The Achterbahn Stream Cipher

Berndt M. Gammel, Rainer Göttfert, Oliver Kniffler

berndt.gammel@infineon.com
rainer.goettfert@infineon.com
oliver.kniffler@infineon.com

Infineon Technologies AG
St.-Martin-Str. 76
81541 Munich, Germany

28 April 2005

Contents

1 Introduction

The proposed stream cipher *Achterbahn* is a binary additive stream cipher. In a binary additive stream cipher, the plaintext is given as a string $m_0, m_1, \ldots$ of elements of the finite field $\mathbb{F}_2$. The keystream $z_0, z_1, \ldots$ is a binary pseudo-random sequence. The sender encrypts the plaintext message according to the rule $c_t = m_t + z_t$ for all $t \geq 0$. The ciphertext $c_0, c_1, \ldots$ is decrypted by the receiver by adding bitwise the keystream $z_0, z_1, \ldots$ to the received ciphertext sequence $c_0, c_1, \ldots$. Sender and receiver produce the keystream $z_0, z_1, \ldots$ via identical copies of the *key stream generator*.

The keystream generator (also called *running key generator*) is a finite state machine having a finite number P of different internal states $S_0, S_1, \ldots, S_{P-1}$. The next-state function Λ governing the state progression has the form $S_{t+1} = \Lambda(S_t)$ for $t = 0, 1, \ldots, P-2$. Thus the internal state at time $t+1$ depends only on the internal state at time t. The initial internal state S_0 is derived from the secret key $K \in \mathbb{F}_2^k$ and a public initial value $IV \in \mathbb{F}_2^l$. The key length k is required to be $k = 80$. The length of the initial value vector (IV-vector) can assume any integer value between 0 and 64 that is congruent to 0 modulo eight. The restriction of the lengths of the IV-vector to multiples of eight is imposed to facilitate a high-speed implementation of the keystream generator. In the high-speed implementation, one byte of keystream is generated per clock cycle.

The output function λ of the keystream generator is parameterized by the key K and the initial value vector IV. We have $z_t = \lambda(S_t) = \lambda_{K,IV}(S_t)$ for $t = 0, 1, \ldots, P-1$. For any fixed pair (K, IV) of key and IV-vector, the produced keystream bit z_t at time t depends only on the internal state S_t at time t. As the keystream bits are independently produced of the plaintext, the proposed stream cipher belongs to the category of *synchronous stream ciphers*.

1.1 A brief description of the keystream generator

The basic ingredients of the keystream generator (KSG) are eight binary nonlinear feedback shift registers (NLFSR's) of lengths between 22 and 31, and a balanced 4th-order correlation immune Boolean combining function $R : \mathbb{F}_2^8 \to \mathbb{F}_2$. The NLFSR's are such that they can produce binary sequences of period $2^N - 1$, where N is the length of the shift register. Each shift register has a nonlinear feedback function, governing the internal state of the shift register, and a modifyable linear feedforward output function. The output functions of the eight NLFSR's deliver the eight input sequences for the Boolean combining function R which in turn outputs the running key.

The initial loading of each NLFSR, as well as the configurations of all output functions, depend on the secret key K and the public IV-vector (initial value). For each particular configuration of the linear feedforward functions, and for all possible initial loadings such that no shift register is loaded into the all-zero initial state, the KSG will produce a binary periodic sequence of period $> 2^{207}$ and linear complexity $> 2^{85}$.

Sequences produced under different configurations of the KSG are cyclically inequivalent. There are 2^{64} possibilities for the configuration, so that the KSG is able

to generate an ensemble $\mathcal{E}$ of 2^{64} translation distinct periodic sequences (each of which has period $> 2^{207}$ and linear complexity $> 2^{85}$). Given a fixed key $K \in \mathbb{F}_2^{80}$, the key-loading algorithm described below has the property that for distinct IV-vectors the corresponding configurations of the output function of the KSG will also be distinct. As a consequence, the produced keystream segments between any two resynchronization steps will belong to distinct sequences of the ensemble $\mathcal{E}$. Thus any unintended re-use of key material is impossible.

The NLFSR's have been selected under the objective to allow fast hardware implementations of the KSG. In a straightforward implementation, the KSG emits one bit of keystream per clock cycle. In the high-speed implementation, the KSG generates one byte (of exactly the same keystream) within each clock cycle.

2 Preliminaries

In this section we introduce the notation and terminology that will be used throughout the proposal. We use the symbol $\mathbb{F}_2$ to denote the binary finite field, and $\mathbb{F}_q$ to denote the general finite field of order q. The ring of polynomials in the indeterminate x and with coefficients from $\mathbb{F}_q$ is designated by $\mathbb{F}_q[x]$. A sequence of elements of $\mathbb{F}_q$ is given in the form $\sigma = (s_n)_{n=0}^{\infty} = (s_0, s_1, \dots)$. The set of all sequences of elements of $\mathbb{F}_q$ is designated by $\mathbb{F}_q^{\infty}$. For sequences $\sigma = (s_n)_{n=0}^{\infty}$ and $\tau = (t_n)_{n=0}^{\infty}$, addition $\sigma + \tau$ and multiplication $\sigma\tau$ are defined termwise: $\sigma + \tau = (s_n + t_n)_{n=0}^{\infty}$ and $\sigma\tau = (s_n t_n)_{n=0}^{\infty}$. A sequence $\sigma = (s_n)_{n=0}^{\infty}$ of $\mathbb{F}_q^{\infty}$ is called *periodic* if there existis a positive integer P such that $s_{n+P} = s_n$ for all $n \geq 0$. Thus here the term "periodic" always means what some authors call "purely periodic". If $\sigma = (s_n)_{n=0}^{\infty}$ is periodic and $P \geq 1$ is the least positive integer such that $s_{n+P} = s_n$ for all $n \geq 0$, then the integer P is referred to as *the period* of σ. We then write $P = \mathrm{per}(\sigma)$. Thus here the term "period of σ" always means what some authors call "the least period of σ".

The set $\mathbb{F}_q^{\infty}$ is an $\mathbb{F}_q$-vector space with respect to the above defined sum $\sigma + \tau$ and the scalar multiplication defined by $c\sigma = (cs_n)_{n=0}^{\infty}$ for all $c \in \mathbb{F}_q$ and for all $\sigma = (s_n)_{n=0}^{\infty} \in \mathbb{F}_q^{\infty}$. The shift operator $T : \mathbb{F}_q^{\infty} \to \mathbb{F}_q^{\infty}$, defined by $T\sigma = (s_{n+1})_{n=0}^{\infty} = (s_1, s_2, \dots)$ for all $\sigma = (s_n)_{n=0}^{\infty} \in \mathbb{F}_q^{\infty}$, is a useful linear operator on the $\mathbb{F}_q$-vector space $\mathbb{F}_q^{\infty}$.

For nonzero polynomials $f, g \in \mathbb{F}_q[x]$ and for positive integers a, b, the greatest common divisor of f and g, or of a and b, respectively is denoted by $\gcd(f, g)$ and $\gcd(a, b)$. Similiarly, $\mathrm{lcm}(f, g)$ and $\mathrm{lcm}(a, b)$ denote the least common multiple of the polynomials f and g, or of the integers a and b, respectively. Euler's function will be designated by the greek letter φ. If m is a positive integer, $\varphi(m)$ is the number of integers k with $1 \leq k \leq m$ and $\gcd(k, m) = 1$.

If $\sigma = (s_n)_{n=0}^{\infty}$ is a periodic sequence in $\mathbb{F}_q^{\infty}$ with $\mathrm{per}(\sigma) = P$, then the least positive integer N such that the N-tuples $\mathbf{s}_n = (s_n, s_{n+1}, \dots, s_{n+N-1})$, $0 \leq n \leq P-1$, are distinct is called the *span* or the *maximum order complexity* of σ. Equivalently, the periodic sequence $\sigma \in \mathbb{F}_q^{\infty}$ has maximum order complexity N (or span N) if σ could be generated by some feedback shift register over $\mathbb{F}_q$ of length N but by no shorter feedback shift register.

Let N be a positive integer. A binary feedback shift register of length N is

uniquely determined by its feedback function $F : \mathbb{F}_2^N \to \mathbb{F}_2$. The shift register consists of N consecutive 2-state memory units called *cells* regulated by an external clock. We denote the cells (or delay elements) of the shift register by $D_{N-1}, D_{N-2}, \ldots, D_1, D_0$. Each cell can store the bit 0 or 1. At each clock pulse, the bit in cell D_j is shifted one position to the right into cell D_{j-1} for $1 \leq j \leq N-1$. Cell D_{N-1} receives a new term, computed by the feedback function F from the N previous terms. The initial loading of the shift register can be described by a binary N-tuple called the *initial state vector*. To be precise, if $\mathbf{s}_0 = (s_0, s_1, \ldots, s_{N-1}) \in \mathbb{F}_2^N$ is the initial state vector, then at the outset the right-most cell D_0 will contain the bit s_0, the next cell D_1 will contain s_1, ..., and the left-most cell D_{N-1} will contain s_{N-1}. If at each clock pulse the content of cell D_0 is emitted, a binary ultimately periodic sequence $(s_0, s_1, \ldots, s_{N-1}, s_N, s_{N+1}, \ldots)$ is obtained. We refer to this sequence as the *standard output sequence* of the given feedback shift register corresponding to the initial state vector $\mathbf{s}_0 = (s_0, s_1, \ldots, s_{N-1})$.

The state of the shift register at time t can be described by the N-tuple $\mathbf{s}_t = (s_t, s_{t+1}, \ldots, s_{t+N-1}) \in \mathbb{F}_2^N$, where $s_{t+j} \in D_j$ for $0 \leq j \leq N-1$. If the next-state function of the feedback shift register (FSR), i.e. the function that maps the state $\mathbf{s}_t$ of the shift register at time t to the state $\mathbf{s}_{t+1}$ of the shift register at time $t+1$, is a bijection, then the FSR, as well as its feedback function, F are called *nonsingular*. A feedback function $F : \mathbb{F}_2^N \to \mathbb{F}_2$ is nonsingular if and only if the algebraic normal form of F has the form

$$F(x_0, x_1, \ldots, x_{N-1}) = x_0 + G(x_1, \ldots, x_{N-1}), \tag{1}$$

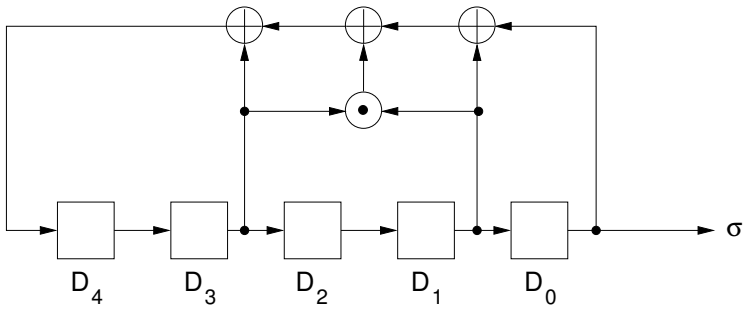
where $G : \mathbb{F}_2^{N-1} \to \mathbb{F}_2$ is a polynomial in the variables $x_1, \ldots, x_{N-1}$ (see Walker [35]).

If the feedback function F of an N-stage FSR is linear, one speaks of a *linear feedback shift register* (LFSR). Otherwise one speaks of a *nonlinear feedback shift register* (NLFSR). All feedback shift registers used in the design of the proposed stream cipher are nonsingular and nonlinear.

Example 1. Consider the binary 5-stage NLFSR with feedback function

$$F(x_0, x_1, x_2, x_3, x_4) = x_0 + x_1 + x_3 + x_1x_3.$$

The feedback shift register is shown in the following figure:



The standard output sequence corresponding to the initial state vector $\mathbf{s}_0 = (0, 0, 0, 0, 1)$ for the given feedback shift register is

$$\sigma = (0\,0\,0\,0\,1\,0\,1\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,0\,0\,1\,0\,0\,0\,1\,1\,1\,1\,1)^\infty,$$

a binary sequence of period $\mathrm{per}(\sigma) = 31$ and linear complexity $L(\sigma) = 30$. Note that the sequence $\sigma = (s_n)_{n=0}^{\infty}$ can be defined by the nonlinear recursion

$$s_{n+5} = s_{n+3}s_{n+1} + s_{n+3} + s_{n+1} + s_n \quad \text{for all } n \geq 0, \tag{2}$$

together with the initial values $s_0 = s_1 = s_2 = s_3 = 0$, and $s_4 = 1$. One can say that the above feedback shift register provides the hardware implementation of recursion (2). □

Definition 1. *Let $F : \mathbb{F}_2^N \to \mathbb{F}_2$ be the feedback function of an N-stage feedback shift register. The FSR is called* primitive *if for any nonzero initial state vector of $\mathbb{F}_2^N$ the corresponding standard output sequence of the FSR has period $2^N - 1$, and if $F(\mathbf{0}) = 0$, where $\mathbf{0}$ is the zero vector of $\mathbb{F}_2^N$.*

Note that a primitive feedback shift register must be nonsingular. The 5-stage NLFSR considered in Example 1 is a primitive feedback shift register.

If the feedback function $F : \mathbb{F}_2^N \to \mathbb{F}_2$ of an N-stage FSR is linear, that is, if

$$F(x_0, x_1, \ldots, x_{N-1}) = a_0x_0 + a_1x_1 + \cdots + a_{N-1}x_{N-1},$$

then the N-degree polynomial $c \in \mathbb{F}_2[x]$ given by

$$c(x) = x^N + F(1, x, x^2, \ldots, x^{N-1}) = x^N + a_{N-1}x^{N-1} + \cdots + a_1x + a_0$$

is called the *characteristic polynomial* of the LFSR defined by the feedback function F. It is well known (see e.g. Lidl and Niederreiter [20, Chap. 8]) that an N-stage LFSR whose characteristic polynomial is a primitive polynomial over $\mathbb{F}_2$, will generate a periodic sequence of period $2^N - 1$ for any nonzero initial state vector. Thus an LFSR with a primitive characteristic polynomial is a primitive FSR in the sense of Definition 1.

