Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

ЛАБОРАТОРНА РОБОТА №4

З дисципліни «Криптографія»

«Побудова регістрів зсуву з лінійним зворотним зв’язком та дослідження їх властивостей»

Виконали:

студенти 3 курсу ФТІ

групи ФБ-74
Опанасюк Олександр

Панчук Олександр

Перевірив:

Чорний 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.

Хід роботи:

0. Уважно прочитати методичні вказівки до виконання комп’ютерного практикуму.
1. Вибрати свій варіант завдання згідно зі списком. Варіанти завдань містяться у файлі Crypto_CP4 LFSR_Var.
2. За даними характеристичними многочленами p1(x), p2(x) скласти лінійні рекурентні співвідношення для ЛРЗ, що задаються цими характеристичними многочленами.
3. Написати програми роботи кожного з ЛРЗ L1 , L2 .
4. За допомогою цих програм згенерувати імпульсні функції для кожного з ЛРЗ і підрахувати їх періоди.
5. За отриманими результатами зробити висновки щодо влавстивостей кожного з характеристичних многочленів p1(x), p2(x): многочлен примітивний над F2 ; не примітивний, але може бути незвідним; звідний.
6. Для кожної з двох імпульсних функцій обчислити розподіл k-грам на періоді, k≤ni, де ni - степінь полінома fi(x), i=1,2 а також значення функції автокореляції A(d) для 0≤ d ≤ 10. За результатами зробити висновки..

Вхідні дані:

8	$P_1(X)= X^{20} + X^{19} + X^{17} + X^{15} + X^{14} + X^4 + 1$ $P_2(X)= X^{24} + X^{21} + X^{15} + X^{14} + X^{11} + X^9 + X^8 + X^5 + X^4 + X^3 + 1$
9	

Результати:

1 поліном: **Період 16777215**

Автокореляція:

0: 0	3: 8388608	6: 8388608	9: 8388608
1: 8388608	4: 8388608	7: 8388608	10: 8388608
2: 8388608	5: 8388608	8: 8388608	

Монограми	Біграми	3-грами	4-грами	5-грами
‘1 - 8388608	00 - 2096567	001 - 700928	0111- 262762	10010 - 105472
‘0 - 8388607	01 - 2099592	101 - 698880	0110 - 262680	11101 - 105472
	10 - 2095881	010 - 698880	1001 - 262614	01001 - 105216
	11 - 2096567	110 - 698880	0101 - 262397	00110 - 105216

		100 - 698880	1101 - 262345	00000 - 105011
		111 - 698880	0001 - 262281	11010 - 104960
		011 - 698880	0011 - 262225	10101 - 104960
		000 - 698197	0000 - 262210	11000 - 104960
			1111 - 262190	10000 - 104960
			1000 - 262121	11111 - 104960
			0100 - 262116	11110 - 104960
			1011 - 262031	10110 - 104960
			0010 - 261792	10111 - 104960
			1010 - 261715	11001 - 104960
			1100 - 261611	10011 - 104960
			1110 - 261213	10001 - 104960
				11100 - 104960
				00010 - 104704
				00001 - 104704
				01000 - 104704
				01010 - 104704
				00011 - 104704
				01100 - 104704
				01110 - 104704
				00100 - 104704
				00101 - 104704
				01101 - 104704
				00111 - 104704
				01011 - 104704
				11011 - 104448
				10100 - 104448
				01111 - 104192

2 поліном: **Період 41943**

Автокореляція:

0: 0	3: 20992	6: 20992	9: 20992
1: 20992	4: 20992	7: 20992	10: 20992
2: 20992	5: 20992	8: 20992	

Монограми	Біграми	3-грами	4-грами
1 - 20992	10 - 5363	101 - 1800	1001 - 701
0 - 20951	11 - 5237	001- 1792	1000 - 671
	00 - 5217	110 - 1776	1010 - 670
	01 - 5154	010 - 1768	1011 - 670
		111 - 1720	1110 - 665
		000 - 1717	0011 - 665
		011 - 1712	0110 - 664
		100 - 1696	0111 - 658
			0001 - 657
			0100 - 652
			0010 - 652
			1111 - 648
			1100 - 643
			1101 - 640
			0000 - 638
			0101 - 591

