1. RSA简述

RSA算法是一种非对称密码算法，所谓非对称，就是指该算法需要一对密钥，使用其中一个加密，则需要用另一个才能解密。

RSA的算法涉及三个参数，n、e1、e2。

其中，n是两个大质数p、q的积，n的表示时所占用的位数，就是所谓的密钥长度。　　e1和e2是一对相关的值，e1可以任意取，但要求e1与(p-1）\*(q-1）互质；再选择e2，要求（e2\*e1）mod((p-1）\*(q-1））=1。　　（n，e1）,(n，e2）就是密钥对。其中 (n，e1）为公钥，(n，e2）为私钥。

RSA加解密的算法完全相同，设m为明文，c为密文，则：m=c^e1 mod n；c=m^e2 mod n；

e1和e2可以互换使用，即：

A=B^e2 mod n；B=A^e1 mod n。

1. 主要攻击

2.1参数的提取

2.1.1 pem文件

针对此类文件可以直接使用openssl提取，大概使用过的方式有：

openssl rsautl -decrypt -in flag.enc -inkey public.pem -out flag.dec

openssl rsa -pubin -text -modulus -in public.pem

2.1.2 pcap文件

针对此类文件可以使用wireshark follow一下。这种问题一般都是写了一个交互的crypto系统，所以可能产生多轮交互。

2.1.3 ppc模式

这种模式是上述pcap文件的交互版，会给一个端口进行一些crypto的交互，参数会在交互中给出。

2.2模数n的分解

见工具

2.3选择密文攻击

2.3.1概念

选择密文攻击是指密码分析者并不知道解密的密钥，但是给出任意的消息，密码分析者都可以将其加密，再解密。或者说，密码分析者能获得解密服务。

2.3.2 原理

设攻击者为A，密文接受者为T，公钥对为(e, n)，私钥为d，T收到的密文为c，c对应的明文为m。

现在A想知道m = c^d mod n，但是他不想分解n。于是T找了一个随机数r，r < n。他进行如下计算：

x = r^e mod n （对r用T的公钥加密，得到临时密文x）

y = (x \* c) mod n （将临时密文x与密文c相乘）

t = r^(-1) mod n

A利用了RSA加密和解密过程的特点，即：

如果x = r^e mod n，那么 r = x^d mod n

现在A要做的是使T用d对t签名：u = t^d mod n。A需要获得u，然后计算

m = (t \* u) mod n

计算结果是这样推导的：

t \*u mod n = [r^(-1) \* y^d] mod n

= [r^(-1) \* x^d \* c^d] mod n

= c^d mod n

= m

2.4低加密指数攻击

2.4.1Coppersmith定理

Coppersmith定理指出在一个e阶的mod n多项式f(x)中，如果有一个根小于n1e'>n1en1e，就可以运用一个O(log n)的算法求出这些根。

这个定理可以应用于RSA算法。如果e = 3并且在明文当中只有三分之二的比特是已知的，这种算法可以求出明文中所有的比特。

2.4.2广播攻击

如果一个实体使用相同的低加密指数给一个接收者的群发送相同的信息，即，选取了相同的加密指数e，就会发动广播攻击(broadcast attack)。

题目链接：http://codezen.fr/2014/01/16/hackyou-2014-crypto-400-cryptonet

2.4.3相关信息攻击

相关信息攻击(related message attack)是由Franklin Reiter提出来的，下面我们就简单描述一下这种攻击。爱丽丝用e = 3加密两个明文P1和P2，然后再把C1和C2发送给鲍勃。如果通过一个线性函数把P1和P2联系起来，那么伊夫就可以在一个可行的计算时间内恢复P1和P2。

2.4.4短填充攻击

短填充攻击(short pad attack)是由Coppersmith提出来的，下面我们就简单描述一下这种攻击。爱丽丝有一条信息M要发送给鲍勃。她先用r1对信息填充，加密的结果是得到了C1，并把C1发送给鲍勃。伊夫拦截C1并把它丢掉。鲍勃通知爱丽丝他还没有收到信息，所以爱丽丝就再次使用r2对信息填充，加密后发送给鲍勃。伊夫又拦截了这一信息。伊夫现在有C1和C2，并且她知道C1和C2都是属于相同明文的密文。Coppersmith证明如果r1和r2都是短的，伊夫也许就能恢复原信息M。

