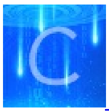
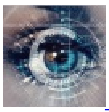
HSCTF-部分writeup

原创

逃课的小学生 于 2020-06-07 17:57:13 发布 421 收藏
分类专栏： crypto ctf

本文链接：https://blog.csdn.net/zhang14916/article/details/106597866


crypto 同时被 2 个专栏收录
20 篇文章 1 订阅
订阅专栏


ctf
30 篇文章 2 订阅
订阅专栏

1.XORed

根据题意，我们很容易发现这是一个异或加密，根据异或的运算法则，我们很容易解密，下面是我们的writeup

```
Key1 = 0x5dcec311ab1a88ff66b69ef46d4aba1aee814fe00a4342055c146533
Key13 = 0x9a13ea39f27a12000e083a860f1bd26e4a126e68965cc48bee3fa11b
Key235 = 0x557ce6335808f3b812ce31c7230ddea9fb32bbaeaf8f0d4a540b4f05
Key145 = 0x7b33428eb14e4b54f2f4a3acaeab1c2733e4ab6bebc68436177128eb
Key34 = 0x996e59a867c171397fc8342b5f9a61d90bda51403ff6326303cb865a
FlagKey12345= 0x306d34c5b6dda0f53c7a0f5a2ce4596cfea5ecb676169dd7d5931139
Key45=Key1^Key145
key12345=Key1^Key13^Key235^Key145^Key34^Key45
Flag=FlagKey12345^key12345
print hex(Flag)[2:-1].decode("hex")
```

2.Chonky E

题目中提到一种加密算法Schmidt-Samoa cryptosystem。我们查询其原理大致是选择两个素数p,q，令N=pow(p,2)*q，c=pow(m,N,N)。其解密公式也是先算出d=pow(N,-1,lcm(p-1,q-1))，再解密m=pow(c,d,p*q)。由于Schmidt-Samoa cryptosystem和开始的RSA使用相同的p,q。而RSA宫要不已知且e接近n。所以我们先使用winner-attack计算出d，phin。再将n分解。然后根据RSA的p,q对Schmidt-Samoa cryptosystem解密

```
import gmpy2
e = 9104311840982855079677374551858598115118020610100513511756586560297872287847849444704878355757181398052
n = 1567490475585830139605132673517694799151104404114480784125905657970315336225098133520931196368355119772
#winner-attack算得
d= 0x5baecf6f9f0a1bd295e9650b4a10b4a717db030223e803f8e964a18ab9bcfc0954a8f410cc00177ad9f6a0d581e12c6dfd0672
phi= 0x4d9b10f2117ddad53727e6ce6599681f7275049e10057e06a66272a5c384f8f6c1f259836af50469b66c560e7d6999ed7071
cc=16267540901004879123859424672087486188548628828063789528428674467464407443871599865993337555869530486241
a=1
b=(phi-1-n)
c=n
delat=pow(b,2)-4*a*c
ii=gmpy2.iroot(delat,2)
p=(ii[0]-b)/(2*a)
q=n//p
assert p*q==n
NN=pow(p,2)*q
d=gmpy2.invert(NN,gmpy2.lcm((p-1),(q-1)))
m=pow(cc,d,n)
print hex(m)[2:].decode("hex")
```

