EE 793: Topics in Cryptology

Home Paper Assignment Problem:
Distribution of Linear Complexities of Maps at local points

Adhikari Tirthankar Taraknath 190070003
Badal Varshney, 19D070015

April 27, 2022

Faculty Mentor
Prof. Virendra Sule
Department of Electrical Engineering
Indian Institute of Technology Bombay

Contents

1 Introduction

The aim of this assignment is to compute and tabulate the linear complexities of maps $\mathbf{F} : \mathbf{F} \rightarrow \mathbf{F}$ at points y for some Finite fields **F**. We consider the maps $y = \zeta^x$ modp for a prime p and primitive elements ζ of the Field F_p.

2 Method

We first take the 13-bit prime 4999 as p and then, define the finite field in SageMath and find a primitive element. Then, we find the value of Zeta() by SageMath Code and then, find two more primitive element using the random sampling. After that, We compute the sequences for the respective maps and their linear complexities and finally, we plot the linear complexities graph for different value of ζ. All these computation are done in SageMath Jupyter Notebook which is attached in this report.

3 Selection of parameters and algorithm

Following tasks are to be performed and the data tabulated.

1. Choose maps **F**. It must map a finite field F_q of size q which is at aleast 12-bits up to a maximum of 16-bits. Results are likely to be better with larger number of bits. You can either choose F_p with prime p of n = 12 to 16 bit size or construct an irreducible polynomial of that much bit size to construct the field F_2^n .
2. Choose $k_0 = 2n$, $k_1 = 2n^2$, $k_2 = 2n^3$.
3. You may be required to choose at least three maps F with the same field for getting a variation in the computed data.
4. Use the Berlekamp-Massey (BM) function in SAGEmath which computes the LC of a sequence.
5. Algorithm:
   - Choose y in the field F_q.
   - Compute the three sequences S(F, y) upto the terms k_0, k_1 and k_2.
   - Find LC of each of these three sequences using the BM function.
   - Repeat step c) over 1000 randomly chosen y in the field.
   - Repeat step c) over multiple maps F that are chosen.
   - Draw the graphs (y, LC) of the data of Linear Complexities of these three sequences for each y and F. You may also draw the histograms of LC plots over the 1000 date points.

4 Analysis

We take the 13-bit prime(p) is 4999. Note that here, n = 13. We first define the finite field in SageMath and find a primitive element.

[2]:
```
import numpy as np
from sage.matrix.berlekamp_massey import berlekamp_massey as bm
import matplotlib.pyplot as plt
```

[3]:
```
p = 4999  # 13-bit prime
F = GF(p, 'zeta', modulus='primitive')
zeta = F.gen()   # value of zeta
k = 2^(int(np.log2(p)) + 1)
ks = [2*k]
zeta
```

[3]: 3

Here, We find the value of Zeta(ζ) which is equal to 3. Now, we find two more primitive elements using random sampling.

[4]:
```
num_elems = 0  # number of new primitive elements found
elems = [zeta]
while (num_elems<2):
    power_prime = np.random.randint(100, p-1)
    new_elem = zeta^power_prime    # generating a new element of the␣
 ↪multiplicative group
    if new_elem.is_primitive_root():  # checking if the new element is a␣
 ↪primitive root
        num_elems = num_elems + 1
        elems.append(new_elem)
elems
```

[4]: [3, 4137, 4364]

Therefore, the primitive elements chosen are $\zeta = 3, 4137, 4364$.

5 Computation

Now, We compute the sequences for the respective maps and their linear complexities.

[5]:
```
complexities = np.zeros((3, 1000)) # initializing the array to store linear␣
↪complexity values
points = np.random.randint(0, p, (3, 1000))
```

