Introduction to Stream Ciphers

Jovan Golić

Security Innovation, Telecom Italia

Winter School on Information Security, Finse 2008, Norway

STREAM CIPHERS

- (1) Definition
- (2) Basic types
- (3) Modes of operation
- (4) Practical security of keystream generators
- (5) Keystream generators based on LFSR’s
- (6) Other types of keystream generators

(1) Definition

$x = (x_t)_{t=0}^{\infty}$ – binary plaintext sequence

$y = (y_t)_{t=0}^{\infty}$ – binary ciphertext sequence

$S = (S_t)_{t=0}^{\infty}$ – M-bit internal state sequence

K – k-bit secret (session) key

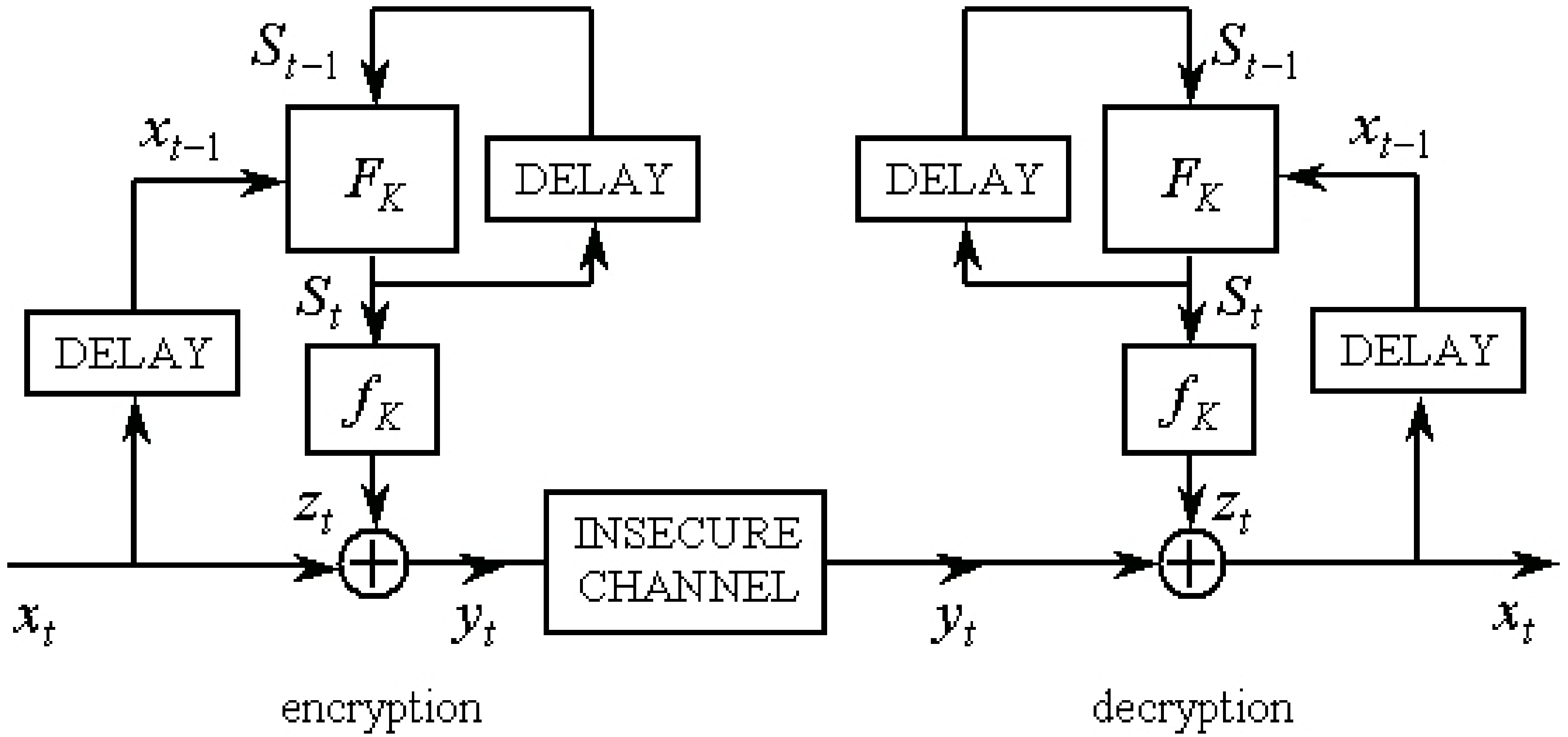
R – r-bit randomizing (message) key

$S_0(K, R)$ – initial state

$F_K : \{0,1\}^{M+1} \to \{0,1\}^M$ – next-state function

$f_K : \{0,1\}^M \to \{0,1\}$ – output function

- General binary stream cipher decipherable without delay is an invertible nonautonomous finite-state machine with one input and one output:
  - encryption (sequential) transform: $x \mapsto y$
    $$S_{t+1} = F_K(S_t, x_t), \qquad t \geq 0$$
    $$y_t = x_t \oplus f_K(S_t), \qquad t \geq 0$$
  - decryption transform: $y \mapsto x$
    $$S_{t+1} = F_K(S_t, y_t \oplus f_K(S_t)), \qquad t \geq 0$$
    $$x_t = y_t \oplus f_K(S_t), \qquad t \geq 0$$
  - encryption/decryption transforms are defined by keystream sequence $z = (z_t)_{t=0}^{\infty} = (f_K(S_t))_{t=0}^{\infty}$ which depends on K, R, and possibly on x
    $$y_t = x_t \oplus z_t$$
    $$x_t = y_t \oplus z_t$$


S_{t-1}
x_{t-1}
F_K
DELAY
DELAY
S_t
f_K
z_t
x_t
y_t
INSECURE CHANNEL
y_t
DELAY
F_K
S_{t-1}
x_{t-1}
S_t
f_K
DELAY
z_t
x_t
encryption
decryption

- Basic criteria
  - implementation: keystream sequence easy to compute if K, R, and x are known
  - practical security = keystream unpredictability: keystream sequence infeasible to compute from known keystream portions if K is unknown (known- plaintext scenario)
  - key size k large enough to prevent exhaustive search, e.g., ≥ 100

- no theoretical secrecy: unicity distance

  $$n_u \approx \frac{k}{1-H}$$

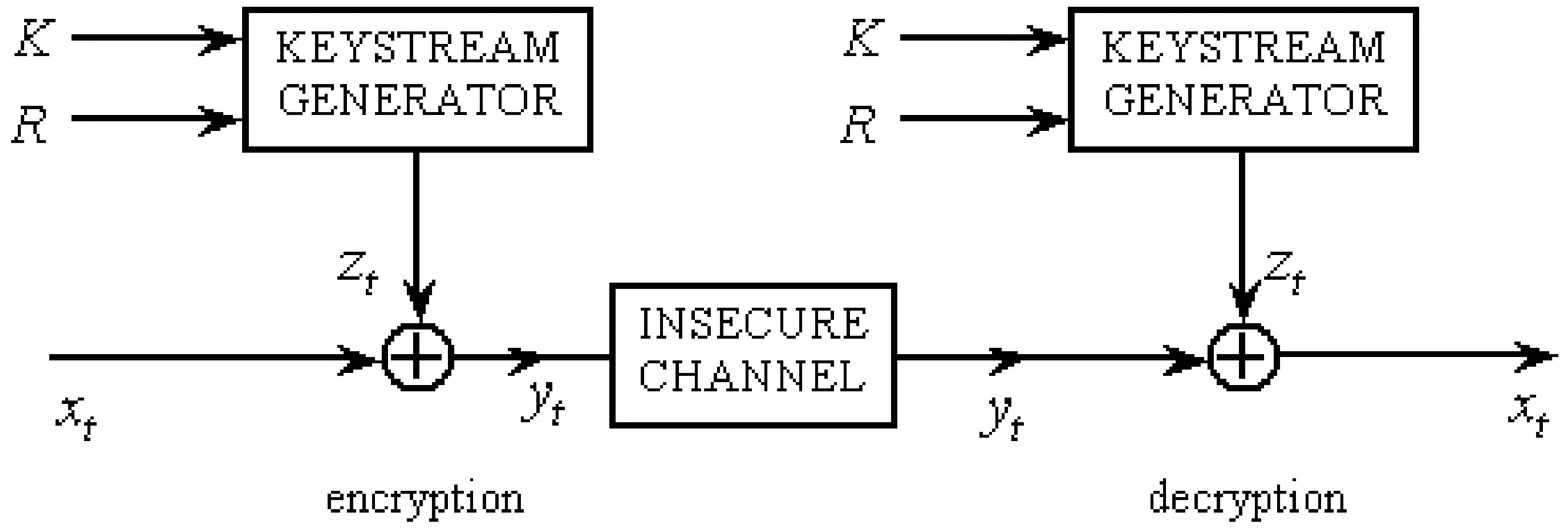
  where H is entropy of x per bit $\left(H \le 1\right)$

- randomizing key R enables resynchronization, that is, repeated use of the same secret key:
  - may be known (transmitted in the clear)
  - r large enough to ensure that R is not repeated with high probability (e.g., $r \ge 32$); more precisely, $2^{r/2} \ge \#m$ where $\#m$ is the number of messages (birthday paradox)

(2) Basic types

- Keystream generator: keystream sequence z independent of plaintext sequence

$$S_{t+1} = F_K(S_t), \qquad t \geq 0$$
$$y_t = x_t \oplus f_K(S_t) = x_t \oplus z_t, \qquad t \geq 0$$



– insensitive to substitution errors, as current ciphertext bit depends on current plaintext bit only and vice versa (not good for authentication purposes)

– sensitive to synchronization errors

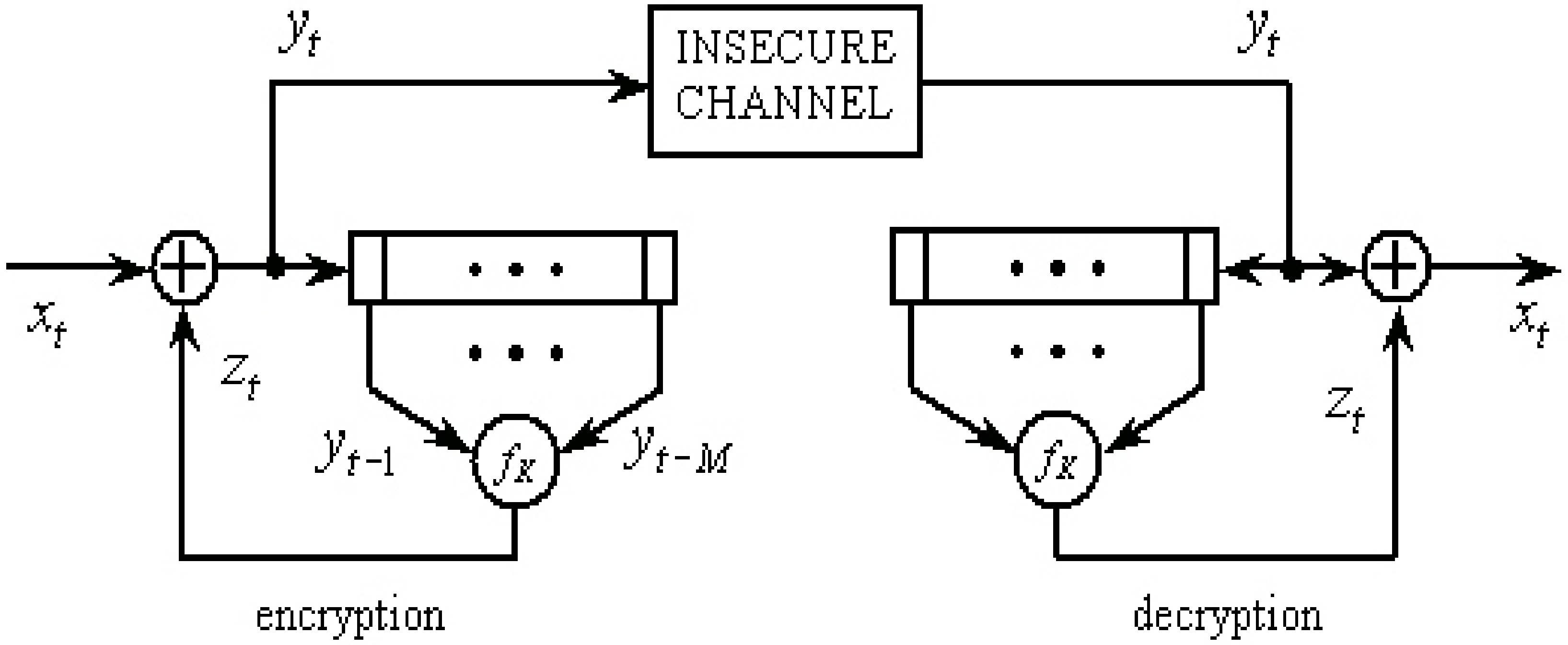
  – change K frequently, without using R

  – use R, transmitted in the clear before encryption; use reinitialization algorithm to generate $S_0(K,R)$ with property $R' \neq R'' \;\Rightarrow\; S_0(K,R') \neq S_0(K,R'')$

– <u>One-time-pad assumption</u>: for any K, the probability that S_t is repeated is very low (long segments of plaintext should not be encrypted by the same keystream!)

