CTF 学习笔记四——LCG 发生器

线性同余发生器（Linear congruential generator），简称 LCG，是一种能产生具有不连续计算的伪随机序列的分段线性方程的算法，它代表了最古老和最知名的伪随机序列生成器算法之一，其理论相对容易理解，并且易于实现和快速，特别是在可以通过存储位截断提供模运算的计算机硬件上。

它的计算公式如下：

$$x_{n+1} = (ax_n + b)mod(m)$$

X 表示伪随机数，a 表示乘数，b 表示增量，m 表示模数。

线性同余发生器的好处是，通过适当选择参数，区间长度可知且很长。虽然不是唯一标准，但是一般情况下太短的区间长度在伪随机数发生器中是一个致命的缺陷。

虽然 LCG 能够产生伪随机数，且可以通过正规的随机性测试，但它对参 a 和 b 的选择极为敏感。例如，a=1，c=1。产生一个简单的 m 进制计数器，它具有长的周期，但显然非随机。

在 CTF 中，LCG 是一种非常容易破解的加密题型。因为题目中的 a,b,m,x 之间是可以互相推导而求出来的。

计算公式如下：

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

图 1 推导表

下面是公式证明：公式 1:

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

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

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

公式 2

解线性方程组

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

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

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

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

公式 3

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

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

公式 4

设 $t_n = X_{n+1} - X_n$

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

$t_{n+1}t_{n-1} - t_nt_n = 0 \pmod m$

所以 m 必然是 $t_{n+1}t_{n-1} - t_nt_n$ 的约数，如果有多组 $t_{n+1}t_{n-1} - t_nt_n$ ，那么取它们的公约数就是 m 的值。

接下来以一道 CTF 例题来具体介绍一下 LCG：

题目代码：

```
import gmpy2
from Crypto.Util.number import bytes_to_long, long_to_bytes
from rsa.prime import getprime

from flag import flag
```

```python


class LCG:
    a: int
    b: int
    m: int
    seed: int

    def __init__(self, a, b, m, seed):
        self.a = a
        self.b = b
        self.m = m
        self.seed = seed

    def next(self):
        self.seed = (self.a * self.seed + self.b) % self.m
        return self.seed


if __name__ == '__main__':
    fp = open("task.enc", "w")
    p = getprime(1024)
    q = getprime(1024)
    e = 0x10001
    n = p * q
    c = gmpy2.powmod(bytes_to_long(flag), e, n)
    seed1 = getprime(1024)
    b1 = getprime(1024)
    fp.write("b1=" + str(b1) + "\n")
    m1 = getprime(1024)
    fp.write("m1=" + str(m1) + "\n")
    lcg1 = LCG(p, b1, m1, seed1)
    seed2 = lcg1.next()
    fp.write("x2=" + str(lcg1.next()) + "\n")
    a2 = getprime(1024)
    b2 = getprime(1024)
    m2 = getprime(1024)
    lcg2 = LCG(a2, b2, m2, seed2)
    l = []
    for index in range(10):
        l.append(lcg2.next())
    fp.write("l=" + str(l) + "\n")
    fp.write("n=" + str(n) + "\n")
    fp.write("c=" + str(long_to_bytes(c)) + "\n")
```

```
    fp.close()
```

