

EWSCS'06

Palmse, Estonia
5-10 March 2006

Lecture 3: **Secret Sharing
and Linear Complexity**

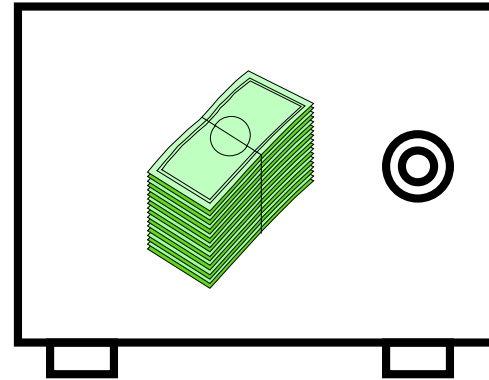
James L. Massey

Prof.-em. ETH Zürich, Adjunct Prof., Lund Univ.,
Sweden, and Tech. Univ. of Denmark
Trondhjemsgade 3, 2TH
DK-2100 Copenhagen East

JamesMassey@compuserve.com

Secret Sharing

The "classical" way that two crooks (or two bank vice presidents), who do not trust one another, can share a secret.



The **secret**:

1 0 0 1 0 1 1 0 0 1

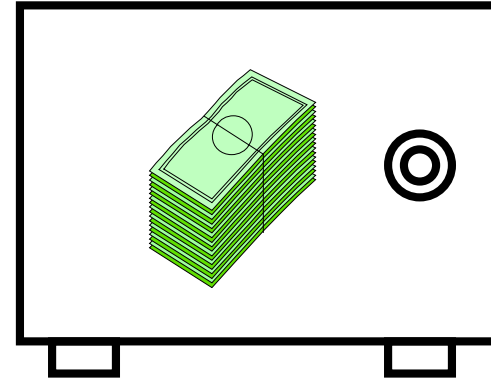
The
shares

1 0 0 1 0

1 1 0 0 1

The **secret "leaks out"**—one share is not worthless!

No-Leak Secret Sharing



The **secret**

1 0 0 1 0 1 1 0 0 1

Share 1:

BSS output

0 0 1 1 0 1 0 1 1 1

Share 2:

secret \oplus BSS output

1 0 1 0 0 0 1 1 1 0

No leakage!

(Share 2 is a Vernam encryption of the secret.)

The two crooks (or two bank vice presidents) in our example were using the (3,2) linear code over $GF(2^{10})$, in which the "symbols" are binary 10-tuples, whose generator matrix and parity-check matrix are

$$G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \text{ and } H = [1 \ 1 \ 1]$$

This code has $d_{\min} = 2 = n-k+1$ and is thus an **MDS code**. The **only non-zero codeword with first component 1** in the **dual code** is $[1 \ 1 \ 1]$, which specifies that in a codeword $[v_1 \ v_2 \ v_3]$ one has

$$v_1 + v_2 + v_3 = 0$$

so that the secret v_1 can be obtained from the shares v_2 and v_3 as $v_1 = v_2 + v_3$.

The idea of secret sharing is due independently to Shamir and to Blakley.

A. Shamir, "How to share a secret", *Communications of the ACM*, vol. 22, pp. 612-613, November 1979.

G. R. Blakley, "Safeguarding cryptographic keys", Proc. AFIPS Natl. Conf., pp. 313-317, 1979.

Shamir showed how a secret can be "divided" into N shares so that **any T shares uniquely determine the secret**, but **any $T - 1$ or fewer shares give no information about the secret**.

This kind of secret-sharing scheme is called a **threshold** scheme and T is called the threshold.

In his introduction of "perfect" secret sharing for a threshold access structure, Shamir actually reinvented the **Reed-Solomon codes** !

The connection to Reed-Solomon codes was made explicitly in R. J. McEliece and D. V. Sarwate, "On sharing secrets and Reed-Solomon codes", *Comm. ACM*, Vol. 24, pp. 583-584, September 1981.

We now show that linear codes can be used to give a convenient description of perfect secret sharing for a general access structure.

The assumption as to how a q -ary linear (n,k) code is used to share a secret:

- the secret is an **arbitrary** element of $GF(q)$
- the "dealer" selects the codeword **uniformly at random** from the q^{k-1} codewords whose **first digit is the secret**
- the $n - 1$ **shares** are the codeword digits in coordinates 2 through n (together with the designations of the coordinate).

How do shares determine the secret?

If we can solve for the **secret** (v_1) as a linear combination of some **shares**, i.e., if

$$v_1 = c_2 v_2 + c_3 v_3 + \dots + c_n v_n$$

$$0 = v_1 - c_2 v_2 - c_3 v_3 + \dots - c_n v_n$$

then $[1 \ -c_2 \ -c_3 \ \dots \ -c_n]$ is a codeword in the **dual code** having a 1 in the first coordinate.