- Self-synchronizing stream cipher: decryption transform has finite input memory

$$S_{t+1} = (y_t, y_{t-1}, \ldots, y_{t-M+1})$$
$$y_t = x_t \oplus f_K(S_t) = x_t \oplus f_K(y_{t-1}, \ldots, y_{t-M})$$



- insensitive to substitution and synchronization errors, that is , propagation of such errors in ciphertext is limited to only $M+1$ plaintext bits (self-synchronizing property)
- practical security: feedback function f_K infeasible to compute if K is unknown (number of inputs to f_K must be large)

- Stream cipher with plaintext memory (SCPM): $F_K(S_t, x_t)$ effectively depends on x_t in such a way that both encryption and decryption transforms have infinite input memory
  - sensitive to both substitution and synchronization errors
  - potential for authentication purposes, but each plaintext bit affects only current and subsequent ciphertext bits **(trivial attack)**

- if R is not used and K is repeated, the same plaintext sequences are encrypted into the same ciphertext sequence
- R can be prepended to message and then encrypted
- rarely treated in open literature (PKZIP)
- can be used to construct block ciphers and (keyed) hash functions *Golić* 2000

(3) Modes of operation

- **Conversion:** Keystream Generator → SCPM
  - Introduce plaintext bit into next-state function of keystream generator so as to achieve propagation effect, i.e., its change should result in a random-looking change of subsequent keystream sequence
  - Easy to achieve, since in keystream generators, change of any initial state bit should result in a random-looking change of keystream sequence
- **Conversions:** SCPM → (keyed) hash function, SCPM → self-synchronizing stream cipher, SCPM → block cipher, *Golić* 2000

- **Conversion:** SCPM $\rightarrow$ keyed hash function
  - Use SCPM with secret key K
  - Plaintext sequence for SCPM: m-bit input to h_K, lM-bit all-zero input ($M \geq n$ is internal memory size of SCPM)
  - Output of h_K: last n ciphertext bits of SCPM, after $m+lM$ clocks (e.g., l=3)
  - <u>*Security:*</u> at least as secure as SCPM
  - Variable message length, no padding
  - Also: encryption + hashing, with the same key

- **Conversion:** SCPM → block cipher
  - Product or cascade connection of three SCPM's
  - *Plaintext sequence for 1. SCPM: n-bit input to E_K*
  - *Plaintext sequence for 2. SCPM: reversed n-bit output of 1. SCM*
  - *Plaintext sequence for 3. SCPM: reversed n-bit output of 2. SCM*
  - Output of E_K: *n*-bit output of 3. SCPM
  - <u>*Security:*</u> cascade at least as secure as SCPM, product at least as secure as SCPM w.r.t. secret key reconstruction related key attacks
  - Variable block size

(4) <u>Practical security of keystream generators</u>

- Basic criterion: keystream unpredictability
- Equivalent to keystream randomness: computationally infeasible to distinguish keystream sequence from purely random sequence
- Derived criteria:

1°- long period of keystream sequence for (almost) all initial states

2°- negligible fraction of equivalent initial states (producing the same keystream sequence)

3° - good ‘long term’ randomness:

– relative frequencies of patterns over a period are very close to corresponding expected values on a purely random sequence; e.g., Golomb’s postulates involving distribution of single bits, distribution of runs (gaps and blocks), and autocorrelation function); also, distribution of block patterns of bits

4° - good ‘short term’ randomness: keystream sequences pass known statistical tests on lengths of practical interest; used statistics typically follow χ^2 or normal distribution when computed on a χ^2 purely random sequence

- – frequency test: single bits
- – serial test: 2-bit overlapping block patterns
- – poker test: m-bit non-overlapping block patterns
- – runs test: gaps and blocks of varying length
- – autocorrelation test: frequency test on bitwise sums of keystream sequence and its phase shifts

- repetition test: number of repeated non-overlapping m-bit block patterns *Gustafson, Dawson, Golić* 1995; follows Poisson distribution; allows twice as long block patterns to be tested
- *Maurer's* universal test: essentially estimates entropy of keystream sequence through a universal data compression code
- instead on block patterns, statistical tests can be applied on functions of block patterns, but the corresponding distributions are no longer uniform

5º - high complexity measures w.r.t. elementary keystream generators adapted to keystream generator under consideration, that is, the minimum size of such an elementary keystream generator that can produce the same keystream sequences, for example:

- linear complexity, w.r.t. linear feedback shift registers (or nonlinear complexity)
- 2-adic complexity, w.r.t. feedback with carry shift registers *Klapper, Goresky* 1997

6º - <u>Immunity to initial state reconstruction cryptanalytic attacks or more generally, to secret key reconstruction attacks</u>

– ideally, attacks more effective than exhaustive search of secret keys should not exist, e.g., divide-and-conquer secret key reconstruction attacks

– realistically, secret key reconstruction attacks should not be computationally feasible

Examples of attacks:

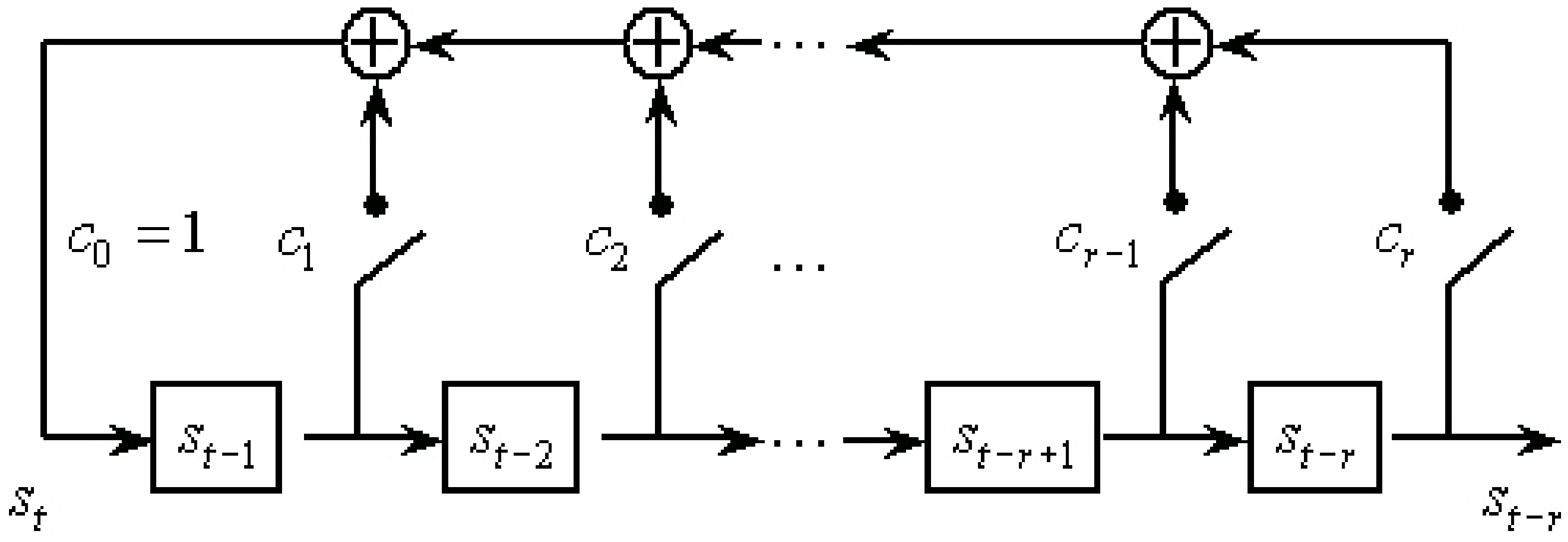
- Correlation attacks
- Fast correlation attacks
- Conditional correlation attacks
- Vectorial fast correlation attacks
- Linear cryptanalysis (LSCA method)
- Linear statistical distinguishers (linear models)
- Time-memory-data tradeoff attacks
- Linear consistency attacks
- Inversion attacks
- Branching attacks
- Guess-and-determine attacks
- Embedding and probabilistic correlation attacks on clock-controlled LFSR's
- Edit-distance and edit-probability correlation attacks on clock-controlled combiners
- Fast correlation attacks on clock-controlled LFSR's
- Resynchronization attacks
- Algebraic attacks

(5) Keystream generators based on LFSR’s

- Use LFSR’s with long period and good statistical properties, but low linear complexity, and destroy linearity, that is, achieve keystream unpredictability by:
  - nonlinear combining functions, memoryless or with memory
  - irregular clocking

3.2.1 Linear feedback shift registers (LFSR's)

- Binary LFSR



- length: r stages (delay elements)
- initial state: $(s_0, s_1, \ldots, s_{r-1})$
- linear recursion: $s_t = \sum_{i=1}^{r} c_i s_{t-i}, \qquad t \geq r$

  (binary sum, mod 2)
- output sequence: $s = (s_t)_{t=0}^{\infty}$
- feedback (connection) polynomial: $f(D) = 1 + \sum_{i=1}^{r} c_i D^i$
- characteristic polynomial: $h(D) = \sum_{i=0}^{r-1} c_{r-i} D^i + D^r$

- Properties

(1) Period: s periodic iff $c_r = 1$ $(\deg f(D) = r)$
(<u>nonsingular LFSR</u>)

- Per $s \mid \exp f(D) = \min\left\{ i \,\middle|\, i \in N,\ f(D) \middle| 1 + D^i \right\}$
- irreducible $f(D)$ $\Rightarrow$ Per $s = \exp f(D)$ for nonzero initial states
- primitive $f(D)$ $\Rightarrow$ Per $s = 2^r - 1$ for nonzero initial states (maximum-length sequence, m-sequence)
  $\Rightarrow$ $2^r - 1$ maximum-length sequences are phase shifts of each other, i.e., lie on the same cycle of length $2^r - 1$

(2) Good ‘long term’ randomness properties of maximum-length sequences:

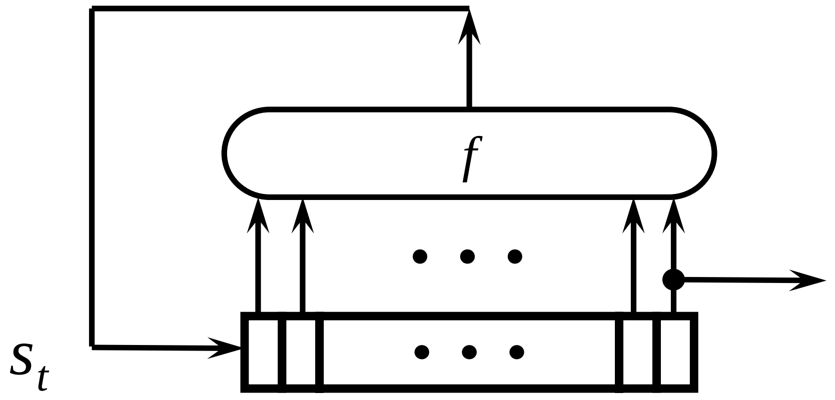
- almost uniform distribution of (overlapping) patterns of length smaller than r
- satisfy *Golomb’s* postulates
  - 2^{r-1} 1’s and $2^{r-1}-1$ 0’s
  - for $1 \leq l \leq r-2$, 2^{r-l-2} blocks and 2^{r-l-2} gaps of length l; one gap of length $r-1$; one block of length r
  - out-of-phase autocorrelation function: $-\frac{1}{2^r-1}$

(3) Empirically good ‘short term’ randomness properties of maximum-length sequences, on reasonably long segments

(4) ‘Bad’ property: predictability, that is, LFSR sequence is easily reconstructed from r consecutive bits if feedback is known, and from $2r$ consecutive bits if feedback is unknown

(5) Easy for hardware implementation, not so easy for software implementation (especially if number of nonzero terms (weight) of $f(D)$ is not low)

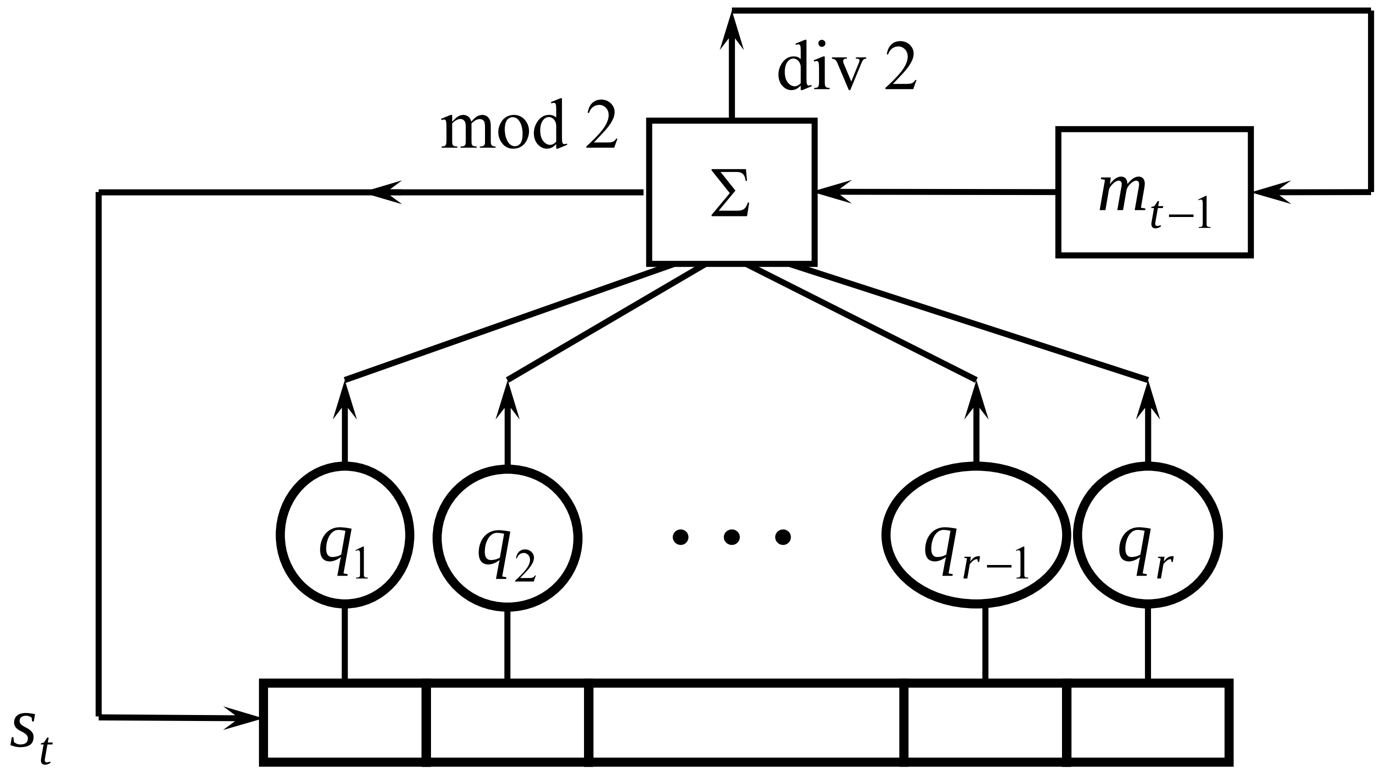
- LFSR over $\mathrm{GF}(2^n)$
- Linear congruential generator: LFSR over Z_{2^n} (addition and multiplication mod 2^n)
- Nonlinear FSR



nonlinear recursion: $s_t = f(s_{t-1},\dots,s_{t-r}), \quad t \ge r$

nonsingular iff $f(s_{t-1},\dots,s_{t-r}) = s_{t-r} \oplus g(s_{t-1},\dots,s_{t-r+1})$

