

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);

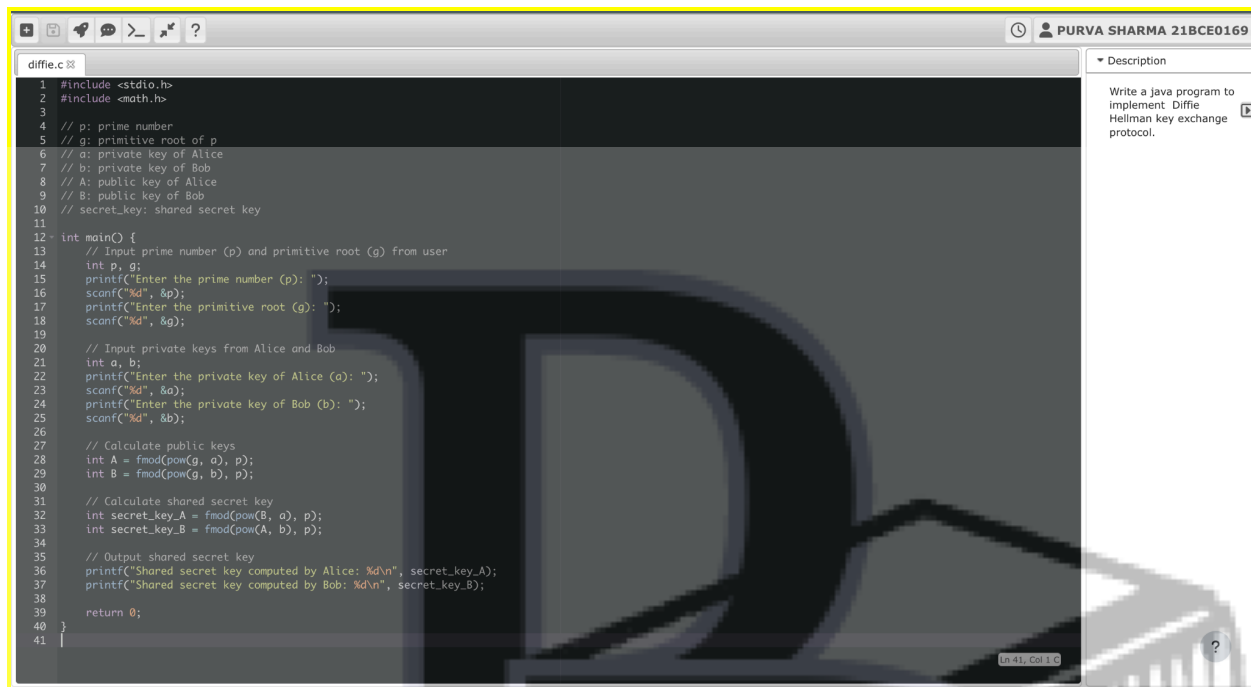
    // Calculate public keys
    int A = fmod(pow(g, a), p);
    int B = fmod(pow(g, b), p);

    // 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:



