Problem 2:

Let One-Time-Pad cipher is defined over the spaces C==P=K={0, 1} , where l is the length of the binary strings. Show that the cipher is not secure under the chosen-plaintext attack. Furthermore, the attack should be implemented in your preferred programming language

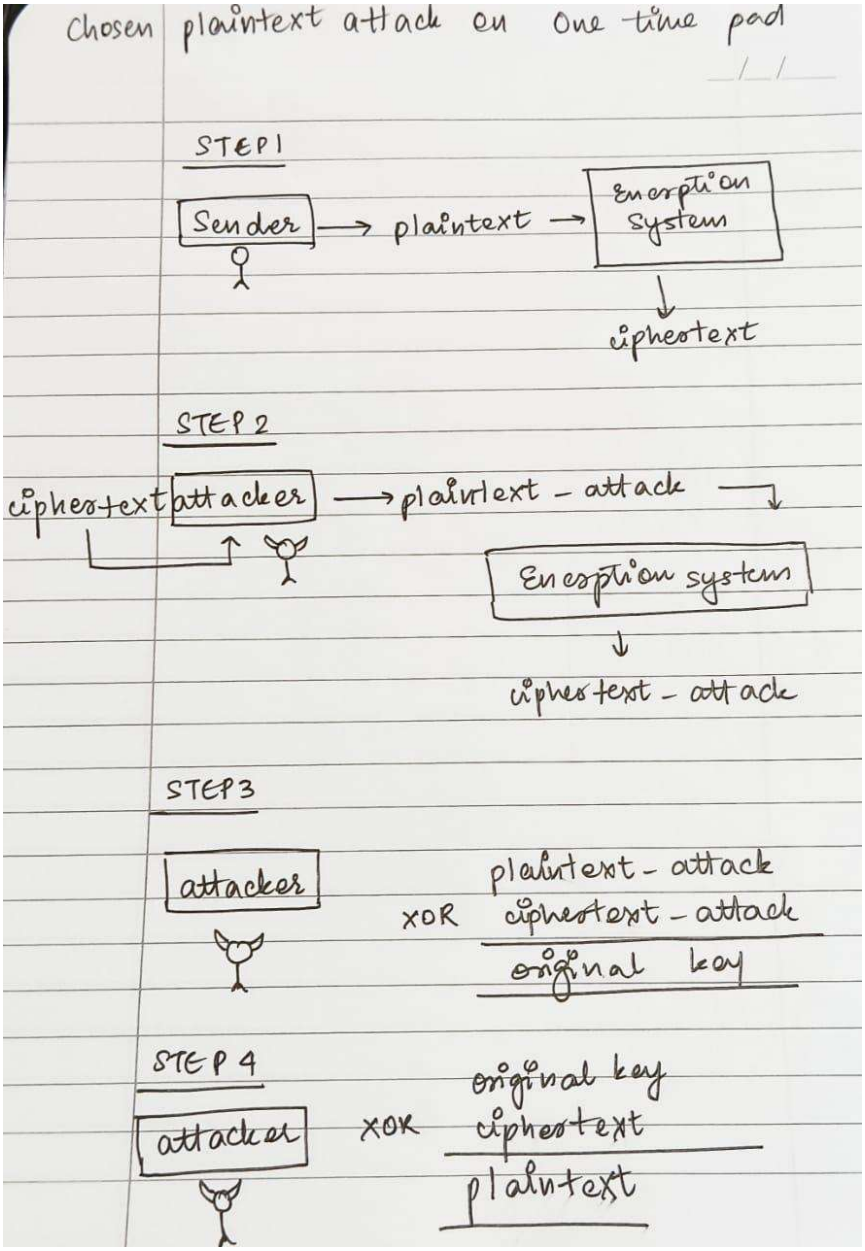
One time pad

- It is the most secure cipher and is an improvement on the Vernam Cipher.
- A single randomly generated key is used to encrypt and decrypt the message. After the use of the key it is destroyed.
- In the case of a one-time pad, the choice of l doesn't aect security (l = 10 is "just as secure" as l = 1000).
- However, the length of the keys and plaintexts must be compatible.
- Should be noted that One time pad cipher is secure only against Ciphertext only attack.
- In ciphertext only attack, the adversary only has knowledge on the ciphertext and nothing else.
- In the case of a chosen plaintext attack, the adversary also has access to some plaintext message and the corresponding ciphertext which was generated using some key. In this kind of attack, One time pad cipher is not secure.

1. Algorithm used to break the cipher under the aforementioned attack model,
2. Clear statement of the goal achieved by the attacker,
3. Code to implement the attack.

Algorithm used to break the One time pad cipher using the chosen plaintext attack model:

1. In the case of chosen plaintext attack model, the adversary has access to the ciphertext corresponding to original plaintext
2. The attacker sends their own plaintext (plaintext_attack) into the encryption system and obtains a new ciphertext name ciphertext_attack.
3. The attacker can now XOR plaintext_attack and ciphertext_attack to obtain the original key for the system.
4. After the key is obtained, the attacker uses the original ciphertext and XORs it with the key to obtain the original plaintext



Clear statement of the goal achieved by the attacker

As the attacker now has access to the original message sent by the sender, the security of the entire system is compromised. Attacker's goal to hack into the system and get access to confidential information is achieved. The main goal of adversaries and malicious insiders is to access high value devices, applications, and data. As this attack proves the system to be insecure, such malicious attacker with access to sensitive apps, data, and networks pose a significant risk to confidential systems.

This code is implemented for length = 4, however the code can easily be modified to take length of the strings as user input.

Code to implement the attack:

```
import java.io.*;
import java.util.*;
class OTP
{
   static String XOR(String a, String b)
      {
      String ans = "";
       for (int i = 0; i < 4; i++)
      {
         // If the Character matches
         if (a.charAt(i) == b.charAt(i))
            ans += "0";
         else
            ans += "1";
      }
      return ans;
      }

       //This is the encryption function, it takes plaintext as an input and generates corrosponding ciphertext
      static String enc(String pt)
      {
        String key= "1111"; //the system has a fixed key
        String ct = XOR(pt,key);
        return ct; //ciphertext is generated by XORing plaintext and key
      }

      static String attack (String ct)
      {
         //this is what the attacker does
           //attacker already has access to original ciphertext obtained from the original plaintext. This is because it is a chosen plaintext attack.
         //attacker has access to ct.

         String pt_attack = "0001";
         //the attacker sends their own plaintext into the enc function.
         String ct_attack = enc (pt_attack);
              //Now the attacker has the original ciphertext, their own plaintext and corresponding ciphertext.
         //attacker calls xor function on pt_attack and ct_attack to obtain original key
         String key = XOR(pt_attack,ct_attack);
```

```
        //attacker calls xor function on original ciphertext and key to obtain original plaintext
        String plaintext = XOR(ct,key);
        return plaintext;
    }

  public static void main(String[] args)
  {
    //String x=XOR("1010","1101");
    //System.out.println(x);
    Scanner sc= new Scanner(System.in);
    System.out.println("You are the sender please enter a four digit binary message");
    String msg= sc.nextLine();
    String y = enc(msg); //this is the original plaintext
    System.out.println("The ciphertext for your message is:" +y );

    String z = attack(y);
     System.out.println("The attacker has hacked your system and has access to your message
which is:"+z);


  }
}
```

```
java -cp /tmp/RNdkqAk5kt OTP
You are the sender please enter a four digit binary message
1010
The ciphertext for your message is:0101
The attacker has hacked your system and has access to your message which is
    :1010
```