Is there any other way to use the shares to determine the secret?

NO! Because the original code is a linear code, an n -tuple is a codeword if and only if it satisfies all the parity checks of the code. Thus, **linear combinations completely exhaust the manner in which the secret can be determined from the shares.**

It follows that the codewords in the **dual code** with a 1 in the first position completely determine which sets of shares can be used to find the secret.

An information set in a q -ary (n,k) code is a set of k coordinates in which every one of the q^k possible k -tuple values appears in exactly one codeword.

The idea is that the digits in the k positions of an information set can be used as the **information symbols** in the code.

A set of k positions is an information set in a linear (n,k) code with generator matrix G if and only if the corresponding k columns of G are linearly independent.

A set of k positions is an information set in a linear (n,k) code with parity-check matrix H if and only if the $n-k$ columns of H not corresponding to positions in the information set are linearly independent.

Recall that an (n, k) code is maximum-distance separable (MDS) if its minimum distance satisfies $d_{\min} = n - k + 1$.

Some properties of a linear (n, k) MDS code.

- Every choice of k positions is an information set.
- For every choice of $n - k + 1$ coordinates, there is a codeword that is non-zero in these and only these coordinates.
- The dual code is a linear $(n, n - k)$ MDS code.

N.B. Reed-Solomon codes are linear MDS codes.

Proposition: A q -ary linear (n, k) **MDS** code gives a “perfect” threshold secret-sharing system in which every collection of $T = k$ shares uniquely determine the secret and no collection of $T-1$ or fewer shares gives any information about the secret.

Proof: The dual of a q -ary linear (n, k) **MDS** code is a q -ary linear $(n, n-k)$ **MDS** code so that $d^\perp = k+1$. The array of codewords in the (n, k) **MDS** code is an orthogonal array of power $t = k$. Thus, with the first digit fixed, the digits in any $k-1$ positions are uniformly random and can give no information about the secret.

Conversely, the first coordinate together with any k other coordinates correspond to a codeword in the dual code. Hence the digit in the first coordinate is uniquely determined by the k other digits.

A codeword $[v_1, v_2, \dots, v_n]$ in a q -ary (n, k) code is minimal if it is non-zero, if its leftmost non-zero component is a 1, and if its **non-zero coordinates cover all the non-zero coordinates of no other codeword whose leftmost non-zero component is a 1.**

The minimal codewords in the dual code with first component equal to 1 correspond exactly to the minimal sets of shares that determine the secret.

Non-threshold access structures for secret sharing:

The **access structure** is the set of collections of shares such that

- the shares in each collection uniquely determine the secret but do not do this if some share is removed.
- any collection of shares whose coordinates do not cover a set in the **access structure** gives no information about the secret. (This is sometimes called perfect secret sharing.)

Proposition: The access structure for perfect secret sharing using a q -ary linear (n, k) code is the set of collections of shares corresponding to the minimal codewords in the dual code whose first component is a 1.

Example: Consider the 2^m -ary $(5, 3)$ code with **generator matrix**

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

which has the **parity-check matrix**

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

The minimal codewords in the dual code are

1 1 1 0 0, **1 0 1 1 1** and **0 1 0 1 1**.

Thus, the access structure consists of the two sets of coordinates $\{2, 3\}$ and $\{3, 4, 5\}$. A collection of shares determines the secret if and only if it contains one of these two minimal access sets.

Linear Complexity in Cryptography

Let

$$\mathbf{s} = s_0, s_1, s_2, s_3, \dots, s_{n-1}$$

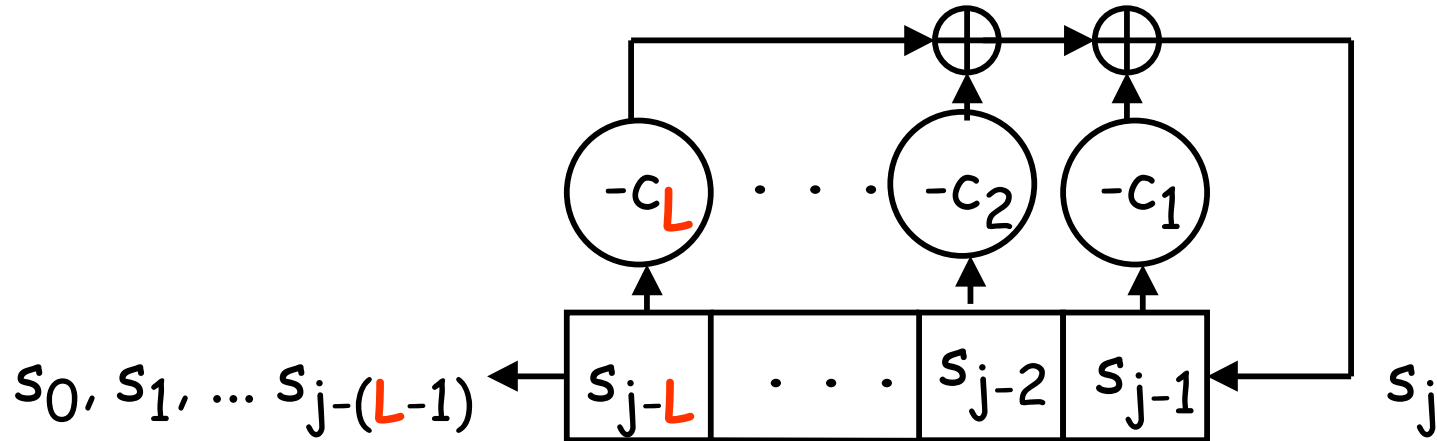
be a sequence of length n over a field F . We allow $n = \infty$, in which case \mathbf{s} is a **semi-infinite** sequence over a **field F**

