Team Name: 我要吃親來

• b03902078 資工三 林書瑾

• b03902060 資工三 江昶翰

• b03902008 資工三 林煦恩

**1. Algebraic Attack**

首先，我們假設x=2 y=3 z=2

然後五個equation :

(1) x^2+y+z-2=0

(2) 2y^2+2z+x^2-x-3=0

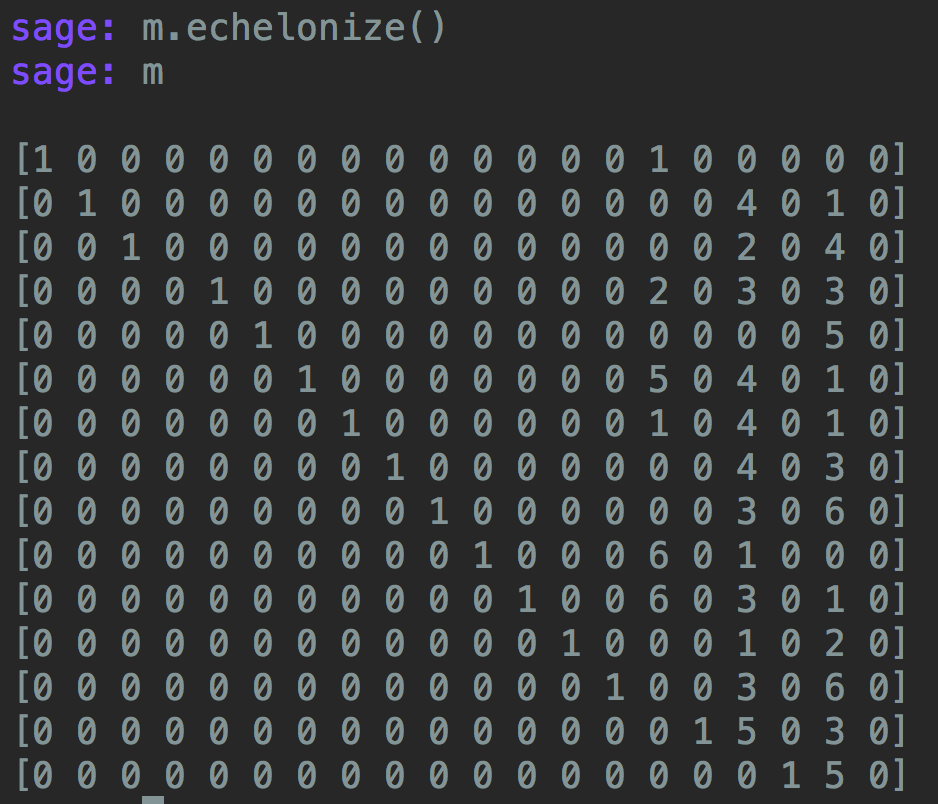
(3) y^2-2=0

(4) y^2-z^2-3y+2x=0

(5) x+y+z=0

接下來是XL algorithm，將5個equation分別乘以x,y,z，得到以下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x^2y | x^2z | y^2x | xyz | z^2x | y^2z | yz^2 | xy | xz | yz | x^3 | x^2 | x | y^3 | y^2 | y | z^3 | z^2 | z | 1 |
|  |  |  |  |  |  |  | 1 | 1 |  | 1 |  | -2 |  |  |  |  |  |  |  |
|  |  | 2 |  |  |  |  |  | 2 |  | 1 | -1 | -3 |  |  |  |  |  |  |  |
|  |  | 1 |  |  |  |  |  |  |  |  |  | -2 |  |  |  |  |  |  |  |
|  |  | 1 |  | -1 |  |  | -3 |  |  |  | 2 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | 1 | 1 |  |  | 1 |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  | 1 |  |  |  |  | 1 | -2 |  |  |  |  |
| 1 |  |  |  |  |  |  | -1 |  | 2 |  |  |  | 2 |  | -3 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  | -2 |  |  |  |  |
|  |  |  |  |  |  | -1 | 2 |  |  |  |  |  | 1 | -3 |  |  |  |  |  |
|  |  |  |  |  |  |  | 1 |  | 1 |  |  |  |  | 1 |  |  |  |  |  |
|  | 1 |  |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  | 1 | -2 |  |
|  | 1 |  |  |  | 2 |  |  | -1 |  |  |  |  |  |  |  |  | 2 | -3 |  |
|  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  | -2 |  |
|  |  |  |  |  | 1 |  |  | 2 | -3 |  |  |  |  |  |  | -1 |  |  |  |
|  |  |  |  |  |  |  |  | 1 | 1 |  |  |  |  |  |  |  | 1 |  |  |

