeshail Mahajan
PRN: 2020BTECS00055

Overview: RSA is a widely used public-key cryptosystem in network security and cryptography.

Code:

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

void file()
{
#ifndef ONLINE_JUDGE
   freopen("input.txt", "r", stdin);
   freopen("output.txt", "w", stdout);
#endif
}

// Function for extended Euclidean Algorithm
int ansS, ansT;
int findGcdExtended(int r1, int r2, int s1, int s2, int t1, int t2)
{
   // Base Case
   if (r2 == 0)
   {
       ansS = s1;
       ansT = t1;
       return r1;
   }

   int q = r1 / r2;
   int r = r1 % r2;

   int s = s1 - q * s2;
   int t = t1 - q * t2;
```

```
    cout << q << " " << r1 << " " << r2 << " " << r << " " << s1 << " " << s2 << " " << s << " " << t1 << " " << t2 << " " << t << endl;

    return findGcdExtended(r2, r, s2, s, t2, t);
}


int modInverse(int A, int M)
{
    int x, y;
    int g = findGcdExtended(A, M, 1, 0, 0, 1);
    if (g != 1) {
        cout << "Inverse doesn't exist";
        return 0;
    }
    else {

        // m is added to handle negative x


        int res = (ansS % M + M) % M;
        cout << "inverse is" << res << endl;
        return res;
    }
}

long long powM(long long a, long long b, long long n)
{
    if (b == 1)
        return a % n;
    long long x = powM(a, b / 2, n);
    x = (x * x) % n;
    if (b % 2)
        x = (x * a) % n;
    return x;
}

int findGCD(int num1, int num2)
{
    if (num1 == 0)
```

```
        return num2;
    return findGCD(num2 % num1, num1);
}

// Code to demonstrate RSA algorithm
int main()
{



    // Two random prime numbers
    long long p, q, e, msg;
    //17 31 7 2

    cout << "Please enter 2 prime number and e and Message to Encrypt" <<
endl;
    cin >> p >> q >> e >> msg;

    cout << "2 random prime numbers selected are " << p << " " << q <<
endl;

    // First part of public key:
    long long n = p * q;
    cout << "Product of two prime number n is " << n << endl;

    // Finding other part of public key.
    // e stands for encrypt

    cout << "Taken e is " << e << endl;

    long long phi = (p - 1) * (q - 1);
    cout << "phi is " << phi << endl;

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

```
    cout << "Final e value is " << e << endl;


    // Private key (d stands for decrypt)

    long long d = modInverse(e, phi);
    cout << "d is " << d << endl;

    cout << "\nso now our public key is " << "<" << e << "," << n << ">" <<
endl;
    cout << "\nso now our private key is " << "<" << d << "," << n << ">"
<< endl << endl;



    // Message to be encrypted

    cout << "Message date is " << msg << endl;

    // Encryption c = (msg ^ e) % n
    long long c = powM(msg, e, n);
    cout << "Encrypted Message is " << c << endl;

    // Decryption m = (c ^ d) % n
    long long m = powM(c, d, n);
    cout << "original Message is " << m << endl;

    return 0;
}
```

Output:

```
-o A6 } ; if ($?) { .\A6 }
Please enter 2 prime number and e and Message to Encrypt
7 11 67 1000
2 random prime numbers selected are 7 11
Product of two prime number n is 77
Taken e is 67
phi is 60
Final e value is 67
1 67 60 7 1 0 1 0 1 -1
8 60 7 4 0 1 -8 1 -1 9
1 7 4 3 1 -8 9 -1 9 -10
1 4 3 1 -8 9 -17 9 -10 19
3 3 1 0 9 -17 60 -10 19 -67
inverse is43
d is 43

so now our public key is <67,77>

so now our private key is <43,77>

Message date is 1000
Encrypted Message is 76
original Message is 76
PS F:\D drive\0Walchand_Sem7\CNS lab\RSA>
```

```
PS F:\D drive\0Walchand_Sem7\CNS lab> python -u "f:\D drive\0Walchand_Sem7\CNS lab\LA2\RSA_encry_decry.py"
____________________________________________________________
RSA  Algorithm - Shreeshail
____________________________________________________________
Public Key (n, e): (352785984459016643095740594717081006963l, 65537)
Private Key (n, d): (352785984459016643095740594717081006963l, 189643258502695520634793552948587608641)
____________________________________________________________
RSA  Algorithm
____________________________________________________________
1. Encrypt Numeric Message
2. Decrypt Numeric Message
3. Encrypt Text Message
4. Decrypt Text Message
5. Find Prime Factors of Number
6. Exit
Enter Choice Code:
```

The RSA (Rivest–Shamir–Adleman) algorithm is a widely used public-key cryptosystem that provides secure encryption and digital signatures. Its security is based on the mathematical properties of prime numbers. Here's an overview of the theory behind the RSA algorithm:

1. Key Generation:

- Select two large prime numbers, typically denoted as "p" and "q." The security of RSA relies on the difficulty of factoring the product of these two primes.

- Calculate the modulus "N" as the product of p and q: `N = p * q`.

- Compute the totient (Euler's totient function) of N, denoted as "φ(N)." It is calculated as: `φ(N) = (p - 1) * (q - 1)` since N is a product of two distinct primes.

- Choose a public exponent "e" such that 1 < e < φ(N), and e is relatively prime to φ(N). Common choices for "e" are small prime numbers like 3 or 65537, which ensures efficient encryption.

- Calculate the private exponent "d" as the modular multiplicative inverse of "e" modulo φ(N). This can be expressed as: `d ≡ e⁻¹ (mod φ(N))`. In other words, `d * e ≡ 1 (mod φ(N))`.

- The public key consists of (N, e), and the private key consists of (N, d).

2. Encryption:

- To encrypt a message "M," the sender uses the recipient's public key (N, e).

- The message "M" is first transformed into a numerical value "m" such that 0 <= m < N.

- The encryption is performed as: `C ≡ m^e (mod N)`, where "C" represents the ciphertext.

- The ciphertext "C" is then sent to the recipient.

3. Decryption:

- The recipient uses their private key (N, d) to decrypt the ciphertext "C."

- Decryption is carried out as: `m ≡ C^d (mod N)`, where "m" is the numerical value of the original message.

- The recipient then converts "m" back to the original message "M."

4. Security:

- The security of RSA is based on the difficulty of factoring the modulus "N" into its two prime factors, p and q. If an attacker can factor "N," they can compute φ(N) and derive the private exponent "d" from the public exponent "e." Thus, the security of RSA relies on the assumption that factoring N is a computationally infeasible task for large, well-chosen primes.

5. Digital Signatures:

- RSA can also be used for digital signatures. To create a digital signature, the sender applies a hash function to the message, encrypts the hash value using their private key, and attaches the resulting signature to the message. The recipient can verify the signature using the sender's public key.

Conclusion:

In practice, RSA is a cornerstone of network security and cryptography. It is employed in secure communication protocols like HTTPS for web encryption and SSH for secure remote login, among many others. However, users and security professionals should always be mindful of key management and security best practices to ensure the ongoing reliability of RSA in the face of evolving threats. As computing power advances, it's essential to periodically review and update the key sizes used in RSA to maintain a high level of security.