Challenge 4

Rishi Vardhan Majji

August 2025

1 Description

Suppose you are given a long key generated using the method from the previous challenge (uni formly sampling each key byte from [0x01, 0xff]). Could you still repurpose this key somehow to make a perfectly secure encryption scheme?
One way to do so is as follows: Say we denote the plaintext by m1,m2,...,mn where each mi is a single byte. Considering each byte as a number in base-256, we can think of the message as a number

$$m = m_1 * 256^{n-1} + m_2 * 256^{n-2} + \; + m_{n-1} * 256 + m_n$$

in decimal. Converting this number to base-255, say we get the base-255 representation as p1p2...pn', that is,

$$m = p_1 * 255^{n'-1} + p_2 * 255^{n'-2} + \; + p_{n'-1} * 255 + p_{n'}$$

, where pi ϵ [0,254] for each i ϵ [1,n'] (where n' is fixed ahead of time– what should it be?). Define the ciphertext as follows: say that for i ϵ [1,n'], we have ci = (pi + ki -1) (mod 255) where ki is the ith byte of the key interpreted as an integer in [1,255]. Interpret c = c1c2...cn' as a number in base-255 and convert it back to base-256 to get a byte sequence which forms the ciphertext. Can you argue that this method of encryption is perfectly secure?

2 Approach

We Just reverse the whole process

```
with open ("ciphertext.enc",'rb') as f:
    x=f.read()
with open ("keyfile",'rb') as f:
    k=f.read()
```

Read the key and cipherthext to x and f.

```
value=0
a=1
for i in range(len(x),0,-1):
    value+=x[i-1]*a
    a*=256
```

From the ciphertext, as we want to convert it to base 255 from 256, we get the value of it.
vspace1em

```
c=[]
for i in range(len(x)):
    c.append(value%255)
    value//=255

c[:]=c[::-1]
```

And then we get the base 255 coefficients from the value, put it in list c.

```
p=[]
for i in range(len(c)):
    p.append((c[i]-k[i]+1)%255)
```

And then, we create list p, and revert the process we used to generate cipher to get the coefficients related to our original message.

```
value=0
a=1
for i in range(len(c),0,-1):
    value+=p[i-1]*a
    a*=255
```

Now,as we want the coefficients of base 256, we convert the base 255 version to a value.

```
m=[]
for i in range(len(c)):
    m.append(value%256)
    value//=256

m[:]=m[::-1]
```

And then, from the value, we get the coefficients of base 256 for our original message.

```
byte_seq = bytes(m)
flag = byte_seq.decode('utf-8')
```

And now, we just decode the bytes to get out message.