Stream cipher

A *stream cipher* is a symmetric cipher which operates with a time-varying transformation on individual plaintext digits. By contrast, block ciphers operate with a fixed transformation on large blocks of plaintext digits. More precisely, in a stream cipher a sequence of plaintext digits, $m_0 m_1 \ldots$, is encrypted into a sequence of ciphertext digits $c_0 c_1 \ldots$ as follows: a pseudo-random sequence $s_0 s_1 \ldots$, called the running-key or the *keystream*, is produced by a finite state automaton whose initial state is determined by a secret key. The i-th keystream digit only depends on the secret key and on the $(i-1)$ previous plaintext digits. Then, the i-th ciphertext digit is obtained by combining the i-th plaintext digit with the i-th keystream digit.

Stream ciphers are classified into two types: synchronous stream ciphers and asynchronous stream ciphers. The most famous stream cipher is the Vernam cipher, also called *one-time pad*, that leads to perfect secrecy (the ciphertext gives no information about the plaintext).

Stream ciphers have several advantages which make them suitable for some applications. Most notably, they are usually faster and have a lower hardware complexity than block ciphers. They are also appropriate when buffering is limited, since the digits are individually encrypted and decrypted. Moreover, synchronous stream ciphers are not affected by error-propagation.

Anne Canteaut.

References

[Rue86] R.A. Rueppel. *Analysis and design of stream ciphers.* Springer-Verlag, 1986.

[Ver26] G.S. Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *Journal of the American Institute of Electrical Engineers*, vol.55, pages 109–115, 1926.

Running-key

In a stream cipher, the *running-key*, also called the *keystream*, is the sequence which is combined, digit by digit, to the plaintext sequence for obtaining the ciphertext sequence. The running key is generated by a finite state automaton called the *running-key generator* or the *keystream generator* (see stream cipher).

Anne Canteaut.

Linear feedback shift register

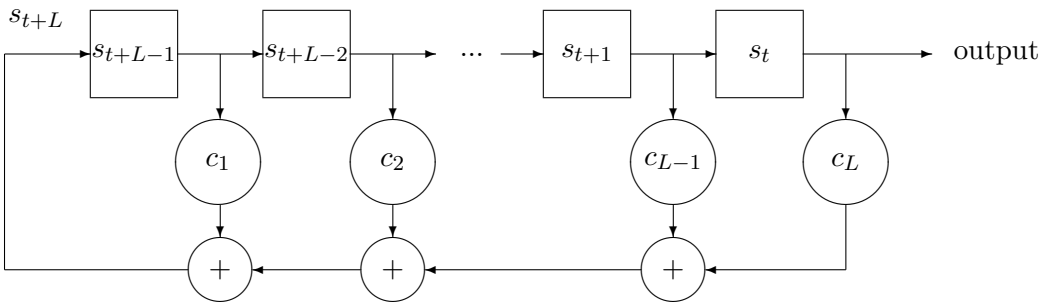
Linear Feedback Shift Registers (LFSRs) are the basic components of many running-key generators for stream cipher applications, because they are appropriate to hardware implementation and they produce sequences with good statistical properties. LFSR refers to a feedback shift register with a linear feedback function (see Nonlinear Feedback Shift Register).

An LFSR of length L over $\mathbf{F}_q$ is a finite state automaton which produces a semi-infinite sequence of elements of $\mathbf{F}_q$, $\mathbf{s} = (s_t)_{t\geq 0} = s_0 s_1 \ldots$, satisfying a linear recurrence relation of degree L over $\mathbf{F}_q$

$$s_{t+L} = \sum_{i=1}^{L} c_i s_{t+L-i}, \quad \forall t \geq 0 \ .$$

The L coefficients $c_1, \ldots, c_L$ are elements of $\mathbf{F}_q$. They are called the *feedback coefficients* of the LFSR.

An LFSR of length L over $\mathbf{F}_q$ has the following form:



The register consists of L delay cells, called *stages*, each containing an element of $\mathbf{F}_q$. The contents of the L stages, $s_t, \ldots, s_{t+L-1}$, form the *state* of the LFSR. The L stages are initially loaded with L elements, $s_0, \ldots, s_{L-1}$, which can be arbitrary chosen in $\mathbf{F}_q$; they form the *initial state* of the register.

The shift register is controlled by an external clock. At each time unit, each digit is shifted one stage to the right. The content of the rightmost stage s_t is output. The new content of the leftmost stage is the *feedback bit*, s_{t+L}. It is obtained by a linear combination of the contents of the register stages, where the coefficients of the linear combination are given by the feedback coefficients of the LFSR:

$$s_{t+L} = \sum_{i=1}^{L} c_i s_{t+L-i} \ .$$

Therefore, the LFSR implements the linear recurrence relation of degree L:

$$s_{t+L} = \sum_{i=1}^{L} c_i s_{t+L-i}, \quad \forall t \geq 0 \ .$$

Example. Table 1 gives the successive states of the binary LFSR of length 4 with feedback coefficients $c_1 = c_2 = 0$, $c_3 = c_4 = 1$ and with initial state $(s_0, s_1, s_2, s_3) = (1, 0, 1, 1)$. This LFSR is depicted in Figure 1. It corresponds to the linear recurrence relation

$$s_{t+4} = s_{t+1} + s_t \bmod 2 \ .$$

The output sequence $s_0 s_1 \ldots$ generated by this LFSR is $1011100 \ldots$.

Figure 1: Binary LFSR with feedback coefficients $(c_1, c_2, c_3, c_4) = (0, 0, 1, 1)$

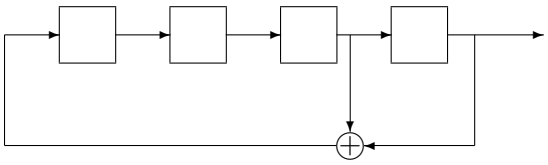


Table 1: Successive states of the LFSR with feedback coefficients $(c_1, c_2, c_3, c_4) = (0, 0, 1, 1)$ and with initial state $(s_0, s_1, s_2, s_3) = (1, 0, 1, 1)$

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s_t	1	0	1	1	1	1	0	0	0	1	0	0	1	1	0	1
s_{t+1}	0	1	1	1	1	0	0	0	1	0	0	1	1	0	1	0
s_{t+2}	1	1	1	1	0	0	0	1	0	0	1	1	0	1	0	1
s_{t+3}	1	1	1	0	0	0	1	0	0	1	1	0	1	0	1	1

Feedback polynomial and characteristic polynomial. The output sequence of an LFSR is uniquely determined by its feedback coefficients and its initial state. The feedback coefficients $c_1, \ldots, c_L$ of an LFSR of length L are usually represented by the LFSR *feedback polynomial* (or *connection polynomial*) defined by

$$P(X) = 1 - \sum_{i=1}^{L} c_i X^i \ .$$

Alternatively, one can use the *characteristic polynomial*, which is the reciprocal polynomial of the feedback polynomial:

$$P^\star(X) = X^L P(1/X) = X^L - \sum_{i=1}^{L} c_i X^{L-i} \ .$$

For instance, the feedback polynomial of the binary LFSR shown in Figure 1 is $P(X) = 1 + X^3 + X^4$ and its characteristic polynomial is $P^\star(X) = 1 + X + X^4$.

An LFSR is said to be *non-singular* if the degree of its feedback polynomial is equal to the LFSR length (*i.e.*, if the feedback coefficient c_L differs from 0). Any sequence generated by a non-singular LFSR of length L is periodic, and its period does not exceed $q^L - 1$. Indeed, the LFSR has at most q^L different states and the all-zero state is always followed by the all-zero state. Moreover, if the LFSR is singular, all generated sequences are *ultimately periodic*, *i.e.*, the sequences obtained by ignoring a certain number of elements at the beginning are periodic.

Characterization of LFSR output sequences. A given LFSR of length L over $\mathbf{F}_q$ can generate q^L different sequences corresponding to the q^L different initial states and these sequences form a vector space over $\mathbf{F}_q$. The set of all sequences generated by an LFSR with feedback polynomial P is characterized by the following property: a sequence $(s_t)_{t\geq 0}$ is

generated by a LFSR of length L over $\mathbf{F}_q$ with feedback polynomial P if and only if there exists a polynomial $Q \in \mathbf{F}_q[X]$ with $\deg(Q) < L$ such that the generating function of $(s_t)_{t\geq 0}$ satisfies

$$\sum_{t\geq 0} s_t X^t = \frac{Q(X)}{P(X)} \ .$$

Moreover, the polynomial Q is completely determined by the coefficients of P and by the initial state of the LFSR:

$$Q(X) = -\sum_{i=0}^{L-1} X^i \left(\sum_{j=0}^{i} c_{i-j} s_j \right) \ ,$$

where $P(X) = \sum_{i=0}^{L} c_i X^i$. This result, which is called the fundamental identity of formal power series of linear recurring sequences, means that there is a one-to-one correspondence between the sequences generated by an LFSR of lenth L with feedback polynomial P and the fractions $Q(X)/P(X)$ with $\deg(Q) < L$. It has two major consequences. On the first hand, any sequence generated by an LFSR with feedback polynomial P is also generated by any LFSR whose feedback polynomial is a multiple of P. This property is used in some attacks on keystream generators based on LFSRs (see Fast Correlation attack). On the other hand, a sequence generated by an LFSR with feedback polynomial P is also generated by a shorter LFSR with feedback polynomial P' if the corresponding fraction $Q(X)/P(X)$ is such that $\gcd(P, Q) \neq 1$. Thus, amongst all sequences generated by the LFSR with feedback polynomial P, there is one which can be generated by a shorter LFSR if and only if P is not irreducible over $\mathbf{F}_q$.