3.Morbid

经查询，我们发现这是一个首先将数字对应到摩斯电码，再将摩斯电码转为明文的解密过程。其中数字[1,2,3,4,5,6,7,8,9]对应字符['..', '.-', '.x', '-.', '--', '-x', 'x.', 'x-', 'xx']，但是我们不知道具体那个数字对应哪一个字符。然后摩斯密码中一个x表示字母之间的分割符，两个x表示单词之间的分隔符。我们据此对密文解密——首先对[1,2,3,4,5,6,7,8,9]和['..', '.-', '.x', '-.', '--', '-x', 'x.', 'x-', 'xx']之间的关系进行爆破。将不同的对应关系一一带入尝试解密，知道可以正确解密且密文中有flag字样表明我们爆破成功

```
import itertools
c="11828929393843419384927146411742936447699424147315766496987969693814568947439364729439273924772165282241
zimucodebook  = {
    "a": ".-",
    "b": "-...",
    "c": "-.-.",
    "d": "-..",
    "e": ".",
    "f": "..-.",
    "g": "--.",
    "h": "....",
    "i": "..",
    "j": ".---",
    "k": "-.-",
    "l": ".-..",
    "m": "--",
    "n": "-.",
    "o": "---",
    "p": ".--.",
    "q": "--.-",
    "r": ".-.",
    "s": "...",
    "t": "-",
    "u": "..-",
    "v": "...-",
    "w": ".--",
    "x": "-..-",
    "y": "-.--",
```

```
    "Z": "--..",
    "0": "-----",
    "1": ".----",
    "2": "..---",
    "3": "...--",
    "4": "....-",
    "5": ".....",
    "6": "-....",
    "7": "--...",
    "8": "---..",
    "9": "----.",
    ".": ".-.-.-",
    ",": "--..--",
    "?": "..--..",
    "'": ".----.",
    "!": "-.-.--",
    "/": "-..-.",
    "(": "-.--.",
    ")": "-.--.-",
    "&": ".-...",
    ":": "---...",
    ";": "-.-.-.",
    "=": "-...-",
    "+": ".-.-.",
    "-": "-....-",
    "_": "..--.-",
    "\"": ".-..-.",
    "$": "...-..-",
    "@": ".--.-.",
    " ": ""
}
zhuan1=['..', '.-', '.x', '-.', '--', '-x', 'x.', 'x-', 'xx']
revzimucodebook={}
for key, value in zimucodebook.items():
    revzimucodebook[value] = key
for perm in itertools.permutations(range(9)):
    m1=c
    for i in range(9):
        m1=m1.replace(str(i+1),zhuan1[perm[i]])
    words = m1.split('x')
    try:
        mm="".join(revzimucodebook[word] for word in words)
        if "flag" in mm:
            print mm
            break
    except KeyError:
        continue
```

3.Randomization 1，Randomization 2

比较简单得逆向，使用了线性方程产生随机数列，根据逆向得到的线性方程和题目中给出得初始数，直接对后面得数字进行预测即可

4.Unexpected

根据题目易知不同的RSA机密公钥的n之间最大公约数不为1.根据这一点我们很快就可以对RSA做分解，解出私钥对密文求解

```
import gmpy2
N1 = 389573830229905951812919842231016962853053655719189056621093978169837233625748218658216363084761241627
N2 = 303668390381967550574109116494546194718900491649463376637217628240940969495870121174827705049910151195
N3 = 479345567729954913738228458501575007323911241436168052925595131821796030084134039909474313028792799656
E = 65537
C1 = 396708474546125804352894757436683688457291028695044217325853929491171136935487190613513217479209066321
C2 = 355006513750551550798931713354683491263062473879176656452255051848683497534660576981575518851351256702
C3 = 924835278307680480966328618545268895077532556525413716080960421925985654497130329688156219485942736928

q=gmpy2.gcd(N1,N2)
r=gmpy2.gcd(N2,N3)
p=gmpy2.gcd(N1,N3)
d1=gmpy2.invert(E,(p-1)*(q-1))
d2=gmpy2.invert(E,(r-1)*(q-1))
d3=gmpy2.invert(E,(p-1)*(r-1))
m1=pow(C1,d1,N1)
m2=pow(C2,d2,N2)
m3=pow(C3,d3,N3)
print hex(m1)[2:].decode("hex")+ hex(m2)[2:].decode("hex")+ hex(m3)[2:].decode("hex")
```

5.smolE

本题使用相同的RSA加密体系对有着不同填充的明文加密。这里我们可以首先使用Random Padding Attack计算两个明文之间的差值是多少，然后使用Related Message Attacks计算出明文。注意这里由于对明文进行的填充，但我们不知道填充有多少位，所以我们需要爆破明文左移1位到8位的情况

Random Padding Attack：

Theorem 4.5. *Let (e, N) be a valid RSA public key with $e = 3$. Let m_1 and m_2 be two plaintext messages satisfying $m_2 = m_1 + b$. Given $c_1 = m_1^3 \bmod N$, $c_2 = (m_1 + b)^3 \bmod N$ and the public key, if $|b| < N^{1/9}$ then the plaintexts m_1 and m_2 can be computed in time polynomial in $\log(N)$.*

