Lab Manual

INFORMATION SECURITY

1) Write a C program that contains a string(char pointer) with a value\Hello World'. The programsshould XOR each character in thisstring with 0 and display the result.

2) Write a C program that contains a string (char pointer) with a value \Hello World'. The program should AND or and XOR each character in this string with 127 and display the result.

3) Write a Java program to perform encryption and decryption using the following algorithms:

i. Ceaser Cipher

ii. Substitution Cipher

iii. Hill Cipher

4) Write a Java program to implement the DES algorithm logic.

5) Write a C/JAVA program to implement the Blowfish algorithm logic.

6) Write a C/JAVA program to implement the Rijndael algorithm logic.

7) Write the RC4 logic in Java Using Java Cryptography, encrypt text "Hello world" usingBlowfish. Create your own key using Java key tool.

8) Write a Java program to implement RSA Algorithm.

9) Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript.

10) Calculate the message digest of a text using the SHA-1 algorithm in JAVA.

11) Calculate the message digest of a text using the MD5 algorithm in JAVA

Lab 1

Write a C program that contains a string(char pointer) with a value\Hello World'. The programsshould XOR each character in thisstring with 0 and display the result.

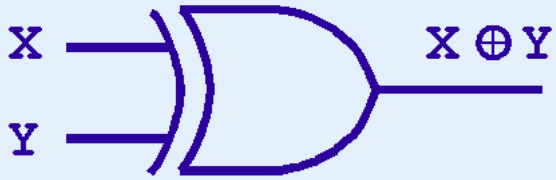
```
class XORExample {
    public static void main(String[] args) {
        String str = "Hello World";
        char[] str1 = new char[str.length()];
        System.out.println("Before XOR with 0: " + str);
      System.out.println("After XOR with 0: " );
        for (int i = 0; i < str.length(); i++) {
            str1[i] = (char) (str.charAt(i) ^ 0);
            System.out.print(str1[i]);
        }
        System.out.println();
    }
}
```

Output:

```
C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab1_XOR_0>java XORExamp
le
Before XOR with 0: Hello World
After XOR with 0:
Hello World
```

X XOR Y

X	Y	X⊕Y
0	0	0
0	1	1
1	0	1
1	1	0



Bitwise XOR of a character or number with 0 is that character or number itself.

For example: 0100 1000

^ 0000 0000

0100 1000

Lab2

2) Write a C program that contains a string (char pointer) with a value \Hello World'. The program should AND or and XOR each character in this string with 127 and display the result.

```
public class ANDExample {

    public static void main(String[] args) {
        String str = "Hello World";
        char[] str1 = new char[str.length()];

        int len = str.length();
      System.out.println("Before AND Operator: "+ str);
      System.out.println("After AND Operator: "+ str);

        // Apply bitwise AND with 127
        for (int i = 0; i < len; i++) {
            str1[i] = (char) (str.charAt(i) & 127);
            System.out.print(str1[i]);
        }

        System.out.println();
    }
}
```

Output:

```
C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab2_XOR_1>java ANDExamp
le
Before AND Operator: Hello World
After AND Operator: Hello World
Hello World
```

Truth Table of Bitwise AND (&) Operator

X	Y	X & Y
0	0	0
0	1	0
1	0	0
1	1	1

Bitwise AND of a character or number with 127 is that character or number itself.

For example: 0100 1000

&0111 1111

0100 1000

Lab 3

3) Write a Java program to perform encryption and decryption using the following algorithms:

i. Ceaser Cipher

ii. Substitution Cipher

iii. Hill Cipher

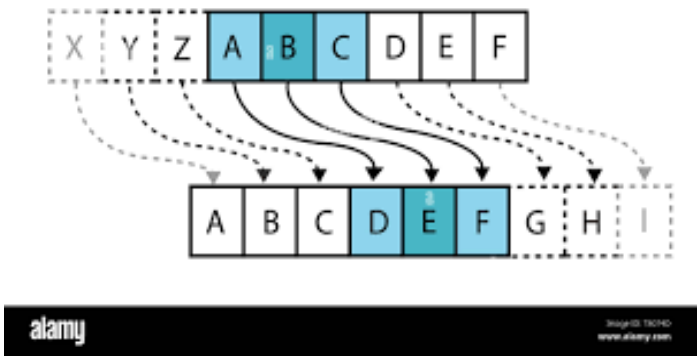
i. Ceaser Cipher :

code:-

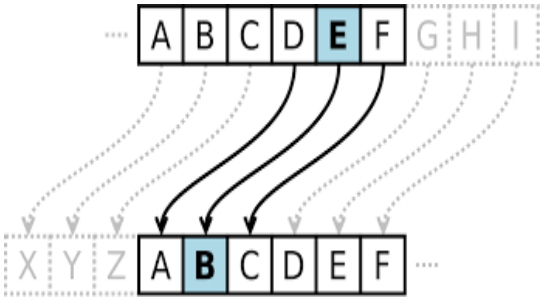
```
import java.util.Scanner;

public class CaesarCipher {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter text to encrypt with Caesar Cipher: ");

        String caesarText = scanner.nextLine();

        System.out.print("Enter the shift value: ");

        int caesarShift = scanner.nextInt();

        String caesarEncrypted = caesarEncrypt(caesarText,
caesarShift);

        System.out.println("Encrypted Text: " + caesarEncrypted);

        String caesarDecrypted = caesarDecrypt(caesarEncrypted,
caesarShift);

        System.out.println("Decrypted Text: " + caesarDecrypted);

        scanner.close();

    }


    private static String caesarEncrypt(String text, int shift) {

        StringBuilder result = new StringBuilder();

        for (char ch : text.toCharArray()) {

            if (Character.isLetter(ch)) {
```

```
                char base = Character.isUpperCase(ch) ? 'A' : 'a';
                result.append((char) ((ch - base + shift) % 26 + base));
            } else {
                result.append(ch);
            }
        }
        return result.toString();
    }

    private static String caesarDecrypt(String text, int shift) {
        return caesarEncrypt(text, 26 - shift);
    }
}
```

```
C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab3_Subs_Cipher>java Ca
esarCipher
Enter text to encrypt with Caesar Cipher: InfoSec
Enter the shift value: 3
Encrypted Text: LqirVhf
Decrypted Text: InfoSec
```



Encryption



Decryption

ii. Substitution Cipher (Monoalphabetic Cipher)

