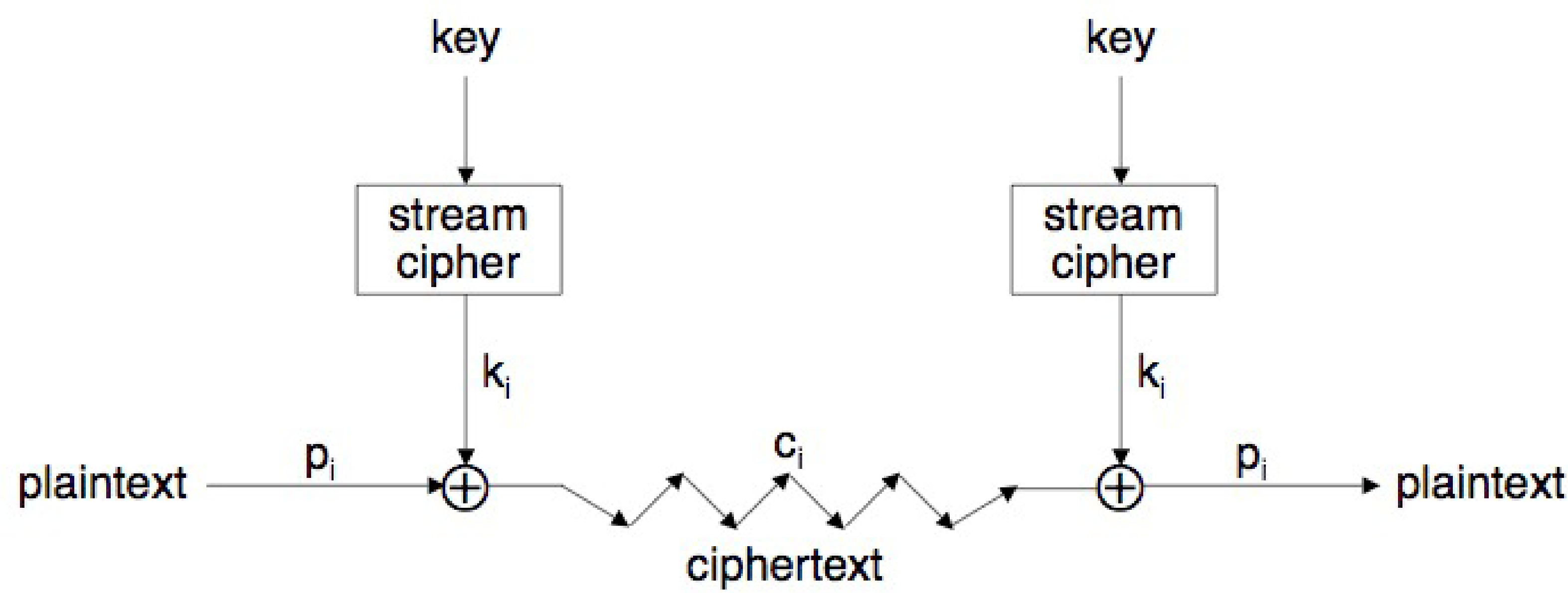
Stream Ciphers

Stream Ciphers

- Generalization of **one-time pad**
- Trade provable security for practicality
- Stream cipher is initialized with short **key**
- Key is “stretched” into long **keystream**
- Keystream is used like a one-time pad
  - XOR to encrypt or decrypt
- Stream cipher is a keystream generator
- Usually, keystream is bits, sometimes bytes

Stream Cipher



- Generic view of stream cipher

Stream Cipher

- We consider 3 real stream ciphers
  - **ORYX** — weak cipher, uses shift registers, generates 1 byte/step
  - **RC4** — strong cipher, widely used but used poorly in WEP, generates 1 byte/step
  - **PKZIP** — intermediate strength, unusual mathematical design, generates 1 byte/step
- But first, we discuss shift registers

