Round-round

chall.py

```
from os import urandom

from hashlib import sha512

from Crypto.Util.strxor import strxor



from aes import *



FLAG = open('flag.txt', 'rb').read()

master_key = urandom(16)

flag_key = sha512(master_key).digest()

enc_flag = strxor(FLAG, flag_key[:len(FLAG)])



plaintext = urandom(16)

plain_state = bytes2matrix(plaintext)

key_matrices = expand_key(master_key, 2)



add_round_key(plain_state, key_matrices[0])

sub_bytes(plain_state)

shift_rows(plain_state)

mix_columns(plain_state)

add_round_key(plain_state, key_matrices[1])
```

```python
sub_bytes(plain_state)

shift_rows(plain_state)

mix_columns(plain_state)

add_round_key(plain_state, key_matrices[2])



ciphertext = matrix2bytes(plain_state)



print(f"Protected flag: {enc_flag.hex()}")

print(f"Plaintext: {plaintext.hex()}")

print(f"Ciphertext: {ciphertext.hex()}")



master_key = [

    0, master_key[1], 0, 0,

    master_key[4], 0, master_key[6], 0,

    0, 0, 0, master_key[11],

    master_key[12], 0, master_key[14], 0,

]



print(f"Hint: {master_key}")

print(f"Hint: {key_matrices[1][2][0]}")

print(f"Hint: {key_matrices[1][3][1]}")
```

output.txt

```
Protected flag:
9dace36c2d7ea765b82c3f7e330eb9ce32dbf575c33d29e12a235e7cc63dfb852c3789f3ef1da1
d866d7c8aabdb5ed0fd4fa6c18
Plaintext: e50b0834f575d775e98b5285ebb08b94
Ciphertext: 001cda9ccdc6bb14e43fb5351d8e52fa
Hint: [0, 131, 0, 0, 93, 0, 142, 0, 0, 0, 0, 187, 129, 0, 42, 0]
Hint: 23
Hint: 16
```

Main problem:
The aes implementation only uses 2 layer of encryption enabling a low data complexity attack.

anddd the rest is just implementation :)