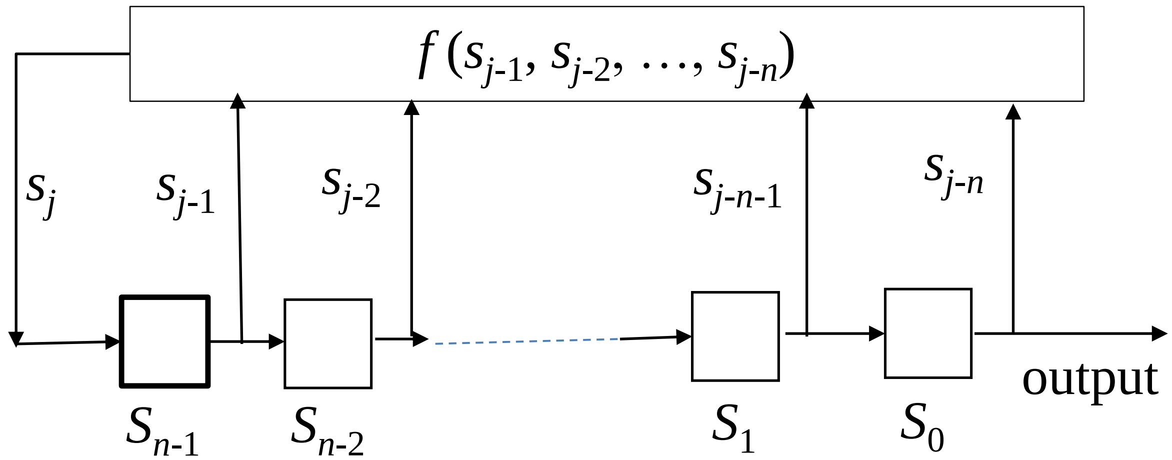
Stream Ciphers II

AK Bhateja

Shift Registers



- LFSRs of order n can generate a maximum-length of $2^n - 1$.
- A maximum-length LFSR have a characteristic polynomial that is primitive.
- Number of primitive polynomials is $\frac{\phi(2^n-1)}{n}$.
- Probability of randomly selecting a primitive polynomial is $\left(\frac{\phi(2^n-1)}{n}\right)/2^n \approx \frac{1}{n}$
- The sequence generated by such a maximum-length LFSR is called an m-sequence.
- m-sequence have good pseudo randomness properties.

Linear complexity or linear equivalence

- Definition: The linear complexity of a binary sequence $\boldsymbol{s}^N$ is the minimal possible order for a LFSR that can generate a sequence having $\boldsymbol{s}^N$ as its first N terms. Denoted $L_N(\boldsymbol{s})$ for the linear complexity of the sequence $\boldsymbol{s}$ of length N.
- For any $N \geq 1$, the linear complexity of the subsequence $\boldsymbol{s}^N$ satisfies $0 \leq L(\boldsymbol{s}^N) \leq N$.
- If $\boldsymbol{s}^N$ is the zero sequence $\boldsymbol{s}^N = 0, 0, 0, \ldots 0$, then $L(\boldsymbol{s}^N) = 0$
- If $\boldsymbol{s}^N = 0, 0, 0, \ldots 1$, then $L(\boldsymbol{s}^N) = N$
- If $\boldsymbol{s}^N$ is periodic with period p, then $L(\boldsymbol{s}^N) \leq p$.

Berlekamp-Massey algorithm

- It is an efficient algorithm for determining the linear complexity of a finite binary sequence $\boldsymbol{s}^N$ of length N.
- It outputs the minimal polynomial of $\boldsymbol{s}^N$.
- $2L$ bits is enough, where L is linear complexity of the keystream.
- Definition (next discrepancy): Consider the finite binary sequence $\boldsymbol{s}^{N+1} = s_0, s_1, \ldots, s_N$. Let an LFSR generates $\boldsymbol{s}^N = s_0, s_1, \ldots, s_{N-1}$ using polynomial $C(x) = 1 + c_1 x + \ldots + c_L x^L$. The next discrepancy d_N is the difference between s_N and the $(N + 1)^{st}$ term generated by the LFSR is

$$d_N = \left(s_N + \sum_{i=1}^{L} c_i s_{N-i} \right) \bmod 2$$

Theorem: Let $\boldsymbol{s}^N$ be a finite binary sequence of linear complexity $L(\boldsymbol{s}^N)$, and let $\langle L(\boldsymbol{s}^N), C(x)\rangle$ be an LFSR which generates $\boldsymbol{s}^N$.