task.enc 的内容为：

```
b1=168327605660256824635635188271572899956804840925312202580221501399784647285241395076250558129589399768761153453790937525893900711587717308822081984187884527005934040735671234712891322152612220403179050706452506432061976734391245087565799802818687118150739197496652738092261163072198427100186203109566995894609
m1=913410521000496044930575527737181050673405147039371462284950929956625205761841167672453273287166404748976521529845718104615790365151646581256067331678693952687226003231939427523820570135865106137213295049705529505673784600664092679468334975170299767917132587520373050758297375911079197354523091295948892514 31
x2=456623886742024327014755434367167367222224006681822054273247147278581736582765750454398585714016431310318325047915629092251092535320063781220715803955603710933887640780364029589282511096193700992800706880190079566837368044512752747561747702955798358715317913685242276670076037472486214617560623431620325867 79
l=[9560112272019225680539705668477100206978472429178474681432217800343899963802664792756877630518797572777307671748428411299989352698201101002393666420638417957101754960805686607698778167464537404523276929682633324005299282625369299355348784748568826545910714793059222590494406818767112851267367344092597844 5115, 4366088268132058104823808364232603521921292562417641408831055182278667058801031648074110355462726559606939790207286648278866774224980809662024171380475672977120971955353621562709370176492927394139419841322525996441719939697650863054039927987540349948370828819010898471617906584798961946175598282021731017892 6, 1427679566093202452064161158928999230814457765704269391900676505803040912081847583411351578958739967374230381806845031716649349990367200270830696536193705753470031813270402685191513107763477106811713088623202963341029164096655836162706442277367280910987130506219020129645787570435833887825271610294858163502 8, 1656477346757735469185227045897846922922114107480541657779953360004288657644156145782332129071250923172298263649212521203068271249143998182543134425922880938760750794826422749470533016814722563160143032981184374006235804961178121980075480227875734932444985746368828625266033534463589875744412949790040184308 96, 8661303862776149098910439334718813048555355288235618406391464387022357022526034852270251241338688810767889580272008411001097928036447413845457173573580759641533525756895374357996856247870248567858308593712250983376247575073774497456463304979848559284571904221181197403555607708503575085298912010550416345112 3, 9923389793921908961345272668602947253941508109233798662851313012734204881412768322940477663992670152219369762410103537793396211359485128400822289145046406397482640826132182420580676029838433996466418561662542634269337197880434540766088510751436773063199707508570827836070989800519166301082518240903405148493 6, 9331039691204466914367843263198077498671637319686264978874260605698431000858681898643688835785420350861738382375359323455377074326602169291786216655242146718150925218591822765701757605427301953997866346098885777588206477576428841294636728491965535333267036338610800084892658206320078576579720072912485477927 5, 157564153609551042719506732601987579724933674540772480184051843201991883408851249860667015011291503624496327932115129042157892048813663701523720730810
```

```
891983319569255352705331757416215827329946934230707745477391951877454230154308836642697509847111864609274101987417938812557546398392486294317997112011281195964, 13443820452075394086420597072424655768261523183056666836329132485475727588219729669032088061445113371438833589235128219362846522200182130537025702009602064080945551620910412776142799057325581168173615635236500736207107610391905172602214905307292159225883291084106617860927846236834254915103319393315296412516 7, 340290115566490977950597269556612225056161948665885820462005223043238224708980389552463189373793683304382762351390838299656923125674549190542493445914211681975369834359040349169406342807564681164136745916981643861535789579952337947374017160551052196050880734988252797295041736476100477385584756629750682775 25]
n=2385802178636520854609924938973258585396599031477952822178687722444994770980457952056361432197842628235529162172735432729086985071587094056475818685644408352404865787333133851292192646184908664683985078419644131252261253803467122567124777114924181887942453976292105341812174350169156230343547463785384630458415833899825863557799075602313633307528011878489595366036459092502493673239759331548937965984902279022607296188730185661510397567249652682803021345869622068113444949176607959300889207631134301063555481148074772017220751126413466987324203438111657567785380253049329547315606506163253071057739310842349801720407 1
c=b"m\xca\xee\xe1f6w\xbe\xc0\xe4]P\xd9\xf1\xd2^\xcf\xb9\xae'~|\xcb\xf6&\x0e\xc3\xa2\x9a\x0c\x81o\x05\xab\x04>`\xd6j\x82e`f\xf4\xf5U/\xd6D8)\x94\x1b\x92\xb2\xe4P\xf5i8\x9e\xa4\xa7\xae\x9f\xc1R\xa88\xd1\xf71V\xd3\xd1\xbb?\xcf\x17[\x98\x9e\xb98\xcb\x9di\x81\xea\x00V\x86\x1b\xc6\xb44\x17\xc7\xa1\x01\xff\xc0|\x0c\xda\xfb\xda\xc6z\xe4D<\xd5\xb9C\x10\xaf\x81c`\tX\x87\xa46\xbb\xd7!xXl\x88\xda\xab\xad\x0c5\x9be\x95\xbc\x8d \x99]\xf2\x8d\xa6\x88\x87a\xfc\xfb+ \x9c(\xfax\xfc\xa5\xbf\xce\xc2\xe8\xb5}\x8b\xf8\xd5\x0e0\xb3\xee\x81\xf5\xe1\xc0>\x8fN\x17%\xba8 g\xc3\n\xa67\x89\xd4\x96b%\x04_\x84\xae\xf8\x06U%pvCd\xf0e\x04HL\x85\xc4\xedW\xa90\xdeA\xbb;\xaf\xa1\x1dr\xcf%\xc0Q\xa7\xf0s\x98E\xf9\xbc/\xb0\x88\x9a\xaa\x7f\xa8\xc4\xc3\xe2X\xfc\xd9\x19\x9c\xed]\xec"
```

