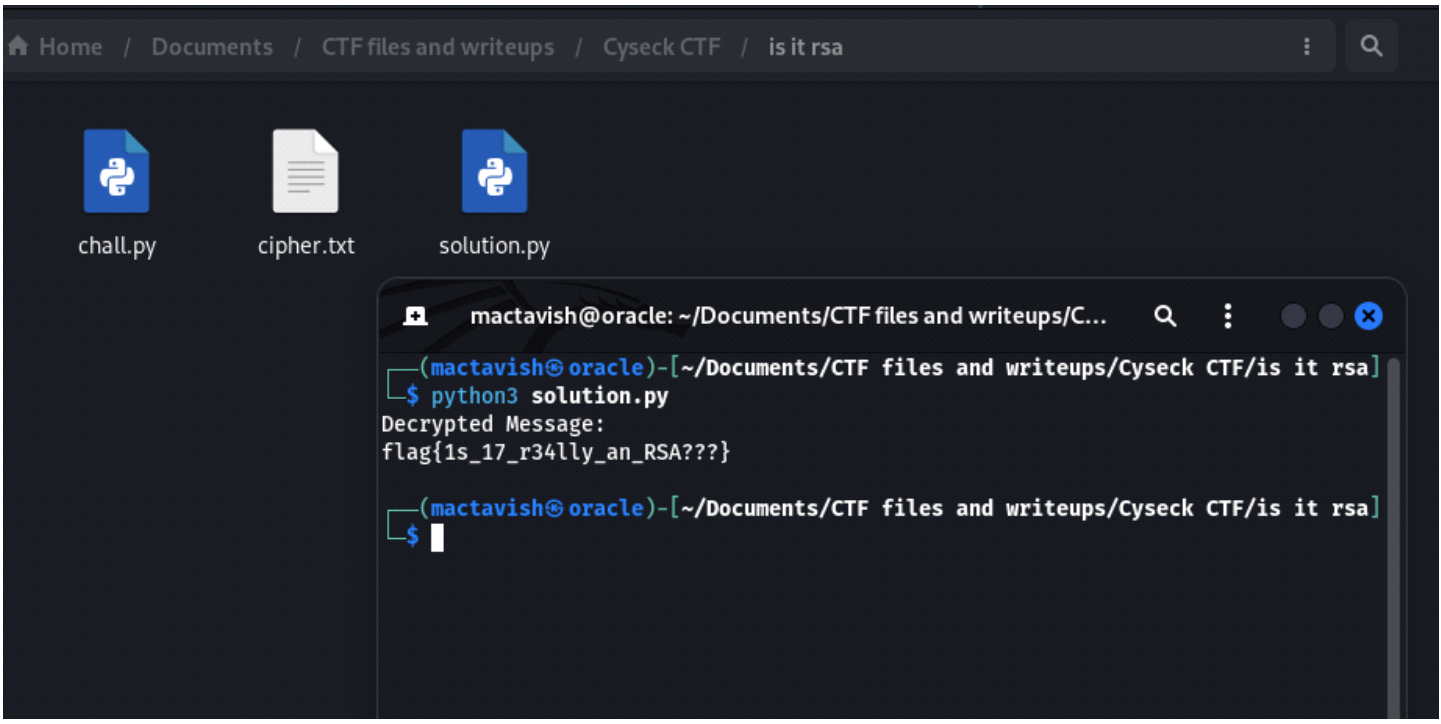
IS IT AN RSA???

flag{1s_17_r34lly_an_RSA???}

08 July 2023 12:33

```
from Crypto.Util.number import inverse

# Given values
C =
300765675631036702440523739351154897383328322310909427639150819891961020790689307369308441255403094732423274431969874729576538743620119420735510045406001994753894520763272198722820558323777418995706994461414667818537719413094635232811184202577172511003889024565202258024532325841442524255115720343283468219385208658431547944302686466963643630617734585172811247868088027325004294939532871542424327865040876468978611871974999372691163464294756117812578408540739824709677959432355997425868877804980417752642107595057064248851537732058086278189683557116314198234770298192400203155340277871276718988519131085372705342068348850422546040347545294934795160228629715928205702963059053926678039307251970985067503945484142259384931005122474633734569350468028096545310923922736277126500494960558901118391826970750599017736592975695396953268648137731306315298363276952213929906449212672277952655169301409708326657804508222562820866018885085111020645767177241778007313870356313737766211807671684466236765782369978681732347669091529835873174707088943878081394207743832495916408484710683636315373611964455948879147334744259927601718737189225141948942454372275019801444709829300827100970958900715300313378107547518939050666002906010324615404445192022240280315907630458853581194336155758396621241419237953860938727690588816954526361793957917189783860306310822734347040467227916427155488984237263951334309867259192368643799454225026080500451877047304229273388998226482371462690188525788808790286231420310667434427953649009749851835345643678202026126838640308587587491539165176588320928304857832498694961323661593475151903835366820810227102061312419087294682432081917413904639688299669590743900799344326135311955543328076735324152278029826551922472334486502065676931796120276566197522523087325802268044548699688695109433084891340455120679761061711197620052128458728329181321749503305859230167473247286327332162638070885298897162507611749649393208158266253582200354380719427286181196920627292677984198089607982851981186674993709576820550096457875619124681598635713218705170000750192136641170764054753456395228025468183484945388029949429676193689771416723243948217122509236756623
E = 65537
N =
209129100507960318308096739776744558573933200961134710001084765982396718232904545390995050063411819278033962300023387354544016464064759126882189231043601014934205874640702733235275672661130856206297771822302726490160790340358261936069417174099111272930339375111272326220253832677480187749714150712321126825180854652536777860758823887512529037588858223352010487830336070591251141297219622370318481602578400427496047016613511046321972681013383565851529932004998472241484282762852809477130258415944729244593308144757965196914965989763662246812271830579440783679886466699838875113867123007330071802887011943514747094514613
P =
124716128750274416177198385152601667810715797395833853677249811082727897461676570581863936449483189681198092903020904507322672470247046160304218067660413360746027028163176276250302488523199660253851031967063877633434238865709326658853456209535821692944228463233716423928577199378853127205934867872502637393289
Q =
167684085934635111138105189988179080219157811889011166123666499425768450437469943386621848048488343683888778929689860701092367929913492900054874870222429427988420144341251557765223160805932679793036379398755587186757197200753836874710801303907347858239825996765304285670427266939478508125758879660649717702749

# Calculate private exponent (d)
phi = (P - 1) * (Q - 1)
d = inverse(E, phi)

# Decrypt the ciphertext
m = pow(C, d, N)

# Convert the decrypted message to bytes and decode as string
decrypted_message = m.to_bytes((m.bit_length() + 7) // 8, 'big').decode()

# Print the decrypted message
print("Decrypted Message:")
print(decrypted_message)
```