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

b) If $d_N = 0$, then $L(\boldsymbol{s}^{N+1}) = L$.

c) If $d_N = 1$. Let m the largest integer $< N$ such that $L(\boldsymbol{s}^m) < L(\boldsymbol{s}^N)$, and let $\langle L(\boldsymbol{s}^m), B(x)\rangle$ be an LFSR of length $L(\boldsymbol{s}^m)$ which generates $\boldsymbol{s}^m$. Then $(L', C'(x))$ is an LFSR of smallest length which generates $\boldsymbol{s}^{N+1}$, where

$$L' = \begin{cases} L & \text{if } L > N/2 \\ N+1-L & \text{if } L \leq N/2 \end{cases}$$

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

Reference: James L. Massey, “Shift-Register Synthesis and BCH”, IEEE Transactions on Information Theory, vol-IT, No. 1, Jan 1969.

Berlekamp-Massey algorithm

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

OUTPUT: the linear complexity $L(\boldsymbol{s}^n)$ of $\boldsymbol{s}^n$, $0 \leq L(\boldsymbol{s}^n) \leq n$.

$C(x) = 1$, $L = 0$, $m = -1$, $B(x) = 1$, $N = 0$.

While ($N < n$)

compute the next discrepancy $d = \left(s_N + \sum_{i=1}^{L} c_i s_{N-i}\right) \bmod 2$

if $d = 1$ then do

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

If $L \leq N/2$ then $L = N + 1 - L$, $m = N$, $B(x) = T(x)$

$N = N + 1$.

return (L)

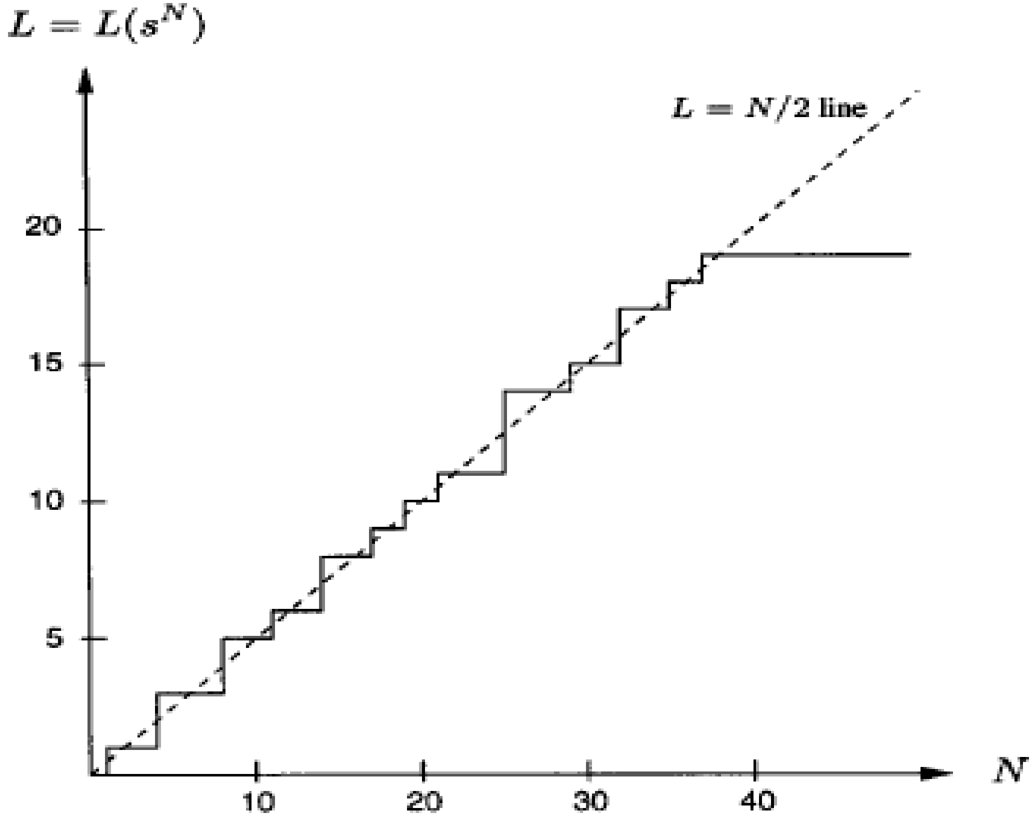
Example: Reconstruct a minimum order LFSR which generates the sequence 001110110 of length 9.

