Challenge 3

Rishi Vardhan Majji

August 2025

1 Description

True one-time pad encryption, as you have studied, is perfectly secure, that is, there is no way for any (potentially unbounded) adversary to distinguish between the encryption of string under one time pad and a string sampled uniformly at random (of the appropriate length) with a non-zero advantage.
Bob, however, felt that the byte 0x00 in the key could potentially lead to security issues because it does not change the message character at all during the encryption. So instead of sampling each key-byte from the interval [0x00, 0xff], he decided to instead sample the key-byte uniformly from the interval [0x01, 0xff]. Show that Bob's encryption scheme is no longer perfectly secure by distinguishing between the encryption of a string you provide and the encryption of a random message. (This is slightly different from IND-onetime security as defined in class, but it is also equivalent to perfect secrecy.)

2 Approach

As the byte 0x00 is removed from the interval from where the key takes its bytes from, The key can never have 0x00 byte in it, meaning that the ciphertext can never have any byte as the same as the message, because for that to happen, the key should be having the 0x00 byte at that location.
So we are going to give an input message such that we have a high chance of finding the same bit of our original message in the randomly generated message.
So lets just give an input of **'00'** (hex) , so if any of the two encryptions have

the byte '00' in them, then it should be the one made from a random message.
But we cannot be sure as just one bit is not enough as the probability of the bit

being '0x00' is 1/256 . So we are going to give a large input -¿ A huge string of '00' bits long enough to have a good probability of getting atleast one 0x00 bit in the randomly generated encrypted message and so that we can distinguish between them.

```
L=5000
```

```
payload = "00"*L
```

Making the input long enough for high chances of distinguishability.

```
def func(c):
        x=bytes.fromhex(c)
        return b'\x00' in x
```

Creating a function to check if the message has the byte '00' in it

```
if func(c1) and not func(c2) :guess=2
elif func(c2) and not func(c1): guess=1
else :guess=1
```

If c1 has it, then it must be the fake one, so c2 would be our encrypted msg and vice versa.
And if both of them dont have it(Rare),then we just guess it to 1. **(So we take a high value of 'L' to reduce the probability of this happening)**

And we end up geting the key cs409{y0u_h4d_fu11_4dv4nt4g3}