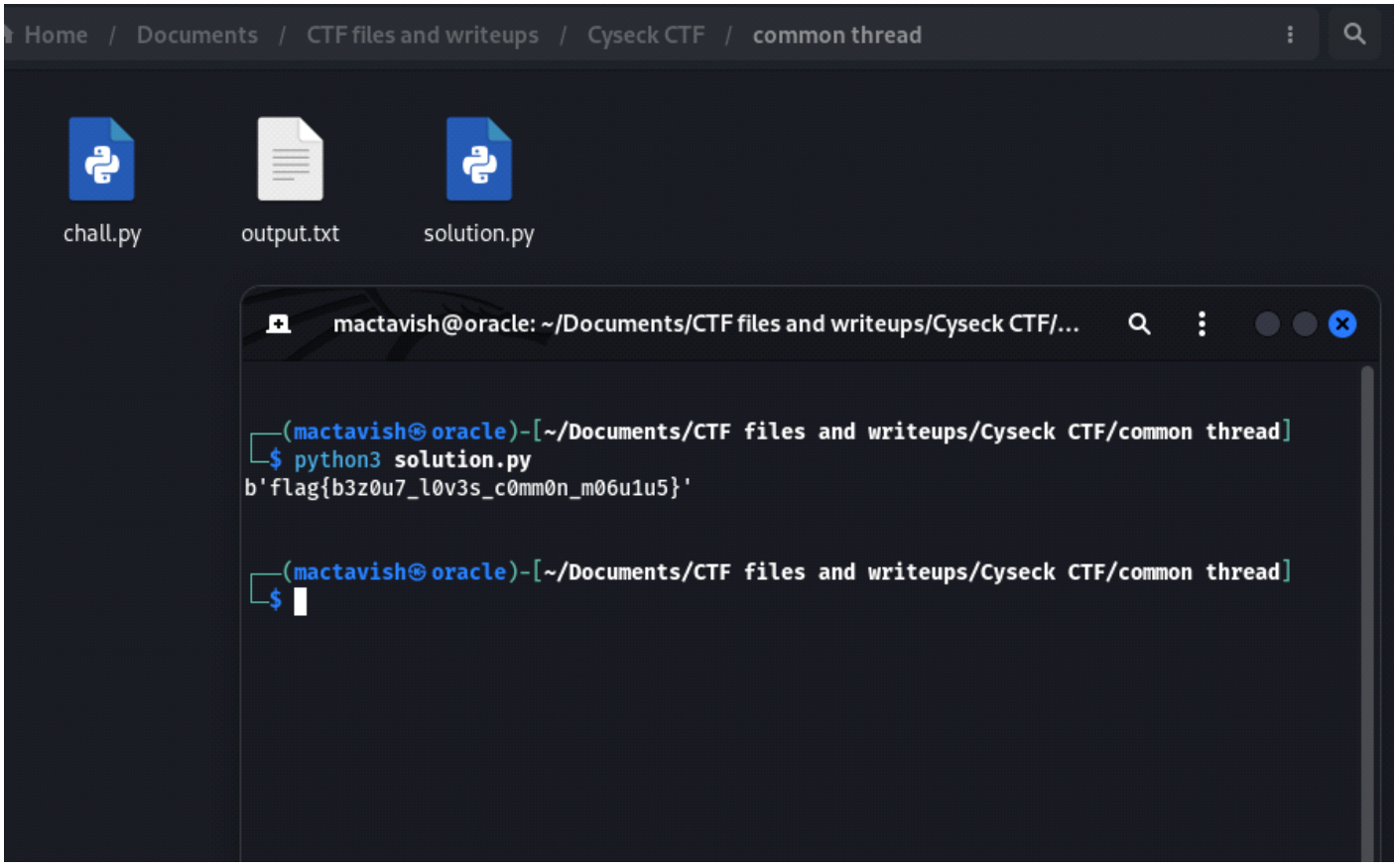
NB: Code collected from ChatGPT/StackOverflow using prompt manipulation