The linear complexity of \mathbf{s} is the smallest L such that there exist c_1, c_2, \dots, c_L for which

$$s_j + c_1 s_{j-1} + \dots + c_{L-1} s_{j-(L-1)} + c_L s_{j-L} = 0, \text{ all } L \leq j < n.$$

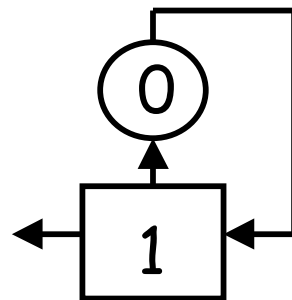
Note that c_L, c_{L-1}, \dots, c_1 are allowed to be 0.

Linear-Feedback Shift Register (LFSR) interpretation:

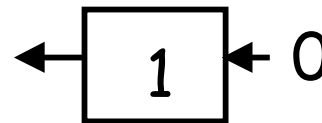


The **linear complexity** of a sequence (finite or semi-infinite) is the length L of the **shortest LFSR** that, for some initial state, produces the sequence as its output.

Example: The semi-infinite sequence $1, 0, 0, 0, \dots$ has linear complexity 1. The shortest LFSR, shown with its initial state, is



equivalently



Note that $c_L = c_1 = 0$.

Extremes of linear complexity

The **all-zero sequence** of length n , $n > 0$, is the only **sequence of length n having linear complexity 0**.

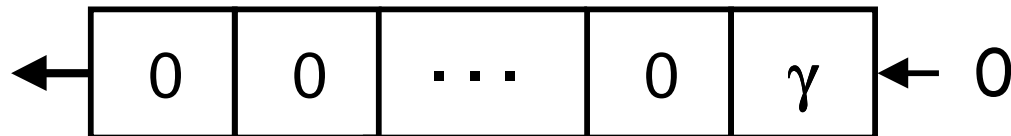
Reason: If $L = 0$, then the equation

$$s_j + c_1 s_{j-1} + \dots + c_{L-1} s_{j-(L-1)} + c_L s_{j-L} = 0, \text{ all } L \leq j < n.$$

becomes simply $s_j = 0$, all $0 \leq j < n$.

The finite sequence

$s = s_0, s_1, s_2, \dots, s_{n-2}, s_{n-1} = 0, 0, 0, \dots, 0, \gamma$
where $\gamma \neq 0$ has linear complexity n .

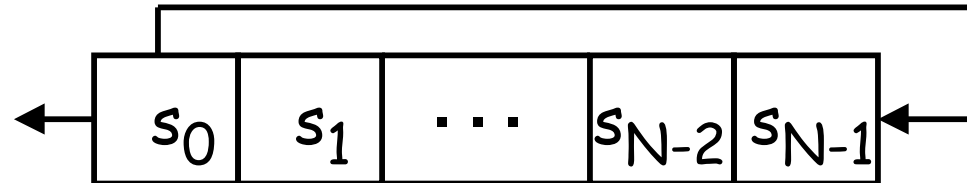


A finite sequence of length n has linear complexity L satisfying $0 \leq L \leq n$, with equality on the left only for the all-zero sequence and with equality on the right only for a sequence of $n-1$ zeroes followed by a nonzero digit.

s is **N-periodic** if s is semi-infinite and, for the positive integer **N**,

$$s_i = s_{i+N} \text{ for all } i \geq 0.$$

The linear complexity of an **N-periodic** sequence is at most **N**.



If the semi-infinite sequence s is **N-periodic**, then its linear complexity is the same as the linear complexity of the first $2N$ digits of the sequence.

Example: The finite sequence 1, 0 has linear complexity 1. The **2-periodic** sequence 1, 0, 1, 0, 1, 0, ... has linear complexity 2, as does the finite sequence 1, 0, 1, 0.

