Algoritmo de factorização de Lenstra ECM

Seja N composto com $(N, 6) = 1$.

Passo 1: Construir uma curva elíptica em $\mathbb{Z}_N$.

In [23]:
```
N=next_prime(210)*next_prime(91211)
R=IntegerModRing(N)
B=10000
```

In [24]:
```
N
```

Out[24]:
```
19249319
```

Construção da curva elíptica.

In [25]:
```
x=R.random_element()
y=R.random_element()
a=R.random_element()
b=y^2-x^3-a*x
while gcd(ZZ(4*a^3+27*b^2),N)!=1:
    x=R.random_element()
    y=R.random_element()
    a=R.random_element()
    b=y^2-x^3-a*x
    if gcd(ZZ(4*a^3+27*b^2),N) not in [1,N]:
        print(gcd(ZZ(4*a^3+27*b^2),N))
        break
```

In [26]:
```
gcd(ZZ(4*a^3+27*b^2),N)
```

Out[26]:
```
1
```

In [27]:
```
E=EllipticCurve(R,[a,b])
P=E([x,y])
```

In [28]:
```
E
```

Out[28]:
```
Elliptic Curve defined by y^2 = x^3 + 2474236*x + 5096819 over Ring of inte
gers modulo 19249319
```

In [29]:
```
P
```

Out[29]:
```
(804078 : 4457497 : 1)
```

Passo 2:

Para cada $1 \leq i \leq \pi(B)$ e p_i primo inferior a B, procura-se o maior inteiro a_i tal que $p_i^{a_i} \leq B$.

Inicia-se um ciclo $1 \leq j \leq a_i$, para cada i, calculando sucessivamente

$$P := p_i P$$

interrompendo quando alguma operação ilegal ocorrer. No caso, o $\lambda = \dfrac{y_1 - y_2}{x_1 - x_2}$ na definição de soma terá uma divisão por um divisor de 0.

Para $d = x_1 - x_2$ no passo ilegal, calcula-se $g = (N, d)$.

In [30]:
```
# procura-se o maior natural a1 tal que p1^a1<B
p1=2
a1=floor(log(B,p1)); a1
```

Out[30]: 13

In [31]:
```
B
```

Out[31]: 10000

In [32]:
```
P
```

Out[32]: (804078 : 4457497 : 1)

In [33]:
```
Paux=P
Pvelho=P
for j in range(1,a1+1):
    print(Paux, j)
    Paux=p1*Paux
```

```
(804078 : 4457497 : 1) 1
(9718072 : 17641292 : 1) 2
(18919397 : 8972585 : 1) 3
(2311334 : 15295124 : 1) 4
(2401710 : 9294208 : 1) 5
(975678 : 5463690 : 1) 6
(3580509 : 16947063 : 1) 7
(8642936 : 8515829 : 1) 8
(18816980 : 3607167 : 1) 9
(4945355 : 5500379 : 1) 10
(17508774 : 12589396 : 1) 11
(3216692 : 1566974 : 1) 12
(13767457 : 4581983 : 1) 13
```

In [34]:

```
# tenta-se com outro primo p2 inferior a B
p2=3
# procura-se o maior natural a2 tal que p2^a2<B
a2=floor(log(B,p2))
Paux=P
Pvelho=P
for j in range(1,a2+1):
    print(Paux,j)
    Paux=p2*Paux
```

```
(804078 : 4457497 : 1) 1
(2770351 : 6259490 : 1) 2
(10590123 : 669810 : 1) 3
(18006612 : 17290107 : 1) 4
(18843197 : 4849106 : 1) 5
(17714603 : 11951583 : 1) 6
(15127123 : 19174015 : 1) 7
(11881010 : 10259726 : 1) 8
```

In [35]:

```
# tenta-se com outro primo p2 inferior a B
p2=5
# procura-se o maior natural a2 tal que p2^a2<B
a2=floor(log(B,p2))
Paux=P
Pvelho=P
for j in range(1,a2+1):
    print(Paux,j)
    Paux=p2*Paux
```

```
(804078 : 4457497 : 1) 1
(4781266 : 11924905 : 1) 2
(1819707 : 18669369 : 1) 3
(14966685 : 16419061 : 1) 4
(14603476 : 3044309 : 1) 5
```

Percorrendo os primos p_i inferiores a B:

