Lecture Four

Basic Efficiency Criteria of LFSR's-Systems

4. Linear complexity (LC) Criterion

Definition (4.1): An LFSR is said to **generate** a finite sequence S^n if there is some initial state for which the output sequence of the LFSR has S^n as its first n terms.

4.1 Linear Complexity Concept

Definition (4.2): The **linear complexity** of an infinite binary sequence S, denoted LC(S), is defined as follows:

(i). if S is the zero sequence S = 0, 0, 0, . . . , then LC(S) = 0;

(ii). if no LFSR generates S, then LC(S) = ∞;

(iii). Otherwise, LC(S) is the length of the shortest LFSR that generates S.

Definition (4.3): The **linear complexity** of a finite binary sequence S^n, denoted $LC(S^n)$, is the length of the shortest LFSR that generates a sequence having S^n as its first n terms.

Remark (4.1) (**properties of linear complexity**) Let S and T be binary sequences.

(i). For any $n \geq 1$, the linear complexity of the subsequence S^n satisfies $0 \leq LC(S^n) \leq n$.

(ii). $LC(S^n) = 0$ if and only if S^n is the zero sequence of length n.

(iii). $LC(S^n) = n$ if and only if $S^n = 0, 0, 0, \dots, 0, 1$.

(iv). If S is periodic with period N, then $LC(S) \leq N$.

(v). $LC(S \oplus T) \leq LC(S) + LC(T)$, where $S \oplus T$ denotes the bitwise XOR of S and T.

4.2 Linear Complexity Profile

Definition (4.4): Let $S=s_0,s_1,...$ be a binary sequence, and let LC_N denote the linear complexity of the subsequence $S^N=s_0,s_1,...,s_{N-1}$, $N\geq 0$. The sequence LC_1, LC_2,... is called the **linear complexity profile** of S. Similarly, if $S^n = s_0, s_1,..., s_{n-1}$ is a finite binary sequence, the sequence $LC_1, LC_2,...,LC_n$ is called the **linear complexity profile** of S^n.

The linear complexity profile of a sequence can be computed using the **Berlekamp-Massey algorithm**. This algorithm can be considered as one of the attack methods, so we will detailed it in lecture six.

The linear complexity profile of a sequence S can be graphed by plotting the points (N,LC_N), $N\geq 1$, in the $N\times LC$ plane and joining successive points by a horizontal line followed by a vertical line, if necessary (see Figure (1)). The graph of a linear complexity profile is non-decreasing. Moreover, a (vertical) jump in the graph can only occur from below the line $LC=N/2$; if a jump occurs, then it is symmetric about this line. It's important to show that the expected linear complexity of a random sequence should closely follow the line $LC=N/2$.

Example (4.1): (**linear complexity profile**) Consider the 20-periodic sequence S with cycle S^{20} = 1,0,0,1,0,0,1,1,1,1,0,0,0,1,0,0,1,1,1,0. The linear complexity profile of S is 1, 1, 1, 3, 3, 3, 3, 5, 5, 5, 6, 6, 6, 8, 8, 8, 9, 9, 10, 10, 11, 11, 11, 11, 14, 14, 14, 14, 15, 15, 15, 17, 17, 17, 18, 18, 19, 19, 19, 19,…. Figure (1) shows the graph of the linear complexity profile of S.

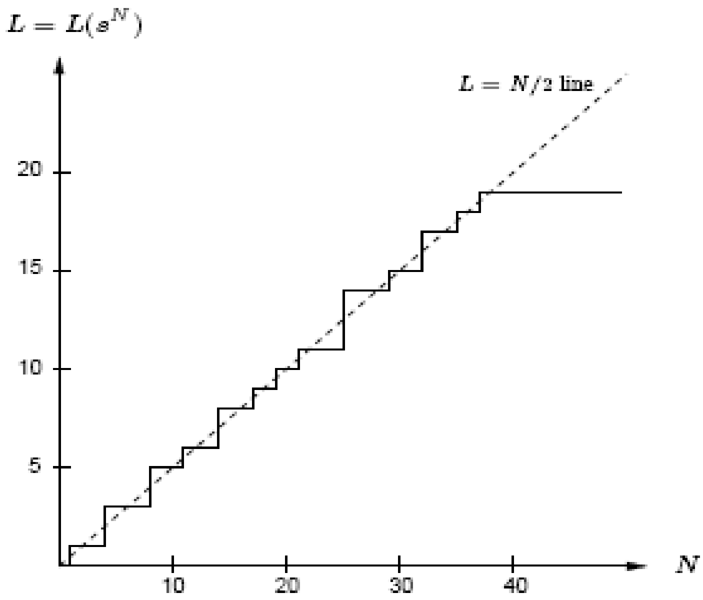

Figure (1): Linear complexity profile of the 20-periodic sequence.

