ctf之lcg算法

原创

小健健健 于 2020-10-08 17:12:31 发布 3825 收藏 30

分类专栏： ctf python 文章标签： 安全

本文链接：https://blog.csdn.net/superprintf/article/details/108964563

ctf 同时被 2 个专栏收录

7 篇文章 0 订阅

订阅专栏

python

7 篇文章 0 订阅

订阅专栏

线性同余方法（LCG）是一种产生伪随机数的方法。

$$X_{n+1} = (aX_n + b) \mod m$$

线性同余法最重要的是定义了三个整数，乘数 a、增量 b和模数 m，其中a,b,m是产生器设定的常数。

为了方便理解，我打个比方

假设现在有随机数X1=1234，乘数a=2，增量b=3，模数m=1000

那么下一个随机数X2=(2*1234+3)%1000=2471%1000=471

解题用到的公式：

目的	公式
1.X_{n+1}反推出X_n	$X_n=(a^{-1}(X_{n+1}-b))\%m$
2.求a	$a=((X_{n+2}-X_{n+1})(X_{n+1}-X_n)^{-1})\%m$
3.求b	$b=(X_{n+1}-aX_n)\%m$
4.求m	$t_n=X_{n+1}-X_n$，$m=gcd((t_{n+1}t_{n-1}-t_nt_n),(t_nt_{n-2}-t_{n-1}t_{n-1}))$

下面是公式证明：

其实公式证明挺复杂的可以最后看，先看看例题也不错哦

公式1:

$X_{n+1} = aX_n + b \pmod m$

$aX_n = X_{n+1} - b \pmod m$

$X_n = a^{-1}(X_{n+1} - b) \pmod m$

模逆运算用到扩展欧几里得算法

```
MMI = lambda A, n,s=1,t=0,N=0: (n < 2 and t%N or MMI(n, A%n, t, s-A//n*t, N or n),-1)[n<1] #逆元计算
```

公式2

解先行方程组

$X_{n+2} = aX_{n+1} + b \pmod m$

$X_{n+1} = aX_n + b \pmod m$

$X_{n+2} - X_{n+1} = a(X_{n+1}-X_n) \pmod m$

$a = (X_{n+2} - X_{n+1})(X_{n+1} - X_n)^{-1} \pmod m$

公式3

$X_{n+1} = aX_n + b \pmod m$

$b = X_{n+1} - aX_n \pmod m$

公式4

设$t_n = X_{n+1} - X_n$

$t_n = (aX_n+b) - (aX_{n-1}+b) = at_{n-1} \pmod m$

$t_{n+1}t_{n-1} - t_nt_n = (aat_{n-1}t_{n-1} - at_{n-1}at_{n-1}) = 0 \pmod m$

即$T_n = t_{n+1}t_{n-1} - t_nt_n$是m的倍数，求$T_n$，$T_{n-1}$最大公因数即为m

$m = gcd((t_{n+1}t_{n-1} - t_nt_n), (t_nt_{n-2} - t_{n-1}t_{n-1}))$

好了，然后根据lcg算法有六种ctf题型

lcg-1

```
from Crypto.Util.number import *
flag = b'Spirit{**********************}'

plaintext = bytes_to_long(flag)
length = plaintext.bit_length()

a = getPrime(length)
b = getPrime(length)
n = getPrime(length)

seed = 334771285231401057643016442247213789640 69
print("seed = ",seed)
for i in range(10):
    seed = (a*seed+b)%n
ciphertext = seed^plaintext
print("a = ",a)
print("b = ",b)
print("n = ",n)
print("c = ",ciphertext)
# seed =  334771285231401057643016442247213789640 69
# a =  216636540518719887613942270143367229109002078444183475587474655399326769391
# b =  186914533399403414430047931765983818420963789311681346652500920904075344361
# n =  155908129777160236018105193822448288416284495517789603884888599242193844951
# c =  209481865531297761516458182436122824479565806914713408748457524641378381493
```

这道题的题意是:

他定义了一个变量叫`plaintext`，长字节字符串`flag`转换为整数得到的.