There are $\varphi(2^N - 1)/N$ primitive binary polynomials of degree N. Therefore, there are $\varphi(2^N-1)/N$ binary primitive N-stage LFSR's. The total number of binary primitive N-stage FSR's, linear or nonlinear, is given by

$$B_N = 2^{2^{N-1}-N}.$$

This is the number of translation distinct (or cyclically inequivalent) binary de-Bruijn sequences [5], and there is a one-to-one correspondence between binary de-Bruijn sequences and sequences produced by binary primitive FSR's. The number of nonsingular binary N-stage FSR's is given by

$$C_N = 2^{2^{N-1}}.$$

This follows immediately from equation (1), since there are C_N different functions $G : \mathbb{F}_2^{N-1} \to \mathbb{F}_2$. Comparing numbers B_N and C_N, we find $B_N/C_N = 1/2^N$. Thus, on the average one out of 2^N nonsingular binary FSR's is primitive.

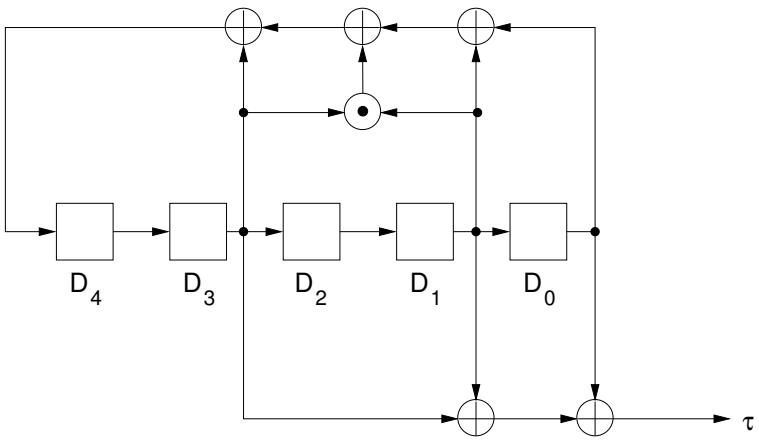
An N-stage binary primitive FSR decomposes the vector space $\mathbb{F}_2^N$ into two disjoint cycles. The long cycle contains all nonzero vectors of $\mathbb{F}_2^N$. The short cycle consists only of the zero vector $\mathbf{0} \in \mathbb{F}_2^N$. Any two standard output sequences of some binary primitive FSR corresponding to distinct nonzero initial state vectors are shifted versions of each other. Since the two sequences are periodic, both sequences possess the same minimal polynomial and, consequently, have the same linear complexity (see [20, Theorem 8.48]). Therefore the following definition makes sense.

Definition 2. *Let $\mathbb{F}_2^N \to \mathbb{F}_2$ be the feedback of some binary primitive N-stage FSR. The minimal polynomial, the period, and the linear complexity of the FSR is defined to be the minimal polynomial, period, and linear complexity, respectively, of the standard output sequence of the FSR corresponding to any nonzero initial state vector.*

Consider the binary primitive 5-stage NLFSR of Example 1. We can say that this shift register has period 31 and linear complexity 30. The minimal polynomial of the shift register is the product of all six irreducible binary polynomials of degree 5. (See also Example 4 in Section 8).

Consider again 5-stage NLFSR of Example 1. If we apply to the stages of the shift register a linear feedforward function, then the shift register will produce a new output sequence $\tau = (t_n)_{n=0}^{\infty}$. In contrast to the standard output sequence $\sigma = (s_n)_{n=0}^{\infty}$ which is obtained by emitting the content of cell D_0 at any clock pulse, the terms of sequence τ are obtained by outputting the contents of several cells and adding together the outputs. For instance, in the particular example illustrated below, we have

$$t_n = s_n + s_{n+1} + s_{n+3} \qquad \text{for all } n \geq 0. \tag{3}$$



Using the shift operator T, equation (3) can be written as

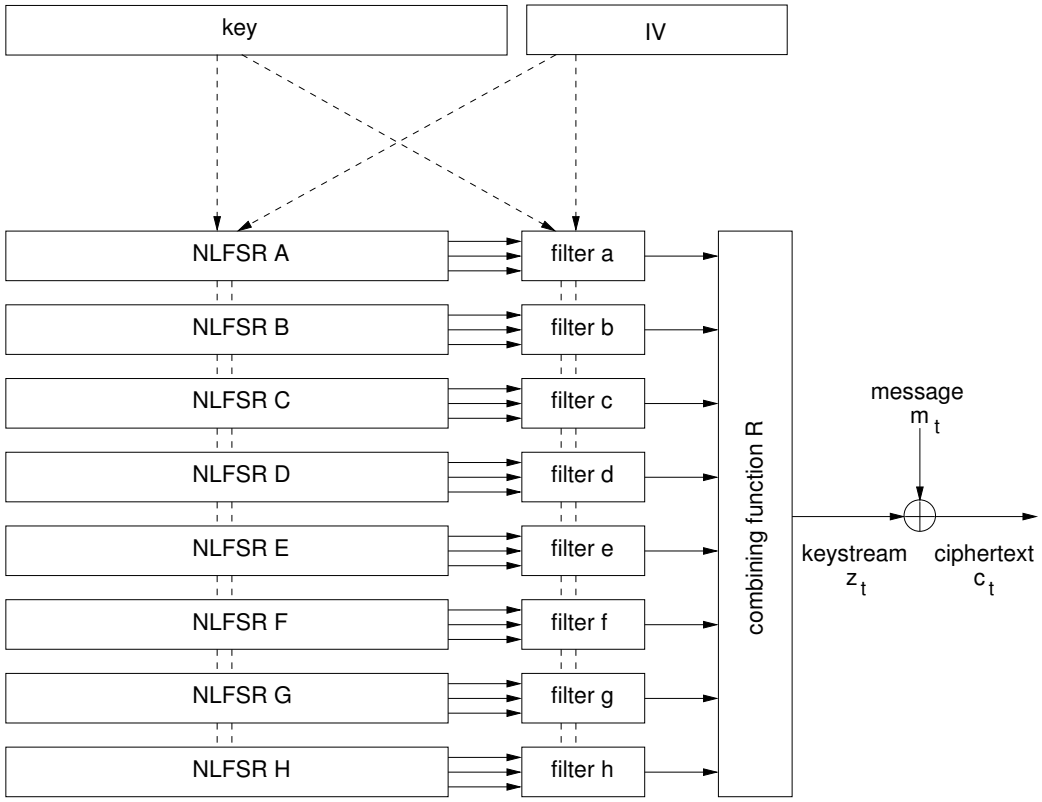
$$\tau = \sigma + T\sigma + T^3\sigma = (1 + T + T^3)\sigma.$$

We call the polynomial $f(x) = x^3 + x + 1$ the *filter polynomial*. The sequence $\tau = f(T)\sigma$ is obtained by *linearly filtering* the standard output sequence σ using the filter polynomial f.

The concept of linearly filtering NLFSR sequences was introduced in [11]. It has been shown that if the underlying N-stage NLFSR is primitive and the degree of the applied filter polynomial f is small compared to N, then the filtered sequence $\tau = f(T)\sigma$ will still have good distribution properties but in general a larger maximum order complexity as the original sequence σ. We include the article [11] as a file on the submission CD.

3 Detailed description of the keystream generator

The overall structure of the keystream generator (KSG) is depicted in the following figure.



The core of the KSG consists of eight primitive (in the sense of Definition 1) binary NLFSR's labeled with capital letters A, B, C, ..., H. Each NLFSR is endowed with a linear feedforward logic described by filter polynomials $a(x)$, $b(x)$, $c(x)$, ..., $h(x)$. The linear feedforward logics supply the Boolean combining function R with inputs. The function R then outputs the keystream. At the outset—under the control of the secret key K and public initial value IV, all eight NLFSR's are loaded and the all linear feedforward functions are adjusted.

3.1 The Boolean combining function

The Boolean combining function $R : \mathbb{F}_2^8 \to \mathbb{F}_2$ has algebraic degree 3 and nonlinearity 64. The algebraic normal form of R is

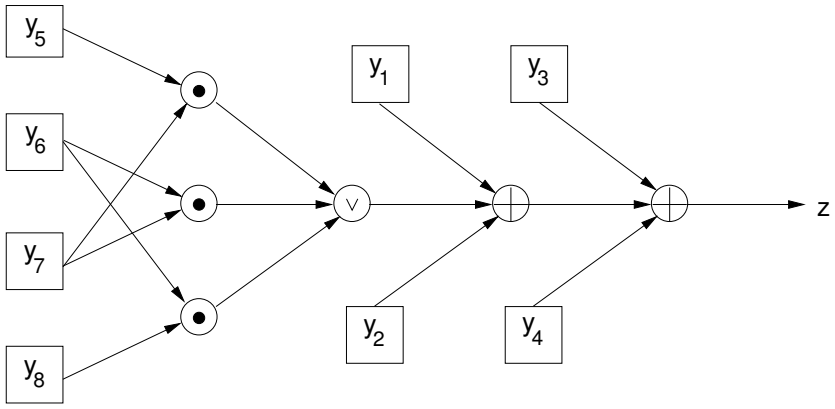
$$\begin{aligned} R(y_1, y_2, \ldots, y_8) = y_1 + y_2 + y_3 + y_4 + y_5y_7 + y_6y_7 + y_6y_8 \\ + y_5y_6y_7 + y_6y_7y_8. \end{aligned} \tag{4}$$

Alternatively, using the logical OR-symbol $\vee$, the function R can be represented in the form

$$R(y_1, y_2, \ldots, y_8) = y_1 + y_2 + y_3 + y_4 + y_5y_7 \vee y_6y_7 \vee y_6y_8. \tag{5}$$

Note that $a \vee b = a + b + ab$. The Boolean function R is balanced and 4th order correlation immune. The latter can, for instance, be checked by evaluating the nonlinear part of R, that is, the expression $y_5y_7 \vee y_6y_7 \vee y_6y_8$ for all 16 possibilities for the variables y_5, y_6, y_7, y_8, and observing that the values 0 and 1 occur equally often.

The algebraic degree 3 of the Boolean function R is large enough to guarantee that the produced keystream ζ will have linear complexity $\geq 2^{85}$. The order of correlation immunity 4 is the maximum possible value for balanced, 8-variable, degree-3 Boolean functions according to Siegenthaler [33]. There are other 8-variable Boolean functions of algebraic degree 3 having order of resiliency 4. The particular one presented in (4) was chosen mainly because its alternative representation (5) which has a simple realization in hardware. See the following picture:



3.2 The feedback shift registers

The principal items of the KSG are eight binary primitive nonlinear feedback shift registers. Throughout the proposal these NLFSR's are labeled by the capital letters A, B, C, ..., H. The lengths, periods, linear complexities, and nonlinearities of the eight NLFSR's are given in the following table. Periods and linear complexities of a binary primitive FSR's are understood in the sense of Definition 2.

NLFRS's				
label	length	period	linear complexity	nonlinearity
A	22	$2^{22} - 1$	$L_A = 2^{22} - 13$	30208
B	23	$2^{23} - 1$	$L_B = 2^{23} - 2$	245760
C	25	$2^{25} - 1$	L_C	499712
D	26	$2^{26} - 1$	L_D	233472
E	27	$2^{27} - 1$	L_E	983040
F	28	$2^{28} - 1$	L_F	237568
G	29	$2^{29} - 1$	L_G	999424
H	31	$2^{31} - 1$	L_H	999424

Of course, the listing of the periods is redundant, as a primitive binary FSR of length N has, by definition, period $2^N - 1$. Note that shift registers G and H have the same nonlinearities.

The computations of the linear complexities of the NLFSR's are currently (25 April 2005) under process. For the NLFSR A and B we found $L_A = 2^{22} - 13$ and $L_B = 2^{23} - 2$, respectively. The upper bound for the linear complexity of an N-stage binary primitive FSR is $2^N - 2$. This is an immediate consequence of Theorem 16. The upper bound $2^N - 2$ also seems to be a typical value for the linear complexity of N-stage binary primitive NLFSR's. By investigating more than 100 Million randomly selected binary N-stage NLFSR's with $4 \leq N \leq 20$, we observed that the upper bound $2^N - 2$ for the linear complexity was attained for more than half the sequences produced by the NLFSR's found. Allmost all sequences had linear complexities close the periods.

This does not come as a surprise though in the light of the investigations carried out by Rueppel [29], Dai and Yang [8], and Meidl and Niederreiter [22] concerning the linear complexity of periodically repeated random strings.

At any rate, since no lower bounds for the linear complexities of binary span-N, period-$2^N - 1$ sequences have been published so far, the linear complexities of the remaining seven driving NLFSR's used in our KSG must be computed. To this end the formula of Laksov [19] in combination with fast implementations of the Euclidean algorithm for binary polynomials is of interest.

For the moment we shall be very conservative assuming that the linear complexities of the envolved primitive NLFSR's are greater than half the periods. Thus we shall assume that $L_C \geq 2^{24}$, $L_D \geq 2^{25}$, $L_E \geq 2^{26}$, $L_F \geq 2^{27}$, $L_G \geq 2^{28}$, and $L_H \geq 2^{30}$. We already know that $L_A = 2^{22} - 13$. $L_B = 2^{23} - 2$.

