EXP NO:5 **DATE:01/03/2024**

DIFFIE-HELLMAN KEY EXCHANGE

Aim: To implement Diffie-Hellman key exchange using C.

Algorithm:

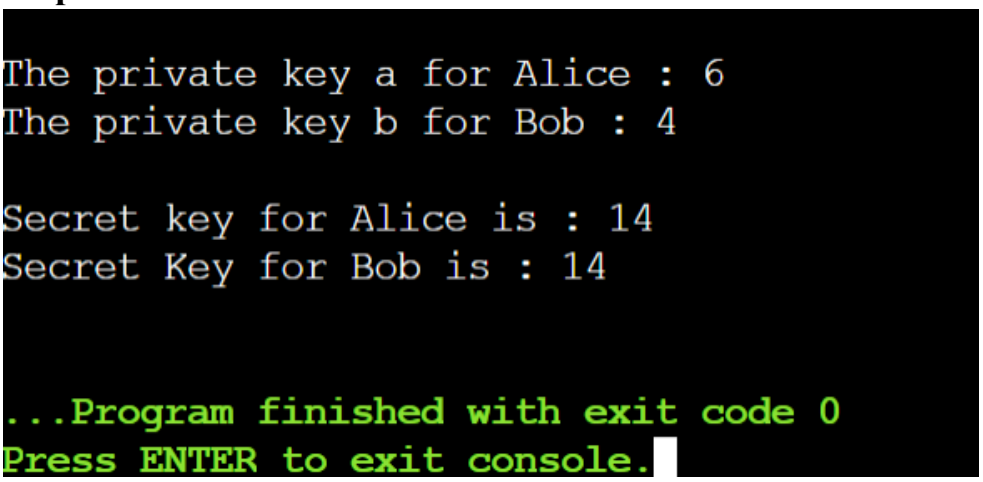
- Step 1: Choose a large prime number P and a primitive root modulo (P), denoted as (G). Both parties agree on these values.
- Step 2: Alice chooses a private key (a), while Bob chooses a private key (b). These private keys are kept secret.
- Step 3: Alice calculates her public key (x) using (x = G^a mod P), and Bob calculates his public key (y) using (y = G^b mod P).
- Step 4: Alice sends her public key (x) to Bob, and Bob sends his public key (y) to Alice.
- Step 5: Using the received public keys, Alice computes the secret key (ka) using (ka= y^a mod P), and Bob computes the secret key (kb) using (kb = x^b mod P).
- Step 6: Both Alice and Bob now have the same shared secret key.
- Step 7: They can now communicate securely using the shared secret key for encryption and decryption.
- Step 8: The security of the Diffie-Hellman Key Exchange relies on the difficulty of calculating discrete logarithms in finite fields.

Program:

```
#include <math.h>
#include <stdio.h>

long long int power(long long int a, long long int b, long long int P) {
if (b == 1)        return a;     else
      return (((long long int)pow(a, b)) % P); }
```

```
int main() {     long long int P, G, x, a,
y, b, ka, kb;

   P = 26;
   printf("The value of P : %lld\n", P);

   G = 12;
   printf("The value of G : %lld\n\n", G);

a=6;     printf("The private key a for Alice :
%lld\n", a);     x = power(G, a, P);

a=4 ;printf("The private key b for Bob : %lld\n\n",
b);     y = power(G, b, P);

   ka = power(y, a, P);
   kb = power(x, b, P);

   printf("Secret key for Alice is : %lld\n", ka);
   printf("Secret Key for Bob is : %lld\n", kb);

   return 0;
}
```

Output:



Result:

The Diffie-hellman key exchange algorithm has been implemented using C.