[crypto] Fries

Points: 241

Challenge description:

> Somtimes we call it chips too!

Files: `encrypted_flag`, `fries.txt`, `fries.py`

Inspecting `fries.py`, we see the following code:

```
import string
import random
import hashlib

FLAG = b'???'

# OTP xor symmetric encryption
# decrypt function same as encrypt function
def encryptFlag(msg, shared_secret):
    sha512 = hashlib.sha512()
    sha512.update(str(shared_secret).encode('ascii'))
    key = sha512.digest()[:len(msg)]
    return bytes([i[0] ^ i[1] for i in zip(key, msg)])

def encrypt(word):
    word = list(word)
    for i in range(len(word)):
        word[i] = alpha[ord(word[i]) - ord('a')]
    return "".join(word)

alpha = list(string.ascii_lowercase)
random.shuffle(alpha)

# English word lists
words = open('words.txt', 'r').read().split('\n')

random.shuffle(words)

words = words[:10000]
key = " ".join(words[-5:])

for i in range(len(words)):
    words[i] = encrypt(words[i])

output = open('fries.txt', 'w')
output.write("\n".join(words))

enc = encryptFlag(FLAG, key)
open('encrypted_flag', 'wb').write(enc)
```

We see that there are two encryption functions: `encryptFlag` and `encrypt`.

The `encryptFlag` Function

```python
# OTP xor symmetric encryption
# decrypt function same as encrypt function
def encryptFlag(msg, shared_secret):
    sha512 = hashlib.sha512()
    sha512.update(str(shared_secret).encode('ascii'))
    key = sha512.digest()[:len(msg)]
    return bytes([i[0] ^ i[1] for i in zip(key, msg)])
```

Taking a closer look at `encryptFlag`, we see that it takes two parameters, `msg` and `shared_secret`. The `shared_secret` is first hashed using the sha512 hash algorithm, before being shortened to be the same length as `msg`, to be used as the `key`. Subsequently, the function returns the value of `msg` ⊕ `key`, producing an encrypted output. (⊕ refers to the xor operation).

The above description of the function probably isn't necessary, since the comment above it helpfully informs us that the function can be used to decrypt the encrypted version of the message (due to the property of xor where `a ^ b = c`, `c ^ b = a`, where `a` is the recovered message, and `^` is the xor operation). Nevertheless, here it is, in this write-up.

From this, we can also deduce that the key to this challenge is to find the value of `shared_secret`.

The `encrypt` Function

```python
def encrypt(word):
    word = list(word)
    for i in range(len(word)):
        word[i] = alpha[ord(word[i]) - ord('a')]
    return "".join(word)

alpha = list(string.ascii_lowercase)
random.shuffle(alpha)
```

Moving on to look at `encrypt`: we see that `encrypt` only takes one parameter, `word`. It first converts `word`, presumably with type string, to a list. For each item in the list (or letter in `word`), it replaces the letter a letter from `alpha = list(string.ascii_lowercase)`, which is all the letters of the alphabet in lowercase. We see also that `alpha` is then randomly shuffled, which mean that we don't know the order that the letters are in.

`ord(word[i]) - ord('a')` produces a number that is within the range of 0 to 25 inclusive. (This is with the assumption that `word[i]` is also a lowercase letter of the alphabet).

To test this, assuming `word[i]` is a maximum of `z` (`ord('z') = 122`), `ord('z') - ord('a')`, where`ord('a') = 97` , is equal to 25. `ord('a') - ord('a') = 0`.

So what this function does is it replaces each letter of `word` with a random letter of the alphabet, without any overlaps in the mappings of letter-to-letter. (It is one-to-one).

This is important, because this means that we can apply frequency analysis to deduce the letter mappings and find both `alpha` and the original `word`. (I used this link). (More elaboration later).