The feedback functions of the eight driving NLFSR's are given by

$$\begin{aligned} A(x_0, x_1, \ldots, x_{21}) = {} & x_0 + x_5 + x_6 + x_7 + x_{10} + x_{11} + x_{12} + x_{13} + x_{17} + x_{20} \\ & + x_2x_7 + x_4x_{14} + x_8x_9 + x_{10}x_{11} + x_1x_4x_{11} + x_1x_4x_{13}x_{14}; \end{aligned}$$

$$\begin{aligned} B(x_0, x_1, \ldots, x_{22}) = {} & x_0 + x_6 + x_7 + x_9 + x_{11} + x_{12} + x_{14} + x_{15} + x_{17} + x_{19} + x_{21} \\ & + x_1x_4 + x_2x_7 + x_5x_9 + x_6x_{10} + x_2x_4x_8 + x_1x_3x_5x_{10} \\ & + x_4x_{11}x_{12}x_{13}; \end{aligned}$$

$$\begin{aligned} C(x_0, x_1, \ldots, x_{24}) = {} & x_0 + x_1 + x_3 + x_5 + x_6 + x_7 + x_9 + x_{12} + x_{14} + x_{15} + x_{17} \\ & + x_{18} + x_{22} + x_1x_6 + x_4x_{13} + x_8x_{16} + x_{12}x_{15} + x_5x_{11}x_{14} \\ & + x_1x_4x_{11}x_{15} + x_2x_5x_8x_{10}; \end{aligned}$$

$$\begin{aligned} D(x_0, x_1, \ldots, x_{25}) = {} & x_0 + x_1 + x_4 + x_5 + x_7 + x_8 + x_9 + x_{13} + x_{14} + x_{16} + x_{20} \\ & + x_{24} + x_1x_6 + x_4x_7 + x_{12}x_{16} + x_{15}x_{17} + x_4x_{15}x_{17} + x_7x_9x_{10} \\ & + x_1x_3x_{14}x_{16} + x_8x_{11}x_{12}x_{17}; \end{aligned}$$

$$\begin{aligned} E(x_0, x_1, \ldots, x_{26}) = {} & x_0 + x_1 + x_2 + x_6 + x_8 + x_9 + x_{10} + x_{13} + x_{14} + x_{16} + x_{19} \\ & + x_{21} + x_{23} + x_1x_8 + x_3x_{12} + x_{11}x_{17} + x_{15}x_{18} + x_5x_6x_{15} \\ & + x_3x_5x_{16}x_{17} + x_7x_{12}x_{14}x_{15}; \end{aligned}$$

$$F(x_0, x_1, \ldots, x_{27}) = x_0 + x_1 + x_2 + x_7 + x_{15} + x_{17} + x_{19} + x_{20} + x_{22} + x_{27} + x_9x_{17} + x_{10}x_{18} + x_{11}x_{14} + x_{12}x_{13} + x_5x_{14}x_{19} + x_6x_{10}x_{12} + x_6x_9x_{17}x_{18} + x_{10}x_{12}x_{19}x_{20};$$

$$G(x_0, x_1, \ldots, x_{28}) = x_0 + x_2 + x_3 + x_5 + x_6 + x_9 + x_{14} + x_{15} + x_{16} + x_{18} + x_{21} + x_{27} + x_5x_7 + x_6x_{20} + x_{10}x_{14} + x_{13}x_{18} + x_8x_{19}x_{21} + x_{11}x_{16}x_{18} + x_1x_5x_{15}x_{21} + x_2x_7x_{17}x_{20};$$

$$H(x_0, x_1, \ldots, x_{30}) = x_0 + x_3 + x_5 + x_7 + x_{10} + x_{16} + x_{17} + x_{18} + x_{19} + x_{20} + x_{21} + x_{24} + x_{30} + x_5x_{15} + x_{11}x_{18} + x_{16}x_{22} + x_{17}x_{21} + x_1x_2x_{19} + x_1x_{12}x_{14}x_{17} + x_2x_5x_{13}x_{20}.$$

There is yet another NLFSR contained in the proposed stream cipher. This NLFSR is labeled by the capital letter V. The NLFSR V is nonsingular but not primitive. The length of this shift register is 64. The contents of shift register V are not changed during encryption. It is only used during key and IV loading. The secret key K and the public IV-vector are fed into the register V which subsequently performs a couple of shifts. The final contents of register V defines the configurations of the linear feedforward output logics of the driving NLFSR's. The feedback function of NLFSR V is

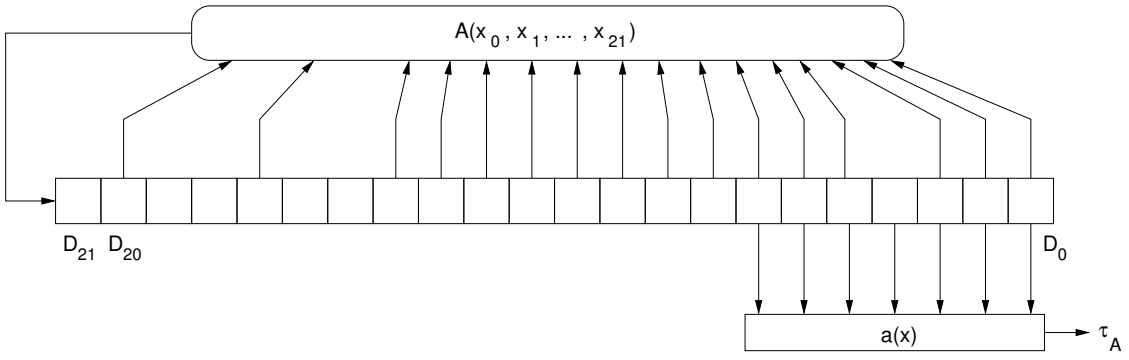
$$V(x_0, x_1, \ldots, x_{63}) = 1 + x_0 + x_3 + x_7 + x_{10} + x_{12} + x_{27} + x_{28} + x_{38} + x_{46} + x_{47} + x_8x_{20} + x_{17}x_{23} + x_{24}x_{25} + x_{29}x_{31} + x_{33}x_{34}x_{37} + x_1x_3x_9x_{10} + x_{39}x_{41}x_{51}x_{52}.$$

3.3 The linear feedforward functions

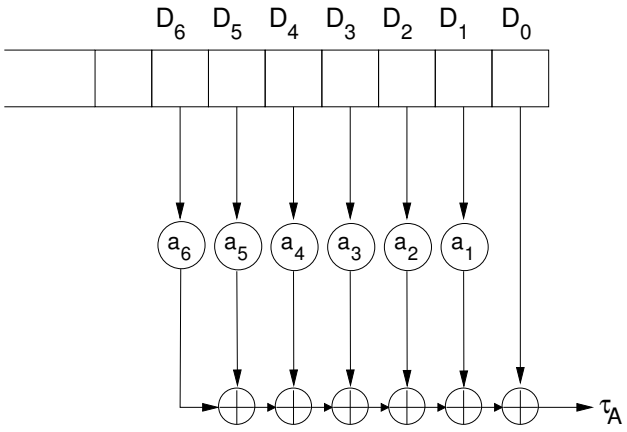
Each driving NLFSR A, B, C, $\ldots$, H is endowed with a configurable linear feedforward output function. The linear feedforward output function can be described by the filter polynomial (see Section 2). The binary filter polynomial $a(x)$ for NLFSR A has degree at most 6. All filter polynomials will have nonzero constant terms. Thus the polynomial $a \in \mathbb{F}_2[x]$ has the form

$$a(x) = a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + 1. \tag{6}$$

There are $2^6 = 64$ possibilities for the filter polynomial $a(x)$, as there appear six binary coefficients $a_1, a_2, \ldots, a_6$ in (6). For the NLFSR A, we depict the situation in the following figure



If we enlarge the right part in the picture, we get:



If the coefficient a_j, $1 \leq j \leq 6$, is 1, the wire is connected and the content of cell D_j contributes to the output. If $a_j = 0$, the wire is disconnected and the content of D_j is ignored. Note that cell D_0 is always connected to the output line. If all six coefficients of $a(x)$ are zero, in other words if the filter polynomial is equal to the constant polynomial $a(x) = 1$, the standard output sequence $\sigma_A = (s_n)_{n=0}^{\infty}$ is emitted. Using the shift operator T, we can write the output sequence $\tau_A = (t_n)_{n=0}^{\infty}$ in the form $\tau_A = a(T)\sigma_A$ (see Section 2).

The filter polynomials that will determine the output values of the eight NLFSR's A, B, C, ..., H are designated by $a(x)$, $b(x)$, $c(x)$, ..., $h(x)$, respectively. For each filter polynomial, the maximum permissible degree is given in the following table.

The Filter Polynomials		
NLFSR	filter polynomials	maximum degrees
A	$a(x)$	6
B	$b(x)$	7
C	$c(x)$	7
D	$d(x)$	8
E	$e(x)$	8
F	$f(x)$	9
G	$g(x)$	9
H	$h(x)$	10

Note that the sum of the maximum permissible degrees of all eight filter polynomials is 64. As a consequence, the KSG has 2^{64} different configurations for its output function.

Theorem 1. *If the NLFSR's A, B, C, ..., H are loaded with any nonzero initial state vectors, then for all filter polynomials $a(x)$, $b(x)$, $c(x)$, ..., $h(x)$, the produced output sequences $\tau_A, \tau_B, \ldots, \tau_H$ have periods $2^N - 1$, where N is the length of the corresponding NLFSR.*

Proof. Let $m_A(x), m_B(x), \ldots, m_H(x)$ be the minimal polynomials associated with the NLFSR's A, B, C, ..., H, respectively. By [11, Proposition 1] it suffices to check that each of the polynomials $m_A(x), m_B(x), \ldots, m_H(x)$ is divisible by a binary primitive polynomial of degree N, where N is the length of the corresponding shift register. Given a periodic binary sequence σ of period $P = 2^N - 1$, it can be checked whether or not a given binary polynomial f of degree N divides the minimal polynomial m_σ of σ without actually knowing the minimal polynomial m_σ. One merely has to check whether the polynomial $g(x) = (x^P - 1)/f(x)$ is still a characteristic polynomial of σ. The polynomial f divides m_σ precisely if g is not a characteristic polynomial of σ. See Corollary 1. In this manner we verified that all minimal polynomials $m_A(x), m_B(x), \ldots, m_H(x)$ are divisible by some primitive binary polynomials of the the required degrees. □

Theorem 2. *If the NLFSR's A, B, C, ..., H are loaded with any nonzero initial state vectors, then for all filter polynomials $a(x)$, $b(x)$, $c(x)$, ..., $h(x)$, the produced output sequences $\tau_A, \tau_B, \tau_C, \ldots, \tau_H$ have linear complexities greater or equal to $L'_A, L'_B, L'_C, \ldots, L'_H$, where these numbers are given in the table below.*

Proof. Recall that by definition the linear complexity of NLFSR A, say, is equal to the linear complexity of the standard output sequence σ_A obtained from any nonzero initial state vector. In other words, $L_A = L(\sigma_A)$. The filter-output sequence τ_A is related to σ_A by $\tau_A = a(T)\sigma_A$, where $a \in \mathbb{F}_2[x]$ is the applied filter polynomial. The assertion now follows from Theorems 12 and 15. □

NLFSR	linear complexity of NLFSR	lower bound for linear complexity of NLFSR-output sequence τ
A	L_A	$L'_A = L_A - 2$
B	L_B	$L'_B = L_B$
C	L_C	$L'_C = L_C - 5$
D	L_D	$L'_D = L_D - 2$
E	L_E	$L'_E = L_E - 6$
F	L_F	$L'_F = L_F - 9$
G	L_G	$L'_G = L_G$
H	L_H	$L'_H = L_H$

3.4 Linear complexity and period of the keystream

Suppose that each NLFSR A, B, ..., H is loaded with a nonzero initial state vector. Let $a(x)$, $b(x)$, ..., $h(x)$ be filter polynomials fulfilling the requirements mentioned in the preceding section. Let σ_A, σ_B, ..., σ_H be the standard output sequences for the shift registers corresponding to the applied nonzero initial state vectors. Let τ_A, τ_B, ..., τ_H be the NLFSR-output sequences produced by applying the above filter polynomials. Using the shift operator T, we can write $\tau_A = a(T)\sigma_A$, $\tau_B = b(T)\sigma_B$, ..., $\tau_H = h(T)\sigma_H$. The sequences τ_A, τ_B, ..., τ_H are combined by the Boolean combining function

$$R(y_1, y_2, \ldots, y_8) = y_1 + y_2 + y_3 + y_4 + y_5y_7 + y_6y_7 + y_6y_8 + y_5y_6y_7 + y_6y_7y_8.$$

to produce the keystream $\zeta = (z_n)_{n=0}^{\infty}$.

The NLFSR outputs are assigned to the Boolean function inputs according to the following mapping:

Input variable	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
NLFSR	A	C	D	E	B	G	H	F

Therefore

$$\zeta = R(\tau_A, \tau_C, \tau_D, \tau_E, \tau_B, \tau_G, \tau_H, \tau_F) = \tau_A + \tau_C + \tau_D + \tau_E + \tau_B\tau_H + \tau_G\tau_H + \tau_F\tau_G + \tau_B\tau_G\tau_H + \tau_F\tau_G\tau_H. \quad (7)$$

We want to determine the minimal polynomial m_ζ of ζ, which then will yield the linear complexity $L(\zeta) = \deg(m_\zeta)$ and the period $\mathrm{per}(\zeta) = \mathrm{ord}(m_\zeta)$ of the keystream ζ. To this end we first consider the minimal polynomials of the nine summands in (7).

Applying Theorems 15 and 16, we conclude that the minimal polynomial of the sequence τ_A, for instance, has a canonical factorization over $\mathbb{F}_2$ of the form

$$m_{\tau_A} = \prod_{j=1}^{r} h_j, \quad (8)$$

where $h_1, \ldots, h_r$ are distinct irreducible binary polynomials whose degrees divide 22 and are greater than 1. Thus all irreducible polynomials appearing in equation (8) have degrees 2, 11, or 22. Similiarly, we conclude that the minimal polynomial of τ_B contains only irreducible binary polynomials of degree 23, that the minimal polynomial of τ_C contains only irreducible factors of degrees 5 or 25, and so on. Applying Theorems 13 and 17, we find all possible degrees for the irreducible polynomials appearing in the canonical factorizations of the minimal polynomials of the product sequences that appear in (7). We summarize the results in the following table and the succeeding theorem.

NLFSR-output sequence	degrees of irreducible polynomials that may occur in the canonical factorization of the minimal polynomial of the sequence
τ_A	2, 11, 22
τ_C	5, 25
τ_D	2, 13, 26
τ_E	3, 9, 27
$\tau_B\tau_H$	713
$\tau_G\tau_H$	899
$\tau_F\tau_G$	58, 116, 203, 406, 812
$\tau_B\tau_G\tau_H$	20677
$\tau_F\tau_G\tau_H$	1798, 3596, 6293, 12586, 25172

Theorem 3. *The canonical factorization of the minimal polynomial of the keystream ζ consists only of irreducible binary polynomials of degrees* 2, 3, 5, 9, 11, 13, 22, 25, 26, 27, 58, 116, 203, 406, 713, 812, 899, 1798, 3596, 6293, 12586, 20677, *and* 25172.

Note that the minimal polynomials of τ_A and τ_C may both contain the irreducible factor $h(x) = x^2 + x + 1$, which is the only irreducible binary polynomial of degree 2. Apart from that the minimal polynomials of the nine sequences listed in the above table are pairwise relatively prime. Using Theorem 11, we find that m_ζ is the product of the minimal polynomials τ_A, τ_C, τ_D, τ_E, $\tau_B\tau_H$, $\tau_G\tau_H$, $\tau_F\tau_G$, $\tau_B\tau_G\tau_H$, and $\tau_F\tau_G\tau_H$ divided by the polynomial $(x^2 + x + 1)^k$, where $k \in \{0, 1, 2\}$.