- Feedback with carry shift register (FCSR) *Klapper, Goresky* 1997: specific nonlinear FSR whose feedback function has memory



$$\sigma_t = \sum_{i=1}^{r} q_i s_{t-i} + m_{t-1} \qquad \text{(integer sum)}$$

$$s_t = \sigma_t \bmod 2$$

$$m_t = \lfloor \sigma_t / 2 \rfloor$$

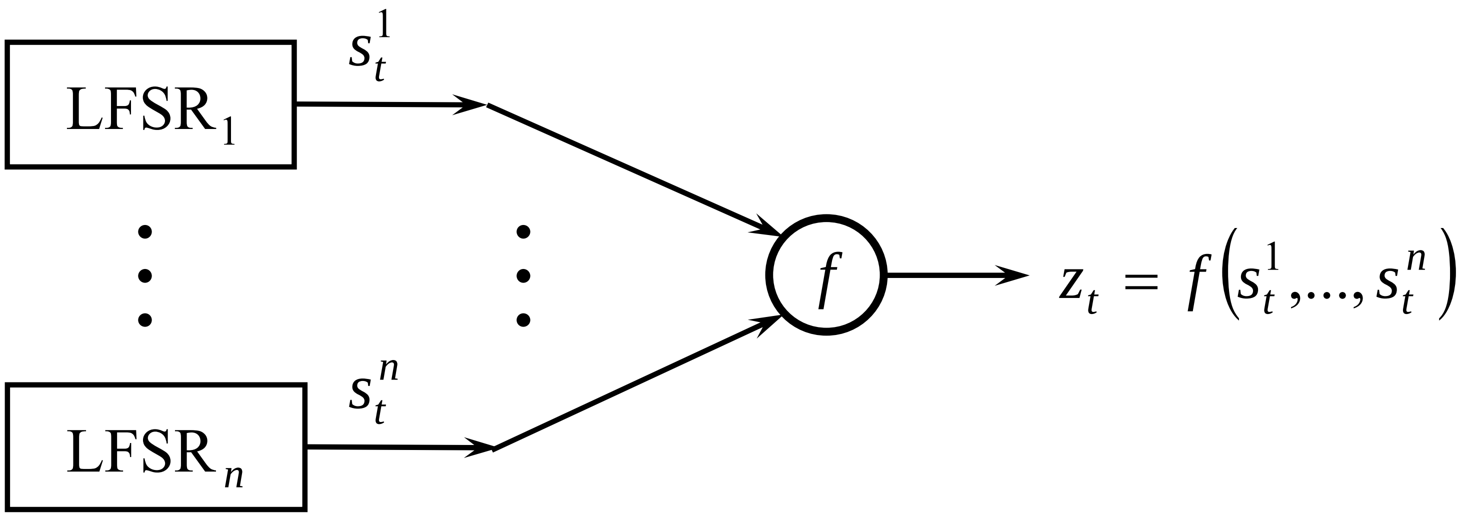
- connection integer: $q = -1 + \sum_{i=1}^{r} q_i 2^i$
- Per $s = q - 1$ iff q is a prime such that multiplicative order of 2 modulo q is $q-1$ (2 is a primitive element in $Z_q^* = \{1, 2, ..., q\text{-}1\}$)
- Good ‘long term’ randomness
- Linearity w.r.t. addition and multiplication in the ring of 2-adic numbers

- LFSR’s in Fibonacci configuration (described)
- LFSR’s in Galois configuration
    - More suitable for software implementation
    - Better short term statistics
- Linear cellular automata (programmable rules)
    - Equivalent to a set of LFSRs with correlated initial states
    - Better short term statistics
    - Some attacks are rendered more difficult
    - Design more complicated

Linear complexity

- Linear complexity, $L(s)$, of a periodic finite field sequence $s = (s_t)_{t=0}^{\infty}$ is the length of the shortest LFSR that generates s (this LFSR is unique)
- Linear complexity of a finite field string is the length of the shortest LFSR that generates this string (this LFSR need not be unique)
- Linear complexity of a keystream generator is the length of the shortest LFSR that generates all output sequences of this generator
- Linear complexity of a string of length n can be computed by well-known *Berlekamp-Massey* algorithm with complexity $O(n^2)$ symbol operations

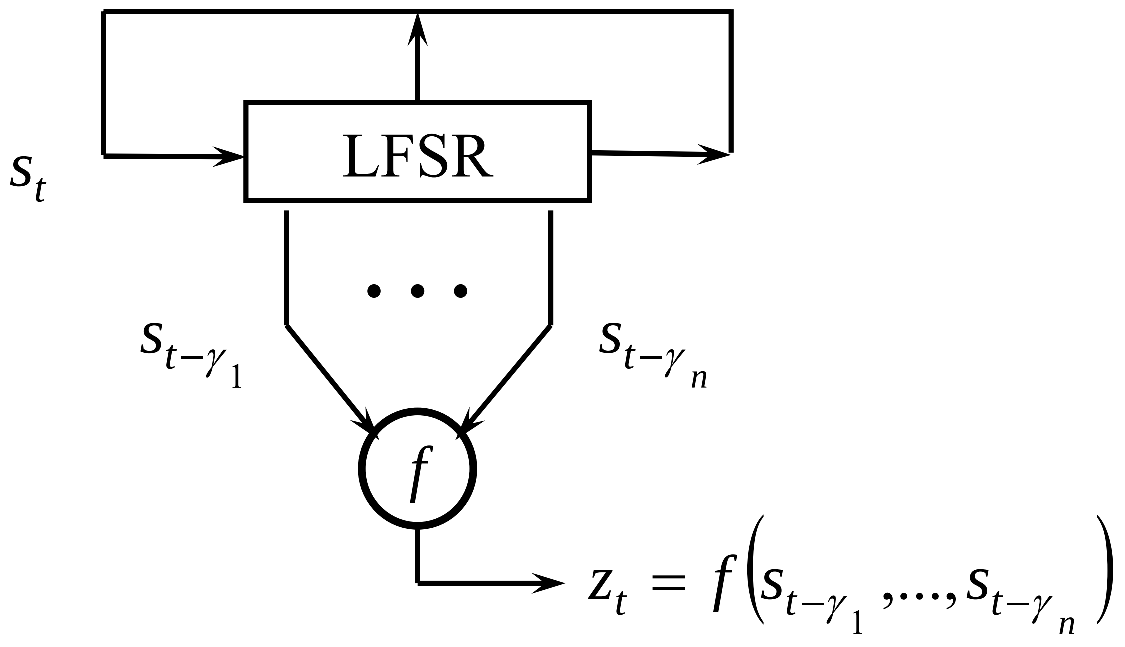
(1) <u>Regular clocking and memoryless combining function</u>



– f balanced Boolean function

– possibly, several outputs/taps from the same LFSR

– nonlinear filter generator

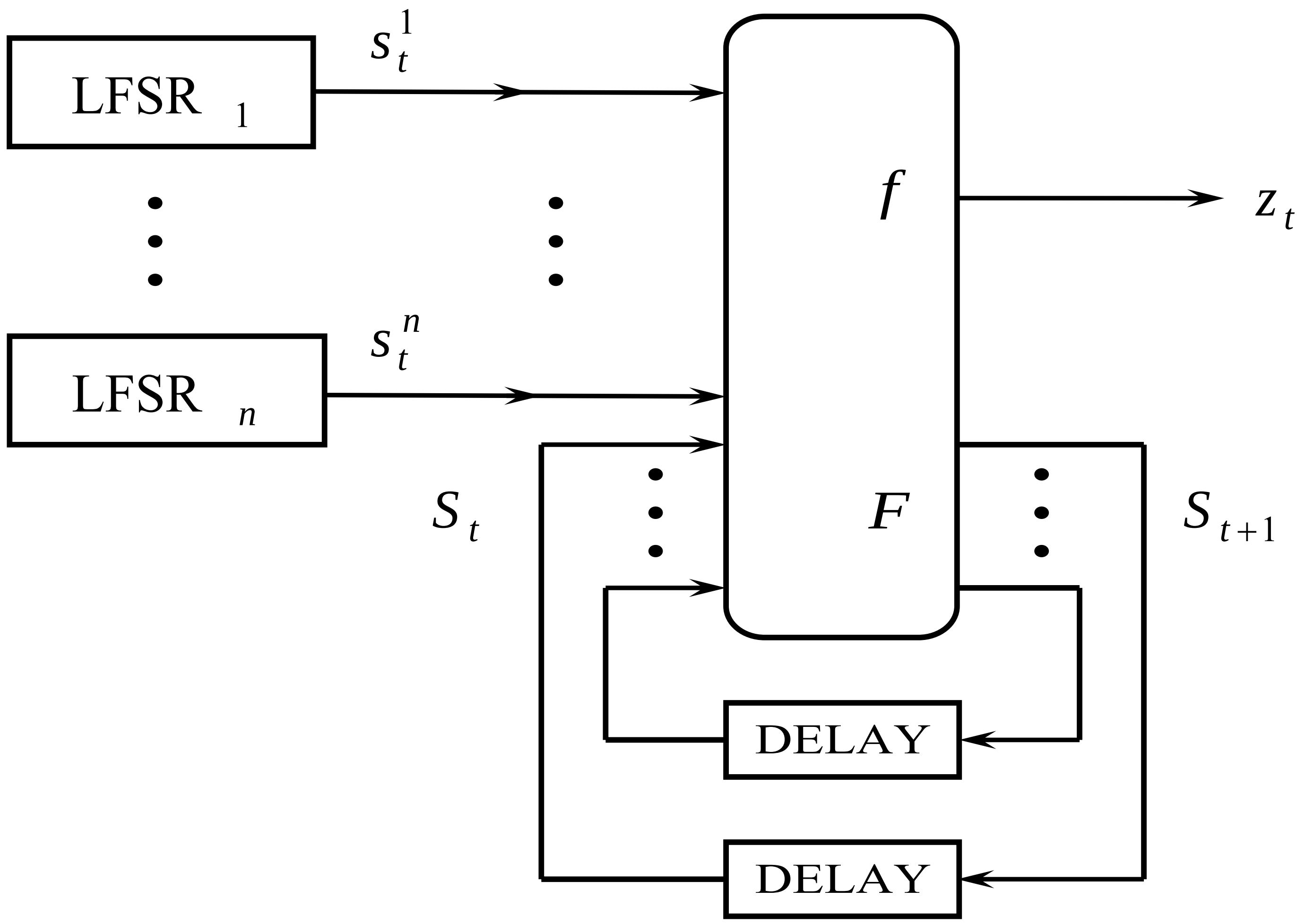


inputs to *f* defined by tapping sequence

$\gamma = (\gamma_i)_{i=1}^{n}, \;\; 1 \le \gamma_1 < \gamma_2 < \cdots < \gamma_n \le r$

– memoryless combiner: LFSR_i are distinct and have single outputs, $1 \le i \le n$

(2) Regular clocking and combining function with memory



– function with memory is a nonautonomous finite-state machine with n inputs and M bits of memory defined in terms of next-state function F and output function f

$$S_{t+1} = F\left(S_t, s_t^1, \ldots, s_t^n\right), \quad t \geq 0$$

$$z_t = f\left(S_t, s_t^1, \ldots, s_t^n\right), \quad t \geq 0$$

where S_t is M-bit internal state vector and S_0 is initial state

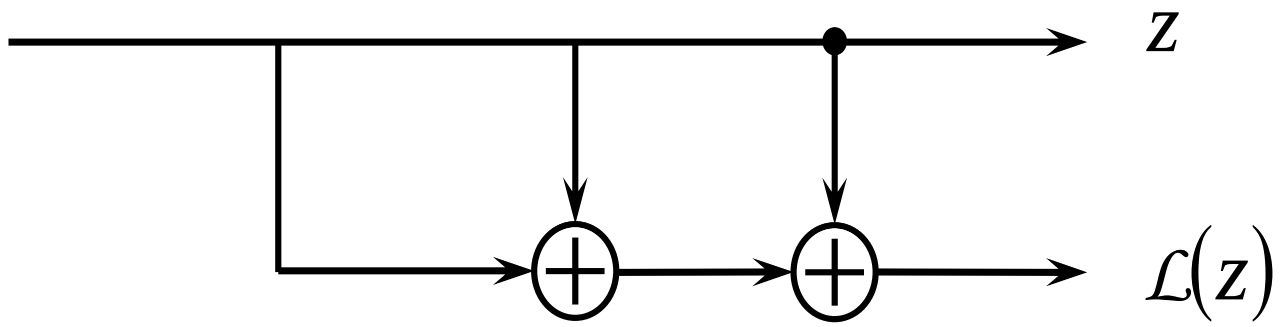
- Period and linear complexity are harder to control analytically
  - output sequence can be ultimately periodic rather than periodic, but its period is most likely at least $\mathrm{lcm}\left(P_1,\ldots,P_n\right)$
  - linear complexity is most likely very high and close to period
- Nonlinear filter generator can be viewed as combiner with finite input memory of $M=\gamma_n-\gamma_1$ bits

- Output statistics may be spoiled by internal memory (e.g., nonlinear filter generator, multiplexer generator); this is not so if $f\left(S, s^1, \ldots, s^n\right)$ is balanced for each S
- Tradeoff between correlation immunity and linear complexity can be overcome by only 1 bit of memory *Rueppel* 1985
  - e.g., maximum order of correlation immunity, $n-1$, is achieved if

$$f\left(S, s^1, \ldots, s^n\right) = g(S) \oplus s^1 \oplus \cdots \oplus s^n$$

  - e.g., summation generator with $\lceil \log_2 n \rceil$ bits of memory *Rueppel* 1985

- Correlation properties *Golić* 1992, 1996
  - one should consider correlation between linear functions of output and input bits or, more precisely, bitwise correlation between linear sequential transforms of output sequence and of input sequences



- if F is balanced, then such a correlation necessarily exists on blocks of $M+1$ consecutive output and input bits
- linear sequential circuit approximation method for finding such correlations
- to minimize correlation coefficients, M should not be small

•***Correlation attacks*** *based on the Hamming distance between linear transforms of input and output sequences or on imbalanced linear functions of input sequences when conditioned on output sequence*

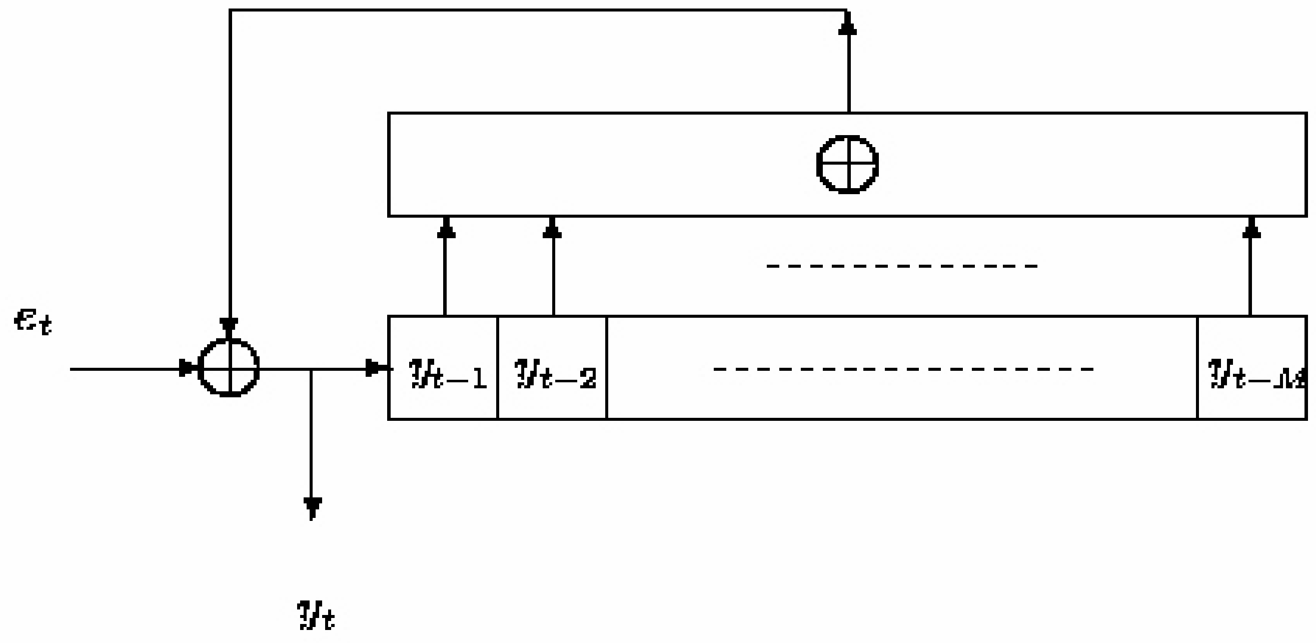
- ***<u>Linear Statistical Distinguishers</u>***

