Exp No. 10

Rishab Mandal

Batch: C23

Code:

Exp10A3Algo.java :

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

public class Exp10A3Algo {

    public static void main(String[] args) {
        // Simulate generating a random Ki (secret key) and RAND
(challenge)
        String ki = generateRandomHexString(32); // 128 bits (16 bytes) key
        String rand = generateRandomHexString(32); // 128 bits (16 bytes)
challenge

        // Display the generated Ki and RAND
        System.out.println("Ki (Secret Key): " + ki);
        System.out.println("RAND (Challenge): " + rand);
        // Calculate the expected response (SRES) using the A3 algorithm
        String sres = calculateSRES(ki, rand);

        // Display the calculated SRES
        System.out.println("SRES (Expected Response): " + sres);
    }

    private static String generateRandomHexString(int length) {
        Random random = new Random();
        StringBuilder randomHex = new StringBuilder();

        for (int i = 0; i < length; i++) {
            int randomInt = random.nextInt(16); // 0-15
            randomHex.append(Integer.toHexString(randomInt));
        }

        return randomHex.toString();
    }

    private static String calculateSRES(String ki, String rand) {
        try {
            // Concatenate Ki and RAND
            String input = ki + rand;

            // Use SHA-1 hash function to calculate SRES
            MessageDigest sha1 = MessageDigest.getInstance("SHA-1");
            byte[] hashBytes = sha1.digest(hexStringToByteArray(input));

            // Convert the hash to a hexadecimal string
            StringBuilder sres = new StringBuilder();
            for (byte b : hashBytes) {
                sres.append(String.format("%02X", b));
            }

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

    private static byte[] hexStringToByteArray(String hexString) {
        int len = hexString.length();
        byte[] data = new byte[len / 2];

        for (int i = 0; i < len; i += 2) {
            data[i / 2] = (byte) ((Character.digit(hexString.charAt(i), 16) << 4)
                    + Character.digit(hexString.charAt(i + 1), 16));
        }
```

```
        return data;
    }
}
```

Output:

```
(base) PS C:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents> cd "c:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents\" ; if ($?) { javac Exp10A3Algo.java } ; if ($?) { java Exp10A3Algo }
Ki (Secret Key): 75b3cd449c491cb7af27683de9dba3f8
RAND (Challenge): 5ed4d5250e87177fd4c4f4e9f9238cf4
SRES (Expected Response): 8A69046163903D54366D9AF1E410B40E56872AC6
```

Exp10A5Algo.java :

```java
import java.lang.Math;

public class A5 {
    static int[] GenerateBits() {
        int[] a = new int[16];
        for (int i = 0; i < 16; i++) {
            double rand = Math.random();
            if (rand >= 0.5) {
                a[i] = 1;
            } else {
                a[i] = 0;
        }
    }
    return a;
}

static int[] XOR(int[] a, int[] b) {
    int[] temp = new int[16];
    for (int i = 0; i < 16; i++) {
        if (a[i] == 1 && b[i] == 1 || a[i] == 0 && b[i] == 0) {
            temp[i] = 0;
        } else {
            temp[i] = 1;
        }
        System.out.print(temp[i]);
    }
    return temp;
}

static int[] AND(int[] a, int[] b) {
    int[] temp = new int[16];
    for (int i = 0; i < 16; i++) {
        if (a[i] == 1 && b[i] == 1) {
            temp[i] = 1;
        } else {
            temp[i] = 0;
        }
        System.out.print(temp[i]);
    }
    return temp;
}
```

```
public static void main(String[] args) {
    int[] a;
    System.out.println("Generating the 1st key identification number");
    a = GenerateBits();
    for (int i = 0; i < 16; i++) {
        System.out.print(a[i]);
    }
    int[] b;
    System.out.println("\n\nGenerating the 2nd key identification number");
    b = GenerateBits();
    for (int i = 0; i < 16; i++) {
        System.out.print(b[i]);
    }
    int[] c;
    System.out.println("\n\nGenerating the random number");
    c = GenerateBits();
    for (int i = 0; i < 16; i++) {
        System.out.print(c[i]);
    }
    int[] d;
    System.out.println("\n\nGenerating the barker code");
    d = GenerateBits();
    for (int i = 0; i < 16; i++) {
        System.out.print(d[i]);
    }
    int[] z;
    System.out.println("\n\nAND of 1st key and 2nd key");
    z = AND(a, b);
    int[] p;
```

```
        System.out.println("\n\nXOR of random number and the AND of 1st key and 2nd key");
        p = XOR(z, c);
        System.out.println("\n\nXOR of the above number and barker code");
        p = XOR(p, d);
        int[] q;
        System.out.println("\n\nXOR of random number and the AND of 1st key and 2nd key");
        q = XOR(z, c);
        System.out.println("\n\nXOR of the above number and barker code");
        q = XOR(q, d);
        int flag = 0;
        for (int i = 0; i < 16; i++) {
            if (p[i] != q[i]) {
                flag = 1;
                break;
            }
        }
        if (flag == 1) {
            System.out.print("\n\nEncryption Failed");
        } else {
            System.out.print("\n\nEncryption Passed");
        }

    }
}
```

Output:

```
(base) PS C:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents>
cd "c:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents\" ; if
($?) { javac Exp10A5Algo.java } ; if ($?) { java Exp10A5Algo }
```

Generating the 1st key identification number

0111001001011110

Generating the 2nd key identification number

1111110111000001

Generating the random number

1011000011011100

Generating the barker code

0001011100001111

AND of 1st key and 2nd key

0111000001000000

XOR of random number and the AND of 1st key and 2nd key

1100000010011100

XOR of the above number and barker code

1101011110010011

XOR of random number and the AND of 1st key and 2nd key

1100000010011100

```
XOR of the above number and barker code
1101011110010011

Encryption Passed
Keystream bit = 0 ^ 0 ^ 0 = 0

Keystream bit = 0 ^ 0 ^ 0 = 0

Keystream bit = 1 ^ 1 ^ 0 = 0

Keystream bit = 0 ^ 1 ^ 1 = 0

Keystream bit = 1 ^ 1 ^ 1 = 1

Keystream bit = 0 ^ 0 ^ 1 = 1
```

Exp10A8Algo.java :

```java
public class A8Algorithm {
    // Secret key (Ki)
    private static final int[] KI = { 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1 };
    // A8 Algorithm
    public static int[] generateKeyStream(int[] rand) {
        int[] keyStream = new int[rand.length];

        // Generate key stream
        for (int i = 0; i < rand.length; i++) {
            keyStream[i] = rand[i] ^ KI[i % KI.length];
        }

        return keyStream;
    }

    // Example usage
    public static void main(String[] args) {
        // Example random number (RAND)
        int[] rand = { 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0 };

        // Generate key stream using A8 algorithm
        int[] keyStream = generateKeyStream(rand);

        // Print key stream
        System.out.println("Key Stream:");
        for (int keyBit : keyStream) {
            System.out.print(keyBit);
        }
    }
}
```

Output:

```
(base) PS C:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents>
cd "c:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents\" ; if
($?) { javac Exp10A8Algo.java } ; if ($?) { java Exp10A8Algo }
Step 1: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1
Step 2: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1
Step 3: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1
Step 4: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1
Step 5: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1
Step 6: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1
Step 7: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1
Step 8: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1
Step 9: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1
Step 10: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1
Step 11: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1
Step 12: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1
Step 13: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1
Step 14: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1
Step 15: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1
Step 16: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1

Key Stream:
1111111111111111
```