Theorem 4. *For all 2^{64} configurations of the output function of the KSG corresponding to all possible combinations of the filter polynomials $a(x)$, $b(x)$, ..., $h(x)$, and for all initializations of the eight NLFSR's with nonzero initial state vectors, the linear complexity $L(\zeta)$ of the produced keystream ζ satisfies*

$$\begin{aligned} L(\zeta) \geq\ & L'_A + L'_C + L'_D + L'_E + L'_B L'_H + L'_G L'_H \\ & + L'_F L'_G + L'_B L'_G L'_H + L'_F L'_G L'_H - 4, \end{aligned}$$

where the primed numbers are related to the linear complexities L_A, L_B, ..., L_H of the underlying NLFSR's by the equations $L'_A = L_A - 2 = 2^{22} - 15$, $L'_B = L_B = 2^{23} - 2$, $L'_C = L_C - 5$, $L'_D = L_D - 2$, $L'_E = L_E - 6$, $L'_F = L_F - 9$, $L'_G = L_G$, and $L'_H = L_H$. Under the assumption made at the end of Section 3.2 we obtain

$$L(\zeta) \geq 2^{85}.$$

Theorem 5. *For all 2^{64} configurations of the output function of the KSG corresponding to all possible combinations of the filter polynomials $a(x)$, $b(x)$, ..., $h(x)$,*

and for all initializations of the eight NLFSR's with nonzero initial state vectors, the produced keystream ζ has period

$$\begin{aligned}\mathrm{per}(\zeta) = \frac{1}{9}\left(2^{22}-1\right)\left(2^{23}-1\right)\left(2^{25}-1\right)\left(2^{26}-1\right) \\ \left(2^{27}-1\right)\left(2^{28}-1\right)\left(2^{29}-1\right)\left(2^{31}-1\right).\end{aligned} \tag{9}$$

This implies that

$$\mathrm{per}(\zeta) \geq 2^{207}.$$

Proof. By Theorem 8, we have $\mathrm{per}(\zeta) = \mathrm{ord}(m_\zeta)$. From Theorems 15 and 17, we know that the minimal polynomial m_ζ has no repeated factors. Let $m_\zeta = g_1, \ldots, g_k$ be the canonical factorization of m_ζ over $\mathbb{F}_2$. By Theorem 9

$$\mathrm{ord}(m_\zeta) = \mathrm{lcm}\left(\mathrm{ord}(g_1), \ldots, \mathrm{ord}(g_k)\right). \tag{10}$$

In Section 3.3, we pointed out that the minimal polynomial of each sequence τ_A, τ_B, $\ldots$, τ_H contains a primitive binary polynomial of degree N, where N is the length of the corresponding NLFSR. Recall that the order of an irreducible binary polynomial of degree N always divides $2^N - 1$. If the polynomial is primitive, its order is equal to $2^N - 1$. Putting these facts together and using Theorem 14, we conclude that equation (10) is equivalent to

$$\mathrm{ord}(m_\zeta) = \mathrm{lcm}\left(2^{22}-1, 2^{23}-1, 2^{25}-1, 2^{26}-1, 2^{27}-1, 2^{28}-1, 2^{29}-1, 2^{31}-1\right).$$

Using Lemma 2 and the relation $\mathrm{lcm}(u,v)\,\gcd(u,v) = uv$, we obtain

$$\mathrm{lcm}\left(2^{22}-1, 2^{26}-1, 2^{28}-1\right) = \frac{1}{9}\left(2^{22}-1\right)\left(2^{26}-1\right)\left(2^{28}-1\right).$$

Another application of Lemma 2 now yields equation (9). □

3.5 The reduced keystream generator

In a reduced form of the KSG, the ability of changing the configuration of the output function of the KSG after each resynchronisation is removed. No linear feedforward logics are implemented in the reduced KSG. Instead the standard output function for each shift register is used. In the standard output function at each clock pulse, the content of cell D_0 is forwarded to the Boolean combiner R. Note that the standard output function of the reduced KSG corresponds to the special configuration of the linear feedforward logics in the full-fledged KSG in which all eight filter polynomials are equal 1. That is $a(x) = 1$, $b(x) = 1$, $\ldots$, $h(x) = 1$.

The motivation of considering besides the full-fledged KSG also a reduced version of the KSG originates in hardware savings. For instance, the NLFSR V with its 64 flip-flops can be saved in the reduced KSG. A comparison between the full-fledged KSG and the reduced KSG in terms of hardware costs is given in Section 7.

4 The key-loading algorithm

In this Section we are going to describe a method for initializing the various feedback shift registers in the KSG. An important design criteria of the key loading algorithm was its resistance against side-channel attacks, especially against differential power analaysis.

In the key-loading algorithm, the eight driving NLFSR's are initialized using the $k = 80$ bits of the secret key K and the bits derived from the public initial value vector IV. In the full-fledged KSG, in addition, the configuration feedback shift register V is loaded using the key bits and the bits of the initial value vector. Let $K = (k_0, k_1, \ldots, k_{79})$ be the secret key, and let $IV = (i_0, i_1, \ldots, i_{l-1})$ be the initial value vector (IV-vector for short). The length l of the IV-vector can be any value in $\{0, 8, 16, 24, 32, 40, 48, 56, 64\}$. Here $l = 0$ means that no IV-vector exists.

We first concatenate the key K and the IV-vector IV to obtain the *interim key*

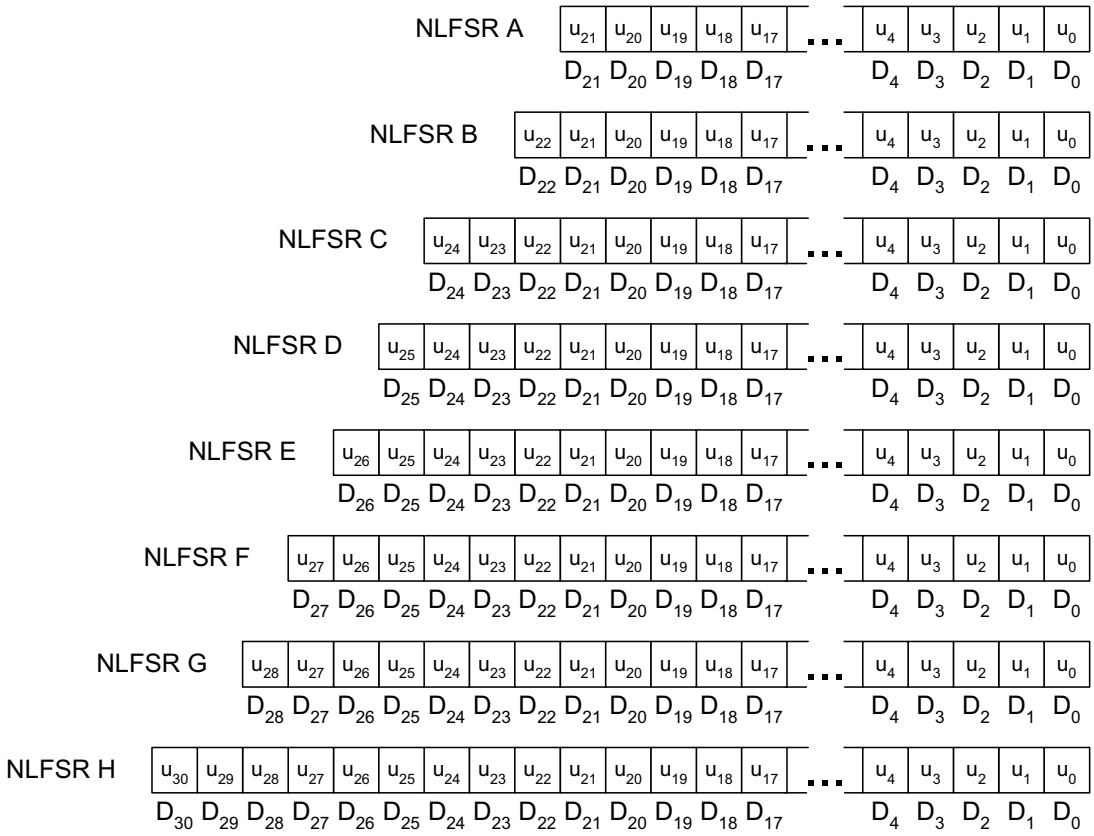
$$\mathbf{u}_r = (K, IV) = (k_0, k_1, \ldots, k_{79}, i_0, i_1, \ldots, i_{l-1}).$$

The length r of the interim key is given by $r = k + l = 80 + l$, and can thus take on any value in $\{80, 88, 96, 104, 112, 120, 128, 136, 144\}$. The key-loading algorithm consists of several steps.

Phase 1. Load the first bits of the interim key $\mathbf{u}_r$ into the cells of the eight driving NLFSR's A, B, ..., H, as well as into the cells of the configuration feedback shift register V. If the relevant shift register has length N, it receives the N bits $u_0, u_1, \ldots, u_{N-1}$. Cell D_j will contain the element u_j for $0 \leq j \leq N - 1$.

The loading of the shift register cells is performed in parallel. *All* 211 *cells of the eight driving shift registers and the* 64 *cells of the configuration register are loaded simultaneously in order to avoid the leakage of side channel information.*

After completion of Phase 1, the cells $D_0, D_1, \ldots, D_{21}$ of the NLFSR A (to give a concrete example), will contain the bits $u_0, u_1, \ldots, u_{21}$, in this order.



Phase 2 and 3. Feed-in into each FSR all bits of the interim key $\mathbf{u}_r = (u_0, u_1, \ldots, u_{r-1})$ that have not already been loaded into the register in Phase 1. If the regarded shift register has length N, the bits $u_N, u_{N-1}, \ldots, u_{r-1}$ are fed into the register. The feeding-in of the bits into the FSR's is now performed serial. (See the discussion of the feeding-in procedure for FSR's below.) For instance, NLFSR A will get the inputs u_{22}, u_{23}, $\ldots$, u_{r-1}.

The designation of this step of the algorithm with "Phase 2 and Phase 3" originates in the accompanying computer program. In the program the feeding-in of key bits takes place in "Phase 2", while the feeding-in of IV-vector bits takes place in "Phase 3". Here, however, it is not necessary to separate the two phases.

Phase 4. In each of the eight FSR's A, B, $\ldots$, H, overwrite the content of cell D_0 with the bit 1. This operation makes sure that no driving NLFSR will be loaded into the all-zero state. For the configuration register V, the all-zero state is acceptable. The all-zero state is as good as all other states. Thus the content of cell D_0 of the register V is not changed.

Phase 5. This phase could be called *warm-up phase*. Each of the eight FSR's A, B, $\ldots$, H performs $N + 32$ shifts, where N is the length of the shift register. For instance, NLFSR A performs 54 warm-up shifts. The longest shift register, NLFSR H, performs 63 warm-up shifts. The given number of warm-up shifts for each shift register has the consequence that all eight driving NLFSR's achieve their final states simultaneously. The configuration register V performs 48 warm-up shifts.

If the keystream generator is the reduced KSG, all mentions concerning the configuration register V in the above key-loading algorithm can be ignored. For the full-fledged KSG, the final state of the register V is used to setup the configuration of the output function of the KSG. (In the accompanying C-program, the configuration setup is described under the title "Phase 6".)

Let the final state of the register V be given in accordance with the following assignments:

D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
c_3	c_2	c_1	b_7	b_6	b_5	b_4	b_3	b_2	b_1	a_6	a_5	a_4	a_3	a_2	a_1

D_{31}	D_{30}	D_{29}	D_{28}	D_{27}	D_{26}	D_{25}	D_{24}	D_{23}	D_{22}	D_{21}	D_{20}	D_{19}	D_{18}	D_{17}	D_{16}
e_4	e_3	e_2	e_1	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	c_7	c_6	c_5	c_4

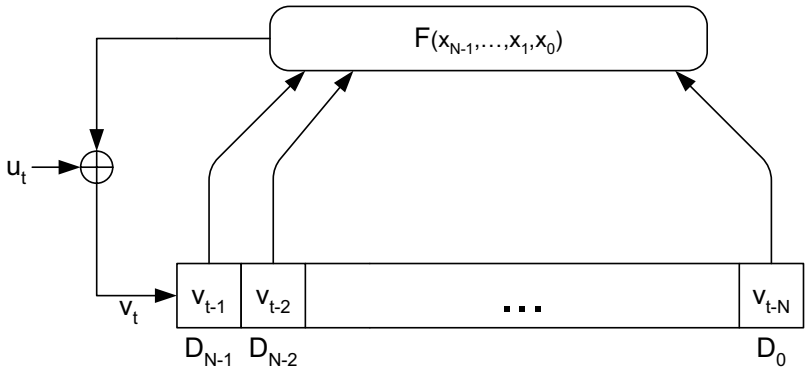
D_{47}	D_{46}	D_{45}	D_{44}	D_{43}	D_{42}	D_{41}	D_{40}	D_{39}	D_{38}	D_{37}	D_{36}	D_{35}	D_{34}	D_{33}	D_{32}
g_3	g_2	g_1	f_9	f_8	f_7	f_6	f_5	f_4	f_3	f_2	f_1	e_8	e_7	e_6	e_5

D_{63}	D_{62}	D_{61}	D_{60}	D_{59}	D_{58}	D_{57}	D_{56}	D_{55}	D_{54}	D_{53}	D_{52}	D_{51}	D_{50}	D_{49}	D_{48}
h_{10}	h_9	h_8	h_7	h_6	h_5	h_4	h_3	h_2	h_1	g_9	g_8	g_7	g_6	g_5	g_4

The filter polynomials $a(x)$, $b(x)$, ..., $h(x)$ which define the linear feedforward functions of the NLFSR's A, B, ..., H, are definded by

$$
\begin{aligned}
a(x) &= a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + 1,\\
b(x) &= b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + 1,\\
c(x) &= c_7x^7 + c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + 1,\\
d(x) &= d_8x^8 + d_7x^7 + d_6x^6 + d_5x^5 + d_4x^4 + d_3x^3 + d_2x^2 + d_1x + 1,\\
e(x) &= e_8x^8 + e_7x^7 + e_6x^6 + e_5x^5 + e_4x^4 + e_3x^3 + e_2x^2 + e_1x + 1,\\
f(x) &= f_9x^9 + f_8x^8 + f_7x^7 + f_6x^6 + f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + 1,\\
g(x) &= g_9x^9 + g_8x^8 + g_7x^7 + g_6x^6 + g_5x^5 + g_4x^4 + g_3x^3 + g_2x^2 + g_1x + 1,\\
h(x) &= h_{10}x^{10} + h_9x^9 + h_8x^8 + h_7x^7 + h_6x^6 + h_5x^5 + h_4x^4 + h_3x^3 + h_2x^2 + h_1x + 1.
\end{aligned}
$$

Here are some remarks to justify the described key-loading algorithm. We first discuss the operation of feeding-in a finite bit string into a binary FSR. Let F be the feedback function of an arbitrary binary FSR of length $N \geq 1$. Let $\mathbf{v} = (v_{-1}, v_{-2}, \ldots, v_{-N}) \in \mathbb{F}_2^N$. We use the vector $\mathbf{v}$ to initialize the FSR defined by the given feedback function F. After loading $\mathbf{v}$ into the register, cell D_{N-j} contains the element v_{-j} for $1 \leq j \leq N$. Consider the r-tuple $\mathbf{u}_r = (u_0, u_1, \ldots, u_{r-1}) \in \mathbb{F}_2^r$, where $r \geq N$. The bits of $\mathbf{u}_r$ are fed into the FSR according to the following figure.