We describe an LFSR by the pair $\langle C(D), L \rangle$ where L is the length of the register and where

$$C(D) = 1 + c_1D + \dots + c_L D^L$$

is the **connection polynomial**.

$$\text{N.B.: Always } C(0) = 1 \text{ and } \deg[C(D)] \leq L.$$

Our previous "boxed" equation shows that

the LFSR $\langle C(D), L \rangle$ generates the semi-infinite sequence s if and only if

$$S(D)C(D) = P(D)$$

where $\deg[P(D)] < L$

and where $S(D) = s_0 + s_1D + s_2D^2 + \dots$
is the **power series** associated with s .

From $S(D)C(D) = P(D) = p_0 + p_1 + \dots + p_{L-1} D^{L-1}$,
we have

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ c_1 & 1 & 0 & \dots & 0 \\ c_2 & c_1 & 1 & \dots & 0 \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \\ c_{L-1} & c_{L-2} & c_{L-3} & \dots & 1 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ \cdot \\ \cdot \\ \cdot \\ s_{L-1} \end{bmatrix} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ \cdot \\ \cdot \\ \cdot \\ p_{L-1} \end{bmatrix}$$

It follows that

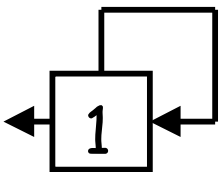
for every $P(D)$ of degree less than L , there is a unique corresponding initial state s_0, s_1, \dots, s_{L-1} of the LFSR, and conversely.

It follows from our analysis that:

The LFSR $\langle C(D), L \rangle$ generates the semi-infinite sequence s if and only if $S(D)C(D) = P(D)$ where $\deg[P(D)] < L$.
 Moreover, such $\langle C(D), L \rangle$ is the shortest LFSR that generates s , i.e., s has linear complexity L , if and only if $\gcd[C(D), P(D)] = 1$.

Example: $S(D) = 1 + D + D^2 + \dots$

$$\Rightarrow \underbrace{(1 - D)}_{C(D)} S(D) = \underbrace{1}_{P(D)}$$

$\Rightarrow L(s) = 1$ and  is the shortest generating LFSR.

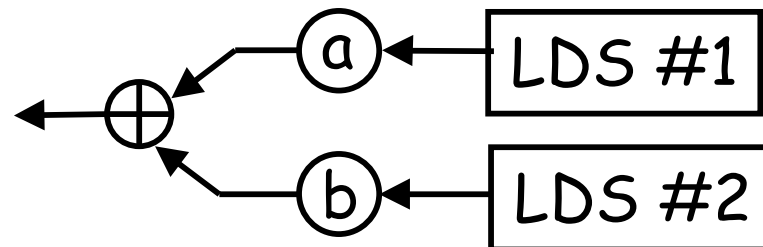
Fundamental Property of Linear Complexity:

The **linear complexity** of a sequence is the smallest number of delay cells in (or equivalently the minimum dimension of) a linear discrete-time system (LDS) which, for some choice of initial state, has this sequence as (a prefix of) its autonomous (i.e., no input) response.

One simple consequence of this fact: The **linear complexity** of a **linear combination of sequences** is **at most the sum of their linear complexities.**

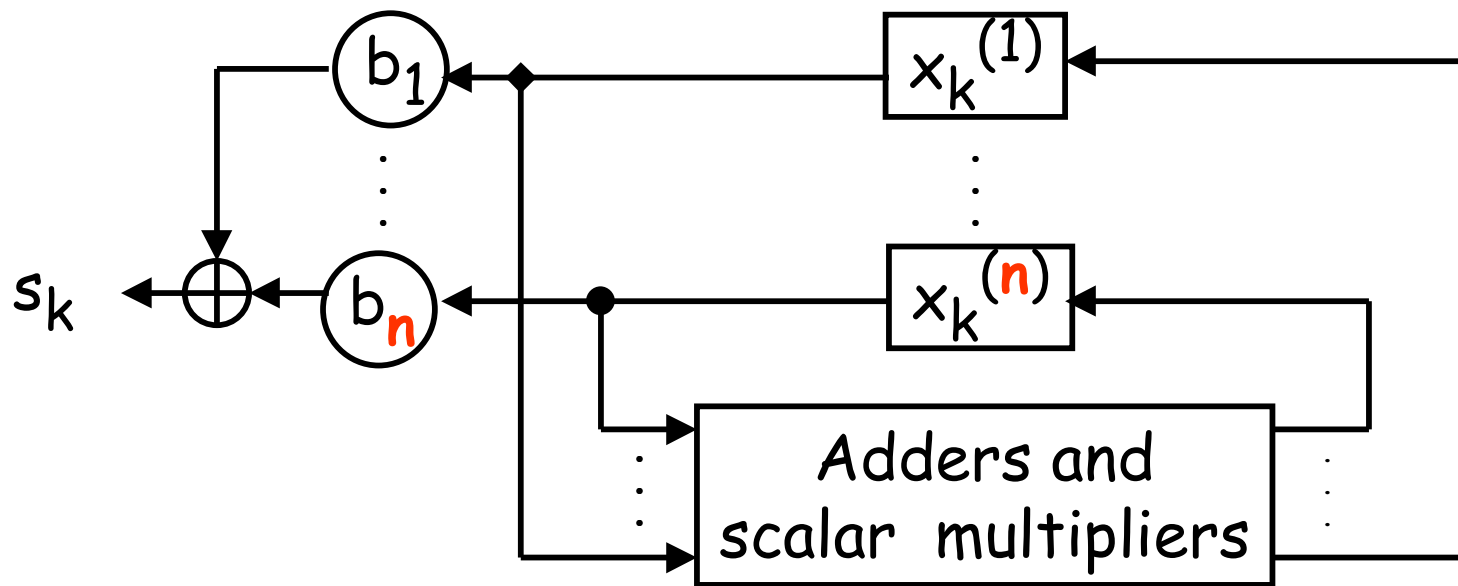
Proof:

Linear combination
of two sequences



Proof of fundamental property:

Suppose $\mathbf{s} = s_0, s_1, s_2, s_3, \dots$ can be produced by an autonomous LDS with n delay cells, i.e.,



where $\mathbf{x}_k = (x_k^{(1)}, \dots, x_k^{(n)})$ is the state at time k .

$\Rightarrow \mathbf{x}_k = A^k \mathbf{x}_0$ where \mathbf{x}_0 is the initial state.

Thus, $s_k = \mathbf{b}^T A^k \mathbf{x}_0$ where $\mathbf{b} = (b_1, \dots, b_n)$.

$$\begin{aligned}\Rightarrow \mathbf{S}(D) &= s_0 + s_1 D + s_2 D^2 + \dots \\ &= \mathbf{b}^T (\mathbf{I} + A D + A^2 D^2 + \dots) \mathbf{x}_0 \\ &= \mathbf{b}^T (\mathbf{I} - A D)^{-1} \mathbf{x}_0 \\ &= \frac{\mathbf{P}(D)}{\mathbf{C}(D)}\end{aligned}$$

where $\mathbf{C}(D) = \det(\mathbf{I} - A D) \Rightarrow \boxed{\mathbf{C}(0) = 1.}$

The entries of the $n \times n$ matrix $\mathbf{I} - A D$ are polynomials of degree at most 1. $\Rightarrow \boxed{\deg[\mathbf{C}(D)] \leq n.}$

Moreover, $\mathbf{P}(D) = \mathbf{b}^T \mathbf{M}(D) \mathbf{x}_0$ where $\mathbf{M}(D)$ is the transpose of the matrix of cofactors $[n-1 \times n-1$ matrices] of $\mathbf{I} - A D$, $\Rightarrow \boxed{\deg[\mathbf{P}(D)] < n.}$ □

There is an efficient [$O(n^2)$] algorithm, the "**LFSR synthesis algorithm**", also called the "**Berlekamp-Massey (BM) algorithm**," that solves the problem, for a sequence of length n , of finding the linear complexity and a shortest LFSR generator for the sequence (and, if desired, all of its prefixes).

This is a further reason that the terminology "linear complexity" is justified.

Length-change Lemma: If there is an LFSR of length L that generates $s_0, s_1, s_2, s_3, \dots, s_{n-1}$ but not $s_0, s_1, s_2, s_3, \dots, s_{n-1}, s_n$, then any LFSR that generates $s_0, s_1, s_2, s_3, \dots, s_{n-1}, s_n$ has length at least $n + 1 - L$, i.e., the linear complexity of $s_0, s_1, s_2, s_3, \dots, s_{n-1}, s_n$ is at least $n + 1 - L$.

Proof: To be given during the lecture.