CODE

```
import java.util.Scanner;


public class MonoalphabeticCipher {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter text to encrypt with Monoalphabetic
Cipher: ");
        String plaintext = scanner.nextLine();
        String key = generateRandomKey();
        System.out.println("Generated Key: " + key);
        String ciphertext = encrypt(plaintext, key);
        System.out.println("Encrypted Text: " + ciphertext);
        String decryptedText = decrypt(ciphertext, key);
        System.out.println("Decrypted Text: " + decryptedText);
        scanner.close();
    }
    private static String generateRandomKey() {
        // This is just an example key, you may use a more secure
key generation method
        String alphabet = "abcdefghijklmnopqrstuvwxyz";
        String shuffledAlphabet = shuffleString(alphabet);
        return shuffledAlphabet;
    }
    private static String shuffleString(String input) {
        char[] characters = input.toCharArray();
        for (int i = characters.length - 1; i > 0; i--) {
            int randomIndex = (int) (Math.random() * (i + 1));
            char temp = characters[i];
```

```
            characters[i] = characters[randomIndex];
            characters[randomIndex] = temp;
        }
        return new String(characters);
    }
    private static String encrypt(String plaintext, String key) {
        StringBuilder ciphertext = new StringBuilder();
        for (char ch : plaintext.toCharArray()) {
            if (Character.isLowerCase(ch)) {
                ciphertext.append(key.charAt(ch - 'a'));
            } else {
                ciphertext.append(ch);
            }
        }
        return ciphertext.toString();
    }
    private static String decrypt(String ciphertext, String key) {
        StringBuilder plaintext = new StringBuilder();
        for (char ch : ciphertext.toCharArray()) {
            if (Character.isLowerCase(ch)) {
                plaintext.append((char) ('a' + key.indexOf(ch)));
            } else {
                plaintext.append(ch);
            }
        }
        return plaintext.toString();
    }
}
```

```
C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab3_Subs_Cipher>java Mo
noalphabeticCipher
Enter text to encrypt with Monoalphabetic Cipher: eastorwestaceisbest
Generated Key: mdfktvownyqephargjczuilxbs
Encrypted Text: tmczajltczmftncdtcz
Decrypted Text: eastorwestaceisbest
```

iii. Hill Cipher

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;

public class HillCipher {

    static float[][] decrypt = new float[3][1];
    static float[][] a = new float[3][3];
    static float[][] b = new float[3][3];
    static float[][] mes = new float[3][1];
    static float[][] res = new float[3][1];
    static BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
    static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) throws IOException {
        getKeyAndMessage();

        // Encryption
        encrypt();

        System.out.print("\nEncrypted string is: ");
        for (int i = 0; i < 3; i++) {
            System.out.print((char) ((res[i][0] % 26) + 97));
            mes[i][0] = res[i][0];
        }

        // Decryption
```

```
        inverse();
        decrypt();

        System.out.print("\nDecrypted string is: ");
        for (int i = 0; i < 3; i++) {
            System.out.print((char) ((decrypt[i][0] % 26) + 97));
        }

        System.out.println("\n");
    }

    public static void getKeyAndMessage() throws IOException {
        System.out.println("Enter a 3x3 matrix for the key (It
should be invertible): ");
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                a[i][j] = sc.nextFloat();

        System.out.print("\nEnter a 3-letter string: ");
        String msg = br.readLine();
        for (int i = 0; i < 3; i++)
            mes[i][0] = msg.charAt(i) - 97;
    }
    public static void encrypt() {
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 1; j++)
                for (int k = 0; k < 3; k++) {
                    res[i][j] = res[i][j] + a[i][k] * mes[k][j];
                }
    }
```

```
public static void inverse() {
    float p, q;
    float[][] c = new float[3][3];

    // Copy matrix 'a' to 'c'
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            c[i][j] = a[i][j];
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++) {
            if (i == j)
                b[i][j] = 1;
            else
                b[i][j] = 0;
        }

    for (int k = 0; k < 3; k++) {
        for (int i = 0; i < 3; i++) {
            p = c[i][k];
            q = c[k][k];
            for (int j = 0; j < 3; j++) {
                if (i != k) {
                    c[i][j] = c[i][j] * q - p * c[k][j];
                    b[i][j] = b[i][j] * q - p * b[k][j];
                }
           }
        }
    }

    for (int i = 0; i < 3; i++)
```

```
            for (int j = 0; j < 3; j++) {
                b[i][j] = b[i][j] / c[i][i];
            }
    }
    public static void decrypt() {
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 1; j++)
                for (int k = 0; k < 3; k++) {
                    decrypt[i][j] = decrypt[i][j] + b[i][k] *
res[k][j];
                }
    }
}
```

Output:

```
C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab3_Subs_Cipher>javac H
illCipher.java

C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab3_Subs_Cipher>java Hi
llCipher
Enter a 3x3 matrix for the key (It should be invertible):
3  1  4
1  5  9
2  6  5

