?GF

In [1]:
```
F3=GF(3); F9.<a>=GF(3^2);
```

In [2]:
```
list(F3)
```

Out[2]:
```
[0, 1, 2]
```

In [3]:
```
list(F9)
```

Out[3]:
```
[0, a, a + 1, 2*a + 1, 2, 2*a, 2*a + 2, a + 2, 1]
```

In [4]:
```
show(F3(1)+F3(5)); show(F3(1/2)); show(F3(2^10))
```

In [5]:
```
show(F9(1)+F9(5)); show(F9(1/2)); show(F9(2^10))
```

In [6]:
```
show(F9(a^9)); show(F9((1+a)^9)); show(F9(1/a))
```

In [7]:
```
show(F9(a*(a+2)))
```

In [8]:
```
show(F9(a^2));
```

In [9]:
```
show(F9(1+a-a^2))
```

In [10]:
```
F3X.<x>=PolynomialRing(GF(3)); P=x^4-x^3+x^2-x+1; P.is_irreducible()
```

Out[10]:
```
True
```

In [11]:
```
F81.<b>=GF(3^4,modulus=P); show(b^9); show(1/b)
```

In [12]:
```
x^9 % P
```

Out[12]:
```
2*x^3 + x^2 + 2*x + 1
```

In [13]:
```
xgcd(x,P)
```

Out[13]:
```
(1, 2*x^3 + x^2 + 2*x + 1, 1)
```

In [14]:
```
b.minpoly(x)
```

Out[14]:
```
x^4 + 2*x^3 + x^2 + 2*x + 1
```

In [15]:
```
a.minpoly(x)
```

Out[15]:
```
x^2 + 2*x + 2
```

In [16]:
```
Phi=cyclotomic_value(3^4-1,x); show(Phi)
```

In [17]:
```
P1=x^3+2*x^2+2*x+1; P1.factor()
```

Out[17]:
```
(x + 1) * (x + 2)^2
```

In [18]:
```
P2=x^3+2*x^2-x-1; P2.factor()
```

Out[18]:
```
x^3 + 2*x^2 + 2*x + 2
```

In [19]:
```
P2.is_irreducible()
```

Out[19]:
```
True
```

In [20]:
```
F27.<c>=GF(3^3,modulus=P2); F27X.<x>=PolynomialRing(F27)
```

In [21]:
```
P2.factor()
```

Out[21]:
```
x^3 + 2*x^2 + 2*x + 2
```

In [22]:
```
F27X(P2).factor()
```

Out[22]:
```
(x + 2*c) * (x + c^2 + 2*c) * (x + 2*c^2 + 2*c + 2)
```

In [23]:
```
x=F3X.gen(); P3=x^4+2*x^3+2*x^2+x+2; P3.factor()
```

Out[23]:
```
x^4 + 2*x^3 + 2*x^2 + x + 2
```

In [24]:
```
F9X.<x>=PolynomialRing(F9); F9X(P3).factor()
```

Out[24]:
```
(x^2 + x + a) * (x^2 + x + 2*a + 1)
```

In [25]:
```
F81X.<x>=PolynomialRing(F81); F81X(P3).factor()
```

Out[25]:
```
(x + b^3 + b^2 + 2) * (x + b^3 + 2*b^2 + 2*b + 1) * (x + 2*b^3 + b^2 + b)
* (x + 2*b^3 + 2*b^2 + 2)
```

In [26]:
```
Phi.factor() # P3 est un facteur de la décomposition mais pas P
```

Out[26]:
```
(x^4 + x + 2) * (x^4 + 2*x + 2) * (x^4 + x^3 + 2) * (x^4 + x^3 + x^2 + 2*
x + 2) * (x^4 + x^3 + 2*x^2 + 2*x + 2) * (x^4 + 2*x^3 + 2) * (x^4 + 2*x^3
+ x^2 + x + 2) * (x^4 + 2*x^3 + 2*x^2 + x + 2)
```

In [27]:
```
F9X(Phi).factor()
```

Out[27]:
```
(x^2 + x + a) * (x^2 + x + 2*a + 1) * (x^2 + 2*x + a) * (x^2 + 2*x + 2*a
+ 1) * (x^2 + a*x + 2*a) * (x^2 + a*x + 2*a + 1) * (x^2 + (a + 1)*x + a +
2) * (x^2 + (a + 1)*x + 2*a) * (x^2 + (a + 2)*x + a) * (x^2 + (a + 2)*x +
a + 2) * (x^2 + 2*a*x + 2*a) * (x^2 + 2*a*x + 2*a + 1) * (x^2 + (2*a + 1)
*x + a) * (x^2 + (2*a + 1)*x + a + 2) * (x^2 + (2*a + 2)*x + a + 2) * (x^
2 + (2*a + 2)*x + 2*a)
```

In [28]:
```
F27X(Phi).factor()
```

Out[28]:
```
(x^4 + x + 2) * (x^4 + 2*x + 2) * (x^4 + x^3 + 2) * (x^4 + x^3 + x^2 + 2*
x + 2) * (x^4 + x^3 + 2*x^2 + 2*x + 2) * (x^4 + 2*x^3 + 2) * (x^4 + 2*x^3
+ x^2 + x + 2) * (x^4 + 2*x^3 + 2*x^2 + x + 2)
```

