Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Лабораторна робота
із Криптографії №4

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

Виконали:
Каширін Євгеній ФБ - 74
Жолоб Тетяна ФБ - 72

Перевірено_____

Київ 2019

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4

Побудова регістрів зсуву з лінійним зворотним зв'язком та дослідження їх властивостей

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.

Порядок виконання 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 1 L , 2 L .
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 2 F ; не примітивний, але може бути незвідним; звідний.
6. Для кожної з двох імпульсних функцій обчислити розподіл k-грам на періоді, $k \le ni$, де ni - степінь полінома $fi(x)$, i=1,2 а також значення функції автокореляції $A(d)$ для $0 \le d \le 10$. За результатами зробити висновки.

Варіант 5

5	$P_1(X)= X^{20}+X^{17}+X^{15}+X^{14}+X^9+X^7+X^5+X^3+X^2+X+1$ $P_2(X)= X^{24}+X^{22}+X^{18}+X^{17}+X^{16}+X^{15}+X^{12}+X^{11}+X^9+X^4+X^2+X+1$

```
#include <iostream>
#include <cstring>
#include <math.h>
#include <cstdio>
#include <cstdlib>
#include <strings.h>

using namespace std;

void process_n_gram(char *mas, int &T, int grams, int *num, int cur_num)
{
	if (cur_num != grams)
```

```
                for (int l = 0; l < 2; l++)
                {
                        *(num + cur_num) = l;
                        process_n_gram(mas, T, grams, num, cur_num + 1);
                }
        else
        {
                int *n = new int[5];
                for (int l = 0; l < 5; l++)
                        n[l] = num[l];
                for (int l = -1; ++l < grams;)
                {
                        cout << num[l];
                        cout << " ";
                }
                long int counter = 0;
                for (int i = 0 - grams; (i = i + grams) < T-T % grams; )
                                                        {
                                                                int m[5];
                                                                int j;
                                                                for (int l = 0; l < grams; l++)
                                                                {
                                                                        m[l] = mas[i + l];
                                                                }
                                                                if (!memcmp(n, m, sizeof(int) * grams))
                                                                        counter++;
                                                        }
                cout << counter << endl;
                delete[] n;
        }
}

void    make_step(char* &mas, int &T, int size, int *def, int *var)
{
        int i;
        int suma;
        mas[T]= *def;
        i = -1;
        suma = 0;
        for (; size > ++i; )
        {
                suma = suma + (*(var + i) * *(def + i));
        }
        i = -1;
        for (; size - 1 > ++i; )
        {
                *(def + i) = *(def + i + 1);
        }
        *(def + size - 1) = suma % 2;
        T += 1;
}

void    create_t(int *var, int *def, int *dif, int size, int &T, char* &mas)
{
        int suma = 0;
        int malloc_size = 100000000;
        int i;
        mas = new char[malloc_size];
        make_step(mas, T, size, def, var);
        //sdvig
        //while (equal(def, def+size, dif) == 0)
        while (memcmp(def, dif, size * sizeof(int)))
                make_step(mas, T, size, def, var);
  cout<<"T = "<<T<<endl;
}

void    custom_bzero(void *mem, int size)
{
```

```
	unsigned char *m = (unsigned char *)mem;
	for (int i = 0; i < size; i++)
		m[i] = 0;
}

void	process_polinom(int *var, int size)
{
	int *def = new int[size];
	int *dif = new int[size];
	custom_bzero(def, size * sizeof(int));
	custom_bzero(dif, size *sizeof(int));
	def[size - 1] = 1;
	dif[size - 1] = 1;
	char *mas;
	int T = 0;
	create_t(var, def, dif, size, T, mas);
	int *mas_int = new int[5];
	for (int i = 2; i <= 5; i++)
	{
		cout << "processing " << i <<"-gram\n";
		process_n_gram(mas, T, i, mas_int, 0);
	}
	cout << "calculating correlation\n";
	int k = 0;
	int i = -1;
	for (int d = -1; 10 >= ++d; (k = 0) || (i = -1))
	{
		cout << d << " ";
		k = 0;
		i = -1;
		for (i = -1; ++i < T; )
		{
			int temp = mas[i];
			temp += *(mas + ((i + d) % T));
			k += temp % 2;
		}
		cout << k << endl;
	}
	delete[] mas_int;
	delete[] mas;
	delete[] def;
	delete[] dif;
}



int		main()
{
	cout << "processing 2nd polinom\n";
	int var2[20] = {1,1,1,1,0,1,0,1,0,1,0,0,0,0,0,1,1,0,1,0,0};
	process_polinom(var2 , 20);
	cout << "processing 1st polinom\n";
	int var[24] = {1,1,1,0,1,0,0,0,0,1,0,1,1,0,0,1,1,1,1,0,0,0,1,0};
	process_polinom((var), 24);
}
```