Enter a 3-letter string: ace

Encrypted string is: sug
Decrypted string is: ace
```

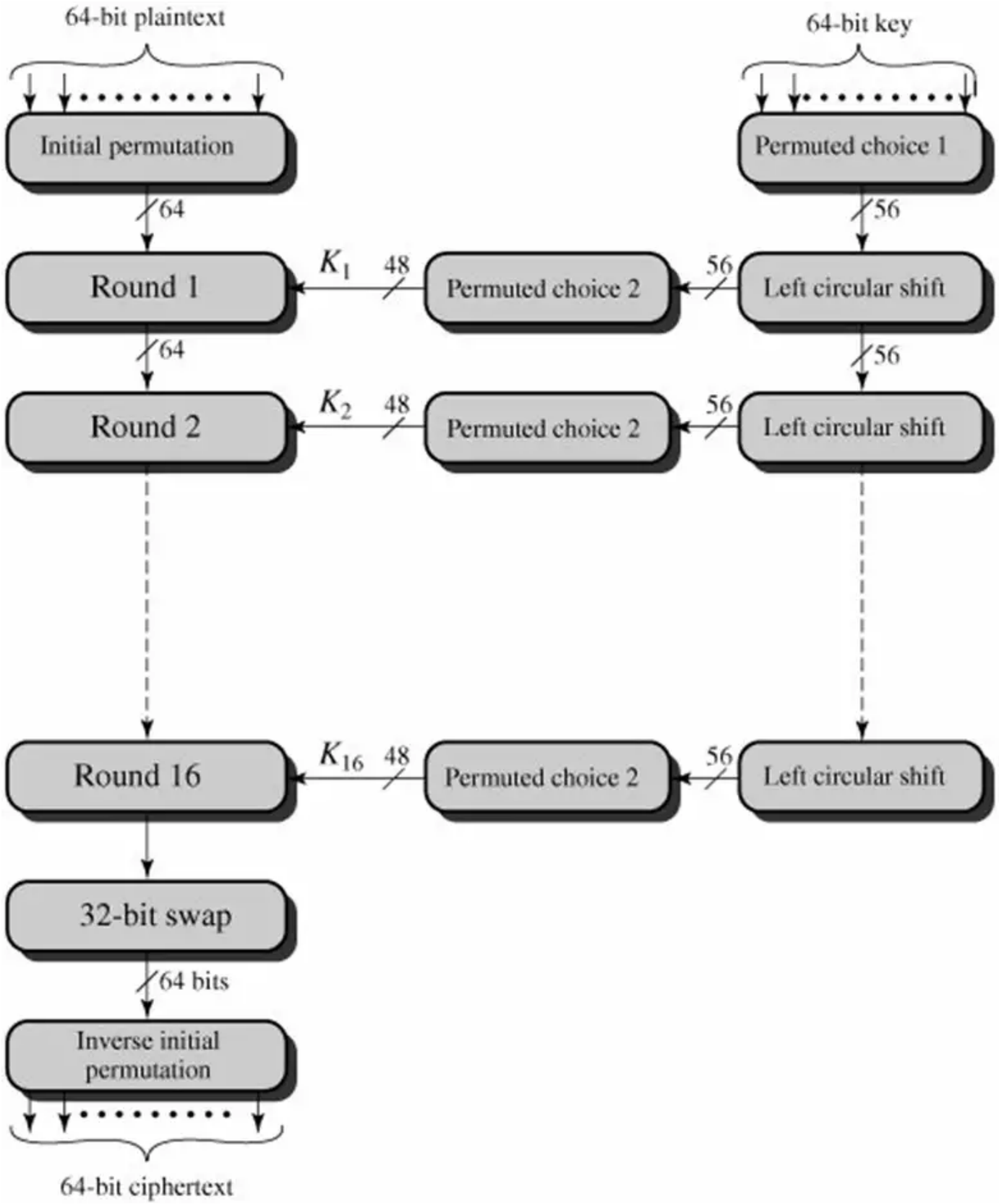
Lab 4

4) Write a Java program to implement the DES algorithm logic.

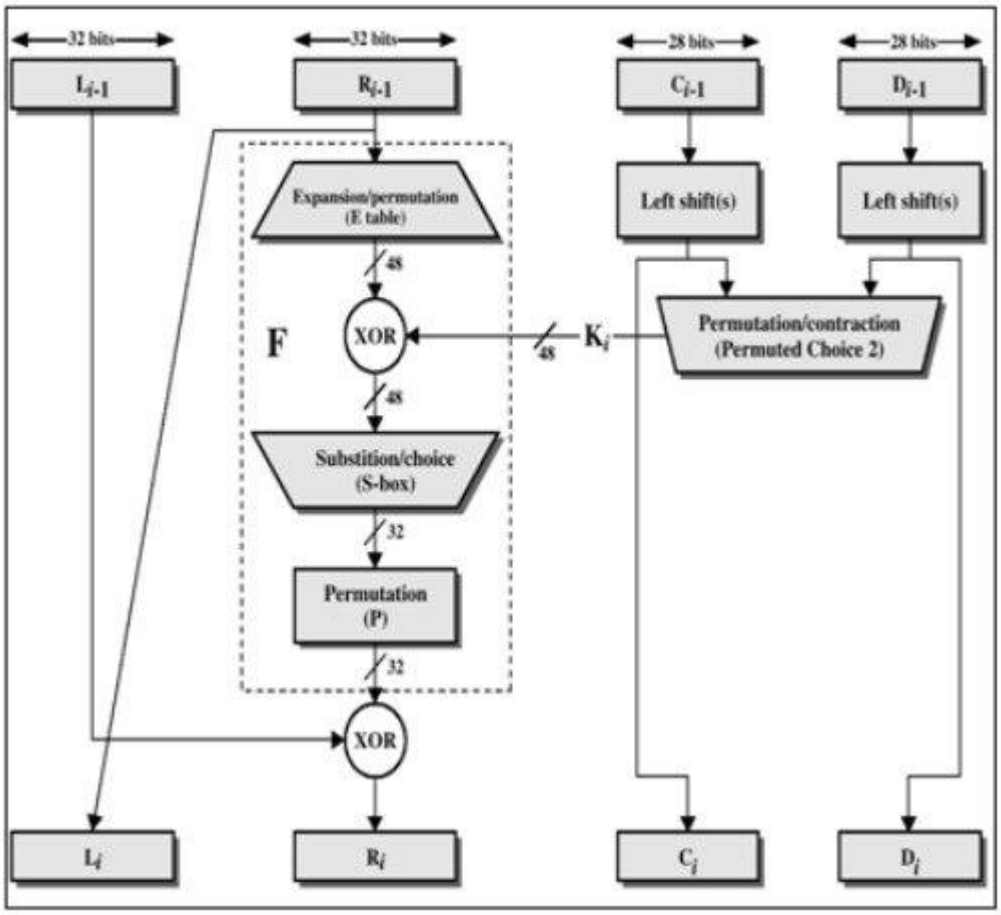
```
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESKeySpec;
import java.util.Base64;
public class DESEncryption {
    public static void main(String[] args) {
        try {
            // Example plaintext and key
            String plaintext = "Hello, DES!";
            String key = "12345678"; // 8-byte key for DES

            // Convert key to bytes
            byte[] keyBytes = key.getBytes("UTF-8");

            // Create DES key
            DESKeySpec desKeySpec = new DESKeySpec(keyBytes);
            SecretKeyFactory keyFactory =
SecretKeyFactory.getInstance("DES");
            SecretKey secretKey =
keyFactory.generateSecret(desKeySpec);

            // Create DES cipher
            Cipher cipher =
Cipher.getInstance("DES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);

            // Encrypt the plaintext
```

```
            byte[] encryptedBytes =
cipher.doFinal(plaintext.getBytes("UTF-8"));


            // Encode the encrypted bytes in Base64 for easier
display
            String encryptedText =
Base64.getEncoder().encodeToString(encryptedBytes);


            // Display results
            System.out.println("Original Text: " + plaintext);
            System.out.println("Encrypted Text: " + encryptedText);


            // Decrypt the ciphertext
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            byte[] decryptedBytes =
cipher.doFinal(Base64.getDecoder().decode(encryptedText));
            String decryptedText = new String(decryptedBytes, "UTF-
8");
            System.out.println("Decrypted Text: " + decryptedText);


        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab4_DES>javac DESEncryp
tion.java

