**Marathwada Mitra Mandal’s**

**COLLEGE OF ENGINEERING, PUNE**

**DEPARTMENT OF COMPUTER ENGINEERING**

**BE Comp-II 2020-21**

**SUBJECT: ICS SEMESTER-II**

**Lab Assignment4**

**Implementation of Diffi-Hellman (DH) algorithm**

**Theory:**

**Introduction of Diffie-Hellman Key Exchange:**

Diffie – Hellman algorithm is an algorithm that allows two parties to get the shared secret key using the communication channel, which is not protected from the interception but is protected from modification.

Diffie – Hellman algorithm is extremely simple in its idea and with it has rather high level of cryptographic stability, which is based on the supposed complexity of the discrete problem of taking the logarithm.

Supposing there are two participants of the exchange (let’s call them Alice and Bob, as it is traditionally established in cryptography). Both of them know two numbers P and G. These numbers are not secret and can be known to anyone. The goal of Alice and Bob is to obtain the shared secret key to help them to exchange messages in future.

**Stage 1:**

For this, they generate two big random numbers (so called private keys): Alice - number Xa, Bob - number Xb. After this, Alice computes the value of the public key:

Ya=(G^Xa) mod P

and sends it to Bob.

In his turn, Bob computes the value of his public key:

Yb=(G^Xb) mod P

and sends it to Alice.

**Stage 2:**

At the second stage, on the basis of her private key and the public key, received from Bob, Alice computes the value

Ka=(Yb^Xa) mod P

Similarly, Bob computes the value

Kb=(Ya^Xb) mod P.

Numbers Ka and Kb are equal because

Ka = (Yb^Xa) mod P = (((G^Xb) mod P)^Xa) mod P = (G^XaXb) mod P = (((G^Xa) mod P)^Xb) mod P = (Ya^Xb) mod P = Kb.

and they can be used as the secret key by Alice and Bob. In practical implementations, numbers of 10^100 order are used for Xa and Xb, for P - 10^300. The G number often has the value in the range of the first ten.

**DH algorithm steps:**

* all users agree on global parameters: (A and B)
  + large prime integer or polynomial q
  + α a primitive root mod q
* each user (eg. A) generates their key
  + chooses a secret key (number): xA < q
  + compute their **public key**: YA = αxA mod q
* each user (eg. B) generates their key
  + chooses a secret key (number): xB < q
  + compute their **public key**: YB = αxB mod q
* each user makes public that key YA AND YB
* shared session key for users A & B is KAB:
  + KAB = αxA.xB mod q
  + = YAxB mod q (which **B** can compute)
  + = YBxA mod q (which **A** can compute)

**DH Example**

* users A & B who wish to swap keys:
* agree on prime q=353 and α=3
* select random secret keys:
  + A chooses xA=97, B chooses xB=233
* compute public keys:
  + yA=397 mod 353 = 40 (A)
  + yB=3233 mod 353 = 248 (B)
* compute shared session key as:
  + KAB= yBxA mod 353 = 24897 = 160 (A)
  + KAB= yAxB mod 353 = 40233 = 160 (B)

**DH pseudocode for implementation:**

Use C/C++/Java programming to code DH as per algorithm steps.

**Implementation steps:**

C++ Program:

#include<iostream>

using namespace std;

long compute(long alpha, long a, long q) {

long long y = 1;

while(a > 0) {

long r = a % 2;

if(r == 1) {

y = (y\*alpha) % q;

} alpha = alpha \* alpha % q;

a /= 2;

} return (long)y;

}

int main() {

long q, alpha; // prime number q and primitive root

cout << "\nEnter large prime number : ";

cin >> q;

cout << "\nEnter primitive root : ";

cin >> alpha;

long a, b; // Secret keys of users Alice and Bob

long A, B; // Public keys of users Alice and Bob

long KA, KB; // Shared session keys of users Alice and Bob

cout << "\nEnter secret key for Alice : ";

cin >> a;

cout << "Enter secret key for Bob : ";

cin >> b;

A = compute(alpha,a,q);

B = compute(alpha,b,q);

cout << "\nPublic key of Alice is : " << A;

cout << "\nPublic key of Bob is : " << B;

cout << "\nPublic keys of Alice and Bob have been shared between them.\n";

KA = compute(B,a,q);

