**4CS401: Cryptography and Network Security**

**B.Tech. (CSE) – I [ 2022-23 ]**

**Assignment No - 8**

**Diffie-Hellman problem**

**Title: Diffie-Hellman problem**

**Aim: To Demonstrate Diffie-Hellman problem**

**Theory:**

**The Diffie Hellman problem is used to share a secret key. The key is shared such away that no one can decrypt the key until he/she has a private key and a public key.**

**Code :**

**#include <bits/stdc++.h>**

**#define ll long long**

**#define ul unsigned long long**

**#define pb emplace\_back**

**#define po pop\_back**

**#define vi vector<ll>**

**#define vii vector<vector<ll>>**

**using namespace std;**

**vector<int> primeNums;**

**vector<bool> prime(100000001,1);**

**void SeiveOfEratosthenes(int n){**

**for(int p=2; p\*p<=n; p++){**

**if(prime[p] ==  true){**

**for (int i = p \* p; i <= n; i += p)**

**prime[i] = false;**

**}**

**}**

**for(int i=3;i<n;i+=2){**

**if(prime[i]) primeNums.push\_back(i);**

**}**

**}**

**ll power(ll a, ll b, ll p){**

**if (b == 1)**

**return a;**

**else**

**return (((long long int)pow(a, b)) % p);**

**}**

**void findPrimefactors(unordered\_set<int> &s, int n){**

**while (n%2 == 0){**

**s.insert(2);**

**n = n/2;**

**}**

**for (int i = 3; i <= sqrt(n); i = i+2){**

**while (n%i == 0){**

**s.insert(i);**

**n = n/i;**

**}**

**}**

**if (n > 2)**

**s.insert(n);**

**}**

**int primitiveRoot(int n){**

**unordered\_set<int> s;**

**int phi = n-1;**

**findPrimefactors(s, phi);**

**for (int r=2; r<=phi; r++){**

**bool flag = false;**

**for (auto it = s.begin(); it != s.end(); it++){**

**if (power((ll)r, (ll)phi/(\*it),(ll)n) == 1)**

**{**

**flag = true;**

**break;**

**}**

**}**

**if (flag == false)**

**return r;**

**}**

**return -1;**

**}**

**int main(){**

**// prime number till 100000000**

**SeiveOfEratosthenes(100000000);**

**int privateNumberA, privateNumberB;**

**cout<<"Enter the privateNumber of A and B respectively \n";**

**cin>>privateNumberA>>privateNumberB;**

**cout<<"Finding prime Number and a primitive root ................\n";**

**int  p = primeNums[rand() % primeNums.size()];**

**int g = primitiveRoot(p);**

**cout<<"Prime Number : "<<p<<"\n";**

**cout<<"Primitive Root :"<<g<<"\n";**

**// calculating the private key for  a**

**ll x = power(g,privateNumberA,p);**

**if(x<0) x = p + x;**

**cout<<"the private key a for A is : "<<x<<"\n";**

**// calculate private key for b**

**ll y = power(g,privateNumberB,p);**

**if(y<0) y = p + y;**

**cout<<"the private key b for B is : "<<y<<"\n";**

**ll ka = power(y, privateNumberA, p); // Secret key for A**

**if(ka<0) ka = p + ka;**

**ll  kb = power(x, privateNumberB, p); // Secret key for B**

**if(kb<0) kb = p + kb;**

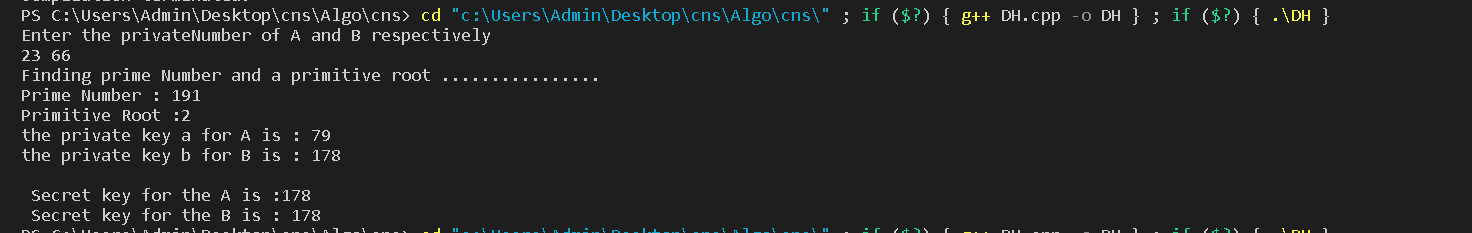
**cout<<"\n Secret key for the A is :"<<ka;**

**cout<<"\n Secret key for the B is : "<<kb;**

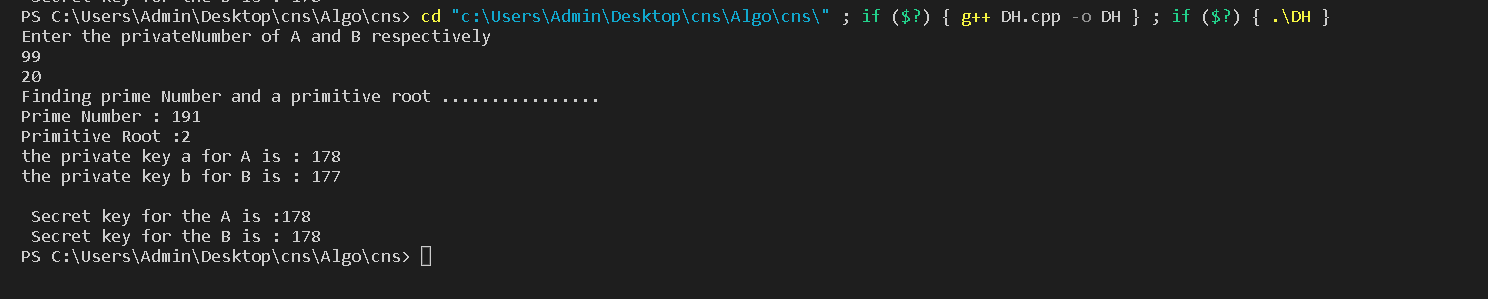
**return 0;**

**}**

**Output 1:**

****

**Output 2:**

****

****Conclusion:****

**Thus Diffie-Hellman is used to share secret keys.**