  Golić 1994, 1996

  – There exist imbalanced linear functions of M+1 consecutive output bits, assuming that initial state is uniformly distributed, for any keystream generator with M bits of internal memory

  – Resulting correlation coefficients are time-invariant for balanced next-state functions; bounds for total correlation (sum of squares) are determined

  – Equivalent representation by linear models, which can be derived for primitive components and then connected together



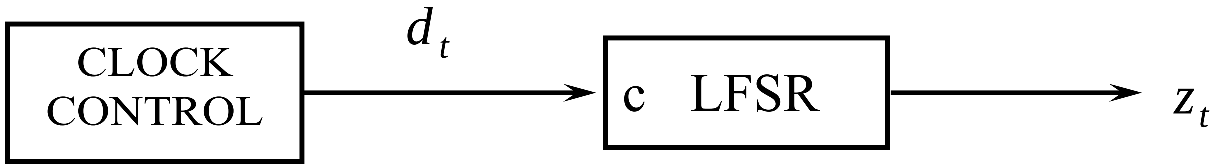
Linear model as a non-autonomous linear feedback shift register of length at most M with an additive input noise sequence of nonbalanced identically distributed binary random variables:

$$y_t = \sum_{i=1}^{M} a_i \, y_{t-i} + e_t, \quad t \geq M.$$

- **LSCA method**
  - For finding linear correlations and linear models in autonomous or non-autonomous finite-state machines
  - *Linearization:* Express output function and each component next-state function as a sum of a linear function and an imbalanced function (noise)
  - Treat noise outputs as inputs to remaining *Linear Sequential Circuit (LSC)* and solve the circuit, e.g., by generating function method
  - Linear correlations and linear models are defined by rational transfer functions
  - Resulting correlation coefficients correspond to sums of noise functions and are time-invariant for balanced next-state functions
  - *Find optimal linearizations, to maximize correlation coefficients*

(3) Irregular clocking

- Clock-controled shift registers



– regularly clocked LFSR sequence $s = (s_t)_{t=0}^{\infty}$

– decimation sequence $d = (d_t)_{t=0}^{\infty}$, d_t defining the number of clocks to be taken before z_t is produced (nonnegative)

– output sequence $z = (z_t)_{t=0}^{\infty}$ is (nonuniformly) decimated LFSR sequence

$$z_t = s\left(\sum_{i=0}^{t} d_i\right), \quad t \geq 0$$

to obtain z_t, $d_t - 1$ bits after z_{t-1} are deleted from s if $d_t \geq 1$, and z_{t-1} is repeated if $d_t = 0$

- decimation sequence is produced by clock-control generator, which may be composed of LFSR's
- if $\mathcal{D}$ is the range of values of d, LFSR is said to be $\mathcal{D}$-clocked, e.g.,
  - $\{0,1\}$-clocking – stop/go clocking
  - $\{1,2\}$-clocking
  - $\{m,n\}$-clocking
  - $[1,d+1]$-clocking – constrained clocking
  - $[1,\infty)$-clocking – unconstrained clocking

- <u>Clock-controlled memoryless combiners</u>
  - Combining Boolean function can be linear
  - Examples: clock-controlled cascades, alternating step generator, bilateral stop/go generator, A5
- <u>Clock-controlled combiners with memory</u>
  - Most general keystream generators based on LFSR's
  - Combining function should be designed so as to prevent (fast) correlation attacks when LFSR's are clocked regularly

- ***<u>Correlation attacks</u>*** *based on edit distances and edit probabilities (divide-and-conquer)*

(6) Other types of keystream generators

- Based on slowly-varying tables, such as RC4 *Rivest* 1987
  - Properties difficult to control
  - Large internal state
  - Efficient for software implementation (word-based)
- Based on iterated block ciphers
  - Complex next-state function
  - Possibly reduced number of rounds
  - Properties difficult to control
  - Multiple outputs so that speed may be comparable
  - Much larger gate count in hardware implementations
  - More difficult to cryptanalyze

- Based on one-way functions *Blum, Micali, Yao, Goldreich, Levin, Luby, Krawczyk, Håstad, Impagliazzo*
  - Next-state function essentially based on an one-way function, generic or concrete
  - One-way functions are hypothetical objects
  - *Security reduction:* any statistical weakness of keystream can be converted into an inversion attack on one-way function (i.e., keystream generator is at least as secure as one-way function)
  - Concrete properties difficult to control
  - Very inefficient for software or hardware implementations
  - In practice, a block cipher as a mapping from key to ciphertext, for a fixed plaintext, may possibly be used

Correlation Attacks

Jovan Golić

Security Innovation, Telecom Italia

Winter School on Information Security, Finse 2008, Norway

CORRELATION ATTACKS

1. **Basic Correlation Attack**
2. **Fast Correlation Attacks**

1. BASIC CORRELATION ATTACK

1.1 Zero-order Correlation Immune Boolean Function

1.2 Higher-order Correlation Immune Boolean Function

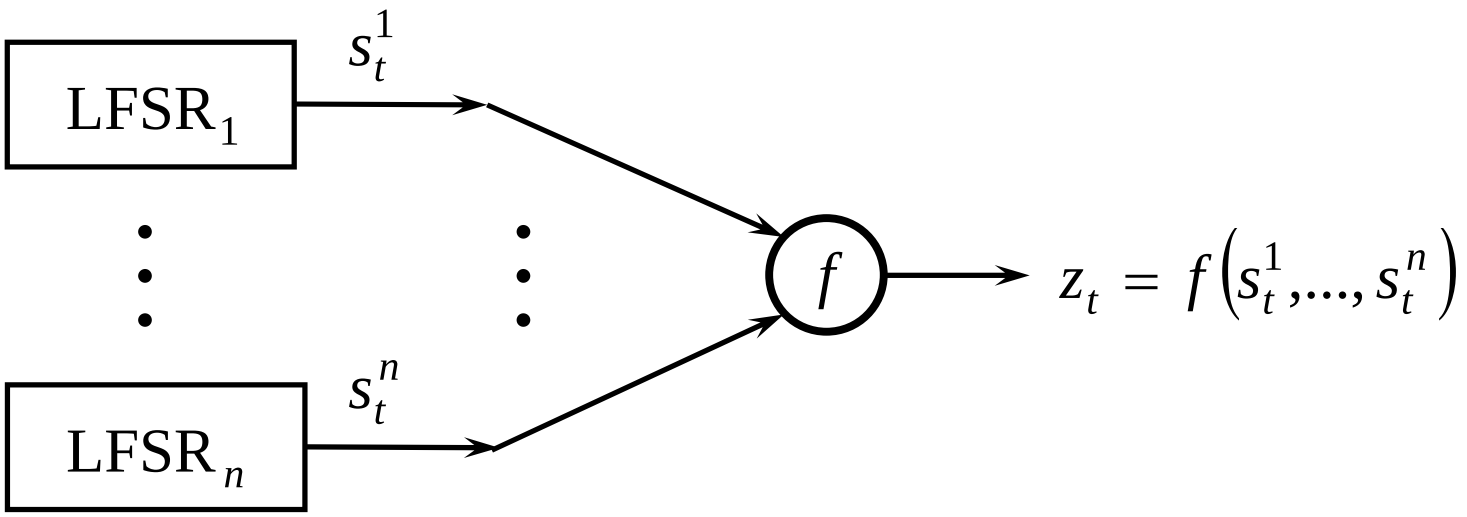
1. BASIC CORRELATION ATTACK

1.1 Zero-order correlation immune Boolean function

$f(x_1,\ldots,x_n) = x_1 + g(x_1,\ldots,x_n)$ nonbalanced g

$$\mathbf{Pr}\{g(x_1,\ldots,x_n)=\mathbf{1}\} = \mathbf{Pr}\{f(x_1,\ldots,x_n) \neq x_1\} = p \neq \tfrac{1}{2}$$

(1) Regular clocking and memoryless combining function



– f balanced Boolean function

– possibly, several outputs/taps from the same LFSR

- Correlation coefficient: $c = 1 - 2p \quad \left(p < \frac{1}{2}\right)$

- Binary symmetric channel (BSC) with output sequence $\{z_i\}_{i=0}^{N}$ and LFSR input sequence $\{a_i\}_{i=0}^{N}$, generated from assumed LFSR initial state, to be reconstructed

- Minimum Hamming distance decision rule:

$d_H\left(\{z_i\}_{i=1}^{N}, \{a_i\}_{i=1}^{N}\right)$ - minimum over assumed LFSR initial states $(\{a_i\}_{i=1}^{N})$

- If guess is correct, d_H has binomial distribution $B(N, p)$; if not, $B(N, \frac{1}{2})$
- Necessary length N for success:

$$N \geq \frac{10\ r}{c^2} \qquad \text{(for } r \text{ bits of uncertainty)}$$

or

$$N \geq \frac{r}{1 - H_2(p)} \sim \frac{r}{c^2} \left(\underset{\text{rate}}{\underset{\uparrow}{R}} = \frac{r}{N} < \underset{\text{capacity of BSC}}{\underset{\uparrow}{C}} \right)$$

1.2 Higher-order correlation immune Boolean function

f independent of any subset of m input variables and m is maximal; $m \leq n-1$; $m = n-1 \leftrightarrow f$ is linear/affine;

tradeoff between m and nonlinear order

$$f(x_1,\ldots,x_n) = \sum_{j=1}^{m+1} x_{i_j} + g(x_1,\ldots,x_n)$$ nonbalanced g

$$\sum_L c_L^2 = 1,$$ over all input linear functions L

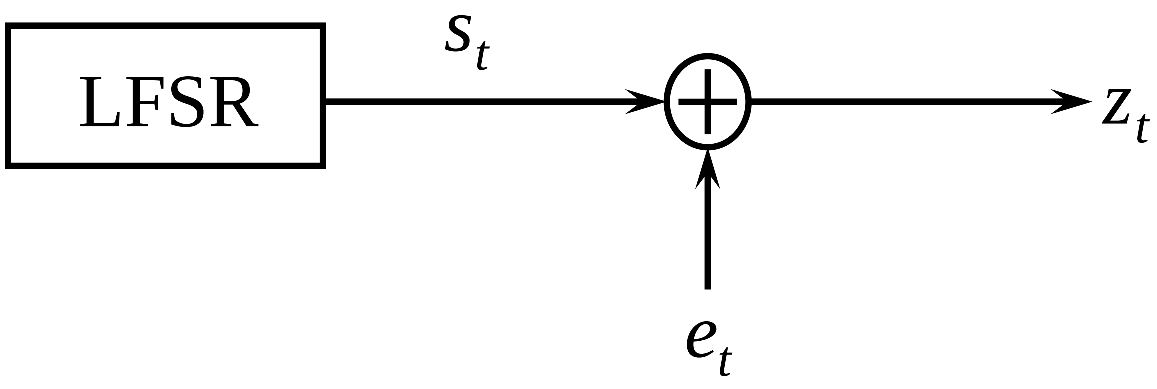
- Correlation to linear functions
  - If $l(x_1,...,x_n)$ is a linear function, then the correlation coefficient between f an l is defined by

    $$c(f,l) = \Pr(f = l) - \Pr(f \neq l) = \frac{1}{2^{n-1}} \left| \{ X \mid f(X) = l(X) \} \right| - 1$$

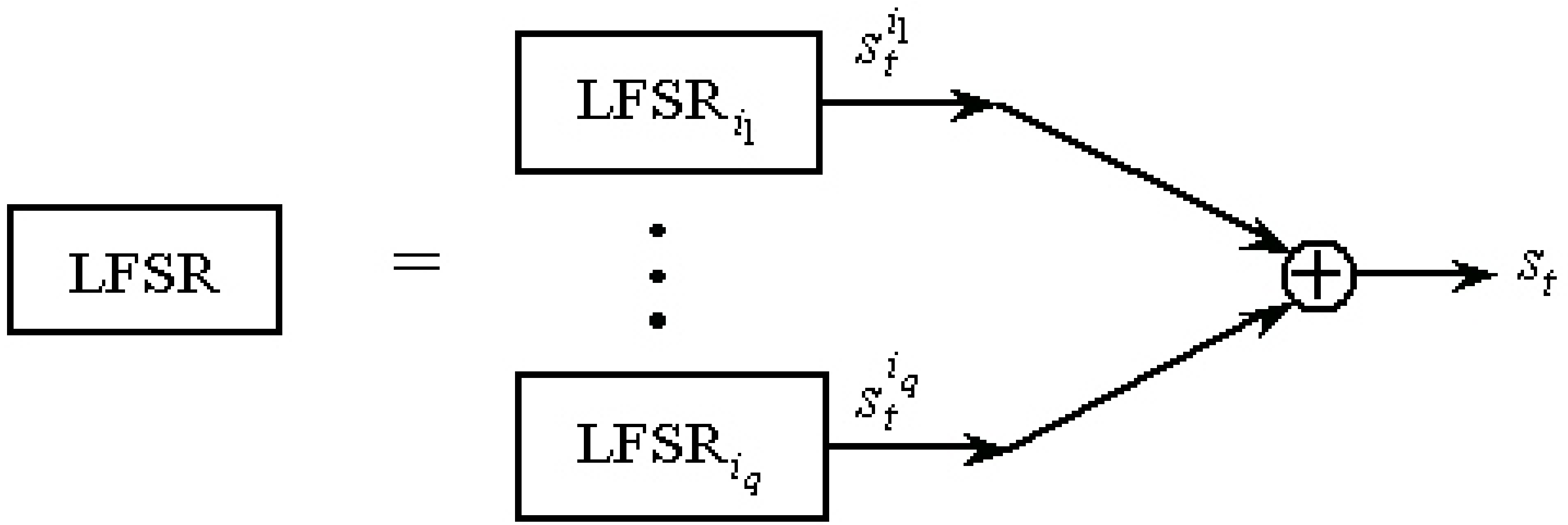
    where $X = (x_1,...,x_n)$
  - $\Pr(f = l) = \dfrac{1 + c(f,l)}{2}$
  - $\sum_l c^2(f,l) = 1$ (*Parseval's* theorem)

