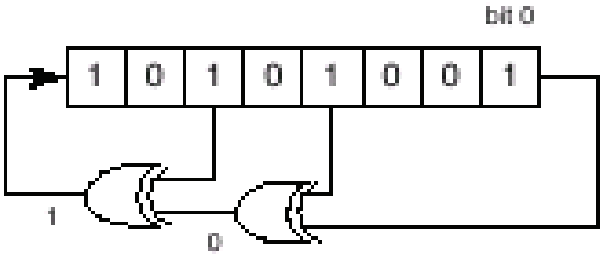
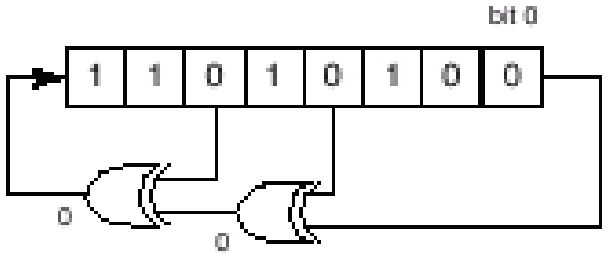
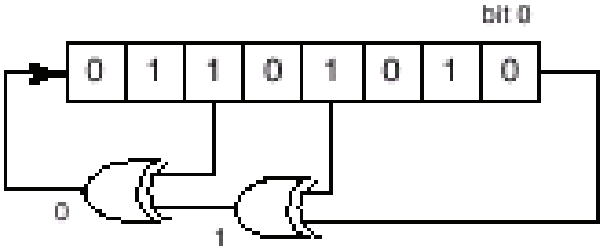
Linear Feedback Shift Registers



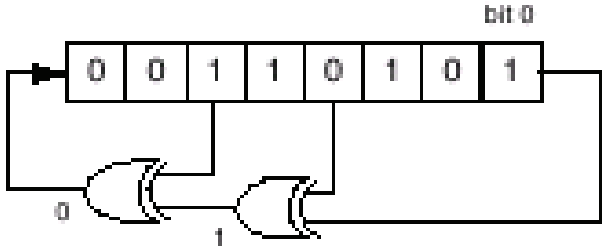
start state



state after one pulse



state after two pulses



state after three pulses

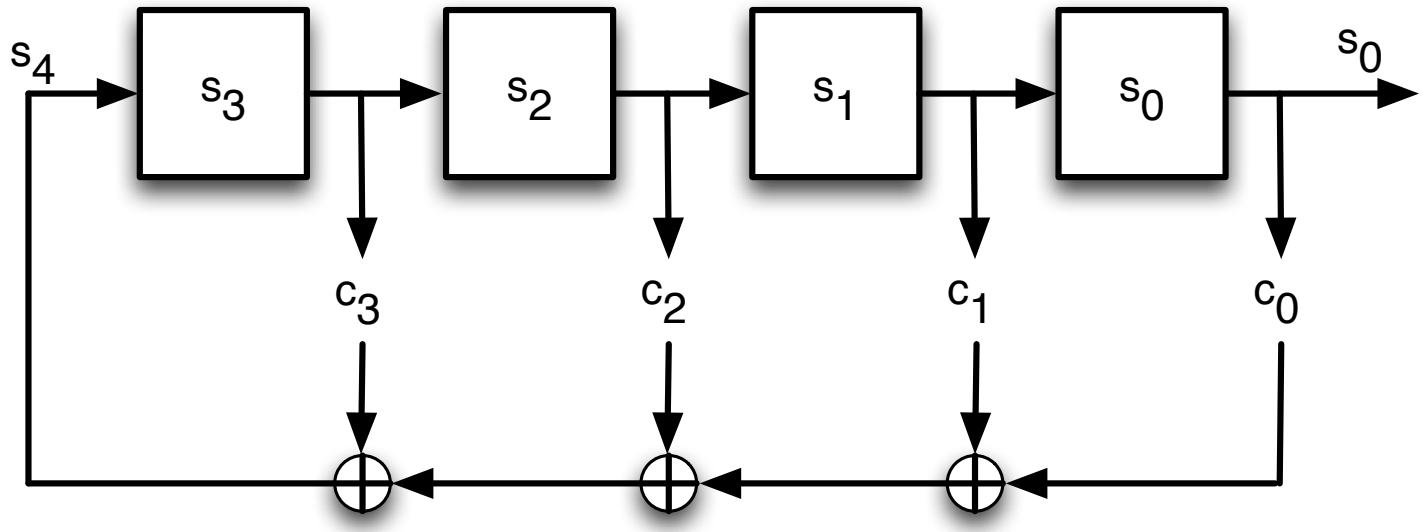
LFSR Structure

- A linearly connected shift register of n cells, each of which is holding a state variable $s_i \in \{0, 1\}$ and set of coefficients $c_i \in \{0, 1\}$, for $i = 0, 1, \ldots, n-1$
- The feedback function which is addition mod 2 (the XOR function), computing the new state value s_n using the coefficients and the state values as

$$\begin{aligned} s_n &= c_0 \cdot s_0 + c_1 \cdot s_1 + \cdots + c_{n-1} \cdot s_{n-1} \pmod 2 \\ &= c_0 \cdot s_0 \oplus c_1 \cdot s_1 \oplus \cdots \oplus c_{n-1} \cdot s_{n-1} \end{aligned}$$

