Tấn công RSA

Đôi điều về RSA, các đại lượng trong bài toán RSA bao gồm: N , p , q , e , phi, m , c

Trong đó

N là modulus , được tao nên từ tích của p và q -> N = p\*q

Với p và q là 2 số NGUYÊN TỐ

Public exponent e có thể là số NGUYÊN TỐ hoặc KHÔNG NGUYÊN TỐ

Đại lượng phi = ( p – 1 ) \* ( q – 1 ) sao cho gcd(e,phi) = 1

Khi gcd(e,phi) = 1 thì sẽ tồn tại trong miền phi 1 số d sao cho e\*d = 1 mod phi , và d ở đây là private exponent

Ta có 1 message m , ta sẽ mã hóa message với với e và N theo công thức

Ciphertext c = pow(m,e,N)

Và giải mã ciphertext c với d và N theo công thức

Plaintext m = pow(c,d,N)

Thường thì đề sẽ cho ta Pubkey(e,N) và Ciphertext c

Ở đây có 2 trường hợp

* Nếu từ N ta có thể phân tích ra được 2 số p và q thì bài toán trở nên cực kì đơn giản, ta có p và q, tìm được phi, ta có e và phi tìm được d, có PribKey(d,N) giải mã ciphertext tìm lại được message

Với dạng đó thì trước tiên cứ bỏ vô trang factordb.com để xem thử xem N có phân tích được hay không rồi tính tiếp

* Nếu như không thể phân tích được N , thì ta có thể lợi dụng các gợi ý hoặc các điểm yếu của Pubkey để tìm cách phục hồi message ban đầu và mình sẽ đi vào 1 vài cách tấn công dạng này

**Dạng 1: tấn công vào public exponent e**

**a/ Public exponent rất bé**

Khi e = 3 thì tốc độ mã hóa sẽ khá nhanh, đảm bảo tính nhanh gọn lẹ nhưng về mức độ an toàn thì không cao

**<Hastad’s Broadcast Attack>**

Ví dụ, ta có 1 message M, được gửi đến 3 nơi khác nhau với cùng 1 public exponent e = 3 với các Pubkey:

Pubkey1(e,N1) - Pubkey2(e,N2) – Pubkey3(e,N3)

Lúc đó ta sẽ có:

C1 = pow(M,e,N1)

C2 = pow(M,e,N2)

C3 = pow(M,e,N3)

Thì ta có thể phục hồi được message M ban đầu mà thậm chí không cần quan tâm đến d1,d2,d3

Bằng cách sử dụng CRT ( Chinesse Remander Theorem ) ta có thể dễ dàng chứng minh được sẽ tồn tại 1 số C sao cho

C = pow(M,3,N1\*N2\*N3)

Gọi

T1 = C1\*(N2\*N3)\*invert\_modular(N2\*N3,N1)

T2 = C2\*(N1\*N3)\*invert\_modular(N1\*N3,N2)

T3 = C3\*(N1\*N2)\*invert\_modular(N1\*N2,N3)

Lúc này C = (T1 + T2 + T3) % (N1\*N2\*N3)

Vậy C = pow(M,3,N1\*N2\*N3)

⬄ C = pow(M,3)

⬄ M = pow ( C , 1/3 )

Ta tìm được Message M ban đầu

Các bạn có thể tham khảo thêm ở [đây](https://crypto.stackexchange.com/questions/6713/low-public-exponent-attack-for-rsa)

Bài tập minh họa: <Broadcast – PicoCTF2017>

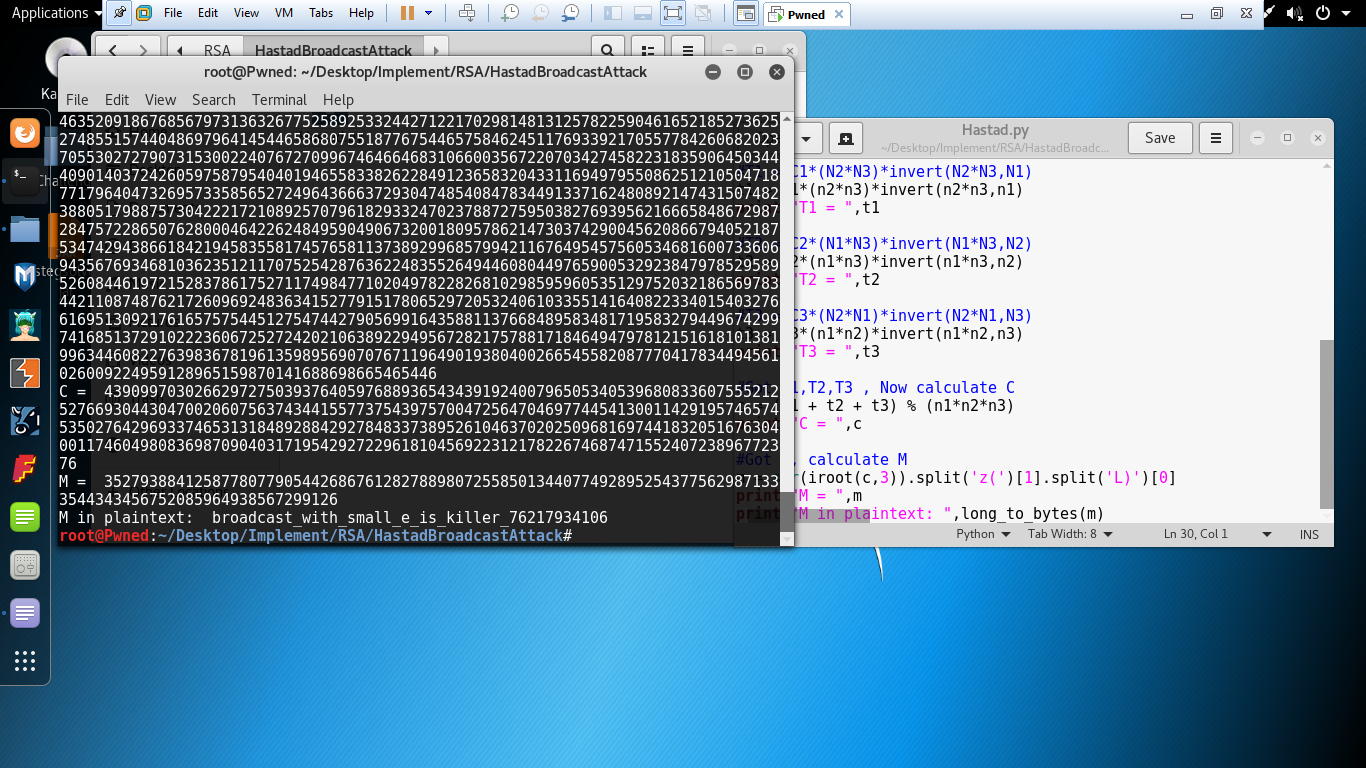
<https://gist.github.com/anonymous/e78677123d95fe1d5f8ad6682eee4165>