2.5低解密指数攻击

2.5.1暴露解密指数攻击

很明显，如果伊夫可以求出解密指数d，她就可以对当前加密的信息进行解密。不过，到这里攻击并还没有停止。如果伊夫知道d的值，她就可以运用概率算法(这里不讨论)来对n进行因数分解，并求出p和q值。因此，如果鲍勃只改变了泄露解密指数但是保持模n相同，因为伊夫有n的因数分解，所以她就可以对未来的信息进行解密。这就是说，如果鲍勃发现解密指数已经泄露，他就要有新的p和q的值还要计算出n，并创建所有新的公钥和私钥。

在RSA中，如果d已经泄露，那么p、q、n、e和d就必须要重新生成。

2.5.2低解密指数攻击

鲍勃也许会想到，运用一个小的私钥d就会加快解密的过程。 Wiener表示如果d < 1/3 n1/4，一种基于连分数(一个数论当中的问题)的特殊攻击类型就可以危害RSA的安全。要发生这样的事情，必须要有q < p < 2q。如果这两种情况存在，伊夫就可以在多项式时间中分解n。

* 1. **明文攻击**

2.6.1条件

因为是相同间隔(0~n1)的整数，RSA中的明文和密文是相互置换的。也就是说，伊夫已经知道了有关明文的一些内容。这一特征也许就会引起一些针对明文的攻击。

2.6.2短信息攻击(short message attack)

在短信息攻击中，如果伊夫知道可能的明文组，那么她除了知道明文是密文的转换之外，还知道一些别的信息。伊夫可以对所有可能的信息进行加密，直到结果和所拦截的信息相同。例如，如果知道爱丽丝正在发送一个4位数的数字，伊夫就可以轻易地试验0000~9999的明文数字，来发现明文。为此，短信息必须要在开头和结尾用随机比特进行填充，来阻止这类攻击。强烈建议在用OAEP方法加密之前，要用随机比特填充信息，在本章后面，我们要讨论这一内容。

2.6.3循环攻击(cycling attack)

循环攻击是基于这样一个事实，那就是密文是明文的一个置换，密文的连续加密最终结果就是明文。也就是说，如果对所拦截的密文C连续加密，伊夫最终就得到了明文。不过，伊夫不知道明文究竟是什么，所以她就不知道什么时候要停止加密。她就要往前多走一步。这样如果她再次得到了密文C，她只要返回一步就得到了明文。

对于RSA来说，这种攻击很严重吗？已经表明算法的复杂性和分解n的复杂性是相当的。也就是说，如果n足够大，就不会有一种有效的算法可以在多项式时间内发动这种攻击。

2.6.4公开信息攻击(unconcealed message attack)

另一种基于明文和密文之间置换关系的攻击是公开信息攻击。一则公开信息就是自身加密(不被隐藏)的信息。已经证明总有一些信息是自身加密的。因为通常加密指数是一个奇数，有一些明文如P = 0和P = 1，都是自身加密的。如果加密指数是仔细选出来的，那就会有更多的这种信息被忽略。如果算出来的密文和明文相同，加密程序总要在提交密文之前阻止并拒绝明文。

2.7共模攻击

2.7.1特点

含有若干次加密，每次加密的n都一样。

2.7.2同模攻击(common modulus attack)

如果一个组织使用一个共同的模n，那就有可能发动同模攻击。例如，一个组织中的人也许会让一个可信机构选出p和q，计算出n和 (n)，并为每一个实体创建一对指数(ei, di)。现在假定爱丽丝要发送一则信息给鲍勃。发给鲍勃的密文是C = mod n。鲍勃用他的私密指数dB来对他的信息P = mod n解密。问题是如果伊夫是该组织中的一个成员，并且像我们在"低解密指数攻击"那一部分中学过的那样，她也得到了分配的指数对(eE和dE)，这样她也就可以对信息解密。运用她自己的指数对(eE和dE)，伊夫可以发动一个概率攻击来分解n并得到鲍勃的dB。为了阻止这种类型的攻击，模必须不是共享的。每一个实体都要计算她或他的模。