The process can be described by the recursion

$$v_t = u_t + F(v_{t-1}, v_{t-2}, \ldots, v_{t-N}) \quad \text{for } 0 \le t \le r-1.$$

Consider the following mapping.

$$\begin{array}{rcl} \Psi_{F,\mathbf{v}} & : & \mathbb{F}_2^r \rightarrow \mathbb{F}_2^N \\ \mathbf{u}_r = (u_0, u_1, \ldots, u_{r-1}) & \mapsto & \mathbf{v}^N = (v_{r-N}, v_{r-N+1}, \ldots, v_{r-1}) \end{array} \tag{11}$$

Lemma 1. *Let $1 \le N \le r$. For each N-stage FSR defined by some feedback function $F(x_{N-1}, \ldots, x_1, x_0)$, and for each initial state vector $\mathbf{v} \in \mathbb{F}_2^N$, the mapping in (11) has the property*

$$|\{\mathbf{a} \in \mathbb{F}_2^r : \Psi_{F,\mathbf{v}}(\mathbf{a}) = \mathbf{b}\}| = 2^{r-N} \quad \textit{for all } \mathbf{b} \in \mathbb{F}_2^N.$$

Proof. See Rueppel, Lai, and Woollven [18]. □

Lemma 1 implies the following theorem.

Theorem 6. *Consider any one of the eight driving FSR's of the KSG. Let N be the length of that shift register. Then for each of the 2^{N-1} possible states of the shift register, where the content of cell D_0 is 1 and the contents of all other cells are arbitrary, there are exactly 2^{r-N+1} different r-tuples $\mathbf{u}_r = (u_0, u_1, \ldots, u_{r-1})$ that will be mapped onto the given state of the shift register by applying the first two steps of the above key-loading algorithm.*

5 Security properties

Various attacks are known on stream ciphers based on linear feedback shift registers. See for instance [23], [7], [14], [6]. One reason for choosing nonlinear feedback shift registers as building blocks in our stream cipher was to avoid such attacks. The working factor for applying a classical correlation attack is larger than 2^{123} given the lengths of the underlying NLFSR's and since the used Boolean function is 4th order correlation immune. As the linear complexity of the keystream is larger than 2^{85}, an attack based on the Berlekamp-Massey algorithm lies beyond the complexity of a brute force attack.

As already pointed out, the proposed KSG is able to produce 2^{64} translation distinct sequences, each of which has period larger than 2^{207} and linear complexity larger than 2^{85}. We refer to the last chapter of the dissertation of Jansen [15] for security evaluation of keystream generators capable to produce an ensemble of different keystreams.

We hereby state that we are not aware of any hidden weaknesses of the proposed stream cipher.

Furthermore, the stream cipher does not seem to have any weak keys.

6 Parallel implementation

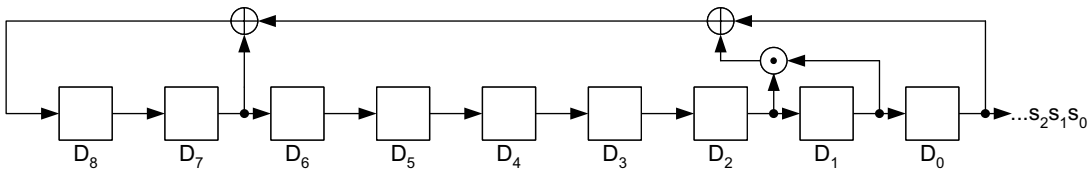
In a straightforward implementation of the KSG, one bit of keystream is generated per clock cycle. In order to increase the speed of generation, it is advisable to implement the underlying FSR's in a certain way described below. In the standard implementation of a FSR, the content of each cell is shifted one position to the right (or left). Using the terminology introduced in Section 2, we can say that the content of cell D_j is shifted into cell D_{j-1}. In a fast implementation of the FSR, the content of cell D_j is shifted into cell D_{j-k}, where k is a positive integer less than N, the length of the FSR.

For practical reasons, we are here mainly interested in step sizes $k = 2, 4$, or 8. Let us assume that the step size k equals 2 for the moment. Then the content of cell D_j is forwarded to cell D_{j-2}. The increase of the step size by a factor of 2 requires the feedback logic to be duplicated. The number of cells (flip-flops), however, remains the same. In total the hardware area increases by the factor 1.15 for the NLFSR's used in our KSG. If the step size $k = 4$ is chosen, the feedback logic must be implemented four times. The number of flip-flops remains the same. The hardware area increases roughly by the factor 1.5. If $k = 8$, the hardware area is increased by the factor 2.5, a relatively small price to pay, considering that now one byte is output from the FSR per clock cycle.

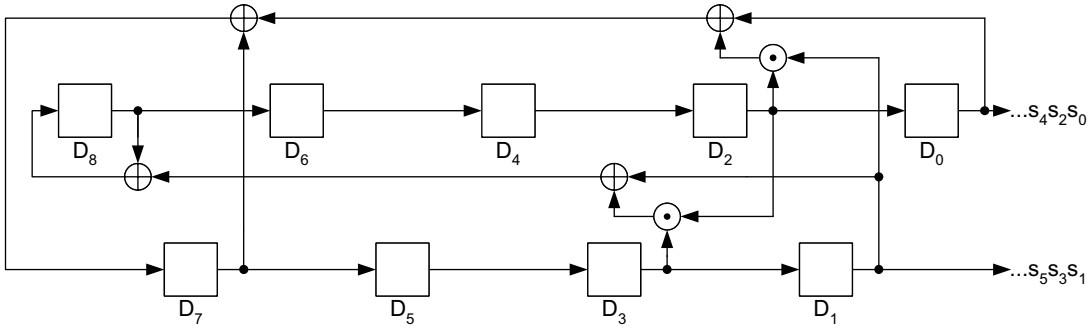
The concept of increasing the performance of a FSR, by choosing larger step sizes, is best illustrated on an example. Consider the 9-stage NLFSR defined by the feedback function

$$F(x_0, x_1, \ldots, x_8) = x_0 + x_7 + x_1x_2.$$

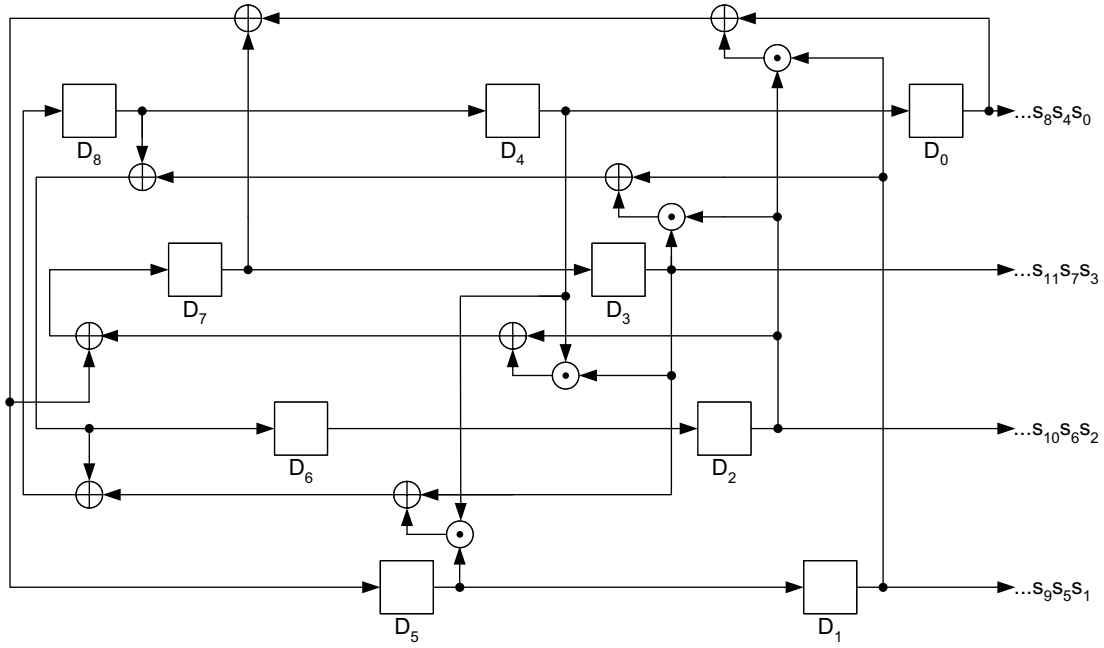
The regular implementation of this FSR is depicted in the following figure.



If we choose step size $k = 2$, the corresponding *fast* implementation looks like this:



The next figure shows the implementation of the FSR for step size $k = 4$.



The increased performance of the underlying FSR's correspond to an equivalent increase of speed of the keystream generation. Thus by implementing the eight driving NLFSR's appropriately, the KSG can produce 2, 4, or 8 bits of keystream per clock cycle.

The case $k = 8$ is probably the one of most practical interest. Think of a eight bit broad bus. At any rate, in its so-called high-speed implementation of the KSG, one byte of keystream is generated per clock cycle. If $\zeta = (z_t)_{t=0}^{\infty}$ is the keystream, then the first byte has the form $(z_0, z_1, z_2, z_3, z_4, z_5, z_6, z_7)$, the second byte is $(z_8, z_9, z_{10}, z_{11}, z_{12}, z_{13}, z_{14}, z_{15})$, and so on. From a cryptographical point of view it is important that in the high-speed implementation of the KSG exactly the same keystream bits are used as before. No new buildings blocks like vectorial Boolean functions are inserted in the design in order to increase speed.

In the high-speed implemention of the KSG not only the encryption rate is increased. The key-loading algorithm is also accelerated.

7 Comparison of hardware designs

In this section we will compare the Achterbahn stream cipher with some stream ciphers from different fields of applications with respect to several figures of merit. A5/1, E0 and RC4 are well known stream ciphers, because they are used in standards. A5/1 is specified for GSM applications, E0 is used in Bluetooth wireless communication, and RC4 in the IEEE 802.11b WLAN standard. We also consider the AES (Rijndael) block cipher standard which can be operated in the output feedback mode (OFB) as a keystream generator.

When evaluating the implementation properties of a certain algorithm in hardware it is important to define the figures of merit precisely. In general the ultimate figure will be a performance/cost ratio. The cost function is not easy to describe, because it depends on several factors which must be weighted differently in different applications. The most important common factors, however, are the *size* of the implementation, the *power consumption*, the *throughput*, the implementation *efficiency*, and the capability of the algorithm to trade off one factor against another. The latter property is often termed *scalability*.

7.1 Area and power

The size of the implementation of an algorithm depends strongly on the minimum feature size of the technology, which is the dimension of the smallest feature actually constructed in the manufacturing process. It also depends on the specific circuit design style, such as CMOS or DCVSL [28], and the number of available metal layers for wire routing. Hence, it is necessary to resort to an approximate, technology and circuit style independent measure. A commonly used measure for the size of a design is the *number of NAND gate equivalents* (GE). This is the area of the circuit implementation divided by the area of the smallest NAND gate in the used standard CMOS cell library. Table 1 shows the sizes of some gates in units of GE for contemporary standard cell library. All CMOS standard cell libraries contain gates with more than two inputs, which generally reduces area, power consumption, and gate propagation delay of a circuit. Examples are AND and OR gates with three or four inputs or XOR gates with three inputs. Obviously, a 4-input NAND gate is smaller than the equivalent circuit built from three 2-input AND gates. Thus the gate equivalent count of a design will always reflect the optimized mixture of available multi-input cells, but not the count of binary operations in the algorithm.

The power consumption of a CMOS design is also related to the gate equivalent count. However, the dynamic power consumption of the implementations of two dif-

gate	size [GE]	gate	size [GE]
2-input NAND	1	2-bit MUX	2.50
2-input AND	1.25	2-input XOR	2.25
3-input AND	1.50	3-input XOR	4.00
4-input AND	1.75	register bit	6.00

Table 1: Typical sizes of some gates in units of NAND gate equivalents (GE).

ferent algorithms which have approximately the same gate count can differ strongly. Power consumption estimations for an algorithm require a detailed analysis of the dynamic switching activity of the gates.

7.2 Throughput

The throughput of a stream cipher is conveniently defined as the average number of output ciphertext bits per second, which is in a synchroneous design equivalent to the average number $\langle \mathcal{N} \rangle$ of output bits per clock cycles times the clock frequency f,

$$\mathcal{P} = \langle \mathcal{N} \rangle f. \tag{12}$$

Hence, the throughput can be either increased by increasing the clock frequency, or by increasing the average number of output bits per clock cycle. The maximum clock frequency a circuit can be operated at is determined by several factors, such as the feature size of the available technology and the operating voltage. Here we can hope, that for the next years Moore's Law will contribute to speed up existing algorithms. However, there is an important factor which is under full control of the algorithm designer. The *number of gate propagation delays* in the longest combinational path of the design, the so-called *critical path*, will ultimately limit the maximally reachable clock frequency. The gate propagation delay is the time required for switching the output of a gate after an input signal has changed. Generally, the sum of the gate propagation delays of the gates in the critical path must be smaller than the cycle time $(1/f)$. An algorithm allowing an implementation with a smaller number of gate propagation delays in the critical path can be operated at higher frequencies. Pipelining techniques can be used to cut down the critical path to some degree. This usually leads to a rapidly increasing number of gates in the design. For the application of a stream cipher in a hardware design it is important that the cipher itself does not contain the critical path of the design. Hence the number of gate propagation delays in the critical path of the cipher (without already implementing pipelining) is an important technology independent figure of merit which determines the maximally reachable throughput.