- The (right) shift function which places s_{i+1} into s_i for $0 \leq i \leq n-1$, and therefore, $(s_{n-1}, s_{n-2}, \ldots, s_1, s_0) \rightarrow (s_n, s_{n-1}, \ldots, s_2, s_1)$

4-bit LFSR Example



4-bit LFSR, $n = 4$
State vector: (s_3, s_2, s_1, s_0)
Coefficient vector: (c_3, c_2, c_1, c_0)
s_0 is the output of the LFSR

$$s_4 = c_3 \cdot s_3 + c_2 \cdot s_2 + c_1 \cdot s_1 + c_0 \cdot s_0 \pmod 2$$

4-bit LFSR Example

- Given the coefficients $(c_3, c_2, c_1, c_0) = (1, 1, 1, 1)$ and the initial states $(s_3, s_2, s_1, s_0) = (0, 0, 0, 1)$, we compute the next state s_4 as

$$\begin{aligned} s_4 &= 1 \cdot s_3 + 1 \cdot s_2 + 1 \cdot s_1 + 1 \cdot s_0 \\ &= 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 1 = 1 \end{aligned}$$

  Therefore, the new state is $(s_4, s_3, s_2, s_1) = (1, 0, 0, 0)$

- Proceeding this way, we find the subsequent state s_5

$$\begin{aligned} s_5 &= 1 \cdot s_4 + 1 \cdot s_3 + 1 \cdot s_2 + 1 \cdot s_1 \\ &= 1 \cdot 1 + 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 0 = 1 \end{aligned}$$

  and thus $(s_5, s_4, s_3, s_2) = (1, 1, 0, 0)$

LFSR Period for $(c_3, c_2, c_1, c_0) = (1, 1, 1, 1)$

$c_3 = 1$	$c_2 = 1$	$c_1 = 1$	$c_0 = 1$	coefficients
0	0	0	1	initial state
1	0	0	0	
1	1	0	0	
0	1	1	0	
0	0	1	1	
0	0	0	1	back to initial state

- The period of this LFSR with this initial state is 5 since there are 5 independent (unequal) states
- Regardless of the connection coefficients, the initial state of all-zero will cause the future states be all-zero; the period will always be 1

$$\begin{aligned} s_4 &= c_3 \cdot s_3 + c_2 \cdot s_2 + c_1 \cdot s_1 + c_0 \cdot s_0 \\ &= c_3 \cdot 0 + c_2 \cdot 0 + c_1 \cdot 0 + c_0 \cdot 0 \;=\; 0 \end{aligned}$$

LFSR Maximal Period

- Since there are n state variables, taking binary values, the number of all possible states is 2^n
- However, due to the linearity of the feedback function, an all-zero state at any given time causes all future states be all zero
- Therefore, an LFSR can have at most $2^n - 1$ unique states, since the all-zero state is excluded
- Question 1: Are there sets of connection coefficients and initial states that produce sequences with the maximal period?
- Question 2: Are there sets of connection coefficients that produce sequences with the maximal period, regardless of the initial state (of course, except all zero state)

Another LFSR: $(c_3, c_2, c_1, c_0) = (0, 0, 1, 1)$

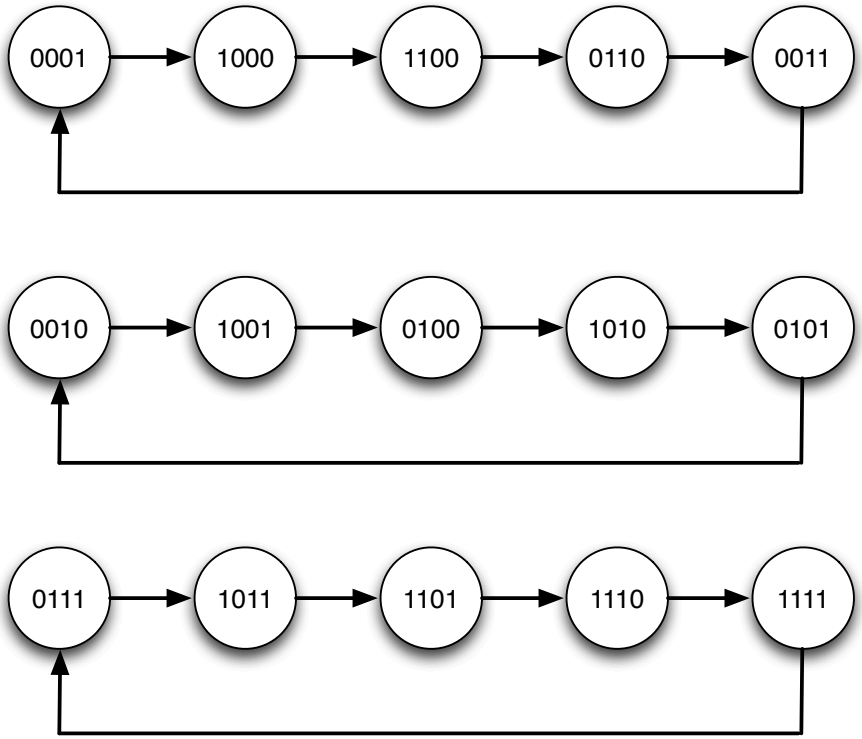
$c_3 = 0$	$c_2 = 0$	$c_1 = 1$	$c_0 = 1$	coefficients
0	0	0	1	(1)
1	0	0	0	(2)
0	1	0	0	(3)
0	0	1	0	(4)
1	0	0	1	(5)
1	1	0	0	(6)
0	1	1	0	(7)
1	0	1	1	(8)
0	1	0	1	(9)
1	0	1	0	(10)
1	1	0	1	(11)
1	1	1	0	(12)
1	1	1	1	(13)
0	1	1	1	(14)
0	0	1	1	(15)
0	0	0	1	back to (1)