C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab4_DES>java DESEncrypt
ion
Original Text: Hello, DES!
Encrypted Text: fy7lEyzVC0iQwyShwfm6Vg==
Decrypted Text: Hello, DES!
```


64-bit plaintext
64-bit key
Initial permutation
Permuted choice 1
64
56
Round 1
K_1
48
Permuted choice 2
56
Left circular shift
64
56
Round 2
K_2
48
Permuted choice 2
56
Left circular shift
Round 16
K_{16}
48
Permuted choice 2
56
Left circular shift
32-bit swap
64 bits
Inverse initial permutation
64-bit ciphertext

DES



Single Round of DES

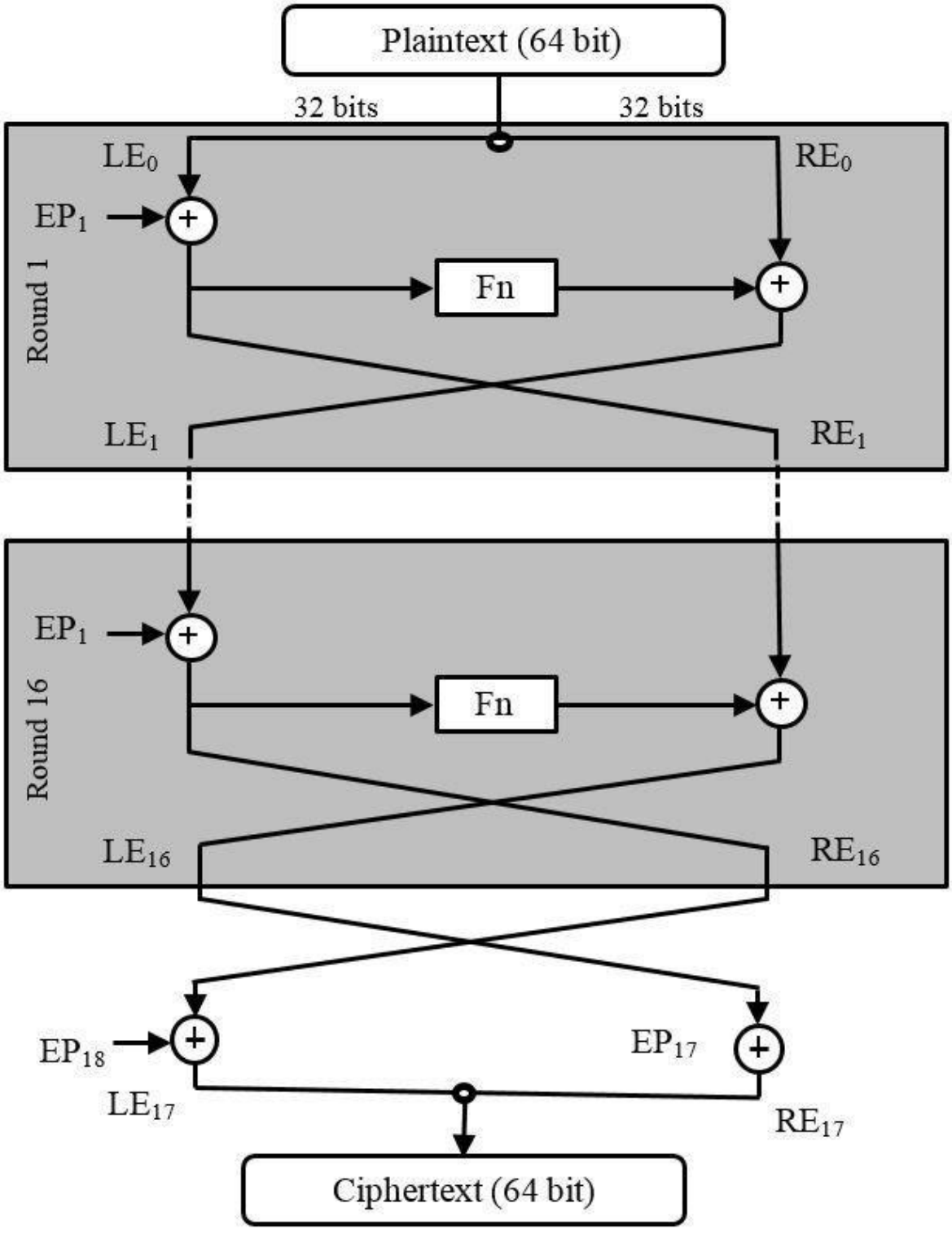
Lab 5

5) Write a C/JAVA program to implement the Blowfish algorithm logic.

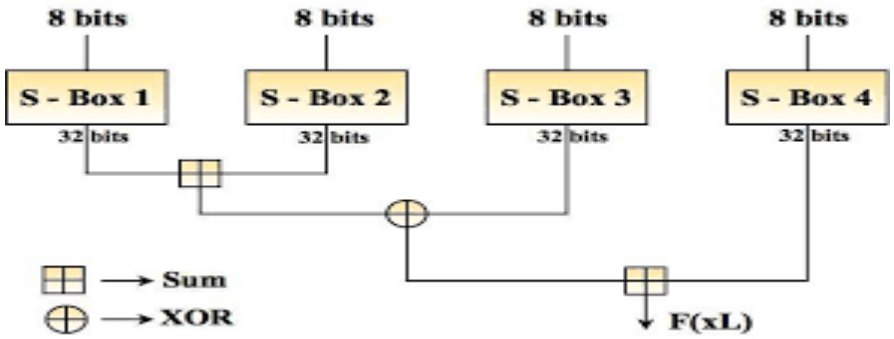
```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.security.Key;
import java.util.Base64;
class BlowFishEncryption {
    public static void main(String[] args) {
        try {
            // Step 1: Generate a Blowfish key using Java keytool
            Key secretKey = generateBlowfishKey();

            // Step 2: Create a Blowfish cipher and initialize it with the key for encryption
            Cipher cipher = Cipher.getInstance("Blowfish");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);

            // Step 3: Encrypt the text "Hello world"
            String plaintext = "Hello world";
            byte[] encryptedBytes = cipher.doFinal(plaintext.getBytes(StandardCharsets.UTF_8));

            // Step 4: Display the encrypted text
            String encryptedText = Base64.getEncoder().encodeToString(encryptedBytes);
          System.out.println("Plain Text: " + plaintext);
            System.out.println("Encrypted Text: " + encryptedText);
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }

    private static Key generateBlowfishKey() throws Exception {
        // Use Java keytool to generate a Blowfish key
        KeyGenerator keyGenerator = 
KeyGenerator.getInstance("Blowfish");
        keyGenerator.init(128); // Specify key size (can be 128, 
192, or 256 bits)

        // Generate the key
        SecretKey secretKey = keyGenerator.generateKey();

        // Convert the key to bytes
        byte[] keyBytes = secretKey.getEncoded();

        // Create a SecretKeySpec from the key bytes
        return new SecretKeySpec(keyBytes, "Blowfish");
    }
}
```

```
C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab5_BlowFish>javac Blow
FishEncryption.java

