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:

Катрич Дар’я ФБ – 72

Марісов Микола ФБ-72

Перевірено_____

Київ 2019

КРИПТОГРАФІЯ

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 $p_1(x)$, $p_2(x)$ скласти лінійні рекурентні співвідношення для ЛРЗ, що задаються цими характеристичними многочленами.

3. Написати програми роботи кожного з ЛРЗ L_1 , L_2 .
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 $p_1(x)$, $p_2(x)$: многочлен примітивний над F_2 ; не примітивний, але може бути незвідним; звідний.
6. Для кожної з двох імпульсних функцій обчислити розподіл k-грам на періоді, $k \leq n_i$, де n_i - степінь полінома $f_i(x)$, i=1,2 а також значення функції автокореляції $A(d)$ для $0 \leq d \leq 10$. За результатами зробити висновки.

Варіант 6;

L1 = x^22 + x^17 + x^15 + x^14 + x^13 + x^11 + x^8 + x^7 + x^6 + x^3 + x^2 + x + 1

the size of seq is : 584073; the size of M-seq should be ((2^22) - 1) = 4194303

L1 не примітивний
L1 звідний, бо : (2^22 -1) /584073 = 7.18113
d = 0, autocorr = 0
d = 1, autocorr = 292040
d = 2, autocorr = 292040
d = 3, autocorr = 292028
d = 4, autocorr = 292040
d = 5, autocorr = 292040
d = 6, autocorr = 292036
d = 7, autocorr = 292028
d = 8, autocorr = 292040
d = 9, autocorr = 292040
d = 10, autocorr = 292028

L2 = x^21 + x^16 + x^15 + x^14 + x^13 + x^10 + x^9 + x^7 + x^2 + x + 1
L2 примітивний
the size of seq is : 2097151; the size of M-seq should be ((2^21) - 1) = 2097151
p(x) for l2 is PRIMITIVNIJ
d = 0, autocorr = 0
d = 1, autocorr = 1048576
d = 2, autocorr = 1048576
d = 3, autocorr = 1048576
d = 4, autocorr = 1048576
d = 5, autocorr = 1048576
d = 6, autocorr = 1048576
d = 7, autocorr = 1048576
d = 8, autocorr = 1048576
d = 9, autocorr = 1048576
d = 10, autocorr = 1048576

Висновок:

В даному комп'ютерному практикумі було набуто навичок роботи з лінійними регістрами зсуву, а саме: їх програмна реалізація, дослідження властивостей характеристичного полінома регістра. Окрім цього було досліджено властивості лінійних рекурентних послідовностей