Moreover, for any linear recurring sequence $(s_t)_{t\geq 0}$, there exists a unique polynomial P_0 with constant term equal to 1, such that the generating function of $(s_t)_{t\geq 0}$ is given by $Q_0(X)/P_0(X)$, where P_0 and Q_0 are relatively prime. Then, the shortest LFSR which generates $(s_t)_{t\geq 0}$ has length $L = \max(\deg(P_0), \deg(Q_0) + 1)$, and its feedback polynomial is equal to P_0. The reciprocal polynomial of P_0, $X^L P_0(1/X)$, is the characteristic polynomial of the shortest LFSR which generates $(s_t)_{t\geq 0}$; it is called the minimal polynomial of the sequence. It determines the linear recurrence relation of least degree satisfied by the sequence. The degree of the minimal polynomial of a linear recurring sequence is the linear complexity of the sequence. It corresponds to the length of the shortest LFSR which generates it. The minimal polynomial of a sequence $\mathbf{s} = (s_t)_{t\geq 0}$ of linear complexity $\Lambda(\mathbf{s})$ can be determined from the knowledge of at least $2\Lambda(\mathbf{s})$ consecutive bits of $\mathbf{s}$ by the Berlekamp-Massey algorithm.

Example:
The binary LFSR of length 10 depicted in Figure 2 has feedback polynomial

$$P(X) = 1 + X + X^3 + X^4 + X^7 + X^{10} \ ,$$

and its initial state $s_0 \dots s_9$ is 1001001001.

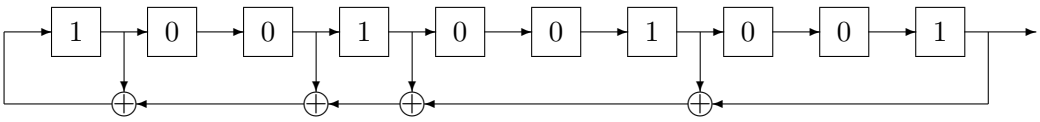
The generating function of the sequence produced by this LFSR is given by

$$\sum_{t\geq 0} s_t X^t = \frac{Q(X)}{P(X)}$$

where Q is deduced from the coefficients of P and from the initial state:

$$Q(X) = 1 + X + X^7 \ .$$

Figure 2: Example of a LFSR of length 10

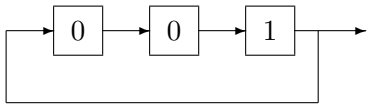


Therefore, we have

$$\sum_{t\geq 0} s_t X^t = \frac{1+X+X^7}{1+X+X^3+X^4+X^7+X^{10}} = \frac{1}{1+X^3}\ ,$$

since $1+X+X^3+X^4+X^7+X^{10} = (1+X+X^7)(1+X^3)$ in $\mathbf{F}_2[X]$. This implies that $(s_t)_{t\geq 0}$ is also generated by the LFSR with feedback polynomial $P_0(X) = 1+X^3$ depicted in Figure 3. The minimal polynomial of the sequence is then $1+X^3$ and its linear complexity is equal to 3.

Figure 3: LFSR of length 3 which generates the same sequence as the LFSR of Figure 2



Period of an LFSR sequence. The minimal polynomial of a linear recurring sequence plays a major role since it completely determines the linear complexity and the least period of the sequence. Actually, the least period of a linear recurring sequence is equal to the period of its minimal polynomial. The *period* (also called the *order*) of a polynomial P in $\mathbf{F}_q[X]$, where $P(0) \neq 0$, is the least positive integer e for which $P(X)$ divides $X^e - 1$. Then, $\mathbf{s}$ has maximal period $q^{\Lambda(\mathbf{s})} - 1$ if and only if its minimal polynomial is a primitive polynomial (*i.e.*, if the period of its minimal polynomial is maximal). For instance, the sequence generated by the LFSR shown in Figure 3 has period 3 because its minimal polynomial $1+X^3$ has period 3. This sequence is $100100100\ldots$. On the other hand, any non-zero sequence generated by the LFSR of length 4 depicted in Figure 1 has period $2^4 - 1 = 15$. Actually, the minimal polynomial of any such sequence corresponds to its characteristic polynomial $P^\star(X) = 1+X+X^4$, because $P^\star$ is irreducible. Moreover, $P^\star$ is a primitive polynomial. Any sequence $\mathbf{s} = (s_t)_{t\geq 0}$ generated by an LFSR of length L which has a primitive feedback polynomial has the highest possible linear complexity $\Lambda(\mathbf{s}) = L$ and the highest possible period $q^L - 1$. Such sequences are called maximal-length linear sequences (m-sequences). Because of the previous optimal properties, the linear recurring sequences used in cryptography are always chosen to be m-sequences. Moreover, they possess good statistical properties (see maximal-length linear sequences for further details). In other terms, the feedback polynomial of a LFSR should always be chosen to be a primitive polynomial.

Keystream generators based on LFSRs. However, it is clear that an LFSR should never be used by itself as a keystream generator. If the feedback coefficients of the LFSR are public, the entire keystream can obviously be recovered from the knowledge of any Λ consecutive bits

of the keystream, where Λ is the linear complexity of the running-key (which does not exceed the LFSR length). If the feedback coefficients are kept secret, the entire keystream can be recovered from any 2Λ consecutive bits of the keystream by the Berlekamp-Massey algorithm. Therefore, a commonly used technique to produce a pseudo-random sequence which can be used as a running-key is to combine several LFSRs in different ways in order to generate a linear recurring sequence which has a high linear complexity (see e.g. combination generator, filter generator...).

Anne Canteaut.

References

[**Gol82**] S.W. Golomb. *Shift register sequences.* Aegean Park Press, revised edition, 1982.

[**LN83**] R. Lidl and H. Niederreiter. *Finite fields.* Cambridge University Press, 1983.

[**Rue86**] R.A. Rueppel. *Analysis and design of stream ciphers.* Springer-Verlag, 1986.

Minimal polynomial

The *minimal polynomial* of a linear recurring sequence $\mathbf{s} = (s_t)_{t\geq 0}$ of elements of $\mathbf{F}_q$ is the polynomial P in $\mathbf{F}_q[X]$ of lowest degree such that $(s_t)_{t\geq 0}$ is generated by the linear feedback shift register (LFSR) with characteristic polynomial P. In other terms, $P = \sum_{i=0}^{L-1} p_i X^i + X^L$ is the characteristic polynomial of the linear recurrence relation of least degree satisfied by the sequence:

$$s_{t+L} + \sum_{i=0}^{L-1} p_i s_{t+i} = 0,\ t \geq 0\ .$$

The minimal polynomial of a linear recurring sequence $\mathbf{s}$ is monic and unique; it divides the characteristic polynomial of any LFSR which generates $\mathbf{s}$. The degree of the minimal polynomial of $\mathbf{s}$ is called its linear complexity. The period of the minimal polynomial of $\mathbf{s}$ is equal to the least period of $\mathbf{s}$. (see linear feedback shift register for further details).

The minimal polynomial of a linear recurring sequence with linear complexity Λ can be recovered from any 2Λ consecutive terms of the sequence by the Berlekamp-Massey algorithm.

Anne Canteaut.

Linear complexity

The *linear complexity* of a semi-infinite sequence $\mathbf{s} = (s_t)_{t\geq 0}$ of elements of $\mathbf{F}_q$, $\Lambda(\mathbf{s})$, is the smallest integer Λ such that $\mathbf{s}$ can be generated by a linear feedback shift register (LFSR) of length Λ over $\mathbf{F}_q$, and is ∞ if no such LFSR exists. By way of convention, the linear complexity of the all-zero sequence is equal to 0. The linear complexity of a linear recurring sequence corresponds to the degree of its minimal polynomial.

The linear complexity $\Lambda(\mathbf{s^n})$ of a finite sequence $\mathbf{s^n} = s_0 s_1 \dots s_{n-1}$ of n elements of $\mathbf{F}_q$ is the length of the shortest LFSR which produces $\mathbf{s^n}$ as its first n output terms for some initial state. The linear complexity of any finite sequence can be determined by the Berlekamp-Massey algorithm. An important result due to Massey [Mas69] is that, for any finite sequence $\mathbf{s^n}$ of length n, the LFSR of length $\Lambda(\mathbf{s^n})$ which generates $\mathbf{s^n}$ is unique if and only if $n \geq 2\Lambda(\mathbf{s^n})$.

The linear complexity of an infinite linear recurring sequence $\mathbf{s}$ and the linear complexity of the finite sequence $\mathbf{s^n}$ composed of the first n digits of $\mathbf{s}$ are related by the following property: if $\mathbf{s}$ is an infinite linear recurring sequence with linear complexity Λ, then the finite sequence $\mathbf{s^n}$ has linear complexity Λ for any $n \geq 2\Lambda$. Moreover, the unique LFSR of length Λ that generates $\mathbf{s}$ is the unique LFSR of length Λ that generates $\mathbf{s^n}$ for every $n \geq 2\Lambda$.

For a sequence $\mathbf{s} = s_0 s_1 \dots$, the sequence of the linear complexities $(\Lambda(\mathbf{s^n}))_{n\geq 1}$ of all subsequences $\mathbf{s^n} = s_0 \dots s_{n-1}$ composed of the first n terms of $\mathbf{s}$ is called the *linear complexity profile* of $\mathbf{s}$.