- f is balanced iff $c(f,0)=0$
- f is statistically independent of $(x_{i_1},\ldots,x_{i_m})$ iff f is statistically independent of every nonzero $l(x_{i_1},\ldots,x_{i_m})$ (i.e., $c(f,l)=0$)
- it follows that if f is m-th order correlation immune, then $c(f,l)=0$ for every nonzero linear function l of m (or less) input variables, and there exists l of $m+1$ input variables such that $c(f,l)\neq 0$
- if f is different from constants 0 and 1, then there exists nonzero linear function l such that $c(f,l)\neq 0$

- Probabilistic model
  - Let $c(f,l)=c\neq 0$, for $l(x_{i_1},\ldots,x_{i_q}) = x_{i_1}\oplus\cdots\oplus x_{i_q}$, for some $1\leq q\leq n$
  - Then the corresponding memoryless combiner can be modeled by



where

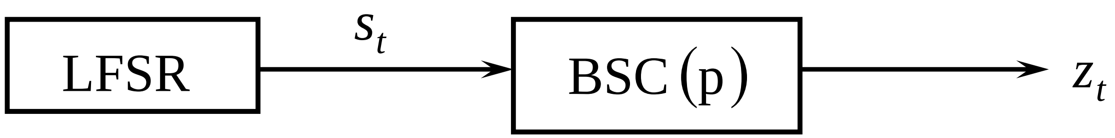


$$\Pr(e_t = 1) = p = \frac{1-c}{2} \neq \frac{1}{2}$$

(correlation noise modeling the effect of remaining LFSR's)

– If feedback polynomials of involved LFSR's are distinct, then LFSR feedback polynomial is their product

– If in addition, e_t and s_t are assumed to be statistically independent (true if f is balanced), then the model reduces to binary symmetric channel $\mathrm{BSC}(p)$



where truncated LFSR sequences represent a (truncated cyclic) linear block code

– If LFSR feedback polynomials are known, cryptanalytic problem is to reconstruct LFSR initial state from output segment of sufficient length N (divide-and-conquer)

– This problem is then equivalent to decoding problem

– Basic correlation attack is equivalent to minimum distance decoding (for $p < 1/2$):
  – guess LFSR initial state and compute Hamming distance between s^N and z^N
  – accept guess if Hamming distance is minimal, or close to being minimal (Hamming distance is around Np for correct guess and around $N/2$ for incorrect guess)
  – necessary keystream length:

$$N \approx 10 \frac{L_{i_1} + \cdots + L_{i_q}}{c^2}$$

  – computational complexity $O\left(2^{L_{i_1} + \cdots + L_{i_q}}\right)$

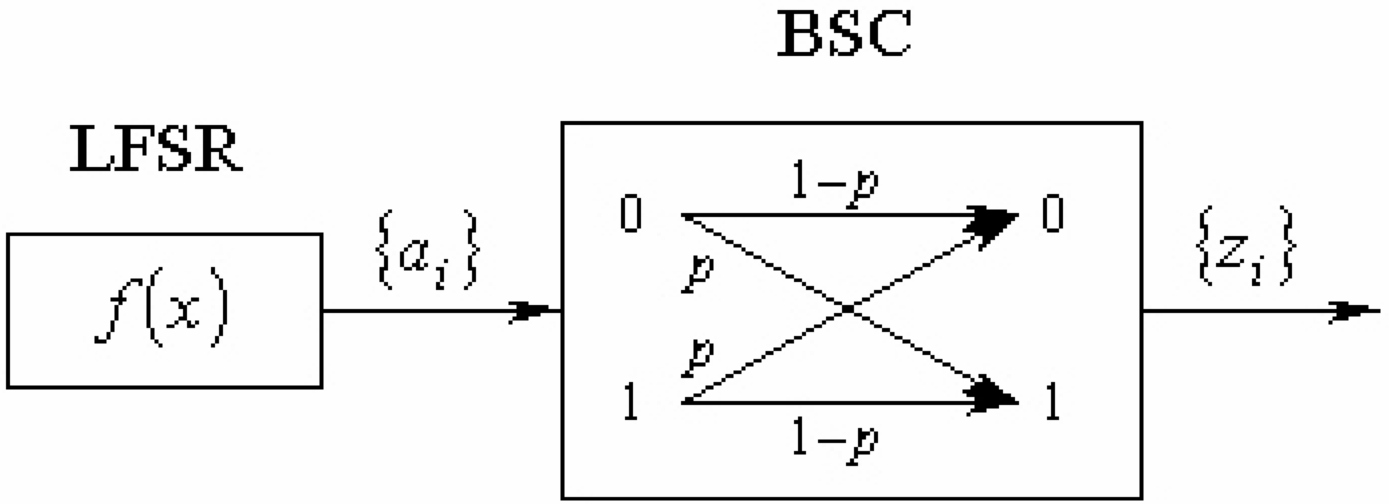
2. FAST CORRELATION ATTACKS

2. FAST CORRELATION ATTACKS

2.1 Probabilistic Model

- Observed keystream sequence $z = \{z_i\}_{i=1}^{N}$ is regarded as a noise-corrupted version of the LFSR sequence $a = \{a_i\}_{i=1}^{N}$, that is, $z_i = a_i + e_i$ where $e = \{e_i\}_{i=1}^{N}$ is a binary noise sequence
- This model is called a Binary Symmetric (Memoryless) Channel, BSC

- The correlation coefficient is defined as $c = 1 - 2p$
- Let $f(x)$ be the feedback polynomial of the LFSR of length r
- The problem is to reconstruct the LFSR sequence, $a = \{a_i\}_{i=1}^N$, from the observed keystream sequence, $z = \{z_i\}_{i=1}^N$



- We will consider two iterative error-correction algorithms based on majority-logic and probabilistic (Bayesian) decoding, respectively

- Each iteration consists of two main stages:
  - first, the parity-checks are recalculated bit-by-bit
  - second, either the majority-logic or the Bayesian bit-by-bit error-correction is made

2.2 Parity-Checks

- A parity-check is any linear relationship satisfied by a LFSR sequence. Parity-checks correspond to polynomial mutiples of $f(x)$.
- Repeated squaring is a simple algebraic procedure for deriving a set of parity-checks. However, this method requires a long segment of the observed keystream sequence and hence does not use the observed keystream sequence efficiently. It may be applied to low-weight polynomials $f(x)$.

- We use a polynomial residue method to generate more parity-checks and thus exploit the observed keystream sequence more efficiently.
  - Let $h(x)$ be a polynomial multiple of $f(x)$ of degree at most $M \geq r$ such that $h(0) = 1$. In order to find all the polynomial multiples of a certain weight (the number of non-zero terms) W and of degree at most M, the residues of the monomials x^m modulo $f(x)$, for all $r < m \leq M$, are first found.
  - If any combination of $W - 1$ of the stored residues sums up to one, the sum of the corresponding monomials plus one is a multiple of $f(x)$.
  - Time-memory tradeoff possible.

- A set of polynomial multiples such that no monomial appears in more than one polynomial multiple is called a set of orthogonal parity-check polynomials. If the exponents of all the monomials in all the parity-check polynomials correspond to a full positive difference set, it is possible to use all different phase shifts of the parity-check equations.
  - In this case, for every bit, z_i, where $M \le i \le N-M$, the number of parity-checks is W times the number of polynomial multiples.
  - In general, in order to preserve orthogonality some phases of some parity-check polynomials are not used.
- Let W denote the weight of a parity-check polynomial and let $w = W - 1$ denote the weight of the corresponding parity-check.

2.3 Majority-Logic Error-Correction Algorithm

- *Assume* that all the parity-check polynomials have the same weight.
- *Input*: The observed keystream sequence $z = \{z_i\}_{i=1}^{N}$, p, and a set of orthogonal parity-checks.
- *Initialization*: Set $j = 0$ where j is the iteration index and define J, the maximum number of iterations without change in the number of satisfied parity-checks.

- *Step 1*: Calculate the number of satisfied and unsatisfied parity-checks, respectively, for each bit z_i, $i=1,\ldots,N$, of the keystream. If all the parity-checks are satisfied or if the number of satisfied parity-checks has not changed in J iterations, go to Step 3.
- *Step 2:* For each $i=1,\ldots,N$, if the number of satisfied parity-checks is smaller than the number of unsatisfied ones, then set $z_i = z_i \oplus 1$. Increment j by 1 and go to Step 1.
- *Step 3:* Set $a_i = z_i$,$i=1,\ldots,N$, and stop the procedure.
- *Output:* The reconstructed LFSR sequence is

$$\{a_i\}_{i=1}^{N} = \{z_i\}_{i=1}^{N}.$$

2.4 Probabilistic Error Correction

- Let $\Pi_i = \{\pi_k(i)\}$ be a set of orthogonal parity-checks related to the i-th bit, $i = 1,2,\ldots,N$, $\quad k = 1,2,\ldots,|\Pi_i|$.
- Let a parity-check value be defined as the modulo 2 sum $c_k(i) = \Sigma_{l\in\pi_k(i)} z_l$.
- For the random variables E_i and $C_k(i)$, for $k = 1,2,\ldots,|\Pi_i|$, the corresponding posterior probability is then for $i = 1,2,\ldots,N$ given by:

$$p_i = \mathbf{Pr}\left(E_i = 1 \middle| \{C_k(i)\}_{k=1}^{|\Pi_i|} = \{c_k(i)\}_{k=1}^{|\Pi_i|}\right)$$

$$\frac{p_i}{1-p_i} = \frac{q_i}{1-q_i} \prod_{l=1}^{|\Pi_i|} \left(\frac{q_l(i)}{1-q_l(i)}\right)^{1-2c_l(i)} \qquad (1)$$

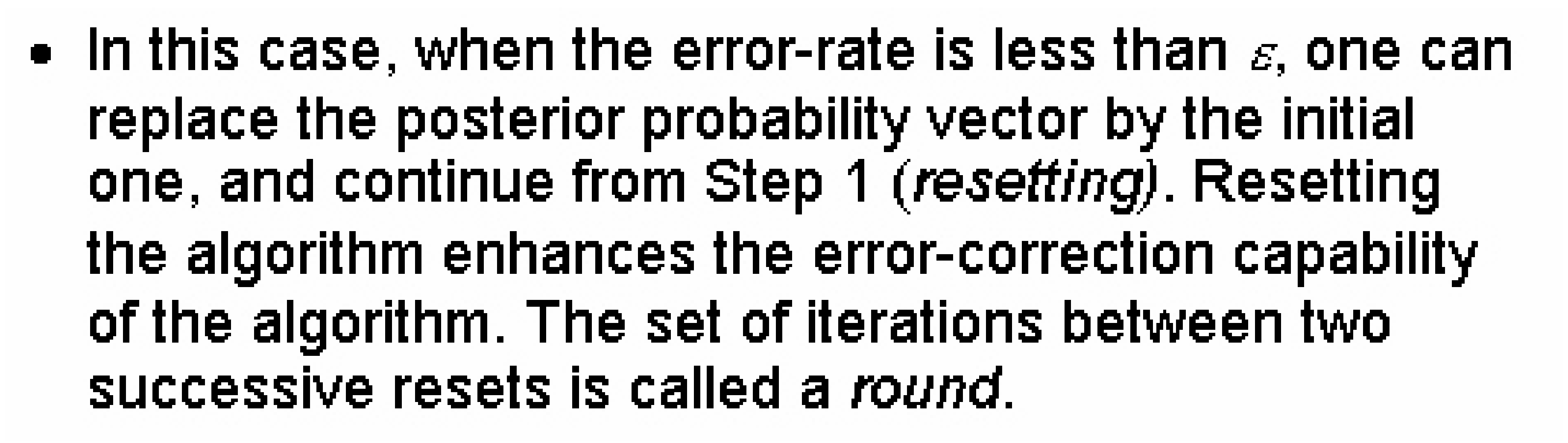
where p_i and q_i respectively denote the posterior and prior probabilities for the current iteration and

$q_l(i) = \left(1 - \Pi_{t=1}^{w}\left(1 - 2q_{m_t}\right)\right)/2$

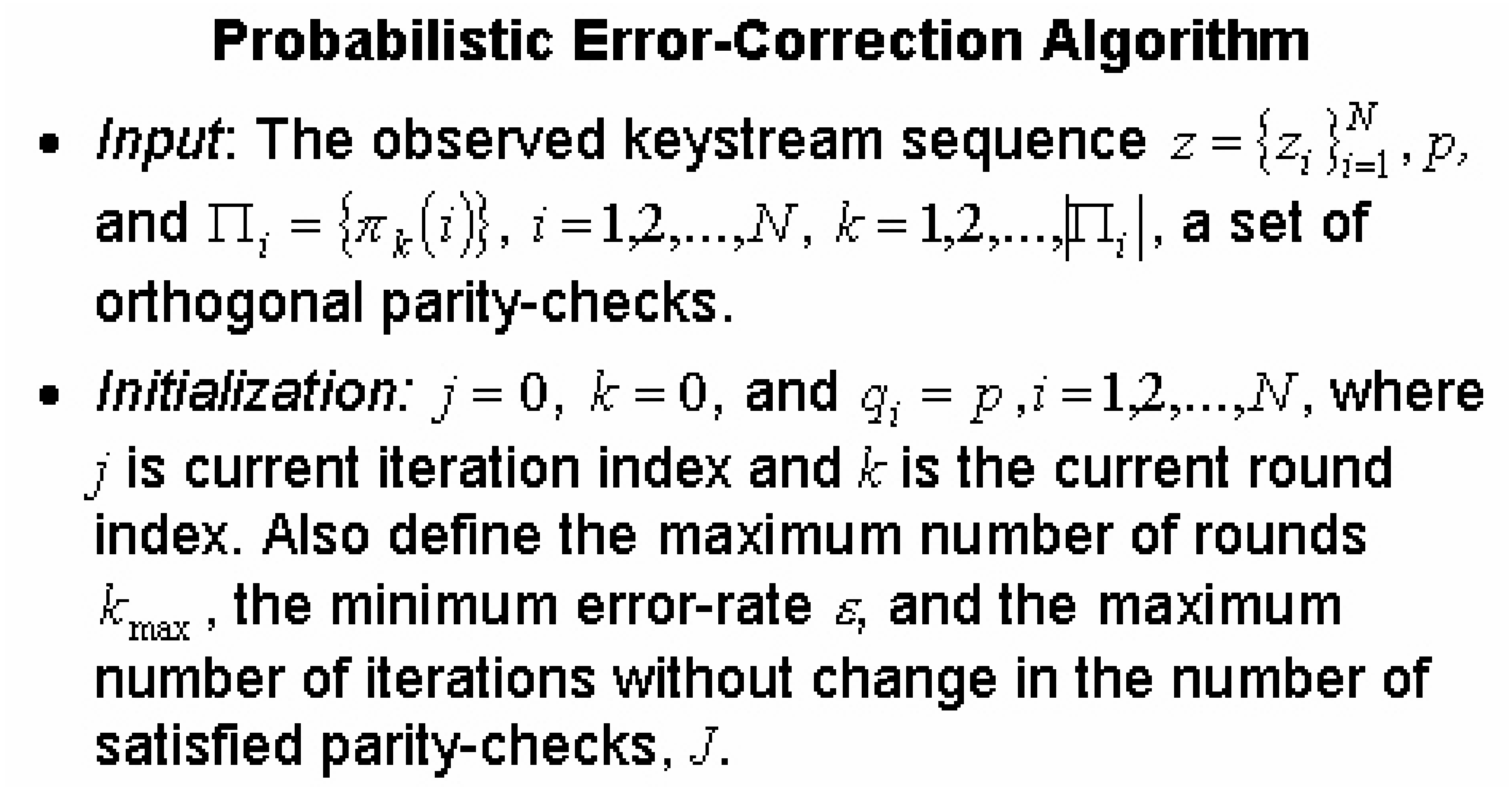
where $\{m_t\}_{t=1}^{w}$ denotes the set of indices of the bits involved in the parity-check $\pi_l(i)$ of weight w, for any $l = 1,2,\ldots,|\Pi_i|$ and $i = 1,2,\ldots,N$.