Theo đề bài, 1 message M đã được mã hóa và gửi đi 3 nơi , và đề cho ta biết các đại lượng là

C1,C2,C3,N1,N2,N3,e trong đó e = 3

Ta áp dụng <Hastad’s Broadcast Attack> ( yêu cầu các bạn phải có kiến thức về lập trình ít nhất 1 trong các ngôn ngữ sau Java , Python , C++ , C 😊 , Assembly 😊 )

Mình sẽ làm bài này bằng python với các Module hỗ trọ là PYCRYPTO và GMPY2



[Script](https://gist.github.com/anonymous/fe0cdc5d068b15f20b4928c87f207b3c)

**<Coopersmith Attack>**

Ta có

Ciphertext C = pow(M,e,N)

⬄ pow(M,e,N) - C = 0

Ví dụ M = hello my name is xxxxxxxx

Với xxxxxxxxxx là phần không biết, ta đặt xxxxxxxx = X

Với phần biết rồi là hello my name is là m

Vậy M = m + x

Ta có M = m + x

* F(x) = pow ( m + x , e , N ) – C

Với coopersmith ta có thể tìm được các giải pháp tìm lại toàn bộ message với x < pow(N,1/e)

Việc của ta là tìm x sao cho thỏa f(x) = 0

Dễ hiểu hơn

Ta có message M = 1234

C = pow ( M , e , N )

Giờ ta biết 1 phần message là 1000

Đăt 1000 = m

Phần không biết ta đặt là x

Giờ ta sẽ tìm x sao cho F(x) = 0 với F(x) = pow(1000 + x ,e , N ) – C

⬄ với F(234) = pow ( 1234,e,N) – C = 0 Thỏa

Ta tìm được x = 234, quá dễ đúng k

Lưu ý là với e nhỏ :D chứ e lớn chà bá thì quỳ

Bài tập ví dụ:

<https://gist.github.com/anonymous/294eded05a21424ed14db9a0d15b5146>

Với message bị lộ = “Hi im xxxx”

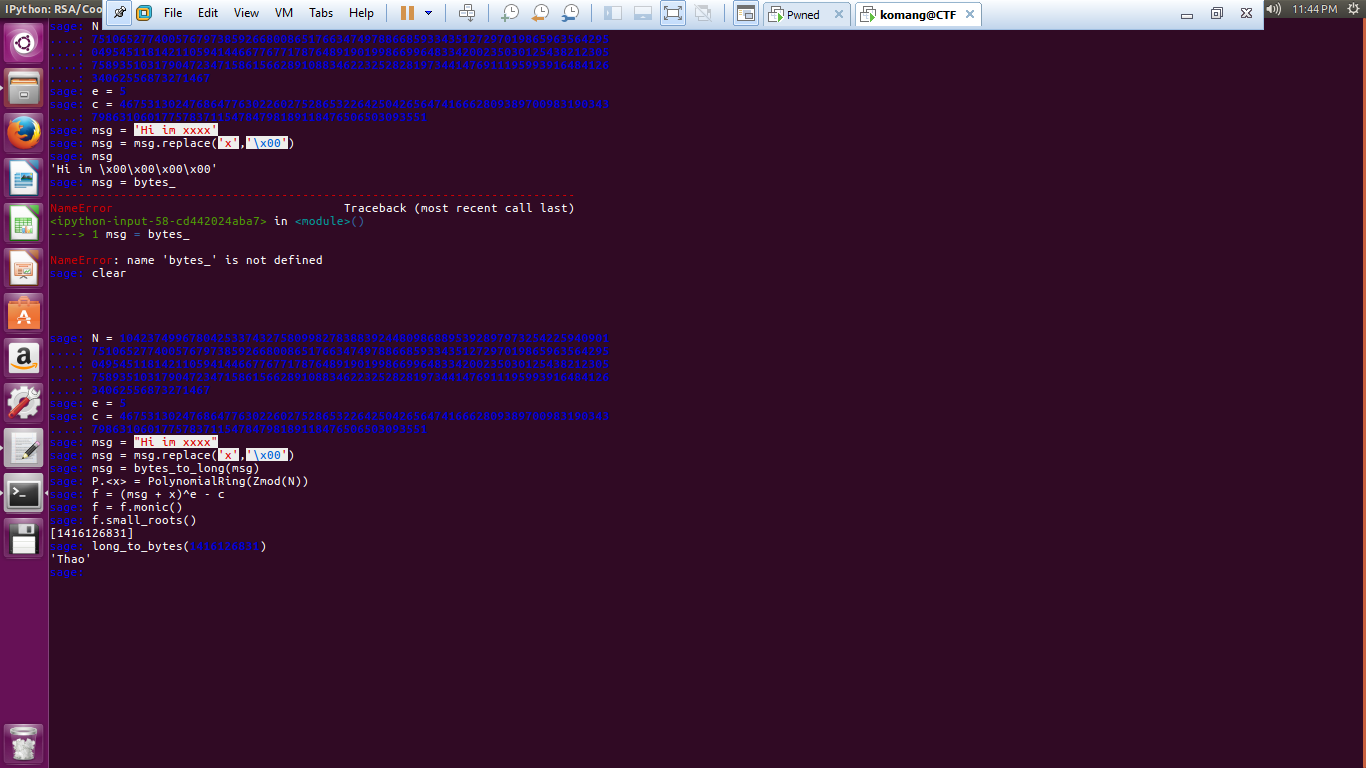
Trong đó xxxx là phần ta không biết

Theo như quan sát, ta thấy e rất bé, ta có thể áp dụng Coopersmith lên nó

\*yêu cầu các bạn sử dụng Sage ( các bạn tự tìm hiểu và cài đặt )