The expected linear complexity of a binary sequence $\mathbf{s^n} = s_0 \dots s_{n-1}$ of n independent and uniformly distributed binary random variables is

$$E[\Lambda(\mathbf{s}^n)] = \frac{n}{2} + \frac{4 + \varepsilon(n)}{18} + 2^{-n}\left(\frac{n}{3} + \frac{2}{9}\right) \ ,$$

where $\varepsilon(n) = n \bmod 2$.

If $\mathbf{s}$ is an infinite binary sequence of period 2^n which is obtained by repeating a sequence $s_0 \dots s_{2^n-1}$ of 2^n independent and uniformly distributed binary random variables, its expected linear complexity is

$$E[\Lambda(\mathbf{s})] = 2^n - 1 + 2^{-2^n} \ .$$

Further results on the linear complexity and on the linear complexity profile of random sequences can be found in [Rue86].

Anne Canteaut.

References

[Mas69] J.L. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, vol. 15, pp. 122–127, 1969.

[Rue86] R.A. Rueppel. *Analysis and design of stream ciphers.* Springer-Verlag, 1986.

Berlekamp-Massey algorithm

The *Berlekamp-Massey algorithm* is an algorithm for determining the linear complexity of a finite sequence and the feedback polynomial of a linear feedback shift register (LFSR) of minimal length which generates this sequence. This algorithm is due to Massey [Mas69], who showed that the iterative algorithm proposed in 1967 by Berlekamp [Ber67] for decoding BCH codes can be used for finding the shortest LFSR that generates a given sequence.

For a given sequence $\mathbf{s}^n$ of length n, the Berlekamp-Massey performs n iterations. The t-th iteration determines an LFSR of minimal length which generates the first t digits of $\mathbf{s}^n$. The algorithm can be described as follows.

Input. $\mathbf{s^n} = s_0 s_1 \dots s_{n-1}$, a sequence of n elements of $\mathbf{F}_q$.

Output. Λ, the linear complexity of $\mathbf{s^n}$ and P, the feedback polynomial of an LFSR of length Λ which generates $\mathbf{s^n}$.

Initialization.
$P(X) \leftarrow 1$, $P'(X) \leftarrow 1$, $\Lambda \leftarrow 0$, $m \leftarrow -1$, $d' \leftarrow 1$.

For t **from** 0 **to** $n-1$ **do**

$d \leftarrow s_t + \sum_{i=1}^{\Lambda} p_i s_{t-i}$.
If $d \neq 0$ then
$T(X) \leftarrow P(X)$.
$P(X) \leftarrow P(X) - d(d')^{-1} P'(X) X^{t-m}$.
if $2\Lambda \leq t$ then
$\Lambda \leftarrow t + 1 - \Lambda$.
$m \leftarrow t$.
$P'(X) \leftarrow T(X)$.
$d' \leftarrow d$.

Return Λ and P.

In the particular case of a binary sequence, the quantity d' does not need to be stored since it is always equal to 1. Moreover, the feedback polynomial is simply updated by

$$P(X) \leftarrow P(X) + P'(X) X^{t-m} \ .$$

The number of operations performed for computing the linear complexity of a sequence of length n is $\mathcal{O}(n^2)$.

It is worth noticing that the LFSR of minimal length that generates a sequence $\mathbf{s^n}$ of length n is unique if and only if $n \geq 2\Lambda(\mathbf{s^n})$, where $\Lambda(\mathbf{s^n})$ is the linear complexity of $\mathbf{s^n}$.

Example:
The following table describes the successive steps of the Berlekamp-Massey algorithm applied to the binary sequence of length 7, $s_0 \dots s_6 = 0111010$. The values

t	s_t	d	Λ	$P(X)$	m	$P'(X)$
			0	1	-1	1
0	0	0	0	1	-1	1
1	1	1	2	$1+X^2$	1	1
2	1	1	2	$1+X+X^2$	1	1
3	1	1	2	$1+X$	1	1
4	1	0	2	$1+X$	1	1
5	0	1	4	$1+X+X^4$	5	$1+X$
6	0	0	4	$1+X+X^4$	5	$1+X$

of Λ and P obtained at the end of step t correspond to the linear complexity of the sequence $s_0 \dots s_t$ and to the feedback polynomial of an LFSR of minimal length that generates it.

The linear complexity $\Lambda(\mathbf{s})$ of a linear recurring sequence $\mathbf{s} = (s_t)_{t\geq 0}$ is equal to the linear complexity of the finite sequence composed of the first n terms of $\mathbf{s}$ for any $n \geq \Lambda(\mathbf{s})$. Thus, the Berlekamp-Massey algorithm determines the shortest LFSR that generates an infinite linear recurring sequence $\mathbf{s}$ from the knowledge of any $2\Lambda(\mathbf{s})$ consecutive digits of $\mathbf{s}$.

It can be proved that the Berlekamp-Massey algorithm and the Euclidean algorithm are essentially the same.

Anne Canteaut.

References

[**Ber67**] E.R. Berlekamp. *Algebraic coding theory.* McGraw-Hill, 1967.

[**Mas69**] J.L. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, vol. 15, pp. 122–127, 1969.

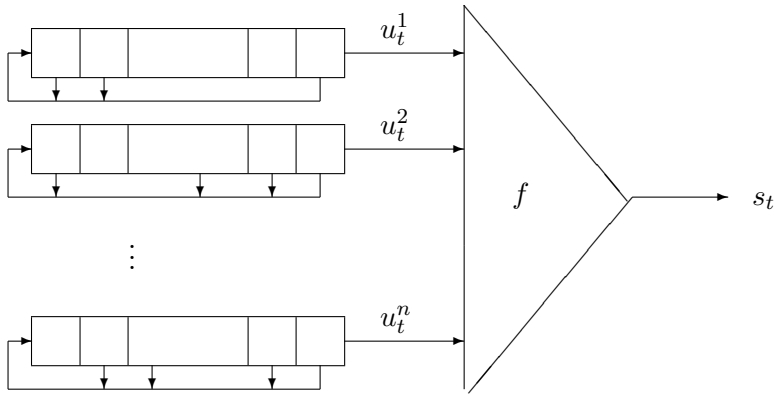
[**Dor87**] J.-L. Dornstetter. On the equivalence between Berlekamp's and Euclid's algorithms. *IEEE Transactions on Information Theory*, vol. 33, pp. 428–431, 1987.

Combination generator

A *combination generator* is a running-key generator for stream cipher applications. It is composed of several linear feedback shift registers (LFSRs) whose outputs are combined by a Boolean function to produce the keystream. Then, the output sequence $(s_t)_{t\geq 0}$ of a combination generator composed of n LFSRs is given by

$$s_t = f(u_t^1, u_t^2, \ldots, u_t^n), \quad \forall t \geq 0 \ ,$$

where $(u_t^i)_{t\geq 0}$ denotes the sequence generated by the i-th constituent LFSR and f is a function of n variables. In the case of a combination generator composed of n LFSR over $\mathbf{F}_q$, the combining function is a function from $\mathbf{F}_q^n$ into $\mathbf{F}_q$.



The combining function f should obviously be *balanced*, *i.e.*, its output should be uniformly distributed. The constituent LFSRs should be chosen to have primitive feedback polynomials for ensuring good statistical properties of their output sequences (see Linear Feedback Shift Register for more details).

The characteristics of the constituent LFSRs and the combining function are usually publicly known. The secret parameters are the initial states of the LFSRs, which are derived from the secret key of the cipher by a key-loading algorithm. Therefore, most attacks on combination generators consist in recovering the initial states of all LFSRs from the knowledge of some digits of the sequence produced by the generator (in a known plaintext attack), or of some digits of the ciphertext sequence (in a ciphertext only attack). When the feedback polynomials of the LFSR and the combining function are not known, the reconstruction attack presented in [CF00] enables to recover the complete description of the generator from the knowledge of a large segment of the ciphertext sequence.

Statistical properties of the output sequence. The sequence produced by a combination generator is a linear recurring sequence. Its period and its linear complexity can be derived from those of the sequences generated by the constituent LFSRs and from the *algebraic normal form* of the combining function (see Boolean function). Indeed, if we consider two linear recurring sequences $\mathbf{u}$ and $\mathbf{v}$ over $\mathbf{F}_q$ with linear complexities $\Lambda(\mathbf{u})$ and $\Lambda(\mathbf{v})$, we have the following properties:

- the linear complexity of the sequence $\mathbf{u} + \mathbf{v} = (u_t + v_t)_{t\geq 0}$ satisfies

$$\Lambda(\mathbf{u} + \mathbf{v}) \leq \Lambda(\mathbf{u}) + \Lambda(\mathbf{v}) \ ,$$

with equality if and only if the minimal polynomials of $\mathbf{u}$ and $\mathbf{v}$ are relatively prime. Moreover, in case of equality, the period of $\mathbf{u}+\mathbf{v}$ is the least common multiple of the periods of $\mathbf{u}$ and $\mathbf{v}$.

- the linear complexity of the sequence $\mathbf{uv} = (u_t v_t)_{t\geq 0}$ satisfies

$$\Lambda(\mathbf{uv}) \leq \Lambda(\mathbf{u})\Lambda(\mathbf{v}) \ ,$$

where equality holds if the minimal polynomials of $\mathbf{u}$ and $\mathbf{v}$ are primitive and if $\Lambda(\mathbf{u})$ and $\Lambda(\mathbf{v})$ are distinct and greater than 2. Other general sufficient conditions for $\Lambda(\mathbf{uv}) = \Lambda(\mathbf{u})\Lambda(\mathbf{v})$ can be found in [Her85], [RS87], [GN95].