processing 2nd polinom
T = 1048575
processing 2-gram
0 0 131111
0 1 130644
1 0 131421
1 1 131111
processing 3-gram
0 0 0 43733
0 0 1 43648
0 1 0 43648
0 1 1 43904
1 0 0 43648
1 0 1 43904
1 1 0 43392
1 1 1 43648
processing 4-gram
0 0 0 0 16374
0 0 0 1 16417
0 0 1 0 16369
0 0 1 1 16470
0 1 0 0 16373
0 1 0 1 16347
0 1 1 0 16061
0 1 1 1 16234
1 0 0 0 16528
1 0 0 1 16403
1 0 1 0 16462
1 0 1 1 16580
1 1 0 0 16206
1 1 0 1 16461
1 1 1 0 16556
1 1 1 1 16302
processing 5-gram
0 0 0 0 0 6547
0 0 0 0 1 6560
0 0 0 1 0 6560
0 0 0 1 1 6496
0 0 1 0 0 6560
0 0 1 0 1 6368
0 0 1 1 0 6624
0 0 1 1 1 6560
0 1 0 0 0 6624
0 1 0 0 1 6688
0 1 0 1 0 6560
0 1 0 1 1 6496
0 1 1 0 0 6432
0 1 1 0 1 6496
0 1 1 1 0 6624
0 1 1 1 1 6560
1 0 0 0 0 6496
1 0 0 0 1 6560
1 0 0 1 0 6560
1 0 0 1 1 6496
1 0 1 0 0 6560
1 0 1 0 1 6624
1 0 1 1 0 6624
1 0 1 1 1 6560
1 1 0 0 0 6624
1 1 0 0 1 6432
1 1 0 1 0 6560
1 1 0 1 1 6496
1 1 1 0 0 6688
1 1 1 0 1 6496
1 1 1 1 0 6624
1 1 1 1 1 6560

calculating correlation
0 0
1 524288
2 524288
3 524288
4 524288
5 524288
6 524288
7 524288
8 524288
9 524288
10 524288
processing 1st polinom
T = 1118481
processing 2-gram
0 0 140126
0 1 139828
1 0 139969
1 1 139317
processing 3-gram
0 0 0 46659
0 0 1 46744
0 1 0 46536
0 1 1 46744
1 0 0 46744
1 0 1 46536
1 1 0 46744
1 1 1 46120
processing 4-gram
0 0 0 0 17591
0 0 0 1 17697
0 0 1 0 17680
0 0 1 1 17369
0 1 0 0 17497
0 1 0 1 17467
0 1 1 0 17471
0 1 1 1 17285
1 0 0 0 17482
1 0 0 1 17330
1 0 1 0 17565
1 0 1 1 17502
1 1 0 0 17219
1 1 0 1 17614
1 1 1 0 17374
1 1 1 1 17477
processing 5-gram
0 0 0 0 0 6984
0 0 0 0 1 6936
0 0 0 1 0 6960
0 0 0 1 1 7078
0 0 1 0 0 6949
0 0 1 0 1 6908
0 0 1 1 0 7062
0 0 1 1 1 6905
0 1 0 0 0 7136
0 1 0 0 1 6987
0 1 0 1 0 7061
0 1 0 1 1 6971
0 1 1 0 0 7093
0 1 1 0 1 6986
0 1 1 1 0 6954
0 1 1 1 1 6817
1 0 0 0 0 7054
1 0 0 0 1 7207
1 0 0 1 0 6972

1 0 0 1 1 6990

1 0 1 0 0 7019
1 0 1 0 1 6988
1 0 1 1 0 6957
1 0 1 1 1 7089
1 1 0 0 0 6944
1 1 0 0 1 6946
1 1 0 1 0 7017
1 1 0 1 1 7008
1 1 1 0 0 6855
1 1 1 0 1 6964
1 1 1 1 0 6980
1 1 1 1 1 6919
calculating correlation
0 0
1 559680
2 559392
3 559392
4 558432
5 560000
6 559488
7 559488
8 559392
9 558432
10 559488