Shift Registers

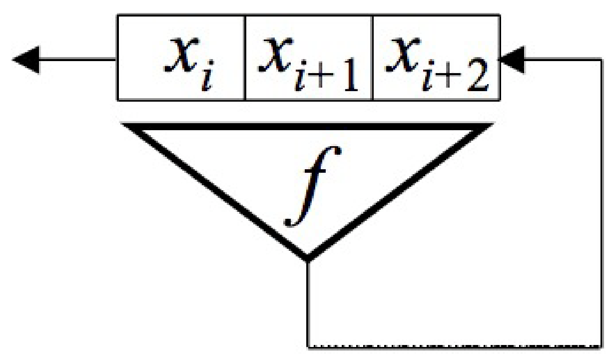
- Traditionally, stream ciphers were based on shift registers
  - Today, a wider variety of designs
- Shift register includes
  - A series of stages each holding one bit
  - A feedback function
- A linear feedback shift register (**LFSR**) has a linear feedback function

Shift Register

- Example (nonlinear) feedback function

  $f(x_i, x_{i+1}, x_{i+2}) = 1 \oplus x_i \oplus x_{i+2} \oplus x_{i+1}x_{i+2}$

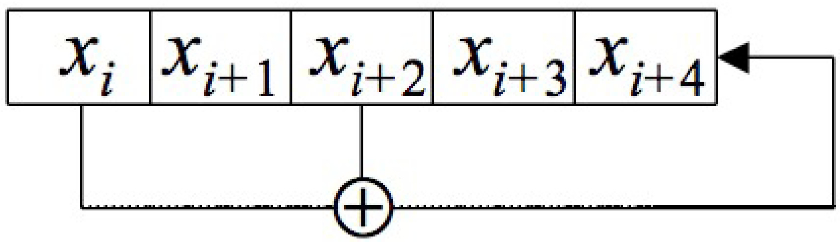
- Example (nonlinear) shift register



- First 3 bits are **initial fill**: (x_0, x_1, x_2)

LFSR

- Example of LFSR

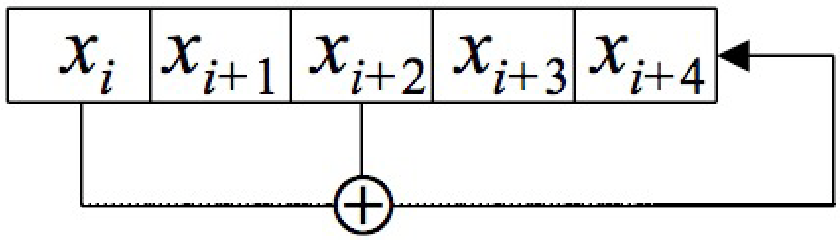


- Then $x_{i+5} = x_i \oplus x_{i+2}$ for all i
- If initial fill is $(x_0,x_1,x_2,x_3,x_4) = 01110$

  then $(x_0,x_1,\ldots,x_{15},\ldots) = 0111010100001001\ldots$

LFSR

- For LFSR



- We have $x_{i+5} = x_i \oplus x_{i+2}$ for all i
- Linear feedback functions often written in polynomial form: $x^5 + x^2 + 1$
- **Connection polynomial** of the LFSR

Berlekamp-Massey Algorithm

- Given (part of) a (periodic) sequence, can find shortest LFSR that could generate the sequence
- Berlekamp-Massey algorithm
  - Order N^2, where N is length of LFSR
  - Iterative algorithm
  - Only 2N consecutive bits required

Berlekamp-Massey Algorithm

- Binary sequence: $s = (s_0, s_1, s_2, \ldots, s_{n-1})$
- **Linear complexity** of s is the length of shortest LFSR that can generate s
- Let L be linear complexity of s
- Then connection polynomial of s is of form
  $C(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_L x^L$
- Berlekamp-Massey finds L and C(x)
  - Algorithm on next slide (where d is known as the **discrepancy**)

Berlekamp-Massey Algorithm

```
// Given binary sequence s = (s_0, s_1, s_2, ..., s_{n-1})
// Find linear complexity L and connection polynomial C(x)
BM(s)
    C(x) = B(x) = 1
    L = N = 0
    m = -1
    while N < n // n is length of input sequence
        d = s_N ⊕ c_1 s_{N-1} ⊕ c_2 s_{N-2} ⊕ ··· ⊕ c_L s_{N-L}
        if d == 1 then
            T(x) = C(x)
            C(x) = C(x) + B(x) x^{N-m}
            if L ≤ N/2 then
                L = N + 1 - L
                m = N
                B(x) = T(x)
            end if
        end if
        N = N + 1
    end while
    return(L)
end BM
```

Berlekamp-Massey Algorithm

- Example:

sequence: $s = (s_0, s_1, \ldots, s_7) = 10011100$
initialize: $C(x) = B(x) = 1,\ L = N = 0,\ m = -1$