In [36]:
```
for p in primes(B):
    Paux=P # recuperar o ponto original
    a=floor(log(B,p))
    print ("p=", p," expoente max a=",a)
    for j in range(1,a+1):
        print(Paux,j)
        Paux=p*Paux
```

```
p= 2  expoente max a= 13
(804078 : 4457497 : 1) 1
(9718072 : 17641292 : 1) 2
(18919397 : 8972585 : 1) 3
(2311334 : 15295124 : 1) 4
(2401710 : 9294208 : 1) 5
(975678 : 5463690 : 1) 6
(3580509 : 16947063 : 1) 7
(8642936 : 8515829 : 1) 8
(18816980 : 3607167 : 1) 9
(4945355 : 5500379 : 1) 10
(17508774 : 12589396 : 1) 11
(3216692 : 1566974 : 1) 12
(13767457 : 4581983 : 1) 13
p= 3  expoente max a= 8
(804078 : 4457497 : 1) 1
(2770351 : 6259490 : 1) 2
(10590123 : 669810 : 1) 3
(18006612 : 17290107 : 1) 4
(18843197 : 4849106 : 1) 5
(17714603 : 11951583 : 1) 6
(15127123 : 19174015 : 1) 7
(11881010 : 10259726 : 1) 8
p= 5  expoente max a= 5
(804078 : 4457497 : 1) 1
(4781266 : 11924905 : 1) 2
(1819707 : 18669369 : 1) 3
(14966685 : 16419061 : 1) 4
(14603476 : 3044309 : 1) 5
p= 7  expoente max a= 4
(804078 : 4457497 : 1) 1
(7307320 : 7100286 : 1) 2
(15744329 : 4951048 : 1) 3
(10929173 : 3603851 : 1) 4
p= 11  expoente max a= 3
(804078 : 4457497 : 1) 1
(18872782 : 16603500 : 1) 2
(8497609 : 9188235 : 1) 3
p= 13  expoente max a= 3
(804078 : 4457497 : 1) 1
(8671752 : 8392664 : 1) 2
(5498139 : 1199980 : 1) 3
p= 17  expoente max a= 3
(804078 : 4457497 : 1) 1
(14571893 : 3963151 : 1) 2
(15663376 : 13083813 : 1) 3
p= 19  expoente max a= 3
(804078 : 4457497 : 1) 1
(10138433 : 3672659 : 1) 2
(3068225 : 5190739 : 1) 3
p= 23  expoente max a= 2
(804078 : 4457497 : 1) 1
(3790632 : 7940109 : 1) 2
p= 29  expoente max a= 2
(804078 : 4457497 : 1) 1
(5670279 : 6092415 : 1) 2
```

```
~/sage-9.2/local/lib/python3.8/site-packages/sage/structure/coerce_actions.
pyx in sage.structure.coerce_actions.IntegerMulAction._act_ (build/cythoniz
ed/sage/structure/coerce_actions.c:9632)()
    757
    758         if integer_check_long(nn, &n_long, &err) and not err:
--> 759             return fast_mul_long(a, n_long)
    760
    761         return fast_mul(a, nn)

~/sage-9.2/local/lib/python3.8/site-packages/sage/structure/coerce_actions.
pyx in sage.structure.coerce_actions.fast_mul_long (build/cythonized/sage/s
tructure/coerce_actions.c:11252)()
    919     n = n >> 1
    920     while n != 0:
--> 921         pow2a += pow2a
    922         if n & 1:
    923             sum += pow2a

~/sage-9.2/local/lib/python3.8/site-packages/sage/structure/element.pyx in
sage.structure.element.Element.__add__ (build/cythonized/sage/structure/ele
ment.c:10947)()
   1227         cdef int cl = classify_elements(left, right)
   1228         if HAVE_SAME_PARENT(cl):
-> 1229             return (<Element>left)._add_(right)
   1230         # Left and right are Sage elements => use coercion model
   1231         if BOTH_ARE_ELEMENT(cl):

~/sage-9.2/local/lib/python3.8/site-packages/sage/structure/element.pyx in
sage.structure.element.ModuleElement._add_ (build/cythonized/sage/structur
e/element.c:15558)()
   2363     Generic element of a module.
   2364     """
-> 2365     cpdef _add_(self, other):
   2366         """
   2367         Abstract addition method

~/sage-9.2/local/lib/python3.8/site-packages/sage/schemes/elliptic_curves/e
ll_point.py in _add_(self, right)
    667                     N1 = N.gcd(Integer(2*y1 + a1*x1 + a3))
    668                     N2 = N//N1
--> 669                     raise ZeroDivisionError("Inverse of %s does not
exist (characteristic = %s = %s*%s)" % (2*y1 + a1*x1 + a3, N, N1, N2))
    670                 else:
    671                     raise ZeroDivisionError("Inverse of %s does not
exist" % (2*y1 + a1*x1 + a3))

ZeroDivisionError: Inverse of 8641716 does not exist (characteristic = 1924
9319 = 211*91229)
```

