

Ex. No.: 5

Date:

DIFFIE-HELLMAN KEY EXCHANGE

Problem Statement :

The simplest and the original implementation of the protocol uses the multiplicative group of integers modulo p , where p is prime, and g is a primitive root modulo p . Here is an example of the protocol, with non-secret values in blue, and secret values in **red**.

- Alice and Bob agree to use a prime number $p = 23$ and base $g = 5$ (which is a primitive root modulo 23).
- Alice chooses a secret integer **$a = 6$** , then sends Bob $A = g^a \bmod p$
 - $A = 5^6 \bmod 23 = 8$
- Bob chooses a secret integer **$b = 15$** , then sends Alice $B = g^b \bmod p$
 - $B = 5^{15} \bmod 23 = 19$
- Alice computes **$s = B^a \bmod p$**
 - $s = 19^6 \bmod 23 = 2$
- Bob computes **$s = A^b \bmod p$**
 - $s = 8^{15} \bmod 23 = 2$
- Alice and Bob now share a secret (the number **2**).

Aim:

To implement Diffie-Hellman key exchange using C.

Algorithm:

- Get a prime number q as input from the user.
- Get a value x_a and x_b which is less than q .
- Calculate primitive root α
- For each user A , generate a key $X_a < q$
- Compute public key, $\alpha^{\text{pow}(X_a)} \bmod q$
- Each user computes Y_a
- Print the values of exchanged keys.

Program Code:

```
//This program uses fast exponentiation function power instead of pow library function
#include <stdio.h>
#include <math.h>
int power( int,unsigned int,int);
int main()
{
    int x,y,z,count,ai[20][20];
```

```

int alpha,xa,xb,ya,yb,ka,kb,q;
printf("\nEnter a Prime Number \"q\":");
scanf("%d",&q);
printf("\nEnter a No \"xa\" which is less than value of
q:"); scanf("%d",&xa);
printf("\nEnter a No \"xb\" which is less than value of q:");
scanf("%d",&xb);
printf("\nEnter alpha:");
scanf("%d",&alpha);
ya =
power(alpha,xa,q);
yb = power(alpha,xb,q);
ka = power(yb,xa,q);
kb = power(ya,xb,q);
printf("\nya = %d \nyb = %d \nka = %d \nkb = %d \n",ya,yb,ka,kb);
if(ka == kb)
    printf("\nThe secret keys generated by User A and User B are same\n");
else
    printf("\nThe secret keys generated by User A and User B are not
same\n"); return 0;
}

```

```

int power(int x, unsigned int y, int p)
{
    int res = 1;    // Initialize result
    x = x % p; // Update x if it is more than or equal to p

    while (y > 0)

    {
        // If y is odd, multiply x
        with result if (y & 1)
        res = (res*x) % p;

        // y must be even
        now y = y>>1; // y =
        y/2 x
        = (x*x) % p;
    }
    return res;
}

```

Output:

```
java -cp /tmp/t2kygKrcFd DH
Both users should agree upon:
PUBLIC KEY OF G:
8
PUBLIC KEY OF P:
20
PRIVATE KEY OF USER1:
16
PRIVATE KEY OF USER2:
32
Secret key of user1 is:1
Secret key of user2 is:7
|
```

Result: