山东大学网络空间安全学院

Python高级程序设计课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202100460065 | 姓名：李昕 | | 班级：21级密码2班 |
| 实验题目：实验13-14. 综合 | | | |
| 实验学时：2 | | 实验日期：2022/12/15 | |
| 实验目的：根据所给条件，编写代码，破解RSA。（1）借助sympy实现；（2）除了我们们介绍的比如numpy、密码模块之外，仅借助gmpy2模块完成。（3）尝试不要借助这些比较完善的第三方模块，完成破解，报告尝试过程以及尝试结果。 | | | |
| 硬件环境：  AMD Ryzen 7 5800H with Radeon Graphics     3.20 GHz  机带 RAM 16.0 GB (13.9 GB 可用 | | | |
| 软件环境：  系统：Windows11  编译器：IDLE | | | |
| 实验步骤与内容：  本次实验包括一个内容：  1、已知有人写了如下的代码，并将生成的(n,e,c)以及(n2,e2,c2,(p2+1)\*(q2+1))输出。   1. from Crypto.Util.number import \* 2. def ef(): 3. p=getPrime(512) 4. q=getPrime(512) 5. flag=open("flag","rb").read() 6. m=bytes\_to\_long(flag) 7. e=65537 8. n=p\*q 9. c=pow(m,e,n) 10. print(n,e,c)  *#* 11. return p,q 12. def epq(p,q): 13. p2=getPrime(1024) 14. q2=getPrime(1024) 15. e2=65537 16. m2=p+q 17. n2=p2\*q2 18. c2=pow(m2,e2,n2) 19. print(n2,e2,c2,(p2+1)\*(q2+1)) *#* 20. p,q=ef() 21. epq(p,q)   输出(n,e,c)以及(n2,e2,c2,(p2+1)\*(q2+1))已知；  请编写代码，求出flag。  注意：根据所给条件，编写代码，破解RSA。（1）借助sympy实现；（2）除了我们们介绍的比如numpy、密码模块之外，仅借助gmpy2模块完成。（3）尝试不要借助这些比较完善的第三方模块，完成破解，报告尝试过程以及尝试结果。  【Crypto根据所给数据破解RSA】  1、求解RSA的过程：假设明文为M，密文为C，选择两个互异并且距离较远的大素数p和q，计算n=p×q和φ(n)=(p-1)×(q-1)，选择与φ(n)互素的密钥e，加密过程为C≡M^e mod n，解密过程为先求出模逆：ed≡1 modφ(n),然后求解C^d mod n 求出的结果即为明文M。  2、RSA为非对称密码，其破解难度基于难以在短时间内将一个大整数因式分解，是一个困难问题。此题中设有两次的RSA加密。  3、第一次加密是用公钥e=65537对明文flag加密，求出φ(n)=(p-1)×(q-1)是很困难的，暴力破解1000多位的RSA花费时间不可估计，所以题目给了一些提示，这便是第二次加密的原因。  第二次加密是用公钥e=65537对第一次加密中的私钥p和q之和加密，加密的原理还是基本的RSA，不同的是虽然第二次加密用的新随机生成的私钥p2和q2，但是题目还额外给出了(p2+1)×(q2+1)这个信息。  4、解题的方法就是向前递推，先破解第二层RSA加密，得到第一层加密中使用的私钥对之和p+q，再继续解第一层加密，最后得到结果flag。  【借助sympy代码实现】  这里借助了sympy模块包含的几个数学函数，如gcd等，加速了代码的编写。   1. from sympy import \* 2. from Crypto.Util.number import \* 3. def exgcd(a, m): 4. if gcd(a, m) != 1: 5. return None 6. u1,u2,u3 = 1, 0, a 7. v1,v2,v3 = 0, 1, m 8. while v3 != 0: 9. q = u3 // v3*#地板除* 10. v1, v2, v3, u1, u2, u3 = (u1 - q \* v1), (u2 - q \* v2), (u3 - q \* v3), v1, v2, v3 11. return u1 % m 12. A=(86902041785171683231305301154659574124170175717542478743469937094129588097899526472297338165763793395403138052107436307148655516237914989260641272558370267587581870375426597336633068656306063961150434589742092844368242428011721324737315708847899038491875643600715933822290329122093126146305056053635875427149, 65537, 15833685544860151615528563318175057334854555807563811248911092184962045410721043935895279999715546874344448072757248024629816018687138160349497178202841685747385455984597261139196670787723191129190281464722284764818569442853999503305719960273532581788705319675785365330814771611391805600982635624166616994909) 13. B=(18591274039015146371355461783271676729436253989312203484465267365829253730222536917260636899143825565341750487371058728119621305795504534209590790135844600724302582139316981344931079461283380928221091934542959757790517904211220170283310587231962911318192160954877212603169707693638590690557518269614342828433745751900636977114625384201443358493405946175157229330447527380345317933922535718700042422977893867943656666930789561406630888405895439054817890293649823406983304364531032464229623788486039136673884818993067818472490283937605836580590843487927803181532325018537076264511576862025094690559279649143498659102849, 14. 65537, 15. 7392251042794448813724942265095985628606500546061544026052413736090062985580390082556940969813307099293888028876436850564855740297176392396726998769680589219160575972891500463789856128741737806143813567755324525667951325489807002065570746854557547784519649141813876707420713784143969266692484619922991587339418980913573323306950701125825931923149350170489304772562424051072424741158162269063304635895188390600892069271483630208193319815348033784877031408074730563809389105637498825225265591866481815230255385046830196748192233124781928655809322769875744414705189416861877961926362356094198048709542228727393696757158, 16. 