$\underline{N = 0}$
$d = s_0 = 1$
$T(x) = 1,\ C(x) = 1 + x$
$L = 1,\ m = 0,\ B(x) = 1$

$\underline{N = 1}$
$d = s_1 \oplus c_1 s_0 = 1$
$T(x) = 1 + x,\ C(x) = 1$

$\underline{N = 2}$
$d = s_2 \oplus c_1 s_1 \oplus c_2 s_0 = 0$

$\underline{N = 3}$
$d = s_3 \oplus c_1 s_2 \oplus c_2 s_1 \oplus c_3 s_0 = 1$
$T(x) = 1,\ C(x) = 1 + x^3$
$L = 3,\ m = 3,\ B(x) = 1$

$\underline{N = 4}$
$\vdots$

Berlekamp-Massey Algorithm

- Berlekamp-Massey is efficient way to determine minimal LFSR for sequence
- With known plaintext, keystream bits of stream cipher are exposed
- With enough keystream bits, can use Berlekamp-Massey to find entire keystream
  - 2L bits is enough, where L is linear complexity of the keystream
- **Keystream must have large linear complexity**

Cryptographically Strong Sequences

- A sequence is **cryptographically strong** if it is a "good" keystream
  - "Good" relative to some specified criteria
- Crypto strong sequence must be **unpredictable**
  - Known plaintext exposes part of keystream
  - Trudy must not be able to determine more of the keystream from a short segment
- Small linear complexity implies predictable
  - Due to Berlekamp-Massey algorithm

Crypto Strong Sequences

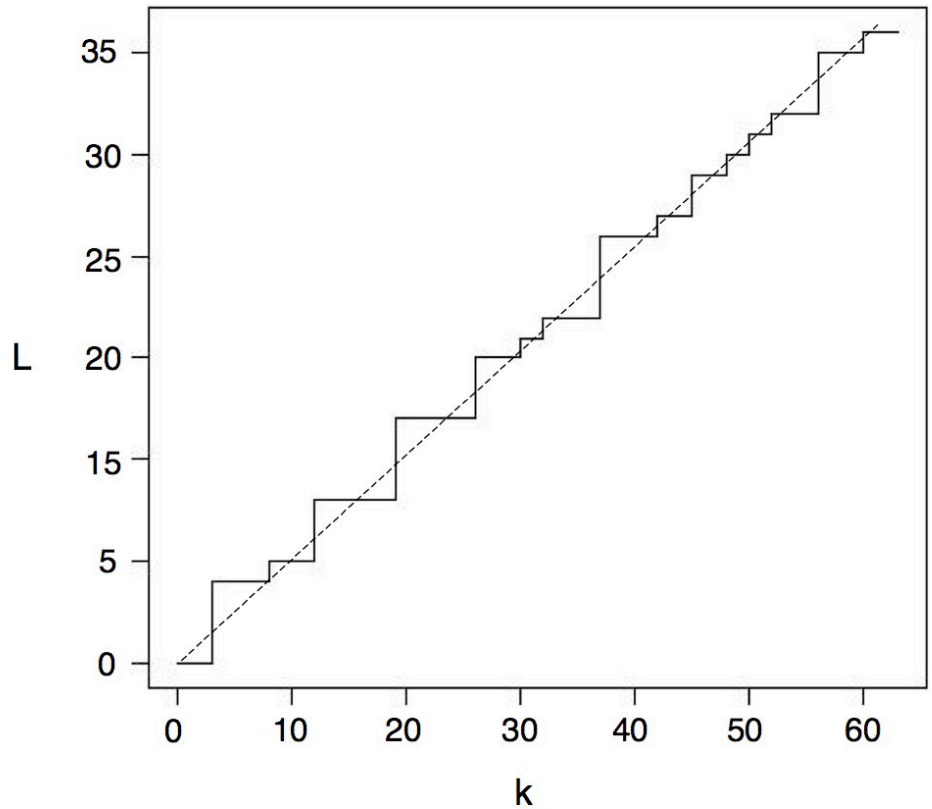
- Necessary for a **cryptographically strong** keystream to have a high linear complexity
- But not sufficient!
- Why? Consider $s = (s_0, s_1, \ldots, s_{n-1}) = 00\ldots01$
- Then s has linear complexity n
  - Smallest shift register for s requires n stages
  - Largest possible for sequence of period n
  - But s is not cryptographically strong
- Linear complexity "concentrated" in last bit