Common Thread

flag{b3z0u7_l0v3s_c0mm0n_m06u1u5}

08 July 2023 16:30

```
from libnum import xgcd, invmod, n2s

def common_modulus(e1, e2, c1, c2, N):
    # Extended Euclidean algorithm
    a, b, d = xgcd(e1, e2)

    # Invert negative factor
    if b < 0:
        c2 = invmod(c2, N)
        b = -b
    if a < 0:
        c1 = invmod(c1, N)
        a = -a

    # Get the message (c1^a * c2^b) % N
    m = (pow(c1, a, N) * pow(c2, b, N)) % N
    return m

n =
825290038541074496558828287793066802613225142915781769148553356601963161360208913302829778623927819555760039082769183362324964283949027017128650508461779194321500242213332488691828191607240040276876153166045249564777268533414717294905797812269047502032677748152668149930077007999094403272001577430016362089 79
c1 =
166814478617099665759599925878885485905004693877676031302405103926304712377128833289451399315285139351023889805260771073135932355790495777710693458441294423702598858530323684819801180130578409276313454673613313827449751162782615878162134 9
c2 =
746491129170129963893896885845390470388141172427878904224127324449405271144654443673807370274298111757927759420295145019842157007516460855485630430074789596679545726468711974790857457507525657804674266342421976330177596275050251638990527690398734992259759565471976031013399465157056975326531965518913803836 92

# Perform common modulus attack
m = common_modulus(3, 65537, c1, c2, n)

# Convert the recovered message to plaintext
flag = n2s(m)
print(flag)
```



NB: Code collected from ChatGPT/StackOverflow using prompt manipulation

Too close for comfort

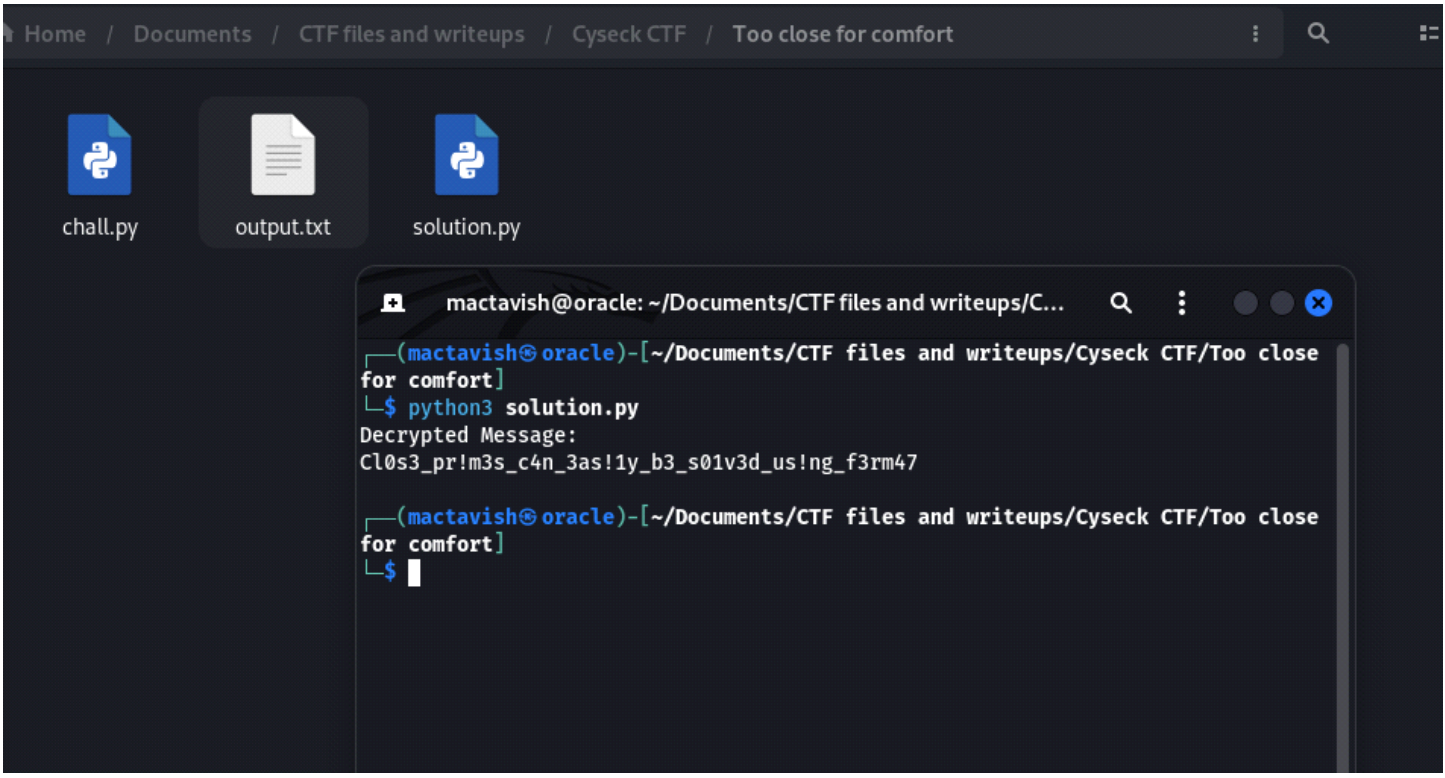
flag{Cl0s3_pr!m3s_c4n_3as!1y_b3_s01v3d_us!ng_f3rm47}