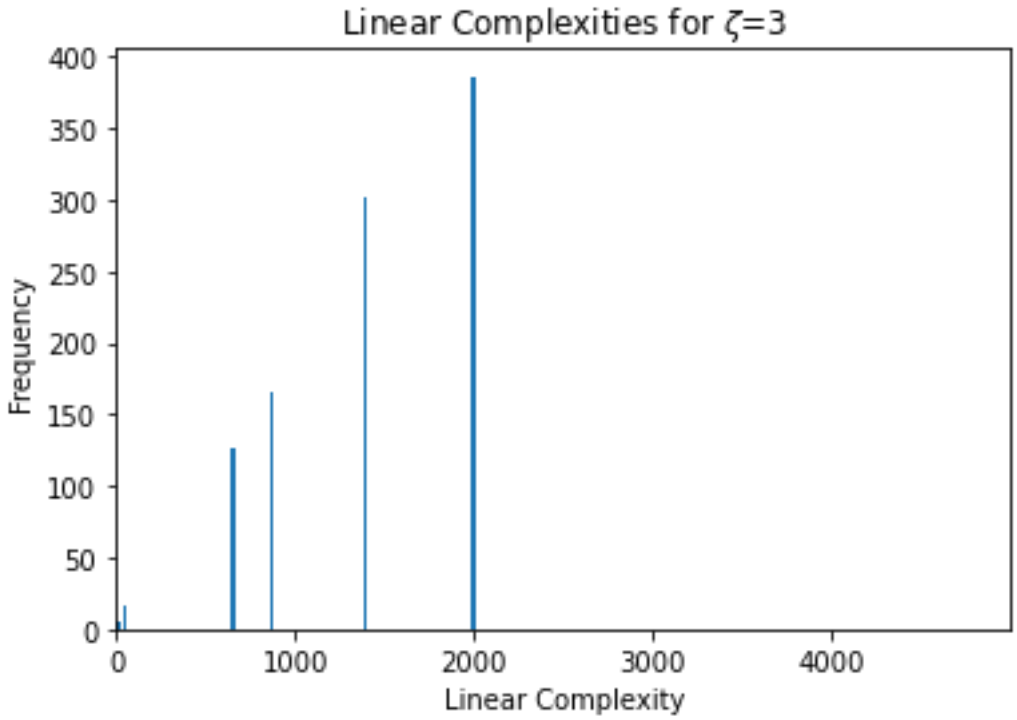
[6]:
```
for i_zeta in range(3):
    zeta_iteration = elems[i_zeta]
    for i_point in range(1000):
        seq = [0]*ks[0]
        point = points[i_zeta, i_point]
        seq[0] = zeta^point
        if (i_point%100 == 0):
            print(i_zeta, i_point)  # print the value of i_zeta and i_point
        for i in range(1, ks[0]):
            seq[i] = zeta_iteration^seq[i-1]
        minimal_poly = bm(seq)          # minimal polynomial
        lin_complexity = int(minimal_poly.degree())  # linear complexity
        complexities[i_zeta, i_point] = lin_complexity
```

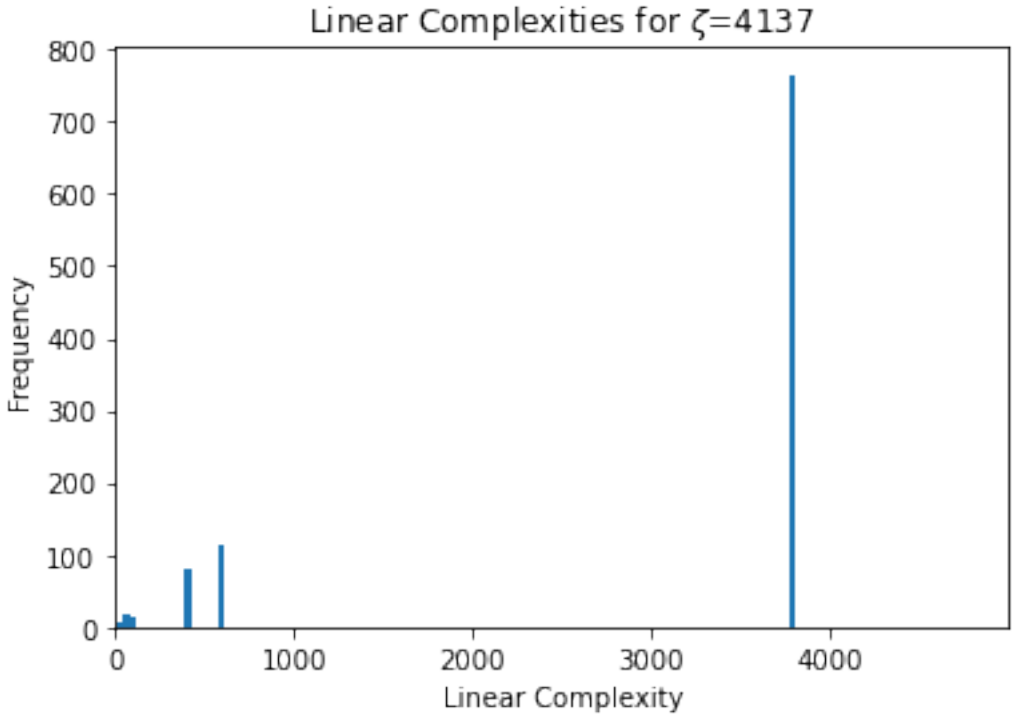
```
0 0
0 100
0 200
0 300
0 400
0 500
0 600
0 700
0 800
0 900
1 0
1 100
1 200
1 300
1 400
1 500
1 600
1 700
1 800
1 900
2 0
2 100
2 200
2 300
2 400
2 500
2 600
2 700
2 800
2 900
```

