Estruturas Criptográficas - TP3-1

PG53721 - Carlos Machado

PG54249 - Tiago Oliveira

Enunciado - Resolver o HNP com soluções aproximadas dos problemas em reticulados

Nesta secção declaramos os parâmetros do problema HNP, parâmetros necessários para a solução e duas funções.

- msb: Aproximação dos bits mais significativos
- build_matrix: Construção da matriz a ser reduzida

In [69]:
```
# HNP:Recover a of Fp such that for many
# known random t of Fp we are given MSBl(a*t)for some l>0.

d = 16
p = next_prime(2^d)

d = ceil(log(p, 2))

Fp = GF(p)

k = ceil(sqrt(d)) + ceil(log(d, 2))
n = 2 * ceil(sqrt(d))
A = 1/(2**k)
B = p/(2**k)
M = 2**(k*20)

def msb2(y):
    while True:
        u = Fp.random_element()
        if 0 <= QQ(y) - B*QQ(u) < B:
            break
    return u

def msb(y):
    return Fp(floor(QQ(y)/B))

def build_matrix(ts,us):
    mat = []

    for i in range(0,n):
        vec = [0] * (n+2)
        vec[i] = p
        mat.append(vec)

    t_row = list(ts) + [A,0]
    last_row = [-B*QQ(u) for u in us] + [0,M]
    mat.append(t_row)
    mat.append(last_row)
    return matrix(QQ,mat)
```

Geramos o *alpha* a ser descoberto e os pares que serão usados para o descobrir

In [70]:
```
alpha = Fp.random_element()

ts = [Fp.random_element() for _ in range(0,n)]
us = [msb(alpha*t) for t in ts]
```

Por fim, geramos a matriz e aplicamos o algoritmo LLL para descobrir o *alpha*

In [71]:
```
mat = build_matrix(ts,us)

v = mat.LLL()

print(f'alpha:{alpha}, found:{Fp(v[-1][-2]*(2**k))}')
```

```
alpha:4345, found:4345
```

In [ ]: