

攻防世界crypto高手题之cr3-what-is-this-encryption

原创

[沐一·林](#) 于 2021-08-29 15:02:21 发布 155 收藏 1

分类专栏: [CTF 密码学](#) 文章标签: [ctf](#)

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/xiao__1bai/article/details/119978997

版权



[CTF](#) 同时被 2 个专栏收录

167 篇文章 6 订阅

订阅专栏



[密码学](#)

51 篇文章 1 订阅

订阅专栏

[攻防世界crypto高手题之cr3-what-is-this-encryption](#)

继续开启全栈梦想之逆向之旅~

这题是攻防世界crypto高手题的cr3-what-is-this-encryption

The screenshot shows a CTF problem page with the following details:

- 标题:** cr3-what-is-this-encryption
- 难度系数:** 1.0
- 题目来源:** alexctf-2017
- 题目描述:** Fady同学以为你是菜鸟, 不怕你看到他发的东西。他以明文形式将下面这些东西发给了他的朋友。题目包含三个被红框圈出的十六进制字符串: `p=0xa6055ec186de51800ddd6fcbf0192384ff42d707a55f57af4fcfb0d1dc7bd97055e8275cd4b78ec63c5d592f567c66393a061324aa2e6a8d8fc2a910cbee1ed9`, `q=0xfa0f9463ea0a93b929c099320d31c277e0b0dbc65b189ed76124f5a1218f5d91fd0102a4c8de11f28be5e4d0ae91ab319f4537e97ed74bc663e972a4a911930`, and `e=0x6d1fdab4ce3217b3fc32c9ed480a31d067fd57d93a9ab52b472dc393ab7852fbc11abbef6d6aaae8032db1316dc22d3f7c3d631e24df13ef23d3b381a1c3e04abcc745d402ee3a031ac2718fae63b240837b4f657f29ca4702da9af22a3a019d68904a969ddb01bcf941df70af042f4fae5cbeb9c2151b324f387e525094c41`.
- 题目场景:** 暂无
- 题目附件:** 暂无

没有附件, 依稀看得出有q、p、e、c, 是简单的RSA题目, 用我之前积累的RSA脚本CTF-RSA-tool跑一下, 。。。跑不出来: (反复试了好多次, 还是报错, 我放弃了)

```
1 p=0xa6055ec186de51800ddd6fcbf0192384ff42d707a55f57af4fcfb0d1dc7bd97055e8275cd4b78ec63c5d592f567c66393a061324aa2e6a8d8fc2a910cbee1ed9
2 q=0xfa0f9463ea0a93b929c099320d31c277e0b0dbc65b189ed76124f5a1218f5d91fd0102a4c8de11f28be5e4d0ae91ab319f4537e97ed74bc663e972a4a9119307
3 e=0x6d1fdab4ce3217b3fc32c9ed480a31d067fd57d93a9ab52b472dc393ab7852fbc11abbef6d6aaae8032db1316dc22d3f7c3d631e24df13ef23d3b381a1c3e04abcc745d402ee3a031ac2718fae63b240837b4f657f29ca4702da9af22a3a019d68904a969ddb01bcf941df70af042f4fae5cbeb9c2151b324f387e525094c41
4 c=0x7fe1a4f743675d1987d25d38111fae0f78bbea6852cba5bed47b76d119a3efe24cb04b9449f53becd43b0b46e269826a983f832abb53b7a7e24a43ad15378344ed5c20f51e268186d24c76050c1e73647523bd5f91d9b6ad3e86bbf9126588b1dee21e6997372e36c3e74284734748891829665086e0dc523ed23c386bb520
5 print(e)
6 print(p*q)
7 print(c)
8
```

