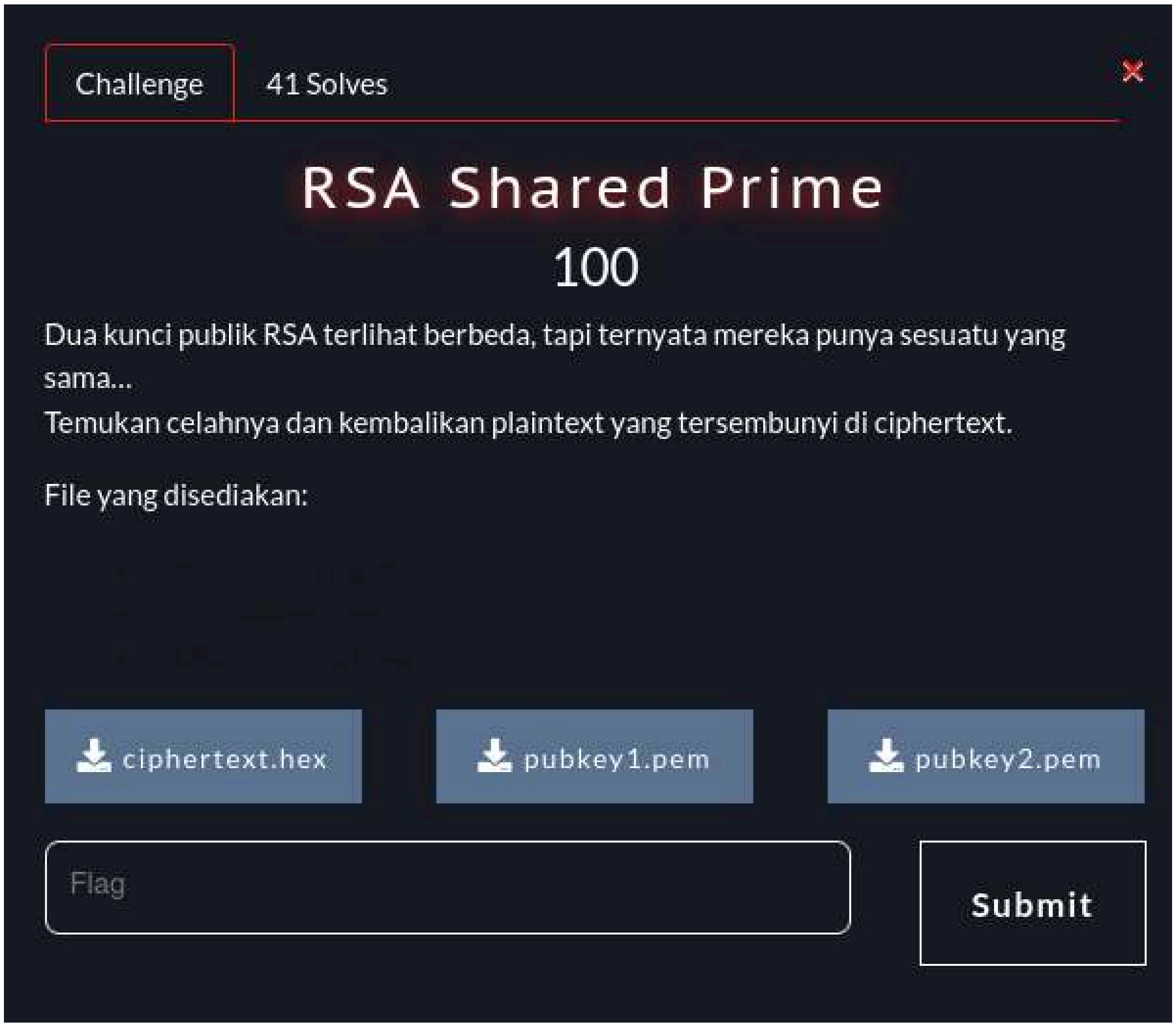
RSA Shared Prime

Player: constantine
Kategori: Cryptography



Dua kunci publik RSA terlihat berbeda, tapi ternyata mereka punya sesuatu yang sama...
Temukan celahnya dan kembalikan plaintext yang tersembunyi di ciphertext.

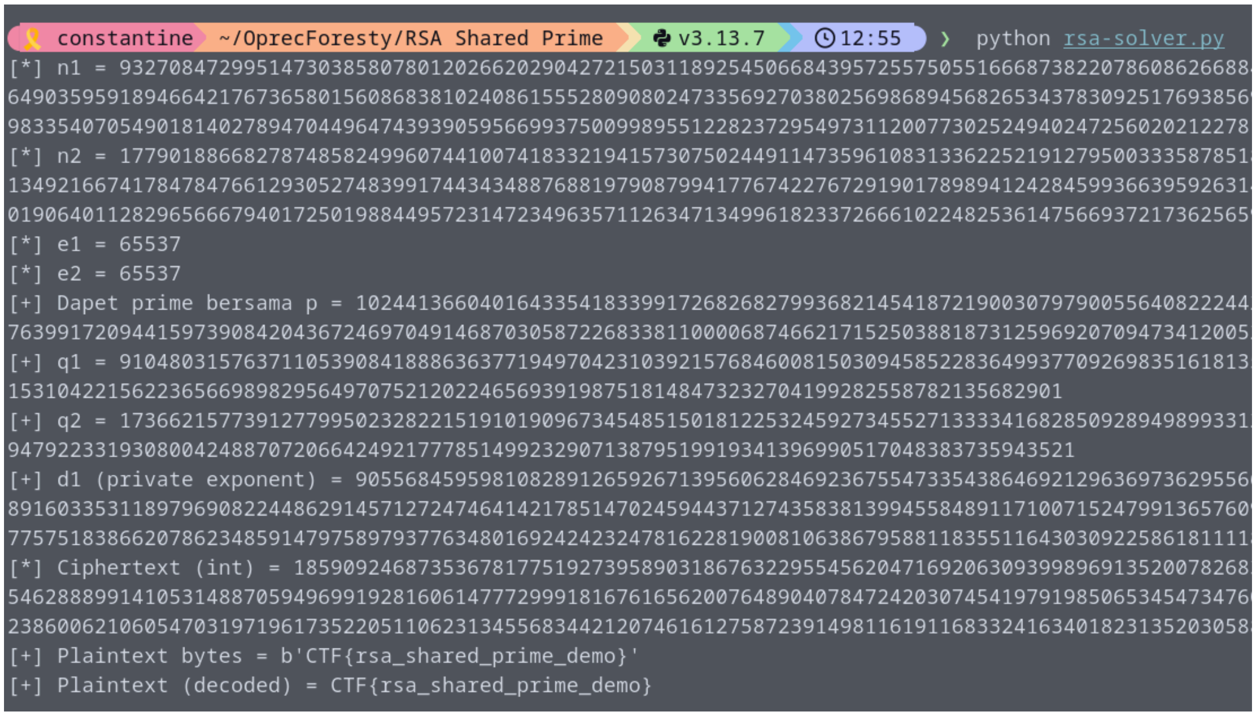
File yang disediakan:

Langsung Ambil Flag

Aku langsung lempar ketiga file ini ke ChatGPT untuk buatkan script python rsa-resolver.py. Dari situ langsung dibuatkan script **rsa-resolver.py**:

```python
from Crypto.PublicKey import RSA
from Crypto.Util.number import long_to_bytes
from math import gcd

# 1. Import 2 public key RSA
with open("pubkey1.pem", "rb") as f:
    k1 = RSA.import_key(f.read())

with open("pubkey2.pem", "rb") as f:
    k2 = RSA.import_key(f.read())

n1, e1 = k1.n, k1.e
n2, e2 = k2.n, k2.e

print("[*] n1 =", n1)
print("[*] n2 =", n2)
print("[*] e1 =", e1)
print("[*] e2 =", e2)

# 2. Cari prime yang dishare pakai gcd
p = gcd(n1, n2)

if p == 1:
    raise SystemExit("[-] Tidak ada prime yang dishare (gcd = 1). CTF ini
seharusnya shared prime, cek lagi filenya.")

print("[+] Dapet prime bersama p =", p)

# 3. Faktorkan kedua modulus
q1 = n1 // p
q2 = n2 // p

print("[+] q1 =", q1)
print("[+] q2 =", q2)

# Sanity check
assert p * q1 == n1
assert p * q2 == n2

# 4. Bangun private key untuk kunci pertama (misalnya ciphertext pakai
pubkey1)
phi1 = (p - 1) * (q1 - 1)
```

```python
# Python 3 punya modular inverse di pow
d1 = pow(e1, -1, phi1)

print("[+] d1 (private exponent) =", d1)

# 5. Baca ciphertext (format hex)
with open("ciphertext.hex", "r") as f:
    c_hex = f.read().strip()

c = int(c_hex, 16)
print("[*] Ciphertext (int) =", c)

# 6. Decrypt
m = pow(c, d1, n1)
pt = long_to_bytes(m)

print("[+] Plaintext bytes =", pt)
try:
    print("[+] Plaintext (decoded) =", pt.decode())
except UnicodeDecodeError:
    print("[+] Plaintext bukan plain UTF-8, cek bentuk lain (mungkin base64
/ flag format khusus).")
```



```
constantine  ~/OprecForesty/RSA Shared Prime   v3.13.7   12:55  ❯  python rsa-solver.py
[*] n1 = 9327084729951473038580780120266202904272150311892545066843957255750551666873822078608626688
6490359591894664217673658015608683810240861555280908024733569270380256986894568265343783092517693856
9833540705490181402789470449647439390595669937500998955122823729549731120077302524940247256020212278
[*] n2 = 1779018866827874858249960744100741833219415730750244911473596108313362252191279500333587851
1349216674178478476612930527483991744343488768819790879941776742276729190178989412428459936639592631
0190640112829656667940172501988449572314723496357112634713499618233726661022482536147566937217362565
[*] e1 = 65537
[*] e2 = 65537
[+] Dapet prime bersama p = 102441366040164335418339917268268279936821454187219003079790055640822244
7639917209441597390842043672469704914687030587226833811000068746621715250388187312596920709473412005
[+] q1 = 9104803157637110539084188863637719497042310392157684600815030945852283649937709269835161813
1531042215622365669898295649707521202246569391987518148473232704199282558782135682901
[+] q2 = 1736621577391277995023282215191019096734548515018122532459273455271333341682850928949899331
94792233193080042488707206642492177785149923290713879519919341396990517048383735943521
[+] d1 (private exponent) = 9055684595981082891265926713956062846923675547335438646921296369736295556
8916033531189796908224486291457127247464142178514702459443712743583813994558489117100715247991365760
7757518386620786234859147975897937763480169242423247816228190081063867958811835511643030922586181111
[*] Ciphertext (int) = 18590924687353678177519273958903186763229554562047169206309399896913520078268
5462888991410531488705949699192816061477729991816761656200764890407847242030745419791985065345473476
2386006210605470319719617352205110623134556834421207461612758723914981161911683324163401823135203058
[+] Plaintext bytes = b'CTF{rsa_shared_prime_demo}'
[+] Plaintext (decoded) = CTF{rsa_shared_prime_demo}
```

Screenshot output decrypt:

Final Flag

FORESTY{rsa_shared_prime_demo}