Semester 1 **Exercises and Solutions for Week 9** 2004

Properties of Binary Sequences and LFSR's

Statistical Properties. The output of a linear feedback shift operator of length n has a period, which must divide $2^n - 1$. The period is independent of the initial state, provided it is non-zero. If the period equals $2^n - 1$, the output sequence is said to be an m-sequence. The following theorem describes the statistical properties of m-sequences.

Theorem 1 *Let $s = s_0 s_1 \ldots$ be an m-sequence, and let k be an integer with $1 \leq k \leq n$. Then in each subsequence of s of length $2^n + k - 2$, every finite nonzero binary sequence of length k appears as a subsequence exactly 2^{n-k} times, and the length k zero subsequence appears exactly $2^{n-k} - 1$ times.*

A polynomial $g(x)$ in $\mathbb{F}_2[x]$ is *irreducible* if it is not the product of two polynomials of degree greater than zero. An irreducible polynomial $g(x)$ of degree n is *primitive* if $g(x)$ divides $x^N - 1$ for $N = 2^n - 1$ and no smaller value of N. (Equivalently, $g(x)$ is primitive if the powers x^i are distinct modulo $g(x)$, for i with $1 \leq i \leq 2^n - 1$.)

Linear Complexity. A linear feedback shift register is said to generate a binary sequence s if there exists some initial state for which its output sequence is s. The linear complexity $L(s)$ of an infinite sequence s is defined to be zero if s is the zero sequence, infinity if s is generated by no linear feedback shift register, and otherwise equal to the minimal length of a linear feedback shift register generating s. The linear complexity $L(s)$ of a finite sequence s is defined to be the minimal length of a linear feedback shift register with initial sequence s for some initial state.

Linear Complexity Profile. For a sequence s, define $L_j(s)$ to be the linear complexity of the first j terms of the sequence. The linear complexity profile of an infinite sequence s is defined to be the infinite sequence $L_1(s), L_2(s), \ldots$, and for a finite sequence $s = s_0 s_1 \ldots s_{N-1}$ is defined to be the finite sequence $L_1(s), L_2(s), \ldots, L_n(s)$.

LFSR Cryptosystems We introduce new utilities for binary stream cryptosystems based on linear feedback shift registers. The functions `BitEncoding` and `BitDecoding` convert ASCII text into its bit sequence and back. In addition, the new binary cryptosystems are:

```
LFSRCryptosystem
ShrinkingGeneratorCryptosystem
```

Unlike the encoding function `StripEncoding` we have used so far, the function `BitEncoding` is information-preserving, taking 8-bit ASCII input and returning the binary encoding string. The inverse function `BitDecoding` recovers the original text.

A linear feedback shift register cryptosystem is created in `Magma` using the function `LFSRCryptosystem`, taking no arguments. A key is defined by means of a pair, consisting of the connection polynomial $g(x)$ over $\mathbb{F}_2$ and a initial bit sequence of length equal to the degree of the sequence. A sample use of the cryptosystem follows. The shrinking generator cryptosystem is a cryptosystem based on a pair of LFSR's as defined in class.

```
> F2 := FiniteField(2);
> P2<x> := PolynomialRing(F2);
> g := x^17 + x^5 + 1;
> IS := [ Random(F2) : i in [1..17] ];
> LFSR := LFSRCryptosystem();
> PT := Encoding(LFSR,"The dog ate my assignment."); PT;
01010100011010000110010100100000011001000110111011001110
01000000110000101110100011001010010000001101101011110010 0
10000001100001011100110111001101101001011001110110111001 1
01101011001010110111001110100001011 10
> K := LFSR!<g,IS>;
> CT := Enciphering(K,PT);
> PT eq Enciphering(K,CT);
true
```

Note that the encoding of the message is not ciphertext – this is the standard ASCII bit encoding.

1. Consider the coefficient sequence for $f(x)/g(x)$ in $\mathbb{F}_2[[x]]$, where $g(x) = 1 + x + x^4$ and $f(x) = 1 + x^3$. Is $g(x)$ an irreducible polynomial? A primitive polynomial? Draw the associated linear feedback shift register. What is the initial state of the shift register?

Solution The polynomial $x^4 + x + 1$ is an irreducible polynomial, which is primitive. The LFSR with this connection polynomial was given in the previous tutorial. The primitivity follows since none of the sequences, computed last week, had a period shorter than 15. The initial state corresponding to the polynomial $f(x) = x^3 + 1$ was the second given value 1110 of the previous tutorial.

2. Compute the linear complexity of the sequences 11, 1011, 10101, 10110, and 10011.

Solution The linear complexity of the sequences 11, 1011, 10101, 10110, and 10011 is 1, 2, 2, 2, and 3. The initial values follow from extending the sequences with period 1, 3, 2, and 3, with connection polynomials $x + 1$, $x^2 + x + 1$, $x^2 + 1$, and $x^2 + x + 1$. The third sequence can be extended to a sequence with period no better than 4, so it generated by no LFSR of length 2. A possible connection polynomial is $x^4 + 1 = (x + 1)^4$, giving a LFSR of length 4 which generates it. However, the divisor $x^3 + x^2 + x + 1 = (x + 1)^3$ defines a recursion for a LFSR of length 3. Hence the linear complexity for this sequence is 3.

3. Compute the first 8 terms of the linear complexity profile of the coefficient sequence from Exercise 1.

Solution The first 8 terms of the linear complexity profile for the sequence of the first question are:

$$[1, 1, 1, 3, 3, 3, 4, 4].$$

On the other hand, since the sequence is generated by a LFSR of length 4 we know that the full infinite sequence becomes constant at 4.

4. Practice encoding and enciphering with the LFSR stream cryptosystem. The cryptographic text prints compactly, but the function `String` returns the underlying `Magma` sequence over $\mathbb{F}_2$. The function `BitDecoding` easily converts this to ASCII text. Use these functions to verify that `PT` is just the binary encoding of the original plaintext message and that the ciphertext is enciphered.

Solution The cryptotext datatype, for binary texts, is just a printing function on top of binary sequences. The underlying sequence can be extracted with `String` and reformed with `Encoding`:

```
> LFSR := LFSRCryptosystem();
> PT := Encoding(LFSR,"The dog ate my assignment."); PT;
010101000110100001100101001000000110010001101111011001110
010000001100001011101000110010100100000011011010111100100
100000011000010111001101110011011010010110011101101110011
011010110010101101110011101000010111 0
> String(PT);
[
0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1,
0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1,
0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0,
0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1,
0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0,
0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0,
1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,
0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1,
0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1,
1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0 ]
> Encoding(LFSR,$1);
010101000110100001100101001000000110010001101111011001110
010000001100001011101000110010100100000011011010111100100
100000011000010111001101110011011010010110011101101110011
011010110010101101110011101000010111
```

Note that `Encoding` can take either text strings or binary sequences. We verify using `BitDecoding` that the underlying sequence is just the ASCII encoding of the message:

```
> BitDecoding(String(PT));
The dog ate my assignment.
```

5. Since the LFSR is the bitsum of the binary keystream, generated by the connection polynomial and initial state, why must the inverse key be equal to the key itself?

Solution Since LFSR ciphertext is the bitsum of plaintext with a keystream, a subsequent bitsum with the same keystream gives the original plaintext:

$$c_i + s_i = (m_i + s_i) + s_i = m_i + (s_i + s_i) = m_i + 0 = m_i,$$

Therefore enciphering map is equal to deciphering map; in particular, the enciphering and deciphering keys are the same.