LFSR Period Properties

- The period of the 4-bit LFSR with coefficients $(c_3, c_2, c_1, c_0) = (0, 0, 1, 1)$ is equal 15, which is the maximal period for a 4-bit LFSR: $2^4 - 1 = 15$
- As can be seen, the period of this LFSR will always be 15 for any of the initial states, except the all-zero state
- In general, the period of an LSFR is a function of the coefficient vector and also the initial state vector
- For the 4-bit LFSR with coefficients $(c_3, c_2, c_1, c_0) = (1, 1, 1, 1)$, the period was 5 for the initial state $(s_3, s_2, s_1, s_0) = (0, 0, 0, 1)$
- In fact, the period will still be 5 for any of the initial states $(0, 0, 0, 1)$, $(1, 0, 0, 0)$, $(1, 1, 0, 0)$, $(0, 1, 1, 0)$, and $(0, 0, 1, 1)$; these are the states visited when starting with $(0, 0, 0, 1)$

LFSR Period Properties

- For $(c_3, c_2, c_1, c_0) = (1, 1, 1, 1)$, the LFSR exhibits the following behavior, depending on the initial state



Solomon Wolf Golomb

- Solomon Wolf Golomb (1932) is an American mathematician and engineer, best known to the general public and fans of mathematical games as the inventor of polyominoes, the inspiration for the computer game Tetris
- He worked on combinatorial analysis, number theory, coding theory and communications
- Golomb invented Cheskers (a variant of checkers) and the pentominoes in 1948 and 1953 respectively
- Golomb pioneered the identification of the characteristics and merits of maximum length shift register sequences
- Because of Golomb's work, we know how to build LFSRs with maximum period

The Connection Polynomial

- Given a coefficient vector $(c_{n-1}, c_{n-1}, \ldots, c_1, c_0)$ for a n-bit LFSR, we can write a polynomial of degree n

$$c(x) = x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_1 x + c_0$$

- This is called the connection polynomial of the n-bit LFSR
- For example, for the 4-bit LFSR with $(c_3, c_2, c_1, c_0) = (1, 1, 1, 1)$, the connection polynomial is $x^4 + x^3 + x^2 + x + 1$, which is of degree 4
- Similarly, for the 4-bit LFSR with $(c_3, c_2, c_1, c_0) = (0, 0, 1, 1)$, the connection polynomial is $x^4 + x + 1$, which is of degree 4

Golomb's First Theorem

- The coefficients are in $\{0, 1\}$, and the arithmetic of these polynomials are performed modulo 2
- In other words, these polynomials are defined over the finite field GF(2)

Theorem

If the connection polynomial (of degree n) of an n-bit LFSR is reducible, then its its period is not maximal ($\neq 2^n - 1$)

Irreducible versus Reducible Polynomials

- Definition: A polynomial is called **reducible** if it can be factored into smaller degree polynomials
- For example, the following polynomials are reducible:

$$\begin{aligned} x^2 &= x \cdot x \\ x^2 + x &= x \cdot (x+1) \\ x^2 + 1 &= (x+1) \cdot (x+1) \end{aligned}$$

- We are factoring the polynomials whose coefficients are also in the set $\{0, 1\}$, i.e., we are performing mod 2 arithmetic with the coefficients
- Since $1 + 1 = 0$, and thus, $x + x = 0$, the polynomial $x^2 + 1$ is reducible:

$$(x+1) \cdot (x+1) = x^2 + x + x + 1 = x^2 + 1$$

Irreducible versus Reducible Polynomials

- A polynomial that is not reducible is called irreducible
- For example, the polynomial $x^2 + x + 1$ is irreducible: we cannot write it as the product lesser degree (that is, of degree 1) polynomials
- To see that, consider all polynomials of degree (up to) 1: 0, 1, x, $x + 1$; if we form all possible products, we obtain all reducible polynomials:

$*$	0	1	x	$x + 1$
0	0	0	0	0
1	0	1	x	$x + 1$
x	0	x	x^2	$x^2 + x$
$x + 1$	0	$x + 1$	$x^2 + x$	$x^2 + 1$

- The polynomial $x^2 + x + 1$ is not in the list, and therefore, it is irreducible