Програмний код:

```
#include <iostream>
#include <bitset>
#include <algorithm>
#include <vector>
#include <iterator>

//l1 = x^22 + x^17 + x^15 + x^14 + x^13 + x^11 + x^8 + x^7 + x^6 + x^3 + x^2 + x + 1
```

```
//l2 = x^21 + x^16 + x^15 + x^14 + x^13 + x^10 + x^9 + x^7 + x^2 + x   + 1

size_t l1(uint64_t impulse = 0b001)
{

   const uint64_t c = 0b11111111111111111111;
   uint64_t ih = impulse & c;
   impulse = ( impulse << 1 ) | ( ( (impulse >> 21) ^ (impulse >> 20) ^ (impulse >> 19) ^ (impulse >> 18) ^ (impulse >> 15) ^
                              (impulse >> 14) ^ (impulse >> 13) ^ (impulse >> 10) ^ (impulse >> 8 ) ^ (impulse >> 7 ) ^
                              (impulse >> 6 ) ^ (impulse >> 4 ) )
                              & (uint64_t)1);

   size_t count = 1;

   while( (impulse & c) != ih)
   {
      if(impulse == 0)
      {
         std::cout << "ERROR, sequense got nullified!" << std::endl;
         return 0 ;
      }
      impulse = ( impulse << 1 ) | ( ( (impulse >> 21) ^ (impulse >> 20) ^ (impulse >> 19) ^ (impulse >> 18) ^ (impulse >> 15) ^
                               (impulse >> 14) ^ (impulse >> 13) ^ (impulse >> 10) ^ (impulse >> 8 ) ^ (impulse >> 7 ) ^
                               (impulse >> 6 ) ^ (impulse >> 4 ) )
                               & (uint64_t)1);

   ++count;

   }
   return count;
}

size_t l2(uint64_t impulse = 0b001)
{

   const uint64_t c = 0b11111111111111111111;
   uint64_t ih = impulse & c;
   impulse = ( impulse << 1 ) | ( ( (impulse >> 20) ^ (impulse >> 19) ^ (impulse >> 18) ^ (impulse >> 13) ^
                              (impulse >> 11) ^ (impulse >> 10) ^ (impulse >> 7 ) ^ (impulse >>  6) ^
                              (impulse >> 5 ) ^ (impulse >> 4 ) )
                              & (uint64_t)1);



   size_t count = 1;


   while( (impulse & c) != ih)
   {
      if(impulse == 0)
      {
         std::cout << "ERROR, sequense got nullified!" << std::endl;
         return 0;
      }

   impulse = ( impulse << 1 ) | ( ( (impulse >> 20) ^ (impulse >> 19) ^ (impulse >> 18) ^ (impulse >> 13) ^
                              (impulse >> 11) ^ (impulse >> 10) ^ (impulse >> 7 ) ^ (impulse >>  6) ^
                              (impulse >> 5 ) ^ (impulse >> 4 ) )
```

```
                            & (uint64_t)1);
    ++count;

    }
    return count;
}

std::vector<size_t> runner()//NOT USED
{

    const uint64_t c = 0b11111111111111111111;//22
    uint64_t run = 2;

    std::vector<size_t> collection_of_t{};
    size_t ht = 0;
    ht = l1();
    collection_of_t.push_back(ht);
    std::cout << ht << std::endl;


    while (run < c)
    {
        ht = l1(run);
        ++run;

        auto result_of_search = std::find(std::begin(collection_of_t), std::end(collection_of_t), ht);
        if(result_of_search == collection_of_t.end())
        {
            collection_of_t.push_back(ht);
            std::cout << ht << std::endl;
        }
        if(run % 400000 == 0)
        {
            std::cout << run << std::endl;
        }

    }

    return collection_of_t;
}

void autocorr_l1()
{
    size_t T = l1();
    uint64_t impulse = 0b001;
    uint64_t autocorrelation = 0;

    for(size_t i = 0; i < 11; ++i)
    {
        std::cout << "d = " << i;
        impulse = 0b001;

        for(size_t pre = 0; pre < i; ++pre)      {
     impulse = ( impulse << 1 ) | ( ( (impulse >> 21) ^ (impulse >> 20) ^ (impulse >> 19) ^ (impulse >> 18) ^ (impulse >> 15) ^
                                (impulse >> 14) ^ (impulse >> 13) ^ (impulse >> 10) ^ (impulse >> 8 ) ^ (impulse >> 7 ) ^
                                (impulse >> 6 ) ^ (impulse >> 4 ) )
                                & (uint64_t)1);
        }
```

```
      for(size_t j = 0; j < T; ++j)
      {
         autocorrelation +=  ((impulse >> i) & (uint64_t)1 ) ^ ( impulse & (uint64_t)1);
         impulse = ( impulse << 1 ) | ( ( (impulse >> 21) ^ (impulse >> 20) ^ (impulse >> 19) ^ (impulse >> 18) ^ (impulse >> 15) ^
                                 (impulse >> 14) ^ (impulse >> 13) ^ (impulse >> 10) ^ (impulse >> 8 ) ^ (impulse >> 7 ) ^
                                 (impulse >> 6 ) ^ (impulse >> 4 ) )
                                 & (uint64_t)1);
      }
      std::cout << ", autocorr = " << autocorrelation << std::endl;
      autocorrelation = 0;
   }

}


void autocorr_l2()
{
   size_t T = l2();
   uint64_t impulse = 0b001;
   uint64_t autocorrelation = 0;

   for(size_t i = 0; i < 11; ++i)
   {
      std::cout << "d = " << i;
      impulse = 0b001;

      for(size_t pre = 0; pre < i; ++pre)
      {
     impulse = ( impulse << 1 ) | ( ( (impulse >> 20) ^ (impulse >> 19) ^ (impulse >> 18) ^ (impulse >> 13) ^
                              (impulse >> 11) ^ (impulse >> 10) ^ (impulse >> 7 ) ^ (impulse >>  6) ^
                              (impulse >> 5 ) ^ (impulse >> 4 ) )
                              & (uint64_t)1);

      }


      for(size_t j = 0; j < T; ++j)
      {
         autocorrelation +=  ((impulse >> i) & (uint64_t)1 ) ^ ( impulse & (uint64_t)1);

         impulse = ( impulse << 1 ) | ( ( (impulse >> 20) ^ (impulse >> 19) ^ (impulse >> 18) ^ (impulse >> 13) ^
                              (impulse >> 11) ^ (impulse >> 10) ^ (impulse >> 7 ) ^ (impulse >>  6) ^
                              (impulse >> 5 ) ^ (impulse >> 4 ) )
                              & (uint64_t)1);
      }
      std::cout << ", autocorr = " << autocorrelation << std::endl;
      autocorrelation = 0;
   }

}

void rozpodil_kgram_l1()
{
   std::vector<uint64_t> ks;
   ks.push_back(0);
   ks.push_back(0);
```

```
  }



int main()
{


    std::cout << "das ist ein lab" << std::endl;
    std::cout << " l1 started " << std::endl;
    std::cout << "l1 = x^22 + x^17 + x^15 + x^14 + x^13 + x^11 + x^8 + x^7 + x^6 + x^3 + x^2 + x + 1" << std::endl << std::endl;

    size_t res = l1();

    std::cout << "the size of seq is : " << res << "; the size of M-seq should be ((2^22) - 1) = 4194303" <<  std::endl;
    std::cout << "l1 NE_PRIMITIVNIJ" << std::endl;
    std::cout << "l1 ZVIDNIJ, bo :  (2^22 -1) /" << res << " = " << ((double)4194303)/((double)res) << std::endl;//2^22 - 1 не делит период l1

    autocorr_l1();

    std::cout << std::endl << "------------------------------------------------------------" << std::endl << std::endl;



    std::cout << " l2 started " << std::endl;
    std::cout << " l2 = x^21 + x^16 + x^15 + x^14 + x^13 + x^10 + x^9 + x^7 + x^2 + x   + 1" << std::endl;

    res = l2();//l2 is good m-sequence

    std::cout << "the size of seq is : " << res << "; the size of M-seq should be ((2^21) - 1) = 2097151" <<  std::endl;
    std::cout << "p(x) for l2 is PRIMITIVNIJ" << std::endl;

    autocorr_l2();




return 0;
}
```