As is the case with all statistical tests for randomness, the condition that a sequence S has a linear complexity profile that closely resembles that of a random sequence is necessary but not sufficient for S to be considered random. This point is illustrated in the following example.

<u>Example (4.2)</u> (**limitations of the linear complexity profile**), the linear complexity profile of the sequence S defined as:

$$s_i = \begin{cases} 1, & \text{if } i = 2^j - 1 \text{ for some } j \geq 0, \\ 0, & \text{otherwise.} \end{cases} \quad \ldots(4.1)$$

follows the line LC=N/2 as closely as possible. That is, $LC(S^N)=\lfloor (N+1)/2 \rfloor$ for all $N \geq 1$. However, the sequence S is clearly non-random.

4.3 Berlekamp-Massey Algorithm

The Berlekamp-Massey algorithm is an efficient algorithm for determining the linear complexity of a finite binary sequence S^n of length n (see Definition (3.5)). The algorithm takes n iterations, with the N^{th} iteration computing the linear complexity of the subsequence S^N consisting of the first N terms of S^n. The theoretical basis for the algorithm is Remark (4.3).

Definition (4.5) Consider the finite binary sequence $S^{N+1}=s_0,s_1,\ldots,s_{N-1}, s_N$. For $C(D)=1+c_1D+\ldots+c_rD^r$, let $\langle r,C(D)\rangle$ be an LFSR that generates the subsequence $S^N = s_0, s_1,\ldots, s_{N-1}$. The next discrepancy d_N is the difference between s_N and the $(N+1)^{st}$ term generated by the LFSR:

$$d_N=(s_N+\sum_{i=1}^{r} c_i s_{N-i}) \bmod 2 \qquad \ldots(4.2)$$

Remark (4.1): Let $S^N = s_0, s_1,\ldots, s_{N-1}$ be a finite binary sequence of linear complexity $LC=LC(S^N)$, and let $\langle r,C(D)\rangle$ be an LFSR which generates S^N.

(i). The LFSR $\langle r,C(D)\rangle$ also generates $S^{N+1} = s_0, s_1,\ldots, s_{N-1}, s_N$ if and only if the next discrepancy d_N is equal to 0.

(ii). If $d_N = 0$, then $LC(S^{N+1}) = r$.

(iii). Suppose $d_N=1$. Let m the largest integer<N such that $LC(S^m)<LC(S^N)$, and let $\langle LC(S^m),B(D)\rangle$ be an LFSR of length $LC(S^m)$ which generates S^m. Then $\langle r',C'(D)\rangle$ is an LFSR of smallest length which generates S^{N+1}, where

$$r' = \begin{cases} r, & \text{if } r > N/2, \\ N+1-r, & \text{if } r \le N/2. \end{cases} \qquad \ldots(4.3)$$

and $C'(D) = C(D) + B(D).D^{N-m}$.

Berlekamp-Massey algorithm

INPUT: a binary sequence $S^n = s_0, s_1, s_2, \ldots, s_{n-1}$ of length n.

OUTPUT: the linear complexity $L(S^n)$ of S^n, $0 \le L(S^n) \le n$.

PROCESS: 1. Initialization. $C(D) \leftarrow 1$, $r \leftarrow 0$, $m \leftarrow -1$, $B(D) \leftarrow 1$, $N \leftarrow 0$.

2. While ($N < n$) do the following:

2.1Compute the next discrepancy d. $d \leftarrow (s_N + \sum_{i=1}^{L} c_i s_{N-i})\bmod 2$.

2.2 If d = 1 then do the following:

$T(D) \leftarrow C(D)$, $C(D) \leftarrow C(D) + B(D).D^{N-m}$.

If $r \le N/2$ then $r \leftarrow N + 1 - r$, $m \leftarrow N$, $B(D) \leftarrow T(D)$.

2.3 $N \leftarrow N + 1$.

3. Return(r).

Remark (4.2): (intermediate results in Berlekamp-Massey algorithm) At the end of each iteration of step 2, $\langle r, C(D)\rangle$ is an LFSR of smallest length which generates S^N. Hence, Berlekamp-Massey algorithm can also be used to compute the linear complexity profile (Definition (3.6)) of a finite sequence.