Thus, the keystream sequence produced by a combination generator composed of n binary LFSRs with primitive feedback polynomials which are combined by a Boolean function f satisfies the following property proven in [RS87]. If all LFSR lengths $L_1, \ldots, L_n$ are distinct and greater than 2 (and if all LFSR initializations differ from the all-zero state), the linear complexity of the output sequence $\mathbf{s}$ is equal to

$$f(L_1, L_2, \ldots, L_n)$$

where the algebraic normal form of f is evaluated over integers. For instance, if four LFSRs of lengths $L_1, \ldots, L_4$ satisfying the previous conditions are combined by the Boolean function $x_1x_2 + x_2x_3 + x_4$, the linear complexity of the resulting sequence is $L_1L_2 + L_2L_3 + L_4$. Similar results concerning the combination of LFSRs over $\mathbf{F}_q$ can be found in [RS87] and [Bry95]. A high linear complexity is desirable property for a keystream sequence since it ensures that Berlekamp-Massey algorithm becomes computationally infeasible. Thus, the combining function f should have a high *algebraic degree* (the algebraic degree of a Boolean function is the highest number of terms occurring in a monomial of its algebraic normal form).

Known attacks and related design criteria. Combination generators are vulnerable to the correlation attack and its variants called fast correlation attacks. In order to make these attacks infeasible, the LFSR feedback polynomials should not be sparse. The combining function should have a high *correlation-immunity order*, also called *resiliency order* when the involved function is balanced (see correlation-immune Boolean function). But, there exists a tradeoff between the correlation-immunity order and the algebraic degree of a Boolean function. Most notably, the correlation-immunity of a balanced Boolean function of n variables cannot exceed $n - 1 - \deg(f)$, when the algebraic degree of f, $\deg(f)$, is greater than 1. Moreover, the complexity of correlation attacks and of fast correlation attacks also increases with the nonlinearity of the combining function (see correlation attack). The tradeoffs between high algebraic degree, high correlation-immunity order and high nonlinearity can be circumvented by replacing the combining function by a finite state automaton with memory. Examples of such combination generators with memory are the summation generator and the stream cipher E_0 used in Bluetooth.

Anne Canteaut.

References

[**Bry85**] L. Brynielsson. On the linear complexity of combined shift register sequences. *In: Advances in Cryptology - EUROCRYPT '85*, Lecture Notes in Computer Science, number 219, pages 156–160. Springer-Verlag, 1986.

[**CF00**] A. Canteaut and E. Filiol. Ciphertext only reconstruction of stream ciphers based on combination generators. *In: Fast Software Encryption 2000*, Lecture Notes in Computer Science, number 1978, pages 165–180. Springer-Verlag, 2001.

[**Her85**] T. Herlestam. On functions of linear shift register sequences. *In: Advances in Cryptology - EUROCRYPT '85*, Lecture Notes in Computer Science, number 219, pages 119–129. Springer-Verlag, 1986.

[**GN95**] R. Göttfert and H. Niederreiter. On the minimal polynomial of the product of linear recurring sequences. *Finite Fields and Their Applications*, 1(2):204–218, 1995.

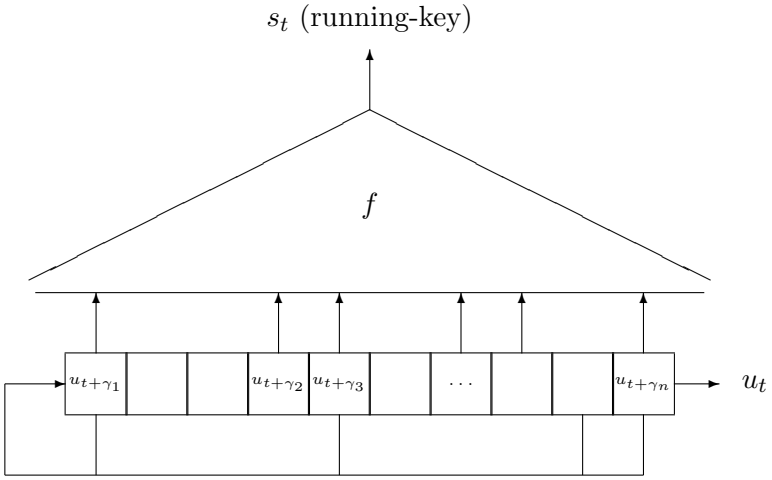
[**RS87**] R.A. Rueppel and O.J. Staffelbach. Products of linear recurring sequences with maximum complexity. *IEEE Transactions on Information Theory*, 33(1):124–131, 1987.

Filter generator

A *filter generator* is a running-key generator for stream cipher applications. It consists of a single linear feedback shift register (LFSR) which is filtered by a nonlinear function. More precisely, the output sequence of a filter generator corresponds to the output of a nonlinear function whose inputs are taken from some stages of the LFSR. If $(u_t)_{t\geq 0}$ denotes the sequence generated by the LFSR, the output sequence $(s_t)_{t\geq 0}$ of the filter generator is given by

$$s_t = f(u_{t+\gamma_1}, u_{t+\gamma_2}, \ldots, u_{t+\gamma_n}), \quad \forall t \geq 0$$

where f is a function of n variables, n is less than or equal to the LFSR length and $(\gamma_i)_{1\leq i\leq n}$ is a decreasing sequence of nonnegative integers called the *tapping sequence*.



In order to obtain a keystream sequence having good statistical properties, the filtering function f should be *balanced* (*i.e.*, its output should be uniformly distributed), and the feedback polynomial of the LFSR should be chosen to be a primitive polynomial (see linear feedback shift register for more details).

In a filter generator, the LFSR feedback polynomial, the filtering function and the tapping sequence are usually publicly known. The secret parameter is the initial state of the LFSR which is derived from the secret key of the cipher by a key-loading algorithm. Therefore, most attacks on filter generators consist in recovering the LFSR initial state from the knowledge of some digits of the sequence produced by the generator (in a known plaintext attack), or of some digits of the ciphertext sequence (in a ciphertext only attack). The attack presented in [Sie85] enables to construct an equivalent keystream generator from the knowledge of a large segment of the ciphertext sequence when the LFSR feedback polynomial is the only known parameter (*i.e.*, when the filtering function, the tapping sequence and the initial state are kept secret).

Any filter generator is equivalent to a particular combination generator, in the sense that both generators produce the same output sequence. An equivalent combination generator consists of n copies of the LFSR used in the filter generator with shifted initial states; the combining function corresponds to the filtering function.

Statistical properties of the output sequence. The output sequence $\mathbf{s}$ of a filter generator is a linear recurring sequence. Its linear complexity, $\Lambda(\mathbf{s})$, is related to the LFSR length and to the *algebraic degree* of the filtering function f (the algebraic degree of a Boolean function is the highest number of terms occurring in a monomial of its algebraic normal form). For a binary LFSR with a primitive feedback polynomial, we have

$$\Lambda(\mathbf{s}) \leq \sum_{i=0}^{d} \binom{L}{i}$$

where L denotes the LFSR length and d denotes the algebraic degree of f [Key76, Mas01]. The period of $\mathbf{s}$ divides $2^L - 1$. Moreover, if L is a large prime, $\Lambda(\mathbf{s})$ is at least $\binom{L}{d}$ for most filtering functions with algebraic degree d (see [Rue86]).

Thus, to achieve a high linear complexity, the LFSR length L and the algebraic degree of the filtering function should be large enough. More precisely, the keystream length available to an attacker should always be much smaller than $\binom{L}{\deg(f)}$.

Known attacks and related design criteria Filter generators are vulnerable to fast correlation attacks because their output sequence is correlated to some linear function of the stages of the constituent LFSR (see fast correlation attack). Efficient fast correlation attacks on filter generators are described in [JJ02] and [CF02]. In order to make the fast correlation attacks computationally infeasible, the filtering function should have a high nonlinearity. Moreover, it should have many nonzero Walsh coefficients. Another design criterion is that the LFSR feedback polynomial should not be sparse.

Another attack on any filter generator is the generalized inversion attack. It highly depends on the *memory size* of the generator, which corresponds to the largest spacing between two taps, *i.e.*, $M = \gamma_1 - \gamma_n$. To make this attack infeasible, the tapping sequence should be such that the memory size is large and preferably close to its maximum possible value, $L - 1$, where L is the LFSR length. Moreover, when the greatest common divisor of all spacing between the taps, $\gcd(\gamma_i - \gamma_{i+1})$, is large, the effective memory size can be reduced by a decimation technique (see inversion attack). Then, the greatest common divisor of all $(\gamma_i - \gamma_{i+1})$ should be equal to 1.

The choice of the tapping sequence also conditions the resistance to the so-called *conditional correlation attacks* [And94, Gol96, LCPP96]. The basic idea of these particular correlation attack is that some information on the LFSR sequence may leak when some patterns appear in the keystream sequence. Actually, the keystream bits s_t and $s_{t+\tau}$, with $\tau \geq 1$, respectively depend on the LFSR-output bits $u_{t+\gamma_1}, \ldots, u_{t+\gamma_n}$ and $u_{t+\gamma_1+\tau}, \ldots, u_{t+\gamma_n+\tau}$. Therefore, the pair $(s_t, s_{t+\tau})$ only depends on $M - I(\tau)$ bits of the LFSR sequence, where M is the memory size and $I(\tau)$ is the size of the intersection between $\{\gamma_i, 1 \leq i \leq n\}$ and $\{\gamma_i + \tau, 1 \leq i \leq n\}$, *i.e.*, the number of pairs (i, j) with $i < j$ such that $\gamma_i - \gamma_j = \tau$. It is then clear that a given observation of $(s_t, s_{t+\tau})$ may provide some information on the $(M - I(\tau))$ involved bits of the LFSR sequence when $I(\tau)$ is large. Thus, $I(\tau)$ should be as small as possible for all values of $\tau \geq 1$. It can be proved that the lowest possible value of $\max_{\tau \geq 1} I(\tau)$ is 1 and it is achieved when the tapping sequence is a *full positive difference set*, *i.e.*, when all differences $\gamma_i - \gamma_j$, $i < j$ are distinct. Such a tapping sequence of n integers only exists if the LFSR length exceeds $n(n-1)/2$ (see [Gol96] for details).