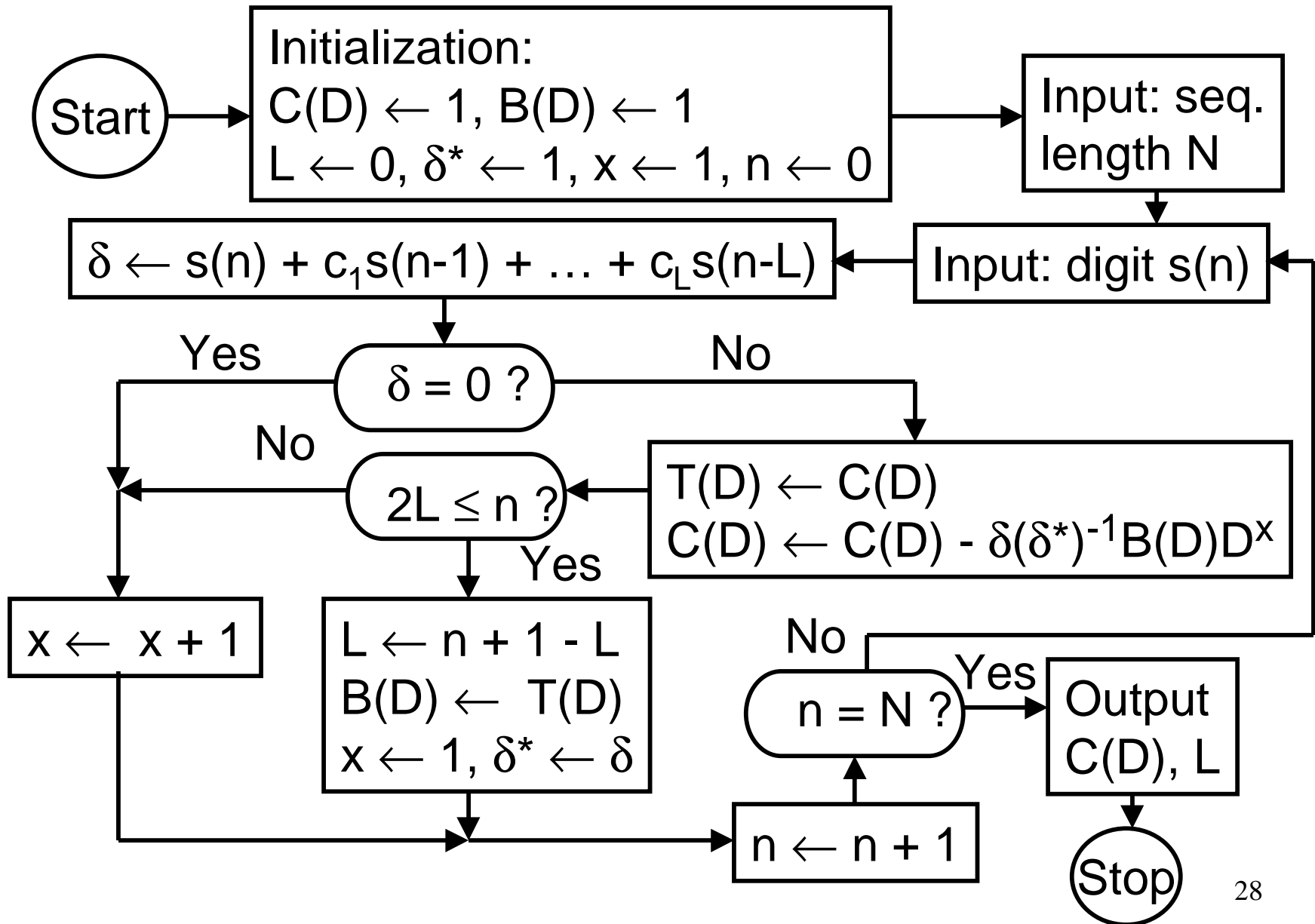
In fact, this is the **exact** linear complexity of $s_0, s_1, s_2, s_3, \dots, s_{n-1}, s_n$ if L is the linear complexity of $s_0, s_1, s_2, s_3, \dots, s_{n-1}$.

J. L. Massey, "Shift-Register Synthesis and BCH Decoding," *IEEE Trans. on Info. Th.*, Vol. IT-15, pp. 122-127, Jan. 1969.

Consequences of the length-change lemma:

- the shortest LFSR that generates the length- n sequence $s_0, s_1, s_2, s_3, \dots, s_{n-1}$ is unique if and only if its length L satisfies $2L \leq n$.
- If the linear complexity L of the length- n sequence $s_0, s_1, s_2, s_3, \dots, s_{n-1}$ satisfies $2L < n$, then the linear complexity of the length- $2L$ sequence $s_0, s_1, s_2, s_3, \dots, s_{2L}$ is also L and the unique shortest LFSR generating the latter sequence coincides with the unique shortest LFSR generating the former.

Flowchart of BM Algorithm [Input $s(0), s(1), \dots, s(N-1)$]



C Program of BM Algorithm for Binary Sequences

```
for (n = 0; n != seqlength; ++n)
{
    d = 0;
    for (k = 0; k != L+1; ++k)
        d = d ^ (cpoly[k] & seq[n-k]);
    if (d != 0)
    {
        for (k = 0; k != L+1; ++k) tpoly[k] = cpoly[k];
        for (k = x; k != oldL+x+1; ++k)
            cpoly[k] = cpoly[k] ^ bpoly[k-x];
        if (2*L > n) x = ++x;
        else
        {
            for (k = 0; k != L+1; ++k)
                bpoly[k] = tpoly[k];
            oldL = L;
            L = n + 1 - L;
            x = 1;
        }
    }
    else x = ++x;
}
```

Recall the expressions for the **DFT** $_{\xi}$ and its inverse:

$$B_i = \sum_{n=0}^{N-1} b_n \xi^{in}$$

and

$$b_n = \frac{1}{N} \sum_{i=0}^{N-1} B_i \xi^{-in}$$

where $\mathbf{b} = [b_0, b_1, \dots, b_{N-1}]$ and $\mathbf{B} = [B_0, B_1, \dots, B_{N-1}]$
and where ξ is a **primitive N^{th} root of unity** in a
field F , i.e., $\xi^N = 1$ but $\xi^i \neq 1$ for $1 \leq i < N$.