18591274039015146371355461783271676729436253989312203484465267365829253730222536917260636899143825565341750487371058728119621305795504534209590790135844600724302582139316981344931079461283380928221091934542959757790517904211220170283310587231962911318192160954877212603169707693638590690557518269614342828434020648157035533967661393926860975983001244775722594600595802713222548261077015534360904269663877978505992018966625264663179236777522008665977063509158288897299644184441313228368572324485490486359818154230942787018240908451883098571389138398322839231059197174670650732257722735400365023996558029601185819826736) 17. phi2=B[0]\*2-B[3]+2 18. *#使用扩展欧几里得求出inv2* 19. inv2=exgcd(B[1],phi2) 20. *#第二层解密* 21. m2=pow(B[2],inv2,B[0]) 22. print('解密解得m2=',m2) 23. phi=A[0]-m2+1 24. *#使用扩展欧几里得求出inv* 25. inv=exgcd(A[1],phi) 26. *#第一层解密* 27. m=pow(A[2],inv,A[0]) 28. print('解密解得m=',m) 29. *#转码得到flag* 30. flag=long\_to\_bytes(m) 31. print('解密解得flag=',flag)   【仅借助gmpy2模块代码实现】  在此部分中使用了gmpy2模块的内置拓展欧几里得算法求乘法逆元函数gmpy2.invert（a，c）#对a，求b，使得a\*b=1（mod c）   1. from gmpy2 import \* 2. from Crypto.Util.number import \* 3. A=(86902041785171683231305301154659574124170175717542478743469937094129588097899526472297338165763793395403138052107436307148655516237914989260641272558370267587581870375426597336633068656306063961150434589742092844368242428011721324737315708847899038491875643600715933822290329122093126146305056053635875427149, 65537, 15833685544860151615528563318175057334854555807563811248911092184962045410721043935895279999715546874344448072757248024629816018687138160349497178202841685747385455984597261139196670787723191129190281464722284764818569442853999503305719960273532581788705319675785365330814771611391805600982635624166616994909) 5. B=(18591274039015146371355461783271676729436253989312203484465267365829253730222536917260636899143825565341750487371058728119621305795504534209590790135844600724302582139316981344931079461283380928221091934542959757790517904211220170283310587231962911318192160954877212603169707693638590690557518269614342828433745751900636977114625384201443358493405946175157229330447527380345317933922535718700042422977893867943656666930789561406630888405895439054817890293649823406983304364531032464229623788486039136673884818993067818472490283937605836580590843487927803181532325018537076264511576862025094690559279649143498659102849, 6. 65537, 7. 7392251042794448813724942265095985628606500546061544026052413736090062985580390082556940969813307099293888028876436850564855740297176392396726998769680589219160575972891500463789856128741737806143813567755324525667951325489807002065570746854557547784519649141813876707420713784143969266692484619922991587339418980913573323306950701125825931923149350170489304772562424051072424741158162269063304635895188390600892069271483630208193319815348033784877031408074730563809389105637498825225265591866481815230255385046830196748192233124781928655809322769875744414705189416861877961926362356094198048709542228727393696757158, 8. 18591274039015146371355461783271676729436253989312203484465267365829253730222536917260636899143825565341750487371058728119621305795504534209590790135844600724302582139316981344931079461283380928221091934542959757790517904211220170283310587231962911318192160954877212603169707693638590690557518269614342828434020648157035533967661393926860975983001244775722594600595802713222548261077015534360904269663877978505992018966625264663179236777522008665977063509158288897299644184441313228368572324485490486359818154230942787018240908451883098571389138398322839231059197174670650732257722735400365023996558029601185819826736) 