Challenge

> This time Fady learned from his old mistake and decided to use onetime pad as his encryption technique, but he never knew why people call it one time pad!

> msg

Solution

The challenge description states that Fady is using OTP encryption. As the name implies, the secret key in OTP encryption can only be used once, otherwise the complete key and plaintext can be recovered using a many time pad attack (aka crib drag). Assuming that each line in the `msg` file is a line of text encrypted with the same OTP key, we can begin the crib drag by xoring 2 lines together and guessing possible letters.

I used the cribdrag tool from SpiderLabs to recover the plaintext.

```
$ python xorstrings.py 0529242a631234122d2b36697f13272c207f2021283a6b0c7908 2f28202a302029142c653f3c7f2a2636273e3f2d6
2a01040053321d06014e09550039011a07411f0c4d044e2d0000

$ python cribdrag.py 2a01040053321d06014e09550039011a07411f0c4d044e2d0000
Your message is currently:
0       ____________________________
Your key is currently:
0       ____________________________
Please enter your crib:
```

After a couple dozen guesses, I was able to decrypt the message as such.

```
Your message is currently:
0   Dear Friend, This time I u
Your key is currently:
0   nderstood my mistake and u
Please enter your crib:
```

Now that we have some plaintext, we can recover the key by simply xoring a plaintext string together with it's corresponding ciphertext. This can be done by modifying `xorstrings.py` slightly.

```
s1 = "0529242a631234122d2b36697f13272c207f2021283a6b0c7908".decode('hex')
s2 = "Dear Friend, This time I u"

s3 = sxor(s1, s2)

print s3
```

Running the modified `xorstring.py` script produces the flag: **ALEXCTF{HERE_GOES_THE_KEY}**.