Linear Complexity Profile

- **Linear complexity profile** is a better measure of cryptographic strength
- Plot linear complexity as function of bits processed in Berlekamp-Massey algorithm
  - Should follow n/2 line "closely but irregularly"
- Plot of sequence $s = (s_0,s_1,\ldots,s_{n-1}) = 00\ldots01$ would be 0 until last bit, then jumps to n
  - Does **not** follow n/2 line "closely but irregularly"
  - Not a strong sequence (by this definition)

Linear Complexity Profile

- A "good" linear complexity profile

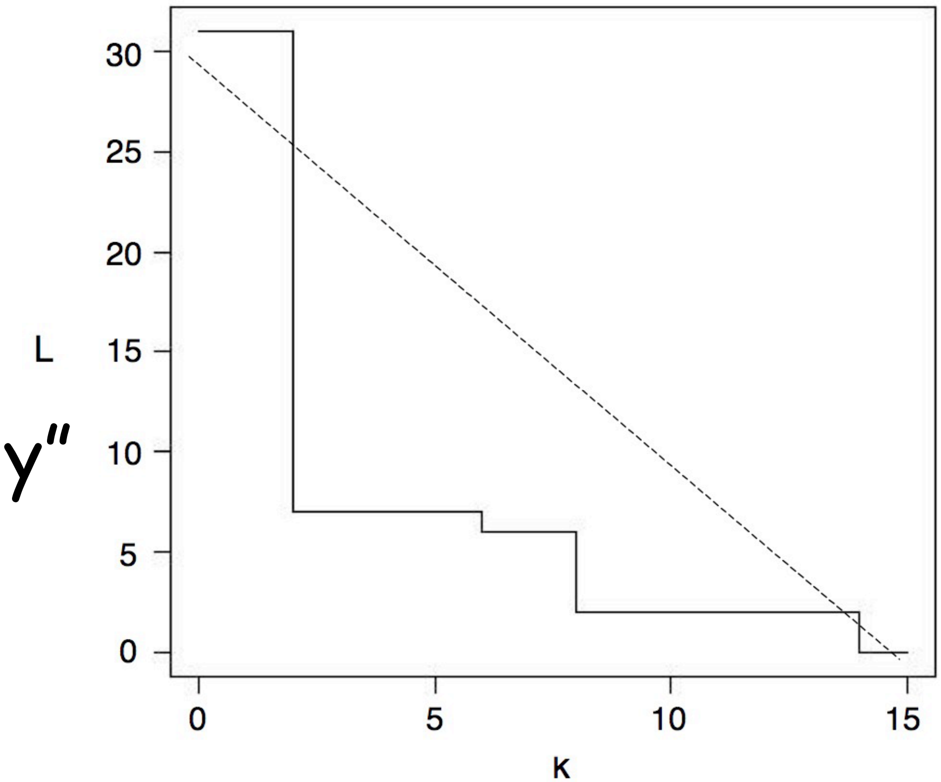


k-error Linear Complexity Profile

- Alternative way to measure cryptographically strong sequences
- Consider again $s = (s_0,s_1,\ldots,s_{n-1}) = 00\ldots01$
- This s has max linear complexity, but it is only 1 bit away from having min linear complexity
- k-error linear complexity is min complexity of any sequence that is "distance" k from s
- 1-error linear complexity of $s = 00\ldots01$ is 0
  - Linear complexity of this sequence is "unstable"

k-error Linear Complexity Profile

- k-error linear complexity profile
  - k-error linear complexity as function of k
- Example:
  - Not a strong s
  - Good profile should follow diagonal "closely"



