|  |
| --- |
| **Github（或者Coding）账号：** https://github.com/Wkingxc/cryptography-rsa-homework |
| **个人博客关于密码学大作业的链接：https://wkingxc.com/2023/01/15/RSA%E5%A4%A7%E7%A4%BC%E5%8C%85/** |
| **题目（中文）：RSA大礼包** |
| **摘要（中文）：**  RSA密码算法是使用最为广泛的公钥密码体制。该体制简单且易于实现，只需要选择5个参数即可（两个素数𝑝和𝑞、模数𝑁=𝑝𝑞 、加密指数𝑒和解密指数𝑑。设𝑚为待加密消息RSA体制破译相当于已知𝑚^𝑒 mod 𝑁能否还原𝑚的数论问题。目前模数规模为1024比特的RSA算法一般情况下是安全的，但是如果参数选取不当，同样存在被破译的可能。  有人制作了一个RSA加解密软件采用的RSA体制的参数特点描述见密码背景部分）。已知该软件发送某个明文的所有参数和加密过程的全部数据（加密案例文件详见附件3-1。Alice使用该软件发送了一个通关密语，且所有加密数据已经被截获，请问能否仅从加密数据恢复该通关密语及RSA体制参数？如能请给出原文和参数，如不能请给出已恢复部分并说明剩余部分不能恢复的理由。 |
| **题目描述（清楚描述题目中文，写出自己的理解，请勿复制原题目）**  **共有20个截获的RSA加密的块，运用所学知识，查询资料，运用常见的对RSA加密的攻击方式来对这些Frame进行攻击，尝试复原出加密的信息。** |
| **过程（包括背景，原理：必要的公式，图表；步骤，如有必要画出流程图，给出主要实现步骤代码）**   1. **密文结构解析&分析**   1.遍历所有的模数N，判断是否存在模数相同的加密片段，如果猜测可以用共模攻击  2.遍历寻找任意两个模数N的公因子，如果得到不为1的公因子则可以因数碰撞  3.遍历所有加密指数e，寻找低加密指数及对应的加密对，可以用低指数广播攻击  e[0]=46786465362686334917265996843779843233606992585424976481745055335468678697948774988450305612127967926533923268260412557000125153569622340353246096040604284883505587337829322949633637609180797447754513992039018904786537115087888005528547900640339270052628915440787357271345416818313808448127098885767015748889 e[1]=e[2]=e[5]=e[6]=e[9]=e[10]=e[13]=e[14]=e[17]=e[18]=e[19]=65537 e[2]=65537 e[3]=e[8]=e[12]=e[16]=e[20]=5 e[4]=152206992575706893484835984472544529509325440944131662631741403414037956695665533186650071476146389737020554215956181827422540843366433981607643940546405002217220286072880967331118344806315756304650248634546597784597963886656422706197757265316981889118026978865295597135470735576032282694348773714479076093197 e[7]=e[11]=e[15]=3  对上述的公钥e进行分析,主要有以下几种，其中e比较小的有3和5，其中  Frame7，Frame11，Frame15采用低加密指数e=3进行加密  Frame3，Frame8，Frame12，Frame16和Frame20均采用低加密指数e=5进行加密  采用费马分解尝试p,q差距比较小的帧和Pollard p-1分解进行尝试p,q差距比较大的帧  Frame10可以用费马分解法破解  Frame2，6，19可以用p-1分解法破解   1. **共模攻击**   对上述的公钥e进行分析,主要有以下几种，其中e比较小的有3和5，其中  Frame7，Frame11，Frame15采用低加密指数e=3进行加密  Frame3，Frame8，Frame12，Frame16和Frame20均采用低加密指数e=5进行加密   1. 采用费马分解尝试p,q差距比较小的帧和Pollard p-1分解进行尝试p,q差距比较大的帧 2. **共模攻击**     **RESUILT:{'Frame0': b'My secre', 'Frame4': b'My secre'}**   1. **公因数碰撞攻击**     #公因数碰撞攻击 def same\_factor\_attack():        p\_of\_prime=gmpy2.gcd(n[1],n[18])        q1=n[1]//p\_of\_prime        q18=n[18]//p\_of\_prime        phi1=(p\_of\_prime-1)\*(q1-1)#计算欧拉函数        phi18=(p\_of\_prime-1)\*(q18-1)#计算欧拉函数        d1=gmpy2.invert(e[1],phi1)#计算私钥        d18=gmpy2.invert(e[18],phi18)#计算私钥        m1=pow(c[1],d1,n[1])#解密        m18=pow(c[18],d18,n[18])#解密        m1=hex(m1)[2:]#去掉0x        m18=hex(m18)[2:]#去掉0x        m1=binascii.unhexlify(m1)[-8:]#hex->str        m18=binascii.unhexlify(m18)[-8:]#hex->str        sloved['Frame1']=m1        sloved['Frame18']=m18  'Frame1': b'. Imagin', 'Frame18': b'm A to B'   1. **低指数广播攻击**     #中国剩余定理 def chinese\_remainder\_theorem(backup):        #计算N的乘积        N=1        for a,n in backup:                N\*=n        #计算Ni        Ni=[]        for a,n in backup:                Ni.append(N//n)        #计算Ni的模逆元        Ni\_inverse=[]        for i in range(0,len(Ni)):                Ni\_inverse.append(gmpy2.invert(Ni[i],backup[i][1]))        #计算x        x=0        for i in range(0,len(Ni)):                x+=backup[i][0]\*Ni[i]\*Ni\_inverse[i]        x=x%N        return x,N ​ #低指数3 def low\_exponent\_attack3():        frame\_range=[7,11,15]        backup=[]        for i in frame\_range:                backup.append([c[i],n[i]])        x,N=chinese\_remainder\_theorem(backup)        #开三次方根        m=gmpy2.iroot(x,3)[0]        m=hex(m)[2:]#去掉0x        