08 July 2023 16:30

Take the sqrt of n. find the nearest prime number around sqrt(n). you will get p and q. along with n decrypt the rsa

Reference link:
https://www.wolframalpha.com/input?i=nearest+prime+to+100

```
from Crypto.Util.number import inverse

# Given values
C =
36352708125237430764760060909529418780804291559038503125980629255501905387145728
37736003534130808633019514020067577852128083412516317887219432130679055627366383
86972350943764801179861752923887050054038705237602609256272241103832307947358569
03192310702009799631105903896037080675097289706993777714380538272695
E = 65537
N =
51262621421762918526093632383370534180768834026547970314343452353598602088230820
80603374000732879555457064231825278966676331525350986648997230463127405088009635
82614001960837739617688711828681829802559542306463651633477383409231648397029599
48086049748616395632937734313167798374046524562505972965628250476311
P =
71597919957051069715186926000974049109390503354788204853419394499252058806471343
25454070146449825578592400158599249475957026040350418341470093545761250817
Q =
71597919957051069715186926000974049109390503354788204853419394499252058806471343
25454070146449825578592400158599249475957026040350418341470093545761250583

# Here we calculate private exponent (d)
phi = (P - 1) * (Q - 1)
d = inverse(E, phi)

# Here we decrypt the ciphertext
m = pow(C, d, N)

# Here we convert the decrypted message to bytes and decode as string
decrypted_message = m.to_bytes((m.bit_length() + 7) // 8, 'big').decode()

# Finally print the decrypted message
print("Decrypted Message:")
print(decrypted_message)
```



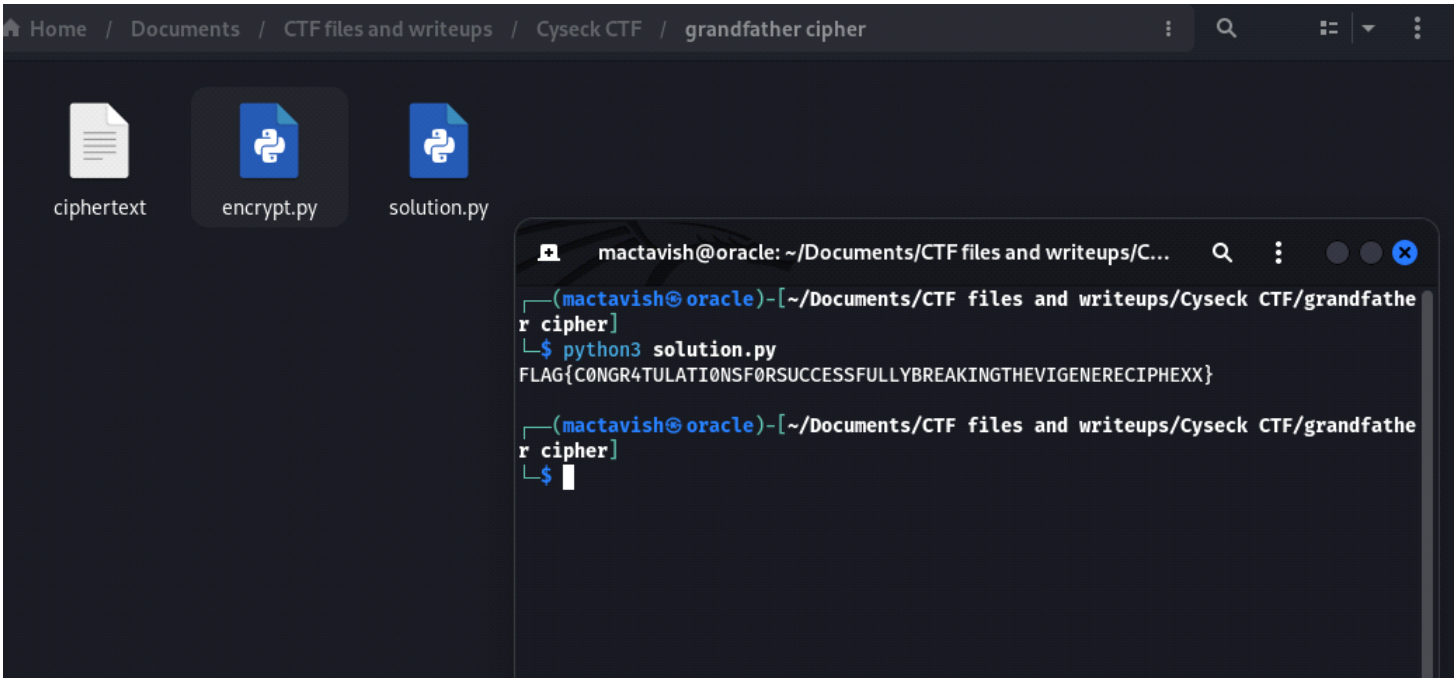
NB: Code collected from ChatGPT/StackOverflow using prompt manipulation

