ctf密码学总结

原创

reset-nnn 于 2018-11-05 17:55:36 发布 13158 收藏 67
分类专栏： CTF 文章标签： ctf密码学 rsa

自己做题总结的小知识点。

1.RSA

公钥标准文件的后缀是 .pem

加密过程 选择两个大素数p和q，计算出模数N = p * q

计算φ = (p−1) * (q−1) 即N的欧拉函数，然后选择一个e (1<e<φ)，且e和φ互质

取e的模反数为d，计算方法: e * d ≡ 1 (mod φ)

对明文m进行加密：c≡m^e (mod n) 或 c = pow(m,e,n)，得到的B即为密文

对密文c进行解密，m≡c^d(mod n) 或 m = pow(c,d,n)，得到的A即为明文

p 和 q ：大整数N的两个因子（factor）

N：大整数N，我们称之为模数（modulus）

e 和 d：互为模反数的两个指数（exponent）

c 和 m：分别是密文和明文，这里一般指的是一个十进制的数

一般有如下称呼：

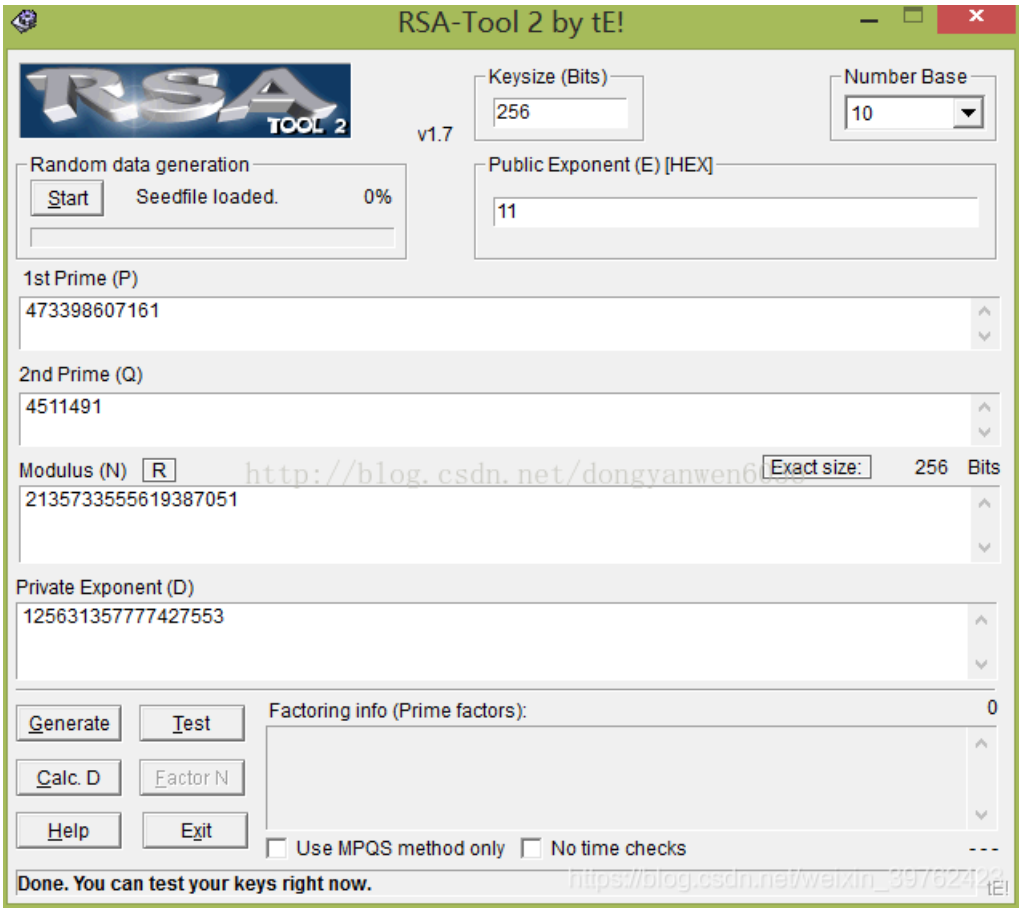
（N，e）：公钥

（N，d）：私钥

分解N可以在线factordb.com，也可以用yafu，也可以用msieve，yafu的使用方法为：

先进入yafu的安装文件夹，然后执行语句 yafu-x64.exe factor(N)