Crypto Strong Sequences

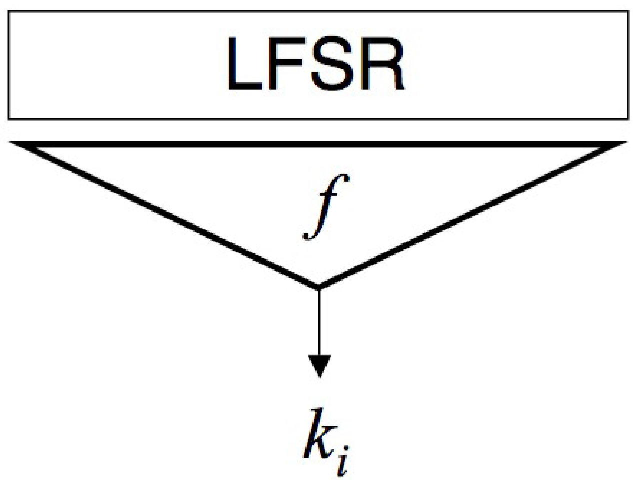
- Linear complexity must be "large"
- Linear complexity profile must n/2 line "closely but irregularly"
- k-error linear complexity profile must follow diagonal line "closely"
- All of this is necessary but not sufficient for crypto strength!

Shift Register-Based Stream Ciphers

- Two approaches to LFSR-based stream ciphers
  - One LFSR with nonlinear combining function
  - Multiple LFSRs combined via nonlinear func
- In either case
  - Key is initial fill of LFSRs
  - Keystream is output of nonlinear combining function

Shift Register-Based Stream Ciphers

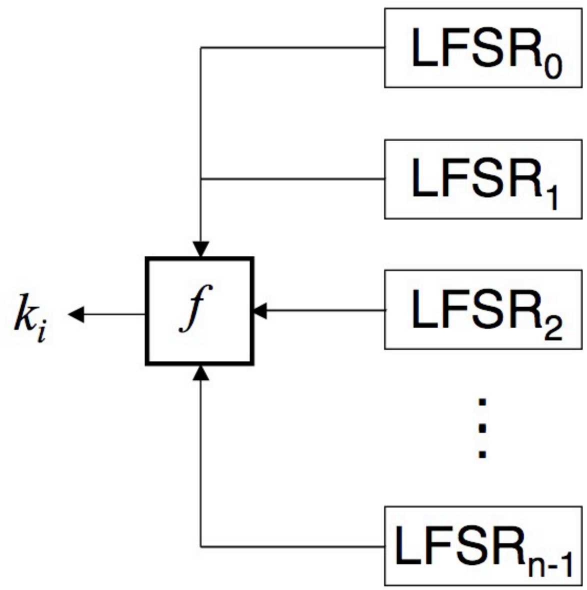
- LFSR-based stream cipher
  - 1 LFSR with nonlinear function $f(x_0,x_1,\ldots,x_{n-1})$



- Keystream: $k_0,k_1,k_2,\ldots$

Shift Register-Based Stream Ciphers

- LFSR-based stream cipher
  - Multiple LFSRs with nonlinear function



- Keystream: $k_0, k_1, k_2, \ldots$

Shift Register-Based Stream Ciphers

- Single LFSR example is special case of multiple LFSR example
- To convert single LFSR case to multiple
  - Let $LFSR_0, \ldots LFSR_{n-1}$ be same as LFSR
  - Initial fill of $LFSR_0$ is initial fill of LFSR
  - Initial fill of $LFSR_1$ is initial fill of LFSR stepped once
  - And so on…

Correlation Attack

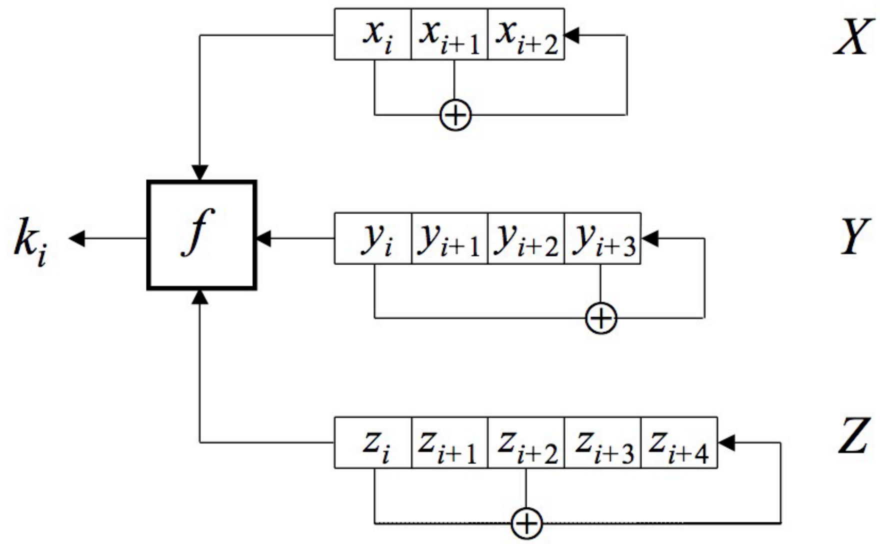
- Trudy obtains some segment of keystream from LFSR stream cipher
  - Of the type considered on previous slides
- Can assume stream cipher is the multiple shift register case
  - If not, convert it to this case
- By Kerckhoffs Principle, we assume shift registers and combining function known
- Only unknown is the key
  - The key consists of LFSR initial fills