- In the first iteration, the optimal Bayesian decision is made to minimize the symbol error-rate, p_e. In the succeeding iterations, the same maximum posterior probability decision rule is applied to posterior probabilities obtained by using posterior probabilities from the previous iteration as the prior probabilities in the current iteration.

- Estimated error-rate almost always decreases due to error-correction and due to self-composition (fixed points). For large noise, the algorithm becomes trapped in a local minimum of the error-rate, $p_e \approx 0$, after which the error-correction of the observed keystream sequence ceases.



- In this case, when the error-rate is less than ε, one can replace the posterior probability vector by the initial one, and continue from Step 1 (*resetting*). Resetting the algorithm enhances the error-correction capability of the algorithm. The set of iterations between two successive resets is called a *round*.



Probabilistic Error-Correction Algorithm

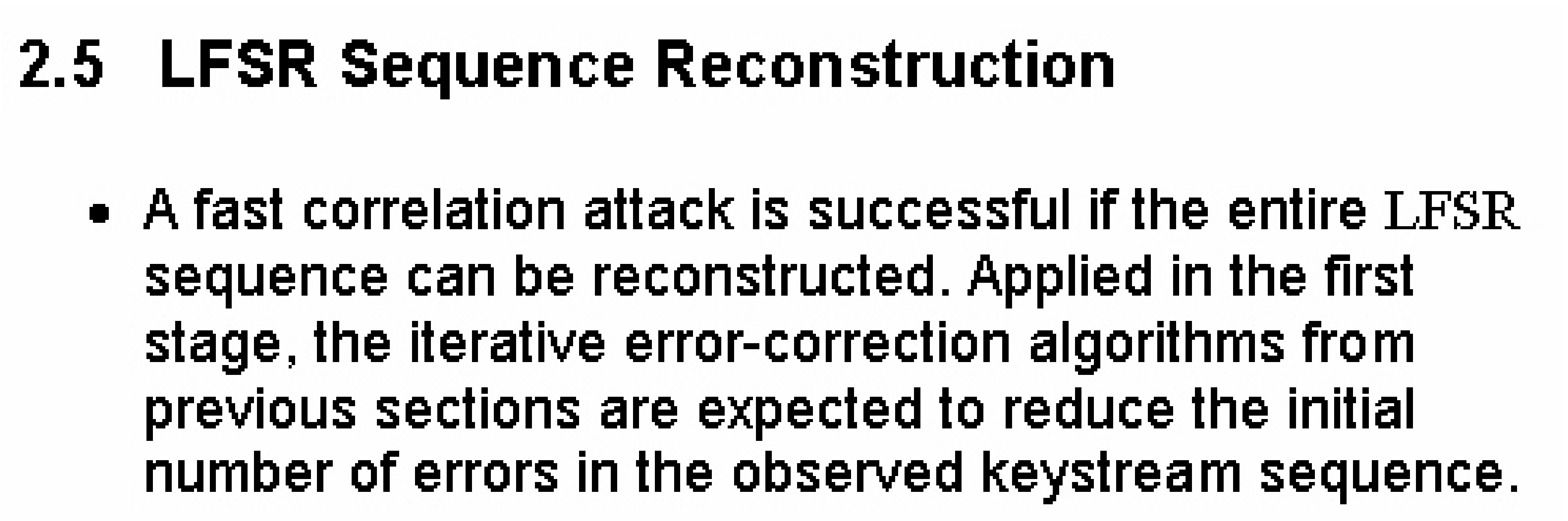
- *Input*: The observed keystream sequence $z = \{z_i\}_{i=1}^{N}$, p, and $\Pi_i = \{\pi_k(i)\}$, $i = 1,2,\ldots,N$, $k = 1,2,\ldots,|\Pi_i|$, a set of orthogonal parity-checks.
- *Initialization:* $j = 0$, $k = 0$, and $q_i = p$, $i = 1,2,\ldots,N$, where j is current iteration index and k is the current round index. Also define the maximum number of rounds $k_{\max}$, the minimum error-rate ε, and the maximum number of iterations without change in the number of satisfied parity-checks, J.

- *Resetting Criteria:* Probabilities are reset when the average per-symbol error probability (error-rate) p_e drops below ε or when the number of satisfied parity-checks has not changed for J iterations.
- *Stopping Criteria:* The algorithm stops when the number of rounds reaches $k_{\max}$, or, ideally, when all the parity-checks are satisfied.

- *Step 1:* Calculate the parity-checks $c_k(i)$ for $k=1,2,\ldots,|\Pi_i|$, $i=1,2,\ldots,N$. If all parity-checks are satisfied, go to Step 7. If the number of satisfied parity-checks has not changed for J iterations, go to Step 6.
- *Step 2:* Calculate the posterior probabilities p_i, $i=1,2,\ldots,N$, by using equation (1).
- *Step 3:* If $p_i > 0.5$, set $z_i = z_i \oplus 1$, $p_i = 1 - p_i$, $i=1,2,\ldots,N$ (error correction).

- *Step 4:* Substitute the posterior probabilities of the current iteration for the prior probabilities of the next iteration: $q_i = p_i$, for $i = 1,2,\ldots,N$.
- *Step 5:* If $p_e = \frac{1}{N}\Sigma_{i=1}^{N} p_i > \varepsilon$, increment j by 1 and go to Step 1.
- *Step 6:* Set $q_i = p$, $i = 1,2,\ldots,N$, and increment k by 1. If $k < k_{\max}$, go to Step 1.
- *Step 7:* Set $a_i = z_i$, $i = 1,2,\ldots,N$, and stop the procedure.
- *Output:* The reconstructed LFSR sequence is

$$\{a_i\}_{i=1}^{N} = \{z_i\}_{i=1}^{N}.$$



2.5 LFSR Sequence Reconstruction

- A fast correlation attack is successful if the entire LFSR sequence can be reconstructed. Applied in the first stage, the iterative error-correction algorithms from previous sections are expected to reduce the initial number of errors in the observed keystream sequence.

- In the second stage, the whole LFSR sequence has to be reconstructed. Ideally, if all the parity-checks are satisfied, then the modified observed keystream sequence is already error-free (with high probability). However, this happens only when the noise probability is relatively small.
- If some parity-checks are not satisfied, then the modified observed keystream sequence is not error-free. An efficient method to be used then is to search for an error-free *information set*, where an *information set* is any set of r linearly independent bit-positions in the LFSR sequence (r is the LFSR length).

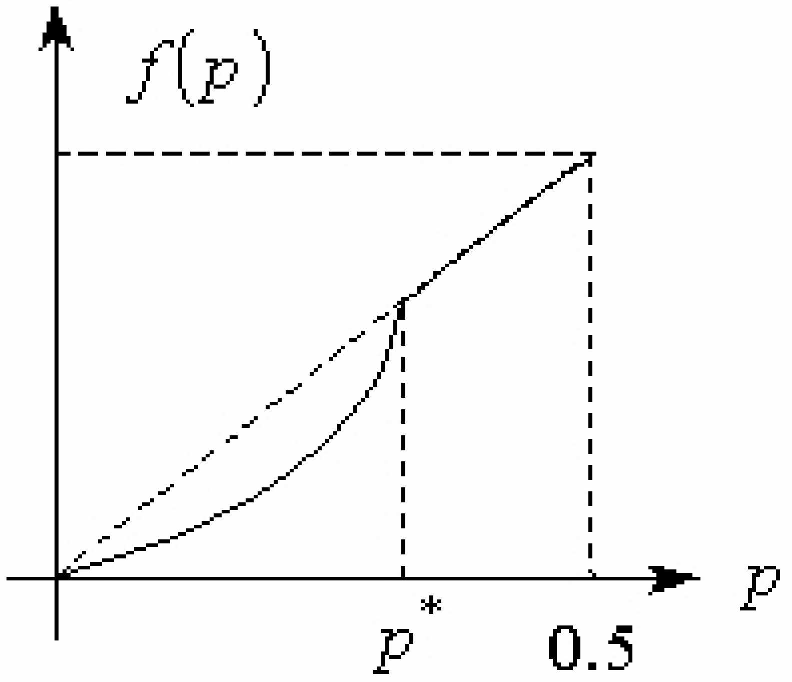
- For any assumed information set, the whole LFSR sequence is generated and then compared with the observed keystream sequence by the (statistically optimal) Hamming distance criterion. One can also allow for a small number of random complementations in the modified observed keystream sequence.
- A sliding window technique without complementations making use of the information sets of r consecutive bits is suggested. Even if the residual number of errors in the modified observed keystream sequence is large, the errors are typically patterned so that such error-free windows exist.

2.6 Convergence Condition

Mihajević, Golić 1992

- Expected (noise error) probability after the first step

$$p' = f(p)$$



- Self-composition

$$p^{(i)} = f\left(p^{(i-1)}\right), \quad p^{(0)} = p$$

$$\boxed{\begin{array}{lll} p^{(i)} \to 0 & \textbf{if} & p < p^* \\ p^{(i)} = p & \textbf{if} & p \geq p^* \end{array}}$$

- Characterization for p^*: minimum p such that

$$\frac{p}{1-p} \prod_{w \in \Omega} \left(\frac{1+(1-2p)^w}{1-(1-2p)^w} \right)^{N_w} \leq 1$$

N_w: number of parity-checks of weight w per bit

Ω: set of possible w

- Approximation for $c = 1-2p \approx 0$: $\sum_{\Omega} N_w c^{w-1} > 1$

2.7 Other Approaches

- *Modified notation*
  - Correlation coefficients for error bits:
    prior $c_i = 1 - 2p_i$ and posterior $\hat{c}_i = 1 - 2\hat{p}_i$
  - Correlation coefficients for a parity check π:

$$c_i(\pi) = \prod_{j \in \pi,\, j \neq i} c_j$$

  - For a given π, only $3(|\pi| - 2)$ multiplications are needed
  - Parity-check value: $s(\pi)$
  - Update expression (1) for orthogonal parity checks:

$$\frac{1-\hat{c}_i}{1+\hat{c}_i} = \frac{1-c_i}{1+c_i} \prod_{\pi \in \Pi_i} \left(\frac{1-c_i(\pi)}{1+c_i(\pi)} \right)^{1-2s(\pi)}$$

- ***<u>Belief propagation</u>***
  - Instead of direct recycling, probabilities are iteratively updated according to

$$\frac{1-\hat{c}_{i,\pi}^{(k)}}{1+\hat{c}_{i,\pi}^{(k)}} = \frac{1-c_i}{1+c_i} \prod_{\pi' \in \prod_i \backslash \pi} \left(\frac{1-c_i^{(k-1)}(\pi')}{1+c_i^{(k-1)}(\pi')} \right)^{1-2s(\pi')}$$

where

$$c_i^{(k-1)}(\pi') = \prod_{j \in \pi',\, j \neq i} c_{j,\pi'}^{(k-1)}$$

  - For final decisions, products over all parity checks are used
  - Number of required multiplications is only 3 times larger
  - Better error-correction capability, as information contained in parity checks is better used

• ***Approximate iterative Hartmann-Rudolph algorithm***

[*Golić 2001*]

– Hartmann-Rudolph algorithm used for optimal symbol-by-symbol (soft-decision) decoding

– Novel expression for updating probabilities

$$\hat{c}_i = \left| c_i + \sum_{\pi \in \prod_i} (-1)^{s(\pi)} c_i(\pi) \right|$$

where $|\cdot|$ is the clipping function

– Better for non-orthogonal parity checks, for example, when large parity-check weights have to be used

– Convergence condition

$$\sum_w \overline{N}_w (1-2p)^{w-1} > 1 \quad \text{or} \quad \sum_w \overline{N}_w (1-2p)^{w} > 1$$

- ***<u>One-step decoding algorithms</u>***
  - For reliable decisions on individual bits, it is required that (capacity argument)
    $$\sum_w \overline{N}_w (1-2p)^{2w} > 1$$
  - For reliable decisions on k-tuples of bits, parity checks with k concentrated terms should be used, and [*Chepyzhov et al. 2000*]
    $$\sum_w \overline{N}_w (1-2p)^{2w} > k$$
- ***<u>Guess a number of bits</u>***
  - A fixed set of k bits are guessed and parity checks with concentrated terms should be used
  - Iterative probabilistic decoding can be applied [*Mihaljević et al. 2000*]
- *<u>Both approaches require precomputation of parity checks</u>*
  - LFSR length is effectively reduced by k
  - Computation complexity increases 2^k times

- ***<u>Parity checks with memory</u>***
  - Parity checks with k concentrated terms which can be shifted along the sequence
  - Precomputation is required to find these parity checks: typically, with time-memory tradeoff techniques time and/or storage complexities are at least $2^{(r-k)/2}$ where r is LFSR length
  - Parity-check values can be viewed as received codeword bits for a convolutional code: each parity check when shifted along the sequence yields one codeword sequence

    [*Johansson, Jonnson 1999*]
  - *Parity-check weight is effectively reduced!*
  - Computation time and memory (RAM) complexities are proportional to 2^k
  - One-step Viterbi decoding or iterative probabilistic decoding for convolutional codes
  - Specialized iterative probabilistic decoding [*Golić 1999*]

- ***<u>Reconstruction of linear polynomials</u>***
  - For a primitive LFSR polynomial, keystream bits can be viewed as outputs of a Boolean function whose inputs are defined by coefficients of linear functions expressing LFSR output bits as functions of initial state
  - LFSR bits are then obtained as a linear function, defined by the initial state, of the same inputs
  - Method from [*Johansson, Jonnson 2000*] is essentially not related to the algorithm for reconstructing linear polynomials [*Goldreich, Rubinfeld, Sudan 1998*]
- ***<u>Tradeoffs in precomputation</u>***
  - [*Golic 96*]: degree $2^{\frac{r}{w-1}}$, complexity $2^{\frac{r}{2}}$
  - [*Wagner 02*]: degree $2^{\frac{r}{1+\log(w-1)}}$, complexity $(w-1)2^{\frac{r}{1+\log(w-1)}}$

- ***<u>Vectorial fast correlation attacks</u>***
  - [*Golic 02,05*] evaluate and iteratively update posterior probabilities for vectors of LFSR bits simultaneously
  - use Fast Fourier Transform
  - application to nonlinear filter generator and combiners with or without memory
- ***<u>Fast correlation attacks on clock-controlled shift registers</u>***
  - [*Golic 95*]: special parity checks; evaluate and iteratively update posterior probabilities for the number of bits deleted at a time
  - [*Golic 01*]: estimation of posterior probabilities for bits in regularly clocked LFSR; time-varying BSC
  - Computation complexity (?)