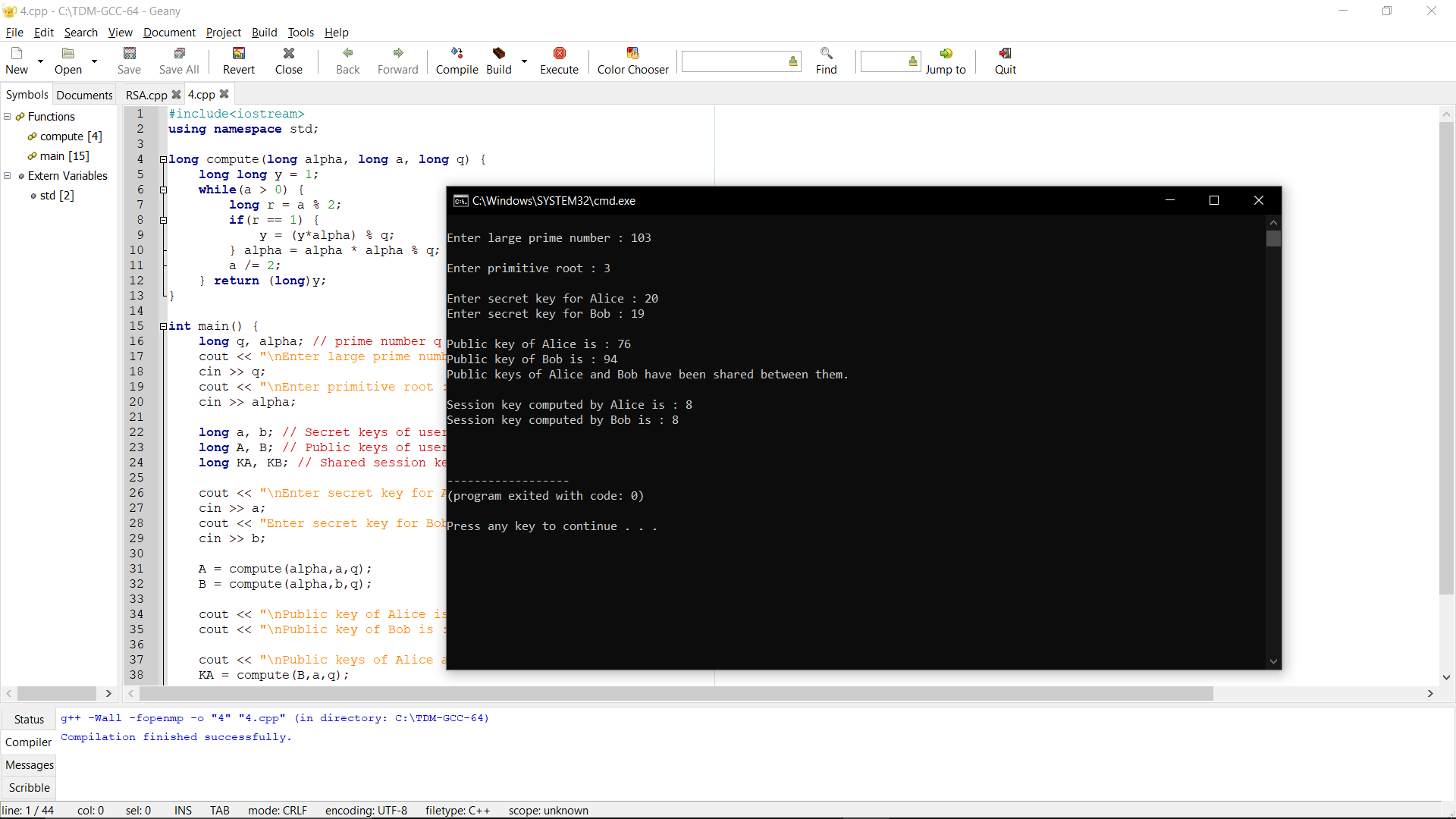
KB = compute(A,b,q);

cout << "\nSession key computed by Alice is : " << KA;

cout << "\nSession key computed by Bob is : " << KB << "\n\n";

}

**Result with screenshot:**

****

**Outcome:**

We are able to

1. Identify need of information security, Public key Encryption Standards and Diffie-Hellman key exchange Algorithm Cipher Operation Techniques

2. Design & implement Public key Encryption Standards and Cipher Operation Techniques