Msieve的使用方法为：msieve153.eve n(如果n是十六进制的，前面要加上0x) -v



最上面一行是固定的

右上方第二行是16进制的，需要用程序员计算器进行转换

Python里命令： print int('xxx',16),原先是几进制就写数字几

Openssl语句：

（1）从公钥中读取e和n：openssl rsa -pubin -text -modulus -in warmup -in public.pem

参数：-pubin导入公钥时特有的，-text直接打印出明文，-modulus模数，-in导入文件(导入公钥文件时前面要有-pubin，导入私钥文件时不需要加-pubin)

```
root@kali:~/桌面# openssl rsa -pubin -text -modulus -in warmup -in public.key
Public-Key: (256 bit)
Modulus:
    00:d9:9e:95:22:96:a6:d9:60:df:c2:50:4a:ba:54:
    5b:94:42:d6:0a:7b:9e:93:0a:ff:45:1c:78:ec:55:
    d5:55:eb
Exponent: 65537 (0x10001)
Modulus=D99E952296A6D960DFC2504ABA545B9442D60A7B9E930AFF451C78EC55D555EB
writing RSA key
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhANmelSKWptlg38JQSrpUW5RC1gp7npMK
/0UceOxV1VXrAgMBAAE=
-----END PUBLIC KEY-----
root@kali:~/桌面#
```


（2）Rsatool.py生成指定的私钥：python rsatool.py -o private.pem -e 65537 -p 123 -q 123

（3）用私钥解密：openssl rsautl -decrypt -in flag.enc -inkey private.pem -out flag.txt

参数：这个语句主要用来用解密flag文件的，如flag.enc，解密这种文件的时候用私钥文件，私钥文件是用qpe三个参数利用python脚本生成的pem文件。将私钥文件和加密的flag文件放到kali里面执行这个语句。 Rsautl是openssl里的解密的包，-decrypt是解密，-in是导入待解密的文件，-inkey是导入私钥文件，-out是生成解密后的文件，一般是txt格式。

如果已知（十六进制的）cdn，进行解码，直接python里 hex(pow(c,d,n))[2:-1].decode('hex')

小总结：最简单的rsa密码解密就是先看给出的信息是几进制，如果是十六进制放进python里转换成十进制，然后找参数，ncpqe，用yafu或者在线进行大质数分解求pq，用脚本求出来十六进制，然后转换成文本字符串。

2.凯撒密码：

from pycipher import Caesar

Caesar(key=1).encipher('The quick brown fox jumps over the lazy dog')

1是偏移量，括号里面是要解密的文字

3.base

Base64：

（1）标准base64只有64个字符（英文大小写、数字和+、/）以及后缀“=”；

（2）base64是把3个字节变成4个可打印字符，所以base64编码后的字符串一定能被4整除

（3）等号一定用作后缀，且数目一定是0个、1个或2个。

（4）base64解码，用python

Python2>>>>import base64>>>>print base64 decodestring(‘xxxxxxxx’)

（5）base64编码还与后面几位有关系，所以要尽可能多的知道信息去转换，当转化完以后去搜索的时候，就只取前四位或者前三位。当用base64解码时，如果解不出来有可能是不完整。

Base32：

只有大写字母A-Z和数字234567

Base16

大写字母ABCDEF和数字0-9

4.替换密码

替换密码用python脚本

5.Hex转字符串

如果得到的全是数字，就是十进制，需要转换成十六进制。两种方法，第一种方法是python命令行里int(‘xxx’,16),第二种方法就是在python脚本里强制转换hex(n)。然后才可以转换成字符串，两种方法，第一种方法是在线，第二种方法是python脚本，在脚本里用binascii库里的a2b_hex方法转换成字符串(用得ASCII码)。

6.utf-9

提示rfc4042的时候就是utf-9编码的方式，解密需要用python脚本。首先需要安装utf9这个库，然后利用打开文件，写入文件，读取文件，利用utf9库里的utf9decode方法来解密。

7.Playfair密码

是一种双字替换密码，，所以字母个数一定是个偶数。最后一位如果有x的话极有可能是为了补位添加的字母，所以在提交flag的时候要适当的灵活。具体原理就不了解了，因为有直接的工具。

如何区分密文和秘钥：秘钥比密文长，并且秘钥里的字母不会重复

解决方法有两个，一个是在线，国外网站需要翻墙并且比较卡，所以不推荐；第二种方法是python，可以直接在交互模式里，也可以写脚本。在交互模式里的语句如下：

>>> from pycipher import Playfair

>>> Playfair('ZKLIPOAGSUMDWFHCBVTRYENXQ').encipher('FMGKYBXTSFBNCQDSPT') 注：第一个括号里秘钥，encipher是加密，所以第二个括号是待加密的。可以看到秘钥里的字母是不重复的。

>>> Playfair('ZKLIPOAGSUMDWFHCBVTRYENXQ').decipher('HDALECIXFTVERYFAIR') 注：第一个括号是秘钥，decipher是解密，所以第二个括号是密文，同样秘钥里的字母是不重复的。

8.压缩包解密

一种是伪加密，这种的不需要密码就可以解开。一种是需要密码的，需要暴力破解，可以用ziperello软件，只针对zip文件，也可以用ARCHPR，针对zip和rar。