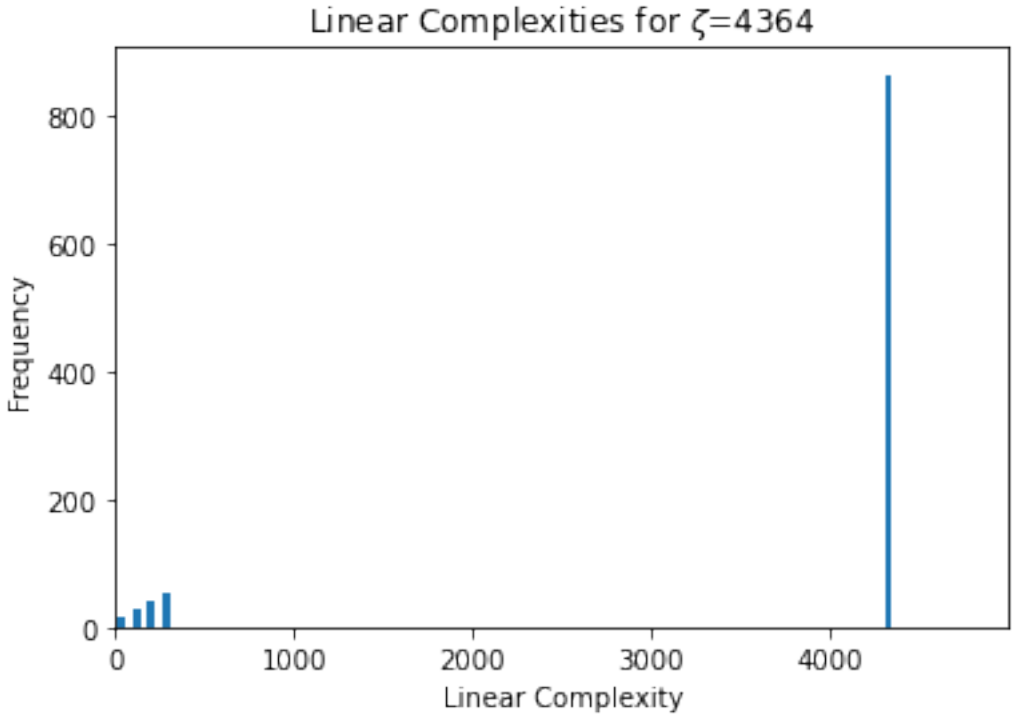
6 Outputs

Now, We plots the linear complexities.

[7]:
```
for i_zeta in range(3):
    plt.figure
    plt.hist(complexities[i_zeta], bins=100, rwidth = 4000)
    plt.xlabel('Linear Complexity')
    plt.ylabel('Frequency')
    plt.title(r"Linear Complexities for $\zeta$=" + str(elems[i_zeta]))
    plt.xlim([0, p])
    plt.show()
```




Linear Complexities for ζ=4137
Frequency
Linear Complexity
0
100
200
300
400
500
600
700
800
0
1000
2000
3000
4000


Linear Complexities for ζ=4364
Frequency
Linear Complexity
0
200
400
600
800
0
1000
2000
3000
4000

Here, We computed the linear complexities for the sequences of length $k_0 = 2^{(n+1)}$ (i.e., n+1 bits) due to memory constraints. Here, note that all the sequences are of twice length of the size of the field, and hence have atleast one periodic cycle.
Also note that, GOE is empty and all elements have their multiplicative inverse due to the nature of the map we assigned, therefore making all sequences periodic.

7 Observations

We observed that for each zeta(ζ), there are clusters of linear complexities at the smaller end and then a large chunk of sequences that have high linear complexity.

8 Conclusions

We find and plot linear complexities for the map y = ζ^x mod p for a prime p and also find three primitive elements ζi, i = 1, 2, 3 for 1000 randomly selected points for each ζi.

9 References

References

[1] The Sage Developers, SageMath, the Sage Mathematics Software System (Version 9.4), 2021. https://www.sagemath.org.

[2] https://doc.sagemath.org/html/en/reference/matrices/sage/matrix/berlekampmassey.html

[3] https://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/lfsr.html