CSDN @沐一一沐, 一沐沐一

The terminal screenshot shows the following output:

```
└─$ python2 CTF-RSA-tool/solve.py --verbose -N 1138791748982046686459671176938061766539472
559382394157624841784399330290399424809923789628217421160421321213841308942712637693921297
285568975045069619940397470969800035442388796916416828402430166605650021995804021584858887
97811110890373870112239180679080062627348028514800765210824649322666826925080318210799 -e
766297813873972426643116709874317578271441392556392807529834168670310153073520143866486739
942179138155817821866364881591859652274493031187833628624358994867175044572336498295631763
539498171499977732764355819103705595946395704361205962111489732270775657394676413094269445
29006537681147498322988959979899800641 -c 898013894435695699573984069547075984927639234185
685360303235460882787583623310431197364379101176970325948359029005820403943674808298008972
319252338077452783893580314042780646333136261493367249458548650414390611494119625092476244
19448003604874406282213609341704339025169015256228029200222643343430028828063008
DEBUG: factor N: try past ctf primes
DEBUG: factor N: try Gimmicky Primes method
DEBUG: factor N: try Wiener's attack
DEBUG: Starting new HTTP connection (1): www.factordb.com:80
DEBUG: http://www.factordb.com:80 "GET /index.php?query=1138791748982046686459671176938061
766539472559382394157624841784399330290399424809923789628217421160421321213841308942712637
693921297285568975045069619940397470969800035442388796916416828402430166605650021995804021584858887
```

CSDN @沐一一沐, 一沐沐一

然后在网上查资料中找到一个讲得比较好的脚本和讲解:
借鉴于:

RSA的密钥对生成算法:

RSA的密钥对生成算法:
选取两个大素数p和q(两个数长度接近,一般在256比特长),而且p和q保密
计算 $n=pq$,将n公开
计算 $\psi(n)=(p-1)(q-1)$,对 $\psi(n)$ 保密
随机选取一个正整数e, $1 < e < \psi(n)$ 满足 $\gcd(e, \psi(n))=1$ 将e公开
根据 $ed=1 \bmod \psi(n)$,求出d,并对d保密

公式: $C = M^E \bmod N$
上图是加密算法:
图中的C是密文, M是明文, E是公钥 (E和 $\phi(N)$ 互为质数), N是公共模数 (质数 P、Q相乘得到N), MOD就是模运算

$M = C^D \bmod N$
上图是解密算法:
图中的C是密文, M是明文, D是私钥 (私钥由这个公式计算得出 $E * D \% \phi(N) = 1$), N是公共模数 (质数 P、Q相乘得到N), MOD就是模运算, $\phi(N)$ 是欧拉函数 (由这个公式计算出 $\phi(N) = (P-1)(Q-1)$)。

大致思路:
先把p,q,e转成十进制,再根据公式求出n,d,m
 $n=p*q$
 $\phi(N) = (p-1)(q-1)$
 $e * d \% \phi(N) = 1$ (d是私钥, e是公钥)
 $m = c^d \bmod n$ (m是明文)

 $d = \text{libnum.invm}(e, (p - 1) * (q - 1))$
#invm(a, n) - 求a对于n的模逆,这里逆向加密过程中计算 $\psi(n) = (p-1)(q-1)$,对 $\psi(n)$ 保密,也就是对应根据 $e*d=1 \bmod \psi(n)$,求出d

 $m = \text{pow}(c, d, n)$
#这里的m是十进制形式,pow(x, y[, z])函数是计算x的y次方,如果z存在,则再对结果进行取模,其结果等效于 $\text{pow}(x,y) \% z$,对应前面解密算法中 $M=D^C = (C^d) \bmod n$

