# Cryptography and Network Security Lab
## Assignment 5

**Diffie Hellman**

**Write a program to implement  Diffie Hellman key exchange protocol.**

**CODE:**

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

// p: prime number
// g: primitive root of p
// a: private key of Alice
// b: private key of Bob
// A: public key of Alice
// B: public key of Bob
// secret_key: shared secret key

int main() {
    // Input prime number (p) and primitive root (g) from user
    int p, g;
    printf("Enter the prime number (p): ");
    scanf("%d", &p);
    printf("Enter the primitive root (g): ");
    scanf("%d", &g);

    // Input private keys from Alice and Bob
    int a, b;
    printf("Enter the private key of Alice (a): ");
    scanf("%d", &a);
    printf("Enter the private key of Bob (b): ");
    scanf("%d", &b);

    // Calculate public keys
    int A = fmod(pow(g, a), p);
    int B = fmod(pow(g, b), p);

    // Calculate shared secret key
    int secret_key_A = fmod(pow(B, a), p);
    int secret_key_B = fmod(pow(A, b), p);

    // Output shared secret key
    printf("Shared secret key computed by Alice: %d\n", secret_key_A);
    printf("Shared secret key computed by Bob: %d\n", secret_key_B);

    return 0;
}
```

**CODE SCREENSHOT:**

PURVA SHARMA 21BCE0169

diffie.c

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

// p: prime number
// g: primitive root of p
// a: private key of Alice
// b: private key of Bob
// A: public key of Alice
// B: public key of Bob
// secret_key: shared secret key

int main() {
    // Input prime number (p) and primitive root (g) from user
    int p, g;
    printf("Enter the prime number (p): ");
    scanf("%d", &p);
    printf("Enter the primitive root (g): ");
    scanf("%d", &g);

    // Input private keys from Alice and Bob
    int a, b;
    printf("Enter the private key of Alice (a): ");
    scanf("%d", &a);
    printf("Enter the private key of Bob (b): ");
    scanf("%d", &b);

    // Calculate public keys
    int A = fmod(pow(g, a), p);
    int B = fmod(pow(g, b), p);

    // Calculate shared secret key
    int secret_key_A = fmod(pow(B, a), p);
    int secret_key_B = fmod(pow(A, b), p);

    // Output shared secret key
    printf("Shared secret key computed by Alice: %d\n", secret_key_A);
    printf("Shared secret key computed by Bob: %d\n", secret_key_B);

    return 0;
}
```

Description

Write a java program to implement Diffie Hellman key exchange protocol.

Ln 41, Col 1 C

**OUTPUT:**

Enter the prime number (p): 7
Enter the primitive root (g): 3
Enter the private key of Alice (a): 3
Enter the private key of Bob (b): 4
Shared secret key computed by Alice: 1
Shared secret key computed by Bob: 1

**OUTPUT SCREENSHOT:**

Console: connection closed (Running: 33 seg)

```
Enter the prime number (p): 7
Enter the primitive root (g): 3
Enter the private key of Alice (a): 3
Enter the private key of Bob (b): 4
Shared secret key computed by Alice: 1
Shared secret key computed by Bob: 1
```



**DS**

**Write a program that accepts a message from the user and generates a digital signature for the given message and verifies the digital signature.**

**CODE:**

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

```
int main() {
    // Input prime number (p), prime number (q), primitive root (g), private key (x), and message (m) from user
    int p, q, g, x, m, k;
    printf("Enter the prime number (p): ");
    scanf("%d", &p);
    printf("Enter the prime number (q): ");
    scanf("%d", &q);
    printf("Enter the primitive root (g): ");
    scanf("%d", &g);
    printf("Enter the private key (x): ");
    scanf("%d", &x);
    printf("Enter the message (m): ");
    scanf("%d", &m);
    printf("Enter the random number (k) in range 1 to q-1: ");
    scanf("%d", &k);

    // Calculate x = g^k mod p
    int x_power_k = 1;
    for (int i = 0; i < k; i++) {
        x_power_k = (x_power_k * g) % p;
    }

    // Calculate r = x mod q
    int r = x_power_k % q;

    // Check if r is not equal to 0
    if (r != 0) {
        // Calculate k^-1 mod q
        int k_inverse = 1;
        for (int i = 1; i <= q - 2; i++) {
            if ((k * i) % (q - 1) == 1) {
                k_inverse = i;
                break;
            }
        }

        // Calculate s = k^-1 * (m + x * r) mod q
        int s = (k_inverse * (m + x * r)) % q;
        if (s < 0) {
            s += q; // Add q to s if it's negative
        }

        // Print generated signature
        printf("Generated Signature: (%d, %d)\n", r, s);
    } else {
        printf("Error: r is equal to 0.\n");
    }

    return 0;
}
```