In [29]:
```
F81X(Phi).factor()
```

Out[29]:
```
(x + b + 1) * (x + 2*b + 2) * (x + b^2 + 2) * (x + b^2 + b) * (x + b^2 +
b + 2) * (x + 2*b^2 + 1) * (x + 2*b^2 + 2*b) * (x + 2*b^2 + 2*b + 1) * (x
+ b^3 + 1) * (x + b^3 + 2*b) * (x + b^3 + 2*b + 2) * (x + b^3 + b^2 + 1)
* (x + b^3 + b^2 + 2) * (x + b^3 + b^2 + b + 1) * (x + b^3 + b^2 + b + 2)
* (x + b^3 + b^2 + 2*b) * (x + b^3 + b^2 + 2*b + 1) * (x + b^3 + 2*b^2 +
b + 1) * (x + b^3 + 2*b^2 + 2*b) * (x + b^3 + 2*b^2 + 2*b + 1) * (x + 2*b
^3 + 2) * (x + 2*b^3 + b) * (x + 2*b^3 + b + 1) * (x + 2*b^3 + b^2 + b) *
(x + 2*b^3 + b^2 + b + 2) * (x + 2*b^3 + b^2 + 2*b + 2) * (x + 2*b^3 + 2*
b^2 + 1) * (x + 2*b^3 + 2*b^2 + 2) * (x + 2*b^3 + 2*b^2 + b) * (x + 2*b^3
+ 2*b^2 + b + 2) * (x + 2*b^3 + 2*b^2 + 2*b + 1) * (x + 2*b^3 + 2*b^2 + 2
*b + 2)
```

In [30]:
```
R1=Integers(3^4-1); show(R1.unit_group_order());
[R1(3^(i+1)).multiplicative_order() for i in range(4)]
```

Out[30]:
```
[4, 2, 4, 1]
```

In [31]:
```
K.<beta>=GF(3^4,modulus=P3); x=F3X.gen();
mul(x-beta^(3^i) for i in range(4))
```

Out[31]:
```
x^4 + 2*x^3 + 2*x^2 + x + 2
```

In [32]:
```
n=3^4-1; m=n/gcd(2,n); show(m)
```

In [33]:
```
Phim=cyclotomic_value(m,x); facts=factor(Phim); show(facts)
```

In [34]:
```
[facts[i][0](beta^2) for i in range(4)]
```

Out[34]:
```
[0, beta^2, 2*beta^3 + beta^2 + 2*beta + 2, beta^3 + beta + 1]
```

In [35]:
```
[R1(2*3^i) for i in range(5)]
```

Out[35]:
```
[2, 6, 18, 54, 2]
```

In [36]:
```
mul(x-beta^(2*3^i) for i in range(4))
```

Out[36]:
```
x^4 + x^2 + x + 1
```

In [37]:
```
minpoly(beta^2)
```

Out[37]:
```
x^4 + x^2 + x + 1
```

In [38]:
```
m=5; n=2^m-1; delta=7; show(n);
```

In [39]:
```
F2X.<x>=PolynomialRing(GF(2)); show(factor(cyclotomic_value(n,x)));
```

In [40]:
```
mu=x^5 + x^2 + 1; K.<beta>=GF(2^5,modulus=mu);
```

In [41]:
```
lcm(minpoly(beta^(i+1)) for i in range(delta-1))
```

Out[41]:
```
x^15 + x^11 + x^10 + x^9 + x^8 + x^7 + x^5 + x^3 + x^2 + x + 1
```

In [42]:
```
g3=minpoly(beta^3); g5=minpoly(beta^5);
g=mu*g3*g5; show(g)
```

In [43]:
```
k=n-g.degree(); show(k);
```

In [44]:
```
t=floor((delta-1)/2); show(t);
```

In [45]:
```
M=F2X.random_element(k-1); show(M);
```

In [46]:
```
c=M*g; show(c);
```

In [47]:
```
e=x^(n-1)+1+x^8; R=c+e; show(R);
```

In [48]:
```
S=add(R(beta^(i+1))*x^i for i in range(2*t));
show(S);
```

In [49]:
```
def Berlekamp(S,t):
    v0=x^(2*t); v1=S; u0=0; u1=1;
    while v1.degree()>=t:
        q,v2=v0.quo_rem(v1); u2=u0-u1*q;
        v0=v1; v1=v2; u0=u1; u1=u2;
    return(u1)
```

In [50]:
```
L=Berlekamp(S,t); show(L);
```

In [51]:
```
L.roots()
```

Out[51]:
```
[(1, 1), (beta, 1), (beta^3 + beta^2 + beta + 1, 1)]
```

In [52]:
```
for i in range(n):
    if L(beta^(-i))==0:
        print(i)
```

```
0
8
30
```

In [53]:
```
show(e);
```

In [54]:
```
e=x^2+x^5+x^15; R=c+e;
S=add(R(beta^(i+1))*x^i for i in range(2*t));
L=Berlekamp(S,t); show(L);
```

In [55]:
```
for i in range(n):
    if L(beta^(-i))==0:
        print(i)
```

```
2
5
15
```

In [56]:
```
e=x^2+x^5+x^15+x^25; R=c+e;
S=add(R(beta^(i+1))*x^i for i in range(2*t));
L=Berlekamp(S,t); show(L);
```

In [57]:
```
for i in range(n):
    if L(beta^(-i))==0:
        print(i)
```

In [ ]:
```
```