定义了一个整数`seed = 334771285231401057643016442247213789640 69`

seed通过10次lcg转换之后再和plaintext二进制异或得到`ciphertext`

`getPrime`是根据输入length产生随机数的函数

思路:

想要得到flag就要知道plaintext

想要知道plaintext只能通过`ciphertext = seed ^ plaintex`这个表达式推出

而ciphertext==c我们知道，式子中的seed可以通过他给的初始seed,a,b,n运用算法lcg十次得到

然后根据异或的特性求解出plaintext，即`plaintext = seed ^ ciphertext`

(异或特性，c=a异或b，那么a=b异或c，或者b=a异或c)

解题代码如下

```
seed = 334771285231401057643016442247213789640 69
a = 216636540518719887613942270143367229109002078444183475587474655399326769391
b = 186914533399403414430047931765983818420963789311681346652500920904075344361
n = 155908129777160236018105193822448288416284495517789603884888599242193844951
c = 209481865531297761516458182436122824479565806914713408748457524641378381493

for i in range(10):
    seed = (a*seed+b)%n
plaintext=seed^c
print(long_to_bytes(plaintext))
```

```
答案Spirit{0ops!___you_know__LCG!!}
```

lcg-2

```
from Crypto.Util.number import *
flag = b'Spirit{****************************}'

plaintext = bytes_to_long(flag)
length = plaintext.bit_length()

a = getPrime(length)
b = getPrime(length)
n = getPrime(length)

seed = plaintext

for i in range(10):
    seed = (a*seed+b)%n
ciphertext = seed

print("a = ",a)
print("b = ",b)
print("n = ",n)
print("c = ",ciphertext)

# a =  59398519837969938359106832224056187683937568250770488082448642852427682484407513407602969
# b =  32787000674666987602016858366912565306237308217749461581158833948068732710645816477126137
# n =  43520375935212094874930431059580037292338304730539718469760580887565958566208139467751467
# c =  85945144528080463573376829115040748580482995137438678879367944391259494181535618418422 76
```

根据题意求flag相当于求plaintext，求plaintext相当于求初始seed，我们知道lcg算法十次之后的seed=ciphertext=c，而c我们已知，我们还知道a，b，n。所以这道题要我们从lcg十次之后的seed推算出初始seed

用公式1：$X_n=(a^{-1}(X_{n+1}-b))\%n$

这里a^{-1}是a相对于模数m的逆元，他也有公式

```
MMI = lambda A, n,s=1,t=0,N=0: (n < 2 and t%N or MMI(n, A%n, t, s-A//n*t, N or n),-1)[n<1] #逆元计算
```

所以解题代码如下

```
a =  593985198379699383591068322240561876839375682507704880824486428524276824844075134076029 69
b =  327870006746669876020168583669125653062373082177494615811588339480687327106458164771261 37
n =  435203759352120948749304310595800372923383047305397184697605808875659585662081394677514 67
c =  859451445280804635733768291150407485804829951374386788793679443912594941815356184184227 6
MMI = lambda A, n,s=1,t=0,N=0: (n < 2 and t%N or MMI(n, A%n, t, s-A//n*t, N or n),-1)[n<1] #逆元计算
ani=MMI(a,n)
seed=c
for i in range(10):
    seed = (ani*(seed-b))%n
print(long_to_bytes(seed)
```

```
答案Spirit{Orzzz__number_the0ry_master!!}
```

lcg-3

```
from Crypto.Util.number import *
flag = b'Spirit{********************}'
plaintext = bytes_to_long(flag)
length = plaintext.bit_length()

a = getPrime(length)
seed = getPrime(length)
n = getPrime(length)

b = plaintext

output = []
for i in range(10):
    seed = (a*seed+b)%n
    output.append(seed)
ciphertext = seed

print("a = ",a)
print("n = ",n)
print("output1 = ",output[6])
print("output2 = ",output[7])

# a =  322781795536447153434915714267864829125829739876721046973412707257153 1
# n =  273155913534969029926147029420074232502157562037767349274757036248435 9
# output1 =  565897873786681926180964326939259355991528156340765285489917686416 73
# output2 =  255179106638051559639398419399518067183953160327340990702687163700246 0
```

求flag相当于求plaintext，plaintext相当于求b，然后在看看我们已知的条件，我们知道a，知道n知道10次lcg中的第6次和第7次的结果，所以我们要根据已知的信息求b

用公式3：$b=(X_{n+1}-aX_n)\%n$直接求出b

上代码

```
a =  322781795536447153434915714267864829125829739876721046973412707257153 1
n =  273155913534969029926147029420074232502157562037767349274757036248435 9
output1 =  56589787378668192618096432693925935599152815634076528548991768641673
output2 =  25517910663805155963939841939951806718395316032734099070268716370024 60
b=(output2-a*output1)%n
plaintext=b
print(long_to_bytes(plaintext))
```

```
答案Spirit{Y0u_@r3_g00d_at__math}
```

