

e=3

n1=93500162517048641546787096055793757535647246115573452820070426952722839105259206386726342320373080028589838409420934957286736144380948165832820988099621769638760228520922601557891245459269555695273287461982149159244732888497451549014876606240205619538734235283434466606740831016126239969278127395932813972501

ct1= 78406286729520927933597606356229817207730090348466337528850035686907731769583595879598630905387554797365268804265749503975166670245660278708562880234673953125

n2=69389646088454519961777612972795170975757344514768971340015442739948594078401371455619333135568702919584689527569525412014628819453197654891812622405663009392306477333014587313114349970319506949532748028160803547659404882094671572233644067789871312676667284969540366763124836901936574299355649024020474292507

ct=2 78406286729520927933597606356229817207730090348466337528850035686907731769583595879598630905387554797365268804265749503975166670245660278708562880234673953125

n3=173343492679580453099087472373077831318468581021117934856501363981932357178004338045575880395223727472909876082267313542279706329989182144352883996336740679533245920391761546352887945995038709117785508115723234046039796085997076590595344206748962306765136674210538263394651485897800900761481602355016363501601

ct3=78406286729520927933597606356229817207730090348466337528850035686907731769583595879598630905387554797365268804265749503975166670245660278708562880234673953125

To solve the challenge using the Chinese Remainder Theorem (CRT) in the context of RSA encryption and decryption, we first need to understand the components involved. The CRT provides a systematic way to solve systems of simultaneous congruences with different moduli, which is particularly useful in optimizing the decryption process of RSA.

We can use crt () function in sympy library in python.

For that we need to import crt function from the sympy.ntheory.modular module.

***from sympy.ntheory.modular import crt***

We create lists fro the moduli and ciphertexts.

***moduli = [n1, n2, n3]***

***ciphertexts = [ct1, ct2, ct3]***

We apply the crt function to compute the combined result and the product of the moduli.

***Result, combined\_result = crt(moduli, ciphertexts)***

**Result**: This is the combined result that satisfies all the given congruences, representing M^e mod  (n1\* n2 \* n3)

**Combined Result**: This is the product of the moduli n1\* n2 \* n3 ​, which defines the range for potential plaintext values M.

Result from the CRT represents M ^ e Where e is public exponent (here 3)

So decript the message we need to find eth root of Result . We can use iroot function from gmpy2 module.

**from gmpy2 import iroot**

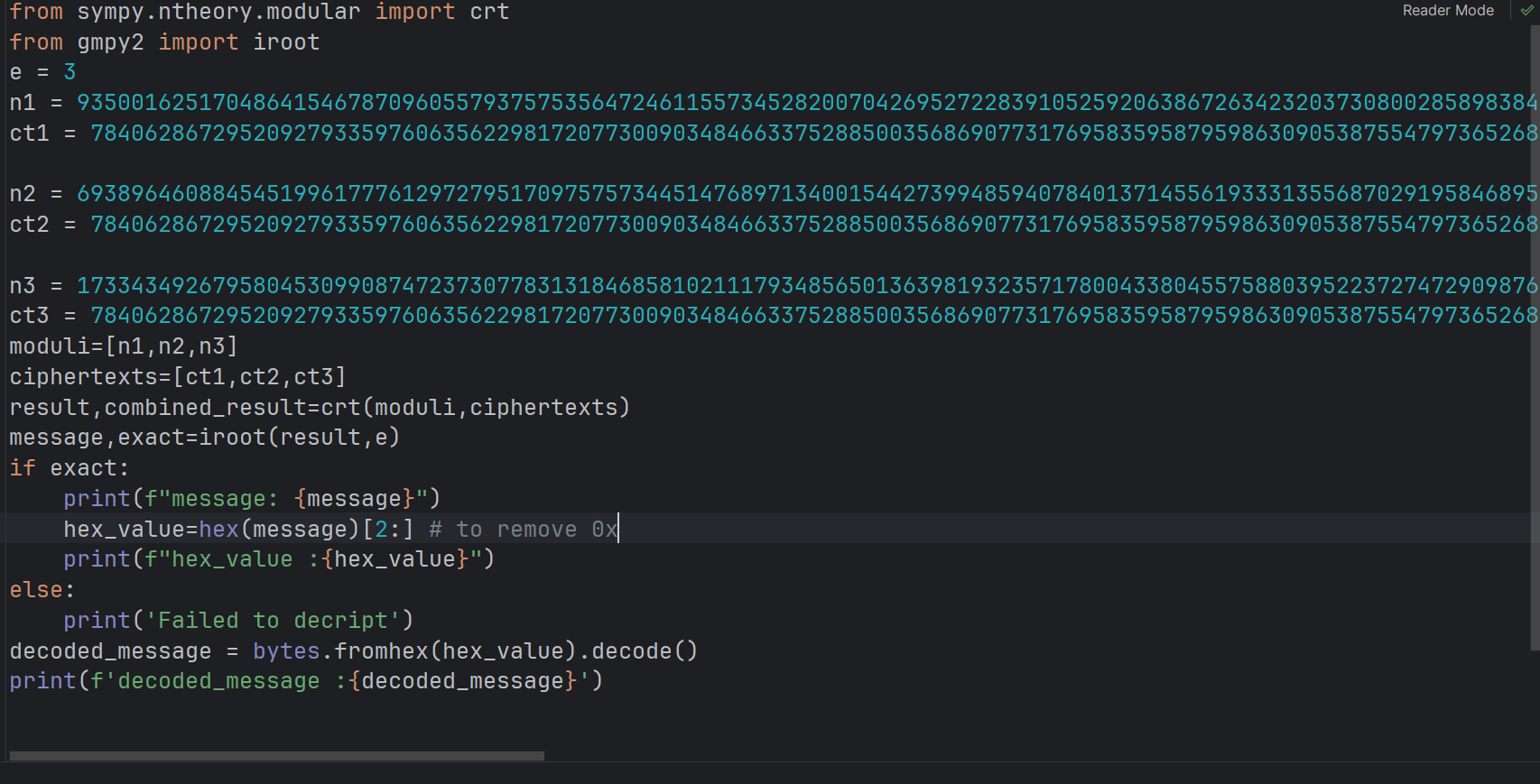
**Message,exact=iroot(Result,e)**

exact is a boolean value indicating whether the computed root is exact or not .

So if exact we will convert to hexadecimal value to decode it.

***if exact:  
 print(f"message : {message}")  
 hex\_value=hex(message)[2:]  
 print(hex\_value)  
else:  
 print('Failed to decript')  
  
decoded\_message = bytes.fromhex(hex\_value).decode()  
print(decoded\_message)***

**Decoded Message = recr{H4s4d\_is\_t0\_g00d}**

****

**OUTPUT:**

