Basic Challenge 2

S Chowdhury

Challenge Description

The flag has been encrypted by a 2-byte key (using xor)
After the flag has been encrypted, the encrypted hexadecimal string has been provided
encrypted hexademial string: 923995328f219126800a92399532ab259527800ac628

Solution

This time we have a 2-byte key. So, when doing the xor, we have to do it in a different way. Suppose we have ciphertext:
16 f2 34 b2
And a key:
10 50
In this scenario since we have a 2-byte xor key the *10* would xor with the *16*, the *50* would xor the *f2*, then the *10* would xor the *34* and the *50* would xor the *b2*. With this in mind, we can write the script.
We were given the hexadecimal string, so first we need to convert to bytes:

```
input_string = "923995328f219126800a92399532ab259527800ac628"
input_string_bytearray = bytes.fromhex(input_string)
```

Then, we have a nested for loop. Here, *ii* represents the first byte of the key and *jj* represents the second byte of the key.

```
for ii in range(256):
    for jj in range(256):
        key = [ii,jj]
        result = encrypt(input_string_bytearray,key)

        for kk in range(len(result)):
            result[kk] = chr(result[kk])
        new_string = "".join(result)
        if "flag" in new_string:
            print(f"key: {hex(ii)},{hex(jj)}  {new_string}")
```

We call the *encrypt()* function and store the result in the *result* variable. Then we have another for loop, where we convert the integers inside the *result* array into characters using the *chr()* function. After that, we use the *join()* method to join the characters in the array to form a string, and save the value in *new_string*. Then, we check if the string “flag” is present in *new_string*, and if it is, then we print out the key and the *new_string*, which contains the decrypted flag.
Here's the *encrypt()* function:

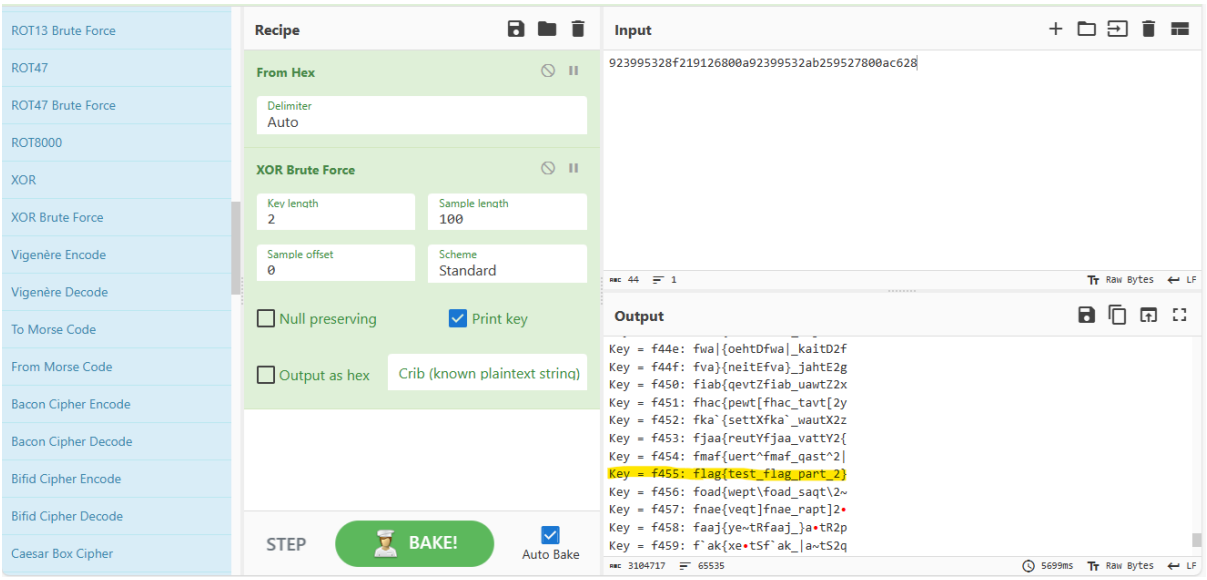
```python
def encrypt(data,key):
    final_encrypted = []
    encrypted_byte = 0
    for ii in range(len(data)):
        encrypted_byte = data[ii] ^ key[ii%len(key)]
        final_encrypted.append(encrypted_byte)

    return final_encrypted
```

So we encrypt each byte and then store it into the array. We pass in *data* as an array and we also pass in *key* as an array. When doing the xor, we have this:
data[ii] ^ key[ii%len(key)]
So its doing xor the same way as the example at the start.

Here's the output of the script:

```
PS C:¥Users¥sc¥Desktop¥challenges¥cryptography¥practice> python .¥xor_2_byte.py
key: 0xf4,0x55  flag{test_flag_part_2}
PS C:¥Users¥sc¥Desktop¥challenges¥cryptography¥practice>
```

We could have also used CyberChef to find the solution:



Python Program

```
def encrypt(data,key):
    final_encrypted = []
    encrypted_byte = 0
    for ii in range(len(data)):
        encrypted_byte = data[ii] ^ key[ii%len(key)]
        final_encrypted.append(encrypted_byte)

    return final_encrypted



input_string = "923995328f219126800a92399532ab259527800ac628"
input_string_bytearray = bytes.fromhex(input_string)




for ii in range(256):
    for jj in range(256):
        key = [ii,jj]
        result = encrypt(input_string_bytearray,key)

        for kk in range(len(result)):
            result[kk] = chr(result[kk])
        new_string = "".join(result)
        if "flag" in new_string:
            print(f"key: {hex(ii)},{hex(jj)}  {new_string}")
```