lcg-4

```
from Crypto.Util.number import *
flag = b'Spirit{******************************}'

plaintext = bytes_to_long(flag)
length = plaintext.bit_length()

a = getPrime(length)
b = getPrime(length)
n = getPrime(length)

seed = plaintext
output = []
for i in range(10):
    seed = (a*seed+b)%n
    output.append(seed)


print("n = ",n)
print("output = ",output)
# n =  714326667532888136341930300469812503108568533171958701229258381897431946521867367344505142446819
# output =  [68388415013556756905470030939308227401527341875501598463921087264162910277613728890533434535822 3, 2
851262210392394013476645787613099356738891932365127021316970507664548810293401471805524098704 25, 276893085775448
203669487661735680485319995668779836512706851431217470824660349740546793492847822, 670041467944152108349892479 46
303380839324947560893311064058038887720670011666107030238257838862 9, 122640993538161410588195475312610802051543 1
550603289714882772241120811667842631531076361088158 24, 695403107966797625391061914491496301998976621394944936827
202540832952594905520247784142392337171, 108297989103402878258100342544600235524390749601427490182149765480916 96
581165200088123050483894 9, 334890160364790302060735621729199964480057977539225173205956219308086252467158423520 3
807354488, 632094372828241320671255647451901056399237760301503199444470380543753167478243100611604222284853, 547
580618792250241258969096450342671069735142431883586773112380708321548837820284372036217092 76]
```

第四题给了n和10次lcg的output序列

用公式2：$a=((X_{n+2}-X_{n+1})(X_{n+1}-X_n)^{-1})\%n$

用已知信息可以求出a

再用a，output序列，n求出b

根据output序列第一个反推出初始seed

即可求解

```
n =  714326667532888136341930300469812503108568533171958701229258381897431946521867367344505142446819
output =  [68388415013556756905470030939308227401527341875501598463921087264162910277613728890533434535822 3, 285
1262210392394013476645787613099356738891932365127021316970507664548810293401471805524098704 25, 27689308577544820
3669487661735680485319995668779836512706851431217470824660349740546793492847822, 670041467944152108349892479463 0
3380839324947560893311064058038887720670011666107030238257838862 9, 122640993538161410588195475312610802051543155
0603289714882772241120811667842631531076361088158 24, 695403107966797625391061914491496301998976621394944936827 20
2540832952594905520247784142392337171, 108297989103402878258100342544600235524390749601427490182149765480916965 8
1165200088123050483894 9, 334890160364790302060735621729199964480057977539225173205956219308086252467158423520380
7354488, 632094372828241320671255647451901056399237760301503199444470380543753167478243100611604222284853, 54758
0618792250241258969096450342671069735142431883586773112380708321548837820284372036217092 76]

MMI = lambda A, n,s=1,t=0,N=0: (n < 2 and t%N or MMI(n, A%n, t, s-A//n*t, N or n),-1)[n<1] #逆元计算
a=(output[2]-output[1])*MMI((output[1]-output[0]),n)%n
ani=MMI(a,n)
b=(output[1]-a*output[0])%n
seed = (ani*(output[0]-b))%n
plaintext=seed
print(long_to_bytes(plaintext))
```

```
答案Spirit{Gr3at__J0b!_You_can_be___better!}
```

lcg-5

```
from Crypto.Util.number import *
flag = b'Spirit{**************************************}'

plaintext = bytes_to_long(flag)
length = plaintext.bit_length()

a = getPrime(length)
b = getPrime(length)
n = getPrime(length)

seed = plaintext
output = []
for i in range(10):
    seed = (a*seed+b)%n
    output.append(seed)

print("output = ",output)
# output =  [999729798627251094776634495949897532313601207578712072142432577500384034155267358948713483029842799
7676238039214108, 494309297248802318427173909499347043027232767942422401675193010036204511537496049412480167539 3
555642497051610643836, 677461289424731964527257862476506387587664384941590397387253666264805166824088240564056 94
4822918859679763679550247 1, 933478045490146092605278525236230555584533515550188808784352532123869571668715125671
7815518958670595053951084051571, 261513694337567702734682104903329609507147660852337110290103844446431487754994 8
1071341149413012904584646118729427 6, 117554918585867226471822654462537012216155941365710385553213783773633413 68
4270703570318827255766779126300503071450 2, 775207027090567349080434475758908065323437567965756842802559987215 53
8764347630657561314768133022756271249080549234 5, 840295753260245169132773715474534079360664960287119061583766180
9359377788072256203797817090151599031273142680590748, 280244008191860459050259614611367009426260095202068718465 9
6053076951511205898169430513225030943635789167734140046 2, 562722631803576583728678902189114159639483587164592 56
8525224168002174026582617976842979264557678038063501411368798 2]
```