伪加密的解密可以用两种方法，一种是手动，一种是自动。手动修改时，对zip或者rar直接用UE或者Hex打开，找到明文中 PK 一部分就是加密的，以此来找到版本标记位00 14 ，在这四个数字之后的四个数字，如果是00 00表示未加密，如果是00 09表示伪加密，只需要把9改成0即可。这个方法是在备份文件上修改。自动修改用ZipCenOp，命令是 java –jar ZipCenOp.jar r rsa.zip。这个方法在原文件上修改。

9.摩尔斯密码，替换密码，培根密码

摩尔斯密码：点，横之间的停顿，每个词之间中等的停顿以及每个句子之间长的停顿。与培根密码比较起来，每组密文的长度不固定。

培根密码：A和B之间的停顿。每一组密文长度为5 。(不止可以A和B，数量不一定)

替换密码：将明文中的元素替换另一密码表的元素。、

10.Rabbit密码直接用在线进行解密

11.哈希函数（MD2、MD4以及MD5等都属于哈希函数）

哈希算法是一个单向函数。它可以将任何大小的数据转化为定长的“指纹”，并且无法被反向计算。另外，即使数据源只改动了一丁点，哈希的结果也会完全不同。这样的特性使得它非常适合用于保存密码，因为我们需要加密后的密码无法被解密，同时也能保证正确校验每个用户的密码。需要提到的是，用于保护密码的哈希函数和你在数据结构中学到的哈希函数是不同的。比如用于实现哈希表这之类数据结构的哈希函数，它们的目标是快速查找，而不是高安全性。只有加密哈希函数才能用于保护密码，例如SHA256，SHA512，RipeMD和WHIRLPOOL。

如何破解哈希函数：

（1）字典攻击和暴力攻击：最简单的办法是猜，两种最常见的猜密码的方法就是字典攻击和暴力攻击。方法一：字典攻击需要一个字典文件，其中的每一个词都是经过哈希后储存的，将字典中的词和需要破解的词进行对比来破解。方法二：暴力攻击就是尝试每一个给定长度下的各种字符的组合，这样会消耗大量的计算，是效率最低的方法但是可以得到正确的密码。

（2）查表法：对于破解一系列算法相同的哈希值很有用。主要就是预先计算密码表中的每个密码，然后把哈希值和对应的密码储存到一个用于快速查询的数据结构中。

（3）反向查表法：可以使攻击者同时对多个哈希值发起字典攻击或者暴力攻击，而且不需要预先计算。方法就是先构造一个基于密码和用户名的一对多的表，数据需要从已经入侵的数据库获得，然后猜测一系列的哈希值并且从表中查找拥有次密码的用户，通常许多用户可能有着相同的密码，因此这种攻击方式比较有效。

（4）彩虹表：彩虹表是一种在时间和空间的消耗上找寻平衡的破解技术。它和查表法很类似，但是为了使查询表占用的空间更小而牺牲了破解速度。因为它更小，于是我们可以在一定的空间内存储更多的哈希值，从而使攻击更加有效。能够破解任何8位及以下长度MD5值的彩虹表已经出现了。

（4）加盐：会让查表法和彩虹表都失去作用。查表法和彩虹法只有在所有密码都以相同的方式进行哈希加密时才有效。如果两个用户密码相同，那么他们密码的哈希值也是相同的。我们可以通过“随机化”哈希来阻止这类攻击，于是当相同的密码被哈希两次之后，得到的值就不相同了。比如可以在密码中混入一段“随机”的字符串再进行哈希加密，这个被字符串被称作盐值。如同上面例子所展示的，这使得同一个密码每次都被加密为完全不同的字符串。为了校验密码是否正确，我们需要储存盐值。通常和密码哈希值一起存放在账户数据库中，或者直接存为哈希字符串的一部分。盐值并不需要保密，由于随机化了哈希值，查表法、反向查表法和彩虹表都不再有效。攻击者无法确知盐值，于是就不能预先计算出一个查询表或者彩虹表。这样每个用户的密码都混入不同的盐值后再进行哈希，因此反向查表法也变得难以实施。

12.对称加密和非对称加密

对称加密：对称加密是对称密码编码的技术，它的特点是文件加密和解密使用相同的密钥加密，也就是密钥可以用作解密密钥。对称机密使用起来简单快捷，密钥较短，并且破译困难。数据加密标准DES，国际数据加密算法IDEA。

非对称加密：非对称加密需要两个密钥，公开密钥和私有密钥。并且两个是成对使用，如果用公钥对数据进行加密，那么就要用对应的私钥进行解密，如果用私钥对数据进行加密，那么就要用对应的公钥进行解密。因为加密和解密是使用不同的密钥，所以是非对称加密算法。