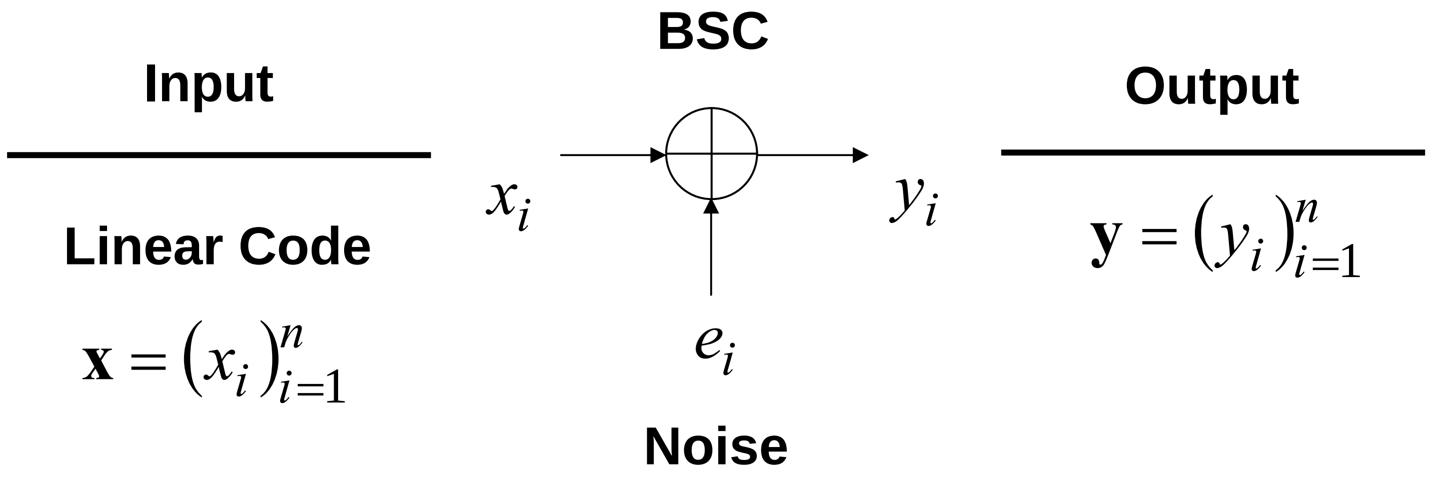
Exact Probabilistic Analysis of Memoryless Combiners

Jovan Golić

Security Innovation, Telecom Italia

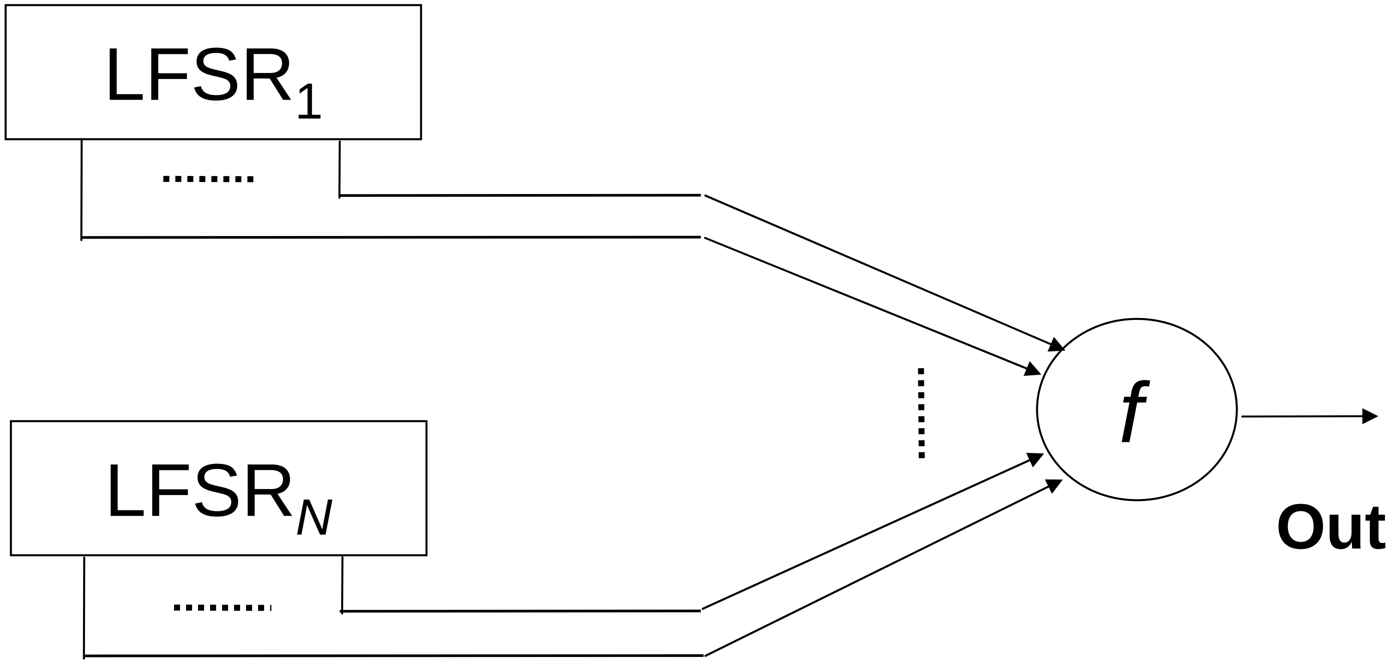
Winter School on Information Security, Finse 2008, Norway

Time-Variant Binary Symmetric Channel



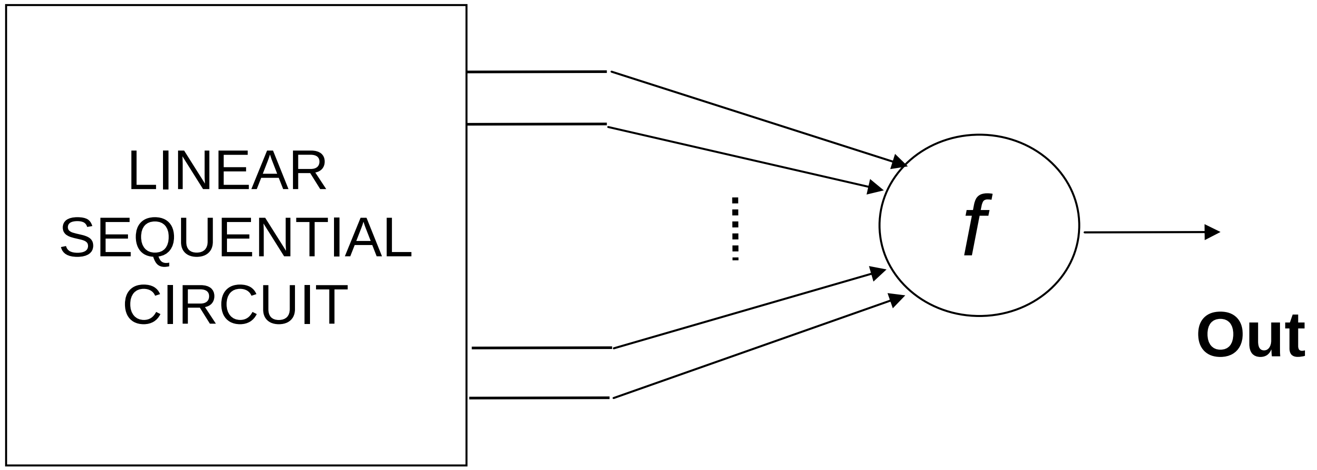
- ***Problem:*** *Probabilities of input bits given the output*
- *Applications: soft decoding, fast correlation attacks (one-step or iterative)*
- *Approximations for low-weight orthogonal parity checks*
- ***Solution:*** *Hartmann, Rudolph - IEEE Trans. Inform. Theory 1976* (exact expression)

Memoryless Combiner



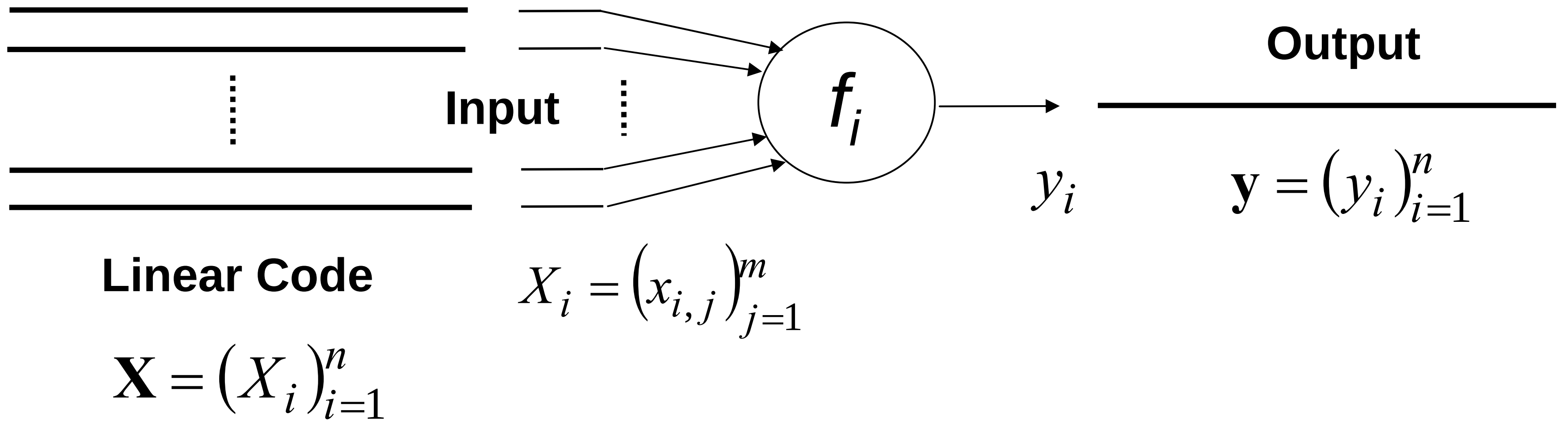
- *LFSR initial states are chosen uniformly at random*

More Generally



- *Inital state is chosen uniformly at random*

Mathematical Model



Input
f_i
Output
y_i
$\mathbf{y} = (y_i)_{i=1}^{n}$
Linear Code
$X_i = (x_{i,j})_{j=1}^{m}$
$\mathbf{X} = (X_i)_{i=1}^{n}$

- *For 2^k inputs/codewords, there are $mn - k$ independent linear relations/parity checks*
- ***For memoryless combiner, parity checks include LFSR recurrences and bit repetitions***

Problems

- ***Determine exact analytical expressions for:***
  - *Probabilities of linear functions of output bits*
  - *Probabilities of subvectors of output bits*
  - *A posteriori probabilities of linear functions of input bits, given the output sequence*
  - *A posteriori probabilities of subvectors of input bits, given the output sequence*
- Codewords are chosen uniformly at random
- Combining Boolean functions are balanced
- *Golić - IEEE Trans. Inform. Theory, Nov. 2007*

Previous Work

- ***Use of BSC model and correlations between f and linear functions (correlations)***
  - Basic correlation attack, single correlations *Siegenthaler '84, '85*
  - Iterative fast correlation attack, single correlations *Meier, Staffelbach '88, '89, Zeng, Huang '88*
  - One-step fast correlation attack, multiple correlations, nonlinear filter generator *Johansson, Jönsson '02 (LILI-128)*
- ***Use of probabilities of linear functions of input bits that are significantly different from 0.5, when conditioned on the output sequence***
  - nonlinear filter generator, **due to repetitions of input bits** *Anderson '94*

Previous Work (2)

- ***Use of combining Boolean function f and hence multiple correlations to all linear functions, repetitions of input bits not considered***
  - Nonlinear filter generator, probabilities of individual input bits, approximation for low-weight orthogonal parity checks, one-step algorithm or iterative update by using BSC model, *Leveiller, Zémor, Guillot, Boutros '02;* related linear statistical distinguisher *Molland, Helleseth '04*
  - Probabilities of input vectors, approximation for low-weight orthogonal parity checks, one-step algorithm or vectorial iterative update *Golić, Hawkes '02, '05* (independently)

Preliminaries

- Problems more difficult than for BSC model
- ***All problems can be solved mathematically***
- ***Walsh-Hadamard transform of Boolean function***

$$\mathbf{f}_i(w) = \frac{1}{2^m} \sum_{X \in \{0,1\}^m} (-1)^{f_i(X) \oplus w \cdot X}$$

$$(-1)^{f_i(X)} = \sum_{w \in \{0,1\}^m} \mathbf{f}_i(\omega)(-1)^{w \cdot X}$$

$$\mathbf{f}_i(w) = \Pr\{f_i(X) = w \cdot X\} - \Pr\{f_i(X) \neq w \cdot X\} = c(f_i(X), w \cdot \mathbf{X})$$

Preliminaries (2)

- ***Fourier transform of probability distribution***

$$\mathbf{P}(\omega) = \sum_{X \in \{0,1\}^n} P(X)(-1)^{\omega \cdot X}$$

$$P(X) = \frac{1}{2^n} \sum_{\omega \in \{0,1\}^n} \mathbf{P}(\omega)(-1)^{\omega \cdot X}$$

$$\mathbf{P}(\omega) = \Pr\{\omega \cdot X = 0\} - \Pr\{\omega \cdot X = 1\} = c(\omega \cdot \mathbf{X})$$

Some Solutions

- Correlation coefficient of output linear function

$$\omega \cdot \mathbf{y}\,, \quad \omega = (\omega_i)_{i=1}^{n}$$

$$c(\omega \cdot \mathbf{y}) \;=\; \sum_{\substack{\lambda \in C^{\perp} \\ \underline{\lambda} = \omega}} \prod_{i=1}^{n} \mathbf{f}_i(\lambda_i)^{\omega_i}$$

$$\text{dual codewords - parity checks :} \quad \lambda = (\lambda_i)_{i=1}^{n}, \quad \lambda_i = (\lambda_{i,j})_{j=1}^{m}$$

- *Support of vector v:* $\underline{v} = 1 \text{ if } v \neq 0, \; \underline{v} = 0 \text{ if } v = 0$

Some Solutions (2)