用公式4

解题代码

```
from Crypto.Util.number import *
def gcd(a,b):
    if(b==0):
        return a
    else:
        return gcd(b,a%b)
s =  [99972979862725109477663449594989753231360120757871207214243257750038403415526735894871348302984279976762380
39214108, 49430929724880231842717390949934704302723276794242240167519301003620451153749604941248016753935556424
97051610643836, 67746128942473196452725786247650638758766438494159039738725366626480516682408824056405694482291 8
8596797636795502471, 93347804549014609260527852523623055558453351555018880878435253212386957166871512567178155 18
95867059505395108405157 1, 26151369433756770273468210490332960950714766085233711029010384444643148775499481071341
14941301290458464611872942706, 11755491858586722647182265446253701221615594136571038555321378377363341368427070 3
57031882725576677912630050307145062, 77520702709056734908043447575890806532343756796575684280255998721553876434 7
63065756131476813302275627124908054923 5, 84029575326024516913277371547453407936066496028711906158376618093593 77
78807225620379781709015159903127314268059074 8, 28024400819186045905025961461136700942626009520206871846596053076
95151120589816943051322503094363578916773414004662, 56272263180357658372867890218911415963948358716459256852522 4
16800217402658261797684297926455767803806350141136879 82]
t = []
for i in range(9):
    t.append(s[i]-s[i-1])
all_n = []
for i in range(7):
    all_n.append(gcd((t[i+1]*t[i-1]-t[i]*t[i]), (t[i+2]*t[i]-t[i+1]*t[i+1])))

MMI = lambda A, n,s=1,t=0,N=0: (n < 2 and t%N or MMI(n, A%n, t, s-A//n*t, N or n),-1)[n<1] #逆元计算
for n in all_n:
    n=abs(n)
    if n==1:
        continue
    a=(s[2]-s[1])*MMI((s[1]-s[0]),n)%n
    ani=MMI(a,n)
    b=(s[1]-a*s[0])%n
    seed = (ani*(s[0]-b))%n
    plaintext=seed
    print(long_to_bytes(plaintext))
```

结果

```
b'\x00'
b'\xd6\xce\xc7P)\xf9l\xc7\xc8{h\xf0\xc0\xdcX\xd7q\xc7\xd4\xd1L;\xb7hh*\xc2\xb4\xa3,\xbb\xa1}&\x94Z\xda\x19\x95\x
bb\xa6\x93g\xec\xd7\xc6\x15c'
b'\xd6\xce\xc7P)\xf9l\xc7\xc8{h\xf0\xc0\xdcX\xd7q\xc7\xd4\xd1L;\xb7hh*\xc2\xb4\xa3,\xbb\xa1}&\x94Z\xda\x19\x95\x
bb\xa6\x93g\xec\xd7\xc6\x15c'
b'Spirit{final__lcg__is__co0m1ing__are_you_ready?}'
b'\xeb\xf7.\xdc!\xe5O\xb4\x11\xed\xf1\xa5\xca\xa1\x9b\x15\xda\x85\xc1\x19\xebGd\n\x93\x04\xb2PW\x9c\xc0^\x8c\xf8
\xdd5\xf9\xcf\x1c\xad\x8b\xdd\xde\x1a/\xa6H\x08'
b'\xeb\xf7.\xdc!\xe5O\xb4\x11\xed\xf1\xa5\xca\xa1\x9b\x15\xda\x85\xc1\x19\xebGd\n\x93\x04\xb2PW\x9c\xc0^\x8c\xf8
\xdd5\xf9\xcf\x1c\xad\x8b\xdd\xde\x1a/\xa6H\x08'
```

lcg-6

```python
from Crypto.Util.number import *
flag = b'Spirit{*****************}'

plaintext = bytes_to_long(flag)
length = plaintext.bit_length()

a = getPrime(length)
b = getPrime(length)
n = getPrime(length)

seed = plaintext

output = []
for i in range(10):
    seed = (seed*a+b)%n
    output.append(seed>>64)
print("a = ",a)
print("b = ",b)
print("n = ",n)
print("output = ",output)
# a =  731111971045863129770849213414583830513204814328949766909151
# b =  456671883153709362919394459405008275757410555181682705944711
# n =  666147691257100304060287710111266554526660232037647662561651
# output =  [169856191484105450834295420352739177466 12, 326337364730292929633260933269325851351 35645, 205318750003210974728532485148226386739 18, 375246131876483873243744876572242790 11, 215311540206999005197633236007747207471 79, 178501657845032628928005342845543968773 2, 158591141774827129543592855014508739398 95, 100775718999283950528060241333209735306 89, 301993916830192963982544016663384105617 14, 213036340140343587981005872366185799956 34]
```