10. *#求得φ(n2)=(p2-1)×(q2-1)=p2×q2-p2-q2+1=2×p2×q2-(p2×q2+p2+q2+1)+2* 11. phi2=B[0]\*2-B[3]+2 12. *#使用扩展欧几里得求出inv2* 13. inv2=gmpy2.invert(B[1],phi2) 14. *#第二层解密* 15. m2=pow(B[2],inv2,B[0]) 16. print('解密解得m2=',m2) 17. *#求得φ(n)=(p-1)×(q-1)=p×q-p-q+1=p×q-(p+q)+1* 18. phi=A[0]-m2+1 19. *#使用扩展欧几里得求出inv* 20. inv=gmpy2.invert(A[1],phi) 21. *#第一层解密* 22. m=pow(A[2],inv,A[0]) 23. print('解密解得m=',m) 24. *#转码得到flag* 25. flag=long\_to\_bytes(m) 26. print('解密解得flag=',flag)   【尝试不要借助这些比较完善的第三方模块，完成破解】  尝试不使用完善的模块，自行编写欧几里得算法函数和拓展欧几里得算法函数：   1. from Crypto.Util.number import \* 2. *#欧几里得求最大公因子* 3. def gcd(a, b): 4. while a != 0: 5. a, b = b % a, a 6. return b 7. *#扩展欧几里得求模逆* 8. def exgcd(a, m): 9. if gcd(a, m) != 1: 10. return None 11. u1,u2,u3 = 1, 0, a 12. v1,v2,v3 = 0, 1, m 13. while v3 != 0: 14. q = u3 // v3 15. v1, v2, v3, u1, u2, u3 = (u1 - q \* v1), (u2 - q \* v2), (u3 - q \* v3), v1, v2, v3 16. return u1 % m 17. A=(86902041785171683231305301154659574124170175717542478743469937094129588097899526472297338165763793395403138052107436307148655516237914989260641272558370267587581870375426597336633068656306063961150434589742092844368242428011721324737315708847899038491875643600715933822290329122093126146305056053635875427149, 65537, 15833685544860151615528563318175057334854555807563811248911092184962045410721043935895279999715546874344448072757248024629816018687138160349497178202841685747385455984597261139196670787723191129190281464722284764818569442853999503305719960273532581788705319675785365330814771611391805600982635624166616994909) 19. B=(18591274039015146371355461783271676729436253989312203484465267365829253730222536917260636899143825565341750487371058728119621305795504534209590790135844600724302582139316981344931079461283380928221091934542959757790517904211220170283310587231962911318192160954877212603169707693638590690557518269614342828433745751900636977114625384201443358493405946175157229330447527380345317933922535718700042422977893867943656666930789561406630888405895439054817890293649823406983304364531032464229623788486039136673884818993067818472490283937605836580590843487927803181532325018537076264511576862025094690559279649143498659102849, 20. 65537, 21. 7392251042794448813724942265095985628606500546061544026052413736090062985580390082556940969813307099293888028876436850564855740297176392396726998769680589219160575972891500463789856128741737806143813567755324525667951325489807002065570746854557547784519649141813876707420713784143969266692484619922991587339418980913573323306950701125825931923149350170489304772562424051072424741158162269063304635895188390600892069271483630208193319815348033784877031408074730563809389105637498825225265591866481815230255385046830196748192233124781928655809322769875744414705189416861877961926362356094198048709542228727393696757158, 22. 18591274039015146371355461783271676729436253989312203484465267365829253730222536917260636899143825565341750487371058728119621305795504534209590790135844600724302582139316981344931079461283380928221091934542959757790517904211220170283310587231962911318192160954877212603169707693638590690557518269614342828434020648157035533967661393926860975983001244775722594600595802713222548261077015534360904269663877978505992018966625264663179236777522008665977063509158288897299644184441313228368572324485490486359818154230942787018240908451883098571389138398322839231059197174670650732257722735400365023996558029601185819826736) 23. phi2=B[0]\*2-B[3]+2 24. inv2=exgcd(B[1],phi2) 25. m2=pow(B[2],inv2,B[0]) 26. print('解密解得m2=',m2) 27. phi=A[0]-m2+1 28. inv=exgcd(A[1],phi) 29. m=pow(A[2],inv,A[0]) 30. print('解密解得m=',m) 31. flag=long\_to\_bytes(m) 32. print('解密解得flag=',flag)   执行结果： | | | |
| 结论分析与体会：  通过本次练习，我熟悉了解了Python的密码学编程知识，非对称密码RSA的加密和解密过程。同时，我还复习了其他的一些python扩展库的函数，对模块的使用有了进一步的了解。 | | | |