m=binascii.unhexlify(m)[-8:]#hex->str        sloved['Frame7']=m        sloved['Frame11']=m        sloved['Frame15']=m #低指数5 def low\_exponent\_attack5():        frame\_range=[3,8,12,16,20]        backup=[]        for i in frame\_range:                backup.append([c[i],n[i]])        x,N=chinese\_remainder\_theorem(backup)        #开五次方根        m=gmpy2.iroot(x,5)[0]        m=hex(m)[2:]#去掉0x        m=binascii.unhexlify(m)[-8:]#hex->str        sloved['Frame3']=m        sloved['Frame8']=m        sloved['Frame12']=m        sloved['Frame16']=m        sloved['Frame20']=m  'Frame7': b'\xb8\xbc\xa2S)s\xcd\xd2', 'Frame11': b'\xb8\xbc\xa2S)s\xcd\xd2', 'Frame15': b'\xb8\xbc\xa2S)s\xcd\xd2', 'Frame3': b't is a f', 'Frame8': b't is a f', 'Frame12': b't is a f', 'Frame16': b't is a f', 'Frame20': b't is a f'  可以看到Frame7，Frame11，Frame15采用低加密指数广播攻击出来是乱码，说明该种方法不正确  Frame3、Frame8、Frame12、Frame16、Frame20可以采用该种攻击方法   1. **Pollard p-1分解法**     #Pollard's p-1算法 def pollard\_p\_1(n):        b=2\*\*20        a=2        # while True:        #         a=gmpy2.powmod(a,b,n)        #         d=gmpy2.gcd(a-1,n)        #         if d!=1 and d!=n:        #                 return d        #         a+=1        for i in range(2,b+1):                a=gmpy2.powmod(a,i,n)                d=gmpy2.gcd(a-1,n)                if d!=1 and d!=n:                        q=n//d                        n=q\*d        return d def pollard\_data(n):        frame\_range=[2,6,19]        for i in frame\_range:                temp\_n=n[i]                temp\_c=c[i]                temp\_e=e[i]                p=pollard\_p\_1(temp\_n)                q=temp\_n//p                phi=(p-1)\*(q-1)                d=gmpy2.invert(temp\_e,phi)                m=pow(temp\_c,d,temp\_n)                m=hex(m)[2:]#去掉0x                m=binascii.unhexlify(m)[-8:]#hex->str                sloved['Frame'+str(i)]=m  'Frame2': b' That is', 'Frame6': b' "Logic ', 'Frame19': b'instein.'   1. **费马分解法**     # 费马分解 def fermat\_factorization(n):        a=gmpy2.iroot(n,2)[0]+1        max=200000        for i in range(0,max):                b2=a\*a-n                b=gmpy2.iroot(b2,2)[0]                if gmpy2.is\_square(b2):                        p=a-b                        q=a+b                        return p,q                a+=1 def fermat\_data():        frame\_range=[10]        for i in frame\_range:                p,q=fermat\_factorization(n[i])                phi=(p-1)\*(q-1)#计算欧拉函数                d=gmpy2.invert(e[i],phi)#计算私钥                m=pow(c[i],d,n[i])#解密                m=hex(m)[2:]#去掉0x                m=binascii.unhexlify(m)[-8:]#hex->str                sloved['Frame'+str(i)]=m ​  经检验,Frame10可以采用该种攻击方法  'Frame10': b'will get'   1. **实验结果**   最后根据解出来的零碎信息，结合搜索引擎得到明文：  My secret is a famous saying of Albert Einstein. That is "Logic will get you from A to B. Imagination will take you everywhere.“ |
| **总结（完成心得与其它，主要自己碰到的问题和解决问题的方法）**  基本了解过RSA攻击方面的知识，但对代码编写十分陌生。整个实验过程都是参考互联网上大佬的博客和同学们的指导。  解密的过程是有趣的，了解其背后的原理是困难的，从中能够感受到密码学大牛们的强大。在此实验中，我对python语言的使用更加熟练了，同时我对RSA密码学的安全性有了基本的认识，并理解如何保护RSA加密算法免受攻击。同时，也可以发现加密算法并不是绝对安全的，随着技术的发展和攻击方式的变化，加密算法需要不断地进行更新和改进，以保障数据的安全。 |
| **参考文献（包括参考的书籍，论文，URL等，很重要）**  [2016 全国高校密码数学挑战赛-赛题三]  https://www.tr0y.wang/2017/10/31/RSA2016/  [现代密码学大作业]  https://blog.csdn.net/m0\_63571390/article/details/122375466?spm=1001.2014.3001.5501  [关于RSA的几种攻击手段]  https://blog.csdn.net/pigeon\_1/article/details/114371456  [Pollard's rho算法]  https://blog.csdn.net/qq\_39972971/article/details/82346390  [分解因子算法——Pollard rho算法]  https://blog.csdn.net/weixin\_46395886/article/details/115073059 |