s_N	d	$T(x)$	$C(x)$	L	m	$B(x)$	N
-	-	-	1	0	-1	1	0
0	0	-	1	0	-1	1	1
0	0	-	1	0	-1	1	2
1	1	1	$1 + x^3$	3	2	1	3
1	1	$1 + x^3$	$1 + x + x^3$	3	2	1	4
0	1	$1 + x + x^3$	$1 + x + x^2 + x^3$	3	2	1	5
1	1	$1 + x + x^2 + x^3$	$1 + x + x^2$	3	2	1	6
1	0	$1 + x + x^2 + x^3$	$1 + x + x^2$	3	2	1	7
1	1	$1 + x + x^2$	$1 + x + x^2 + x^5$	5	7	$1 + x + x^2$	8
0	1	$1 + x + x^2 + x^5$	$1 + x^3 + x^5$	5	7	$1 + x + x^2$	9

This sequence is found to have linear complexity 5, and the polynomial is $1 + x^3 + x^5$.

Linear Complexity Profile

- Definition: Let $\boldsymbol{s} = s_0, s_1, \ldots$ be a binary sequence, and let L_N denote the linear complexity of the subsequence $\boldsymbol{s}^N = s_0, s_1, \ldots, s_{N-1}$, $N \geq 0$. The sequence $L_1, L_2, \ldots$ is called the linear complexity profile of $\boldsymbol{s}$.



Linear complexity profile of a random sequence with L.C. = 20

Are LC and LCP good?

- Both are good
- However, there are certain predictable sequences that can be detected by neither of them
- Example: The following sequence has perfect LCP, yet it is predictable

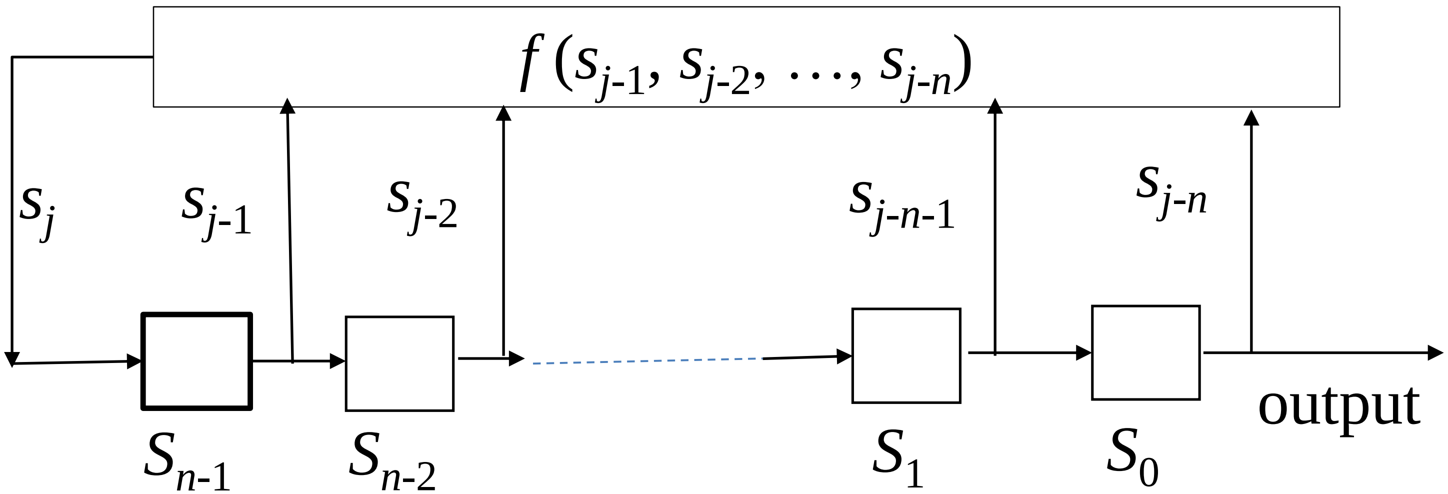
$$s_i = \begin{cases} 1 \text{ if } i = 2^j - 1 \text{ for some } j \geq 0 \\ 0 \qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

- Use LC and LCP together with other statistical or complexity tests.