C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab5_BlowFish>java BlowF
ishEncryption
Plain Text: Hello world
Encrypted Text: SNWrLxz+TOW0ghmJjKmSsg==
```



Blowfish Algorithm


8 bits
8 bits
8 bits
8 bits
S - Box 1
S - Box 2
S - Box 3
S - Box 4
32 bits
32 bits
32 bits
32 bits
Sum
XOR
F(xL)

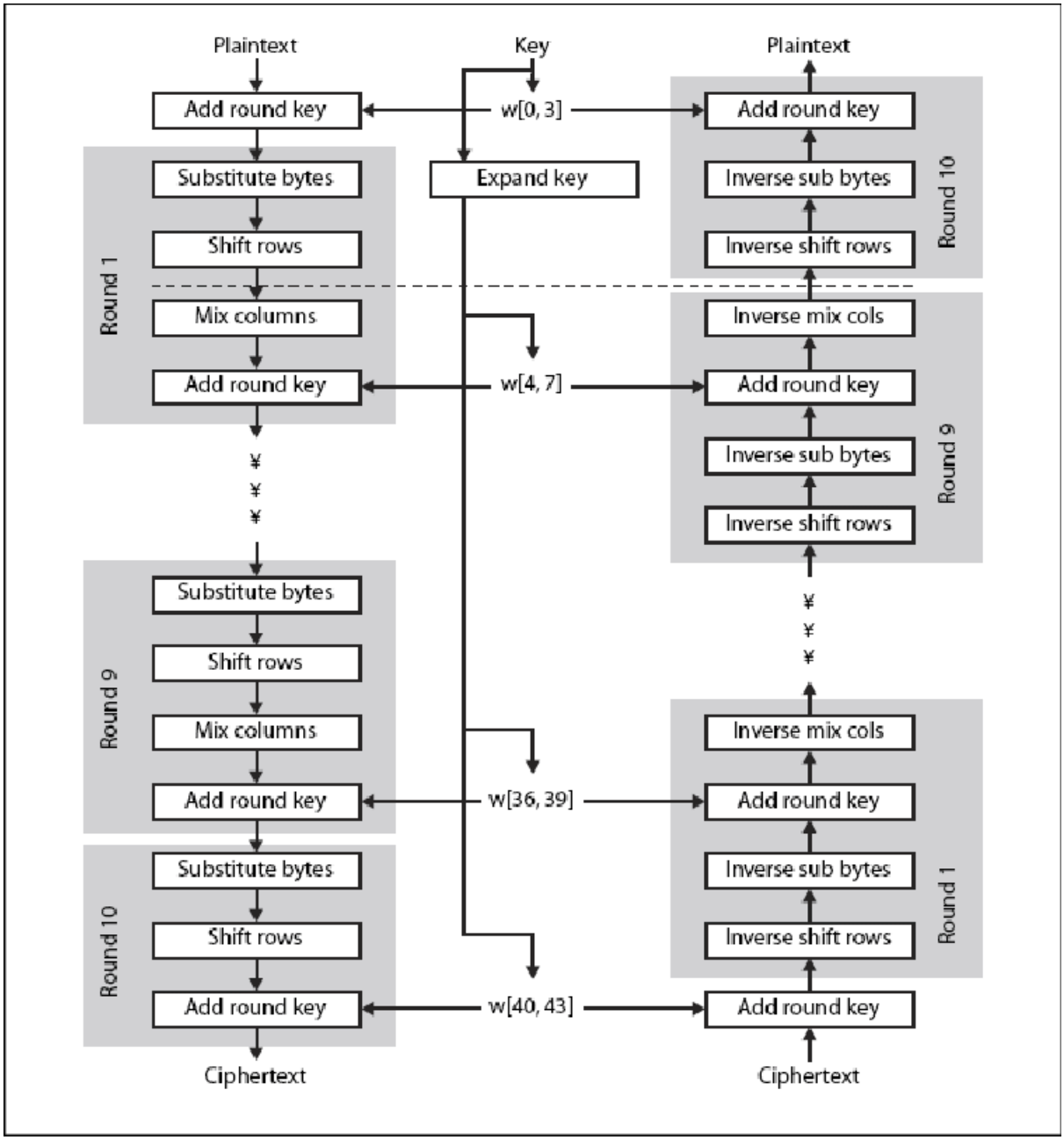
Lab 6

6) Write a C/JAVA program to implement the Rijndael(AES) algorithm logic.

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;
public class AESEncryption {

    public static void main(String[] args) {
        try {
            // Generate a random AES key
            SecretKey secretKey = generateAESKey();

            // Example plaintext
            String plaintext = "Hello, AES!";
          System.out.println("Plain Text: " + plaintext);

            // Encrypt the plaintext
            String encryptedText = encrypt(plaintext, secretKey);
            System.out.println("Encrypted Text: " + encryptedText);

            // Decrypt the ciphertext
            String decryptedText = decrypt(encryptedText,
secretKey);
            System.out.println("Decrypted Text: " + decryptedText);

        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }

    private static SecretKey generateAESKey() throws Exception {
        KeyGenerator keyGen = KeyGenerator.getInstance("AES");
        keyGen.init(128); // 128-bit key size for AES
        return keyGen.generateKey();
    }

    private static String encrypt(String plaintext, SecretKey
secretKey) throws Exception {
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedBytes =
cipher.doFinal(plaintext.getBytes());
        return Base64.getEncoder().encodeToString(encryptedBytes);
    }

    private static String decrypt(String ciphertext, SecretKey
secretKey) throws Exception {
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] decodedBytes =
Base64.getDecoder().decode(ciphertext);
        byte[] decryptedBytes = cipher.doFinal(decodedBytes);
        return new String(decryptedBytes);
    }
}
```

```
C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab6_AES>javac AESEncryp
tion.java

C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab6_AES>java AESEncrypt
ion
Plain Text: Hello, AES!
Encrypted Text: iT/KCOSbRmaUtP81peSdyw==
Decrypted Text: Hello, AES!
```


Plaintext
Key
Plaintext
Add round key
w[0, 3]
Add round key
Substitute bytes
Expand key
Inverse sub bytes
Shift rows
Inverse shift rows
Round 1
Round 10
Mix columns
Inverse mix cols
Add round key
w[4, 7]
Add round key
Inverse sub bytes
Round 9
Inverse shift rows
Substitute bytes
Shift rows
Inverse mix cols
Round 9
Mix columns
Add round key
w[36, 39]
Add round key
Substitute bytes
Inverse sub bytes
Shift rows
Inverse shift rows
Round 1
Round 10
Add round key
w[40, 43]
Add round key
Ciphertext
Ciphertext

AES

Lab 7

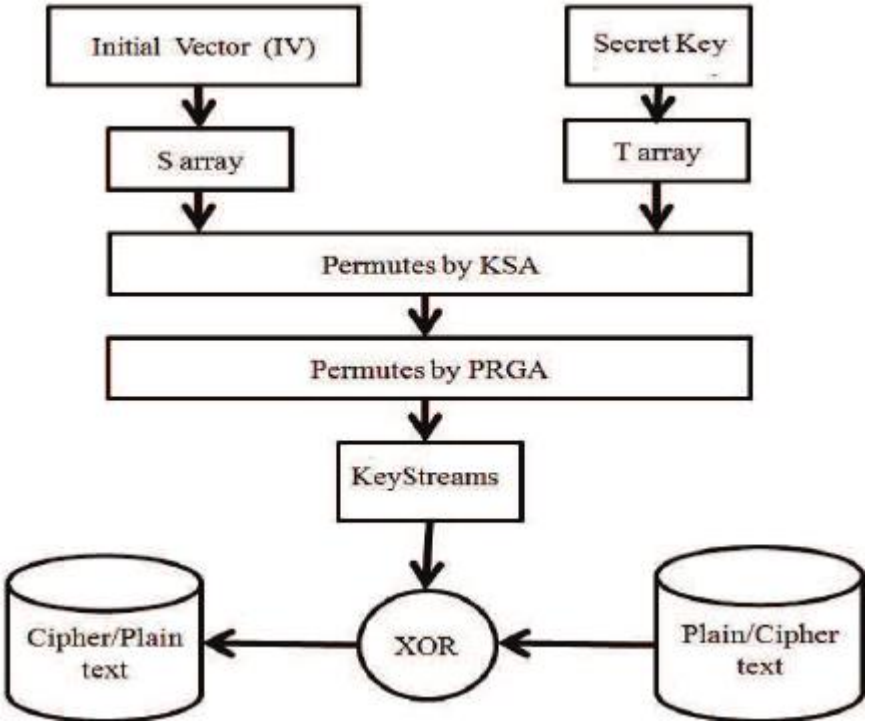
7) Write the RC4 logic in Java Using Java Cryptography, encrypt text “Hello world” usingBlowfish. Create your own key using Java key tool.