Grandfather Cipher

FLAG{C0NGR4TULATI0NSF0RSUCCESSFULLYBREAKINGTHEVIGENERECIPHEXX}

08 July 2023 16:30

```
import itertools

letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789{}'
encrypted_flag =
"O8Q2HZE9PCID38QDRL3COL7C3ZS01DVEU8CX01Q6R{WDQ1}4P13001S4Y4UH6W"

def decrypt(ciphertext):
    ciphertext = ciphertext.upper()
    char_to_val = {char: val for val, char in enumerate(letters)}
    key_length = 1
    while True:
        for key in itertools.product(letters, repeat=key_length):
            plaintext = ""
            for i, char in enumerate(ciphertext):
                ciphertext_val = char_to_val[char]
                key_val = char_to_val[key[i % len(key)]]
                plaintext_val = (ciphertext_val - key_val) % len(letters)
                plaintext_char = letters[plaintext_val]
                plaintext += plaintext_char

            if plaintext.startswith("FLAG{") and plaintext.endswith("}"):
                return plaintext

        key_length += 1

decrypted_flag = decrypt(encrypted_flag)
print(decrypted_flag)
```



NB: Code collected from ChatGPT/StackOverflow using prompt manipulation

Wojtek's Enigma

08 July 2023 16:30

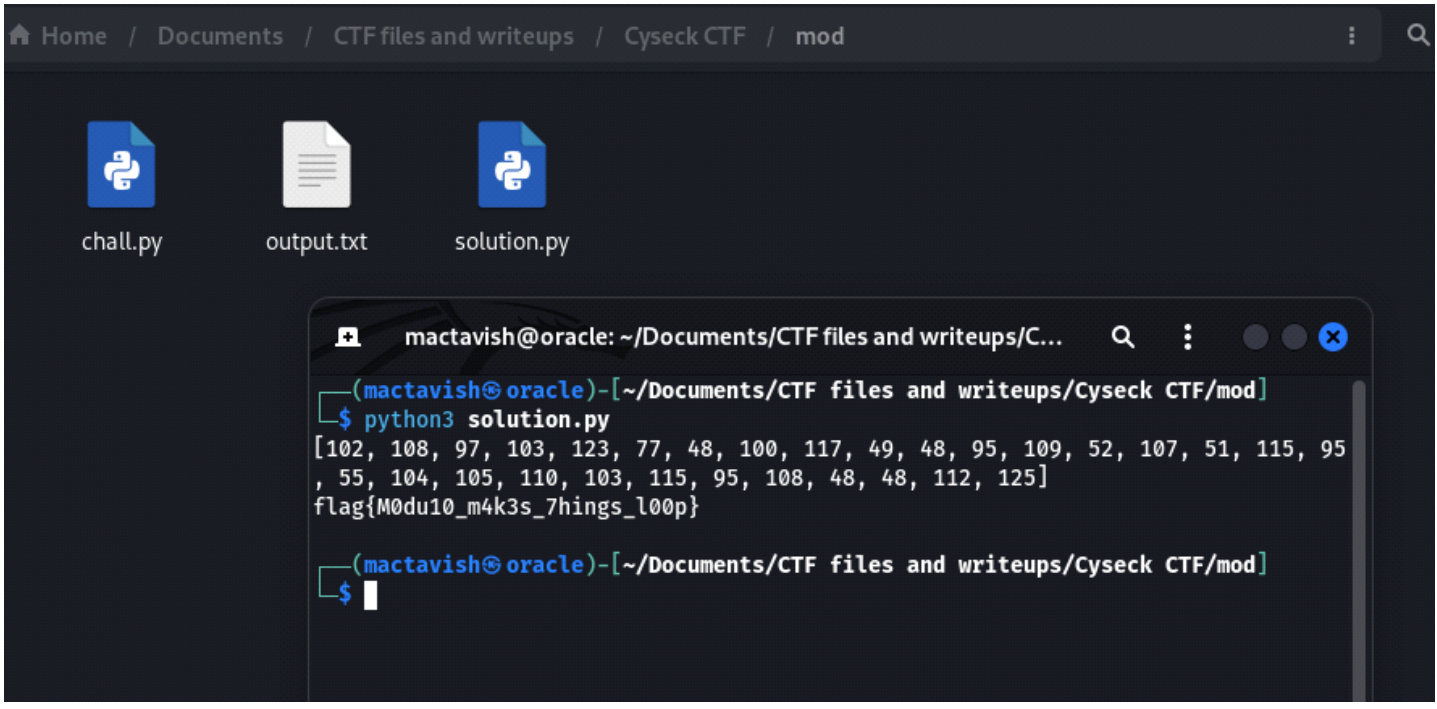
https://cryptii.com/pipes/enigma-machine

MOD

flag{M0du10_m4k3s_7hings_l00p}

08 July 2023 16:30

```
f = [5, 11, 0, 6, 26, 77, 48, 3, 20, 49, 48, 95, 12, 52, 10, 51, 18, 95, 55, 7, 8, 13, 6, 18, 95, 11, 48, 48, 15,
28]

for i in range(len(f)):
    if f[i] >= 0 and f[i] <= 28:
        f[i] += 97

print(f)

n = ""
for i in f:
    n += chr(i)

print(n)
```