Giờ ta sẽ đổi phần xxxx thành các biến null là \x00

Sau đó đổi msg sang int , tiếp tục là xác định hàm Polynomial và tính toán :D

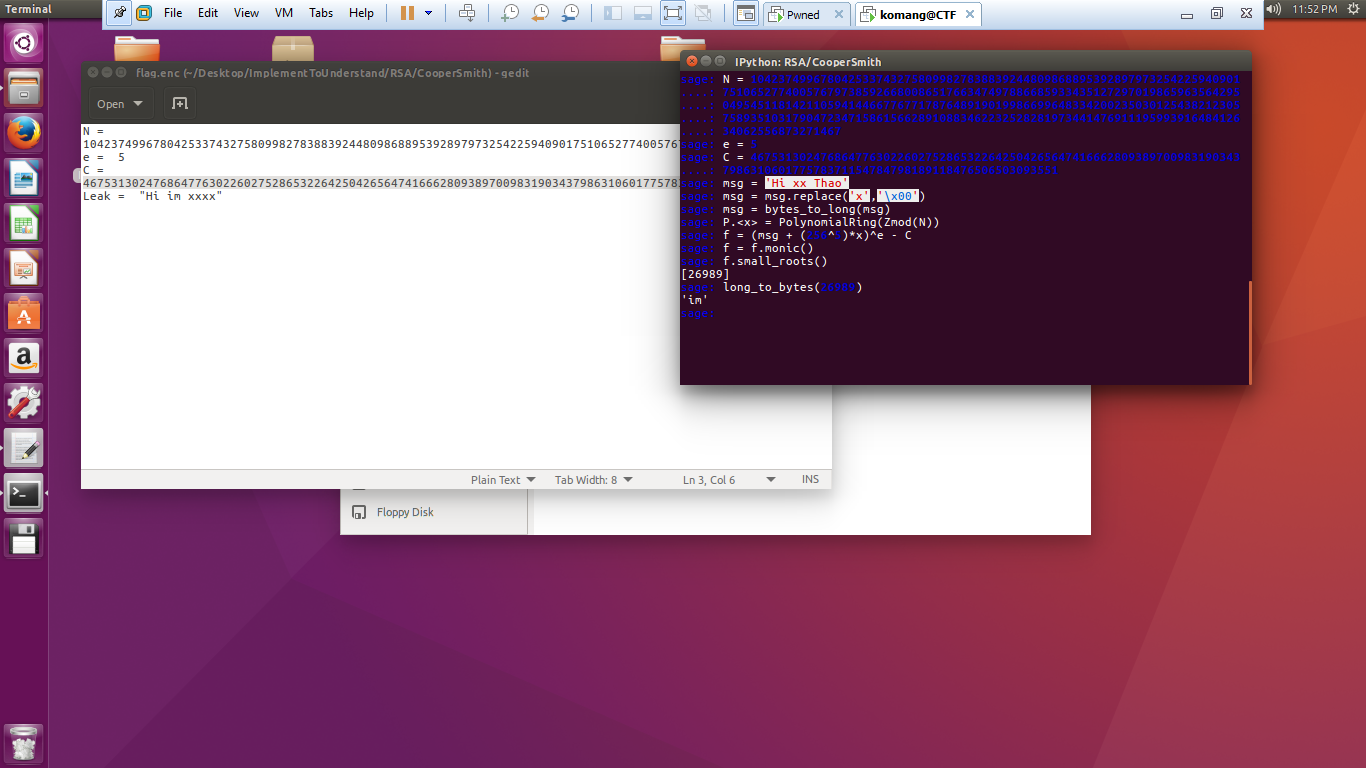


Khỏi cần tìm p,q,d gì cả vui không hehe :D

Tiếp tục giờ giả sử phần bị che là : “Hi xx Thao” thì sao?

Ta sẽ tính toán: từ sau chữ xx là 5 chữ -> 256^5

Lúc này F(x) = (msg + (256^5)\*x)^e – c



Đẹp trai luôn <3

**Franklin related message attack**

Thường kiểu tấn công này sẽ có dạng

Với e bé

C1 = pow(M1,e,N)

⬄ pow(M1,e,N) – C1 = 0 (1)

C2 = pow(M1 + diff , e , N )

Đặt M2 = M1 + diff

⬄ C2 = pow(M2,e,N) (2)

⬄ pow(M2,e,N) – C2 = 0

Ta có

1. Ta có g(x1) = pow(x1,e) – C1
2. Ta lại có h(x2) = pow(x2,e) – C2 hay h(x2) = pow(x1 + diff,e) – C2

Ta tìm gcd của g và h -> -gcd(g,h).coefficents[0] sẽ trả về cho ta kết quả M1.

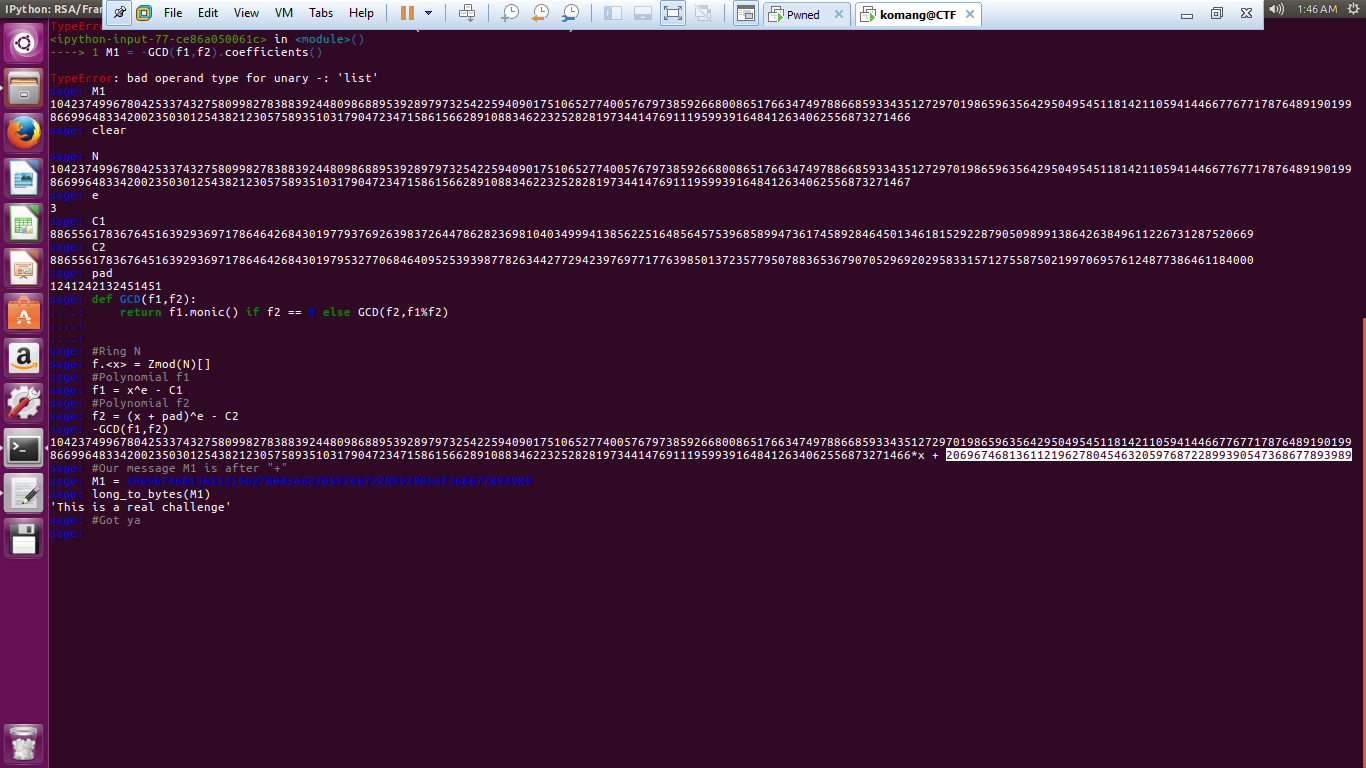
Ví dụ minh họa:

<https://gist.github.com/anonymous/59f8bb86cdd077308eeebd9398d5ff5c>

Ta thấy  
C1 = pow ( M1 , e , N ) với e = 3

C2 = pow ( M2 , e , N ) với M2 = M1 + pad

Cách làm \*\* yêu cầu phải có sage



Vẫn tấn công tìm được message mà không cần tìm d , p , q :D

**<Partial Key Exposed Attack>**

Về dạng attack này thì nôm na nếu như các bạn đã biết bit thấp của d = ¼ bit của N, bạn có thể khôi phục được d

Ngoài ra còn có nhiều biến thể tấn công nếu biết bit m , p , d

Các bạn có thể tham khảo tài liệu rất rõ ở đây: <http://inaz2.hatenablog.com/entry/2016/01/20/022936>

**b/ Public exponent e rất lớn**

Để tốc độ giải mã nhanh hơn thì người ta thường chọn Private key d nhỏ -> e sẽ lớn

**<Weiner Attack>**

Nôm na với 1 nùi chứng minh ta sẽ tìm được cặp k/d = e/N :v

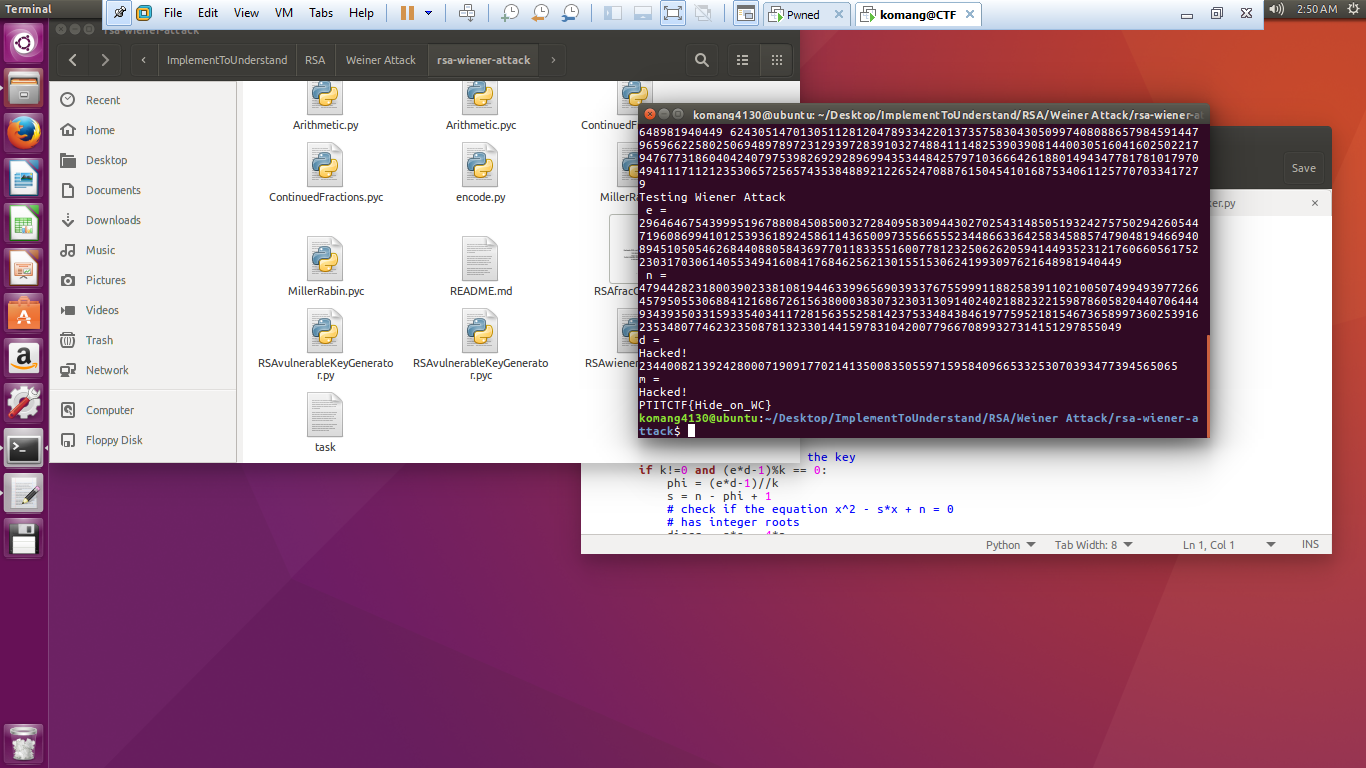
Điều kiện : d < pow(N,1/4) / 3

Các bạn có thể tải tool Weiner có sẵn tại : <https://github.com/pablocelayes/rsa-wiener-attack>

Ví dụ minh họa : <https://gist.github.com/anonymous/abf2739868d2a8289f5e14b5a70c951b>

Ta thấy rằng bài này e rất lớn, mình đã modify file [RSAweinerhacker.py](https://gist.github.com/anonymous/2163bfacafcc811d3029e2a07b7e4324) 1 tí

Giờ các bạn chỉ cần truyền đối số vào theo thứ tự n , e , c là xong



Ez right?

