

CTFRSA加解密

原创

[lens](#) 已于 2022-01-30 20:11:46 修改 192 收藏

分类专栏: [密码学](#) 文章标签: [密码学](#)

于 2021-10-06 00:06:35 首次发布

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/weixin_52369224/article/details/120618308

版权



[密码学](#) 专栏收录该内容

18 篇文章 0 订阅

订阅专栏

最近做题遇到好几次RSA解密, 最近密码学课也学到了 RSA 密码算法, 写一篇文章促进对 RSA 的理解。

一: **RSA 的基本原理的理解:**

1. 学习 RSA 加密得知道欧拉函数, 想理解欧拉函数又得弄明白同余类与剩余系的知识, 这些是基础。

2. **RSA 的密钥生成原理:**

第一步: 选择两个较大素数 P, Q .

第二步: 计算 $n=pq, z=(p-1)(q-1)$ 。

第三步: 随机选取 e (其中 $e < n$), e 与 z 没有公因数 (e, z 互为质数)

第四步: 选取 d 使得 $ed-1$ 能够被 z 完全整除。 $ed \bmod z=1$

第五部: 公钥是 (n, e) 私钥是 (n, d)

加密/解密算法：

如上所述给出 (n, e) 和 (n, d) 。

加密：由 $c = m^e \bmod n$ 将明文 m 转变为密文 c （即：当 m^e 除以 n 所得的余数）。

注意： $m < n$ （如果需要，则分块）

解密： $m = c^d \bmod n$ （即： c^d 除以 n 所得的余数）。

$$\text{核心思想： } m = \underbrace{(m^e \bmod n)}_c^d \bmod n$$

CSDN @lens7

二 下面以SUCTF 2019 signin记录一下RSA的应用与解密方法

```
IDA View-A Pseudocode-C Pseudocode-B Pseudocode-A Hex View-1 Struct
1 int64fastcall main(int a1, char **a2, char **a3)
2 {
3 char v4[16]; // [rsp+0h] [rbp-4A0h] BYREF
4 char v5[16]; // [rsp+10h] [rbp-490h] BYREF
5 char v6[16]; // [rsp+20h] [rbp-480h] BYREF
6 char v7[16]; // [rsp+30h] [rbp-470h] BYREF
7 char v8[112]; // [rsp+40h] [rbp-460h] BYREF
8 char v9[1000]; // [rsp+80h] [rbp-3F0h] BYREF
9 unsigned int64 v10; // [rsp+498h] [rbp-8h]
10
11 v10 = __readfsqword(0x28u);
12 puts("[sign in]");
13 printf("[input your flag]: ");
14 __isoc99_scanf("%99s", v8);
15 sub_96A(v8, v9);
16 __gmpz_init_set_str(v7, "ad939ff59f6e70bcbfad406f2494993757eee98b91bc244184a377520d06fc35", 16LL);
17 __gmpz_init_set_str(v6, v9, 16LL);
18 __gmpz_init_set_str(v4, "103461035900816914121390101299049044413950405173712170434161686539878160984549", 10LL);
19 __gmpz_init_set_str(v5, "65537", 10LL);
20 __gmpz_powm(v6, v6, v5, v4);
21 if ( (unsigned int) __gmpz_cmp(v6, v7) )
22 puts("GG!");
23 else
24 puts("TTTTTTTTTTq!");
25 return 0LL;
26 }
```

CSDN @lens7

这里函数分析得到

$N = 103461035900816914121390101299049044413950405173712170434161686539878160984549$

$c = 0xad939ff59f6e70bcbfad406f2494993757eee98b91bc244184a377520d06fc35$

$e = 65537$

第一步我们要求取的是pq两个素数的值 从其他大佬那里偷来神器RSAYafu 他可以讲给定的数分成两个相乘的素数

```
=== Starting work on batchfile expression ===
factor(103461035900816914121390101299049044413950405173712170434161686539878160984549)
=====
fac: factoring 103461035900816914121390101299049044413950405173712170434161686539878160984549
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits

starting SIQS on c78: 103461035900816914121390101299049044413950405173712170434161686539878160984549

==== sieving in progress (1 thread): 36224 relations needed ====
==== Press ctrl-c to abort and save state ====
36241 rels found: 18882 full + 17359 from 185645 partial, (3336.14 rels/sec)))

SIQS elapsed time = 62.3549 seconds.
Total factoring time = 62.3769 seconds

***factors found***

P39 = 366669102002966856876605669837014229419
P39 = 282164587459512124844245113950593348271

ans = 1

eof; done processing batchfile
```

CSDN @lens7

直呼nb

第二种使用方法:

如果因数过长, 将 因数 用文本文件存放在 yafu 目录下, 例如: data.txt。文件最后一行一定要换行, 否则 eof; done processing batchfile。

```
.\yafu-x64.exe "factor(@)" -batchfile data.txt
```

p = 366669102002966856876605669837014229419

q = 282164587459512124844245113950593348271

下面就是使用python解密了

(先得安装gmpy2库)

学习一下gmpy2库的使用

gmpy2.invert(a, c) 对a, 求b, 使得 $a*b=1 \pmod{c}$

gmpy2.powmod(a,n,p) 对于给定的整数p,n,a,计算 $a^n \pmod{p}$

补充一下binascii.unhexlify函数

unhexlify(hexstr)将十六进制转换为二进制形式。

hex() 函数记得加[2:]去掉开头的0x。

求d脚本

```
import gmpy2
from Crypto.Util import number
e=65537
n=87924348264132406875276140514499937145050893665602592992418171647042491658461
p=275127860351348928173285174381581152299
q=319576316814478949870590164193048041239
d=gmpy2.invert(e,(p-1)*(q-1))
print(d)
```

密文拿到手之后，需要密文的数字形式，用winhex打开查看十六进制形式

最后得到的密文的十六进制形式数字个数需要是偶数被，不是则尝试加0或者删除