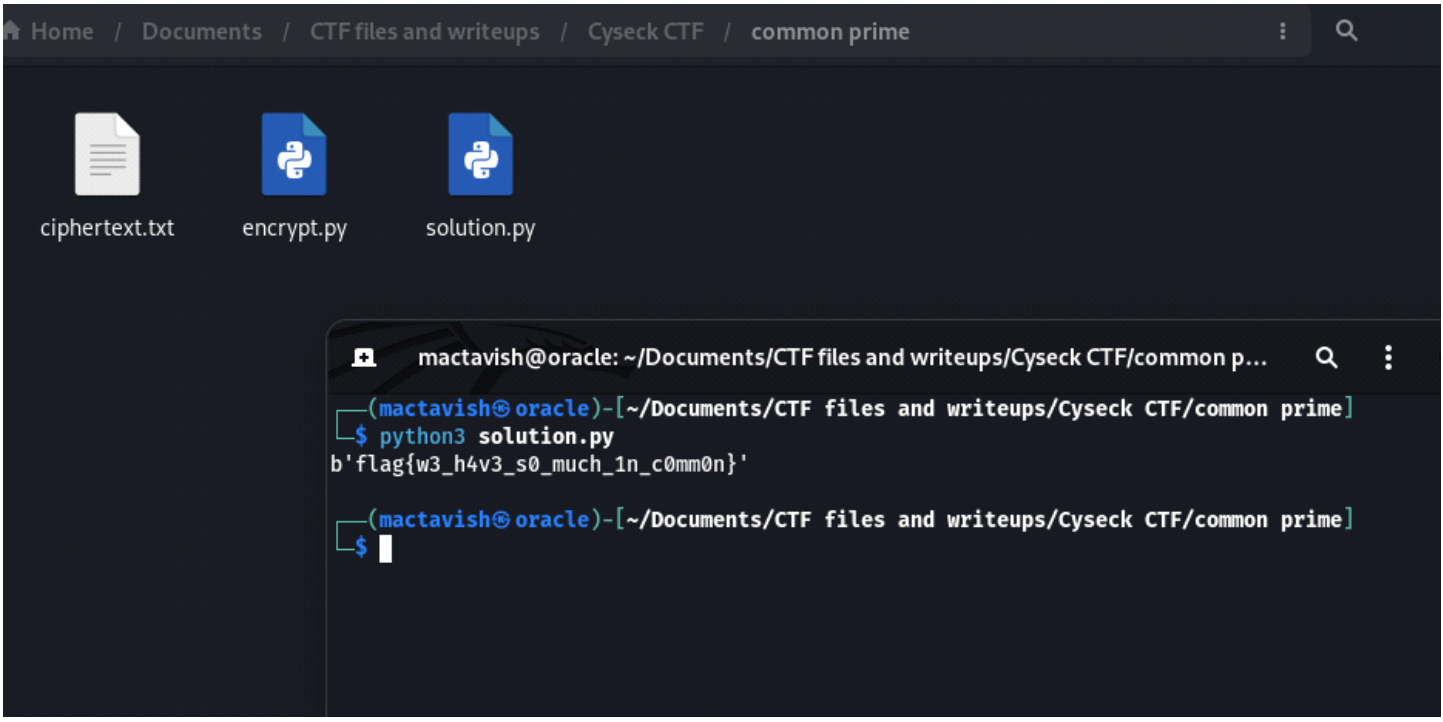
NB: Code collected from ChatGPT/StackOverflow using prompt manipulation

Common Primes

flag{w3_h4v3_s0_much_1n_c0mm0n}

08 July 2023 16:31

```
from Crypto.Util.number import *
import math

with open("ciphertext.txt") as f:
	ciphertext = eval(f.read())

modulus_list =
[92917019109246520946111919259944727745544542783982355430716972006653222462839194
37979135813881115890243658015083727250381321335556764390935740279912077953261513
30956912141124302793001576397891096804762781431160079584195468504514899040994003
77262985350595742313358525673308487492326205442219054566734280849463,
13056821224028637749214461399459669546042471470448591536225585572125538259951396
52513012475947742508361234723959074694095896896883257743292956617984421861873729
59761946857650743912997464565794093976101055331289489952844031378077429339913607
08370638941041810094887987379388337181934968707872549690967818969428 3,
85811324434028990250875422438123814864093618278891295811031083960296969537454874
94314197029750765707557866932278876331393511235604833531687398418400961383719017
30995170587497793034943090260262363210899681452544031292643096108523459554535061
83006563596075448244968379981815939543003661119788848975062475811483,
11506407970434477652231071415436507790883816180775441473022089472995171283449620
01700640371057186810388838883120947541044962169262389774764834810796289654042181
12773573213774902089911524937553684498516876841154214074392188918966248250666407
59954952307285100035743456804872654549984277545068389956580102978031 7]

e = 65537
for m, n in enumerate(modulus_list):
	for i in modulus_list[m + 1:]:
		p = math.gcd(n, i)
		if p != 1:
			q = n // p
			break
	else:
		continue
	break
phi = n - p - q + 1
d = pow(e, -1, phi)
msg = pow(ciphertext[m], d, modulus_list[m])
print(long_to_bytes(msg))
```