Advanced algebraic techniques, like Gröbner bases, also provide powerful known plaintext attacks on filter generator, called *algebraic attacks* [CM03]. Any keystream bit can be expressed as a function of the L initial bits of the LFSR. Thus, the knowledge of any N keystream bits lead to an algebraic system of N equations of L variables. The degree of these equations correspond to the algebraic degree of the filtering function. But, efficient Gröbner bases techniques enable to substantially lower the degree of the equations by multiplying them by well-chosen multivariate polynomials. Then, it may be possible to recover the LFSR initial state by solving the algebraic system even if the filtering function has a high degree.

Anne Canteaut.

References

[**And94**] R.J. Anderson. Searching for the optimum correlation attack. In *Fast Software Encryption 1994*, Lecture Notes in Computer Science number 1008, pages 137–143. Springer-Verlag, 1995.

[**CM03**] N.T. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology - EUROCRYPT 2003*, Lecture Notes in Computer Science number 2656, pages 345–359. Springer-Verlag, 2003.

[**Gol96**] J.Dj. Golić. On the security of nonlinear filter generators. In *Fast Software Encryption 1996*, Lecture Notes in Computer Science number 1039, pages 173–188. Springer-Verlag, 1996.

[**Key76**] E.L. Key. An analysis of the structure and complexity of nonlinear binary sequence generators. *IEEE Transactions on Information Theory*, vol. 22, pp. 732–736, 1976.

[**LCPP96**] S. Lee, S. Chee, S. Park, and S. Park. Conditional correlation attack on nonlinear filter generators. In *Advances in Cryptology - ASIACRYPT'96*, Lecture Notes in Computer Science number 1163, pages 360–367. Springer-Verlag, 1996.

[**Mas01**] J.L. Massey. The ubiquity of Reed-Muller Codes. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes - AAECC-14*, Lecture Notes in Computer Science number 2227, pages 1–12. Springer-Verlag, 2001.

[**Rue86**] R.A. Rueppel. *Analysis and design of stream ciphers.* Springer-Verlag, 1986.

Correlation attack

The *correlation attack* was proposed by Siegenthaler in 1985. It applies to any running-key generator composed of several linear feedback shift registers (LFSRs). The correlation attack is a divide-and-conquer technique: it aims at recovering the initial state of each constituent LFSRs separately from the knowledge of some keystream bits (in a known plaintext attack). A similar ciphertext only attack can also be mounted when there exists redundancy in the plaintext (see [Sie85]).

The original correlation attack presented in [Sie85] applies to some combination generators composed of n LFSRs of lengths $L_1, \ldots, L_n$. It enables to recover the complete initialization of the generator with only $\sum_{i=1}^{n} \left(2^{L_i} - 1\right)$ trials instead of the $\prod_{i=1}^{n} \left(2^{L_i} - 1\right)$ tests required by an exhaustive search. Some efficient variants of the original correlation attack can also be applied to other keystream generators based on LFSRs, like filter generators (see fast correlation attack for details).

Original correlation attack on combination generators. The correlation attack exploits the existence of a statistical dependence between the keystream and the output of a single constituent LFSR. In a binary combination generator, such a dependence exists if and only if the output of the combining function f is correlated to one of its inputs, *i.e.*, if

$$p_i = Pr[f(x_1, \ldots, x_n) \neq x_i] \neq \frac{1}{2}$$

for some i, $1 \leq i \leq n$. It equivalently means that the keystream sequence $\mathbf{s} = (s_t)_{t \geq 0}$ is correlated to the sequence $\mathbf{u} = (u_t)_{t \geq 0}$ generated by the i-th constituent LFSR. Namely, the correlation between both sequences calculated on N bits

$$\sum_{t=0}^{N-1} (-1)^{s_t + u_t \bmod 2}$$

(where the sum is defined over real numbers) is a random variable which is binomially distributed with mean value $N(1-2p_i)$ and with variance $4Np_i(1-p_i)$ (when N is large enough). It can be compared to the correlation between the keystream $\mathbf{s}$ and a sequence $\mathbf{r} = (r_t)_{t \geq 0}$ independent of $\mathbf{s}$ (*i.e.*, such that $Pr[s_t \neq r_t] = 1/2$). For such a sequence $\mathbf{r}$, the correlation between $\mathbf{s}$ and $\mathbf{r}$ is binomially distributed with mean value 0 and with variance N. Thus, an exhaustive search on the initialization of the i-th LFSR can be performed. The value of the correlation enables to distinguish the correct initial state from a wrong one since the sequence generated by a wrong initial state is assumed to be statistically independent of the keystream. Table 2 gives a complete description of the attack.

In practice, an initial state is accepted if the magnitude of the correlation exceeds a certain decision threshold which is deduced from the expected false alarm probability P_f and the non-detection probability P_n (see [Sie85]). The required keystream length N depends on the probability p_i and on the length L_i of the involved LFSR: for $P_n = 1.3 \cdot 10^{-3}$ and $P_f = 2^{-L_i}$, the attack requires

$$N \simeq \left(\frac{\sqrt{\ln(2^{L_i - 1})} + 3\sqrt{2p_i(1-p_i)}}{\sqrt{2}(p_i - 0.5)} \right)^2$$

running-key bits. Clearly, the attack performs $2^{L_i - 1}$ trials in average where L_i is the length of the target LFSR. The correlation attack only applies if the probability p_i differs from $1/2$.

Input. $s_0 s_1 \dots s_{N-1}$, N keystream bits,
$p_i = Pr[f(x_1, \dots, x_n) \neq x_i] \neq 1/2$.

Output. $u_0 \dots u_{L_i-1}$, the initial state of the i-th constituent LFSR.

For each possible initial state $u_0 \dots u_{L_i-1}$

Generate the first N bits of the sequence $\mathbf{u}$ produced by the i-th LFSR from the chosen initial state.

Compute the correlation between $s_0 s_1 \dots s_{N-1}$ and $u_0 u_1 \dots u_{N-1}$:

$$\alpha \leftarrow \sum_{i=0}^{N-1} (-1)^{s_t + u_t \bmod 2}$$

If α is close to $N(1-2p_i)$

return $u_0 u_1 \dots u_{L_i-1}$

Table 2: Correlation attack

Correlation attack on other keystream generators. More generally, the correlation attack applies to any keystream generator as soon as the keystream is correlated to the output sequence $\mathbf{u}$ of a finite state machine whose initial state depends on some key bits. These key bits can be determined by recovering the initialization of $\mathbf{u}$ as follows: an exhaustive search on the initialization of $\mathbf{u}$ is performed, and the correct one is detected by computing the correlation between the corresponding sequence $\mathbf{u}$ and the keystream.

Correlation attack on combination generators involving several LFSRs. For combination generators, the correlation attack can be prevented by using a combining function f whose output is not correlated to any of its inputs. Such functions are called 1st-order correlation-immune (or 1-resilient in the case of balanced functions). In this case, the running-key is statistically independent of the output of each constituent LFSR; any correlation attack should then consider several LFSRs simultaneously. More generally, a correlation attack on a set of k constituent LFSRs, namely LFSR i_1, ..., LFSR i_k, exploits the existence of a correlation between the running-key $\mathbf{s}$ and the output $\mathbf{u}$ of a smaller combination generator, which consists of the k involved LFSRs combined by a Boolean function g of k variables (see Fig. 4). Since $Pr[s_t \neq u_t] = Pr[f(x_1, \dots, x_n) \neq g(x_{i_1}, \dots, x_{i_k})] = p_g$, this attack only succeeds when $p_g \neq 1/2$. The smallest number of LFSRs that can be attacked simultaneously is equal to $m+1$ where m is the highest correlation-immunity order of the combining function. Moreover, the Boolean function g of $(m+1)$ variables which provides the best approximation of f is the affine function $\sum_{j=1}^{m+1} x_{i_j} + \varepsilon$ [CT00, Zha00]. Thus, the most efficient correlation attacks that can be mounted relies on the correlation between the keystream $\mathbf{s}$ and the sequence $\mathbf{u}$ obtained by adding the outputs of LFSRs $i_1, i_2, \dots, i_{m+1}$. This correlation corresponds to

$$Pr[s_t \neq u_t] = \frac{1}{2} - \frac{1}{2^{n+1}} |\hat{\mathrm{f}}(t)| \ ,$$

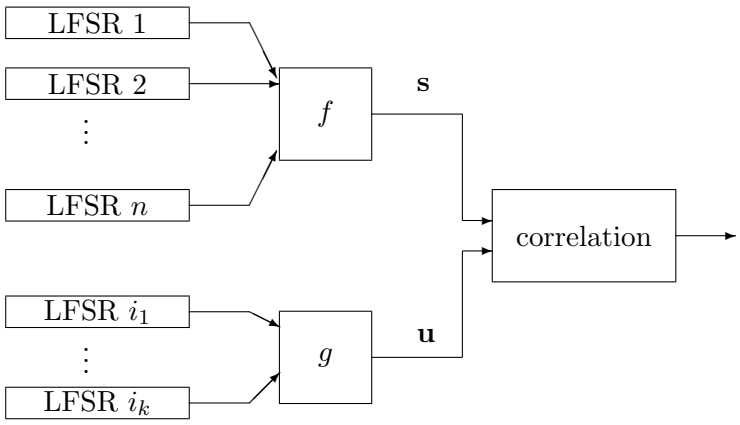


Figure 4: Correlation attack involving several constituent LFSRs of a combination generator

where n is the number of variables of the combining function, t is the n-bit vector whose i-th component equals 1 if and only if $i \in \{i_1, i_2, \ldots, i_{m+1}\}$ and $\hat{\mathrm{f}}$ denotes the Walsh transform of f (see Boolean functions). In order to increase the complexity of the correlation attack, the combining function used in a combination generator should have a high correlation-immunity order and a high nonlinearity (more precisely, its Walsh coefficients $\hat{\mathrm{f}}(t)$ should have a low magnitude for all vectors t with a small Hamming weight). For an m-resilient combining function, the complexity of the correlation attack is $2^{L_{i_1}+L_{i_2}+\ldots+L_{i_{m+1}}}$. It can be significantly reduced by using some improved algorithms, called fast correlation attacks.

