MiniFFI

两解

题目描述 miniffi，Algebra是密码学中基础的数学原理，学会使用sagemath并尝试解决这一有限域有关的问题。
hint1：ffi代表有限域同构(Finite Field Isomorphism)，尝试去构造此同态映射的逆映射。

理论前提

有限域构造

此题中`R`是定义的一元多项式环，设为$F_q[x]$，并且$f(x), F(x)$是$F_q[x]$ 中的`128`次不可约多项式，因此$F_q[x]/(f(x))$以及$F_q[x]/(F(x))$是含有q^n个元素的有限域，并且它的每一个元素可以唯一地表示成

$$c_0 + c_1u + c_2u^2 + ... + c_{n-1}u^{n-1}$$

对$f(x)$，其中$c_i \in F_q, u = x + f(x)$。如果$f(x) = a_0 + a_1x + ... + a_nx^n$，则满足$a_0 + a_1u + a_nu^n = 0$。

实际上，对于实数域来说，假设`R[x]`，取一个二次不可约多项式$m(x) = x^2 + 1$，做商环$R[x]/(x^2 + 1)$，则他是域，并且与复数域同构，原因就是域中元素可以表示成$\{a + bu|a, b \in R\}$，而$u^2 + 1 = 0$。

设$q = p^n$，p为素数，则q元有限域一定存在，并且任意两个q元有限域都是同构的。

对于本题来说，$f(x), F(x)$均是F_q上的不可约多项式，那么$F_q[x]/f(x)$，$F_q[x]/F(x)$都是q^n个元素的有限域，并且同构，而满足$f(\phi(x))\ mod\ F(x) = 0$，实际上令$u = \phi(x)$，即为在有限域$F_q[x]/F(x)$中$f(u) = 0$，故$\phi(x)$为$F_q[x]/f(x) \to F_q[x]/F(x)$的同构映射。

加密函数：$C(x) = (2 * \phi(x) * r(\phi(x)) + m(\phi(x)))\ mod\ F(x)$

其中$m(x), r(x)$系数为0或1，只需要求得其逆映射使得$\psi(\phi(y)) \equiv y\ (mod\ F(y))$即$\phi(\psi(x)) \equiv x\ (mod\ f(x))$，可以得到$C(\psi(x)) = 2 * x * r(x) + m(x)\ (mod\ f(x))$。然后换到二元域上即可得到$m(x)$。

解题方法

上述问题转化为如何求取这个同态映射的逆映射：

总体上太概有三种方法，exp将给出利用线性代数解决方法，其余方法请自行尝试。

1、在有限域$F_q[x]/(f(x))$中求得$F(y)$的解，需要注意的是，因为会有n个不同的根，需要找到与上述映射相对应的逆映射。

2、在有限域$F_q[x]/(f(x))$中求得$\phi(y) - x$的根。

3、使用线性代数的方法。

方法三：

不妨设$\psi(x) = c_0 + c_1x + ... + c_{n-1}x^{n-1}$，我们的目标是构造$\psi(\phi(y)) \equiv y\ (mod\ F(y))$。

$$\sum_{i=0}^{n-1} c_i \phi(y)^i \equiv y \ (mod\ F(y))$$

$$\sum_{i=0}^{n-1} c_i \phi(y)^i = \sum_{i=0}^{n-1} c_i \sum_{j=0}^{n-1} a_{ij} y^j = \sum_{j=0}^{n-1} (\sum_{i=0}^{n-1} a_{ij} c_i) y^j \ (mod\ F(y))$$

因此只需要当$j = 1$时$\sum_{i=0}^{n-1} a_{ij}c_i = 1$，否则$\sum_{i=0}^{n-1} a_{ij}c_i = 0$即可。