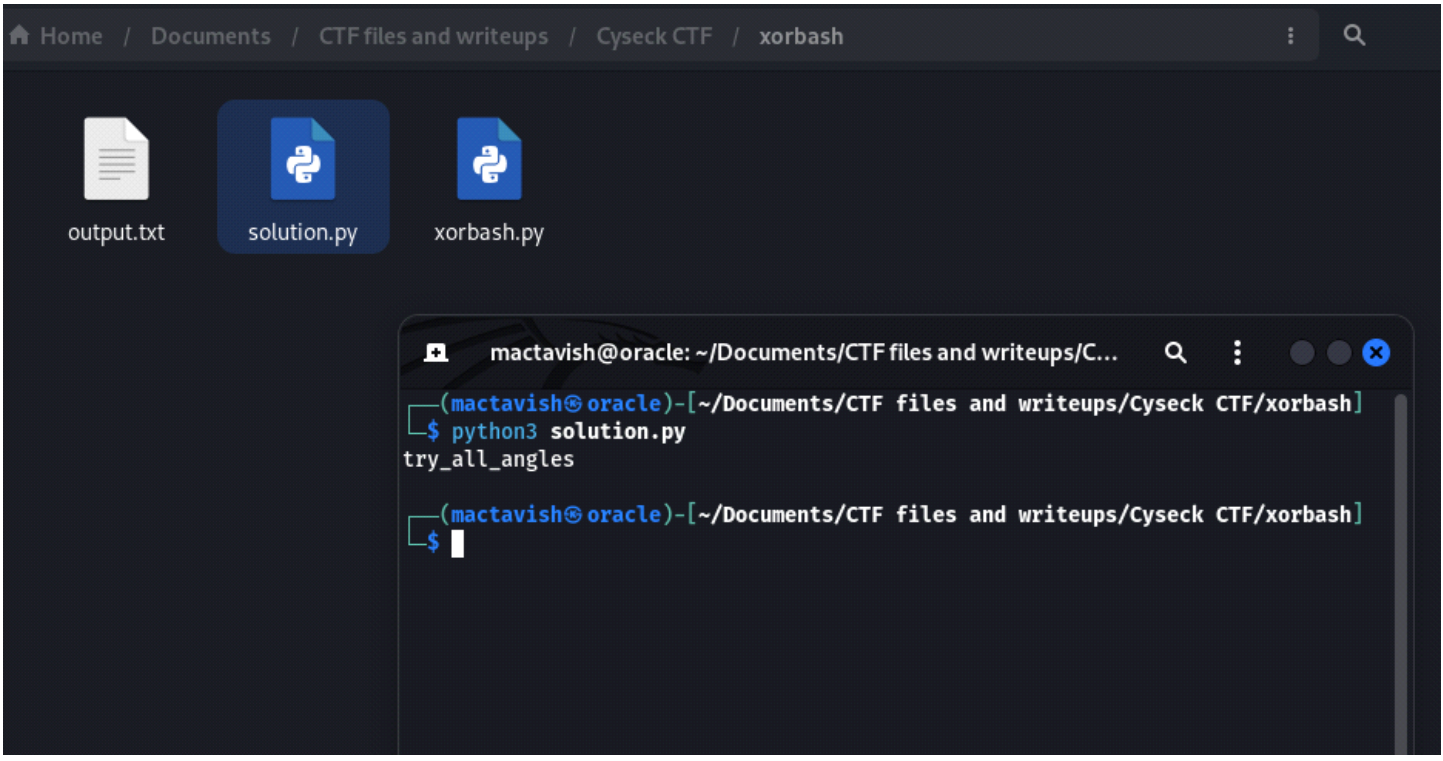
NB: Code collected from ChatGPT/StackOverflow using prompt manipulation

XOrbash

flag{try_all_angles}

08 July 2023 16:31

```
import base64

def affine_cipher(text):
    alphabet = "abcdefghijklmnopqrstuvwxyz"
    reverse_alphabet = alphabet[::-1]
    result = ""
    for char in text.lower():
        if char.isalpha():
            index = reverse_alphabet.index(char)
            reversed_char = alphabet[index]
            result += reversed_char
        elif char == '_':
            result += '_'
        else:
            result += char
    return result

def xor_cipher(text, key):
    decoded_bytes = base64.b64decode(text)
    a = decoded_bytes
    b = key.encode('utf-8')
    decrypted_bytes = bytes([a[i] ^ b[i % len(b)] for i in range(len(a))])
    decrypted_text = decrypted_bytes.decode('utf-8')
    return decrypted_text

encoded_text = "HQYQMAAAHTAAAgYADAc="
key = "zoro"

decrypted_text = xor_cipher(encoded_text, key)
reversed_text = affine_cipher(decrypted_text)

print(reversed_text)
```