Irreducible Polynomials

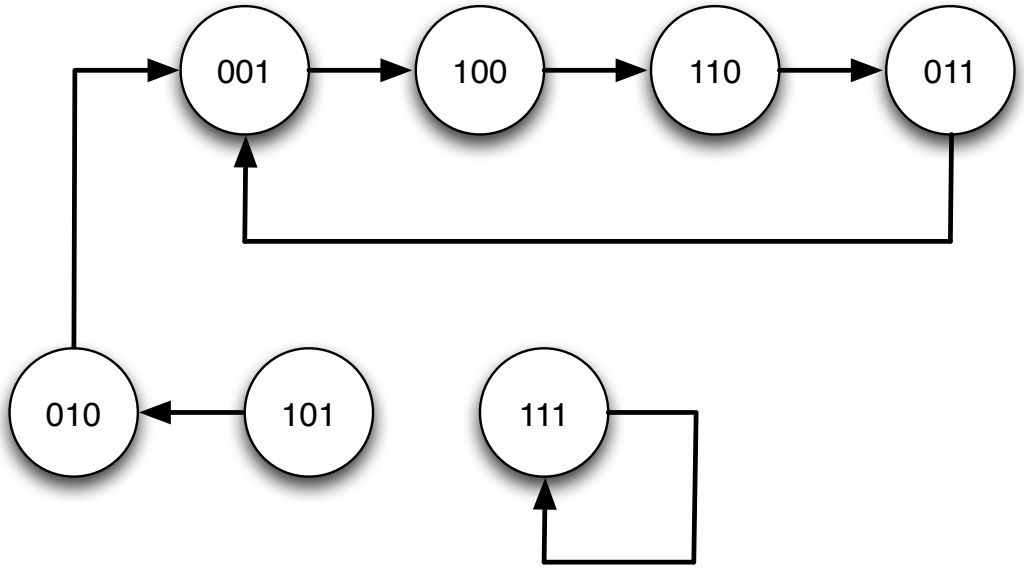
- Mathematicians and computer scientists are interested in irreducible polynomials for various reasons
- Algorithms for finding irreducible polynomials and deciding if a given polynomial is irreducible have been developed
- Particularly of interest are polynomials whose coefficients are from the set $\{0, 1\}$ and arithmetic is mod 2
- These are called polynomials over the finite field GF(2), i.e., Galois field of 2 elements
- Lists of irreducible polynomials over GF(2) have been published

Irreducible Polynomials over GF(2)

n	irreducible polynomials		
1	x	$x+1$	
2	x^2+x+1		
3	x^3+x+1	x^3+x^2+1	
4	x^4+x+1	x^4+x^3+1	$x^4+x^3+x^2+x+1$
5	x^5+x^2+1	x^5+x^3+1	$x^5+x^3+x^2+x+1$
	$x^5+x^4+x^3+x+1$	$x^5+x^4+x^3+x^2+1$	$x^5+x^4+x^2+x+1$
6	x^6+x+1	x^6+x^3+1	x^6+x^5+1
	$x^6+x^4+x^2+x+1$	$x^6+x^4+x^3+x+1$	$x^6+x^5+x^2+x+1$
	$x^6+x^5+x^3+x^2+1$	$x^6+x^5+x^4+x^2+1$	$x^6+x^5+x^4+x+1$

A Reducible Polynomial Example

- For example, consider the 3-bit LFSR with $(c_2, c_1, c_0) = (1, 1, 1)$, and its degree-3 connection polynomial $c(x) = x^3 + x^2 + x + 1$
- Since this polynomial is reducible, the LFSR is not maximal
- Indeed, the iteration of this LFSR with different initial states gives its period at most as 4



Irreducible Polynomials

- Irreducible polynomials are important for binary extension fields $GF(2^k)$
- The elements of $GF(2^k)$ are polynomials of degree at most $k-1$, with coefficients from the ground field $GF(2)$
- The multiplication of two elements of $GF(2^k)$ are performed using

$$r(x) = a(x)b(x) \bmod g(x)$$

where $g(x)$ is an irreducible polynomial of degree k

- Irreducible polynomials show up in a different context again!
- The coefficient polynomial $c(x)$ of an LFSR needs to **at least** irreducible for it to have maximal period

Is Irreducibility Sufficient for Maximality?

- Unfortunately, irreducibility is a **necessary but not a sufficient** condition for maximality of the LFSRs
- Indeed, we discovered that the 4-bit LFSR with the coefficients $(c_3, c_2, c_1, c_0) = (1, 1, 1, 1)$ was not maximal
- The period was at most 5, even though its connection polynomial

$$x^4 + x^3 + x^2 + x + 1$$

irreducible over GF(2), as can be seen from the list above

- It turns out that for ensured maximality the connection polynomial needs to be **primitive**

Golomb's Second Theorem

Theorem

If the connection polynomial of degree n is a primitive polynomial, then the associated LFSR is maximal, with period $2^n - 1$.