exp

```
from  Crypto.Util.number import *
from Crypto.Cipher import AES

p = 41
n = 128
R = PolynomialRing(GF(p),'x')
x = R.gen()
f = x^128 + 25*x^8 + 36*x^7 + 8*x^6 + 11*x^5 + 14*x^4 + 17*x^3 + 6*x^2 + 23*x + 4
F = x^128 + 4*x^59 + 8*x^58 + 14*x^57 + 5*x^56 + 17*x^55 + 13*x^54 + 15*x^53 + 17*x^52 + 31*x^50 + 18*x^49 + 5*x^48 + 3*x^47 + 3*x^46 + 35*x^45 + 9*x^44 + 38*x^43 + 14*x^42 + 37*x^41 + 16*x^40 + 32*x^39 + 6*x^38 + 18*x^37 + 27*x^36 + 39*x^35 + 25*x^34 + 37*x^32 + 2*x^31 + 6*x^30 + 18*x^29 + 5*x^28 + 25*x^27 + 28*x^25 + 14*x^24 + 32*x^23 + 14*x^22 + 17*x^21 + 12*x^20 + 36*x^19 + 25*x^18 + 36*x^17 + 34*x^16 + 27*x^15 + 14*x^14 + 19*x^13 + 14*x^12 + 14*x^11 + 15*x^10 + 20*x^9 + 24*x^8 + 18*x^7 + 5*x^6 + 19*x^5 + 34*x^4 + 40*x^3 + 32*x^2 + 10*x + 38
phi = 23*x^126 + 38*x^125 + 15*x^124 + 19*x^123 + 32*x^122 + 3*x^121 + 15*x^120 + 26*x^118 + 3*x^117 + 5*x^116 + 34*x^115 + 31*x^114 + 37*x^113 + 26*x^112 + 7*x^111 + 40*x^110 + 23*x^109 + 15*x^108 + 5*x^107 + 8*x^106 + 5*x^105 + 38*x^104 + 22*x^103 + 26*x^102 + 24*x^101 + 25*x^100 + 28*x^99 + 12*x^98 + 2*x^97 + 26*x^96 + 6*x^95 + 29*x^94 + 2*x^93 + 22*x^92 + 23*x^90 + 31*x^89 + 12*x^88 + 23*x^87 + 14*x^86 + 29*x^85 + 29*x^84 + 28*x^83 + 36*x^82 + 22*x^81 + 39*x^80 + 31*x^79 + 9*x^78 + 2*x^76 + 32*x^75 + 22*x^74 + 34*x^73 + 34*x^72 + 16*x^71 + 19*x^70 + 35*x^69 + 35*x^68 + 23*x^67 + x^66 + 5*x^65 + 28*x^64 + 22*x^63 + 23*x^62 + 3*x^61 + 21*x^59 + 36*x^58 + 25*x^57 + 28*x^56 + 30*x^55 + 12*x^54 + 32*x^53 + 31*x^52 + 9*x^51 + 38*x^50 + 38*x^49 + 2*x^48 + 18*x^47 + 12*x^46 + 9*x^45 + 21*x^44 + 10*x^43 + 6*x^42 + 39*x^41 + 13*x^40 + 29*x^39 + x^38 + 30*x^36 + 22*x^35 + 35*x^34 + 17*x^33 + 14*x^32 + 2*x^31 + 24*x^30 + 37*x^29 + 8*x^28 + 6*x^27 + 15*x^26 + 5*x^25 + 17*x^24 + 22*x^23 + 14*x^22 + 37*x^21 + 28*x^20 + 4*x^19 + 5*x^18 + 39*x^17 + 12*x^16 + 19*x^15 + 30*x^13 + 39*x^11 + 2*x^10 + 9*x^9 + 37*x^8 + 18*x^7 + 5*x^5 + 15*x^4 + 9*x^3 + 19*x^2 + 23*x + 39
assert f(phi(x))%F(x) == 0

polys = []
for i in range(n):
    polys.append(pow(phi(x),i,F(x)))

monomials = [x**i for i in range(n)]
B = Matrix(Zmod(p), n)

for ii in range(1,n):
```

```
    for jj in range(n):
        if monomials[jj] in polys[ii].monomials():
            B[ii, jj] = polys[ii].monomial_coefficient(monomials[jj])

t = Matrix(GF(p), 1,n)
t[0,1] = 1;B[0,0] = 1

psi = t*B**(-1)
psi = tuple(list(psi)[0])
psi = R(psi)

C=30*x^127 + 28*x^126 + 20*x^125 + 28*x^124 + x^123 + 16*x^122 + 23*x^121 + 34*x^120
+ 31*x^119 + 7*x^118 + 4*x^117 + 31*x^116 + 34*x^115 + 12*x^114 + 33*x^113 + x^112 +
4*x^111 + 17*x^110 + 7*x^109 + 17*x^108 + 16*x^107 + 9*x^106 + 35*x^105 + 7*x^104 +
20*x^103 + 24*x^102 + 20*x^101 + 17*x^100 + 13*x^99 + 39*x^98 + 16*x^97 + 23*x^96 +
36*x^95 + 25*x^94 + 12*x^93 + 40*x^92 + 19*x^91 + 36*x^90 + 17*x^89 + 40*x^88 +
23*x^87 + 29*x^86 + 29*x^85 + 13*x^84 + 5*x^83 + 31*x^82 + 14*x^81 + 23*x^80 + x^79
+ 20*x^78 + 36*x^77 + 7*x^76 + 34*x^75 + 11*x^74 + 23*x^73 + 11*x^72 + 15*x^71 +
35*x^70 + 33*x^69 + 20*x^68 + 28*x^67 + 35*x^66 + 33*x^65 + 37*x^64 + 33*x^63 +
33*x^62 + 34*x^61 + 5*x^60 + 8*x^59 + 23*x^58 + 10*x^57 + 23*x^56 + 26*x^55 +
18*x^54 + 39*x^53 + 9*x^52 + 30*x^51 + 27*x^50 + x^49 + 8*x^48 + 35*x^47 + 33*x^46 +
2*x^44 + 12*x^43 + 2*x^42 + 16*x^41 + 39*x^40 + 13*x^39 + 33*x^38 + 5*x^37 + 36*x^36
+ 26*x^35 + 34*x^34 + 34*x^33 + 16*x^32 + 29*x^31 + 32*x^30 + x^29 + 7*x^28 +
27*x^27 + 22*x^26 + 6*x^25 + 11*x^24 + 36*x^23 + 18*x^22 + 19*x^21 + 6*x^20 +
14*x^19 + 34*x^18 + 15*x^17 + 4*x^15 + 4*x^14 + 6*x^13 + 29*x^12 + 38*x^11 + 37*x^9
+ 36*x^8 + 8*x^7 + 5*x^6 + 26*x^4 + 14*x^3 + 15*x^2 + 35*x + 5
cipher=b'\x1eO\xfe\xdd\xeeV\x04\xff\x9e\x9b\xadLbC]\x08\xb1\xf3e\x11U\xa4\xe7\xe2\x1
bO\x84\xe52\xfd\xe0J&\xa3\xad8\x88\xbf{\x04&\xe87\xc6A62\xba'

c_1 = C(psi(x))
m_1 = c_1(x)%f(x)
m1 = m_1.change_ring(Zmod(2))

t1 = ''.join(str(i) for i in m1.coefficients(sparse=False))
if len(t1) != 128:
    t1 += '0'*(128-len(t1))

key = int('0b'+t1 , 2)

aes = AES.new(long_to_bytes(key), mode=1)
print(aes.decrypt(cipher))
```