Another significant factor determining the effective throughput in practice is the overhead time for setting up the encryption in the communication protocol. In the majority of applications the communication is packet-oriented: the message text is split into small packets, which are separately encrypted and transmitted. Typical packet lengths are, e.g., 224 bits in GSM applications, 512 bits for most of the TCP/IP packets in the internet, or up to 2745 bits in the Bluetooth wireless communication standard. To achieve a resynchronization after a transmission error the packets are marked with a frame number and other public information (like time stamps). To prevent the reuse of key material this block of public information, called the initialization value vector, is combined with the secret key. The period of time, starting with the processing of the initialization vector, until the first output of cipher text, is called the *resynchronization time*. Hence, the throughput is reduced by a factor which depends on the resynchronization time and the size of the packets. Consequently, an important figure of merit is a small resynchronization time.

7.3 Implementation efficiency

It is well known that different algorithms can be more or less well suited for a hardware implementation. In order to express, how efficiently a stream cipher design uses the gates to achieve a certain throughput, we introduce the *implementation efficiency* $\mathcal{E}$ of a stream cipher. Normalizing the average number $\mathcal{N}$ of generated key stream bits per cycle by the number of gate equivalents $\mathcal{G}$ of the implementation, we define

$$\mathcal{E} = \frac{\mathcal{N}}{\mathcal{G}} \left[\frac{bit}{cyc \cdot kGE} \right].$$

For convenience the number of gate equivalents is given in units of 1000 GE = 1 kGE. This figure reflects how many kGE are necessary to generate one bit of keystream per cycle on average.

7.4 Scalability

To cover a broad range of possible applications a stream cipher algorithm should be suitable for a very small implementation with small throughput requirements, as well as for high throughput applications, where a larger area and power consumption can be tolerated. Examples for the first kind are mobile and smartcard applications. Future pervasive computing applications, such as RFID tags or sensor networks, will pose even more restrictive area and power constraints on the implementation of cryptographic primitives. Hence, the *minimal implementation size* of an algorithm is certainly an important figure of merit. Examples for applications with an intermediate bandwidth are video signals with serial bit rates between 143.18 Mbps (NTSC standard) and 1.458 Gbps (high definition video standard SMPTE 292M). On the high-end scale there are Gigabit ATM networks and I/O interconnections for distributed computing with bandwidths between 1 Gbps and 30 Gbps (e.g. InfiniBand). In these fields of applications the *maximum throughput* is the important figure of merit.

7.5 Discussion

We implemented the Achterbahn stream cipher in VHDL and synthesized the design for a 0.13μ CMOS standard cell library. The design is configurable for a 1-bit-serial, and 2-bit, 4-bit, or 8-bit parallelization. Additiontally to the full-fledged KSG, we will also consider the reduced version of the KSG described in Section 3.5. In the design of the stream cipher we strived for minimum area without introducing pipelining to increase the maximum frequency. In the first four columns of Table 2 the figures of merit for the eight different implementation versions of Achterbahn are reported. The figures in parenthesis refer to the reduced version of the KSG.

The 1-bit-serial implementation has a comparably small *minimal implementation size*. The size of the design is approximately 3000 GE and the resynchronization time is given by 112 cycles plus the length l of the initial value IV. This implementation is suitable for securing the communication in pervasive computing applications, like RFID tags, for contactless or contact-based smartcard applications, or for wireless communications with moderate throughput requirements. It is also appropriate

for securing serial data links in multi-chip solutions, for masking on-chip signals in security devices, or as a pseudorandom generator. The small number of gate propagation delays in the critical path allows very high target frequencies. In a 0.13μm CMOS technology a frequency of more than 1 GHz can be achieved.

As described in the previous sections the Achterbahn consists only of simple building blocks: feedback shift registers, filter functions, and a Boolean combining function. In Sec. 6 we showed that this construction principle allows a straightforward throughput scaling by applying a parallelization technique. Paradoxically the bit sequential nature of a nonlinear feedback shift register neither prohibits an efficient parallelization nor a pipelined implementation. The implementation with 2-fold parallelization is only 15% larger, whereas the throughput is increased by a factor of 2, and the resynchronization time is also reduced by a factor of 2. The implementation with 8-fold parallelization is 2.5 times larger and the resynchronization time is 8 times smaller (i.e. 22 clock cycles for an initial value of length $l = 64$, respectively 24 clock cycles for an initial value of length $l = 80$). It is important to note that the number of gate delays in the critical path is not increased by the parallelization. Hence the 8-bit parallel design can be operated with the same maximum frequency as the bit serial design - correspondingly the maximum throughput is 8 times as big. In a 0.13μm CMOS technology a throughput of more than 8 Gbps can be achieved. If a higher thoughputs is required one or two pipeline stages in the feedback functions of the NLFSR's, the linear feedforward functions, and the combining function can be introduced. The resulting reduction of the number of gate delays in the critical path will increase the maximum throughput to values between 15 Gbps and 30 Gbps. It is also possible to push the degree of parallelization beyond the step size $k = 8$, say up to $k = 16$, to further increase the throughput. We mention that the step size k is not limited to powers of 2 but k can be any integer value between 1 and 16, say. Thus the throughput can be fine-tuned.

We implemented Achterbahn also on an FPGA (of type Stratix-I). In the full-fledged 8-bit parallel version, the design was operated at a frequency of 240 MHz.

We now compare Achterbahn with three implementations of the AES with 128 bit key length, for which figures are publically available. We assume that the AES is operated in the OFB mode. Hence, it has a state of 128 bits which is updated by repeated encryption operations, starting with a 128 bit IV. The resynchronization time for this configuration is then given by the time required by one encryption operation. For the area comparison with the genuine stream ciphers we have to add to the reported areas the gates necessary to implement the state. According to Table 1 the implementation of the state corresponds to 768 GE. The considered AES implementations are not protected against side channel attacks, such as differential power analysis (DPA). The well-known masking approaches [1, 4] for the nonlinear operations of the S-Boxes lead to additional hardware costs of roughly 200 GE for each S-box [34]. Furthermore, the masking leads to a significant increase of the propagation delay of the critical path. In [34] fifteen additional gate delay times for a specific implementation are reported. To have a basis for comparison with the Achterbahn stream cipher, whose key-loading algorithm is presumably robust against side-channel attacks, the figures for the AES implementations are corrected by the corresponding overhead areas and delay time penalties. In the footnotes of Table 1 the original figures are reported. Masking of the AES is also necessary

during the keystream generation, because the key is inserted in each encryption of the state. It is believed that stream ciphers are in general robust against DPA during keystream generation. The other three reference stream ciphers implementations, E0, A5/1, and RC4, do not contain specific DPA counter measures. Although the key and IV lengths are different for the considered stream ciphers, the comparison gives an indication about strengths and scalability of the different designs. Some designs (E0, A5/1, RC4) have already been attacked successfully.

The proposed stream cipher has a comparably high implementation efficiency. The efficiency grows with the degree of the applied parallelizaton (at least up to $k = 8$). The efficiency of the 8-bit parallel version is approximately two times greater than the efficiency of the high speed AES implementation. At a frequency of approximatly 190 MHz the 8-bit parallel Achterbahn implementation reaches the maximum throughput of 1.5 Gbps of the high speed AES implementation. However, the frequency of the Achterbahn design can still be increased by more than a factor of 5. The introduction of pipeline stages, and the parallelization beyond step size $k = 8$, are options to further increase the throughput.

Table 2: Comparison of key figures for the hardware implementation of several stream ciphers.

	Achterbahn				**AES**			**E0**	**A5/1**	**RC4**
	1	2	4	8	minimal	small	high speed			
reference					[9]	[30, 24]		[16]	[2]	[17]
key/IV size [*bit*]	80/0,8,...,64				128/128			128/74	64/22	8,...,128[e]
state size [*bit*]	275 (211 for reduced version)[a]				128			132	64	2064
word size [*bit*]	1	2	4	8	128			1	1	8
notes	serial	2 bit parallel	4 bit parallel	8 bit parallel	1 S-box 1016 cyc/enc	4 S-boxes 54 cyc/enc	20 S-boxes 11 cyc/enc			3 cyc/byte
resync. time [*cyc*]	$112 + l(IV)$	$56 + \frac{l(IV)}{2}$	$28 + \frac{l(IV)}{4}$	$14 + \frac{l(IV)}{8}$	1016	54	11	239	188	768[f]
design size [*GE*]	2988 (2173)	3427 (2412)	4633 (3113)	7547 (4778)	4563[b]	6966[c]	26105[d]	1902	932	12952
throughput $\langle \mathcal{N} \rangle$ [*bit/cyc*]	1	2	4	8	0.13	2.37	11.64	1	1	2.66
efficiency $\mathcal{E}$ [*bit/cyc/kGE*]	0.33 (0.46)	0.58 (0.83)	0.86 (1.28)	1.06 (1.67)	0.028	0.34	0.45	0.53	1.07	0.21
critical path[g] [*#gate delays*]	8	8	8	8	nn	50[h]	35[k]	2	3	nn
max. frequency	>1 GHz[m]	>1 GHz[m]	>1 GHz[m]	>1 GHz[m]	100 kHz	95 MHz[n]	130 MHz[p]	>3 GHz	>2 GHz	nn
max. throughput	>1 Gbps[m]	>2 Gbps[m]	>4 Gbps[m]	>8 Gbps[m]	12.5 kbps	220 Mbps[n]	1.5 Gbps[p]	>3 Gbps	>3 Gbps	nn

nn: not known, cyc: clock cycle, kGE: 1000 gate equivalents, $l(IV)$: length of IV,
[a]all values for Achterbahn given in parenthesis refer to the reduced keystream generator, see Sec. 3.5,
[b]we added to the reported size of 3595 GE the 128 bit state with 768 GE and 200 GE for DPA protection (masking, see [34]),
[c]we added to the reported size of 5398 GE the 128 bit state with 768 GE and 800 GE for DPA protection (masking, see [34]),
[d]we added to the reported size of 21337 GE the 128 bit state with 768 GE and 4000 GE for DPA protection (masking, see [34]),
[e]the 8-bit word implementation of RC4 allows key+IV sizes from 8 to 2048 bits using various resynchronization schemes,
[f]following the recommendations of dropping the first 512 bytes [12] will increase the setup time to 2304 cycles,
[g]the critical path is calculated assuming the use of multi-input gates,
[h]we estimate a critical path of approx. 35 gate delays and add 15 gate delays for DPA protection (masking, see [34]),
[k]we estimate a critical path of approx. 20 gate delays and add 15 gate delays for DPA protection (masking, see [34]),
[m]for a 0.13μ CMOS process (with a pipelined implementation the throughput can be further increased by a factor of 2 to 4),
[n]estimated reduction of the reported throughput of 311Mbps at 131MHz (for a 0.11μm CMOS process) due to DPA counter measures,
[p]estimated reduction of the reported throughput of 2.6Gbps at 224MHz (for a 0.11μm CMOS process) due to DPA counter measures.

8 Mathematical Background

Let $\mathbb{F}_q$ be the finite field of order q. The set of all sequences of elements of $\mathbb{F}_q$ is denoted by $\mathbb{F}_q^\infty$. If we define for $\sigma = (s_n)_{n=0}^\infty \in \mathbb{F}_q^\infty$ and $\tau = (t_n)_{n=0}^\infty \in \mathbb{F}_q^\infty$ and for $c \in \mathbb{F}_q$ the *sum* $\sigma + \tau = (s_n + t_n)_{n=0}^\infty$ and the *scalar product* $c\sigma = (cs_n)_{n=0}^\infty$, then $\mathbf{F}_q^\infty$ becomes a vector space over $\mathbb{F}_q$. An important linear operator on the vector space $\mathbf{F}_q^\infty$ is the shift operator T, defined by $T\sigma = (s_{n+1})_{n=0}^\infty$ for all sequences $\sigma = (s_n)_{n=0}^\infty \in \mathbb{F}_q^\infty$.

A sequence $\sigma = (s_n)_{n=0}^\infty$ in $\mathbb{F}_q^\infty$ is called *ultimately periodic* if there are integers $n_0 \geq 0$ and $P \geq 1$ such that $s_{n+P} = s_n$ for all $n \geq n_0$. The smallest such integers n_0 and P are called the *length of the preperiod* and the *period* of σ, respectively. We then write $\mathrm{per}(\sigma) = P$. If $s_{n+P} = s_n$ for all $n \geq 0$, then the sequence is called *periodic*. Note that the expression *ultimately periodic* allows the possibility that the sequence is actually periodic.