2.8执行攻击

2.8.1时序攻击(timing attack)

Paul Kocher精密地论证了这种纯密文攻击，我们称为时序攻击。这种攻击基于快速指数算法。如果私密指数d中的相关比特是0的话，这种算法只应用平方；如果相关的位是1，这种算法就既应用平方也应用乘法。也就是说，如果相关的比特是1，完成每一个迭代需要的时序会更长。这种时序上的不同就可以使伊夫逐一找出d中的比特值。

假定伊夫已经拦截了一个密文的一个大数，C1 ~ Cm。我们还假定伊夫已经注意到鲍勃对密文进行解密所用的时间，T1~Tm。由于伊夫知道基本硬件处理乘法运算所用的时间，她就可以计算出t1~tm的值，这里ti就是计算乘法运算Result = Result×Ci mod n所需要的时间。

伊夫可以运用特定算法用来计算d (d0~dk 1)中所有的比特。

算法设置d0 = 1(因为d应该是奇数)，并且计算出Ti的新值(与从d1~dk 1有关的解密时间)。然后该算法假定下一个比特值是1。并求出一些基于假定的D1~Dm的新值。如果这个假定是正确的，每一个Di也许就小于相应的Ti。不过算法运用方差(或别的相关标准)来考虑Di 和 Ti的所有变化。如果在方差上的不同为正，算法就假定下一个比特为1；否则，就假定下一个比特为0。然后算法计算出新的Ti并用作冗余比特。

有两种方法可以阻止时序攻击：

· 把随机延迟加到指数上使每一个指数所消耗的时间相同。

·Rivest引入了盲签名(blinding)。这个概念就是在解密前用一个随机数乘以密文。

2.8.2能量攻击(power attack)

能量攻击和时序攻击相似。Kocher表示，如果伊夫能够准确测量出解密过程中所消耗的能量，她就可以发动一个基于时序攻击原则的能量攻击。涉及乘法和平方的迭代所消耗的能量要比只涉及平方的迭代多。可以用来避免时序攻击的同类技术也可以用来阻止能量攻击。

1. 工具
2. 明文计算工具：大数计算器
3. RSA大数分解网站<http://www.factordb.com/index.php>
4. 大数分解平台https://cloud.sagemath.com/

factor(0x123)

factor(123)

4、RSA tool2

yafu 如果两个素数相差很近，可采用费马分解

factor(0x123)

factor(123)

5、rsatool.py 标准工具 用于已有p,q生成私钥

./rsatool.py -p num1 -q num2 -o priv.key

6、RsaCtfTool.py 提供公钥可输出n,e

7、openssl

输出公钥信息

openssl rsa -noout -text -inform PEM -in public.key -pubin

openssl rsautl -decrypt -in level1bin.passwd.enc -inkey ../priv.key -out level1.passwd -oaep

1. 解题方法及例题

4.1直接分解n

4.1.1介绍：

素数分解问题是困难的，但是可以通过计算机进行暴力分解。1999年，名为Cray的超级计算机用了5个月时间分解了512bit的n。2009年，一群研究人员成功分解了768bit的n。2010年，又提出了一些针对1024bit的n的分解的途径，但是没有正面分解成功。通常意义上来说，一般认为2048bit以上的n是安全的。现在一般的公钥证书都是4096bit的证书。

如果n比较小，那么可以通过工具进行直接n分解，从而得到私钥。如果n的大小小于256bit，那么我们通过本地工具即可爆破成功。例如采用windows平台的RSATool2v17，可以在几分钟内完成256bit的n的分解。

如果n在768bit或者更高，可以尝试使用一些在线的n分解网站，这些网站会存储一些已经分解成功的n，比如：http://factordb.com

通过在此类网站上查询n，如果可以分解或者之前分解成功过，那么可以直接得到p和q。然后利用前述方法求解得到密文。

4.1.2识别：

此类问题一般是分值较小的题目，提取出n之后可以发现n的长度小于等于512bit，可以直接取分解n。如果大于512bit，建议在使用每个题目都用后面所说的方法去解题。

4.1.3例题：

链接：<https://www.jarvisoj.com/challenges> (easy rsa)

