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 генератора псевдовипадкових послідовностей на лінійних регістрах зсуву (генератора Джиффі) та його кореляційний криптоаналіз»

Виконали:

студенти 3 курсу ФТІ

групи ФБ-74

Постолюк Діана та Хацько Микита

Перевірили:

Чорний О.

Савчук М. М.

Завадська Л. О.

Мета роботи:

Ознайомлення з деякими принципами побудови криптосистем на лінійних регістрах зсуву; практичне освоєння програмної реалізації лінійних регістрів зсуву (ЛРЗ); ознайомлення з методом кореляційного аналізу криптосистем на прикладі генератора Джиффі.

Порядок виконання роботи:

0. Уважно прочитати методичні вказівки до виконання комп'ютерного практикуму.
1. За даними характеристичними многочленами написати програму роботи ЛРЗ L1 , L2 , L3 і побудованого на них генератора Джиффі.
2. За допомогою формул (4) – (6) при заданому α визначити кількість знаків вихідної послідовності N* , необхідну для знаходження вірного початкового заповнення, а також поріг C для регістрів L1 та L2 .
3. Організувати перебір всіх можливих початкових заповнень L1 і обчислення відповідних статистик R з використанням заданої послідовності (zi) , i = 0, N* –1.
4. Відбракувати випробувані варіанти за критерієм R > C і знайти всі кандидати на істинне початкове заповнення
L1 .
5. Аналогічним чином знайти кандидатів на початкове заповнення L2 .
6. Організувати перебір всіх початкових заповнень L3 та генерацію відповідних послідовностей (si) .
7. Відбракувати невірні початкові заповнення L3 за тактами, на яких xi ≠ yi , де (xi) , (yi) – послідовності, що генеруються регістрами L1 та L2 при знайдених початкових заповненнях.
8. Перевірити знайдені початкові заповнення ЛРЗ L1 , L2 , L3 шляхом співставлення згенерованої послідовності (zi) із заданою при i = 0, N –1.

Вихідні дані

Характеристичні многочлени:
– для L1 : p(x) = x30 ⊕ x6 ⊕ x4 ⊕ x⊕1, що відповідає співвідношенню між членами послідовності xi+30 = xi ⊕ xi+1 ⊕ xi+4 ⊕ xi+6 ;
– для L2 : p(x) = x31 ⊕ x3 ⊕1, відповідна рекурента: yi+31 = yi ⊕ yi+3 ;
– для L3 : p(x) = x32 ⊕ x7 ⊕ x5 ⊕ x3 ⊕ x2 ⊕ x ⊕1, відповідна рекурента: si+32 = si ⊕ si+1 ⊕ si+2 ⊕ si+3 ⊕ si+5 ⊕ si+7 .
Імовірність помилки першого роду α = 0,01.

Результати:

Знайдені критерії та довжини для регістрів:
LFSTR1: C = 73 N = 256

Варіант 11

```
11100100100010000100111011101001001110110111010110100001111000110111000001100010111101011010
11001010111110100000111011001011101000110010010111001011001010010111000101000101100101001110
10011001000010101101001100001100111100100010111110110001011111100011101110111000111111000111
11001001100000100111111110000000000110100010000010110101011011111010011111001000101111000100
01101111010110101111011000110110010111100110111010000111000000101100001010001001011011100010
11100101000011101000011111000111101000001100111100100011111110110011101000000001010011000101
10100001010011111100101110101100110101110000000010100111000100111100011100000110000110101010
01110111001111000010000101110110000100101010000001111100010010011000110111100001000100111101
01110001110111111111100100110111010010001100101000011001100110000101110111101111000001000010
01100001101100111001001010011100100010000011100110100101110101101011100110111110010101010011
01001010011101011110001000101010010101010011111010010111110011100001010100101101010111111111
00010100010110011111111001110101001011110010110001010100110110000000011000011111111011111011
11110110000101011010000100111011011100101010111100011000000010010111000011001011111110001000 1
11011100001101111010110100101101111110000101011011111100000001011100011111110011000110000101 0
10111110110001100010011101101000010100001111001011101110110100000010100100101111001000100001 0
10111100010011001100101101000101101101010111001011001110110010001000110111000011011011011000 0
10100000110010100010101111101100010010111000001111100101111101010110101001101011100100000110 0
01101110101110011100010101110101101011000110011111000010011110100100101011110010111010110010 0
10101101100110100000110000011011101011010001010100011001011111010011000011000110011101001011 1
10100100000100000110000100110010101110000101011010011011101011001111010001000110100011111000 1
11001100110101111010101010101010001100001111001001011001101011010000011010101101000000001101 1
00100110101010110100100011101101011100000111000000001001100001010000000111110111100001110011 1
11
```

Початкові значення послідовностей:

L1: 00111011101110100011101110111100

L2: 00001010110001100010000111010101

L3: 11110101000101110000111101000111

Код програми:

```
package crypto;

    import java.nio.file.Files;
    import java.nio.file.Paths;
    import java.time.Duration;
    import java.time.LocalDateTime;
    import java.util.Set;
    import java.util.stream.Collectors;
    import java.util.stream.IntStream;

public class Geffe {

   private static final int I30 = 1 << 30;
   // Same as (1 << 31) - 1
   private static final int I31 =
Integer.MAX_VALUE;
   private static final int C = 73;
   private static final int N = 8;
   private static int[] Z = new int[N];

   public static void main(String[] args) throws
Exception {
      String zString =
Files.readString(Paths.get("resources/lab4-11.txt"));

      for (int i = 0, tmp = 0; i < N; i++, tmp += 32) {
         Z[i] =
Integer.reverse(Integer.parseUnsignedInt(zString.sub
string(tmp, tmp + 32), 2));
      }

      LocalDateTime start = LocalDateTime.now();

      Set<Integer> xSeeds = IntStream.range(1, I30)
            .parallel()
            .filter(x -> countStatisticL1(x) < C)
            .boxed()
            .collect(Collectors.toSet());

      System.out.println(Duration.between(start,
LocalDateTime.now()));

      Set<Integer> ySeeds = IntStream.range(1, I31)
            .parallel()
```

```
            .filter(y -> countStatisticL2(y) < C)
            .boxed()
            .collect(Collectors.toSet());

        System.out.println(Duration.between(start,
LocalDateTime.now()));

        for (final Integer potentialX : xSeeds) {
            for (final Integer potentialY : ySeeds) {
                IntStream.range(Integer.MIN_VALUE,
Integer.MAX_VALUE)
                        .parallel()
                        .filter(s -> {
                            int x = potentialX;
                            // fill 30-th and 31-th bits
                            x = (((x ^ (x >>> 1) ^ (x >>> 4) ^ (x
>>> 6)) & 3) << 30) | x;
                            int y = potentialY;
                            // fill 31-th bit
                            y = (((y ^ (y >>> 3)) & 1) << 31) |
y;
                            if ((s & x ^ (~s & y)) != Z[0]) {
                                return false;
                            }
                            // ordered
                            for (int j = 1; j < N; j++) {
                                // shift all registers on 32 bits
(integer size)
                                for (int k = 0, bit; k <
Integer.SIZE; k++) {
                                    bit = 1 & ((x >>> 2) ^ (x >>>
3) ^ (x >>> 6) ^ (x >>> 8));
                                    x >>>= 1;
                                    x |= bit << 31;
                                    bit = 1 & ((y >>> 1) ^ (y >>>
4));
                                    y >>>= 1;
                                    y |= bit << 31;
                                    bit = 1 & (s ^ (s >>> 1) ^ (s
>>> 2) ^ (s >>> 3) ^ (s >>> 5) ^ (s >>> 7));
                                    s >>>= 1;
                                    s |= bit << 31;
                                }
                                if (((s & x) ^ (~s & y)) != Z[j]) {
                                    return false;
                                }
                            }
                            return true;
                        })
                        .findAny()
                        .ifPresent(s -> {
                            String s1 = String.format("%32s",
Integer.toBinaryString(Integer.reverse(potentialX <<
2)))
                                    .replaceAll(" ", "0");
                            System.out.println(s1);
                            String s2 = String.format("%32s",
Integer.toBinaryString(Integer.reverse(potentialY <<
1)))
                                    .replaceAll(" ", "0");
                            System.out.println("L2: " + s2);
                            String s3 = String.format("%32s",
Integer.toBinaryString(Integer.reverse(s)))
                                    .replaceAll(" ", "0");
                            System.out.println("L3: " + s3);

System.out.println(Duration.between(start,
LocalDateTime.now()));
                            System.exit(0);
                        });
            }
        }
    }

    private static int countStatisticL1(int x) {
        int R = 0;
        // fill 30-th and 31-th bits
        int bits = (x ^ (x >>> 1) ^ (x >>> 4) ^ (x >>> 6))
& 3;
        x = (bits << 30) | x;
        // Ordered
        for (int z : Z) {
            R = R + Integer.bitCount(x ^ z);
            // shift register on 32 bits (integer size)
            for (int j = 0; j < Integer.SIZE; j++) {
                x = ((((x >>> 2) ^ (x >>> 3) ^ (x >>> 6) ^
(x >>> 8)) & 1) << 31) | (x >>> 1);
            }
        }
        return R;
    }

    private static int countStatisticL2(int y) {
        int R = 0;
        // fill 31-th bit
        y = (((y ^ (y >>> 3)) & 1) << 31) | y;
        // Ordered
        for (int z : Z) {
            R = R + Integer.bitCount(y ^ z);
            // shift register on 32 bits (integer size)
            for (int j = 0; j < 32; j++) {
                y = ((((y >>> 1) ^ (y >>> 4)) & 1) << 31) |
(y >>> 1);
            }
        }
        return R;
    }
}
```