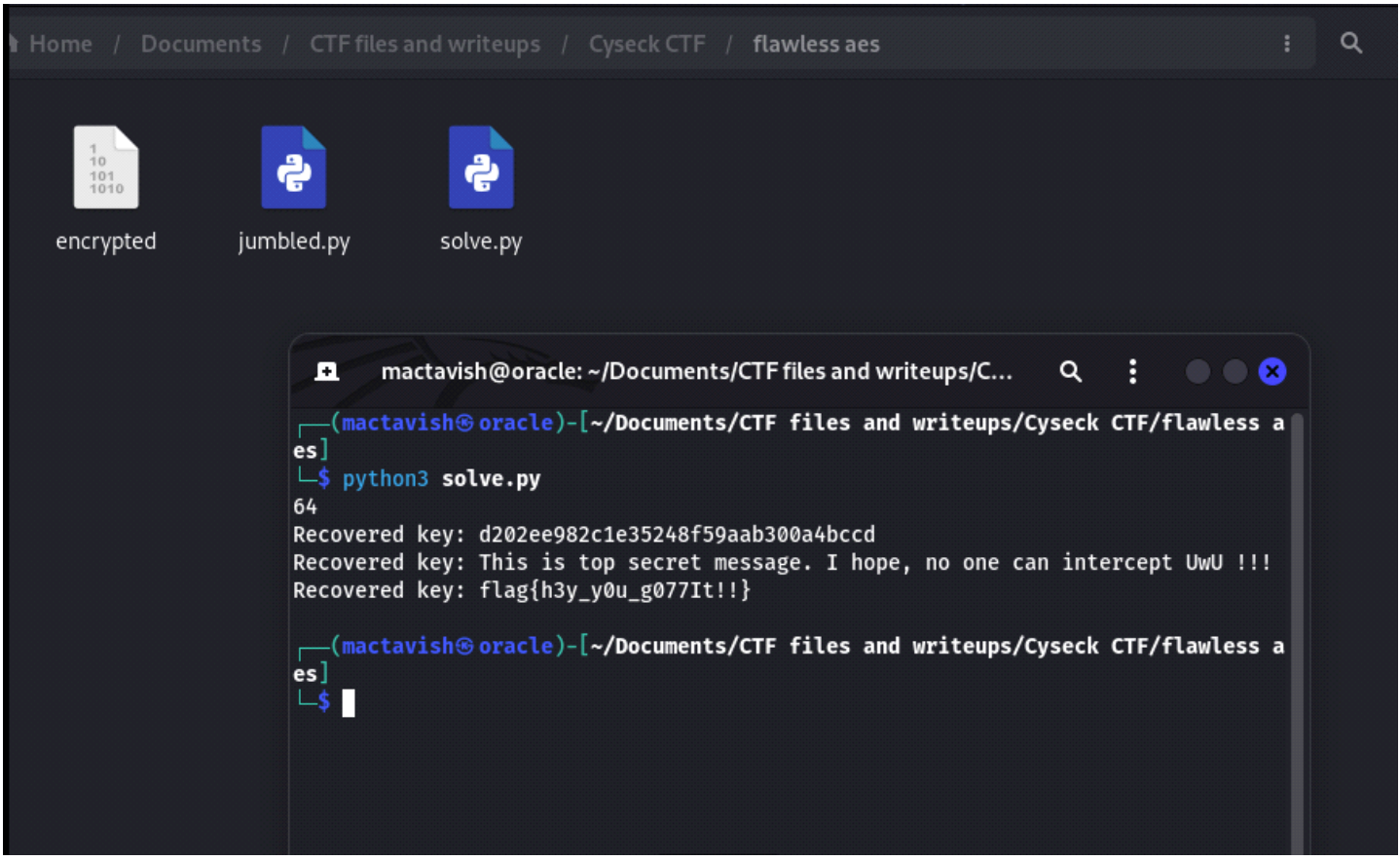
NB: Code collected from ChatGPT/StackOverflow using prompt manipulation

Flawless AES

flag{h3y_y0u_g077It!!}

09 July 2023 18:19

```
from Crypto.Cipher import AES

def xor_bytes(a, b):
   return bytes(x ^ y for x, y in zip(a, b))

with open("encrypted", "rb") as f:
   iv1 = f.read(16)
   ciphertext1 = f.read(64)
   iv2 = f.read(16)
   ciphertext2 = f.read()

# Known values


plaintext1 = b"This is top secret message. I hope, no one can intercept UwU !!!"
print(len(plaintext1))

# Set IV to all zeros
zero_iv = b"\x00" * len(iv1)

# Decrypt ciphertext1 using zero IV
cipher = AES.new(iv1, AES.MODE_CBC, zero_iv)
decrypted1 = cipher.decrypt(ciphertext1)

# Recover the original IV
recovered_iv1 = xor_bytes(decrypted1, plaintext1)
recovered_iv1 = recovered_iv1[:16]

# Print the recovered IV
print("Recovered key:", recovered_iv1.hex())
# d202ee982c1e35248f59aab300a4bccd

cipher = AES.new(iv1, AES.MODE_CBC, recovered_iv1)
decrypted1 = cipher.decrypt(ciphertext1)

print("Recovered key:", decrypted1.decode())

cipher = AES.new(recovered_iv1, AES.MODE_CBC, iv2)
decrypted2 = cipher.decrypt(ciphertext2)

print("Recovered key:", decrypted2.decode())
```



NB: Code collected from ChatGPT/StackOverflow using prompt manipulation

Treasure Count

flag{7h3_0n3P1eC3_15_R34L!!}

09 July 2023 18:25

```
with open('chall.txt','r') as f:
 data = f.read()

data = [int(data[i*2: (i+1)*2], 16) for i in range(len(data)//2) ]

def find_key(data):
 a = data[:16]
 b = [137, 80, 78, 71, 13, 10, 26, 10, 0, 0, 0, 13, 73, 72, 68, 82]
 key = [i^j for i ,j in zip(a,b)]
 return key
key = find_key(data)

def decrypt(data, key):
 return bytes([i^j for i,j in zip(data, key)])

f = open('flag.png', 'wb')
for i in range(len(data)//16):
 current = data[i*16: (i+1)*16]
 dec = decrypt(current, key)
 f.write(dec)
f.close()
```

NB: Code collected from ChatGPT/StackOverflow using prompt manipulation