Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

ЛАБОРАТОРНА РОБОТА №4

з дисципліни

«Криптографія»

на тему: «Побудова регістрів зсуву з лінійним зворотним зв’язком та дослідження їх властивостей**»**

Виконали:

студенти 3 курсу ФТІ

групи ФБ-72

Король Михайло, Степанець Антон

Перевірили:

Чорний О.

Савчук М. М.

Завадська Л. О.

Мета роботи:

Ознайомлення з принципами побудови регістрів зсуву з лінійним зворотним зв’язком; практичне освоєння їх програмної реалізації; дослідження властивостей лінійних рекурентних послідовностей та їх залежності від властивостей характеристичного полінома регістра.

Варіант:

P1(X)= X20 + X19 + X17 + X15 + X14 + X4 + 1 4

P2(X)= X24 + X21 + X15 + X14 + X11 + X9 + X8 + X5 + X4 + X3 + 1

Результати:

Перший поліном P1(X)= X20 + X19 + X17 + X15 + X14 + X4 + 1 4

Період лінійної рекурентної послідовності заданої поліномом: 41943

Розподіл монограм:

0	20951
1	20992

Розподіл біграм:

01	5154
11	5237
10	5216
00	5364

Автокореляція:

Зсув	Значення
1	20992
2	20992
3	20992
4	20992
5	20992
6	20992
7	20992
8	20992

9	20992
10	20992

Другий поліном P2(X)= X24 + X21 + X15 + X14 + X11 + X9 + X8 + X5 + X4 + X3 + 1

Період лінійної рекурентної послідовності заданої поліномом: 16777215

Розподіл монограм:

0	8388607
1	8388608

Розподіл біграм:

01	2099592
11	2096566
10	2095882
00	2096567

Автокореляція:

Зсув	Значення
1	8388608
2	8388608
3	8388608
4	8388608
5	8388608
6	8388608
7	8388608
8	8388608
9	8388608
10	8388608

Код програми:

```
import  threading

from time import time

PL1 = [1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,1] #x20 has to to be the Sum of others that meens that x20 has to zero

PL2 = [0,0,1,0,0,0,0,0,1,1,0,0,1,0,1,1,0,0,1,1,1,0,0,1]#used to be degree 25 but polinom meens that he coef near to hirest degree if the sum of priviost,so just dont using it


def Linear_shift_register1(moment):
  sum_of_moment = 0
  for i in range(len(PL1)):
    if PL1[i] == 1:
      sum_of_moment += moment[i]
  return  [sum_of_moment % 2] + moment[:-1]


def Linear_shift_register2(moment):
  sum_of_moment = 0
  for i in range(len(PL2)):
    if PL2[i] == 1:
      sum_of_moment += moment[i]
  return [sum_of_moment % 2] + moment[:-1]


def CreateImpulse(lenght):
  impulse = [0] *(lenght - 1)
  impulse.append(1)
  return impulse


def Calculating_period(polinom):
  sequence = "0"
  initial_state = CreateImpulse(len(polinom))
  count = 1
  if len(initial_state) > 20:
     new_state = Linear_shift_register2(initial_state)
     while  new_state != initial_state:
       sequence += str(new_state[0])
       new_state = Linear_shift_register2(new_state)
       count += 1
```

```
  else:
    new_state = Linear_shift_register1(initial_state)
    while new_state != initial_state:
      sequence += str(new_state[0])
      new_state = Linear_shift_register1(new_state)
      count += 1
  print(count)
  return  (polinom ,count,sequence)

def Polinom_description(polinom,count):
  print(f"{polinom},Current period: {count}, MAX- Period:{(2  len(polinom)) - 1  }")
  if (2  len(polinom)) - 1  == count:
    print("Primary polinom")
  elif ((2 ** len(polinom)) - 1 ) % count == 0:
    print("NEPRIVODIMUY polinom")
  else:
    print("POLINOM IS COULD BE BOTH VARs")



def sum_of_two_shifts(i,j):
  return (sequence[j] + sequence[(j + i) % period]) %2

def Auto(polinom,d,period,sequence):
  print(period,len(sequence))
  Dictionary = {i + 1: 0 for i in range(d)}
  for i in Dictionary.keys():
    for j in range(period):
      Dictionary[i] += (int(sequence[j]) + int(sequence[(i + j) % period])) % 2
  print(Dictionary)

def  Destribute(polinom,k,sequence):
  frequences = {}
  for i in range(0,len(sequence),k):
    if str(sequence[i:i + k]) in frequences:
      frequences[str(sequence[i:i + k])] += 1
    else:
```

```
        frequences[str(sequence[i:i + k])] = 1
    print(frequences)


start = time()
tpl = Calculating_period(PL1)
task1 =  threading.Thread(target = Destribute, args = (PL1,2,tpl[2]))
task1.start()
task2 = threading.Thread(target = Auto, args = (PL1,10,tpl[1],tpl[2]))
task2.start()
tpl2 = Calculating_period(PL2)
task3 =  threading.Thread(target = Destribute, args = (PL2,2,tpl2[2]))
task3.start()
task4 = threading.Thread(target = Auto, args = (PL2,10,tpl2[1],tpl2[2]))
task4.start()
task4.join()
end = time()
print(f"ВРЕМЯ РАБОТЫ:{end - start}")
#task5 = threading.Thread(target = Auto, args = (PL2,10,tpl2[1],tpl2[2]))
#task5.start()
```