CTF-RSA1（已知p、q、dp、dq、c）

原创

RyanWang0000 于 2019-10-24 15:35:34 发布 5822 收藏 32

分类专栏： CTF-crypt 文章标签： CTF crypto

本文链接：https://blog.csdn.net/qq_32350719/article/details/102719279

题目来源：buuCTF-crypto-RSA1

题目：

```
p = 8637633767257008567099653486541091171320491509433615447539162437911244175885667806398411790524083553445158113502227745206205327690939504032994699902053229
q = 1264067497399647276917604793717088342092705082148001058159313713537247388059561373733763062975257734614703928403008259349077663057258495995420533688022846 9
dp = 6500795702216834621109042351193261530650043841056252930930949663358625016881832840728066026150264693076109354874099841380454881716097778307268116910582929
dq = 7834722636735534490195325803864706723805740335513038891379117604388816836745560980982567956735122019630021754387627675169680435995825275391608111205500 41
c = 2472230540388738207356731646764908066263155290596022939907910799560215441817605633580063888752761416407353043765708507967615735020535194522298935131607648657359957604197833987226592506276431853608900731027027852615967893743190386289240074791552511898395997060793414297473667578432599344594203137210734210385 2
```

很明显，给了密文c和一些参数，解密求m

一切以解题为目的的抄代码都是耍流氓，我们还是要从数学理论上去解决它，最后再根据数学理论来写代码。

公式推导：

先摆出已知条件：

$c \equiv m \ mod n$

目的很明确，要想得到m，就要得到c

利用中国剩余定理,我们可以得到

$m1 \equiv c \ mod p, m2 \equiv c \ mod q$

这里肯定有很多人不理解，简单证明一下

由$m \equiv c \ mod n$ 可以得到式子

上述式子，同时取余q和p，可以分别得到

$m1 \equiv c \ mod p, m2 \equiv c \ mod q$

带入上面的公式，可以得到$c = kp + m1$

我们把这个带入m2可以得到

$m2 \equiv (kp + m1) mod q$

等式两边同时减去m1,可以得到

$(m2 - m1) \equiv kp mod q$

这里因为$gcd(p,q) = 1$

所以可以求p的逆元，得到$(m2 - m1) * p \equiv k mod q$

$k \equiv (m2 - m1) * p \mod q$

我们上下两个式子合并，得到

$c =$ $p + m1$，

得到$m =$

$d \equiv d \mod (p-1), d \equiv d \mod (q-1)$

分别带入m1,m2，有

$m1 \equiv c \mod q$

这里肯定有人又不理解为什么可以直接带入了，我们再证明一下，这里用到了费马小定理即假如p是质数，且$gcd(k,p) = 1$，则

所以如果我们有等式

d=dp+k□(p−1)

我们直接带入，有

$m2 \equiv c \mod p$

这里的指数，我们拆开，为

$m2 \equiv c * c \mod p$

这里的

$c \equiv 1 mod p$（用了刚才说的费马小定理）

那么m1根据对称性也可以同理得到

$m1 \equiv c \mod q$

最终，我们拥有了如下的条件：
m1≡c^{dq }mod q
m2≡c^{dp} mod p
m≡(((m2−m1)□p−1 mod q)p+m1) mod n

一切就绪，等号右边的全部都是已知的，开算
上代码

```
import libnum
def egcd(a, b):
    if (b == 0):
        return 1, 0, a
    else:
        x, y, q = egcd(b, a % b)  # q = GCD(a, b) = GCD(b, a%b)
        x, y = y, (x - (a // b) * y)
        return x, y, q


def mod_inv(a, b):
    return egcd(a, b)[0] % b  # 求a模b得逆

p = 86376337672570085670996534865410911713204915094336154475391624379112441758856678063984117905240835534451581135022277452062053276909395040329946999020532 29
q = 12640674973996472769176047937170883420927050821480010581593137135372473880595613737337630629752577346147039284030082593490776630572584959954205336880228469
dp = 65007957022168346211090423511932615306500438410562529309309496633586250168818328407280660261502646930761093548740998413804548817160977783072681169105829 29
dq = 78347226367355344901953258038647067238057403355130388913791176043888168367455609809825679567351220196300217543876276751696804359958252753916081112055004 1
c = 24722305403887382073567316467649080662631552905960229399079107995602154418176056335800638887527614164073530437657085079676157350205351945222989351316076486573599576041978339872265925062764318536089007310270278526159678937431903862892400747915525118983959970607934142974736675784325993445942031372107342103852

invq=mod_inv(p,q)
mp=pow(c,dp,p)
mq=pow(c,dq,q)
m=((mp-mq)*invq%p)*q+mq
print(libnum.n2s(m))
```

最终得到答案

```
noxCTF{W31c0m3_70_Ch1n470wn}
```