**RSA**

**Code**

#include <iostream>

#include <cmath>

using namespace std;

int gcd(int a, int b) {

int t;

while(1) {

t = a % b;

if (t == 0)

return b;

a = b;

b = t;

}

}

int main() {

double p, q, e, message;

cout << "Enter the value of p and q: ";

cin >> p>>q;

double n = p \* q;

double track;

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

cout << "Enter the value of e (such that 1 < e < " << phi << " and gcd(e, phi) = 1): ";

cin >> e;

while (e >= phi || gcd(e, phi) != 1) {

cout << "Invalid value of e! Enter a value such that 1 < e < " << phi << " and gcd(e, phi) = 1: ";

cin >> e;

}

cout << "Enter the message to encrypt: ";

cin >> message;

double d1 = 1 / e;

double d = fmod(d1, phi);

double c = pow(message, e);

double m = pow(c, d);

c = fmod(c, n);

m = fmod(m, n);

cout << "Original Message = " << message << endl;

cout << "p = " << p << ", q = " << q << endl;

cout << "n = pq = " << n << ", phi = " << phi << endl;

cout << "e = " << e << ", d = " << d << endl;

cout << "Encrypted message = " << c << endl;

cout << "Decrypted message = " << m << endl;

return 0;

}

**Output:**

Enter the value of p and q: 17 7

Enter the value of e (such that 1 < e < 96 and gcd(e, phi) = 1): 11

Enter the message to encrypt: 21

Original Message = 21

p = 17, q = 7

n = pq = 119, phi = 96

e = 11, d = 0.0909091

Encrypted message = 98

Decrypted message = 21

**Data encryption standard (DES)**

**Code:**

#include <iostream>

#include <bitset>

#include <vector>

using namespace std;

vector<int> initial\_perm = {2, 6, 3, 1, 4, 8, 5, 7};

vector<int> inv\_initial\_perm = {4, 1, 3, 5, 7, 2, 8, 6};

string permute(const string& input, const vector<int>& permutation) {

string result;

for (int idx : permutation) {

result += input[idx - 1];

}

return result;

}

string xor\_strings(const string& a, const string& b) {

string result;

for (size\_t i = 0; i < a.size(); ++i) {

result += (a[i] == b[i] ? '0' : '1');

}

return result;

}

string encrypt(const string& plaintext, const string& key) {

string permuted\_plaintext = permute(plaintext, initial\_perm);

string expanded\_key = key.substr(0, 8); // Use the first 8 characters of the key

string ciphertext = xor\_strings(permuted\_plaintext, expanded\_key);

return ciphertext;

}

string decrypt(const string& ciphertext, const string& key) {

string expanded\_key = key.substr(0, 8); // Use the first 8 characters of the key

string permuted\_plaintext = xor\_strings(ciphertext, expanded\_key);

string plaintext = permute(permuted\_plaintext, inv\_initial\_perm);

return plaintext;

}

int main() {

string plaintext = "10101100";

string key = "11001010";

string ciphertext = encrypt(plaintext, key);

cout << "Ciphertext: " << ciphertext << endl;

string decrypted\_plaintext = decrypt(ciphertext, key);

cout << "Decrypted plaintext: " << decrypted\_plaintext << endl;

return 0;

}

**Output:**

Ciphertext: 00001111

Decrypted plaintext: 10101100