Weaknesses of LFSR

- Cryptanalytic attack is very dangerous if keystream is ever repeated
  - Self-cancellation property of XOR: $x \oplus x = 0$
  - $(m_1 \oplus \text{key stream}) \oplus (m_2 \oplus \text{keystream}) = m_1 \oplus m_2$
  - If attacker knows m_1, then easily recovers m_2
- Known plaintext attack
  - If the interceptor knows the plaintext equivalent of $m \geq 2n$ consecutive positions of ciphertext of n-stage LFSR based stream cipher, then plaintext can be easily recovered.

Nonlinear feedback shift registers



- Shift registers with non-linear feedback function.
- NLFSRs are known to be more resistant to cryptanalytic attacks than Linear Feedback Shift Registers.
- NLFSR can produce sequences with much higher linear complexity than LFSRs.
- $f\,(x_1, x_2, x_3\ , x_4) = 1 \oplus x_2 \oplus x_3 \oplus x_4 x_5 \oplus x_1 x_3 x_4 x_5$ has nonlinear order 4.
- The number of feedback functions for an n-stage shift register is 2^{2^n}.

- Truth table for $n = 3$

S. No.	Input			Output
	s_0	s_1	s_2	s_3
1	0	0	0	1
2	0	0	1	1
3	0	1	0	0
4	0	1	1	1
5	1	0	0	0
6	1	0	1	0
7	1	1	0	1
8	1	1	1	0

- The function $f(s_0, s_1, s_2)$ which describes the truth table:

$$f(s_0, s_1, s_2) = (s_0 + 1)(s_1 + 1)(s_2 + 1) \cdot 1 + (s_0 + 1)(s_1 + 1)s_2 \cdot 1 + (s_0 + 1) s_1 (s_2 + 1) \cdot 0 + (s_0 + 1) s_1 s_2 \cdot 1 + s_0 (s_1 + 1)(s_2 + 1) \cdot 0 + s_0 (s_1 + 1) s_2 \cdot 0 + s_0 s_1 (s_2 + 1) \cdot 1 + s_0 s_1 s_2 \cdot 0$$

$$= 1 + s_0 + s_1 + s_1 s_2$$

S. No.	Input			Output
	s_0	s_1	s_2	s_3
1	0	0	0	1
2	0	0	1	1
3	0	1	0	0
4	0	1	1	1
5	1	0	0	0
6	1	0	1	0
7	1	1	0	1
8	1	1	1	0

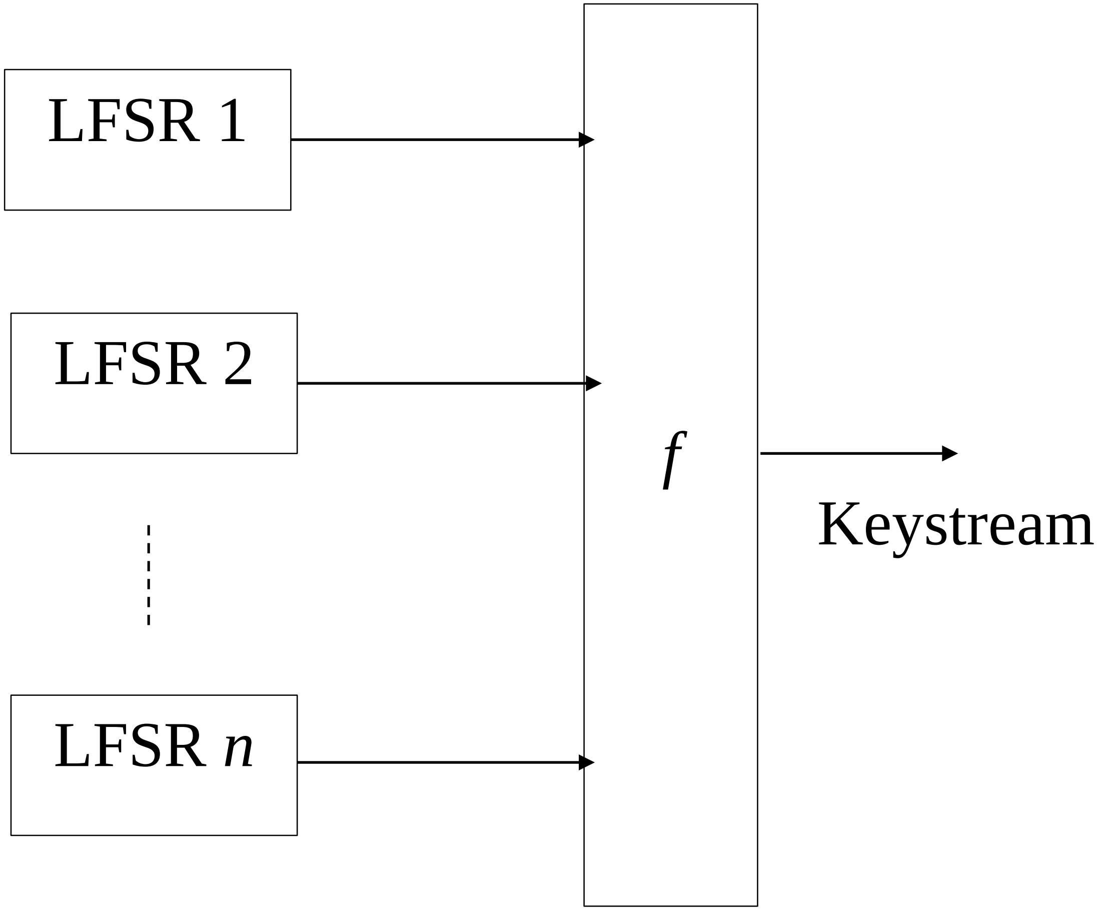
$$f(s_0, s_1, s_2) = 1 + s_0 + s_1 + s_1 s_2$$