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.security.NoSuchAlgorithmException;

public class RC4Encryption {

    public static void main(String[] args) {
        try {
            // Example plaintext and key
            String plaintext = "Hello world";
            String key = "RC4_Key12345678";

            // Generate a secret key for RC4
            SecretKey secretKey = generateRC4Key(key);
          System.out.println("Plain Text: " + plaintext);
          System.out.println("Secret Key: " + secretKey);

            // Encrypt the plaintext using RC4
            String encryptedText = encryptRC4(plaintext, secretKey);
            System.out.println("Encrypted Text: " + encryptedText);

            // Decrypt the ciphertext using RC4
```

```
            String decryptedText = decryptRC4(encryptedText,
secretKey);
            System.out.println("Decrypted Text: " + decryptedText);


        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    private static SecretKey generateRC4Key(String keyString) throws
NoSuchAlgorithmException {
        byte[] keyBytes =
keyString.getBytes(StandardCharsets.UTF_8);


        KeyGenerator keyGenerator = KeyGenerator.getInstance("RC4");
        keyGenerator.init(keyBytes.length * 8);
        return new SecretKeySpec(keyBytes, "RC4");
    }


    private static String encryptRC4(String plaintext, SecretKey
secretKey) throws Exception {
        Cipher cipher = Cipher.getInstance("RC4");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);


        byte[] encryptedBytes =
cipher.doFinal(plaintext.getBytes(StandardCharsets.UTF_8));
        return bytesToHex(encryptedBytes);
    }


    private static String decryptRC4(String ciphertext, SecretKey
secretKey) throws Exception {
        Cipher cipher = Cipher.getInstance("RC4");
```

```
        cipher.init(Cipher.DECRYPT_MODE, secretKey);

        byte[] decodedBytes = hexToBytes(ciphertext);
        byte[] decryptedBytes = cipher.doFinal(decodedBytes);
        return new String(decryptedBytes, StandardCharsets.UTF_8);
    }

    private static String bytesToHex(byte[] bytes) {
        StringBuilder hexString = new StringBuilder();
        for (byte b : bytes) {
            String hex = Integer.toHexString(0xFF & b);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }
        return hexString.toString();
    }

    private static byte[] hexToBytes(String hex) {
        int length = hex.length();
        byte[] data = new byte[length / 2];
        for (int i = 0; i < length; i += 2) {
            data[i / 2] = (byte) ((Character.digit(hex.charAt(i),
16) << 4)
                    + Character.digit(hex.charAt(i + 1), 16));
        }
        return data;
    }
}
```

```
C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab7_RC4>javac RC4Encryp
tion.java

C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab7_RC4>java RC4Encrypt
ion
Plain Text: Hello world
Secret Key: javax.crypto.spec.SecretKeySpec@1a104
Encrypted Text: aa9f48f1df8a48643d2cb7
Decrypted Text: Hello world
```



RC4

Lab 8

8) Write a Java program to implement RSA Algorithm.

```
import java.util.Random;
public class SimpleRSA {

    private static final long p = 61; // prime number
    private static final long q = 53; // Another prime number
    private static final long modulus = p * q;
    private static final long phi = (p - 1) * (q - 1);

    private static final long publicKey = 17;
    private static final long privateKey = modInverse(publicKey,
phi);

    public static void main(String[] args) {
        // Example message to be encrypted
        long message = 42;

        // Encryption
        long ciphertext = modPow(message, publicKey, modulus);

        System.out.println("p: " + p);
        System.out.println("q: " + q);
        System.out.println("n: " + modulus);
        System.out.println("phi: " + phi);
        System.out.println("Public Key (e): " + publicKey);
        System.out.println("Private Key (d): " + privateKey);
        System.out.println("Original Message: " + message);
        System.out.println("Encrypted Message: " + ciphertext);
```

```
        // Decryption
        long decryptedMessage = modPow(ciphertext, privateKey,
modulus);
        System.out.println("Decrypted Message: " +
decryptedMessage);
    }

    // Utility method to calculate the modular inverse
    private static long modInverse(long a, long m) {
        for (long x = 1; x < m; x++) {
            if ((a * x) % m == 1) {
                return x;
            }
        }
        return 1; // Default return if no inverse exists
    }

    // Utility method for modular exponentiation (a^b mod m)
    private static long modPow(long base, long exponent, long
modulus) {
        long result = 1;
        base = base % modulus;

        while (exponent > 0) {
            if (exponent % 2 == 1) {
                result = (result * base) % modulus;
            }
            exponent = exponent >> 1;
            base = (base * base) % modulus;
        }
```

```
        return result;
    }
}
```

```
C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab8_RSA>javac SimpleRSA
.java

C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab8_RSA>java SimpleRSA
p: 61
q: 53
n: 3233
phi: 3120
Public Key (e): 17
Private Key (d): 2753
Original Message: 42
Encrypted Message: 2557
Decrypted Message: 42
```

Key Generation

Select p, q	p and q both prime
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \bmod \phi(n)$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Encryption

Plaintext	$M < n$
Ciphertext	$C = M^e \pmod{n}$

Decryption

Ciphertext	C
Plaintext	$M = C^d \pmod{n}$

RSA Algorithm

Lab 9

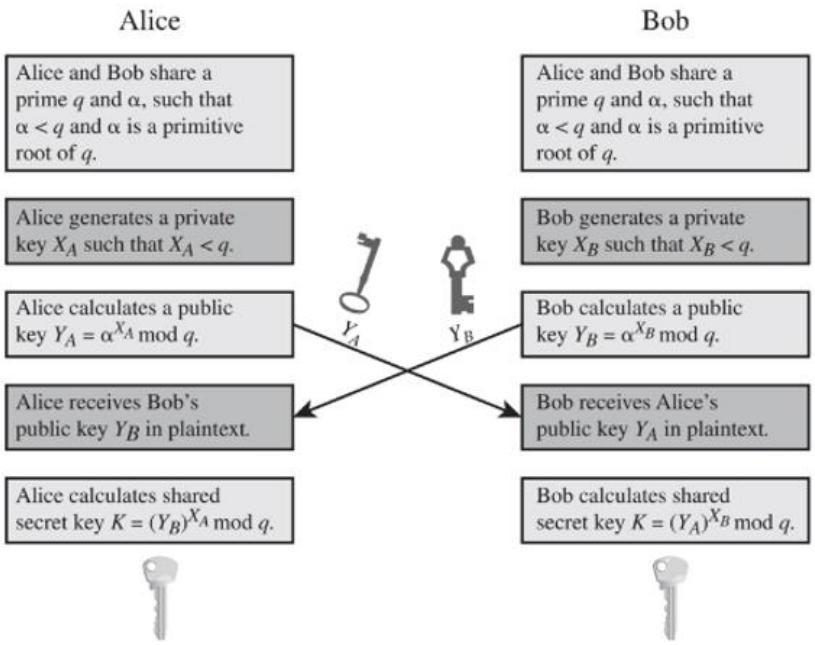
9) Implement the Diffie-Hellman Key Exchange mechanism.