**<Boneh Duffee Attack>**

Kiểu tấn công này được áp dụng khi d <= pow(N,0.292)

Ví dụ minh họa:

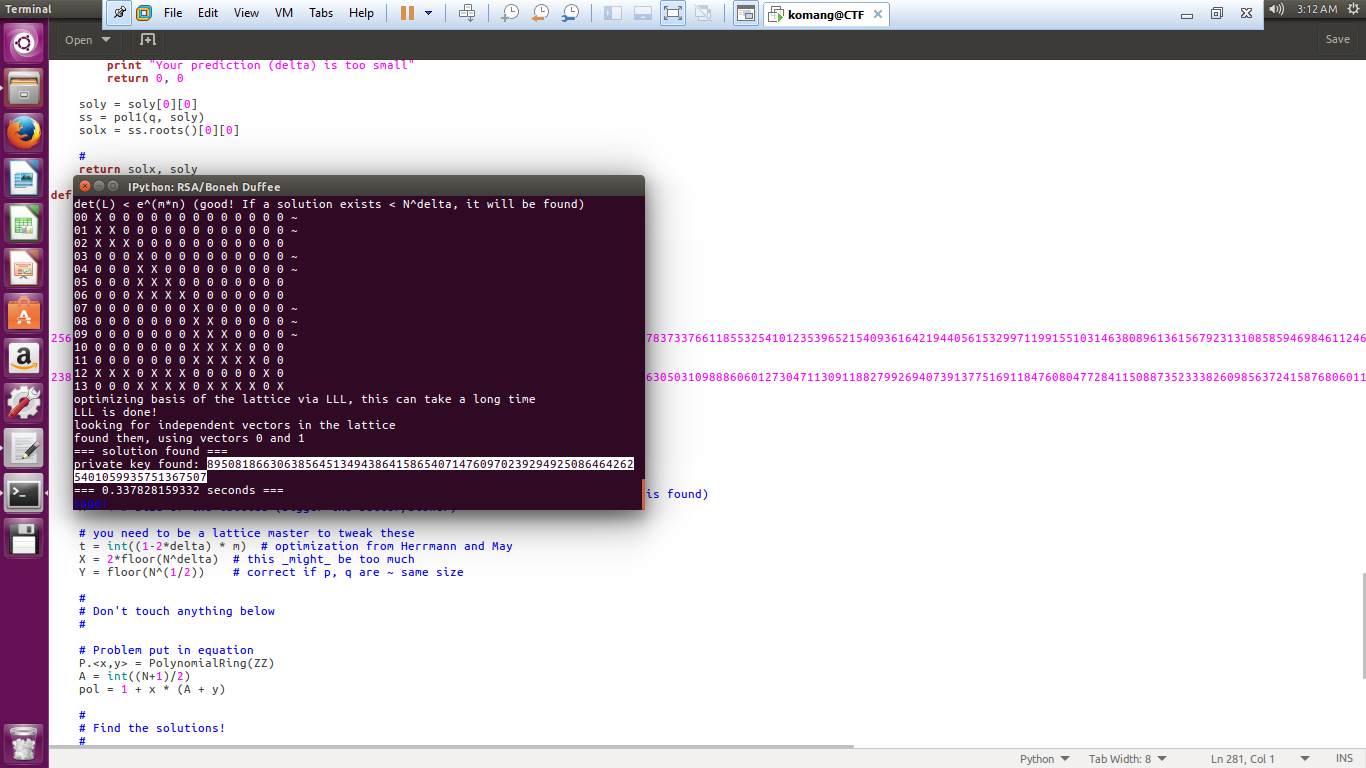
<https://gist.github.com/anonymous/e52535e3c1bb11ab1d011c22cd78b192>

Các bạn có thể tham khảo : <https://github.com/mimoo/RSA-and-LLL-attacks>

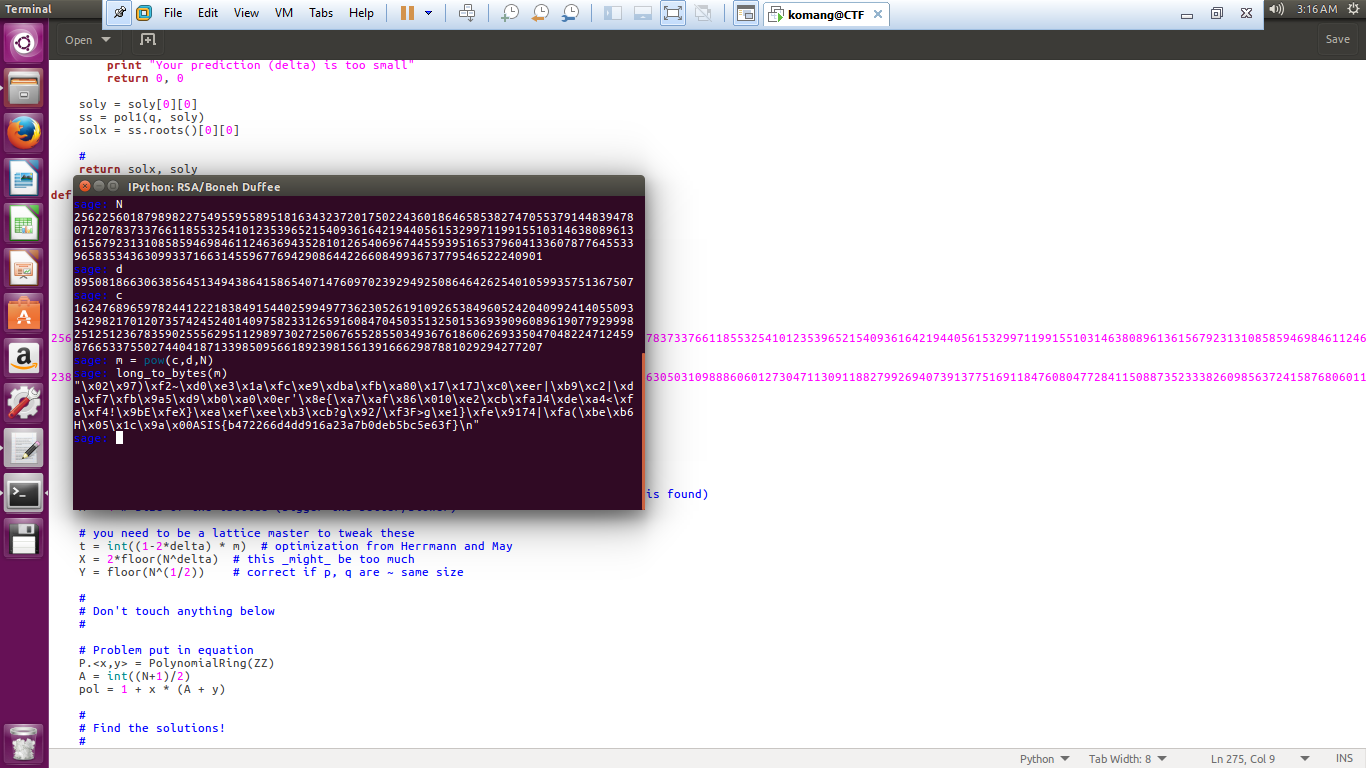
Mình sẽ sử dụng Script **boneh\_durfee.sage \*\*Yêu cầu phải có sage**

