Assignment 2

RSA Algorithm:

Code:-

```
#include <iostream>
#include <ctime>
#include <string>
#include <sstream>
#include <vector>
#include <iomanip>

using namespace std;

// Function to check if a number is prime
bool isPrime(int n) {
    if (n <= 1) return false;
    if (n <= 3) return true;

    if (n % 2 == 0 || n % 3 == 0) return false;

    for (int i = 5; i * i <= n; i += 6) {
        if (n % i == 0 || n % (i + 2) == 0)
            return false;
    }

    return true;
}

// Function to generate a list of prime numbers up to a given limit
vector<int> generatePrimes(int limit) {
    vector<int> primes;

    for (int i = 2; i <= limit; i++) {
        if (isPrime(i)) {
            primes.push_back(i);
        }
    }

    return primes;
}

// Function to calculate the greatest common divisor
int gcd(int a, int b) {
    if (b == 0)
```

```
        return a;
    return gcd(b, a % b);
}

// Function to generate the RSA key pair
void generateKeyPair(int& n, int& e, int& d) {
    srand(time(nullptr));

    // Generate a list of prime numbers up to a reasonable limit
    vector<int> primes = generatePrimes(100);

    int p, q;

    // Randomly select two prime numbers
    cout << "Enter the first prime number (p): ";
    cin >> p;

    cout << "Enter the second prime number (q): ";
    cin >> q;

    n = p * q;
    int phi = (p - 1) * (q - 1);

    // Find e (public key)
    for (e = 2; e < phi; e++) {
        if (gcd(e, phi) == 1)
            break;
    }

    // Find d (private key)
    d = 2;
    while ((d * e) % phi != 1) {
        d++;
    }
}

// Function to encrypt a message
string encrypt(const string& message, int e, int n) {
    string ciphertext = "";

    for (char c : message) {
        int m = static_cast<int>(c);
        int crypted = 1;
        for (int i = 0; i < e; i++) {
            crypted = (crypted * m) % n;
        }
        ciphertext += to_string(crypted) + " ";
    }

    return ciphertext;
}
```

```
// Function to decrypt a ciphertext
string decrypt(const string& ciphertext, int d, int n) {
    stringstream ss(ciphertext);
    string decrypted = "";
    int c;

    while (ss >> c) {
        int decryptedChar = 1;
        for (int i = 0; i < d; i++) {
            decryptedChar = (decryptedChar * c) % n;
        }
        decrypted += static_cast<char>(decryptedChar);
    }

    return decrypted;
}

int main() {
    int n, e, d;
    string message;

    // Generate RSA keys
    generateKeyPair(n, e, d);

    cout << "Enter the message to be encrypted: ";
    cin.ignore();
    getline(cin, message);

    // Time taken for encryption
    clock_t start = clock();
    string ciphertext = encrypt(message, e, n);
    clock_t end = clock();
    double encryptionTime = static_cast<double>(end - start) / CLOCKS_PER_SEC;

    // Time taken for decryption
    start = clock();
    string decryptedMessage = decrypt(ciphertext, d, n);
    end = clock();
    double decryptionTime = static_cast<double>(end - start) / CLOCKS_PER_SEC;
    cout << fixed << setprecision(6);  // Format time output

    cout << "\nTime taken for encryption: " << encryptionTime << " seconds\n";
    cout << "Time taken for decryption: " << decryptionTime << " seconds\n";

    cout << "\nMessage provided for encryption: " << message << "\n";
    cout << "Ciphertext generated: " << ciphertext << "\n";
    cout << "Decrypted message: " << decryptedMessage << "\n";

    return 0;
}
```

Output:-

```
Enter the first prime number (p): 11
Enter the second prime number (q): 13
Enter the message to be encrypted: Anjali

Time taken for encryption: 0.000019 seconds
Time taken for decryption: 0.000020 seconds

Message provided for encryption: Anjali
Ciphertext generated: 65 33 50 59 4 118
Decrypted message: Anjali
```

Conclusion:-

- RSA is a robust asymmetric encryption algorithm widely used for secure communication and data protection.
- The algorithm's mathematical foundation ensures that while encryption is efficient, decryption is computationally difficult without the private key.
- RSA's key generation, encryption, and decryption processes are well-defined and provide a secure way to transmit information over insecure channels.
- However, RSA is computationally intensive, especially for large values of n, and its security relies on the proper selection of key sizes and the quality of random number generators.
- As computing power increases, larger key sizes are necessary to maintain security.