- Every truth table give a unique feedback function
- No two functions can give the same truth table
- Thus the number of feedback functions for *n*-stage shift register is the same as the number of truth tables
- A truth table is completely determined by its last column
- Since a truth table has 2^n rows, last column has 2^n positions & for each one, there are two possible entries (0 or 1). Thus the number of feedback functions for an *n*-stage shift register is 2^{2^n}.

Nonlinear combination generators

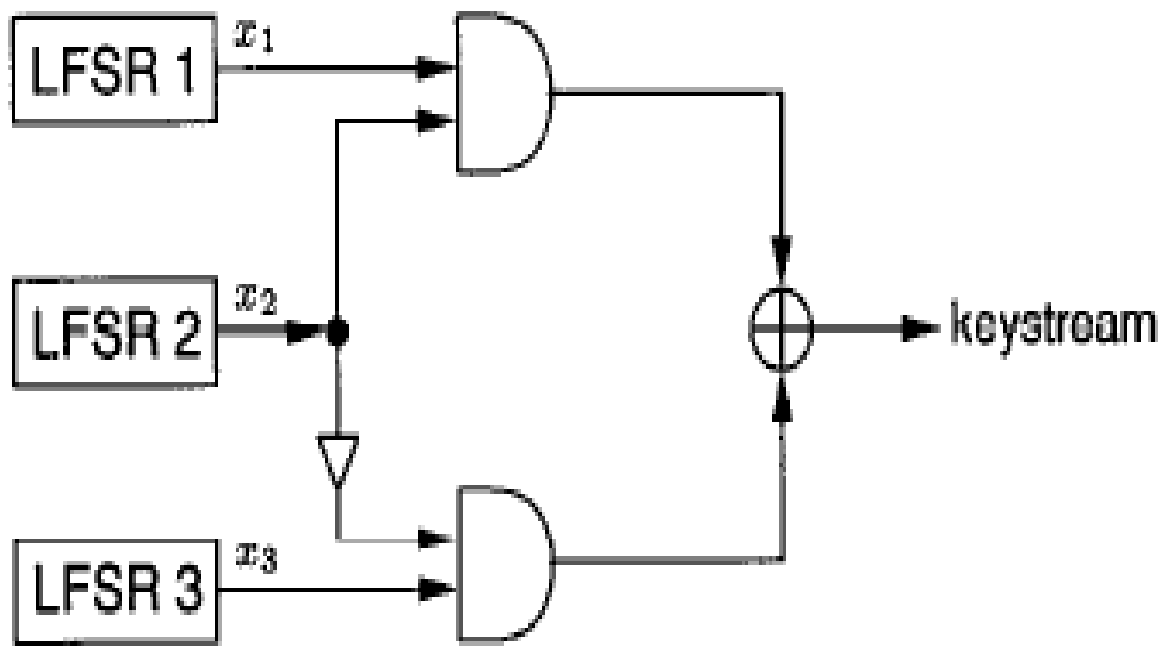
- One general technique for destroying the linearity inherent in LFSRs is to use several LFSRs in parallel.
- The keystream is generated as a nonlinear function f of the outputs of the component LFSRs.
- Such keystream generators are called nonlinear combination generators.
- Function f is called the combining function.