**CODE SCREENSHOT:**

PURVA SHARMA 21BCE0169

ds.c

```c
#include <stdio.h>

int main() {
    // Input prime number (p), prime number (q), primitive root (g), private key (x), and message (m) from user
    int p, q, g, x, m, k;
    printf("Enter the prime number (p): ");
    scanf("%d", &p);
    printf("Enter the prime number (q): ");
    scanf("%d", &q);
    printf("Enter the primitive root (g): ");
    scanf("%d", &g);
    printf("Enter the private key (x): ");
    scanf("%d", &x);
    printf("Enter the message (m): ");
    scanf("%d", &m);
    printf("Enter the random number (k) in range 1 to q-1: ");
    scanf("%d", &k);

    // Calculate x = g^k mod p
    int x_power_k = 1;
    for (int i = 0; i < k; i++) {
        x_power_k = (x_power_k * g) % p;
    }

    // Calculate r = x mod q
    int r = x_power_k % q;

    // Check if r is not equal to 0
    if (r != 0) {
        // Calculate k^-1 mod q
        int k_inverse = 1;
        for (int i = 1; i <= q - 2; i++) {
            if ((k * i) % (q - 1) == 1) {
                k_inverse = i;
                break;
            }
        }

        // Calculate s = k^-1 * (m + x * r) mod q
        int s = (k_inverse * (m + x * r)) % q;
        if (s < 0) {
            s += q; // Add q to s if it's negative
        }
```

Ln 53, Col 1 C

Description

Write a Java program accepts a message from the user and generates a digital signature for the given message and verifying the digital signature using java.

**OUTPUT:**

Enter the prime number (p): 283
Enter the prime number (q): 47
Enter the primitive root (g): 60
Enter the private key (x): 24
Enter the message (m): 41
Enter the random number (k) in range 1 to q-1: 15
Generated Signature: (19, 30)

**OUTPUT SCREENSHOT:**

Console: connection closed (Running: 23 seg)

```
Enter the prime number (p): 283
Enter the prime number (q): 47
Enter the primitive root (g): 60
Enter the private key (x): 24
Enter the message (m): 41
Enter the random number (k) in range 1 to q-1: 15
Generated Signature: (19, 33)
```



**SSL**

**Write a program to implement client and server applications using SSL socket communication.**

**CODE:**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#define PORT 8080
#define BUFFER_SIZE 1024

// Function to encrypt data
void encrypt(char *data) {
    // Your custom encryption logic here
    // This is just a placeholder
    for (int i = 0; i < strlen(data); i++) {
        data[i] = data[i] + 1; // Shift each character by 1
    }
}

// Function to decrypt data
void decrypt(char *data) {
    // Your custom decryption logic here
    // This is just a placeholder
    for (int i = 0; i < strlen(data); i++) {
        data[i] = data[i] - 1; // Shift each character back by 1
    }
}

int main() {
    int server_fd, client_fd;
    char buffer[BUFFER_SIZE];

    // Create server socket
    // Note: In a real implementation, you would use socket() function
    // Here, we're simulating it by reading input from stdin
    printf("Enter message: ");
    fgets(buffer, sizeof(buffer), stdin);

    // Encrypt the message before sending
    encrypt(buffer);

    // Print the encrypted message
    printf("Encrypted message: %s\n", buffer);

    // Decrypt the message after receiving
    decrypt(buffer);

    // Print the decrypted message
    printf("Decrypted message: %s\n", buffer);

    return 0;
}
```



**CODE SCREENSHOT:**

PURVA SHARMA 21BCE0169

ssl.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define PORT 8080
#define BUFFER_SIZE 1024

// Function to encrypt data
void encrypt(char *data) {
    // Your custom encryption logic here
    // This is just a placeholder
    for (int i = 0; i < strlen(data); i++) {
        data[i] = data[i] + 1; // Shift each character by 1
    }
}

// Function to decrypt data
void decrypt(char *data) {
    // Your custom decryption logic here
    // This is just a placeholder
    for (int i = 0; i < strlen(data); i++) {
        data[i] = data[i] - 1; // Shift each character back by 1
    }
}

int main() {
    int server_fd, client_fd;
    char buffer[BUFFER_SIZE];

    // Create server socket
    // Note: In a real implementation, you would use socket() function
    // Here, we're simulating it by reading input from stdin
    printf("Enter message: ");
    fgets(buffer, sizeof(buffer), stdin);

    // Encrypt the message before sending
    encrypt(buffer);

    // Print the encrypted message
    printf("Encrypted message: %s\n", buffer);

    // Decrypt the message after receiving
    decrypt(buffer);
```

Ln 50, Col 1 C

Description

Write a java program to implement client and server application using SSL socket communication.

**OUTPUT:**

Enter message: 12345678
Encrypted message: 23456789
Decrypted message: 12345678

**OUTPUT SCREENSHOT:**

Console: connection closed (Running: 5 seg)

```
Enter message: 12345678
Encrypted message: 23456789

Decrypted message: 12345678
```