Example (4.3) (Berlekamp-Massey algorithm), Table (3.1) shows the steps of Berlekamp-Massey algorithm for computing the linear complexity of the binary sequence $S^n = 0, 0, 1, 1, 0, 1, 1, 1, 0$ of length n=9. This sequence is found to have linear complexity 5, and an LFSR which generates it is $\langle 5, 1 + D^3 + D^5\rangle$.

Remark (4.3): Let S^n be a finite binary sequence of length n, and let the linear complexity of S^n be LC. Then there is a unique LFSR of length LC which generates S^n if and only if $LC \le n/2$.

Table (3.1) Steps of the Berlekamp-Massey algorithm of example (3.3).

s_N	d	T(D)	C(D)	r	M	B(D)	N

-	-	-	1	0	-1	1	0
0	0	-	1	0	-1	1	1
0	0	-	1	0	-1	1	2
1	1	1	$1+D^3$	3	2	1	3
1	1	$1+D^3$	$1+D+D^3$	3	2	1	4
0	1	$1+D+D^3$	$1+D+D^2+D^3$	3	2	1	5
1	1	$1+D+D^2+D^3$	$1+D+D^2$	3	2	1	6
1	0	$1+D+D^2+D^3$	$1+D+D^2$	3	2	1	7
1	1	$1+D+D^2$	$1+D+D^2+D^5$	5	7	$1+D+D^2$	8
0	1	$1+D+D^2+D^5$	$1+D^3+D^5$	5	7	$1+D+D^2$	9

4.4 Non-Linearity of Combining Functions

One general technique for destroying the linearity inherent in LFSRs is to use several LFSRs in parallel. The keystream is generated as a nonlinear function f of the outputs of the component LFSRs. Such keystream generators are called **nonlinear combination generators**, and f is called the **combining function**. The remainder of this subsection demonstrates that the function f must satisfy several criteria in order to withstand certain particular cryptographic attacks.

Definition (4.6) A product of m distinct variables is called an **m^{th} order product** of the variables. Every Boolean function $f(x_1,x_2,\ldots,x_n)$ can be written as a modulo 2 sum of distinct m^{th} order products of its variables, $0 \le m \le n$; this expression is called the **algebraic normal form of f**. The nonlinear order of f is the maximum of the order of the terms appearing in its algebraic normal form.

Example (4.4) the Boolean function $f(x_1,x_2,x_3,x_4,x_5)=1\oplus x_2\oplus x_3\oplus x_4x_5\oplus x_1x_3x_4x_5$ has nonlinear order 4. Note that the maximum possible nonlinear order of a Boolean function in n variables is n. Remark (4.3) demonstrates that the output sequence of a nonlinear combination generator has high linear complexity, provided that a combining function f of high nonlinear order is employed.

Remark (4.4): Suppose that n maximum-length LFSRs, whose lengths r_1, $r_2,\ldots,r_n$ are pairwise distinct and greater than 2, are combined by a nonlinear function $f(x_1,x_2,...,x_n)$ which is expressed in algebraic normal form. Then the linear complexity of the keystream is $f(r_1,r_2,\ldots,r_n)$. (The expression $f(r_1,r_2,\ldots,r_n)$ is evaluated over the integers rather than over Z_2).

Let CF=F_n, so that in general $LC(S)\leq F_n^*(r_1,r_2,\ldots,r_n)$, F_n^* is the integer function corresponding to F_n s.t. $F_n^*:Z^+\rightarrow Z^+$. Since the 2nd and 3rd conditions are hold, then:

$$LC(S)=F_n^*(r_1,r_2,\ldots,r_n) \qquad \ldots(4.4)$$

Notice that LC(S) depends on LFSR and CF units. The basic condition to construct efficient KG is "Lengths of combined LFSR's must be long as possible". This condition will contributes to make S has maximum period. The other condition is "CF has high non-linear order", so if the five conditions are holding, this will make S has a high LC to pass the computer ability in exhaustive search or brute forces attack.

Now when applying the LC criterion on the studied cases we get:

1. **n-LKG**: S has $LC(S)=\sum_{i=1}^{n} r_i$.
2. **n-PKG**: S has $LC(S)=\prod_{i=1}^{n} r_i$.
3. **3-BKG**: S has $LC(S)=r_1\cdot r_2 + r_1\cdot r_3 + r_2\cdot r_3$.

Example (4.5):

Table (3) describes linear complexity of different examples of the three study cases.

Table (3) linear complexity of some examples of the three study cases.

n	r_i	LC(S)		
		n-LKG	**n-PKG**	**3-BKG**
3	2,3,5	10	30	31
3	4,5,7	16	140	83
4	2,3,5,7	17	210	-----