Cryptography Assignment - 12

Pramod Aravind Byakod

Exercise 7.30

Alice and Bob decide to communicate using NTRU Encrypt with parameters (N, p, q) = (7, 3, 29). Alice's public key is

h(x) = 3 + 14X – 4X^2 + 13X^3 – 6X^4 + 2X^5 + 7X^6.

Bob sends Alice the plaintext message m(x) = 1 + X – X^2 – X^3 – X^6 using the random element r(x) = –1 + X^2 – X^5 + X^6.

(a) What ciphertext does Bob send to Alice?

(b) Alice's private key is f(x) = –1 + X – X^2 + X^4 + X^6 and F3(x) = 1 + X + X^2 + X^4 + X^5 – X^6. Check your answer in (a) by using f and F3 to decrypt the message.

Solution:

```
def make_lift(a_x,q):
    a_x_l=list(reversed(a_x.list()))
    a_x_l_new=[]
    for a_x_co in a_x_l:
        a_x_co=a_x_co%q
        if a_x_co>-q/2 and a_x_co<q/2:
            a_x_l_new.append(a_x_co)
        else:
            a_x_l_new.append(a_x_co-q)
    a_x_new=0
    for i in range(len(a_x_l)):
        a_x_new=a_x_l_new[i]*y^(6-i)+a_x_new
    return a_x_new
N=7
p=3
q=29
Ro.<y>=ZZ[]
Rop.<yp>=Integers(p)[]
Roq.<yq>=Integers(q)[]
R.<x>=QuotientRing(Ro,y^N-1)
Rp.<xp>=QuotientRing(Rop,yp^N-1)
Rq.<xq>=QuotientRing(Roq,yq^N-1)
h_x=3+14*y-4*y^2+13*y^3-6*y^4+2*y^5+7*y^6
m_x=1+y-y^2-y^3-y^6
r_x=-1+y^2-y^5+y^6
e_x=Rq(p*r_x*h_x+m_x)
print 'e_x:',e_x
f_x=-1+y-y^2+y^4+y^6
a_x=Rq(f_x*Ro(lift(e_x)))
print 'f_x*e_x: ',a_x
a_x=Ro(lift(a_x))
a_x_cl=make_lift(a_x,q)
print 'a_x_cl:',a_x_cl
F3_x=1+y+y^2+y^4+y^5-y^6
m_x=Rp(F3_x*a_x_cl)
print 'F3_x*a_x:',m_x
m_x=Ro(lift(m_x))
print 'm_x:',make_lift(m_x,p)
```

Output:

```
e_x: 14*xq^6 + 16*xq^5 + 20*xq^4 + 7*xq^3 + 19*xq^2 + 16*xq + 23
f_x*e_x:  24*xq^6 + 27*xq^5 + 7*xq^4 + xq^3 + 26*xq^2 + 3*xq + 27
a_x_cl: -5*y^6 - 2*y^5 + 7*y^4 + y^3 - 3*y^2 + 3*y - 2
F3_x*a_x: 2*xp^6 + 2*xp^3 + 2*xp^2 + xp + 1
m_x: -y^6 - y^3 - y^2 + y + 1
```

Ciphertext is: 14*x^6 + 16*x^5 + 20*x^4 + 7*x^3 + 19*x^2 + 16*x + 23
Message after decrypting is m_x: -x^6 - x^3 - x^2 + x + 1, it is the same as the plaintext, so verified.

Exercise 7.45
Apply Gauss's lattice reduction algorithm (Proposition 6.63) to solve SVP for the following two-dimensional lattices having the indicated basis vectors. How many steps do the algorithm take?
(a) v1 = (120670, 110521) and v2 = (323572, 296358).
(b) v1 = (174748650, 45604569) and v2 = (35462559, 9254748).
(c) v1 = (725734520, 613807887) and v2 = (3433061338, 2903596381).

Solution:
Program:

```
def reduction(v1,v2):
    steps = 0
    while True:
        steps = steps+1
        if v2.norm() < v1.norm():
            temp = v1
            v1 = v2
            v2 = temp
            #v1, v2 = v2, v1 # swap step
        m = round((v1 * v2) / (v1 * v1))
        if m == 0:
            print("Vector v1: "+str(v1))
            print("Vector v2: "+str(v2))
            print("Number of steps taken: "+str(steps))
            break
        v2 = v2 - m*v1

v1_1 = vector([120670,110521])
v2_1 = vector([323572, 96358])
v1_2 = vector([174748650,45604569])
v2_2 = vector([35462559,9254748])
v1_3 = vector([725734520,613807887])
v2_3 = vector([3433061338,2903596381])
reduction(v1_1,v2_1)
reduction(v1_2,v2_2)
reduction(v1_3,v2_3)
```

Output:
(a)
Vector 1: (14, -47)
Vector 2: (-362, -131)
Number of steps taken: 6

(b)
Vector 1: (147, 330)
Vector 2: (690, -207)
Number of steps taken: 7

(c)
Vector 1: (4690, 126)
Vector 2: (2086, 4235)
Number of steps taken: 11