分析代码，发现这道题构造了两个 LCG 生成器，其中第二个生成器给出了生成器生成的十个数字，第一个生成器给出了 b 和 m 的值，以及生成器生成的一个数字。

所以，这道题的解题思路为：先根据十个数字推断出第二个生成器的初始参数，也就是 seed2，然后根据 seed2 的值，以及第一个生成器的相关参

数，b 和 m，来求出第一个生成器的 a 的值，于是就可以轻松破解这个 RSA 加密了。

首先根据公式 4，我们可以求出模数 m 的值，代码如下：

```
import gmpy2
s =  [956011227201922568053970566847710020697847242917847468143221780034389996380266479275687763051879757277730767174842841129998935269820110100239366642063841795710175496080568660769877816746453740452327692968263332400529928262536929935534878474856882654591071479305922259049440681876711285126736734409259784451 15, 436608826813205810482380836423260352192129256241764140883105518227866705880103164807411035546272655960693979020728664827886677422498080966202417138047567297712097195535362156270937017649292739413941984132252599644171993969765086305403992798754034994837082881901089847161790658479896194617559828202173101789 26, 142767956609320245206416115892899923081445776570426939190067650580304091208184758341135157895873996737423038180684503171664934999036720027083069653619370575347003181327040268519151310776347710681171308862320296334102916409665583616270644227736728091098713050621902012964578757043583388782527161029485816350 28, 165647734675773546918522704589784692292211410748054165777995336000428865764415614578233212907125092317229826364921252120306827124914399818254313442592288093876075079482642274947053301681472256316014303298118437400623580496117812198007548022787573493244498574636882862526603353446358987574441294979004018430896, 866130386277614909891043933471881304855535528823561840639146438702235702252603485227025124133868881076788958027200841100109792803644741384545717357358075964153352575689537435799685624787024856785830859371225098337624757507377449745646330497984855928457190422118119740355560770850357508529891201055041634511 23, 992338979392190896134527266860294725394150810923379866285131301273420488141276832294047766399267015221936976241010353779339621135948512840082228914504640639748264082613218242058067602983843399646641856166254263426933719788043454076608851075143677306319970750857082783607098980051916630108251824090340514849 36, 933103969120446691436784326319807749867163731968626497887426060569843100085868189864368883578542035086173838237535932345537707432660216929178621665524214671815092521859182276570175760542730195399786634609888577758820647757642884129463672849196553533326703633861080008489265820632007857657972007291248547792 75, 157564153609551042719506732601987579724933674540772480184051843201991883408851249860667015011291503624496327932115129042157892048813663701523720730810891983319569255352705331757416215827329946934230707745477391951877454230154308836642697509847111864609274101987417938812557546398392486294317997112011281195964, 13443820452075394086420597072424655768261523183056666836329132485475727588219729669032088061445113371438833589235128219362846522200182130537025702009602064080945551620910412776142799057325581168173615635236500736207107610391905172602214905307292159225883291084106617860927846236834254915103319393315296412516 7, 340290115566490977950 59
```

```
72695566122250561619486658858204620052230432382247089803895524631893737936833043827
62351390838299656923125674549190542493445914211681975369834359040349169406342807564
68116413674591698164386153578957995233794737401716055105219605088073498825279729504
17364761004773855847566297506827752 5]
t = []
for i in range(1,len(s)):
    t.append(s[i]-s[i-1])
m = abs(t[2]*t[0]-t[1]*t[1])
for i in range(2,len(t)-1):
    res = abs(t[i+1]*t[i-1]-t[i]*t[i])
    m = gmpy2.gcd(m,res)
print(m)
```

然后由于 $t_n = at_{n-1} (\bmod m)$

所以可以求出 a 的值，代码很简单，这边就略过不写了。

然后由公式 3：$b = X_{n+1} - aX_n \ (\bmod m)$

就可以求出 b 的值，代码略。

知道了 a，b，m 的值之后，就可以由公式 1：$X_n = a^{-1} (X_{n+1} - b) \ (\bmod m)$

求出 seed2 的值

然后来看第一个 LCG 生成器，我们知道该生成器产生的连续的两个数字，以及 b 和 m 的值，那么 a 的值很快就能求出来。$a = X_n^{-1} (X_{n+1} - b) \ (\bmod m)$

接着需要注意，这里的 **a** 的值不是唯一的，所有满足$a_k = a + k * m$，$\boldsymbol{k} \in Z$的数都符合题意。再观察代码，发现 **a** 是 **n** 的一个因数，所以可以通过代码求出它。