Nonlinear combination generators



Geffe generator

- It has three maximum-length LFSRs whose lengths n_1, n_2, n_3 are pairwise relatively prime, with nonlinear combining function
- $f(x_1, x_2, x_3) = x_1\, x_2 \oplus (1 + x_2)\, x_3 = x_1\, x_2 \oplus x_2\, x_3 \oplus x_3$



Period and LC of Geffe Generator

- The keystream generated has period $(2^{n_1} - 1) \cdot (2^{n_2} - 1) \cdot (2^{n_3} - 1)$

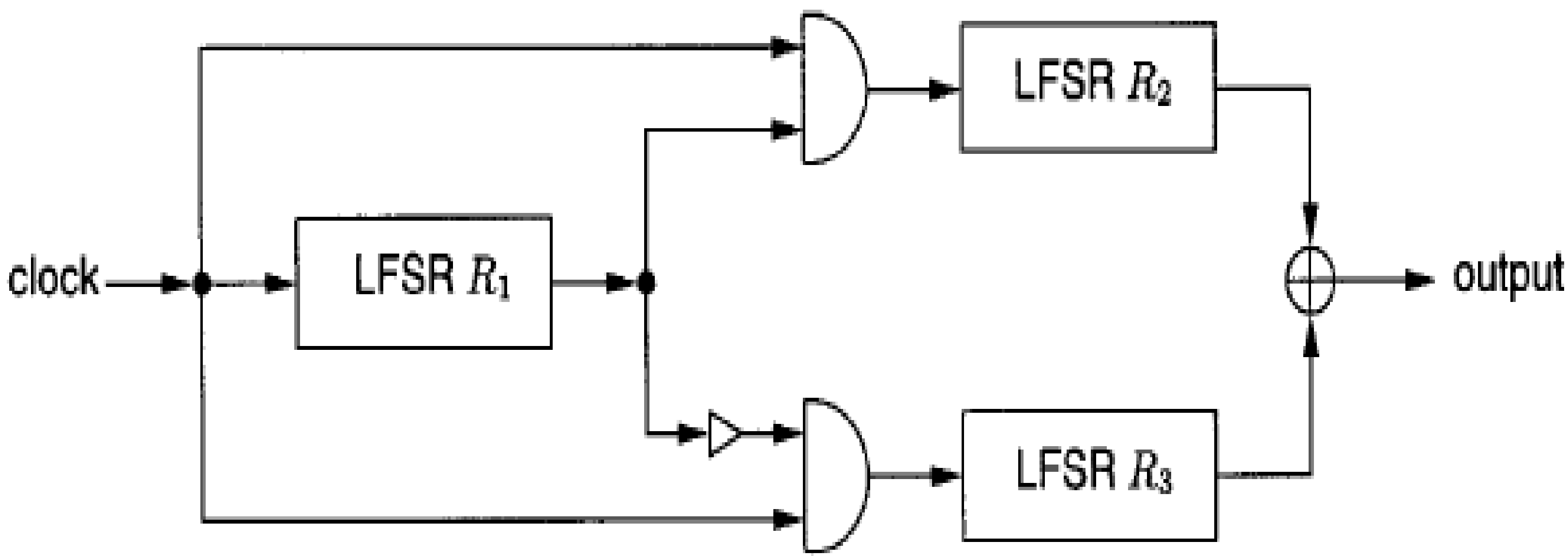
- Linear complexity L = $n_1 n_2 + n_2 n_3 + n_3$.

Clock Controlled generators

- The component LFSRs are clocked regularly; i.e., the movement of data in all the LFSRs is controlled by the same clock.
- The main idea behind a clock-controlled generator is to introduce nonlinearity into LFSR-based keystream generators by having the output of one LFSR control the clocking (i.e., stepping) of the other LFSRs
- Since the other LFSR clocked in an irregular manner, the attacks based on the regular motion of LFSRs gets foiled.
- Some clock controlled generators
  - The alternating step generator (ASG)
  - Shrinking Generator (SG)
  - Self-Shrinking generator (SSG)

The alternating step generator (ASG)

- A clock controlled generator
- It uses three LFSRs (R_1, R_2 & R_3)
- LFSR R_1 to control the clocking / stepping of two LFSRs, R_2 and R_3.
- The keystream produced is the XOR of the output sequences of R_2 and R_3.