Anne Canteaut.

References

[CT00] A. Canteaut and M. Trabbia. Improved fast correlation attacks using parity-check equations of weight 4 and 5. In *Advances in Cryptology - EUROCRYPT 2000*, Lecture Notes in Computer Science number 1807, pages 573–588. Springer-Verlag, 2000.

[Sie84] T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Trans. Inform. Theory*, IT-30(5):776–780, 1984.

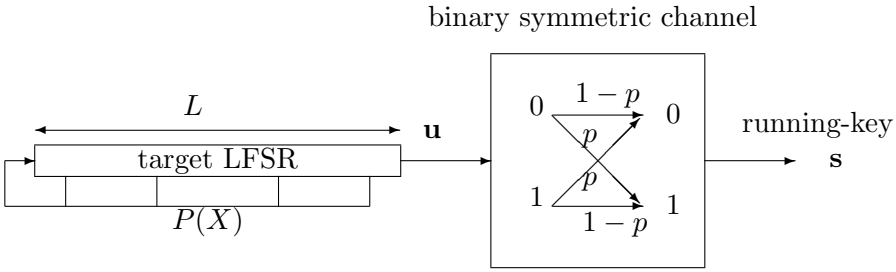
[Sie85] T. Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Trans. Computers*, C-34(1):81–84, 1985.

[Zha00] M. Zhang. Maximum correlation analysis of nonlinear combining functions in stream ciphers. *Journal of Cryptology*, 13(3):301–313, 2000.

Fast correlation attack

Fast correlation attacks were first proposed by Meier and Staffelbach in 1988. They apply to running-key generators based on linear feedback shift registers (LFSRs), exactly in the same context as the correlation attack, but they are significantly faster. They rely on the same principle as the correlation attack: they exploit the existence of a correlation between the keystream and the output of a single LFSR, called the *target LFSR*, whose initial state depends on some bits of the secret key. In the original correlation attack, the initial state of the target LFSR is recovered by an exhaustive search. Fast correlation attacks avoid examining all possible initializations of the target LFSR by using some efficient error-correcting techniques. But, they require the knowledge of a longer segment of the keystream (in the context of a known-plaintext attack). As for the correlation attack, similar algorithms can be used for mounting a ciphertext only attack when there exists redundancy in the plaintext.

Fast correlation attacks as a decoding problem. The key idea of fast correlation attacks consists in viewing the correlation attack as a decoding problem. If there exists a correlation between the keystream $\mathbf{s}$ and the output $\mathbf{u}$ of the target LFSR, then the running-key subsequence $(s_t)_{t<N}$ can be seen as the result of the transmission of $(u_t)_{t<N}$ through the binary symmetric channel with error probability $p = Pr[s_t \neq u_t] < 1/2$ (if $Pr[s_t \neq u_t] > 1/2$, the bitwise complement of $\mathbf{s}$ is considered). Moreover, all bits of the LFSR sequence $\mathbf{u}$ depend linearly on the LFSR initial state, $u_0 \dots u_{L-1}$. Therefore, $(u_t)_{t<N}$ is a codeword of a linear code of length N and dimension L defined by the LFSR feedback polynomial. Thus, recovering the LFSR initial state consists in decoding the running-key subsequence relatively to the LFSR code.



With this formulation, the original correlation attack proposed by Siegenthaler consists in applying a maximum-likelihood decoding algorithm to the linear code defined by the LFSR feedback polynomial. It has complexity 2^L, where L is the length of the target LFSR. The complexity can be reduced by using faster decoding algorithms. But, they usually require a larger number of running-key bits.

Decoding techniques for fast correlation attacks. Several algorithms can be used for decoding the LFSR code, based on the following ideas:

- find many sparse linear recurrence relations satisfied by the LFSR sequence (these relations correspond to sparse multiples of the feedback polynomial), and use them in an iterative decoding procedure dedicated to low-density parity-check codes [MS89, CT00, MFI00]. The complexity of this attack may significantly decrease when the feedback

polynomial of the target LFSR is sparse. Thus, the use of sparse LFSR feedback polynomials should be avoided in LFSR-based running-key generators;

- construct a convolutional code [JJ99a] (or a turbo code [JJ99b]) from the LFSR code, and use an appropriate decoding algorithm for this new code (Viterbi algorithm or turbo-decoding);
- construct a new linear block code with a lower dimension from the LFSR code and apply to this smaller linear code a maximum-likelihood decoding algorithm [CJS00], or a polynomial reconstruction technique [JJ00].

A survey on all these techniques and their computational complexities can be found in [Jön02]. In practice, the most efficient fast correlation attacks enable to recover the initial state of a target LFSR of length 60 for an error-probability $p = 0.4$ in a few hours on a PC from the knowledge of 10^6 running-key bits.

Fast correlation attacks on combination generators. In the particular case of a combination generator, the target sequence $\mathbf{u}$ is the sequence obtained by adding the outputs of $(m+1)$ constituent LFSRs, where m is the correlation-immunity order of the combining function (see correlation attack). Thus, this sequence $\mathbf{u}$ corresponds to the output of a unique LFSR whose feedback polynomial is the greatest common divisor of the feedback polynomials of the $(m+1)$ involved LFSRs. Since the feedback polynomials are usually chosen to be primitive, the length of the target LFSR is the sum of the lengths of the $(m+1)$ LFSRs. The keystream corresponds to the received word as output of the binary symmetric channel with error-probability

$$p = Pr[s_t \neq u_t] = \frac{1}{2} - \frac{1}{2^{n+1}}|\hat{\mathrm{f}}(t)| \ ,$$

where n is the number of variables of the combining function, t is the n-bit vector whose i-th component equals 1 if and only if $i \in \{i_1, i_2, \ldots, i_{m+1}\}$ and $\hat{\mathrm{f}}$ denotes the Walsh transform of f (see correlation attack and Boolean functions).

Fast correlation attacks on filter generators. In the case of a filter generator, the target LFSR has the same feedback polynomial as the constituent LFSR, but a different initial state. Actually, if the keystream is given by

$$s_t = f(v_{t+\gamma_1}, v_{t+\gamma_2}, \ldots, v_{t+\gamma_n}) \ ,$$

where $\mathbf{v}$ is the sequence generated by the constituent LFSR. The optimal target sequence $\mathbf{u}$ then corresponds to

$$u_t = \sum_{i=1}^{n} \alpha_i v_{t+\gamma_i}$$

where $\alpha = (\alpha_1, \ldots, \alpha_n)$ is the vector which maximizes the magnitude of the Walsh transform of the filtering function. Thus, the keystream corresponds to the received word as output of the binary symmetric channel with error-probability

$$p = Pr[s_t \neq u_t] = \frac{\mathcal{NL}(f)}{2^n} \ ,$$

where $\mathcal{NL}(f)$ is the nonlinearity of the filtering function. The fast correlation attacks on filter generators can be improved by using several target LFSRs together [JJ02, CF02].

Other particular fast correlation attacks apply to filter generators, like conditional block-oriented correlation attacks [And94, Gol96, LCPP96] (see filter generator).

Anne Canteaut.

References

[**And94**] R.J. Anderson. Searching for the optimum correlation attack. In *Fast Software Encryption 1994*, Lecture Notes in Computer Science number 1008, pages 137–143. Springer-Verlag, 1995.

[**CF02**] A. Canteaut and E. Filiol. On the Influence of the Filtering Function on the Performance of Fast Correlation Attacks on Filter Generators. In *Symposium on information theory in the Benelux*, May 2002.

[**CJS00**] V. Chepyshov, T. Johansson, and B. Smeets. A simple algorithm for fast correlation attacks on stream ciphers. In *Fast Software Encryption 2000*, Lecture Notes in Computer Science number 1978, pages 181–195. Springer-Verlag, 2000.

[**CT00**] A. Canteaut and M. Trabbia. Improved fast correlation attacks using parity-check equations of weight 4 and 5. In *Advances in Cryptology - EUROCRYPT 2000*, Lecture Notes in Computer Science number 1807, pages 573–588. Springer-Verlag, 2000.

[**Gol96**] J.Dj. Golić. On the security of nonlinear filter generators. In *Fast Software Encryption 1996*, Lecture Notes in Computer Science number 1039, pages 173–188. Springer-Verlag, 1996.