 $\text{string} = \text{long_to_bytes}(m)$ # 获取m明文

CSDN @沐一一沐, 一沐沐一

讲得很透彻了, 解密的算法也提供了, 附上脚本和我的一点点见解:

```

import libnum
from Crypto.Util.number import long_to_bytes

q = int(
    "0xa6055ec186de51800ddd6fcbf0192384ff42d707a55f57af4fcfb0d1dc7bd97055e8275cd4b78ec63c5d592f567c66393a061324a
a2e6a8d8fc2a910cbee1ed9",
    16)
p = int(
    "0xfa0f9463ea0a93b929c099320d31c277e0b0dbc65b189ed76124f5a1218f5d91fd0102a4c8de11f28be5e4d0ae91ab319f4537e97
ed74bc663e972a4a9119307",
    16)

e = int(
    "0x6d1fdab4ce3217b3fc32c9ed480a31d067fd57d93a9ab52b472dc393ab7852fbc11abbebfd6aaae8032db1316dc22d3f7c3d631e
24df13ef23d3b381a1c3e04abcc745d402ee3a031ac2718fae63b240837b4f657f29ca4702da9af22a3a019d68904a969ddb01bcf941df70
af042f4fae5cbeb9c2151b324f387e525094c41",
    16)

c = 0x7fe1a4f743675d1987d25d38111fae0f78bbea6852cba5beda47db76d119a3efe24cb04b9449f53becd43b0b46e269826a983f832a
bb53b7a7e24a43ad15378344ed5c20f51e268186d24c76050c1e73647523bd5f91d9b6ad3e86bbf9126588b1dee21e6997372e36c3e74284
734748891829665086e0dc523ed23c386bb520

n = q * p

d = libnum.invmod(e, (p - 1) * (q - 1)) #invmod(a, n) - 求a对于n的模逆,这里逆向加密过程中计算 $\psi(n)=(p-1)(q-1)$ ,对 $\psi(n)$ 
保密,也就是对应根据 $ed=1\text{mod}\psi(n)$ ,求出d
m = pow(c, d, n) # pow(x, y[, z])--函数是计算 x 的 y 次方,如果 z 在存在,则再对结果进行取模,其结果等效于 pow(x,y) %z
,对应前面解密算法中 $M=D(C)=C^d\text{mod } n$ 
#print(m) #明文的十进制格式
string = long_to_bytes(m) # m明文,用长字节划范围
print(string.decode())

.
.

```

总结:

嗯~没啥好讲的,学了个RSA加密解密基本算法和对脚本有一点点理解吧。

RSA的密钥对生成算法:

RSA的密钥对生成算法:
选取两个大素数p和q(两个数长度接近,一般在256比特长),而且p和q保密
计算 $n=pq$,将n公开
计算 $\psi(n)=(p-1)(q-1)$,对 $\psi(n)$ 保密
随机选取一个正整数e, $1 < e < \psi(n)$ 满足 $\text{gcd}(e, \psi(n))=1$ 将e公开
根据 $ed=1 \bmod \psi(n)$,求出d,并对d保密

$$\text{公式: } C = M^E \bmod N$$

上图是加密算法:

图中的C是密文, M是明文, E是公钥 (E和 $\varphi(N)$ 互为质数), N是公共模数 (质数 P、Q相乘得到N), MOD就是模运算

$$M = C^D \bmod N$$

上图是解密算法:

图中的C是密文, M是明文, D是私钥 (私钥由这个公式计算出 $E * D \% \varphi(N) = 1$), N是公共模数 (质数 P、Q相乘得到N), MOD就是模运算, $\varphi(N)$ 是欧拉函数 (由这个公式计算出 $\varphi(N) = (P-1)(Q-1)$).

大致思路:

先把p,q,e转成十进制,再根据公式求出n,d,m

$$n=p*q$$

$$\varphi(N) = (p-1)(q-1)$$

$$e * d \% \varphi(N) = 1 \quad (d \text{ 是私钥, } e \text{ 是公钥})$$

$$m = c^d \bmod n \quad (m \text{ 是明文})$$

$$d = \text{libnum.invm}(e, (p-1) * (q-1))$$

#invm(a, n) - 求a对于n的模逆,这里逆向加密过程中计算 $\psi(n) = (p-1)(q-1)$,对 $\psi(n)$ 保密,也就是对应根据 $e*d=1 \bmod \psi(n)$,求出d

$$m = \text{pow}(c, d, n)$$

#这里的m是十进制形式,pow(x, y[, z])—函数是计算x的y次方,如果z存在,则再对结果进行取模,其结果等效于 $\text{pow}(x,y) \% z$,对应前面解密算法中 $M=D^C = (C^d) \bmod n$

string = long_to_bytes(m) # 获取m明文

CSDN @沐一一沐, 一沐沐一

解毕! 敬礼!



[创作打卡挑战赛](#)

[赢取流量/现金/CSDN周边激励大奖](#)