- Primitivity of polynomials are related to the primitivity of integers, but it is based on different principles
- For polynomial to be primitive, first it needs to be irreducible
- *Definition:* An irreducible polynomial $p(x)$ of degree n over GF(2) is called primitive, if the smallest integer e for which $p(x)$ divides $x^e + 1$ is $e = 2^n - 1$

Primitive Polynomials over GF(2)

- Consider the irreducible polynomial $p(x) = x^4 + x^3 + x^2 + x + 1$, and apply the division algorithm to divide the polynomial $x^e + 1$ by $p(x)$ for increasing values of e, and find the smallest e value such that the remainder (R) is zero:

$$e = 4 \rightarrow \frac{x^4+1}{x^4+x^3+x^2+x+1} \rightarrow \text{Q} = 1 \qquad \text{R} = x^3 + x^2 + x$$

$$e = 5 \rightarrow \frac{x^5+1}{x^4+x^3+x^2+x+1} \rightarrow \text{Q} = x + 1 \quad \text{R} = 0$$

- Since the irreducible polynomial $x^4 + x^3 + x^2 + x + 1$ divides $x^5 + 1$ and $e = 5 \neq 2^4 - 1 = 15$, we conclude that $x^4 + x^3 + x^2 + x + 1$ is not primitive

Primitive Polynomials over GF(2)

- Now, consider another irreducible polynomial $p(x) = x^3 + x + 1$, and apply the division algorithm again to find the smallest e for this polynomial

$$
\begin{array}{lcll}
e = 3 \rightarrow & \frac{x^3+1}{x^3+x+1} & = & \text{Quo: } 1 \qquad\qquad \text{Rem: } x \\
e = 4 \rightarrow & \frac{x^4+1}{x^3+x+1} & = & \text{Quo: } x \qquad\qquad \text{Rem: } x^2 + x + 1 \\
e = 5 \rightarrow & \frac{x^5+1}{x^3+x+1} & = & \text{Quo: } x^2 + 1 \qquad\qquad \text{Rem: } x^2 + x \\
e = 6 \rightarrow & \frac{x^6+1}{x^3+x+1} & = & \text{Quo: } x^3 + x + 1 \qquad\qquad \text{Rem: } x^2 \\
e = 7 \rightarrow & \frac{x^7+1}{x^3+x+1} & = & \text{Quo: } x^4 + x^2 + x + 1 \qquad\qquad \text{Rem: } 0
\end{array}
$$

- Since the irreducible polynomial $x^3 + x + 1$ divides $x^7 + 1$ and $e = 7 = 2^3 - 1$, we conclude that $x^3 + x + 1$ is primitive

Example: A Primitive Polynomial over GF(2)

- Since $p(x) = x^3 + x + 1$ is a primitive polynomial, the 3-bit LFSR with the connection coefficients $(c_2, c_1, c_0) = (0, 1, 1)$ produces a sequences with the maximal period 7

$c_2 = 0$	$c_1 = 1$	$c_0 = 1$	coefficients
0	0	1	(1)
1	0	0	(2)
0	1	0	(3)
1	0	1	(4)
1	1	0	(5)
1	1	1	(6)
0	1	1	(7)
0	0	1	back to (1)

List of Primitive Polynomials over GF(2)

n	primitive polynomials		
1	$x+1$		
2	x^2+x+1		
3	x^3+x+1	x^3+x^2+1	
4	x^4+x+1	x^4+x^3+1	
5	x^5+x^2+1	x^5+x^3+1	$x^5+x^3+x^2+x+1$
	$x^5+x^4+x^3+x+1$	$x^5+x^4+x^3+x^2+1$	$x^5+x^4+x^2+x+1$
6	x^6+x+1	x^6+x^5+1	$x^6+x^4+x^3+x+1$
	$x^6+x^5+x^2+x+1$	$x^6+x^5+x^3+x^2+1$	$x^6+x^5+x^4+x+1$

Example: Irreducible but Not Primitive

- We discovered that $p(x) = x^4 + x^3 + x^2 + x + 1$ was an irreducible polynomial, but not primitive
- Therefore, the 4-bit LFSR with the connection coefficient

$$(c_3, c_2, c_1, c_0) = (1, 1, 1, 1)$$

  is not a maximal LFSR
- Indeed, its period was 5, not 15

Example: Another Primitive Polynomial

- Since $p(x) = x^3 + x + 1$ is a primitive polynomial, the 3-bit LFSR with the connection coefficients $(c_2, c_1, c_0) = (0, 1, 1)$ produces a sequences with the maximal period 7

$c_2 = 0$	$c_1 = 1$	$c_0 = 1$	coefficients
0	0	1	(1)
1	0	0	(2)
0	1	0	(3)
1	0	1	(4)
1	1	0	(5)
1	1	1	(6)
0	1	1	(7)
0	0	1	back to (1)