```
import java.math.BigInteger;

import java.security.SecureRandom;

public class DiffieHellman2 {

    public static void main(String[] args) {

        // Prime number and primitive root for the example

        BigInteger prime = new BigInteger("23");

        BigInteger primitiveRoot = new BigInteger("5");


        // Alice's private key

        BigInteger alicePrivateKey =
getRandomInt(10).add(BigInteger.ONE); // Random number between 1 and
10


        // Calculate Alice's public key

        BigInteger alicePublicKey = calculatePublicKey(prime,
primitiveRoot, alicePrivateKey);


        // Bob's private key

        BigInteger bobPrivateKey =
getRandomInt(10).add(BigInteger.ONE); // Random number between 1 and
10


        // Calculate Bob's public key

        BigInteger bobPublicKey = calculatePublicKey(prime,
primitiveRoot, bobPrivateKey);


        // Exchange public keys (simulate over an insecure channel)

        BigInteger sharedKeyA = calculateSharedKey(prime,
bobPublicKey, alicePrivateKey);
```

```
        BigInteger sharedKeyB = calculateSharedKey(prime,
alicePublicKey, bobPrivateKey);


        // Display results
        System.out.println("Alice's private key: " +
alicePrivateKey);
        System.out.println("Alice's public key: " + alicePublicKey);
        System.out.println("Bob's private key: " + bobPrivateKey);
        System.out.println("Bob's public key: " + bobPublicKey);
        System.out.println("Shared secret (Alice): " + sharedKeyA);
        System.out.println("Shared secret (Bob): " + sharedKeyB);
    }


    private static BigInteger getRandomInt(int max) {
        SecureRandom random = new SecureRandom();
        return new BigInteger(max, random);
    }


    private static BigInteger modExp(BigInteger base, BigInteger
exponent, BigInteger modulus) {
        return base.modPow(exponent, modulus);
    }


    private static BigInteger calculatePublicKey(BigInteger prime,
BigInteger primitiveRoot, BigInteger privateKey) {
        return modExp(primitiveRoot, privateKey, prime);
    }


    private static BigInteger calculateSharedKey(BigInteger prime,
BigInteger publicKey, BigInteger privateKey) {
        return modExp(publicKey, privateKey, prime);
    }
```

```
}
```

```
C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab9_DH>javac DiffieHell
man2.java

C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab9_DH>java DiffieHellm
an2
Alice's private key: 226
Alice's public key: 8
Bob's private key: 309
Bob's public key: 5
Shared secret (Alice): 8
Shared secret (Bob): 8
```



Diffie Hellman Key Exchange

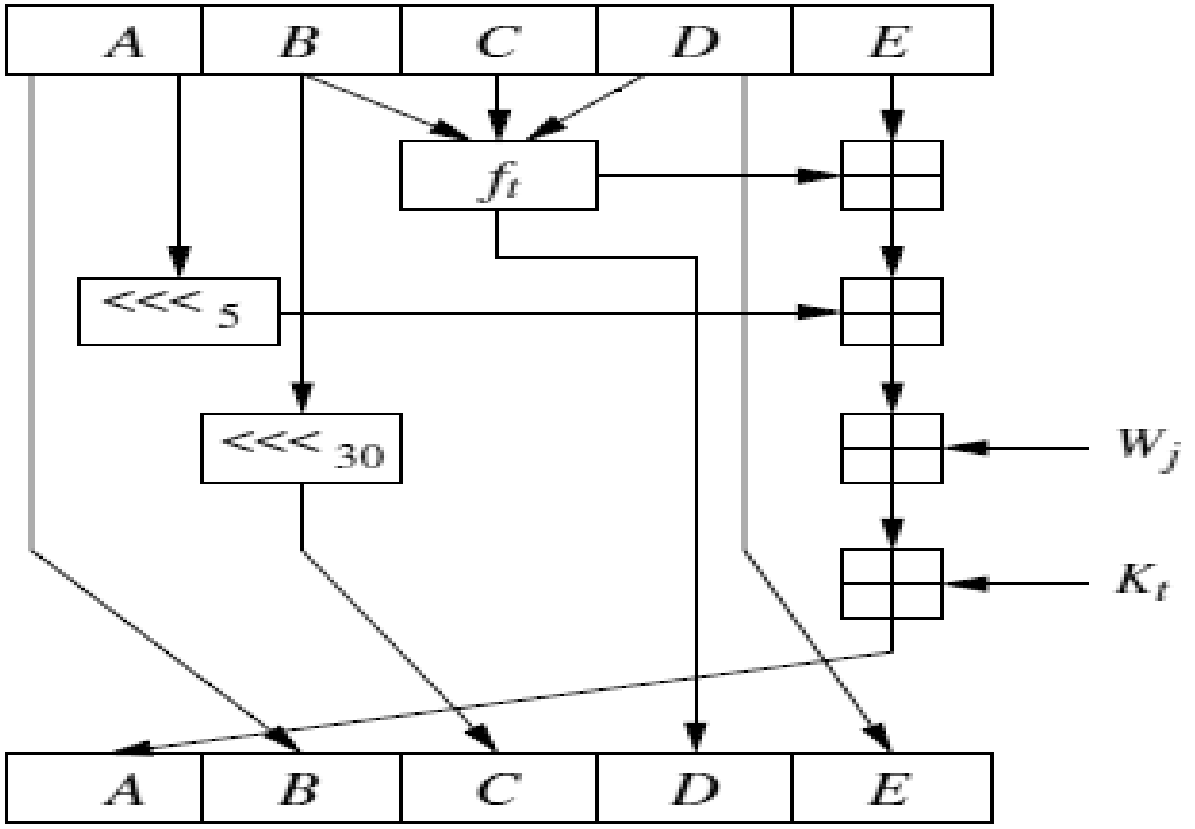
Lab 10

10) Calculate the message digest of a text using the SHA-1 algorithm in JAVA.

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class SHA1Example {

    public static void main(String[] args) {
        // Example input string
        String input = "Hello, SHA-1!";

        // Generate SHA-1 hash
        String sha1Hash = generateSHA1Hash(input);

        // Display the result
        System.out.println("Input String: " + input);
        System.out.println("SHA-1 Hash: " + sha1Hash);
    }

    private static String generateSHA1Hash(String input) {
        try {
            // Create a MessageDigest object for SHA-1
            MessageDigest sha1Digest =
MessageDigest.getInstance("SHA-1");

            // Update the message digest with the input bytes
            byte[] inputBytes = input.getBytes();
            sha1Digest.update(inputBytes);
```

```
            // Generate the hash
            byte[] hashedBytes = sha1Digest.digest();

            // Convert the byte array to a hexadecimal string
            StringBuilder stringBuilder = new StringBuilder();
            for (byte hashedByte : hashedBytes) {
                stringBuilder.append(Integer.toString((hashedByte &
0xff) + 0x100, 16).substring(1));
            }

            return stringBuilder.toString();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