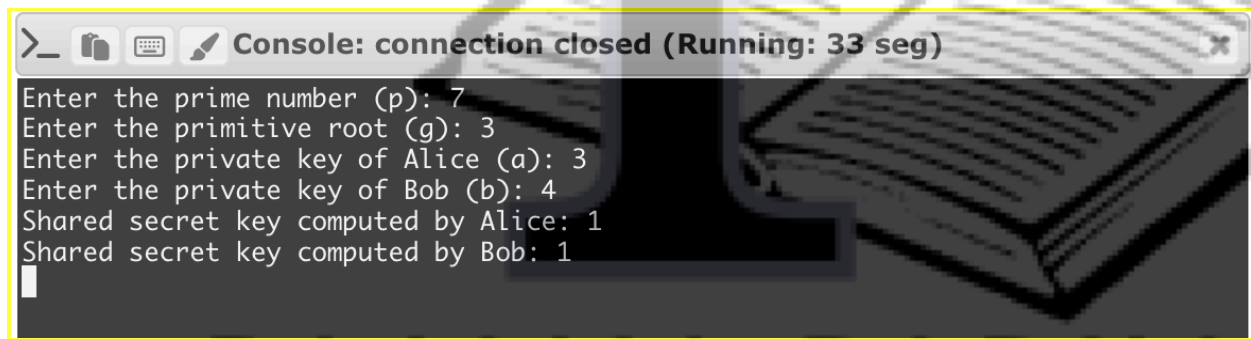
The screenshot shows a C program named 'diffie.c' in a code editor. The program implements the Diffie-Hellman key exchange protocol. It includes headers for `<stdio.h>` and `<math.h>`. It defines variables for prime number `p`, primitive root `g`, private keys `a` and `b`, public keys `A` and `B`, and shared secret keys `secret_key_A` and `secret_key_B`. The `main` function prompts the user to input `p`, `g`, `a`, and `b`. It then calculates the public keys `A = g^a mod p` and `B = g^b mod p`. Finally, it calculates the shared secret keys `secret_key_A = B^a mod p` and `secret_key_B = A^b mod p`, and prints them out. The user input values are 7 for `p`, 3 for `g`, 3 for `a`, and 4 for `b`, resulting in shared secret keys of 1 for both Alice and Bob.

```
1 #include <stdio.h>
2 #include <math.h>
3
4 // p: prime number
5 // g: primitive root of p
6 // a: private key of Alice
7 // b: private key of Bob
8 // A: public key of Alice
9 // B: public key of Bob
10 // secret_key: shared secret key
11
12 int main() {
13     // Input prime number (p) and primitive root (g) from user
14     int p, g;
15     printf("Enter the prime number (p): ");
16     scanf("%d", &p);
17     printf("Enter the primitive root (g): ");
18     scanf("%d", &g);
19
20     // Input private keys from Alice and Bob
21     int a, b;
22     printf("Enter the private key of Alice (a): ");
23     scanf("%d", &a);
24     printf("Enter the private key of Bob (b): ");
25     scanf("%d", &b);
26
27     // Calculate public keys
28     int A = fmod(pow(g, a), p);
29     int B = fmod(pow(g, b), p);
30
31     // Calculate shared secret key
32     int secret_key_A = fmod(pow(B, a), p);
33     int secret_key_B = fmod(pow(A, b), p);
34
35     // Output shared secret key
36     printf("Shared secret key computed by Alice: %d\n", secret_key_A);
37     printf("Shared secret key computed by Bob: %d\n", secret_key_B);
38
39     return 0;
40 }
41
```

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:



The screenshot shows the output of the program in a console window. The title bar reads 'Console: connection closed (Running: 33 seg)'. The output text matches the expected results from the previous block: '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', and '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:

```
#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  $x = g^k \bmod p$ 
    int x_power_k = 1;
    for (int i = 0; i < k; i++) {
        x_power_k = (x_power_k * g) % p;
    }

    // Calculate  $r = x \bmod q$ 
    int r = x_power_k % q;

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

        // Calculate  $s = k^{-1} * (m + x * r) \bmod q$ 
        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:

The screenshot shows a C program in a code editor. The program implements a digital signature algorithm. It takes inputs for prime numbers p and q, a primitive root g, a private key x, and a message m. It then calculates a random number k, computes the signature components (19, 30), and prints the generated signature. The code is as follows:

```
1 #include <stdio.h>
2
3 int main() {
4     // Input prime number (p), prime number (q), primitive root (g), private key (x), and message (m) from user
5     int p, q, g, x, m, k;
6     printf("Enter the prime number (p): ");
7     scanf("%d", &p);
8     printf("Enter the prime number (q): ");
9     scanf("%d", &q);
10    printf("Enter the primitive root (g): ");
11    scanf("%d", &g);
12    printf("Enter the private key (x): ");
13    scanf("%d", &x);
14    printf("Enter the message (m): ");
15    scanf("%d", &m);
16    printf("Enter the random number (k) in range 1 to q-1: ");
17    scanf("%d", &k);
18
19    // Calculate x = g^k mod p
20    int x_power_k = 1;
21    for (int i = 0; i < k; i++) {
22        x_power_k = (x_power_k * g) % p;
23    }
24
25    // Calculate r = x mod q
26    int r = x_power_k % q;
27
28    // Check if r is not equal to 0
29    if (r != 0) {
30        // Calculate k^-1 mod q
31        int k_inverse = 1;
32        for (int i = 1; i <= q - 2; i++) {
33            if ((k * i) % (q - 1) == 1) {
34                k_inverse = i;
35                break;
36            }
37        }
38
39        // Calculate s = k^-1 * (m + x * r) mod q
40        int s = (k_inverse * (m + x * r)) % q;
41        if (s < 0) {
42            s += q; // Add q to s if it's negative
43        }
44    }
```

The right sidebar contains a 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:

The screenshot shows a console window titled "Console: connection closed (Running: 23 seg)". The output of the program is displayed as follows:

```
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);