Cryptographic Strength of LFSRs

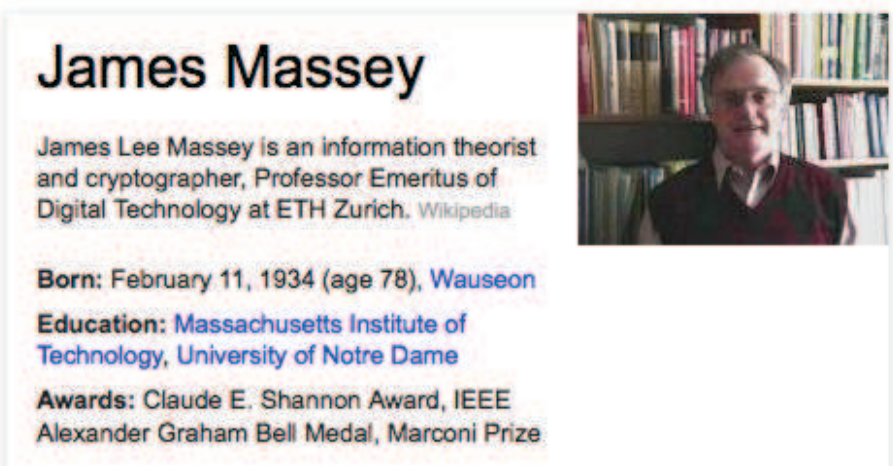
- Does the maximal LFSR satisfy requirements R1 and R2?
- An n-bit maximal LFSR scans all $2^n - 1$ states, and therefore, of all possible n-bit binary sequences, the output contains $2^n - 1$ of them
- Therefore, as long as the LFSR length is kept large ($n > 100$), and therefore, they have large period, they are statistically random
- Unfortunately, the LCGs do not satisfy R2 since they are is highly predictable, because the state values, the coefficients and the output of the LFSR are linearly related
- Similar to the discussion we had about LCGs, we need to make some assumptions about the structural parameters and the seed, and analyze the LFSRs

Cryptographic Strength of LFSRs

- LFSRs are typically implemented in hardware due to their simplicity and low-cost, for example, a 100-bit LFSR requires only a few hundred transistors
- In hardware implementations, the connection polynomial is fixed, and cannot be (generally) changed, therefore, we can only think of it as a (fixed) secret key, along with its length and the coefficients
- On the other hand, in a typical usage the LFSR starts with an initial state determined by the seed, normally, the initial state is the seed
- Furthermore, the running key bits are simply the outputs of the LFSR, which are just the state values shifted right

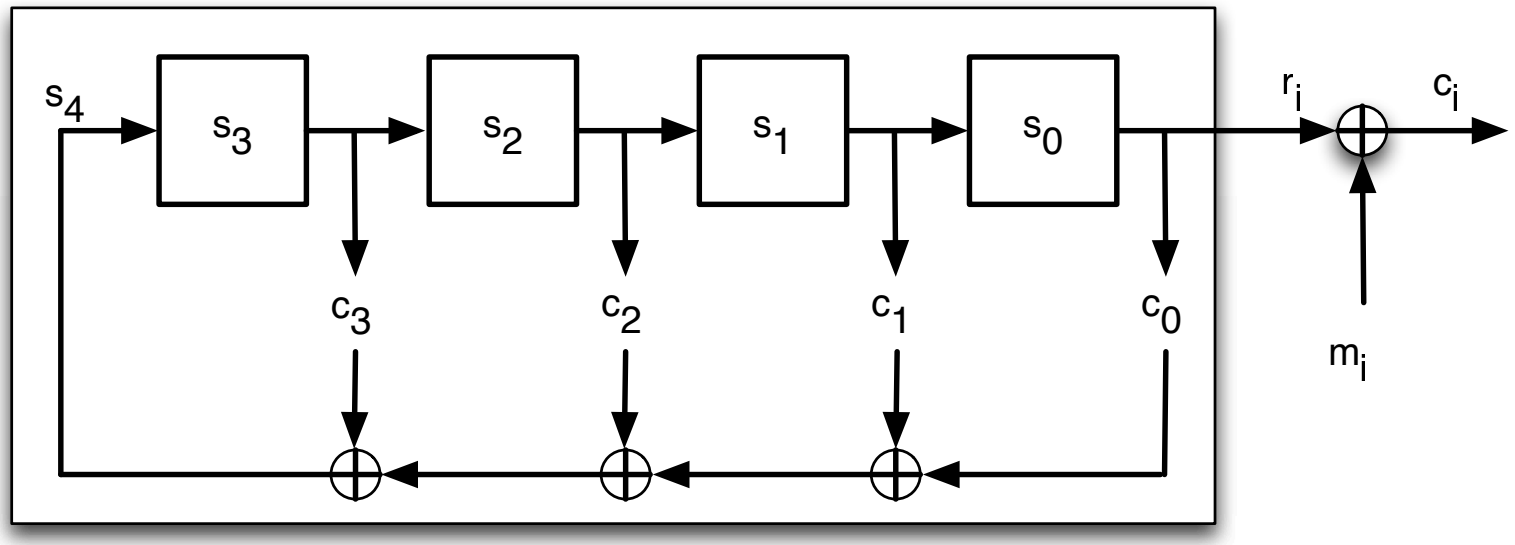
Cryptographic Strength of LFSRs

- With these assumptions, we can pose the following question: *Is it possible to compute the degree and coefficients of the connection polynomial in an LFSR, given a set of the running key bits (obtained via known or chosen text attack)?*
- The answer is affirmative: The Berlekamp-Massey algorithm, developed by two American mathematicians, solves this problem



Computing the Connection Polynomial

- First, assume we obtain a set of plaintext and ciphertext pairs (m_i, c_i), and thus, we have a set of running key bits $r_i = c_i \oplus m_i$ for $i = 0, 1, \ldots, k-1$
- Without loss of generality, also assume that these running key bits are simply the consecutive bits of the state $s_i = r_i$ for $i = 0, 1, \ldots, k-1$ at the given time, and $k > n$