- *A posteriori* correlation coefficient of input linear function $l \cdot \mathbf{X}$

$$c(l \cdot \mathbf{X} \mid \mathbf{y}) = \Pr\{l \cdot \mathbf{X} = 0 \mid \mathbf{y}\} - \Pr\{l \cdot \mathbf{X} = 1 \mid \mathbf{y}\}$$

$$c(l \cdot \mathbf{X} \mid \mathbf{y}) = \frac{\displaystyle\sum_{\lambda \in C^{\perp}} \prod_{i=1}^{n} \left(\mathbf{f}_i(\lambda_i \oplus l_i)(-1)^{y_i}\right)^{\overline{\lambda_i \oplus l_i}}}{\displaystyle\sum_{\lambda \in C^{\perp}} \prod_{i=1}^{n} \left(\mathbf{f}_i(\lambda_i)(-1)^{y_i}\right)^{\overline{\lambda_i}}}$$

$$l = (l_i)_{i=1}^{n}, \quad l_i = (l_{i,j})_{j=1}^{m}$$

Binary Symmetric Channel

- Noise correlation coefficient $c_i = \Pr\{e_i = 0\} - \Pr\{e_i = 1\}$

$$c(\omega \cdot \mathbf{y}) = \begin{cases} \prod_{i=1}^{n} c_i^{\omega_i} & \text{if} \quad \omega \in C^{\perp} \\ 0 \quad \text{if} & \omega \notin C^{\perp} \end{cases}$$

$$c(l \cdot \mathbf{X} \mid \mathbf{y}) = \frac{\sum_{\lambda \in C^{\perp}} \prod_{i=1}^{n} \left(c_i (-1)^{y_i} \right)^{\lambda_i \oplus l_i}}{\sum_{\lambda \in C^{\perp}} \prod_{i=1}^{n} \left(c_i (-1)^{y_i} \right)^{\lambda_i}}$$

Binary Symmetric Channel (2)

- *A posteriori* subvector probability distribution

$$P(\mathbf{x}_I \mid \mathbf{y}) = \frac{\prod_{i \in I}^{n} \left(1 + c_i (-1)^{y_i \oplus x_i}\right)}{2^{|I|}} \frac{\sum_{\lambda \in C^{\perp}} (-1)^{\lambda_I \cdot \mathbf{x}_I \oplus \lambda_{\bar{I}} \cdot \mathbf{y}_{\bar{I}}} \prod_{i \in \bar{I}}^{n} c_i^{\lambda_i}}{\sum_{\lambda \in C^{\perp}} \prod_{i=1}^{n} \left(c_i (-1)^{y_i}\right)^{\lambda_i}}$$

- *Chepyzhov, Johansson, Smeets 2000* one-step fast correlation attack is not statistically optimal *(left term missing)*

Applications

- ***By selecting low-weight parity checks, novel, computable expressions can be obtained***
- Generality: arbitrary LFSR polynomials, non-orthogonal parity checks, repetitions of input bits, other linear dependences among input bits
- Usage:
  - Statistical distinguishers
  - One-step fast correlation attacks, for linear functions or subvectors of input bits
  - Iterative fast correlation attacks, new expressions for updating the underlying correlation coefficients

Applications (2)

- *Statistical distinguishers:* $c(\omega\cdot\mathbf{y}) \approx \sum_{\substack{w\subseteq\Theta_\omega \\ w\neq\mathbf{0}}} \mathbf{f}(w)^{\|\omega\|}$

$\Theta_\omega = (\theta_j)_{j=1}^{m}, \quad \theta_j = 1$ if $p_j \mid h_\omega$, and $\theta_j = 0$ otherwis

- *Iterative fast correlation attacks:*

$$c_i(v\cdot X_i)^{(k+1)} = c_i(v\cdot X_i)^{(k)} + \sum_{\omega\in \underline{C_i^\perp}} \sum_{\substack{w\subseteq\Theta_\omega \\ w\neq\mathbf{0}}} c_i((w\oplus v)\cdot X_i)^{(k)} \prod_{\substack{j=1,\ j\neq i \\ \omega_j=1}}^{n} c_j(w\cdot X_i)^{(k)}$$

$$c_i(v\cdot X_i)^{(0)} = \mathbf{f}_i(v)(-1)^{y_i}, \quad c_i(0)^{(k)} = 1$$

$\underline{C_i^\perp}$ *- set of supports of dual codewords* λ *s.t.* $\lambda_i \neq \mathbf{0}$

Algebraic Attacks

Jovan Golić

Security Innovation, Telecom Italia

Winter School on Information Security, Finse 2008, Norway

Introduction

- ***Objective:*** Reconstruct secret key by manipulating and solving underlying algebraic equations
- ***Linearization Algorithm:*** Replace monomials by new variables and solve the linear system if number of equations is sufficiently large; necessary that equations have low algebraic degree
- ***XL Algorithm [SPCK00]:*** Generate new equations by multiplying original equations by monomials of bounded degree and then apply linearization algorithm; necessary that original system is overdefined

Application to Block Ciphers

- Find low-degree algebraic equations for nonlinear components (S-boxes) and apply XL algorithm
- Exploit sparseness of resulting equations [CP02]
- Alternatively, use Gröbner basis algorithms
- Complexity grows polynomially with number of rounds
- Effectiveness is limited due to large number of new variables and possible linear dependences

Application to Stream Ciphers

- Assume that keystream is produced by applying a Boolean function to an autonomous linear finite-state machine (e.g., memoryless combiners or combiners with finite input memory such as nonlinear filter generator)
- Find (induced) low-degree algebraic equations for this Boolean function [CM03]
- Express input variables as linear functions of secret key, for each keystream bit; algebraic degree is preserved
- Apply linearization algorithm, for a sufficiently long keystream [PF-S00]

Induced Algebraic Degrees

- Boolean functions [CM03]: for *k* variables, there exists an equation of degree at most $\lceil k/2 \rceil$
- Combiners with memory [AK03]: for a binary combiner with *k* inputs and *l* bits of memory, there exists an equation over *l*+1 consecutive *k*-bit inputs of degree at most $\lceil k(l+1)/2 \rceil$, when conditioned on *l*+1 consecutive output bits
- Combiners with *m* outputs [C04]: there exists an equation over $\lceil (l+1)/m \rceil$ consecutive inputs and outputs of degree at most $\lceil k\lceil (l+1)/m \rceil /2 \rceil$ in input bits

Fast Algebraic Attacks

- For a Boolean function of k variables, if $e+d \geq k$, then there exists an equation over input and output variables of degree at most d in input variables such that the monomials whose degrees in input variables are larger than e do not depend on output variable [C03]
- Monomials with highest degrees in input variables can be eliminated from equations by using stream cipher linear complexity properties (pre-computation complexity reconsidered in [HR04])
- Algebraic degree effectively reduced to at most e

Induced Algebraic Equations [G06]

- **Objective:** study algebraic equations induced by a *vectorial Boolean function* f, i.e., by equation $f(X,Y) = Z$
- An algebraic equation is specified by a (non-zero) Boolean function, defined by a multivariate binary polynomial (algebraic normal form), required to be equal to zero on a subset of input values (e.g., determined by known Z)
- **Conditional scenario:** Z is fixed and algebraic equations involve only X
- **Unconditional scenario:** Z is variable and algebraic equations involve both X and Z
- *Divide-and-conquer:* Y has to be eliminated

Mathematical Foundations

- For any subset $\mathcal{S} \subset \{0,1\}^n$, a non-trivial algebraic equation induced by $\mathcal{S}$ is an equation $g(S)=0$, $S \in \mathcal{S}$, where g is a non-zero multivariate polynomial in $S=(s_1, \ldots, s_n)$
- The set of all such g is a vector space, as it is closed under addition
- *Algebraic equations of low degree are defined by polynomials of degree at most d*
- *They also form a vector space*

Mathematical Foundations (2)

- Let $\wp_n^d$ denote the set of all multivariate polynomials in *n* variables of degree at most *d;* any such polynomial can be characterized as a linear combination of all $\sum_{i=0}^{d}\binom{n}{i}$ monomials of degree at most d
- Let $\mathbf{P}_n^d(\mathbb{S})$ be a matrix whose rows and columns are indexed by the vectors from $\mathbb{S}$ and by the monomials from $\wp_n^d$, resp., with entries defined by evaluating the monomials on these vectors

Mathematical Foundations (3)

- For any $\mathcal{S} \subseteq \{0,1\}^n$, there exists a non-trivial algebraic equation on $\mathcal{S}$ iff $|\mathcal{S}| < 2^n$
- For any $\mathcal{S} \subset \{0,1\}^n$ and $g \in \mathcal{P}_n^d$, g defines a non-trivial algebraic equation on $\mathcal{S}$ iff the columns of $\mathbf{P}_n^d(\mathcal{S})$ corresponding to the monomials in g add up to zero
- For any $\mathcal{S} \subset \{0,1\}^n$, there exists a $g \in \mathcal{P}_n^d$ that defines a non-trivial algebraic equation on $\mathcal{S}$ if

$$\sum_{i=0}^{d} \binom{n}{i} > |\mathcal{S}|$$

Basic Algorithm

- If the binary coefficients defining any $g \in \mathcal{P}_n^d$ as a linear combination of all $\sum_{i=0}^{d} \binom{n}{i}$ monomials of degree at most *d* are represented as a binary one-column matrix **C**, then the vector space of all algebraic equations induced by $\mathcal{S}$ can be obtained by solving the system $\mathbf{P}_n^d(\mathcal{S}) \cdot \mathbf{C} = \mathbf{0}$ of linear equations in **C**, e.g., by Gaussian elimination
- Time complexity: $O\left(|\mathcal{S}| \left(\sum_{i=0}^{d} \binom{n}{i} \right)^2 \right) \quad \rightarrow \quad O(2^{3n})$
- Space complexity: $O\left(|\mathcal{S}| \sum_{i=0}^{d} \binom{n}{i} \right) \quad \rightarrow \quad O(2^{2n})$
- Dimension: $\sum_{i=0}^{d} \binom{n}{i} - \text{rank } \mathbf{P}_n^d(\mathcal{S})$

Other Algorithms

- Polynomials defining induced algebraic equations form an ideal in the ring of binary multivariate polynomials
- Gröbner basis algorithms may be useful for finding induced algebraic equations of low degree, as low-degree elements of a basis may yield low degree polynomials via (polynomial) linear combinations
- *For a Boolean function, multivariate polynomial interpolation algorithm [ACGKMR06]; time complexity*
  $O\left(\left(\sum_{i=0}^{d}\binom{n}{i}\right)^2\right)$
- If equations are allowed to hold with high probability instead of certainty, are there more efficient algorithms?

Conditional Algebraic Equations

- *Conditional scenario:* Given a vectorial Boolean function $Z = f(X,Y)$, find algebraic equations over X when Z is assumed to be fixed and known
- Variables in Y should be eliminated
  - To eliminate internal memory (explosion of variables)
  - To achieve divide-and-conquer effect
- This scenario is interesting for the cryptanalysis of stream ciphers in the known keystream sequence scenario, provided that f is used as an output function producing a binary keystream sequence from an internal state sequence generated by a next-state function
- *Sufficient conditions in* [Golić, "Vectorial Boolean functions and induced algebraic equations," *IEEE IT*, 2006]

Unconditional Algebraic Equations

- *Unconditional scenario:* Given a vectorial Boolean function $Z = f(X,Y)$, find algebraic equations over X, Z (variables in Y should be eliminated)
- This scenario is interesting for the cryptanalysis of block and stream ciphers where f is an internal function whose output is unknown, e.g., corresponding to an intermediate round of a product block cipher or to an intermediate stage of an iterated construction of an output function in a stream cipher
- *Sufficient conditions in* [Golić, "Vectorial Boolean functions and induced algebraic equations," *IEEE IT*, 2006]

Constrained Unconditional Algebraic Equations

- *Constrained unconditional scenario:* Given a vectorial Boolean function $Z = f(X, Y)$, find algebraic equations over X, Z having any degree in Z such that the monomials with the highest degrees in X do not depend on Z (variables in Y should be eliminated)
- This scenario is interesting for fast algebraic attacks, as high-degree monomials in X may be eliminated by using linear complexity properties of the corresponding stream cipher
- *Sufficient conditions in* [Golić, "Vectorial Boolean functions and induced algebraic equations," *IEEE IT*, 2006]

Combiners with Memory

- In all three scenarios, various sufficient conditions for the existence of induced algebraic equations are derived, thus generalizing previously known results
- *Importance of the range of vectorial Boolean function*
- *Finite input memory (e.g., nonlinear filter generator) can considerably reduce the degree of induced algebraic equations*
- *A divide-and-conquer effect, especially for multiple binary outputs, can significantly reduce the complexity of algebraic attacks*
  - For example, for combiner with k LFSRs and m outputs, $2 \leq m \leq k$, any subset of at most m-1 LFSRs can be eliminated, regardless of how many outputs are used in each LFSR

Algebraic Immunity Order

- Consider a vectorial Boolean function $f:\ \{0,1\}^n \to \{0,1\}^m$, $Z = f(X)$; let $m_f = \log|f_z|$, where f_Z is the range of f
- Let X' denote a generic subset of variables in X of size $n_{x'} = |X'|$ and let $X \setminus X'$ denote the subset of the remaining variables in X having size $n - n_{x'}$
- We can then write $Z = f(X', X \setminus X')$ and consider algebraic equations involving X' that are induced by f
- **There are no induced (conditional) algebraic equations over X' for any Z iff for each fixed value of X', the set of values of $f(X', X \setminus X')$ for all values of $X \setminus X'$ equals f_Z** [G05,06]

Algebraic Immunity Order (2)

- The maximal $k = |X'|$ such that this property holds for every *X'* is called the *algebraic immunity order* of f; if $f_z = \{0,1\}^m$, then it is called the *algebraic resiliency order* [G05,06]
- Generalization of correlation immunity/resiliency order, where $f(X', X \setminus X')$ is considered as multiset, i.e., distribution of output values matters
- $k \leq n - m_f$, not smaller than correlation immunity order
- **There are no induced (unconditional) algebraic equations over *X'*, *Z* iff for each fixed value of *X'*, the set of output values of** $f(X', X \setminus X')$ **for all values of** $X \setminus X'$ **equals** $\{0,1\}^m$ [PZG05]

Induced Algebraic Degree Profile

- For each $k+1 \le i \le n$, we can find the minimal degree, d_i, of induced non-trivial algebraic equations over all subsets *X'* of size *i* and over all output values Z
- The resulting non-increasing sequence $(d_i)_{i=k+1}^{n}$ can be called the *induced algebraic degree profile* of f
- The minimal value, d_n, can be called the *induced algebraic degree* of f and denoted as d_f, which is also known as algebraic immunity [MPC04]
- d_f is smaller than or equal to the minimal *d* such that
- For m_f= 1, $d \le \lceil n/2 \rceil$ [CM03]

$$\sum_{i=0}^{d} \binom{n}{i} > 2^{n-m_f}$$

Some Criteria for Cryptographic Functions

- Algebraic immunity order should be close to the maximum, $n - m_f$
- Induced algebraic degree should be close to the maximum, which is the minimum d such that

$$\sum_{i=0}^{d} \binom{n}{i} > 2^{n-m_f}$$

- Induced algebraic profile $(d_i)_{i=k+1}^{n}$ should not be too flat; namely, induced algebraic equations involving less terms could have higher degrees
- New research problems