Gaussian elimination (結果如右)

最後一個equation 可以得到z^2+5z=0，可以算出z=2或是0，recursive倒解回來可以得到兩組解(x,y,z)=(0,0,0,) or (2,3,2) （（0,0,0）那組不合）

**2. Linear Attack**

For this question, about 2^14 ~ 2^15 pairs of plain/cipher text is enough to recover part of the 5th subkey. (i.e. key[20:24], key[28:32])

**Output**

This is the key = 01111111110011010000000101101011

The attack result --------------------0001----1011

**Script**

#!/usr/bin/env python3

import random

KEY\_LENGTH = 4

PLAINTEXT\_LENGTH = 2

PAIR\_RANDOM\_TIMES = 2 \*\* 15

def hex2bin(hex\_value, fillzero = 4):

if isinstance(hex\_value, str):

hex\_value = int(hex\_value, 16)

return bin(hex\_value)[2:].zfill(fillzero)

def substi(text):

return ''.join([sbox(text[i: i+4]) for i in range(0, len(text), 4)])

def sbox(index):

box = 'E4D12FB83A6C5907'

return hex2bin(box[int(index, 2)])

def sboxRev(index):

box = 'E3481CAF7D96B205'

return hex2bin(box[int(index, 2)])

def permu(text):

box = [1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15, 4, 8, 12, 16]

result = [None for \_ in range(len(text))]

for char, new\_pos in zip(text, box):

result[new\_pos - 1] = char

return ''.join(result)

def genSubKey(key, n\_round):

start = 4 \* n\_round - 3 - 1

return key[start: start+16]

def strXor(a, \*args):

for b in args:

assert(len(a) == len(b))

a = ''.join(('0' if i == j else '1') for i, j in zip(a, b))

return a

def roundEncrypt(w, sub\_key):

u = strXor(w, sub\_key)

v = substi(u)

return permu(v)

def encrypt(plaintext, key):

w = plaintext

for n\_round in range(1, 4):

w = roundEncrypt(w, genSubKey(key, n\_round))

u = strXor(w, genSubKey(key, 4))

v = substi(u)

return strXor(v, genSubKey(key, 5))

# Generate random key

key = ''.join(random.choice('01') for \_ in range(KEY\_LENGTH \* 8))

# Generate plaintext/ciphertext pairs

dic = {}

for \_ in range(PAIR\_RANDOM\_TIMES):

plaintext = ''.join(random.choice('01') for \_ in range(PLAINTEXT\_LENGTH \* 8))

if plaintext not in dic:

dic[plaintext] = encrypt(plaintext, key)

# Attack

count = [0 for \_ in range(2 \*\* 8)]

for x, y in dic.items():

y = [y[i: i+4] for i in range(0, len(y), 4)]

y2, y4 = y[1], y[3]

x5, x7, x8 = x[4], x[6], x[7]

for i in range(2 \*\* 8):

L1, L2 = hex2bin(i>>4), hex2bin(i%16)

u2 = sboxRev(strXor(L1, y2))

u4 = sboxRev(strXor(L2, y4))

if '0' == strXor(x5, x7, x8, u2[1], u2[3], u4[1], u4[3]):

count[i] += 1

max\_value = -1

for i in range(2 \*\* 8):

count[i] = abs(count[i] - len(dic)//2)

if count[i] > max\_value:

max\_value, max\_key = count[i], i

# Check the answer

try:

assert(int(key[20:24] + key[28:32], 2) == max\_key)

except AssertionError:

print('Attack failed. Try to increase the pair of plain/cipher texts. (i.e. increase PAIR\_RANDOM\_TIMES)')

raise SystemExit

print("This is the key =", key)

print("The attack result", '-'\*20 + hex2bin(max\_key>>4) + '-'\*4 + hex2bin(max\_key%16))

**3. Differential Attack**

**Generate random key**

key = ''.join(random.choice('01') for \_ in range(KEY\_LENGTH \* 8))

**Generate plaintext/ciphertext pairs**

dic = {}

for \_ in range(PAIR\_RANDOM\_TIMES):

plaintext = ''.join(random.choice('01') for \_ in range(PLAINTEXT\_LENGTH \* 8))

if plaintext not in dic:

dic[plaintext] = encrypt(plaintext, key)

**Attack**