To understand the solution to this chall we need to understand how could we reverse the convolutions.

Examining the formula for the ith plain character reveals that, if we were to know all characters from 0 to i-1 we would be able to find the ith character, since it effectively is ((res[i]+k*256)-constant)/key[0], where the constant is a calculable value based on known values and k is an arbitrary value we will need to bruteforce based on a few conditions I will explain later.

Step 1: Length of plaintext and key:

```
len1 = len(res)-len(array2)
len2 = len(array2)
array1 = [0] * len1
```

I also created an array to use later.

Step 2: Calculate the constant (and the partial csum):

```
csum = res[i]
for j in range(max(0, i - len2 + 1), i):
      csum -= array1[j] * array2[i - j]
```

I effectively reversed the csum building process using the same code as in the source code. (no need for the min logic as len1 is 366 and len2 is 32)

Step 3: Calculate the initial csum and find the plaintext character:

```
while csum < 0:
            csum += 256
        while csum % array2[0] != 0:
            csum += 256
        array1[i] = csum // array2[0]
        while array1[i] < 32 or array1[i] > 126:
            csum += 256
            array1[i] = csum // array2[0]
```

Condition 1: csum must be positive.

Condition 2: the resulting plaintext character would have an integer value in ascii, so we check if the calculated value is indeed an integer

Condition 3: From the description we know that all plaintext character are printable, so we check for that as well.

Combining these steps results in a reverse of the convolution function. The end result that calculates the flag is:

```
def reverse_conv(res: bytes, array2: bytes) -> bytes:
    len1 = len(res)-len(array2)
    len2 = len(array2)
    array1 = [0] * len1
    for i in range(len1):
        csum = res[i]
```

```
        for j in range(max(0, i - len2 + 1), i):
            csum -= array1[j] * array2[i - j]
        while csum < 0:
            csum += 256
        while csum % array2[0] != 0:
            csum += 256
        array1[i] = csum // array2[0]
        while array1[i] < 32 or array1[i] > 126:
            csum += 256
            array1[i] = csum // array2[0]
    return bytes(array1)

key =
b'\xab\xec\xe9<\xaaC\x7fr\xeb\x8dgQ\xc0\x94\x01\x1d\xc03\x14\x97\xe2\x91\x97\xcf\
x8b\x13?\x1d24w|'
res_hex =
'17c080c00398a06e4661e403b2b571b578221bba83e235a0feece7213ad4d65c1d89c2a3afae5ef9
1bf7f2181f0c797505b7bd55c62d1edf2614b17f88f85eac674fbd6d7be4e2a617605c68e1baf8603
cb9b1d32b2bc1ab60d8c62b20be0bc0fb73a546b5641988a3bf8eeb778731e048970308d941a8bd5f
6cb56159069364c93b5429afdb85f9dfb5f5b0ca44d314af68bc9d56b39321fe5cc072c9508978693
ee60a9bffff5b52f6aa0ca37f9b421eb402a4886b742570926b7479d2b89528caceb7121a338c2331
64c33a120b9813bc56b855c914124ecb30df3d4a14c92788faa7c9e32b544e24d9d9fe2a5539a280c
28466dc6b276ba4b089fa26f8bace95f43f6c5d491e14e5fa09a853fff2dfd73a8cf8d7b54d3d8d69
3db7b182789f47e343e9cf56f8663e181a1e98276aface8b1052e3ee9c6630d69ad479bfe1106ec1a
b585a030ca130a6d849f9c4bed9d0b16f46890f1efa66c8f21f078088f426ef0e1f9af315ae3b2356
123df174bb4095ad2361237bedc3e62c294f8ccc135f9766f0ec2a462087cd2648'
res = bytes.fromhex(res_hex)
plain1 = reverse_conv(res, key)
print(plain1)
```

Made with love by: AndreiCat