**\*\*delta = .25**

**Ta tìm được d**



Tìm lại m thôi :D



**Dạng 2: Tấn công vào Modulus**

Thì như các bạn đã biết, nếu với bài toán RSA ta đã tìm được p và q và có cả e , c thì ta có thể phục hồi m dễ dàng

Vậy nên tấn công vào Modulus thực tế là BRUTE , quan trọng là thời gian brute nhanh hay chậm và dựa vào tính chất và mối quan hệ của p và q để boost tốc độ factor :D

**a/ Modulus Factorization with p and q are close Prime ( Fermat Factorize )**

Nếu như p và q là 2 số nguyên tố gần nhau ( ví dụ 7 và 11 ) thì ta có thể brute tìm p và q nhanh hơn

Qua 1 đống thuật toán và phép toán tùm lum tá lả gì đấy ( sorry đọc xong đầu quay vòng vòng không phân tích nổi :v )

Bài tập minh họa:

<http://material.wargame.whitehat.vn/challenges/2/Crypto001_f7f2e7fd395e40903f0e24e2a52f4b2e.zip>

Ta để ý file encrypt có đoạn sau:

# get 2 first prime number from random\_seed

p, q = 0,0

while True:

if is\_prime(num):

p = num

break

num+=1

while True:

if is\_prime(num):