[**JJ99a**] T. Johansson and F. Jönsson. Improved fast correlation attack on stream ciphers via convolutional codes. In *Advances in Cryptology - EUROCRYPT'99*, Lecture Notes in Computer Science number 1592, pages 347–362. Springer-Verlag, 1999.

[**JJ99b**] T. Johansson and F. Jönsson. Fast correlation attacks based on turbo code techniques. In *Advances in Cryptology - CRYPTO'99*, Lecture Notes in Computer Science number 1666, pages 181–197. Springer-Verlag, 1999.

[**JJ00**] T. Johansson and F. Jönsson. Fast correlation attacks through reconstruction of linear polynomials. In *Advances in Cryptology - CRYPTO'00*, Lecture Notes in Computer Science number 1880, pages 300–315. Springer-Verlag, 2000.

[**JJ02**] F. Jönsson and T. Johansson. A fast correlation attack on LILI-128. *Information Processing Letters*, 81(3):127–132, 2002.

[**Jn02**] F. Jönsson. *Some results on fast correlation attacks.* PhD thesis, University of Lund, Sweden, 2002.

[**LCPP96**] S. Lee, S. Chee, S. Park, and S. Park. Conditional correlation attack on nonlinear filter generators. In *Advances in Cryptology - ASIACRYPT'96*, Lecture Notes in Computer Science number 1163, pages 360–367. Springer-Verlag, 1996.

[**MFI00**] M.J. Mihaljevic, M.P.C. Fossorier, and H. Imai. A low-complexity and high performance algorithm for the fast correlation attack. In *Fast Software Encryption 2000*, Lecture Notes in Computer Science number 1978, pages 196–212. Springer-Verlag, 2000.

[**MS88**] W. Meier and O. Staffelbach. Fast correlation attacks on stream ciphers. In *Advances in Cryptology - EUROCRYPT'88*, Lecture Notes in Computer Science number 330, pages 301–314. Springer-Verlag, 1988.

[**MS89**] W. Meier and O. Staffelbach. Fast correlation attack on certain stream ciphers. *J. Cryptology*, pages 159–176, 1989.

Inversion attack

The *inversion attack* is a known plaintext attack on some particular filter generators. It was proposed by Golić in 1996 [Gol96]. A generalization to any filter generator, called *generalized inversion attack*, was presented by Golić, Clark and Dawson in 2000. Both inversion attack and generalized inversion attack aim at recovering the initial state of the linear feedback shift register (LFSR) from a segment of the running-key when the LFSR feedback polynomial, the tapping sequence and the filtering function are known.

Original inversion attack. The original inversion attack only applies when the filtering function f is linear in its first input variable (forward attack) or in its last input variable (backward attack), *i.e.*, when

$$f(x_1, x_2, \ldots, x_n) = x_1 + g(x_2, \ldots, x_n)$$

or

$$f(x_1, x_2, \ldots, x_n) = g(x_1, \ldots, x_{n-1}) + x_n$$

where g is a Boolean function of $n-1$ variables. In the first case, the keystream $\mathbf{s}$ is defined by

$$\begin{aligned} s_t &= f(u_{t+\gamma_1}, u_{t+\gamma_2}, \ldots, u_{t+\gamma_n}) \\ &= u_{t+\gamma_1} + g(u_{t+\gamma_2}, \ldots, u_{t+\gamma_n}) \ , \end{aligned}$$

where $(u_t)_{t\geq 0}$ is the sequence generated by the LFSR and $(\gamma_i)_{1\leq i\leq n}$ is a decreasing sequence of non-negative integers. The attack relies on the fact that the bit $u_{t+\gamma_1}$ can be deduced from the $(\gamma_1 - \gamma_n)$ previous terms, $(u_{t+\gamma_1+1}, \ldots, u_{t+\gamma_n})$ if the running-key bit s_t is known. The relevant parameter of the attack is then the *memory size* of the filter generator, defined by $M = \gamma_1 - \gamma_n$. Indeed, the complete initialization of the LFSR can be recovered by an exhaustive search on only M bits as described in Table 3.

The backward attack, which applies when the filter function is linear in its last variable, is similar. The complexity of both forward and backward attacks is $\mathcal{O}(L2^M)$. It follows that the memory size of a filter generator should be large and preferably close to its maximum possible value $L-1$.

Moreover, the complexity of the attack dramatically decreases when the greatest common divisor of all spacings between the taps, $d = \gcd(\gamma_i - \gamma_{i+1})$, is large. Indeed, the inversion attack can be applied to the d-decimation of the LFSR sequence, *i.e.*, to the sequence obtained by sampling the LFSR sequence at intervals of d clock cycles (see [Gol96]). Therefore, the effective memory size of the filter generator corresponds to

$$M' = \frac{\gamma_1 - \gamma_n}{\gcd(\gamma_i - \gamma_{i+1})} \ .$$

The related design criterion is then that the greatest common divisor of all spacings between the taps should be equal to 1.

Generalized inversion attack. A similar attack can be mounted even if the filtering function is not linear in its first or last variable. In the general case, the keystream is given by

$$s_t = f(u_{t+\gamma_1}, u_{t+\gamma_2}, \ldots, u_{t+\gamma_n}) \ .$$

Input. $s_0 s_1 \dots s_{N-1}$, N keystream bits.

Output. $u_{\gamma_n} \dots u_{L+\gamma_n-1}$, L consecutive bits of the LFSR sequence, where L is the LFSR length.

For each choice of the M-bit vector $u_{\gamma_n} \dots u_{\gamma_1-1}$

Compute the next $(L-M)$ bits of the LFSR sequence by

$$u_{t+\gamma_1} \leftarrow s_t + g(u_{t+\gamma_2}, \dots, u_{t+\gamma_n}), \;\; 0 \le t \le L-M \; .$$

Compute $(N-L)$ additional bits of the LFSR sequence with the LFSR recurrence relation, and the corresponding running-key bits, $\hat{s}_t$, for $L-M \le t < N-M$.

If the $N-L$ bits $\hat{s}_t$ are equal to the observed keystream sequence, then

return $(u_{\gamma_n} \dots u_{L+\gamma_n-1})$.

Table 3: Inversion attack

Exactly as in the original inversion attack, the basic step of the attack consists in deducing the bit $u_{t+\gamma_1}$ from the knowledge of the keystream bit s_t and of the M previous terms of the LFSR sequence, $(u_{t+\gamma_1+1}, \dots, u_{t+\gamma_n})$. For fixed values of s_t and of $(u_{t+\gamma_1+1}, \dots, u_{t+\gamma_n})$, the unknown bit $u_{t+\gamma_1}$ may take 0, 1 or 2 possible values. Then, an exhaustive search on the M bits $u_{\gamma_n} \dots u_{\gamma_1-1}$ of the LFSR sequence, can still be performed. For a given value of the M-bit vector $u_{\gamma_n} \dots u_{\gamma_1-1}$, a binary tree of depth $L-M$ representing all the solutions for the next $(L-M)$ bits of $\mathbf{u}$ is formed. Each node at level t corresponds to a guessed value of $(u_{t+\gamma_n} \dots u_{t+\gamma_1-1})$. Then, the number of edges out of this node is 0, 1 or 2 according to the number of solutions x of the equation $s_t = f(x, u_{t+\gamma_2}, \dots, u_{t+\gamma_n})$. If a tree of depth $L-M$ can be constructed from a given M-bit root, some additional bits of the LFSR sequence are computed and their consistency with the observed keystream is checked. It is shown that the typical number of surviving nodes at level $L-M$ is linear in L. Then, the typical complexity of the attack is $\mathcal{O}(L2^M)$. Exactly as in the inversion attack, the parameter involved in the attack is the effective memory size, *i.e.*,

$$M' = \frac{\gamma_1 - \gamma_n}{\gcd(\gamma_i - \gamma_{i+1})} \; .$$

Another technique based on a trellis representation and on the Viterbi algorithm is described in [LBGZ01]. Its efficiency is comparable to the generalized inversion attack.

Anne Canteaut.

References

[Gol96] J. Dj. Golić. On the security of nonlinear filter generators. In *Fast Software Encryption 1996*, Lecture Notes in Computer Science number 1039, pages 173–188. Springer-Verlag, 1996.

[**GCD00**] J. Dj. Golić, A. Clark, and E. Dawson. Generalized inversion attack on nonlinear filter generators. *IEEE Transactions on Computers*, 49(10):1100–1108, 2000.

[**LBGZ01**] S. Leveiller, J. Boutros, P. Guillot, and G. Zmor. Cryptanalysis of nonlinear filter generators with $\{0,1\}$-metric Viterbi decoding. In *Cryptography and Coding - 8th IMA International Conference*, Lecture Notes in Computer Science number 2260, pages 402–414. Springer-Verlag, 2001.

A5/1

A5/1 is the symmetric cipher used for encrypting over-the-air transmissions in the GSM standard. A5/1 is used in most European countries, whereas a weaker cipher, called A5/2, is used in other countries (a description of A5/2 and an attack can be found in [PF00]). The description of A5/1 was first kept secret but its design was reversed engineered in 1999 by Briceno, Golberg and Wagner. A5/1 is a synchronous stream cipher based on linear feedback shift registers (LFSRs). It has a 64-bit secret key.

A GSM conversation is transmitted as a sequence of 228-bit frames (114 bits in each direction) every 4.6 millisecond. Each frame is xored with a 228-bit sequence produced by the A5/1 running-key generator. The initial state of this generator depends on the 64-bit secret key, K, which is fixed during the conversation, and on a 22-bit public *frame number*, F.