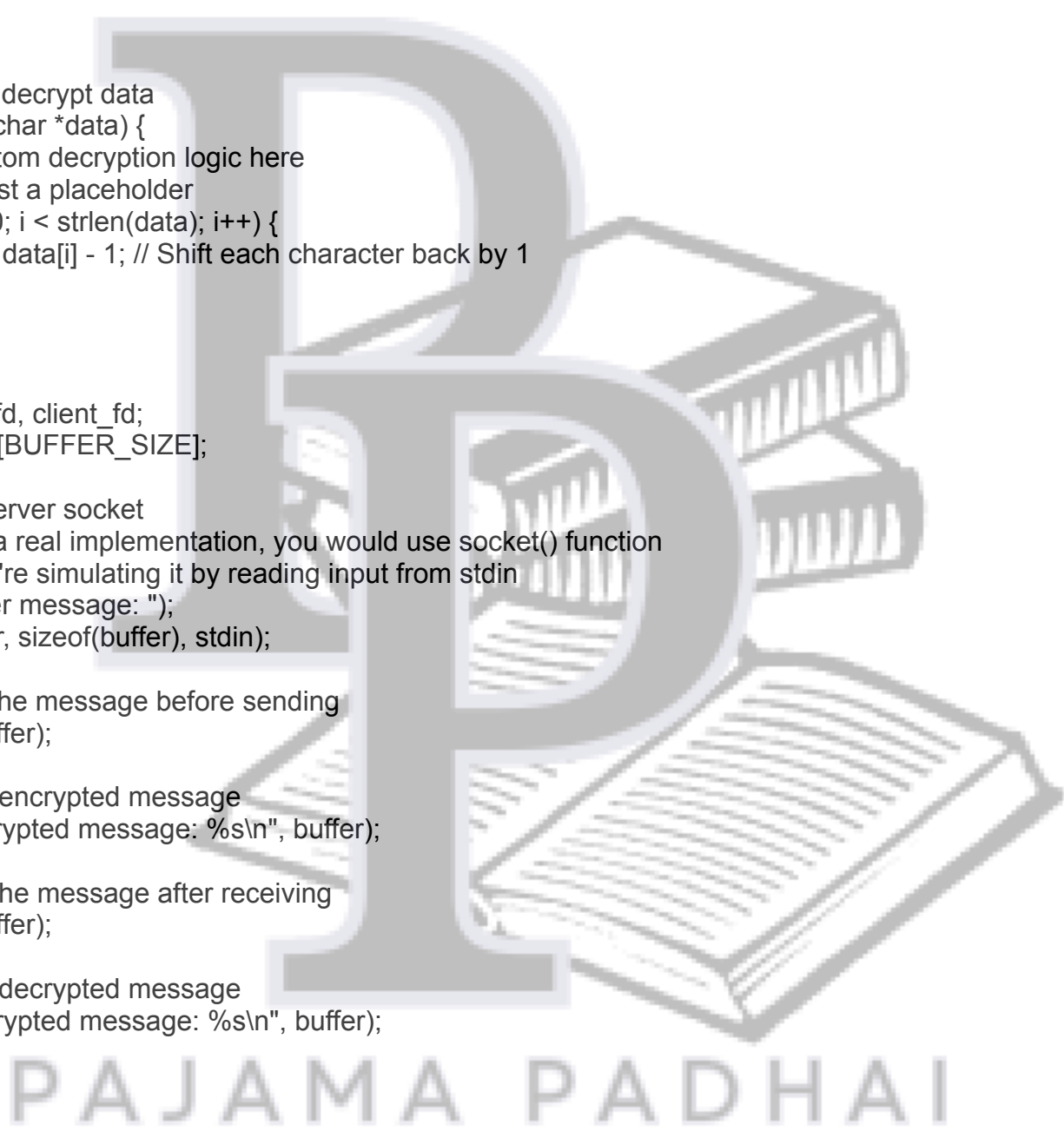
    // Encrypt the message before sending
    encrypt(buffer);

    // 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:

ssl.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 #define PORT 8080
6 #define BUFFER_SIZE 1024
7
8 // Function to encrypt data
9 void encrypt(char *data) {
10     // Your custom encryption logic here
11     // This is just a placeholder
12     for (int i = 0; i < strlen(data); i++) {
13         data[i] = data[i] + 1; // Shift each character by 1
14     }
15 }
16
17 // Function to decrypt data
18 void decrypt(char *data) {
19     // Your custom decryption logic here
20     // This is just a placeholder
21     for (int i = 0; i < strlen(data); i++) {
22         data[i] = data[i] - 1; // Shift each character back by 1
23     }
24 }
25
26 int main() {
27     int server_fd, client_fd;
28     char buffer[BUFFER_SIZE];
29
30     // Create server socket
31     // Note: In a real implementation, you would use socket() function
32     // Here, we're simulating it by reading input from stdin
33     printf("Enter message: ");
34     fgets(buffer, sizeof(buffer), stdin);
35
36     // Encrypt the message before sending
37     encrypt(buffer);
38
39     // Print the encrypted message
40     printf("Encrypted message: %s\n", buffer);
41
42     // Decrypt the message after receiving
43     decrypt(buffer);
44 }
```

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:

