Name: Tan Hoe Zhun
Challenge Category: Cryptography
Challenge Name: Small-E 2

Challenge Description

Damn! Someone decrypted my flag!
Nevermind, I have upgraded my python script,
now no one can decrypt my flag hehe..
Flag format: SKR{flag} or SKR{m}

Given File: chall.py

```python
from Crypto.Util.number import *
from secret import flag
# Generate 2 large prime number
p = getPrime(512)
q = getPrime(512)
n = p*q
e = 5
# Convert bytes string to decimal number
m = bytes_to_long(flag)
# Encrypt the flag
c = pow(m,e,n)
print(f"n={n}")
print(f"e={e}")
print(f"c={c}")
# Output:
#
  n=919160629297558999914194520987760702112574145968752182753806033775918701826034353875927995976016774127254633300226183044919672260952745327015953955130814877868807743752612427199628430533320948173897058015216070976440460549578957184240755146721649462088678400117629334320756459420108873157724863540777530989 21
# e=5
#
  c=357832435534842730906770899270599909200075990844991435866131828950285184980966022896989910441522166746329524841030494226710442574040095880385225595151187204790258715642185579410676018156020709125352206111647619048494388279075516207156080946451946205715661188910250172180471597232722773109542998263594596038 93
```

Hints:

1. What happens when the difference between m^e and n is small?
2. What is modular arithmetic?

Solution

1. The first hint suggests that the difference between m^e and a multiple of n is small. In simpler terms, m^e is very close to $k * n$ for some integer k. From the encrypted message, we know:

   $c \equiv m^e \bmod n$

   Which means:

   $c \approx m^e - k * n$

   The difference of $m^e - c$ is small. So, for some small value of **x**, $c + x * n$ would be a perfect 5th power (since $e = 5$)

2. The goal is to find a small **x** such that $c + x * n$ is the perfect 5th power of some integer. Once this value is found, taking the 5th root gives the original message m.

3. Using the `gmpy2` library, we can efficiently check for perfect powers. We iterate over a range of possible values for **x** and check if $c + x * n$ is a perfect 5th power.

```python
from gmpy2 import iroot
from Crypto.Util.number import long_to_bytes

n = 919160629297558999914194520987760702112574145968752182753806033775918701826034353875922
e = 5
c = 357832435534842730906770899270599909200075990844991435866131828950285184980966022896982

for X in range(1000):
    root, exact = iroot(c + X * n, e)
    if exact:
        print(long_to_bytes(root).decode('utf-8'))
        break
```

```
┌──(zhun㉿kali)-[~/CTF/SKR_CTF/cryptography/Small-E_2]
└─$ python decrypt.py
SKR{17_St1ll_t00_5m411}
```

By exploiting properties of modular arithmetic and understanding the hints about the nearness of m^e to a multiple of n, we were able to decrypt the ciphertext without needing the private key.