The Main Part

Finally, looking at how the functions are called:

```
# English word lists
words = open('words.txt', 'r').read().split('\n')

random.shuffle(words)

words = words[:10000]
key = " ".join(words[-5:])

for i in range(len(words)):
    words[i] = encrypt(words[i])

output = open('fries.txt', 'w')
output.write("\n".join(words))

enc = encryptFlag(FLAG, key)
open('encrypted_flag', 'wb').write(enc)
```

1. `words.txt`, not provided, is opened, converted to a list, randomly shuffled and saved into `words`.
2. `key` is the last 5 of the shuffled words converted back into a string
3. For each word in `words`, `encrypt(word)` is called.
4. These encrypted words are then saved into `fries.txt`.
5. `FLAG`, the thing that we want to find, is then encrypted with `encryptFlag`, where `FLAG` is the previously discussed `msg`.

The goal, now that we understand what's going on, is to try to find the value of key, given by the last 5 words in the scrambled word list. These last 5 words correspond to the last 5 scrambled words in the `fries.txt` file, which are:

```
qbgbsj
hsgnwlmcmhap
ikibanimqsmit
ahskvg
lghsmnr
```

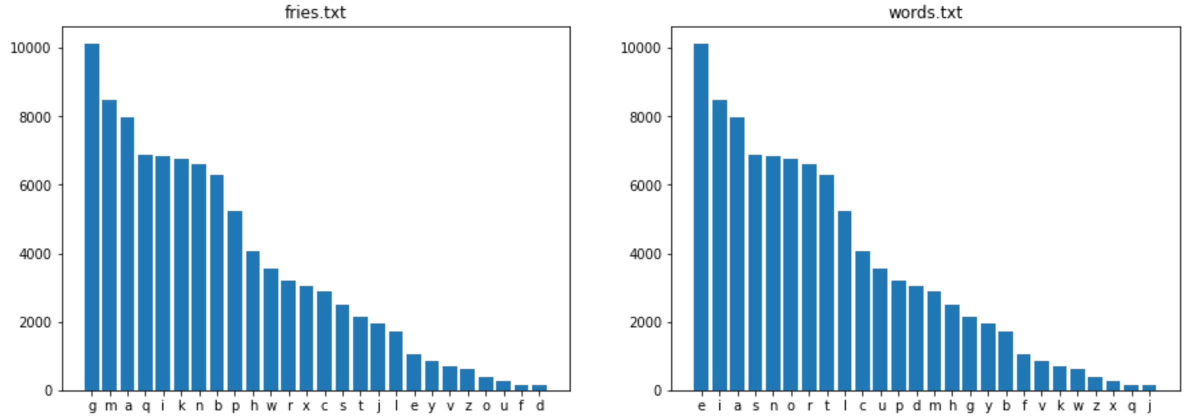
So we will need to decrypt them.

Using the previously mentioned frequency analysis (and this helpful tool to crack the substitution cipher and figure out the key), we get the following suggested key, `alhxgetsmdvpcikrfnqbwyzujo`, as well as these 5 words:

```
stethy
cherubimical
nontarnishing
achoke
bechirp
```

Don't know what their definitions are, but they look like English.

Also, for an illustration of how letter frequency is retained:



```
alhxgetsmdvpcikrfnqbwyzujo (key)
abcdefghijklmnopqrstuvwxyz
```

The solve:

```python
import hashlib

f = open('encrypted_flag', 'rb')
FLAG = f.read()
f.close()

# OTP xor symmetric encryption
# decrypt function same as encrypt function
def encryptFlag(msg, shared_secret):
    sha512 = hashlib.sha512()
    sha512.update(str(shared_secret).encode('ascii'))
    key = sha512.digest()[:len(msg)]
    return bytes([i[0] ^ i[1] for i in zip(key, msg)])

words = ["stethy",
        "cherubimical",
        "nontarnishing",
        "achoke",
        "bechirp"]
key = " ".join(words)

enc = encryptFlag(FLAG, key)
open('decrypted_flag', 'wb').write(enc)
```

And yay, in the `decrypted_flag` file, we see the flag.

Flag: `greyhats{M@yb3_y0u_c@n_7rY_5paN15h}`

This rather long-winded write-up was written by `lct`, from team `WXYZ`.