Computing the Connection Polynomial

- Therefore, we can write a set of linear equations in GF(2) as

$$
\begin{aligned}
s_n &= c_{n-1}s_{n-1} + c_{n-2}s_{n-2} + \cdots + c_1 s_1 + c_0 s_0 \\
s_{n+1} &= c_n s_n + c_{n-1}s_{n-1} + \cdots + c_2 s_2 + c_1 s_1 \\
s_{n+2} &= c_{n+1}s_{n+1} + c_n s_n + \cdots + c_3 s_3 + c_2 s_2 \\
&\vdots \\
s_{2n-2} &= c_{2n-3}s_{2n-3} + c_{2n-4}s_{2n-4} + \cdots + c_{n-3}s_{n-3} + c_{n-2}s_{n-2} \\
s_{2n-1} &= c_{2n-2}s_{2n-2} + c_{2n-3}s_{2n-3} + \cdots + c_{n-4}s_{n-4} + c_{n-3}s_{n-3} \\
&\vdots
\end{aligned}
$$

- If we can obtain as many state bits s_i as we need, we can write n equations, and solve a linear system of n equations with n unknowns, which implies that we need to know $2n$ state bits: $s_0, s_1, \ldots, s_{2n-1}$

Computing the Connection Polynomial

- For example, for $n = 4$, we write the set of equations as

$$\begin{aligned} s_4 &= c_3 s_3 + c_2 s_2 + c_1 s_1 + c_0 s_0 \\ s_5 &= c_3 s_4 + c_2 s_3 + c_1 s_2 + c_0 s_1 \\ s_6 &= c_3 s_5 + c_2 s_4 + c_1 s_3 + c_0 s_1 \\ s_7 &= c_3 s_6 + c_2 s_5 + c_1 s_4 + c_0 s_1 \end{aligned}$$

- In the matrix form as

$$\begin{bmatrix} s_3 & s_2 & s_1 & s_0 \\ s_4 & s_3 & s_2 & s_1 \\ s_5 & s_4 & s_3 & s_2 \\ s_6 & s_5 & s_4 & s_3 \end{bmatrix} \begin{bmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix}$$

Computing the Connection Polynomial

- Therefore, the coefficients (c_3, c_2, c_1, c_0) of the connection polynomial in a 4-bit LFSR can be computed if we know 8 consecutive state bits $s_0, s_1, \ldots, s_7$
- For an n-bit LFSR, we need $2n$ consecutive values of the state bits in order to compute the n coefficients of the connection polynomial
- However, n is also unknown!
- This problem can be circumvented, if we have twice as many (or more) state bits available: s_i for $i = 0, 1, 2, \ldots, k-1$ such that $k \geq 2n$
- We can start with the smallest possible of n (perhaps, just $n = 1$) and iteratively increase the value of n, at each step solving a linear system of equations of dimension n

Computing the Connection Polynomial - An Example

- The following running key bit sequence (0101000010010110) was produced by an LFSR of unknown length n and an unknown connection polynomial $p(x)$
- Note that the given sequence $r_0, r_1, \ldots, r_{15}$ is of length 16, and therefore, we know the first 16 bits of the states

$$(s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10} s_{11} s_{12} s_{13} s_{14} s_{15}) = (0101000010010110)$$

- Our objective is to write the set of linear equations of dimension n for increasing values of $n = 1, 2, 3, \ldots$, until we find the correct value of n and the connection polynomial coefficients $(c_{n-1}, c_{n-2}, \ldots, c_1, c_0)$ that produces the entire sequence

Computing the Connection Polynomial

- According to our method, we will write a set of linear of equations of dimension n for increasing values of n
- We will solve these equations, and find the connection polynomial, which gives us an LFSR of length n
- We will then check to see if this LFSR correctly produces the rest of (all of) the given sequence; if not, then we need to increase n
- The Berlekamp-Massey algorithm is a slight modification of our method, where the solution of the linear equations and the checking to see if the obtained LFSR produces the rest of the sequence is more efficiently (compactly) performed

Computing the Connection Polynomial for $n = 1$

- First, we assume that there is an LFSR of length $n = 1$ produced this sequence
- However, we immediately see that $n = 1$ cannot be correct since a 1-bit LFSR can only produce either the sequence $000\cdots$ or the sequence $111\cdots$, depending whether the initial state was 0 or 1, respectively
- For $n = 1$, we write a linear system of equations with 1 unknown:

$$s_1 = c_0 \cdot s_0 \;\Rightarrow\; 1 = c_0 \cdot 0$$

There is no value of c_0 that would satisfy this equation, and therefore, this sequence (obviously) could not have been produced by a 1-bit LFSR

Computing the Connection Polynomial for $n = 2$

- For $n = 2$, we write a linear system of equations with 2 unknowns:

$$\begin{bmatrix} s_1 & s_0 \\ s_2 & s_1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} s_2 \\ s_3 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- This implies $(c_1, c_0) = (0, 1)$, however, these choices do not satisfy the rest of sequence, for example,

$$\begin{bmatrix} s_7 & s_6 \\ s_8 & s_7 \end{bmatrix} \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} s_8 \\ s_9 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \neq \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

and therefore, $n \neq 2$

Computing the Connection Polynomial for $n = 3$

- For $n = 3$, we write a linear system of equations with 3 unknowns:

$$\begin{bmatrix} s_2 & s_1 & s_0 \\ s_3 & s_2 & s_1 \\ s_4 & s_3 & s_2 \end{bmatrix} \begin{bmatrix} c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} s_3 \\ s_4 \\ s_5 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