If we identify the \mathbf{N} -tuple $\mathbf{b} = [b_0, b_1, \dots, b_{N-1}]$ with the polynomial

$$\mathbf{b}(D) = b_0 + b_1 D + \dots + b_{N-1} D^{N-1},$$

then the DFT definition can be written simply as

$$\boxed{B_i = \mathbf{b}(\xi^i).$$

Similarly, if we identify $\mathbf{B} = [B_0, B_1, \dots, B_{N-1}]$ with the polynomial

$\mathbf{B}(D) = B_0 + B_1 D + \dots + B_{N-1} D^{N-1}$, the inverse DFT can be written as:

$$\boxed{b_n = \frac{1}{\mathbf{N}} \mathbf{B}(\xi^{-n})}$$

where here \mathbf{N} denotes the sum of \mathbf{N} 1's in the field F .

Example: Let ξ be a primitive element of $GF(2^2)$, i.e.,
 $\xi^2 = \xi + 1 \neq 0$ and $\xi^3 = 1$. $\Rightarrow N = 3$.

$$\text{Let } \mathbf{b} = (1, \xi, 1+\xi)$$

$$\begin{aligned} \Rightarrow \mathbf{b}(D) &= 1 + \xi D + (1+\xi)D^2 \\ \Rightarrow \mathbf{b}(\xi^0) &= \mathbf{b}(1) = 1 + \xi + 1 + \xi = 0 \\ \mathbf{b}(\xi) &= 1 + \xi^2 + \xi^2 + \xi^3 = 0 \\ \mathbf{b}(\xi^2) &= 1 + \xi^3 + \xi^4 + \xi^5 = \xi + \xi^2 = 1 \end{aligned}$$

$$\Rightarrow \mathbf{B} = (0, 0, 1)$$

$$\begin{aligned} \Rightarrow \mathbf{B}(D) &= D^2 \\ \mathbf{B}(1) &= 1 \\ \mathbf{B}(\xi^{-1}) &= \mathbf{B}(\xi^2) = \xi^4 = \xi \\ \mathbf{B}(\xi^{-2}) &= \mathbf{B}(\xi) = \xi^2 = \xi + 1 \\ \mathbf{N} &= 1 + 1 + 1 = 1 \Rightarrow \mathbf{N}^{-1} = 1 \\ \Rightarrow \mathbf{N}^{-1}(\mathbf{B}(1), \mathbf{B}(\xi^{-1}), \mathbf{B}(\xi^{-2})) &= (1, \xi, 1+\xi), \text{ which checks!} \end{aligned}$$

Suppose that \mathbf{s} is N -periodic (i.e., $s_i = s_{N+i}$, all $i \geq 0$)

$$\Rightarrow \mathbf{S}(D) = s_0 + s_1D + s_2D^2 + \dots \\ = s^N(D) (1 + D^N + D^{2N} + \dots)$$

where $s^N(D) = s_0 + s_1D + s_2D^2 + \dots + s_{N-1}D^{N-1}$.

Hence $\mathbf{S}(D)(1 - D^N) = s^N(D)$.

Multiplying by the connection polynomial $\mathbf{C}(D)$ of any LFSR generating \mathbf{s} gives

$$\mathbf{P}(D)(1 - D^N) = s^N(D)\mathbf{C}(D),$$

which ensures that $\deg[\mathbf{P}(D)] < \deg[\mathbf{C}(D)]$.

Lemma: $\mathbf{C}(D) = 1 + c_1D + \dots + c_L D^L$ is the connection polynomial of an LFSR that generates the N -periodic sequence \mathbf{s} if and only if $s^N(D)\mathbf{C}(D) = \mathbf{P}(D)(1 - D^N)$. Moreover, this is the shortest LFSR that generates \mathbf{s} if and only if $\gcd[\mathbf{P}(D), \mathbf{C}(D)] = 1$.

Suppose that ξ is a primitive N^{th} root of unity in F .
 \Rightarrow The zeroes of $1 - D^N$ are precisely the elements of the set $\{1, \xi^{-1}, \xi^{-2}, \dots, \xi^{-(N-1)}\}$.

Suppose further that $\gcd(P(D), C(D)) = 1$ and that $s^N(D)C(D) = P(D)(1 - D^N)$.