In [37]:
```
gcd(N,8641716)
```

Out[37]:
```
211
```

In [38]:
```
N/211
```

Out[38]:
```
91229
```

In [39]:
```
211*91229 == N
```
Out[39]:
```
True
```
In [ ]:
```
```

In [17]:

```
L=10
B=10000
for iter in range(1,L):
    x=R.random_element()+y*iter
    y=R.random_element()+x*iter
    a=R.random_element()+b*iter
    #x=R(1211); y=R(71212); a=R(216)
    b=y^2-x^3-a*x
    while gcd(ZZ(4*a^3+27*b^2),N)!=1:
        x=R.random_element()*y
        y=R.random_element()*x
        a=R.random_element()*b
        b=y^2-x^3-a*x
        if gcd(ZZ(4*a^3+27*b^2),N) not in [1,N]:
            print (gcd(ZZ(4*a^3+27*b^2),N))
            break
    print( "iter=",iter, E,P)
    for p in primes(B):
        Paux=P # recuperar o ponto original
        a=floor(log(B,p))
        #print "p=", p," expoente max a=",a
        for j in range(1,a+1):
            Paux=p*Paux
            print (Paux)
```

```
iter= 1 Elliptic Curve defined by y^2 = x^3 + 14157076*x + 4634113 over Rin
g of integers modulo 19249319 (8954365 : 9221682 : 1)
(16853205 : 18232373 : 1)
(15615527 : 3470115 : 1)
(5655826 : 5862329 : 1)
(12153950 : 17013693 : 1)
(7677151 : 16000268 : 1)
(17537874 : 17491346 : 1)
(1309357 : 10031408 : 1)
(6865508 : 15321618 : 1)
(9681760 : 5896918 : 1)
(11441001 : 10433836 : 1)
(1336662 : 8266637 : 1)
(9789088 : 2767085 : 1)
(12827747 : 8282457 : 1)
(8423829 : 17995009 : 1)
(6884985 : 16883337 : 1)
(17846814 : 2078128 : 1)
(11368622 : 14895488 : 1)
(9761126 : 16859283 : 1)
(15539318 : 6878378 : 1)
(9411808 : 3051214 : 1)
(8193818 : 11792751 : 1)
(13453291 : 17500690 : 1)
(11626381 : 2716777 : 1)
(717626 : 18222117 : 1)
(10558555 : 16501861 : 1)
(12516062 : 13147426 : 1)
(10317398 : 3406122 : 1)
(6312021 : 8147048 : 1)
(15733230 : 16236715 : 1)
(5266166 : 15305212 : 1)
(2116540 : 1475148 : 1)
(14140205 : 4931966 : 1)
(17234110 : 15430513 : 1)
(1468851 : 4717817 : 1)
(9577332 : 16747444 : 1)
(682867 : 8701060 : 1)
(6415617 : 4897559 : 1)
(15168613 : 14720766 : 1)
(17820585 : 9597408 : 1)
(4146913 : 4786110 : 1)
(7107359 : 12138383 : 1)
(15591475 : 4636009 : 1)
(11240624 : 8453037 : 1)
(18296049 : 2275118 : 1)
(17041737 : 17068326 : 1)
(11197340 : 783230 : 1)
(623514 : 15404956 : 1)
(17140233 : 16569560 : 1)
(9000 : 10586308 : 1)
(17679121 : 2087992 : 1)
(15667499 : 19232161 : 1)
(1667834 : 5979053 : 1)
(13080402 : 9709744 : 1)
(2646594 : 17860773 : 1)
(14956020 : 13237691 : 1)
(10095594 : 5014904 : 1)
(10554701 : 392799 : 1)
(4217837 : 9716266 : 1)
(3180344 : 9710334 : 1)
```

```
(11818584 : 5451735 : 1)
(17627002 : 2780438 : 1)
(2261701 : 13539028 : 1)
(6145366 : 18862729 : 1)
(14438185 : 4407224 : 1)
(12379410 : 5857941 : 1)
(14001156 : 10193462 : 1)
(3941894 : 11155568 : 1)
(18511550 : 16742283 : 1)
```

In [18]:
```
n=3551; R=IntegerModRing(n)
```

In [19]:
```
a=9; b=1; E=EllipticCurve(R,[a,b]); E
```

Out[19]:
```
Elliptic Curve defined by y^2 = x^3 + 9*x + 1 over Ring of integers modulo
3551
```

In [20]:
```
P=E(0,1)
```

In [21]:
```
P
```

Out[21]:
```
(0 : 1 : 1)
```

In [22]:
```
2*P
```

Out[22]:
```
(908 : 3015 : 1)
```

In [23]:

```
2*2*P
```

In [ ]:

```
# uso do try: except

    for p in pis:
        Paux = P
        ai = floor(log(B, p))
        try:
            for j in range(ai):
                Paux *= p
        except ZeroDivisionError as zde:
            zde = str(zde)
            splited = zde.split(' ')
            for s in splited:
                if s[0].isdigit():
                    x1x2 = int(s)
                    break
            return "Fator: "+str(gcd(x1x2, n))
            break
```

In [ ]:

```
```