- This implies $(c_2, c_1, c_0) = (0, 1, 0)$, however, these choices do not satisfy the rest of sequence, for example,

$$\begin{bmatrix} s_{10} & s_9 & s_8 \\ s_{11} & s_{10} & s_9 \\ s_{12} & s_{11} & s_{10} \end{bmatrix} \begin{bmatrix} c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} s_{11} \\ s_{12} \\ s_{13} \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \neq \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

and therefore, $n \neq 3$

Computing the Connection Polynomial for $n = 4$

- For $n = 4$, we write a linear system of equations with 4 unknowns:

$$\begin{bmatrix} s_3 & s_2 & s_1 & s_0 \\ s_4 & s_3 & s_2 & s_1 \\ s_5 & s_4 & s_3 & s_2 \\ s_6 & s_5 & s_4 & s_3 \end{bmatrix} \begin{bmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix}$$

- This gives

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- By solving these linear system of equations we find $(c_3, c_2, c_1, c_0) = (0, 0, 0, 0)$, which obviously is a contradiction; such LFSR cannot produce any nonzero sequence, and thus, $n \neq 4$

Computing the Connection Polynomial for $n = 5$

- For $n = 5$, we write a linear system of equations with 5 unknowns:

$$\begin{bmatrix} s_4 & s_3 & s_2 & s_1 & s_0 \\ s_5 & s_4 & s_3 & s_2 & s_1 \\ s_6 & s_5 & s_4 & s_3 & s_2 \\ s_7 & s_6 & s_5 & s_4 & s_3 \\ s_8 & s_7 & s_6 & s_5 & s_4 \end{bmatrix} \begin{bmatrix} c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} s_5 \\ s_6 \\ s_7 \\ s_8 \\ s_9 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

- By solving these equations we find $(c_4, c_3, c_2, c_1, c_0) = (0, 0, 1, 0, 1)$

Computing the Connection Polynomial for $n = 5$

- If we apply this coefficient set at another part of the sequence, we find

$$\begin{bmatrix} s_9 & s_8 & s_7 & s_6 & s_5 \\ s_{10} & s_9 & s_8 & s_7 & s_6 \\ s_{11} & s_{10} & s_9 & s_8 & s_7 \\ s_{12} & s_{11} & s_{10} & s_9 & s_8 \\ s_{13} & s_{12} & s_{11} & s_{10} & s_9 \end{bmatrix} \begin{bmatrix} c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} s_{10} \\ s_{11} \\ s_{12} \\ s_{13} \\ s_{14} \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

In fact, we can easily verify that the LFSR with $n = 5$ and the connection polynomial is $x^5 + x^2 + 1$ indeed produces the given sequence

Linear Complexity

- The Berlekamp-Masses algorithm (and our simplified version) computes the length and the connection polynomial for *any sequence*, whether or not this sequence was originally produced by an LFSR
- What we obtain is the minimal length LFSR that produces the given sequence – The length n is called **linear complexity** of the sequence
- Linear complexity is proposed as a measure of RNGs, the higher values of n imply that the RNG is more complex (less linear or more nonlinear)
- We should bear in mind, however, that the linear complexity is a measure of the given linear sequence, not the underlying RNG
- An (perfect) RNG can produce any sequence, including all zeros: $000\cdots$, whose linear complexity is 1

Linear Complexity

- As we have seen an LFSR with $n = 1$ can only produce the sequences $000\cdots$ or $111\cdots$, i.e., the linear complexity of these sequence is just 1
- Similarly, the linear complexity of the sequence $010101\cdots$ can be shown to be equal 2, regardless of its length
- The period of a sequence and its linear complexity are related, but they will not be the same
- We know that an n-bit maximal LFSR produces a sequence with period $2^n - 1$, thus, the linear complexity of such a sequence will be n while its period is $2^n - 1$
- In cryptanalysis, we cannot hope to obtain very long sequences of running keys, and thus, to discover the period (in most cases)

Linear Complexity

- Question: Given a "random" sequence of length k what is its linear complexity?
- The smallest value of linear complexity for a sequence of length will be 1 (if the sequence happens to be all-zero or all-one)
- The largest possible value of linear complexity for a sequence of length will be k
- The value of k essentially indicates our failure to find a smaller LFSR producing the sequence, and thus, we just build a k-bit LFSR and set the initial state as the bits of the given sequence, which produces the sequence by right shift in k clock cycles
- Otherwise, we will obtain a value between 1 and k

Linear Complexity

- Higher value of linear complexity does not imply randomness (statistical or unpredictability)
- It is quite easy to construct a sequence of length k whose linear complexity is k (which is the highest possible value): $k-1$ zeros followed by 1

$$\overbrace{000\cdots000}^{k-1 \text{ times}}1$$

- This sequence has the maximum possible linear complexity, however, it is not statistically random and highly predictable,
- However, randomness implies higher linear complexity; we expect a truly random source to produce sequences whose linear complexity is unbounded