4.2利用公约数

4.2.1介绍：

如果在两次公钥的加密过程中使用的n1 和n2具有相同的素因子，那么可以利用欧几里得算法直接将n1和n2分解。

通过欧几里得算法可以直接求出n1和n2的最大公约数p：(n1,n2)=p，可以得出：

n1=pq1

n2=pq2

直接分解成功。而欧几里得算法的时间复杂度为：O(log n)。这个时间复杂度即便是4096 bit也是秒破级别。

def gcd(a, b):

if a < b:

a, b = b, a

while b != 0:

temp = a % b

a = b

b = temp

return a

4.2.2识别

此类题目，通常会发现题目给了若干个n，均不相同，并且都是2048bit，4096bit级别，无法正面硬杠，并且明文都没什么联系，e也一般取65537。

4.2.3例题：

在一个题目中，你拿到了两个n，e都为65537，两个n分别为：

n1=9051013965404084482870087864821455535159008696042953021965631089095795348830954383127323853272528967729311045179605407693592665683311660581204886571146327720288455874927281128121117323579691204792399913106627543274457036172455814805715668293705603675386878220947722186914112990452722174363713630297685159669328951520891938403452797650685849523658191947411429068829734053745180460758604283051344339641429819373112365211739216160420494167071996438506850526168389386850499796102003625404245645796271690310748804327

n2=13225948396179603816062046418717214792668512413625091569997524364243995991961018894150059207824093837420451375240550310050209398964506318518991620142575926623780411532257230701985821629425722030608722035570690474171259238153947095310303522831971664666067542649034461621725656234869005501293423975184701929729170077280251436216167293058560030089006140224375425679571181787206982712477261432579537981278055755344573767076951793312062480275004564657590263719816033564139497109942073701755011873153205366238585665743

通过直接分解，上factordb都分解失败。通过尝试发现：

print gcd(n1,n2)

打印出：

1564859779720039565508870182569324208117555667917997801104862601098933699462849007879184203051278194180664616470669559575370868384820368930104560074538872199213236203822337186927275879139590248731148622362880471439310489228147093224418374555428793546002109

则此致即为两个n共有的素因子p，然后进一步求出q，求解完毕。

4.3Fermat方法与Pollard rho方法

4.3.1介绍：

针对大整数的分解有很多种算法，性能上各有优异，有Fermat方法，Pollard rho方法，试除法，以及椭圆曲线法，连分数法，二次筛选法，数域分析法等等。其中一些方法应用在RSA的攻击上也有奇效。

在p，q的取值差异过大，或者p，q的取值过于相近的时候，Format方法与Pollard rho方法都可以很快将n分解成功。

此类分解方法有一个开源项目yafu将其自动化实现了，不论n的大小，只要p和q存在相差过大或者过近时，都可以通过yafu很快地分解成功。

4.3.2识别：

在直接分解n无望，不能利用公约数分解n之后，都应该使用yafu去试一下。

4.3.3例题：

L链接：https://www.jarvisoj.com (Medium RSA)

4.4 **e=3时的小明文攻击**

4.4.1**介绍：**

当e=3时，如果明文过小，导致明文的三次方仍然小于n，那么通过直接对密文三次开方，即可得到明文。

即：

c≡mec≡me modmod nn

如果e=3，且me<nme<n，那么：

c=me,c=me, e=3e=3

m=3√cm=c3

如果明文的三次方比n大，但是不是足够大，那么设k，有：

c=me+knc=me+kn

爆破k，如果c−knc−kn能开三次根式，那么可以直接得到明文。

**4.4.2识别：**

推荐在e=3的时候首先尝试这种方法。

**4.4.3例题：**

链接：[https://www.jarvisoj.com](https://www.jarvisoj.com/) (Extremely hard RSA)

4.4.4关键代码：

此题通过不断给明文+n开三次方即可求得：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | i=0     while 1:     if(gmpy.root(c+i\*N, 3)[1]==1):       print gmpy.root(c+i\*N, 3)       break     i=i+1 |

