C&NS Lab Assignment 8

Onkar Santosh Gavali (2019BTECS00037)

Batch B2

# Index

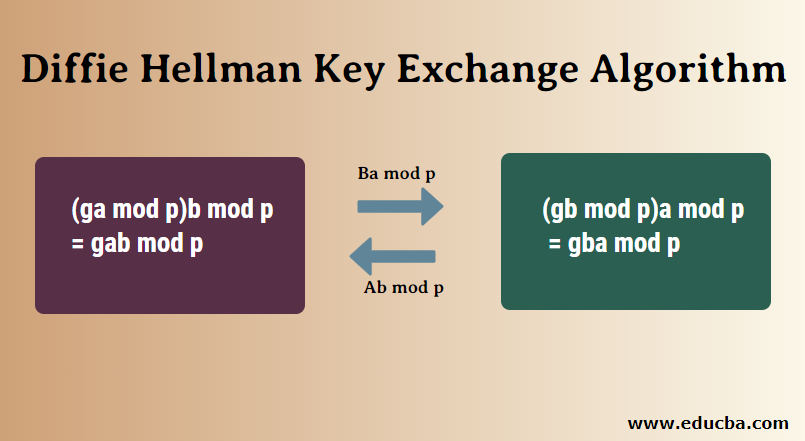
Diffie Hellman.

* Explain Diffie Hellman.
* Implement the Diffie Hellman algorithm using any programming language.

# 

# 

# Diffie Hellman



Here

G and p are public keys

P is prime and g is primitive root of p

a and b are private keys

Generate i=g^a mod p and j =g^b mod p

Exchange i and j

And generate key using i and j

Key = j^a mod p = i^b mod p;

**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 : ";

cin>>privateNumberA>>privateNumberB;

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

srand(time(0));

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

int g = primitiveRoot(p);

cout<<"\tPrime Number : "<<p<<"\n";

cout<<"\tPrimitive Root :"<<g<<"\n";

// calculating the private key for a

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

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

cout<<"\nThe 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;

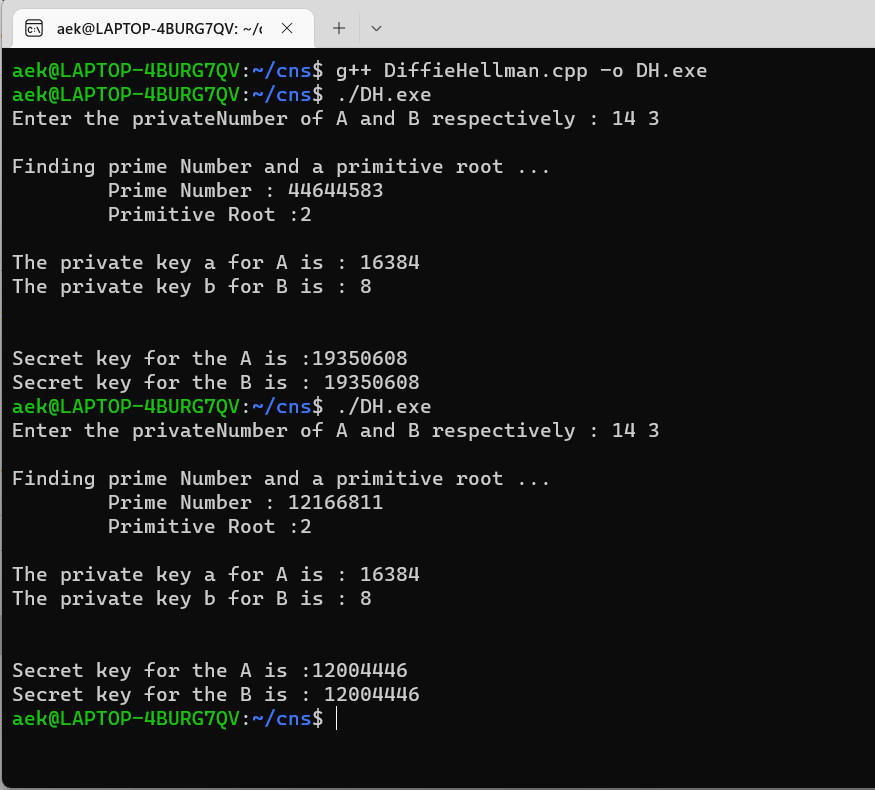
cout<<"\n\nSecret key for the A is :"<<ka;

cout<<"\nSecret key for the B is : "<<kb<<endl;

return 0;

}

**Output :**



# 

# 