q = num

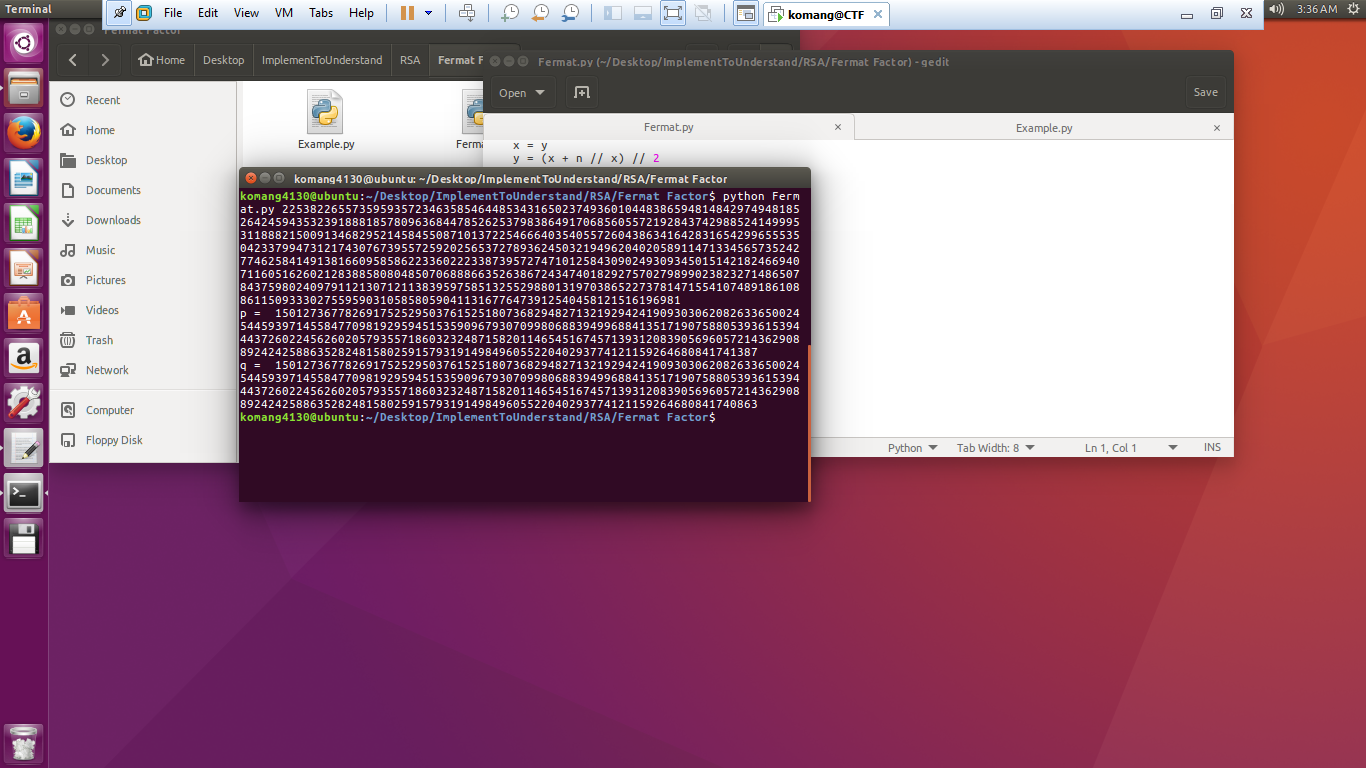
break

num+=1

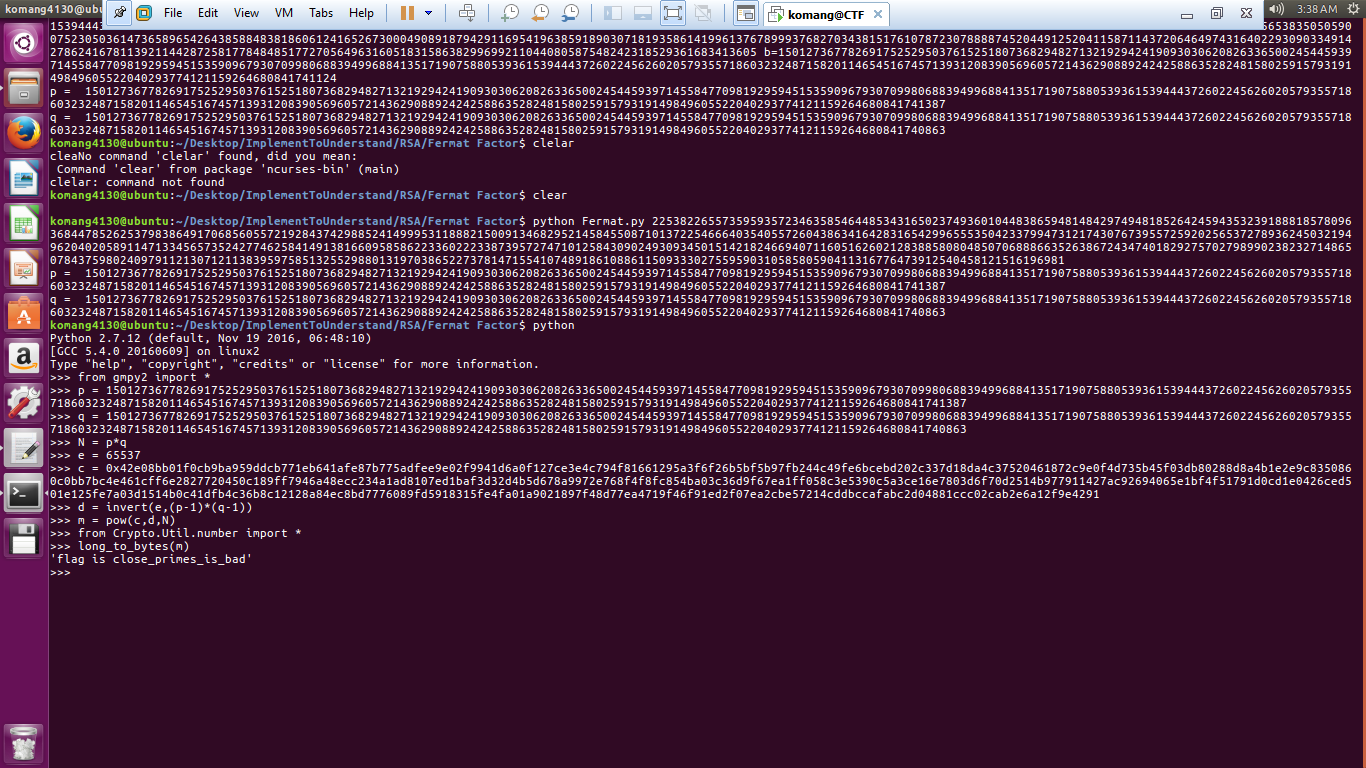
Ok p và q là close prime chứ gì nữa

Ta sẽ sử dụng Fermat theory để tiến hành factor N tìm p và q

[Script](https://gist.github.com/anonymous/5a7321091d7942f8fb0b49886fe58618)



Tìm được p và q , dễ dàng tìm được d và giải mã :D



**b/ Commom Modulus attack**

Giờ ta có 1 message, nhưng được mã hóa với 2 public exponent khác nhau và dùng chung 1 Modulus N

Gọi e1,e2 là 2 public exponent đó

Ta có

C1 = pow(M,e1,N)

C2 = pow(M,e2,N)

Nếu gcd(e1,e2) = 1 -> tồn tại a và b sao cho a\*e1 + b\*e2 = 1

Lúc đó:

C1\*C2 = pow(M,e1,N) \* pow(M,e1,N)

⬄ pow(C1,a)\*pow(C2,b) = pow(M,e1\*a,N) \* pow ( M,e2\*b,N )

⬄ pow(C1,a)\*pow(C2,b) = pow( M , e1\*a + e2\*b , N)

⬄ pow(C1,a)\*pow(C2,b) = pow(M,1,N)

⬄ pow(C1,a)\*pow(C2,b) = M mod N

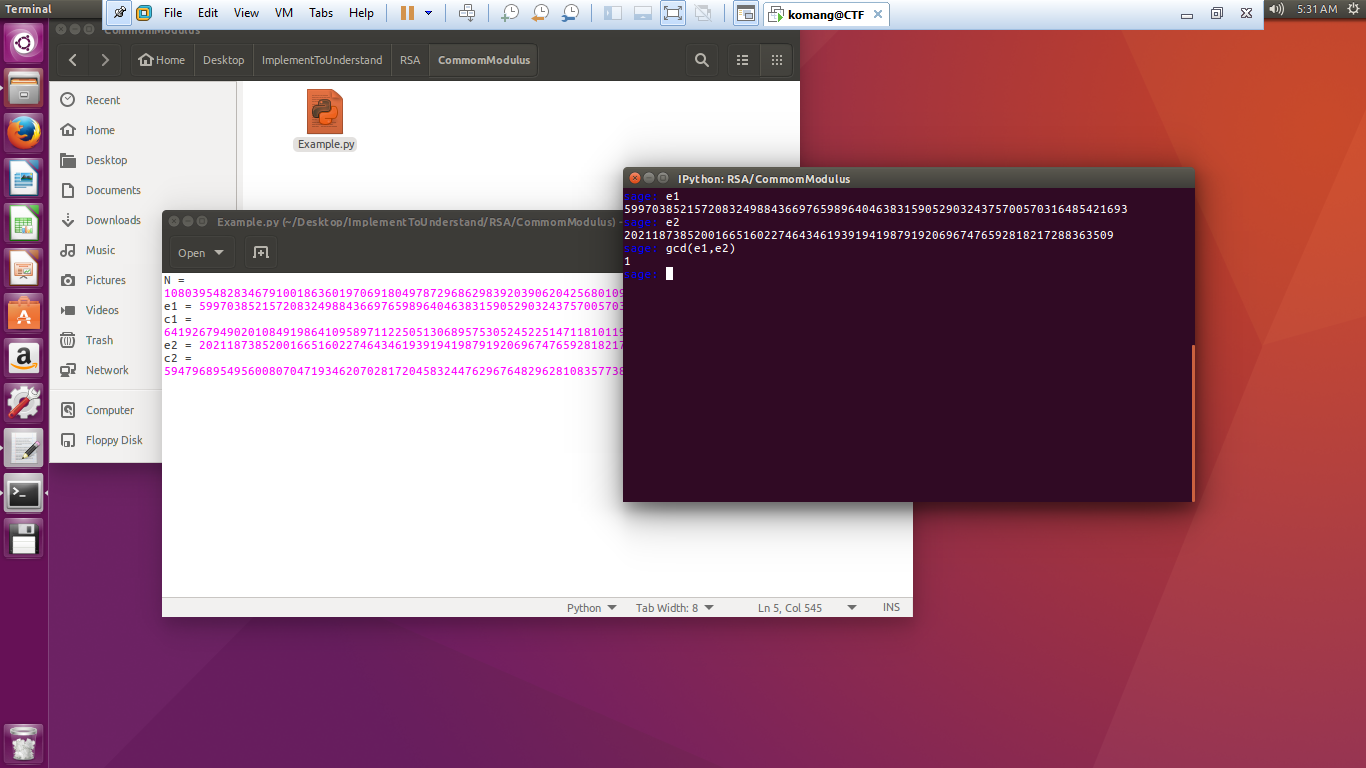
Vì với mã hoá RSA M không được lớn hơn N -> ta có thể bỏ mod N

