Ciphertext CTF 2020

Cryptography & Steganography

HashString5

Description:
irreversibility of hashes implies security. is it that simple? nope :)

Files:

Code	Size: 0.54 KB	MD5: fee30f723c25fa24898b82c61bec10e6
hashed_flag	Size: 2.53 KB	MD5: 9772ebf36db392182608bfde19ee7b98

Solution:
We have a ciphertext and a code, so first step is to understand the code:

```
import hashlib
from random import randint
from string import ascii_letters

flag=open('flag.txt').read().strip()
permuted_flag=''
for i,j in enumerate(flag):
    if i%2:
        permuted_flag=f'{permuted_flag}{j}{ascii_letters[randint(1,len(ascii_letters)-1)]}'
    else:
        permuted_flag = f'{j}{permuted_flag}{ascii_letters[randint(1,len(ascii_letters)-1)]}'

permuted_flag=permuted_flag.strip()
open('permuted_flag','w').write(permuted_flag)
hashed_flag= ''.join([hashlib.md5(i.encode()).hexdigest() for i in permuted_flag])
open('hashed_flag','w').write(hashed_flag)
```

so, first it generates a permuted flag, then hashes each character.
Let's start reversing the process, we will read the hashed flag and substitute hashes with corresponding characters by creating a dictionary that maps characters to hashes:

```
import hashlib
from string import printable


d = {}
for i in printable:
    d[hashlib.md5(i.encode()).hexdigest()] = i
hashed_flag = open('C:/Users/venom/Desktop/CTCTF_writeups/CTCTF/crypto/hashstring2/hashed_fla
g').read()
permuted_flag = hashed_flag[:]
for i in d.keys():
    permuted_flag=permuted_flag.replace(i, d[i])
print(permuted_flag)
```

output:

```
nu_nha_oewe_oko_wFCCGTkkTIG{tneUpdWI_DgnsDwVQdlH_tcaZs_uwfFghMwsmNiBugSTqBkilJ}eW
```

we know that there are many characters that are irrelevant, so we should get rid of them and reorder the plain text to get the original plain text which is the flag.
Let's understand how permutation is done, there is a loop with 2 conditions which specify the way of permutation at each added character:

1. If the index is odd, then $PF = PF + char + ?$
2. If the index is even, then $PF = char + PF + ?$

To understand it better, lets write a function that encrypts a bunch of A's but instead of a random character, it adds a question mark '?':

```
flag='A'*40
permuted_flag=''
for i,j in enumerate(flag):
    if i%2:
        permuted_flag=f'{permuted_flag}{j}?'
    else:
        permuted_flag = f'{j}{permuted_flag}?'

print(permuted_flag)
```

output:

```
AAAAAAAAAAAAAAAAAAAA?A??A??A??A??A??A??A??A??A??A??A??A??A??A??A??A??A??A??A?
```

Interesting, lets put numbers instead of A's to see how they are mapped:
28 26 24 22 20 18 16 14 12 10 8 6 4 2 0 ? 1?? 3?? 5?? 7?? 9?? 11?? 13?? 15?? 17?? 19?? 21?? 23?? 25?? 27??

Looks like it starts from a specific index, let's call it mid, and a question mark next to it, then it puts all even-indexed characters on lower indexes with a step of 1, and all odd-indexed characters on higher indexes with a step of 3.
Ok, so now we understand it, and we can write a script that puts everything on the right positions:

```
mid=19
i=mid
j=mid+2

while i>=0 and j < len(permuted_flag):
    flag += permuted_flag[i]
    flag += permuted_flag[j]
    i-=1
    j+=3
print(flag)
```

output:

```
CTCTF{we_do_know_de_wae_of_hashing_quin}
```

aaaaand we got the flag!
CTCTF{we_do_know_de_wae_of_hashing_quin}.