Код програми:

main.js

```
let mas = [1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0];
let mas1 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1];

let shift=[];
let i = 0;

while(mas!=mas1){
  i++;
  let test = mas[i] * mas1[i];
   if(test = 0){
      mas1.shift();
      mas1.push(0);
      shift.push(mas1.shift());
   }
   else if (test != 0){
     mas1.shift();
     mas1.push(1);
     shift.push(mas1.shift());
   }
}
```

lsfr.java

```
public class LFSR {

    public static void main(String[] args) {
        // initial fill
        boolean[] a = {
```

```
            false, true, false, false, false,
            false, true, false, true, true, false
        };
        int trials = Integer.parseInt(args[0]);    // number of steps
        int n = a.length;                          // length of register
        int TAP = 8;                               // tap position


        // Simulate operation of shift register.
        for (int t = 0; t < trials; t++) {

            // Simulate one shift-register step.
            boolean next = (a[n-1] ^ a[TAP]);  // Compute next bit.

            for (int i = n-1; i > 0; i--)
                a[i] = a[i-1];                  // Shift one position.

            a[0] = next;                        // Put next bit on right end.

            if (next) System.out.print("1");
            else      System.out.print("0");
        }
        System.out.println();
    }
}
```

quan.py

```
f = open('tmp.txt')
data = f.read().replace('\n', '')
f.close

mas = ['0', '1']
report = ''


#uno
a = []
tmp = 0
while 1:
    a.append(data[tmp:tmp+1])
    tmp = tmp + 1
    if (tmp > len(data)):
        break
report += 'Uno:\n'
for i in mas:
    find = i
    report += find + ' : ' + str(a.count(find)) + '\n'

#duo
a = []
tmp = 0
while 1:
    a.append(data[tmp:tmp+2])
    tmp = tmp + 2
    if (tmp > len(data)):
        break
print(a)
report += '\nDuo:\n'
for i in mas:
    for j in mas:
        find = i + j
        report += find + ' : ' + str(a.count(find)) + '\n'

#trio
a = []
tmp = 0
while 1:
    a.append(data[tmp:tmp+3])
    tmp = tmp + 3
    if (tmp > len(data)):
        break
report += '\nTrio:\n'
for i in mas:
    for j in mas:
        for k in mas:
            find = i + j + k
            report += find + ' : ' + str(a.count(find)) + '\n'

#quatro
```

```
a = []
tmp = 0
while 1:
    a.append(data[tmp:tmp+4])
    tmp = tmp + 4
    if (tmp > len(data)):
        break
report += '\nQuatro:\n'
for i in mas:
    for j in mas:
        for k in mas:
            for t in mas:
                find = i + j + k + t
                report += find + ' : ' + str(a.count(find)) + '\n'

#fivto
a = []
tmp = 0
while 1:
    a.append(data[tmp:tmp+5])
    tmp = tmp + 5
    if (tmp > len(data)):
        break
report += '\nFivto:\n'
for i in mas:
    for j in mas:
        for k in mas:
            for t in mas:
                for m in mas:
                    find = i + j + k + t + m
                    report += find + ' : ' + str(a.count(find)) + '\n'

f = open('report.txt', 'w')
f.write(report)
f.close
```

auto.js

```
let result = document.getElementById('result').innerHTML.split('');
console.log(result);
const per = 41943;

function autocorrelation(result, per){
  let d = 0;
  while(d!=11){
     let sum = 0;
     let i = 0;
     while(i != per){
       sum += (result[i] + result[i+d] % per) % 2;
       i ++;
      }
  console.log('Autocorrelation ' + 'd: ' + d + 'sum: ' + sum);
}
  d ++;
}
```

Висновок:

Під часа данного практикуму ми ознайомилися з принципами побудови регістрів зсуву з лінійним зворотнім зв’язком та дослідили властивості рекурентних послідовностей та їх залежностей від вдастивостей полніома регістра.