Any ultimately periodic sequence σ of $\mathbb{F}_q^\infty$ possesses a unique polynomial $m_\sigma \in \mathbb{F}_q[x]$, called the *minimal polynomial* of σ. There are various approaches to the minimal polynomial, one uses ideal theory. If $g \in \mathbb{F}_q[x]$ is a polynomial over $\mathbb{F}_q$, then $g(T)$ defines a linear operator on the vector space $\mathbb{F}_q^\infty$. For instance, let $g(x) = x^3+x+1$. Then $g(T)\sigma = T^3\sigma+T\sigma+\sigma = (s_{n+3}+s_{n+1}+s_n)_{n=0}^\infty$ for all $\sigma = (s_n)_{n=0}^\infty$ of $\mathbb{F}_q^\infty$. We say that a polynomial $g \in \mathbb{F}_q[x]$ *annulates* σ, if $g(T)\sigma$ is the zero sequence $\mathbf{0} = (0, 0, \dots)$. For instance, if $\sigma = (s_n)_{n=0}^\infty \in \mathbb{F}_q^\infty$ is ultimately periodic such that $s_{n+P} = s_n$ for all $n \geq n_0$, then $g(x) = x^{n_0+P} - x^{n_0} \in \mathbb{F}_q[x]$ annulates σ. Thus, for every ultimately periodic sequence $\sigma \in \mathbb{F}_q^\infty$,

$$J_\sigma = \{g \in \mathbb{F}_q[x] : g(T)\sigma = \mathbf{0}\}$$

is a nonzero ideal in the principal ideal domain $\mathbb{F}_q[x]$. The minimal polynomial m_σ of σ is the unique monic polynomial over $\mathbb{F}_q$ generating J_σ, that is,

$$J_\sigma = (m_\sigma) = \{hm_\sigma : h \in \mathbb{F}_q[x]\}.$$

Theorem 7. *Let σ be an ultimately periodic sequence in $\mathbb{F}_q^\infty$. Any polynomial $g \in \mathbb{F}_q[x]$ that annulates σ is called a* characteristic polynomial *of σ. A polynomial $g \in \mathbb{F}_q[x]$ is a characteristic polynomial of σ if and only if m_σ divides g.*

Proof. The assertion follows directly from the fact that the minimal polynomial of σ generates the ideal J_σ and from the definition of the characteristic polynomial of σ. □

Corollary 1. *Let σ be a periodic sequence in $\mathbb{F}_q^\infty$ with minimal polynomial $m_\sigma \in \mathbb{F}_q[x]$. Let $c \in \mathbb{F}_q[x]$ be a characteristic polynomial of σ without multiple roots, and let $f \in \mathbb{F}_q[x]$ be an irreducible factor of c. Then f divides m_σ if and only if the polynomial $g = c/f$ is not a characteristic polynomial of σ.*

Proof. Since the minimal polynomial divides any characteristic polynomial we have $c = bm_\sigma$ for some $b \in \mathbb{F}_q[x]$. Clearly, $g = c/f$ is a multiple of m_σ if and only if f divides b. Thus g is not a multiple of m_σ if and only if f divides m_σ. □

The minimal polynomial m_σ of an ultimately periodic sequence $\sigma \in \mathbb{F}_q^\infty$ contains a lot of information about σ.

1. The multiplicity of the element 0 as a root of m_σ coincides with the length of the preperiod of σ. In particular, σ is periodic if and only if $m_\sigma(0) \neq 0$.

2. The polynomial m_σ is the characteristic polynomial of the shortest linear feedback shift register that can generate σ when appropriately initialized.

3. By definition, the linear complexity of σ is equal to the degree of m_σ.

4. The order of the polynomial m_σ coincides with the period of σ.

For future reference we restate the last property as a theorem.

Theorem 8. *Let σ be an ultimately periodic sequence in $\mathbb{F}_q^\infty$ with minimal polynomial $m_\sigma \in \mathbb{F}_q[x]$. Then the period of σ is equal to the order of the minimal polynomial of σ, denoted by* $\operatorname{ord}(m_\sigma)$.

Proof. See Lidl and Niederreiter [20, Theorem 8.44]. □

The order of a polynomial f is sometimes also called the *period* of f or the *exponent* of f. We quote another theorem from [20] concerning the order of a polynomial.

Theorem 9. *Let $g_1, \ldots, g_k$ be pairwise relatively prime nonzero polynomials over $\mathbb{F}_q$, and let $f = g_1 \cdots g_k$. Then*

$$\operatorname{ord}(f) = \operatorname{lcm}\left(\operatorname{ord}(g_1), \ldots, \operatorname{ord}(g_k)\right).$$

Proof. See Lidl and Niederreiter [20, Theorem 3.9]. □

Another interesting approach to the minimal polynomial of an ultimately periodic sequence $\sigma \in \mathbb{F}_q^\infty$ makes use of generating functions. Following Niederreiter [26], we assign to an arbitrary sequence $\sigma = (s_n)_{n=0}^\infty$ of elements of $\mathbb{F}_q$ the generating function

$$G_\sigma(x) = s_0 x^{-1} + s_1 x^{-2} + s_2 x^{-3} + \cdots,$$

regarded as an element of the field $\mathbb{F}_q((x^{-1}))$ of formal Laurent series in the indeterminate x^{-1}. The field $\mathbb{F}_q((x^{-1}))$ contains the field $\mathbb{F}_q(x)$ of rational functions as a subfield. A sequence $\sigma \in \mathbb{F}_q^\infty$ is ultimately periodic if and only if the associated generating function G_σ belongs to the subfield $\mathbb{F}_q(x)$.

Theorem 10. *Let $m \in \mathbb{F}_q[x]$ be a monic polynomial, and let $\sigma = (s_n)_{n=0}^\infty$ be a sequence of elements of $\mathbb{F}_q$. Then σ is ultimately periodic and m is the minimal polynomial of σ if and only if*

$$\sum_{n=0}^{\infty} s_n x^{-n-1} = \frac{h(x)}{m(x)}$$

with a polynomial $h \in \mathbb{F}_q[x]$ with $\deg(h) < \deg(m)$ and $\gcd(h, m) = 1$.

Proof. See Niederreiter [26], [20, p. 218]. □

Theorem 11. *For each $j = 1, \ldots, k$, let σ_j be an ultimately periodic sequence in $\mathbb{F}_q^\infty$ with minimal polynomial $m_j \in \mathbb{F}_q[x]$. If the polynomials $m_1, \ldots, m_k$ are pairwise relatively prime, then the minimal polynomial of the sum $\sigma = \sigma_1 + \cdots + \sigma_k$ is equal to the product $m_1 \cdots m_k$.*

Conversely, let σ be an ultimately periodic sequence in $\mathbb{F}_q^\infty$ whose minimal polynomial $m \in \mathbb{F}_q[x]$ is the product of pairwise relatively prime monic polynomials $m_1, \ldots, m_k \in \mathbb{F}_q[x]$. Then, for each $j = 1, \ldots, k$, there exists a uniquely determined ultimately periodic sequence σ_j with minimal polynomial $m_j \in \mathbb{F}_q[x]$ such that $\sigma = \sigma_1 + \cdots + \sigma_k$.

Proof. A proof of the first part of the theorem can be found on page 426 in [20]. To prove the second part, let $h/m \in \mathbb{F}_q(x)$ be the generating function of σ in the sense of Theorem 10. Let

$$\frac{h}{m} = \frac{h_1}{m_1} + \cdots + \frac{h_k}{m_k} \tag{13}$$

be the partial fraction decomposition of h/m. By Theorem 10, $\deg(h) < \deg(m)$ and $\gcd(h, m) = 1$. This implies $\deg(h_j) < \deg(m_j)$ and $\gcd(h_j, m_j) = 1$ for $1 \leq j \leq k$. The rational functions h_j/m_j correspond to uniquely determined ultimately periodic sequences $\sigma_j \in \mathbb{F}_q^\infty$ with minimal polynomials m_j according to Theorem 10. Equation (13) implies that $\sigma = \sigma_1 + \cdots + \sigma_k$. □

Let σ be an ultimately periodic sequence of $\mathbb{F}_q^\infty$, and let f be a nonconstant polynomial over $\mathbb{F}_q$. We call the sequence $\tau = f(T)\sigma$ a *linearly filtered* sequence derived from σ. The polynomial f is called the *filter polynomial*. Note that the sequence τ is a linear combination of shifted versions of σ.

Theorem 12. *Let σ be an ultimately periodic sequence of elements of $\mathbb{F}_q$ with minimal polynomial $m_\sigma \in \mathbb{F}_q[x]$, and let f be a nonzero polynomial over $\mathbb{F}_q$. Then the sequence $\tau = f(T)\sigma$ is again ultimately periodic and has minimal polynomial*

$$m_\tau = \frac{m_\sigma}{\gcd(m_\sigma, f)}.$$

Proof. See Niederreiter [25], or Blackburn [3], or Göttfert [13, Chap. 2]. □

If $\sigma \in \mathbb{F}_q^\infty$ is periodic and $f \in \mathbb{F}_q[x]$ is arbitrary, then $\tau = f(T)\sigma$ is also periodic. This is trivial if f is the zero polynomial. Otherwise, recall that σ is periodic if and only if $m_\sigma(0) \neq 0$. Since m_τ, the minimal polynomial of τ, divides m_σ, by Theorem 12, we have $m_\tau(0) \neq 0$, so that τ is periodic.

A periodic sequence σ in $\mathbb{F}_q^\infty$ of span N and period $q^N - j$, where j is a small nonnegative integer (say $j = 0$, 1, or 2) has automatically almost ideal k-tuple distribution for all $k \leq N$. (Such sequences can always be produced by suitable N-stage feedback shift registers over $\mathbb{F}_q$.) If the sequence is linearly filtered, using a filter polynomial f of relatively small degree (compared to N), then the resulting sequence will still have good distribution properties. On the other hand, if the linear complexity of the original sequence is large enough, the application of the operator $f(T)$ to the original sequence, where $d = \deg(f) < N$, cannot effect the period and will decrease the linear complexity at most by d according to Theorem 12. For more information on linear filtering see [11].

We recall some results from Selmer [31, Chap. 4], and Zierler and Mills [36]. Let $f, g, \ldots, h \in \mathbb{F}_q[x]$ be nonconstant polynomials without multiple roots in their respective splitting fields over $\mathbb{F}_q$ and with nonzero constant terms. (The latter restriction is meant to exclude the irreducible polynomial $p(x) = x$.) Then $f \vee g \vee \cdots \vee h$ is defined to be the monic polynomial whose roots are the distinct products $\alpha\beta\cdots\gamma$, where α is a root of f, β a root of g, and γ a root of h. The polynomial $f \vee g \vee \cdots \vee h$ is again a polynomial over the ground field $\mathbb{F}_q$. This follows from the fact that all conjugates (over $\mathbb{F}_q$) of a root of $f \vee g \vee \cdots \vee h$ are roots of $f \vee g \vee \cdots \vee h$. The importance of the polynomial $f \vee g \vee \cdots \vee h$ stems from the fact that if $\sigma = (s_n)_{n=0}^{\infty}$, $\tau = (t_n)_{n=0}^{\infty}$, $\ldots$, $\upsilon = (u_n)_{n=0}^{\infty}$ are periodic sequences of elements of $\mathbb{F}_q$ with characteristic polynomials $f, g, \ldots, h$, respectively, then $f \vee g \vee \cdots \vee h$ is a characteristic polynomial of the product sequence $\sigma\tau\cdots\upsilon = (s_n t_n \cdots u_n)_{n=0}^{\infty}$.

Theorem 13. *Let $f, g, \ldots, h \in \mathbb{F}_q[x]$ be polynomials over $\mathbb{F}_q$ without multiple roots and with nonzero constant terms. The polynomial $f \vee g \vee \cdots \vee h \in \mathbb{F}_q[x]$ is irreducible if and only if the polynomials $f, g, \ldots, h$ are all irreducible and of pairwise relatively prime degrees. In this case*

$$\deg(f \vee g \vee \cdots \vee h) = \deg(f)\deg(g)\cdots\deg(h). \tag{14}$$

If $\sigma, \tau, \ldots, \upsilon$ are periodic sequences in $\mathbb{F}_q$ with irreducible minimal polynomials $f, g, \ldots, h \in \mathbb{F}_q[x]$ of pairwise relatively prime degrees and with $f(0)g(0)\cdots h(0) \neq 0$, then $f \vee g \vee \cdots \vee h$ is the minimal polynomial of the product sequence $\sigma\tau\cdots\upsilon$.

Proof. See Selmer [31, Chap. 4]. □

Example 3. Consider the binary irreducible polynomials $f(x) = x^2 + x + 1$ and $g(x) = x^3 + x + 1$. Over the splitting field $\mathbb{F}_{64}$ of $fg \in \mathbb{F}_2[x]$ we can write

$$\begin{aligned} f(x) &= x^2 + x + 1 = (x - \alpha)(x - \alpha^2) = x^2 + (\alpha + \alpha^2)x + \alpha^3, \\ g(x) &= x^3 + x + 1 = (x - \beta)(x - \beta^2)(x - \beta^4). \end{aligned}$$

In particular, for the root $\alpha \in \mathbb{F}_4$ of f we have

$$\alpha + \alpha^2 = 1 \quad \text{and} \quad \alpha^3 = 1.$$

Using these identities, we obtain

$$\begin{aligned} (f \vee g)(x) &= (x - \alpha\beta)(x - \alpha\beta^2)(x - \alpha\beta^4)(x - \alpha^2\beta)(x - \alpha^2\beta^2)(x - \alpha^2\beta^4) \\ &= \frac{1}{\alpha^3}\left(\frac{x}{\alpha} - \beta\right)\left(\frac{x}{\alpha} - \beta^2\right)\left(\frac{x}{\alpha} - \beta^4\right)\frac{1}{\alpha^6}\left(\frac{x}{\alpha^2} - \beta\right)\left(\frac{x}{\alpha^2} - \beta^2\right)\left(\frac{x}{\alpha^2} - \beta^4\right) \\ &= g\left(\frac{x}{\alpha}\right) g\left(\frac{x}{\alpha^2}\right) = \left(x^3 + \alpha^2 x + 1\right)\left(x^3 + \alpha x + 1\right) \\ &= x^6 + x^4 + x^2 + x + 1. \end{aligned}$$

Let $\sigma = (s_n)_{n=0}^{\infty}$ be the periodic binary sequence defined by the linear recursion $s_{n+2} = s_{n+1} + s_n$ for all $n \geq 0$ and the initial values $s_0 = 0$ and $s_1 = 1$. Similiarly, let $\tau = (t_n)_{n=0}^{\infty}$ be the periodic binary sequence defined by $t_{n+3} = t_{n+1} + t_n$ for all

$n \geq 0$ and $t_0 = 1$ and $t_1 = t_2 = 0$. Then σ has minimal polynomial f and period $\operatorname{ord}(f) = 3$ whereas the sequence τ has minimal polynomial g and period $\operatorname{ord}(g) = 7$.

$$\sigma = 0\,1\,1\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\,1\,1\,\ldots$$
$$\tau = 1\,0\,0\,1\,0\,1\,1\,1\,0\,0\,1\,0\,1\,1\,1\,0\,0\,1\,0\,1\,1\,\ldots$$
$$\sigma\tau = 0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,1\,0\,0\,1\,1\,0\,0\,1\,0\,1\,1\,\ldots$$

One readily checks that the product sequence $\sigma\tau = (w_n)_{n=0}^{\infty}$ satisfies the 6th-order linear recursion

$$w_{n+6} = w_{n+4} + w_{n+2} + w_{n+1} + w_n \qquad \text{for all } n \geq 0.$$

Note that the first five terms of $(w_n)_{n=0}^{\infty}$ are zero, so that the sequence $\sigma\tau$ cannot satisfy any shorter linear recursion. Thus $(f \vee g)(x) = x^6 + x^4 + x^2 + x + 1$ is the minimal polynomial of $\sigma\tau$. □

The following Lemma will be needed in the sequel. We skip its rather simple proof here.

