TP3_Exercicio1

April 30, 2024

1 TP3 - Exercício 1

1.0.1 Autores

Afonso Ferreira - pg52669
Tiago Rodrigues - pg52705

1.0.2 Enunciado

No capítulo 5 dos apontamentos é descrito o chamado Hidden Number Problem. No capítulo 8 dos apontamentos é discutido um artigo de Nguyen & Shparlinsk , onde se propõem reduções do HNP a problemas difíceis em reticulados. Neste trabalho pretende-se construir, com a ajuda do Sagemath, uma implementação da solução discutida nos apontamentos para resolver o HNP com soluções aproximadas dos problemas em reticulados.

Criação das variáveis conforme o artigo

[ ]:
```
# Hidden Number Problem -> with this we can recover a secret value from a␣
 ↪public value

# p - A prime number for our field.
p = next_prime(2^16) # 65537  p do professor

Fp = GF(p) # Fq

alpha = Fp.random_element() # s do professor

# n - The number of bits in `p`.
n = ceil(log(p, 2))

# l - The number of significant bits revealed by the oracle.
l = ceil(sqrt(n)) + ceil(log(n, 2)) # k do professor
```

`MSB` - Função que devolve os bits mais significativos de x

[ ]:
```
def msb(x):
    """
    Returns the MSB of x based on the global paramters p, l.
    """
    while True:
```

```
        z = Fp.random_element()
        answer = x - z
        if Integer(answer) < Integer(p) / 2^(l+1):
            break
    return z
```

`Create_Oracle` - devolve um par (xi,ui) que corresponde ao oráculo, onde $u_i = msb_x(alpha * t \mod p)$ e $x_i = t$

[ ]:
```
def create_oracle(alpha):
    """
    Returns a randomized MSB oracle using the specified alpha value.
    """
    alpha = alpha
    def oracle():
        t = Fp.random_element() # xi do professor.
        return t, msb((alpha * t) % p) # par (xi,ui) que corresponde ao oraculo.
    return oracle
```

1.0.3 Reticulados

`Basis` - Função usada para a criação do reticulado/"base".

`Closest Vector` - Dado um vetor não reticulado, encontramos um vetor reticulado provavel de relevar o valor de alpha.

[ ]:
```
# d - Numero de queries ao oracle.
d = 2 * ceil(sqrt(n)) # sacado do pdf.


def basis(oracle_inputs):
    """
    Returns a basis using the HNP game parameters and inputs to our oracle
    """
    basis_vectors = []
    for i in range(d):
        p_vector = [0] * (d+1)
        p_vector[i] = p
        basis_vectors.append(p_vector)
    basis_vectors.append(list(oracle_inputs) + [QQ(1)/QQ(p)])
    return Matrix(QQ, basis_vectors)

def approximate_closest_vector(basis, v):
    """
    Returns an approximate CVP solution using Babai's nearest plane algorithm.
    """
    BL = basis.LLL()
    G, _ = BL.gram_schmidt()
```

```
    _, n = BL.dimensions()
    small = vector(ZZ, v)
    for i in reversed(range(n)):
        c = QQ(small * G[i]) / QQ(G[i] * G[i])
        c = c.round()
        small -= BL[i] * c
    return (v - small).coefficients()
```

1.0.4 Teste

Para este testo temos $p = 65537$, $d = 10$ com esta variável a depender de p, mas sendo assim a probabilidade de o algoritmo não fornecer a solução exata é inferior a $prob \approx 0.11$

[ ]:
```
# Hidden alpha scalar
alpha = Fp.random_element()
print("This is the original alpha: %d" % alpha)

# Create a MSB oracle using the secret alpha scalar
oracle = create_oracle(alpha)

# Using terminology from the paper: inputs = `t` values, answers = `a` values
inputs, answers = zip(*[ oracle() for _ in range(d) ])

# Build a basis using our oracle inputs
lattice = basis(inputs)
print("Solving CVP using lattice with basis:\n%s\n" % str(lattice))

# The non-lattice vector based on the oracle's answers
u = vector(ZZ, list(answers) + [0])
print("Vector of MSB oracle answers:\n%s\n" % str(u))

# Solve an approximate CVP to find a vector v which is likely to reveal alpha.
v = approximate_closest_vector(lattice, u)
print("Closest lattice vector:\n%s\n" % str(v))

# Confirm the recovered value of alpha matches the expected value of alpha.
recovered_alpha = (v[-1] * p) % p
assert recovered_alpha == alpha
print("Recovered alpha! Alpha is %d" % recovered_alpha)
```

```
This is the original alpha: 7709
Solving CVP using lattice with basis:
[  65537       0       0       0       0       0       0       0       0       0
0]
[      0   65537       0       0       0       0       0       0       0       0
0]
[      0       0   65537       0       0       0       0       0       0       0
0]
```

```
[      0       0       0   65537       0       0       0       0       0       0
0]
[      0       0       0       0   65537       0       0       0       0       0
0]
[      0       0       0       0       0   65537       0       0       0       0
0]
[      0       0       0       0       0       0   65537       0       0       0
0]
[      0       0       0       0       0       0       0   65537       0       0
0]
[      0       0       0       0       0       0       0       0   65537       0
0]
[      0       0       0       0       0       0       0       0       0   65537
0]
[   3296   39129   49008   60562   28832   42337    9073   45783   51367   26946
1/65537]

Vector of MSB oracle answers:
(46035, 44182, 47379, 52400, 29918, 1649, 15749, 24371, 13617, 39932, 0)

Closest lattice vector:
[46045, 44187, 47404, 52407, 29921, 1673, 15778, 24402, 13649, 39961,
7709/65537]

Recovered alpha! Alpha is 7709
```