Algorithm: Alternating step generator

Register R_1 is clocked.

If the output of R_1 is 1 then

R_2 is clocked; R_3 is not clocked but its previous output bit is repeated.

(For the first clock cycle, the previous output bit of R_3 is taken to be 0)

If the output of R_1 is 0 then

R_3 is clocked; R_2 is not clocked but its previous output bit is repeated.

(For the first clock cycle, the previous output bit of R_2 is taken to be 0)

The output bits of R_2 and R_3 are XORed; the resulting bit is part of the keystream.

Example:

LFSRs R_1: Initial contents: 100, Polynomial: $1 + x^2 + x^3$

LFSRs R_2: Initial contents: 1101, Polynomial: $1 + x^3 + x^4$

LFSRs R_3: Initial contents: 10010,

Polynomial: $1 + x + x^3 + x^4 + x^5$

Sequence generated by R_1: 1001011 period 7

Sequence generated by R_2: 110101111000100 period 15

Sequence generated by R_3:

1001010110000111001101111101000 period 31

Output bits of R_2 : 1 1 1 1 1 0 1 0 0 …

Output bits of R_3 : 0 1 0 0 0 0 0 0 1 …

The resulting bits: 1 0 1 1 1 0 1 0 1 …

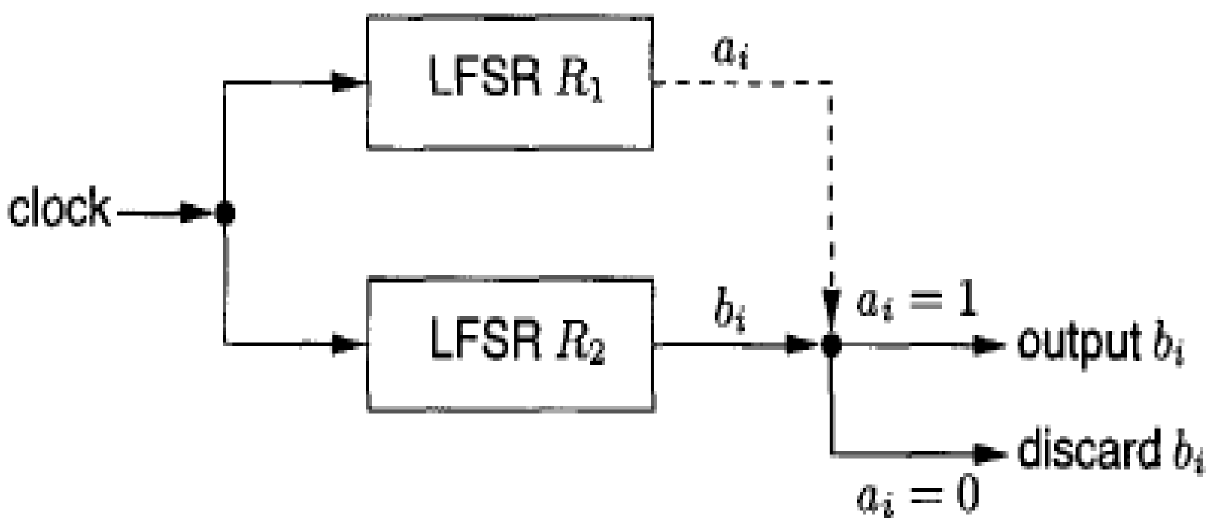
Period & Linear Complexity of ASG

- Suppose R_1, R_2 & R_3 are maximum length LFSRs of length n_1, n_2 and n_3 respectively such that gcd $(n_2, n_3) = 1$.
- The period of the sequence ($\boldsymbol{s}$) generated by ASG is $(2^{n_1} - 1) \cdot (2^{n_2} - 1) \cdot (2^{n_3} - 1)$
- The linear complexity of ASG satisfies
- $(n_2 + n_3)2^{n_1 - 1} < L(s) < (n_2 + n_3)2^{n_1}$

Reference: Günther, Christoph G. "Alternating step generators controlled by de Bruijn sequences" Workshop on the Theory and Application of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1987.