The A5/1 running-key generator consists of 3 LFSRs of lengths 19, 22 and 23. Their characteristic polynomials are $X^{19}+X^5+X^2+X+1$, $X^{22}+X+1$ and $X^{23}+X^{15}+X^2+X+1$. For each frame transmission, the 3 LFSRs are first initialized to zero. Then, at time $t = 1, \ldots, 64$, the LFSRs are clocked, and the key bit K_t is xored to the feedback bit of each LFSR. For $t = 65, \ldots, 86$, the LFSRs are clocked in the same fashion, but the $(t-64)$-th bit of the frame number is now xored to the feedback bits.

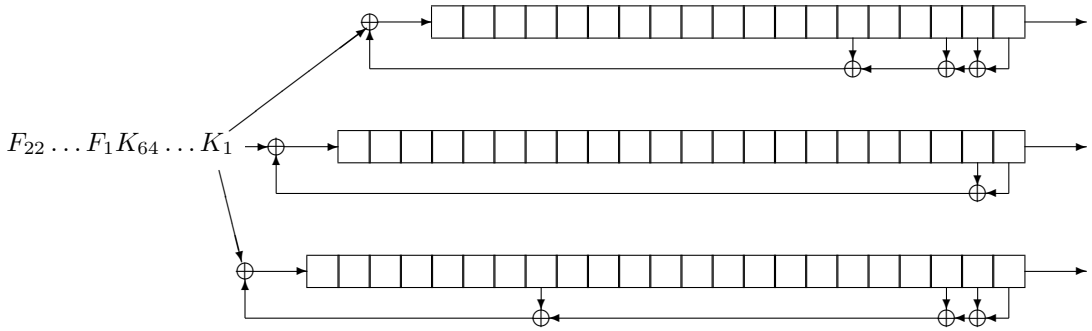


Figure 5: Initialization of the A5/1 running-key generator

After these 86 cycles, the generator runs as follows. Each LFSR has a clocking tap: tap 8 for the first LFSR, tap 10 for the second and the third ones (where the feedback tap corresponds to tap 0). At each unit of time, the majority value b of the 3 clocking bits is computed. A LFSR is clocked if and only its clocking bit is equal to b. For instance, if the 3 clocking bits are equal to $(1, 0, 0)$, the majority value is 0. The second and third LFSRs are clocked, but not the first one. The output of the generator is then given by the xor of the outputs of the 3 LFSRs. After the 86 initialization cycles, 328 bits are generated with the previously described irregular clocking. The first 100 ones are discarded and the following 228 bits form the running-key.

Several time-memory trade-off attacks have been proposed on A5/1 [BD00, BSW01]. They require the knowledge of a few seconds of conversation plaintext and run very fast. But, they need a huge precomputation time and memory. Another attack due to Ekdahl and Johansson [EJ03] exploits some weaknesses of the key initialization procedure. It requires a few minutes using 2-5 minutes of conversation plaintext without any notable precomputation and storage capacity.

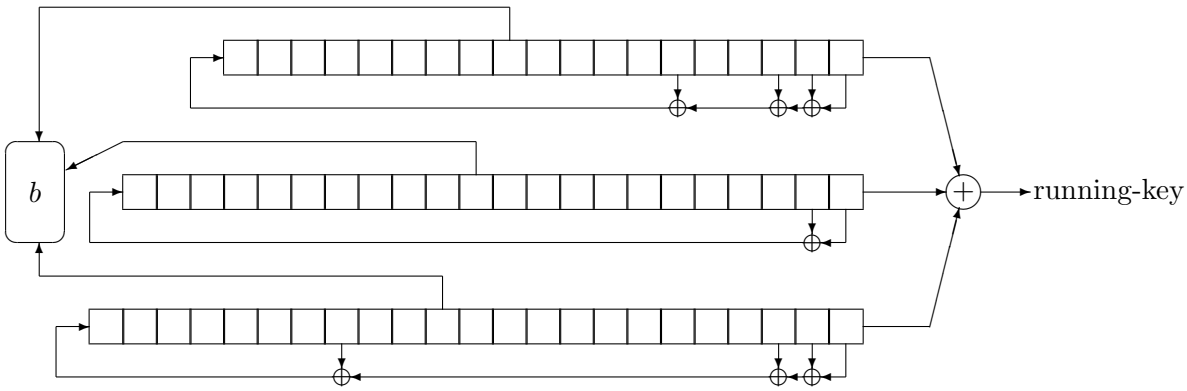


Figure 6: A5/1 running-key generator

Anne Canteaut.

References

[**BD00**] E. Biham and O. Dunkelman. Cryptanalysis of the A5/1 GSM stream cipher. In *Indocrypt 2000*, Lecture Notes in Computer Science number 1977, pages 43-51. Springer-Verlag, 2000.

[**BSW00**] A. Biryukov, A. Shamir, and D. Wagner. Real time attack of A5/1 on a PC. In *Fast Software Encryption 2000*, Lecture Notes in Computer Science number 1978, pages 1–18. Springer-Verlag, 2000.

[**EJ03**] P. Ekdahl and T. Johansson. Another attack on A5/1. *IEEE Transactions on Information Theory*, 49(1):284-289, 2003.

[**PF00**] S. Petrović and A. Fúster-Sabater. Cryptanalysis of the A5/2 algorithm. *Cryptology ePrint Archive, Report 2000/052*, 2000. Available on `http://eprint.iacr.org/`.

Linear syndrome attack

The *linear syndrome attack* is an attack on LFSR-based keystream generators which was presented by Zeng and Huang in 1988 [ZH88]. It is a weak version of the fast correlation attack which was independently proposed by Meier and Staffelbach [MS88].

Anne Canteaut.

References

[MS88] W. Meier and O. Staffelbach. Fast correlation attacks on stream ciphers. In *Advances in Cryptology - EUROCRYPT'88*, Lecture Notes in Computer Science number 330, pages 301–314. Springer-Verlag, 1988.

[ZH88] K. Zeng and M. Huang. On the linear syndrome method in cryptanalysis. In *Advances in Cryptology - CRYPTO'88*, Lecture Notes in Computer Science number 403, pages 469–478. Springer-Verlag, 1988.

[ZYR90] K. Zeng, C.H. Yang, and T.R.N. Rao. An improved linear syndrome algorithm in cryptanalysis with applications. In *Advances in Cryptology - CRYPTO'90*, Lecture Notes in Computer Science number 537, pages 34–47. Springer-Verlag, 1990.

Linear consistency attack

The *linear consistency attack* is a divide-and conquer technique which provides a known plaintext attack on stream ciphers. It was introduced by Zeng, Yang and Rao in 1989. It has been applied to various keystream generators, like the *Jenning generator* [ZYR89], the *stop-and-go generator* of Beth and Piper [ZYR89] and the E0 cipher used in Bluetooth [FL01].

The linear consistency attack applies as soon as it is possible to single out a portion K_1 of the secret key and to form a system $Ax = b$ of linear equations, where the matrix A only depends on K_1 and the right-side vector b is determined by the known keystream bits. Then, an exhaustive search on K_1 can be performed. The correct value of K_1 can be distinguished from a wrong one by checking whether the linear system is consistent or not. Once K_1 has been recovered, the solution x of the system may provide some additional bits of the secret key.

Anne Canteaut.

References

[FL01] S.R. Fluhrer and S. Lucks. Analysis of the E0 encryption system. In *Selected Areas in Cryptography - SAC 2001*, Lecture Notes in Computer Science number 2259, pages 38–48. Springer-Verlag, 2001.

[ZYR89] K. Zeng, C.H. Yang, and T.R.N. Rao. On the linear consistency test (LCT) in cryptanalysis with applications. In *Advances in Cryptology - CRYPTO'89*, Lecture Notes in Computer Science number 435, pages 164–174. Springer-Verlag, 1989.

Linear cryptanalysis for stream ciphers

A *linear cryptanalysis technique for stream ciphers* was presented by Golić in 1994. It relies on the same basic principles as the linear cryptanalysis for block ciphers introduced by Matsui. The linear cryptanalysis provides a known plaintext attack on various synchronous stream ciphers, which enables to distinguish the keystream from a truly random sequence. Such a *distinguishing attack* can be used for reducing the uncertainty of unknown plaintexts, or for recovering the unknown structure of the keystream generator. It may also lead to a divide-and-conquer attack when the structure of the keystream generator depends on a portion of the secret-key.

The linear cryptanalysis consists in finding some linear functions of the keystream bits which are not *balanced*, *i.e.*, which are not uniformly distributed. Such linear correlations are used for distinguishing the keystream sequence from a random sequence by a classical statistical test. Efficient techniques for finding biased linear relations between the keystream bits are presented in [Gol94, CHJ02].

The linear cryptanalysis leads to distinguishing attacks on several types of stream ciphers [Gol94, CHJ02] and to a reconstruction attack on combination generators [CF00].

Anne Canteaut.

References

[CF00] A. Canteaut and E. Filiol. Ciphertext only reconstruction of stream ciphers based on combination generators. In *Fast Software Encryption 2000*, Lecture Notes in Computer Science, number 1978, pages 165–180. Springer-Verlag, 2001.

[CHJ02] D. Coppersmith, S. Halevi, and C. Jutla. Cryptanalysis of stream ciphers with linear masking. In *Advances in Cryptology - CRYPTO 2002*, Lecture Notes in Computer Science number 2442, pages 515–532. Springer-Verlag, 2002.

[Gol94] J.Dj. Golić. Linear cryptanalysis of stream ciphers. In *Fast Software Encryption 1994*, Lecture Notes in Computer Science, number 1008, pages 154–169. Springer-Verlag, 1994.