STUCK IN THE MIDDLE WITH YOU - CRYPTO 200

R-Boy is struggling with the encrypted messages. He's found a clue that leads him to a possible solution – though it looks too complex to solve. Even during his NASA training, he's never had to deal with something like this. Can you help him see if a flag is there?

The challenge provides us a python script with a simple encryption function and the encrypted flag. The script also print an hint for the solution:

StuckInTheMiddle:b3783a044b06141859d65a1e6fdbfc44

The first thing we did is to write a decryption function that given the original key and the ciphertext recover the original message, as follow:

```
def decr(key, ciphertext):
    key = bytearray(key)
    perm_plain = bytearray(BLK)
    plaintext = bytearray(ciphertext)

    for r in range(RND):
        for i in range(BLK):
            perm_plain[i] = plaintext[permutation.index(i)]
        plaintext = perm_plain
        for i in range(BLK):
            plaintext[i] = s.index(plaintext[i])
        plaintext = xor(plaintext, key)

    return hexlify(plaintext)
```

As the key is a global parameter to the script we fought that both the flag and the message "StuckInTheMiddle" were encrypted using the same key. According to the double_enc function the key is a string of length 8 characters, too long to a bruteforce attack.

The name "StuckInTheMiddle" remember us the meet-in-the-middle attack: we can halve the research space of 8 characters to 4 using this kind of attack.

As the double_enc function do a two step enc call we can try all the keys to encrypt the message, all the keys to decrypt the ciphertext and when a collision is found the key is the concatenation of the two keys used.

The code used to the recover the key is:

```
def generate_strings(length):
    for item in itertools.product(string.printable, repeat=length):
        yield "".join(item)

for key1 in generate_strings(4):
  e = enc(unhexlify(md5(key1).hexdigest()) , b"StuckInTheMiddle")
  mem[e] = key1

for key2 in generate_strings(4):
  e = decr(unhexlify(md5(key2).hexdigest()),
unhexlify(b"b3783a044b06141859d65a1e6fdbfc44"))
  if e in mem:
  print(mem[e] + key2)
```

The key used to encrypt the message is **replockx**

After found the key recover the original message for the flag was trivial:

```
plain_p1 = decr(unhexlify(md5("ockx").hexdigest()), unhexlify(final))
plain = decr(unhexlify(md5("repl").hexdigest()), unhexlify(plain_p1))
print(unhexlify(plain))
```

The plaintext is *B3Th3H0p3W3N33d!*

{FLG:B3Th3H0p3W3N33d!}