Proof: Since $m_1^3 - c_1 \equiv 0 \pmod N$ and $(m_1 + b)^3 - c_2 \equiv 0 \pmod N$, it follows that

$$\begin{aligned} &\text{Resultant}_{m_1}\left(m_1^3 - c_1, (m_1 + b)^3 - c_2\right) \\ &\quad \equiv \; b^9 + (3c_1 - 3c_2)b^6 + (3c_1^2 + 21c_1c_2 + 3c_2^2)b^3 + (c_1 - c_2)^3 \pmod N \\ &\quad \equiv \; 0 \pmod N. \end{aligned}$$

From this resultant computation, notice that the monic degree 9 polynomial $f_N(x) \in \mathbb{Z}_N[x]$, given by

$$f_N(x) = x^9 + (3c_1 - 3c_2)x^6 + (3c_1^2 + 21c_1c_2 + 3c_2^2)x^3 + (c_1 - c_2)^3 \bmod N,$$

Related Message Attacks：

Theorem 4.2. *Let (e, N) be a valid RSA public key with $e = 3$. Let m_1 and m_2 be two plaintext messages satisfying $m_2 = am_1 + b$. Given a, b, $c_1 = m_1^3 \bmod N$, $c_2 = m_2^3 \bmod N$, and the public key, both m_1 and m_2 can be compute in time polynomial in $\log(N)$.*

Proof: Given c_1, c_2, a, b and N, the plaintext m_1 can be directly computed since

$$\frac{b(c_2 + 2a^3c_1 - b^3)}{a(c_2 - a^3c_1 + 2b^3)} \bmod N = \frac{m_1(3a^3bm_1^2 + 3a^2b^2m_1 + 3ab^3)}{3a^3bm_1^2 + 3a^2b^2m_1 + 3a\beta^3} \bmod N = m_1.$$

Once m_1 is known, we simply compute $m_2 = am_1 + b$. If the computation fails (*i.e.*, the denominator does not exists modulo N) then a factor of N is found and the system is completely broken. Since all computations can be done in time polynomial in $\log(N)$, the result follows. □

writeup如下——我们首先使用Random Padding Attack计算两个明文之间的差，再使用Related Message Attacks计算出明文的值

```
N = 1637410392895129134482113164442084150896962811565987075462399390609300053008010500411105934458085900198
E = 3
C1 = 110524539798470366613834133888472781069399552085868942087632499354651575111511036068021885688092481936

C2 = 424068377350933679416828578921815505223462204275047549885441408869973397097853803036824713681681020026

PR.<x> = PolynomialRing(Zmod(N))
f = x^9+(3*C1-3*C2)*x^6+(3*C1^2+21*C1*C2+3*C2^2)*x^3+(C1-C2)^3
x0 = f.small_roots(X=2^64, beta=0.2)[0]

b=x0
fenzi=b*(C2+2*C1-b^3)
fenmu=C2-C1+2*(b^3)
m1=(fenzi/fenmu)%N
m2=m1+b
print m1
print m2
```

接下来我们去掉明文填充

```
import gmpy2
def shuchu(mingwenstr):
    if mingwenstr[len(mingwenstr)-1]=='L':
        mingwenstr=mingwenstr[2:len(mingwenstr)-1]
    else:
        mingwenstr=mingwenstr[2:len(mingwenstr)]
    if not len(mingwenstr)%2==0:
            mingwenstr='0'+mingwenstr
    i=len(mingwenstr)
    mingwen=""
    while i>=1:
        str1=mingwenstr[i-2:i]
        if int(str1,16)>33 and int(str1,16)<128:
            mingwen=chr(int(str1,16))+mingwen
        else :
            mingwen=" "+mingwen
        i=i-2
    print mingwen


m1=14260511615962734137955566543283201051454393321475854185075767758707804505903795674536414290826408429359
shuchu(hex(m1<<7))
```

6.Extremely Complex Challenge

我们已知椭圆曲线的b,p和椭圆曲线上的基点P一个Q，求私钥。我们首先根据点将椭圆曲线的参数a求出。然后我们暴力求解私钥

```
b=54575449882
p=404993569381
xp=391109997465
yp=167359562362
xpinv=inverse_mod(xp,p)
a=(((yp)^2-(xp)^3-b)*inverse_mod(xp,p))%p
E=EllipticCurve(GF(p), [a,b])
Ep=E([391109997465, 167359562362])
Eq=E([209038982304, 168517698208])
d1 = discrete_log(Eq, Ep, Ep.order(), operation="+")
print d1
```