[Crypto] Telur (436 points)

By: FieryBanana101

chall.py:

```python
from sage.all import next_prime
from sage.all import RealField
from sage.all import ZZ
from Crypto.Util.number import getPrime
from Crypto.Util.number import bytes_to_long as b2l

def get_primes(bit_length):
       return getPrime(bit_length), getPrime(bit_length)

FLAG = b'CTFITB{*}'

p, q = get_primes(512)
n = p * q
e = 0x10001

ct = pow(b2l(FLAG), e, n)

tp = next_prime(p)
tq = next_prime(q)

leak = RealField(2048)(ZZ(tp)/ZZ(tq))

print(f'ct = {ct}')
print(f'e = {e}')
print(f'n = {n}')
print(f'leak = {leak}')
```

Pada *chall.py* dilakukan skema enkripsi RSA dengan generasi *private key* p dan q 512 bit. Skema dilakukan secara normal untuk mengenkripsi flag, hanya saja diberikan sebuah petunjuk berupa:

$$tp = nextPrime(p)$$
$$tq = nextPrime(q)$$
$$leak = \frac{tp}{tq}$$

Digunakan *real division*, bukan *floor division*. Hal ini mengakibatkan kita dapat melakukan pemulihan nilai p dan q dengan teknik yang trivial. Perhatikan bahwa:

$$tp = p + 2 \cdot x_1$$
$$tq = q + 2 \cdot x_2$$
$$x_1, x_2 \in \mathbb{Z}^+$$

Kemudian kita dapat melakukan manipulasi aljabar sebagai berikut:

$$leak = \frac{tp}{tq} = \frac{p + 2 \cdot x_1}{q + 2 \cdot x_2}$$

$$q \cdot leak + 2 \cdot x_2 \cdot leak = p + 2 \cdot x_1$$

$$p = q \cdot leak + 2 \cdot x_2 \cdot leak - 2 \cdot x_1$$

$$p \cdot q = q^2 \cdot leak + 2 \cdot q \cdot (x_2 \cdot leak - x_1)$$

$$n = q^2 \cdot leak + 2 \cdot q \cdot (x_2 \cdot leak - x_1)$$

$$q^2 \cdot leak + 2 \cdot q \cdot (x_2 \cdot leak - x_1) - n = 0$$

Dihasilkan persamaan kuadrat dengan q sebagai variabel. Nilai x_1 dan x_2 dapat kita *bruteforce*, kemudian faktor dari polinomial tersebut adalah kemungkinan nilai q, tinggal kita validasi saja dengan memastikan n % q == 0. *Upper Bound* dari nilai x_1 dan x_2 dapat diperkirakan dengan melakukan beberapa tes lokal:

```
sage: from Crypto.Util.number import *
sage: def check():
....:     p = getPrime(512)
....:     np = next_prime(p)
....:     return (np-p) // 2
....:
sage: print([check() for _ in range(20)])
[50, 7, 485, 29, 95, 51, 438, 50, 211, 58, 67, 147, 23, 29, 81, 219, 63, 38, 547, 30]
```

Diperkirakan *upper bound* yang dibutuhkan hanya sekitar 500-600 (kemungkinan besar *gap* sebenarnya lebih kecil). Ketika q sudah ditemukan, maka kita dapat lanjut melakukan dekripsi secara trivial dengan skema standar RSA. Berikut adalah implementasi solver menggunakan SageMath.

solver.sage:

```
from Crypto.Util.number import long_to_bytes as l2b

exec(preparse(open("out.txt", "r").read())) # ct, n, e, leak

F.<q> = ZZ[]

for x1 in range(1, 600):
   for x2 in range(1, 600):

      poly = leak * q**2 + 2 * q * (leak*x2 - x1) - n

      for factor in poly.factor():
          result = floor(factor[0][0])
          if n % result == 0:
              _q = result
              _p = n // _q
              print(f"[FOUND] p: {_p}")
              print(f"[FOUND] q: {_q}")
              print("[FLAG] ", l2b(pow(ct, pow(e, -1, (_p-1)*(_q-1), ),n)))
              print()
```

FLAG: CTFITB{aseli-padang-style-omelette-with-chicken-and-duck-eggs-insert-emot-ngiler}