At Home

Anakin Dey

Last Edited on 7/2/23 at 21:39

chal.py

```python
from Crypto.Util.number import getRandomNBitInteger

flag = int.from_bytes(b"uiuctf{******************}", "big")

a = getRandomNBitInteger(256)
b = getRandomNBitInteger(256)
a_ = getRandomNBitInteger(256)
b_ = getRandomNBitInteger(256)

M = a * b - 1
e = a_ * M + a
d = b_ * M + b

n = (e * d - 1) // M

c = (flag * e) % n

print(f"{e = }")
print(f"{n = }")
print(f"{c = }")
```

chal.txt

```
e = 35905038915282155341613958150350534705792520856045186442663410033311656042231363926028398149...
n = 26866112476805004406608209986673337296216833710860089901238432952384811714684404001885354052...
c = 67743374462448582107440168513687520434594529331821740737396116407928111043815084665002104196...
```

Solution

At first glance with similar variable names to RSA, this problem seems like you need to do discrete log. However, we notice that $c \equiv \mathit{flag} \cdot e \pmod{n}$ rather than $c \equiv \mathit{flag}^e \pmod{n}$. If we can find d such that $e \cdot d \equiv 1 \pmod{n}$ then we're done since we can cancel out d and recover *flag*. Why does this inverse exist? Well it is because e and n are coprime. We can use the Euclidean Algorithm to compute this. Note that $e = a'M + a = a'ab + a' + a$ and $n = \frac{ed-1}{M} = a'b'ab - a'b' + ab' + a'b + 1$.

$$\begin{aligned}
n = (a'ab - a' + a)b' + a'b + 1 &\quad\Longrightarrow \gcd(e, n) = \gcd(a'b + 1, a'ab - a' + a) \\
a'ab - a' + a = (a'b + 1)a - a' &\quad\Longrightarrow \gcd(a'b + 1, a'ab - a' + a) = \gcd(a'b + 1, -a') \\
a'b + 1 = (-a')(-b) + 1 &\quad\Longrightarrow \gcd(a'b + 1, -a') = \gcd(-a', 1) = 1
\end{aligned}$$

This means that such d exists such that $d \cdot e \equiv 1 \pmod{n}$. You can actually find this d by reversing this algorithm. However, using Python is so much easier: `pow(e, -1, n)`. Thus we get that

$$c \cdot d \equiv (e \cdot \mathit{flag}) \cdot d \equiv \mathit{flag} \pmod{n}.$$

```python
res = long_to_bytes(flag)
print(res) # uiuctf{W3_hav3_R5A_@_h0m3}
```

□