代码如下

```
import gmpy2
seed2 = 32780240664994253004367364516857043877769926854035013658034217807355178223117371311457434516573943032635688441806042019664910691316513533259711601368054286271938880665763628510820626195728400894683637765916704762715221804457451152212936661065932132007644946514824953747288424673017865920988600579773443161928
x2 = 456623886742024327014755434367167367222224006681822054273247147278581736582765750454398585714016431310318325047915629092251092535320063781220715803955603710933887640780364029589282511096193700992800706880190079566837368044512752747561747702955798358715317913685242276670076037472486214617560623431620325867 79
b1=1683276056602568246356351882715728999568048409253122025802215013997846472852413950762505581295893997687611534537909375258939007115877173088220819841878845270059340407356712347128913221526122204031790507064525064320619767343912450875657998028186871181507391974966527380922611630721984271001862031095669958946 09
m1=913410521000496044930575527737181050673405147039371462284950929956625205761841167672453273287166404748976521529845718104615790365151646581256067331678693952687226003231939427523820570135865106137213295049705529505673784600664092679468334975170299767917132587520373050758297375911079197354523091295948892514 31
a1 = ((x2-b1)*gmpy2.invert(seed2,m1))%m1
n=2385802178636520854609924938973258585396599031477952822178687722444994770980457952056361432197842628235529162172735432729086985071587094056475818685644408352404865787333133851292192646184908664683985078419644131252261253803467122567124777114924181887942453976292105341812174350169156230343547463785384630458415833899825863557799075602313633307528011878489595366036459092502493673239759331548937965984902279022607296188730185661510397567249652682803021345869622068113444949176607959300889207631134301063555481148074772017220751126413466987324203438111657567785380253049329547315606506163253071057739310842349801720407 1
for i in range(100000):
    res = gmpy2.gcd(a1+i*m1,n)
    if res!=1:
        print(res)
        break
```

最后就是常规的 **RSA** 操作了，代码如下：

```
from Crypto.Util.number import *
import gmpy2
e = 0x10001
n = 23858021786365208546099249389732585853965990314779528221786877224449947709804579520563614321978426282355291621727354327290869850715870940564758186856444083524048657873331338512921926461849086646839850784196441312522612538034671225671247771149241818879424539762921053418121743501691562303435474637853846304584158338998258635577 99
```

```
07560231363330752801187848959536603645909250249367323975933154893796598490227902260
72961887301856615103975672496526828030213458696220681134449491766079593008892076311
34301063555481148074772017220751126413466987324203438111657567785380253049329547315
6065061632530710577393108423498017204071
c=b"m\xca\xee\xe1f6w\xbe\xc0\xe4]P\xd9\xf1\xd2^\xcf\xb9\xae'~|\xcb\xf6&\x0e\xc3\xa2
\x9a\x0c\x81o\x05\xab\x04>`\xd6j\x82e`f\xf4\xf5U/\xd6D8)\x94\x1b\x92\xb2\xe4P\xf5i8
\x9e\xa4\xa7\xae\x9f\xc1R\xa88\xd1\xf71V\xd3\xd1\xbb?\xcf\x17[\x98\x9e\xb98\xcb\x9d
i\x81\xea\x00V\x86\x1b\xc6\xb44\x17\xc7\xa1\x01\xff\xc0|\x0c\xda\xfb\xda\xc6z\xe4D<
\xd5\xb9C\x10\xaf\x81c`\tX\x87\xa46\xbb\xd7!xXl\x88\xda\xab\xad\x0c5\x9be\x95\xbc\x
8d \x99]\xf2\x8d\xa6\x88\x87a\xfc\xfb+ \x9c(\xfax\xfc\xa5\xbf\xce\xc2\xe8\xb5}\x8b\
xf8\xd5\x0e0\xb3\xee\x81\xf5\xe1\xc0>\x8fN\x17%\xba8 g\xc3\n\xa67\x89\xd4\x96b%\x04
_\x84\xae\xf8\x06U%pvCd\xf0e\x04HL\x85\xc4\xedW\xa90\xdeA\xbb;\xaf\xa1\x1dr\xcf%\xc
0Q\xa7\xf0s\x98E\xf9\xbc/\xb0\x88\x9a\xaa\x7f\xa8\xc4\xc3\xe2X\xfc\xd9\x19\x9c\xed]
\xec"

p = 13668403254759426667437637193310907196839897756322686409754674900543193430721061
55238782013838959983640293647364949551240425378035393069488141417952407667495698327
07625737546223631740927581850956407272891728920346583685053624890058487157086502497
251968657166230528159789214311670053953830316464032953074363749
q = n//p
c = bytes_to_long(c)
n = p*q
d = gmpy2.invert(e,(p-1)*(q-1))
m = pow(c,d,n)
print(long_to_bytes(m))
```