4.5低加密指数广播攻击

4.5.1介绍：

如果选取的加密指数较低，并且使用了相同的加密指数给一个接受者的群发送相同的信息，那么可以进行广播攻击得到明文。

即，选取了相同的加密指数e（这里取e=3），对相同的明文m进行了加密并进行了消息的传递，那么有：

c1≡me mod n1

c2≡me mod n2

c3≡me mod n3

对上述等式运用中国剩余定理，在e=3时，可以得到：

cx≡m3 mod n1n2n3

通过对cx进行三次开方可以求得明文。

4.5.2识别：

这个识别起来比较简单，一般来说都是给了三组加密的参数和明密文，其中题目很明确地能告诉你这三组的明文都是一样的，并且e都取了一个较小的数字。

4.5.3例题：

链接：<http://pan.baidu.com/s/1eSwfWyu> 密码：nvs9

题目第二轮中通过流量包的方式给了广播攻击的参数。

国外类似一题的网址：<http://codezen.fr/2014/01/16/hackyou-2014-crypto-400-cryptonet>

4.6低解密指数攻击

4.6.1介绍：

与低加密指数相同，低解密指数可以加快解密的过程，但是者也带来了安全问题。Wiener表示如果满足：

d<1/3gn^1/4

那么一种基于连分数(一个数论当中的问题)的特殊攻击类型就可以危害RSA的安全。此时需要满足：

q<p<2q

如果满足上述条件，通过Wiener Attack可以在多项式时间中分解n。

rsa-wiener-attack的攻击源码开源在了github中，采取python编写，可以很容易使用。

4.6.2识别：

非常简单，e看起来很大就行了。

4.6.3例题：

链接：<http://www.shiyanbar.com/ctf/730>

直接github用工具就行。<https://github.com/pablocelayes/rsa-wiener-attack>

这里注意一个细节问题，如果在运行脚本的时候报错，请在脚本前加上：

import sys

sys.setrecursionlimit(10000000)

4.7**共模攻击**

**4.7.1介绍：**

如果在RSA的使用中使用了相同的模n对相同的明文m进行了加密，那么就可以在不分解n的情况下还原出明文m的值。

即：

c1≡me1c1≡me1 modmod nn

c2≡me2c2≡me2 modmod nn

此时不需要分解n，不需要求解私钥，如果两个加密指数互素，就可以通过共模攻击在两个密文和公钥被嗅探到的情况下还原出明文m的值。

过程如下，首先两个加密指数互质，则：

(e1,e2)=1(e1,e2)=1

即存在s2s2，s2s2使得：

s1e1+s2e2=1s1e1+s2e2=1

又因为：

c1≡me1c1≡me1 modmod nn

c2≡me2c2≡me2 modmod nn

通过代入化简可以得出：

cs11cs22≡mc1s1c2s2≡m modmod nn

明文解出。

**4.7.2识别：**

非常简单，若干次加密，每次n都一样，明文根据题意也一样即可。

**4.7.3例题：**

链接：[https://www.jarvisoj.com](https://www.jarvisoj.com/)(very  hard RSA)

4.7.4关键代码：

如果已知：n1，n2，c1，c2，e1，e2，并且其中n1=n2的话：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | s = egcd(e1, e2)   s1 = s[1]   s2 = s[2]     print s   n=n1     if s1<0:       s1 = - s1       c1 = modinv(c1, n)     elif s2<0:       s2 = - s2       c2 = modinv(c2, n)   m=(pow(c1,s1,n)\*pow(c2,s2,n)) % n |

4.8私钥恢复

4.8.1识别

一般会直接告诉你，或者压缩包里有corrupted文件。

4.8.2

链接：<https://www.jarvisoj.com> (god like rsa）

4.9未分类的题

·链接：<http://www.shiyanbar.com/ctf/1811>

·链接：<http://pan.baidu.com/s/1mhIEa1M> 密码：rt27

·链接：<http://pan.baidu.com/s/1qYTB6O0> 密码：1kzu

·链接：<https://www.jarvisoj.com/challenges> （rsa）

1. 参考资料

<http://bobao.360.cn/learning/detail/3058.html?isappinstalled=0>

<http://book.51cto.com/art/200901/105942.htm>

<http://www.bystudent.com/?p=234>

<http://www.bystudent.com/?p=86>

<https://www.40huo.cn/blog/rsa-private-key-recovery-and-oaep.html>