Shrinking Generator (SG)

- A control LFSR R_1 is used to control the output of a second LFSR R_2 and select a portion of the output sequence of a second LFSR R_2
- Both R_1 and R_2 are clocked
- If the output of R_1 is 1, the output bit of R_2 forms part of the keystream.
- If the output of R_1 is 0, the output bit of R_2 is discarded.



Example: Shrinking Generator

LFSRs R_1: Initial contents: 001, Polynomial: $1 + x + x^3$

LFSRs R_2: Initial contents: 10100, Polynomial: $1 + x^3 + x^5$

Sequence generated by R_1 is

0011101

Sequence generated by R_2 is

1010000100101100111110001101110

The keystream generated by SG is

1000010111101110 ....

Period & LC of Shrinking Generator

- Let R_1 (selector) and R_2 form a shrinking generator. If both R_1 and R_2 are maximal length (have primitive feedback polynomials of order n_1, n_2 respectively) and $\gcd(2^{n_1} - 1, 2^{n_2} - 1) = 1$, then the shrunken sequence has
  - period $(2^{n_2} - 1) \cdot 2^{n_1 - 1}$
  - linear complexity $n_2 \cdot 2^{n_1 - 1}$

Reference: Coppersmith, Don, Hugo Krawczyk, and Yishay Mansour. "The shrinking generator." Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 1993.

Self-Shrinking generator (SSG)

- A self-shrinking generator is a pseudorandom generator that is based on the shrinking generator concept.
- It uses one LFSR and alternating output bits of the single register is used to control its final output.
- The procedure for clocking SSG
  - Clock the LFSR twice to obtain a pair of bits as LFSR output
  - If the pair is 10 output a zero
  - If the pair is 11 output a one
  - Otherwise, output nothing.

Example: Consider LFSR of degree 8

Initial contents ***s*** = 0 1 1 0 1 1 0 1

Primitive polynomial: $1 + x^2 + x^3 + x^4 + x^8$

Time	State of LFSR	Output	SSG bit
0	0 1 1 0 1 1 0 1	-	-
1	1 1 0 1 1 0 1 1	0	-
2	1 0 1 1 0 1 1 1	1	
3	0 1 1 0 1 1 1 1	1	0
4	1 1 0 1 1 1 1 1	0	
5	1 0 1 1 1 1 1 0	1	1
6	0 1 1 1 1 1 0 0	1	

Sequence generated by SSG is: 0 1 1 0

RC4 (Rivest Cipher 4)

- RC4 was designed in 1987 by Ron Rivest and is one of the most widely software stream cipher
- It was proprietary cipher owned by RSA Data Security Inc (RSADSI)
- Became public in 1994
- Simple and effective design
- Variable key size, byte-oriented stream cipher
- RC4 is a symmetric key cipher and bite-oriented algorithm
- Widely used (web SSL/TLS, wireless WEP)
- It’s considered fast and simple in terms of software
- However, the simplicity of RC4 makes it vulnerable to different security attacks.

RC4: Basic Constituents

- Key scheduling algorithm (KSA)
  - A simple scrambling of input keystream and the initial state is performed
  - KSA generates initial permutation S of $\{0, \cdots, N-1\}$
  - First, the array S is initialized to the identity permutation and then mixes bytes of the key within it.
- Pseudo Random-number Generation Algorithm (PRGA)
  - A pseudorandom output byte sequence is generated from internal permutation

Algorithm: Key Scheduling

for i from 0 to $N - 1$

S[i] = i

$j = 0$

for i from 0 to $N - 1$

j = (j + S[i] + key[i mod keylength]) mod N

swap(S[i], S[j])

Algorithm: Pseudo Random Generation

$i = j = 0$

while Generating_Output key stream

i = (i + 1) mod N

j = (j + S[i]) mod N

Swap (S[i], S[j])

Output = S[(S[i] + S[j]) mod N]

RC4

- Many stream ciphers are based on LFSRs, which are efficient in hardware but less efficient in software.
- The design of RC4 avoids the use of LFSRs, and is ideal for software implementation, as it requires only byte manipulations.
- It uses
  - 256 bytes (for $N = 256$) of memory for the array, $S[0]$ through $S[255]$,
  - l bytes of memory for the key, key[0] through key[l-1],
  - integer variables, i, j.