Correlation Attack

- Trudy wants to recover LFSR initial fills
  - She knows all connection polynomials and nonlinear combining function
  - She also knows N keystream bits, $k_0, k_1, \ldots, k_{N-1}$
- Sometimes possible to determine initial fills of the LFSRs independently
  - By correlating each LFSR output to keystream
  - A classic **divide and conquer** attack

Correlation Attack

- For example, suppose keystream generator is of the form:



- And $f(x,y,z) = xy \oplus yz \oplus z$
- Note that key is 12 bits, initial fills

Correlation Attack

- For stream cipher on previous slide
- Suppose initial fills are
  - X = 011, Y = 0101, Z = 11100

bits i = 0,1,2,…23

x_i	0	1	1	1	0	0	1	0	1	1	1	0	0	1	0	1	1	1	0	0	1	0	1	1
y_i	0	1	0	1	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0
z_i	1	1	1	0	0	0	1	1	0	1	1	1	0	1	0	1	0	0	0	0	1	0	0	1
k_i	1	1	1	1	0	0	1	0	0	1	1	0	0	1	0	1	1	0	0	0	1	0	1	1

Correlation Attack

- Consider truth table for combining function: $f(x,y,z) = xy \oplus yz \oplus z$
- Easy to show that

  $f(x,y,z) = x$ with probability 3/4

  $f(x,y,z) = z$ with probability 3/4
- Trudy can use this to recover initial fills from known keystream

Correlation Attack

- Trudy sees keystream in table
- Trudy wants to find initial fills
- She guesses X = 111, generates first 24 bits of putative X, compares to k_i

x_i	1	1	1	0	0	1	0	1	1	1	0	0	1	0	1	1	1	0	0	1	0	1	1	1
k_i	1	1	1	1	0	0	1	0	0	1	1	0	0	1	0	1	1	0	0	0	1	0	1	1

- Trudy finds 12 out of 24 matches
- As expected in random case

Correlation Attack

- Now suppose Trudy guesses correct fill, X = 011
- First 24 bits of X (and keystream)

x_i	0	1	1	1	0	0	1	0	1	1	1	0	0	1	0	1	1	1	0	0	1	0	1	1

k_i	1	1	1	1	0	0	1	0	0	1	1	0	0	1	0	1	1	0	0	0	1	0	1	1

- Trudy finds 21 out of 24 matches
- Expect 3/4 matches in causal case
- Trudy has found initial fill of X

Correlation Attack

- How much work is this attack?
  - The X,Y,Z fills are 3,4,5 bits, respectively
- We need to try about half of the initial fills before we find X
- Then we try about half of the fills for Y
- Then about half of Z fills
- Work is $2^2 + 2^3 + 2^4 < 2^5$
- Exhaustive key search work is 2^{11}

Correlation Attack

- Work factor in general…
- Suppose n LFSRs
  - Of lengths $N_0, N_1, \ldots, N_{n-1}$
- Correlation attack work is

$$2^{N_0-1} + 2^{N_1-1} + \cdots + 2^{N_{n-1}-1}$$

- Work for exhaustive key search is

$$2^{N_0+N_1+\cdots+N_{n-1}-1}$$

Conclusions

- Keystreams must be **cryptographically strong**
  - Crucial property: unpredictable
- Lots of theory available for LFSRs
  - Berlekamp-Massey algorithm
  - Nice mathematical theory exists
- LFSRs can be used to make stream ciphers
  - LFSR-based stream ciphers must be **correlation immune**
  - Depends on properties of function f

Coming Attractions

- Consider attacks on 3 stream ciphers
  - **ORYX** — weak cipher, uses shift registers, generates 1 byte/step
  - **RC4** — strong, widely used but used poorly in WEP, generates 1 byte/step
  - **PKZIP** — medium strength, unusual design, generates 1 byte/step