Vậy M = pow(C1,a)\*pow(C2,b)

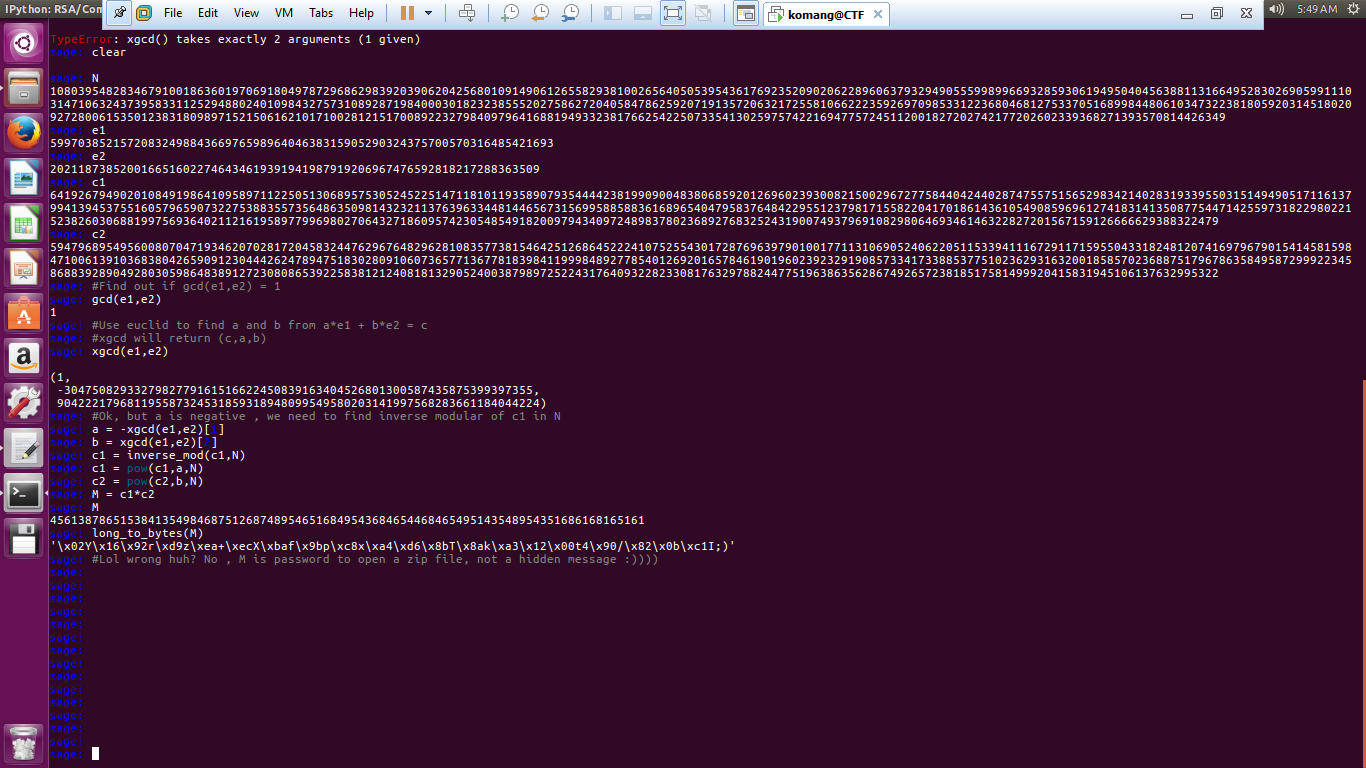
Bài tập minh họa:

https://gist.github.com/anonymous/8f606574e9480fe2ac912d6c87273b2b

Trước tiên ta thử tìm gcd(e1,e2)



Ok vậy sẽ tồn tại 2 số a và b sao cho a\*e1 + b\*e2 = 1, và như ta đã chứng minh, hoàn toàn có thể khôi phục được M



Ngoài ra còn có Implement Attack bao gồm 3 kiểu

* Random fault
* Timing Attack
* Bleichenbacher's Attack on PKCS 1

Đây là các dạng khó :D các bạn có gì google tham khảo thêm :D

**Sử dụng openssl để xử lý các bài cho pubkey.PEM**

Các bạn tham khảo tài liệu tại đây : <https://rietta.com/blog/2012/01/27/openssl-generating-rsa-key-from-command/>

Lưu ý nếu đề cho các bạn Privkey.PEM thì quá easy, cho luôn N,p,q,d,e rồi đấy, extract với lệnh

openssl rsa -in privatekey.pem -noout -text

Còn với pubkey thì

openssl rsa -inform PEM -pubin -in pubkey.pem -noout -text

Để mã hóa 1 file

openssl rsautl -encrypt -pubin -inkey pubkey.pem -in File2Encrypt -out FileEncrypted

Để giải mã 1 file

Openssl rsautl -decrypt -pubin -inkey privatekey.pem

openssl rsautl -decrypt -inkey privatekey.pem -in File2Decrypt -out FileDecrypted

Theo kinh nghiệm mình làm thì có 1 vài bài mình tìm được p,q,d mà decrypt vẫn không ra khi gặp dạng cho pubkey privkey, thì do nó sử dụng padding default là PKCS#1 nên khi decrypt dẫn đến lỗi, phải thực hiện trên dòng lệnh command openssl