```
C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes>cd Lab10_SHA1

C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab10_SHA1>javac SHA1Exa
mple.java

C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab10_SHA1>java SHA1Exam
ple
Input String: Hello, SHA-1!
SHA-1 Hash: f322e078fef4f49da1618d3793d3272a91f0488c
```



SHA-1

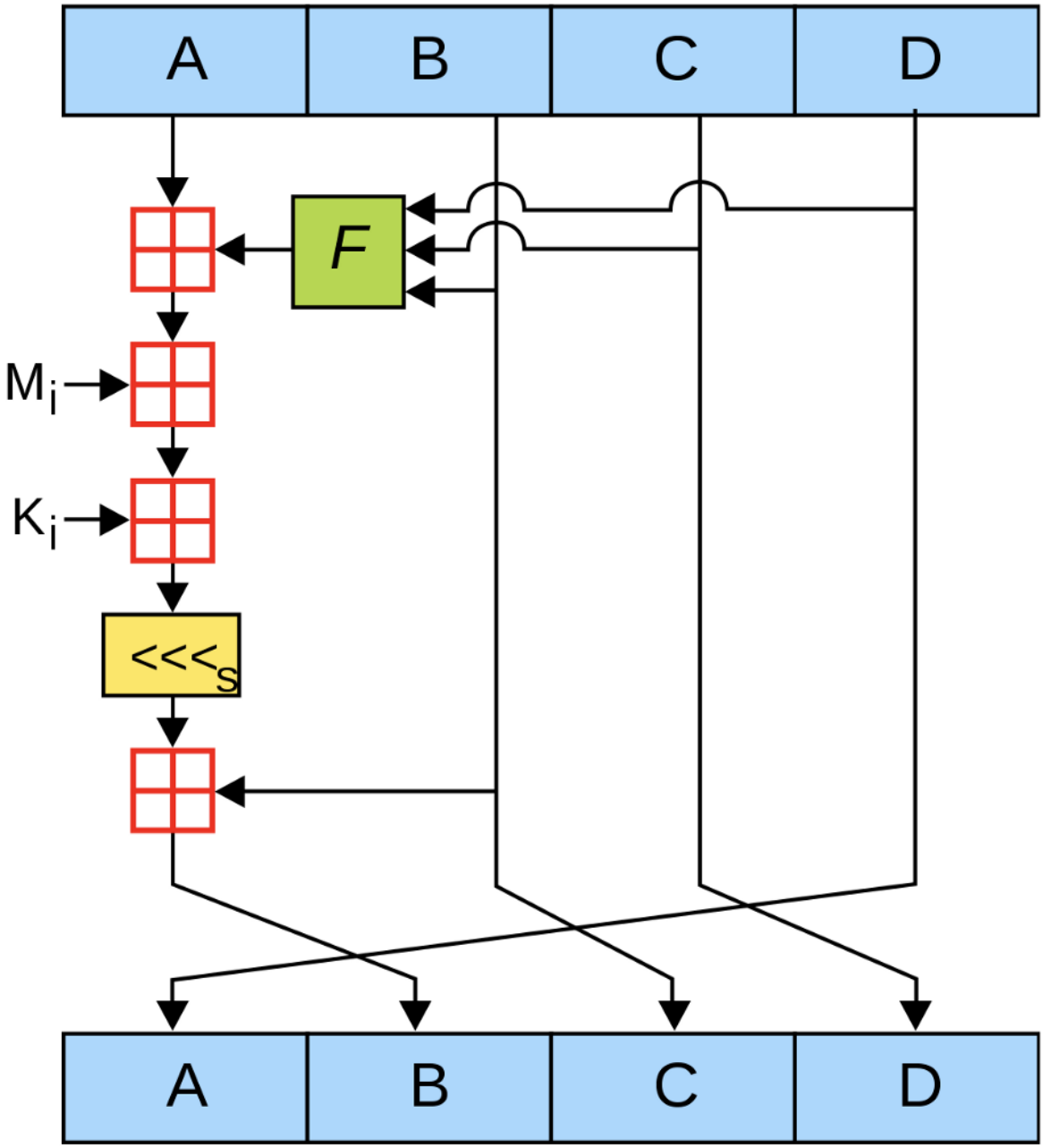
Lab 11

11) Calculate the message digest of a text using the MD5 algorithm in JAVA

```
import java.security.*;
public class MD5 {
public static void main(String[] a) {
// TODO code application logic here
try {
MessageDigest md = MessageDigest.getInstance("MD5");
System.out.println("Message digest object info: ");
System.out.println(" Algorithm = " +md.getAlgorithm());
System.out.println(" Provider = " +md.getProvider());
System.out.println(" ToString = " +md.toString());
String input = ""; md.update(input.getBytes());
byte[] output = md.digest();
System.out.println();
System.out.println("MD5(\""+input+"\") = " +bytesToHex(output));
input = "abc";
md.update(input.getBytes());
output = md.digest();
System.out.println();
System.out.println("MD5(\""+input+"\") = " +bytesToHex(output));
input = "abcdefghijklmnopqrstuvwxyz";
md.update(input.getBytes());
output = md.digest();
System.out.println();
System.out.println("MD5(\"" +input+"\") = "+bytesToHex(output));
```

```
System.out.println("");
}
catch (Exception e)
{ System.out.println("Exception: " +e); }
}
public static String bytesToHex(byte[] b) {
char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
'A', 'B', 'C', 'D', 'E', 'F'};
StringBuffer buf =new StringBuffer();
for (int j=0; j<b.length;j++)
{ buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
 buf.append(hexDigit[b[j] & 0x0f]);
 }
return buf.toString();
 }
}
```

```
C:\Users\sonus\Desktop\Information Security\Info_Sec_Codes\Lab11_MD5>java MD5
Message digest object info:
 Algorithm = MD5
 Provider = SUN version 1.8
 ToString = MD5 Message Digest from SUN, <initialized>


MD5("") = D41D8CD98F00B204E9800998ECF8427E

MD5("abc") = 900150983CD24FB0D6963F7D28E17F72

MD5("abcdefghijklmnopqrstuvwxyz") = C3FCD3D76192E4007DFB496CCA67E13B
```



MD5