$\Rightarrow \xi^{-n}$ is a zero of either $C(D)$ or $s^N(D)$,
but not both, for $0 \leq n < N$.

$\Rightarrow C(D)$ has no other zeroes since $\gcd(P(D), C(D)) = 1$.

$\Rightarrow \deg\{C(D)\} = \text{number of } n \text{ with } s^N(\xi^{-n}) \neq 0 \text{ for } 0 \leq n < N$

$\Rightarrow L(s) = \deg\{C(D)\} = \text{Hamming weight of the}$
inverse DFT of $[s_0, s_1, \dots, s_{N-1}]$.

This is Serconek's and my proof of **Blahut's Theorem!**

J. L. Massey and S. Serconek, "Linear Complexity of Periodic Sequences: A General Theory," in *Advances in Cryptology-CRYPTO'96* (Ed. N. Koblitz), Lecture Notes in Computer Science No. 1109. New York: Springer, 1996, pp. 358-371.

Blahut's Theorem:

For the DFT in any field F , if $\mathbf{B} = \text{DFT}_\xi(\mathbf{b})$, then the **linear complexity** of the periodically repeated frequency-domain sequence $\mathbf{B}, \mathbf{B}, \mathbf{B}, \dots$ equals $w_H(\mathbf{b})$, the **Hamming weight** of the time-domain sequence \mathbf{b} . Similarly, the linear complexity of the periodically repeated time-domain sequence $\mathbf{b}, \mathbf{b}, \mathbf{b}, \dots$ equals $w_H(\mathbf{B})$, the Hamming weight of the frequency-domain sequence \mathbf{B} .

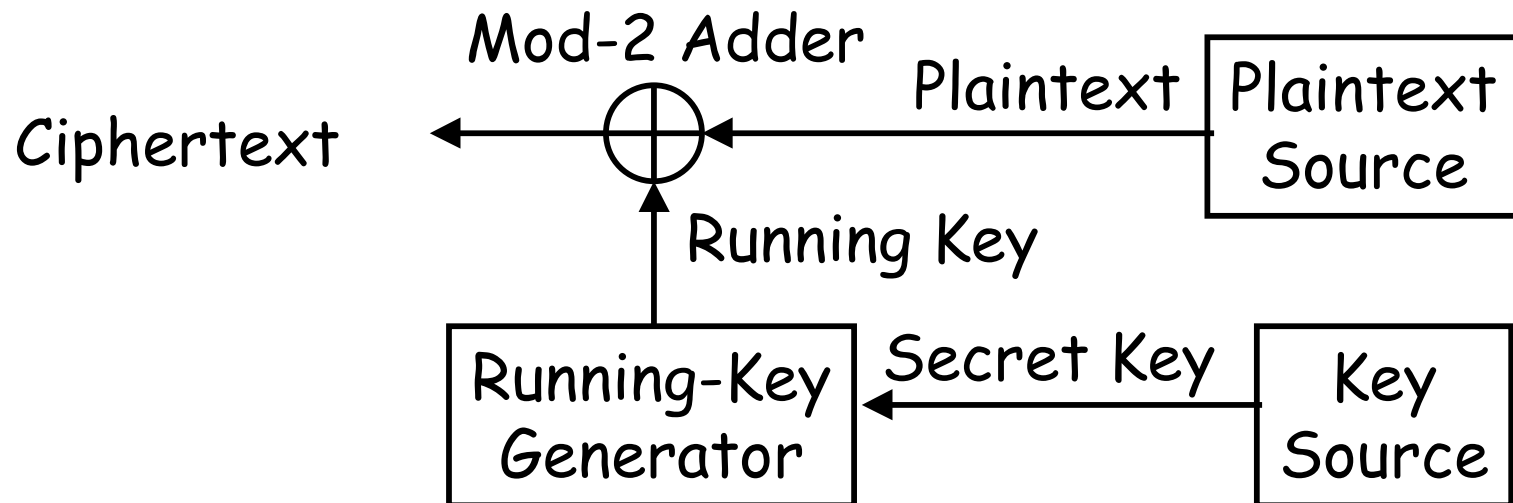
Example:

The 3-periodic $GF(2^2)$ -ary sequence $1, \xi, 1+\xi, 1, \xi, 1+\xi, \dots$ has linear complexity 1.

The 3-periodic binary sequence $0, 0, 1, 0, 0, 1, 0, 0, 1, \dots$ has linear complexity 3.

Reason: The DFT of $\mathbf{b} = (1, \xi, 1+\xi)$ is $\mathbf{B} = (0, 0, 1)$.

Attack on an Additive Stream Cipher



In a **known-plaintext** (or chosen-plaintext) **attack**, the goal is, when given some segment of the running key, to **predict the future of the running key reliably**. For this reason the running key must always have a large linear complexity if the stream cipher is to be secure.