Lemma 2. *Let a and b be positive integers. Then*

$$\gcd(2^a - 1, 2^b - 1) = 2^{\gcd(a,b)} - 1. \tag{15}$$

In particular, $2^a - 1$ and $2^b - 1$ are relatively prime if and only if a and b are.

From now on we restrict ourselves to the binary case $q = 2$.

Theorem 14. *Let $f, g, \ldots, h \in \mathbb{F}_2[x]$ be irreducible binary polynomials without multiple roots, of pairwise relatively prime degrees, and with nonzero constant terms. Then*

$$\operatorname{ord}(f \vee g \vee \cdots \vee h) = \operatorname{ord}(f)\operatorname{ord}(g)\cdots\operatorname{ord}(h). \tag{16}$$

Proof. It suffices to prove the assertion for two polynomials $f, g \in \mathbb{F}_2[x]$. Let $\deg(f) = a$, and let $\alpha \in \mathbb{F}_{2^a}$ be a root of f. Since f is irreducible, $\operatorname{ord}(f)$ coincides with the order of α as an element of the group $\mathbb{F}_{2^a}^*$, the multiplicative group formed by all nonzero elements of $\mathbb{F}_{2^a}$. The order of any element of $\mathbb{F}_{2^a}^*$ divides the order of the group $\mathbb{F}_{2^a}^*$ which is $2^a - 1$. Let $\deg(g) = b$, and let $\beta \in \mathbb{F}_{2^b}$ be a root of g. Then, by the same argument, we conclude that the order of β in $\mathbb{F}_{2^b}^*$ is equal to $\operatorname{ord}(g)$ and both numbers divide $2^b - 1$. By hypothesis, the greatest common divisor $\gcd(a, b)$ of a and b is 1, so that, by Lemma 2, $\gcd(2^a - 1, 2^b - 1) = 1$. It follows that α and β are elements of relatively prime orders in the group $\mathbb{F}_{2^{ab}}^*$. By Theorem 13, the polynomial $f \vee g$ is irreducible over $\mathbb{F}_2$. Thus the order of the polynomial $f \vee g$ is equal to the order of $\gamma = \alpha\beta$ in $\mathbb{F}_{2^{ab}}^*$. It is well known (see e.g. McEliece [21, p. 38]) that the order of the product of two elements in a commutative group is the product of the orders of the two elements if these orders are relatively prime. Hence $\operatorname{ord}(f \vee g) = \operatorname{ord}(\alpha\beta) = \operatorname{ord}(\alpha)\operatorname{ord}(\beta) = \operatorname{ord}(f)\operatorname{ord}(g)$. □

For the binary polynomials $f(x) = x^2 + x + 1$, $g(x) = x^3 + x + 1$, and $h(x) = (f \vee g)(x) = x^6 + x^4 + x^2 + x + 1$ appearing in Example 1, we find $\operatorname{ord}(f) = 3$, $\operatorname{ord}(g) = 7$, and $\operatorname{ord}(h) = 21$. Thus $21 = \operatorname{ord}(f \vee g) = \operatorname{ord}(f)\operatorname{ord}(g) = 3 \cdot 7$, in agreement with equation (16). The imposed restriction in Theorem 14 to the binary field $\mathbb{F}_2$ is necessary. Consider, for instance, the two polynomials $f(x) = x+1 = \mathbb{F}_3[x]$ and $g(x) = x^2+1 \in \mathbb{F}_3[x]$ over the finite field of order 3. Then $\operatorname{ord}(f) = 2$, $\operatorname{ord}(g) = 4$, and $f \vee g = g$, and equation (16) does not hold in this case.

Theorem 15. *Let N be a positive integer, and let $\sigma = (s_n)_{n=0}^{\infty}$ be a binary periodic sequence with period $P = 2^N - 1$. The canonical factorization of the minimal polynomial $m_\sigma \in \mathbb{F}_2[x]$ of σ over $\mathbb{F}_2$ consists of distinct irreducible polynomials whose degrees all divide N. In particular, m_σ contains no repeated factors.*

Proof. Since σ has period P, the polynomial $c(x) = x^P - 1 \in \mathbb{F}_2[x]$ is a characteristic polynomial of σ. By Theorem 7, $m_\sigma(x)$ divides $c(x)$. Consequently, $m_\sigma(x)$ divides $x^{2^N} - x$, which is the product of all irreducible binary polynomials whose degrees divide N (see [20, Theorem 3.20]). □

Theorem 16. *Let $N \geq 1$, and let $\sigma = (s_n)_{n=0}^{\infty}$ be a binary periodic sequence of period $P = 2^N - 1$ and of span N. If the zero vector $\mathbf{0} \in \mathbb{F}_2^N$ does not occur among the N-tuples $\mathbf{s}_n = (s_n, s_{n+1}, \ldots, s_{n+N-1})$, $0 \leq n \leq P-1$, then $x - 1$ does not divide the minimal polynomial m_σ of σ.*

Proof. The N-tuples $\mathbf{s}_n$, $0 \leq n \leq P-1$, run through all nonzero vectors of $\mathbb{F}_2^N$. Therefore, the element 1 occurs exactly 2^{N-1} times among the first coordinates of these N-tuples. Thus

$$s_0 + s_1 + \cdots + s_{P-1} = 0.$$

Since σ is periodic with period P, we get

$$s_n + s_{n+1} + \ldots + s_{n+P-1} = 0 \qquad \text{for all } \; n \geq 0,$$

which means that

$$c(x) = x^{P-1} + x^{P-2} + \cdots + x + 1 \in \mathbb{F}_2[x]$$

is a characteristic polynomial of σ. Since $c(1) \neq 0$, the polynomial $c(x)$ is not divisible by $x-1$, nor is the minimal polynomial $m_\sigma(x)$ which is a divisor of $c(x)$. □

Theorem 17. *Let $S, T, \ldots, U$ be pairwise relatively prime integers greater than 1. Let $\sigma = (s_n)_{n=0}^{\infty}$, $\tau = (t_n)_{n=0}^{\infty}$, $\ldots$, $\upsilon = (u_n)_{n=0}^{\infty}$ be binary periodic sequences of periods $\mathrm{per}(\sigma) = 2^S - 1$, $\mathrm{per}(\tau) = 2^T - 1$, $\ldots$, $\mathrm{per}(\upsilon) = 2^U - 1$, respectively. Assume that the canonical factorizations over $\mathbb{F}_2$ of the minimal polynomials of $\sigma, \tau, \ldots, \upsilon$ are*

$$m_\sigma = \prod_{i=1}^{s} f_i, \qquad m_\tau = \prod_{j=1}^{t} g_j, \qquad \ldots, \qquad m_\upsilon = \prod_{k=1}^{u} h_k. \tag{17}$$

Then the minimal polynomial of the product sequence $\sigma\tau\cdots\upsilon = (s_n t_n \cdots u_n)_{n=0}^{\infty}$ is given by

$$m_{\sigma\tau\cdots\upsilon} = \prod_{i=1}^{s}\prod_{j=1}^{t}\cdots\prod_{k=1}^{u}(f_i \vee g_j \vee \cdots \vee h_k). \tag{18}$$

In fact, (18) *represents the canonical factorization of the minimal polynomial of $\sigma\tau\cdots\upsilon$ over $\mathbb{F}_2$.*

Proof. It suffices to carry out the details of the proof for the product of two such sequences σ and τ. The general statement then follows by induction. Consider the canonical factorization of the minimal polynomials m_σ and m_τ in (17). By Theorem 15, the irreducible polynomials $f_1, \ldots, f_s \in \mathbb{F}_2[x]$ are distinct and $\deg(f_i)$

divides S for $1 \leq i \leq s$. Similiarly, the irreducible polynomials $g_1, \ldots, g_t$ are distinct and $\deg(g_j)$ divides T for $1 \leq j \leq t$. Since the sequences σ and τ are periodic, their minimal polynomials m_σ and m_τ are not divisible by x. Thus, the first-degree irreducible polynomial $p(x) = x$ does not occur among the polynomials $f_1, \ldots, f_s$ and $g_1, \ldots, g_t$.

By Theorem 11, the sequences σ and τ possess unique representations

$$\sigma = \sum_{i=1}^{s} \sigma_i \quad \text{and} \quad \tau = \sum_{j=1}^{t} \tau_j,$$

where σ_i is a binary periodic sequence with minimal polynomial f_i for $1 \leq i \leq s$, and τ_j is a binary periodic sequence with minimal polynomial g_j for $1 \leq j \leq t$. It follows that

$$\sigma\tau = \sum_{i=1}^{s} \sum_{j=1}^{t} \sigma_i \tau_j.$$

By hypothesis, $\gcd(S, T) = 1$. It follows that for each $i \in \{1, \ldots, s\}$ and $j \in \{1, \ldots, t\}$, the corresponding irreducible polynomials f_i and g_j have relatively prime degrees. Invoking Theorem 13, we conclude that for each $i \in \{1, \ldots, s\}$ and $j \in \{1, \ldots, t\}$, the sequence $\sigma_i\tau_j$ has the irreducible minimal polynomial $f_i \vee g_j \in \mathbb{F}_2[x]$.

As will be shown below, the irreducible polynomials $f_i \vee g_j$, $1 \leq i \leq s$, $1 \leq j \leq t$, are distinct. Another application of Theorem 11 therefore shows that the minimal polynomial of $\sigma\tau$ has the form

$$m_{\sigma\tau} = \prod_{i=1}^{s} \prod_{j=1}^{t} (f_i \vee g_j). \tag{19}$$

It remains to show that the polynomials $f_i \vee g_j$, $1 \leq i \leq s$, $1 \leq j \leq t$, are distinct. To see this, let f_i and f_i' be any two factors from the canonical factorization of m_σ, and let g_j and g_j' be any two factors from the canonical factorization of m_τ. Assume to the contrary that the two irreducible polynomials $f_i \vee g_j$ and $f_i' \vee g_j'$ are equal. Note that two irreducible polynomials over the finite field $\mathbb{F}_q$ are equal if and only if they have a common root (in some extension field of $\mathbb{F}_q$). Let γ be a common root of $f_i \vee g_j$ and $f_i' \vee g_j'$. Then we can write γ in the form

$$\gamma = \alpha\beta = \alpha'\beta', \tag{20}$$

where α, β, α', and β' are roots of the polynomials f_i, g_j, f_i', and g_j', respectively. Since α is a root of the irreducible polynomial f_i, we have $\alpha \in \mathbb{F}_{2^{\deg(f_i)}}$, which is a subfield of $\mathbb{F}_{2^S}$, as $\deg(f_i)$ divides S. Similiarly, we conclude that $\alpha' \in \mathbb{F}_{2^S}$ and $\beta, \beta' \in \mathbb{F}_{2^T}$. From (20) we obtain

$$\frac{\alpha}{\alpha'} = \frac{\beta'}{\beta}. \tag{21}$$

Clearly, $\alpha/\alpha' \in \mathbb{F}_{2^S}$ and $\beta'/\beta \in \mathbb{F}_{2^T}$. Since S and T are relatively prime, we have $\mathbb{F}_{2^S} \cap \mathbb{F}_{2^T} = \mathbb{F}_2$, so that both sides of (21) must be equal to 1. Hence $\alpha = \alpha'$ and $\beta = \beta'$. This, however, implies $f_i = f_i'$ and $g_j = g_j'$. □

Corollary 2. *Let* $\sigma = (s_n)_{n=0}^{\infty}$, $\tau = (t_n)_{n=0}^{\infty}$, $\ldots$, $\upsilon = (u_n)_{n=0}^{\infty}$ *be binary periodic sequences of periods* $\mathrm{per}(\sigma) = 2^S - 1$, $\mathrm{per}(\tau) = 2^T - 1$, $\ldots$, $\mathrm{per}(\upsilon) = 2^U - 1$, *and linear complexities* $L(\sigma), L(\tau), \ldots, L(\upsilon)$ *respectively. If the integers* $S, T, \ldots, U$ *be pairwise relatively prime and greater than* 1, *then the product sequence* $\sigma\tau\cdots\upsilon = (s_n t_n \cdots u_n)_{n=0}^{\infty}$ *has linear complexity*

$$L(\sigma\tau\cdots\upsilon) = L(\sigma)L(\tau)\cdots L(\upsilon), \tag{22}$$

and period

$$\mathrm{per}(\sigma\tau\cdots\upsilon) = \left(2^S - 1\right)\left(2^T - 1\right)\cdots\left(2^U - 1\right). \tag{23}$$

Proof. Let the minimal polynomials of $\sigma, \tau, \ldots, \upsilon$ be given by the expressions in (17). Then, by Theorem 17 and equation (14), we obtain

$$\begin{aligned}
L(\sigma\tau\cdots\upsilon) = \deg(m_{\sigma\tau\cdots\upsilon}) &= \sum_{i=1}^{s}\sum_{j=1}^{t}\cdots\sum_{k=1}^{u}\deg(f_i \vee g_j \vee \cdots \vee h_k) \\
&= \sum_{i=1}^{s}\sum_{j=1}^{t}\cdots\sum_{k=1}^{u}\deg(f_i)\deg(g_j)\cdots\deg(h_k)) \\
&= \left(\sum_{i=1}^{s}\deg(f_i)\right)\left(\sum_{j=1}^{t}\deg(g_j)\right)\cdots\left(\sum_{k=1}^{u}\deg(h_k)\right) \\
&= L(\sigma)L(\tau)\cdots L(\upsilon).
\end{aligned}$$

This proves equation (22). For the proof of (23), recall that over an arbitrary finite field $\mathbb{F}_q$, the period of a periodic sequence of field elements is equal to the order of the sequence's minimal polynomial (Theorem 8). Using Theorems 17, 9, and 14, we get

$$\begin{aligned}
&\mathrm{per}(\sigma\tau\cdots\upsilon) = \mathrm{ord}(m_{\sigma\tau\cdots\upsilon}) \\
&\quad = \mathrm{lcm}\{\mathrm{ord}(f_i \vee g_j \vee \cdots \vee h_k) : 1 \le i \le s,\ 1 \le j \le t, \ldots, 1 \le k \le u\} \\
&\quad = \mathrm{lcm}\{\mathrm{ord}(f_i)\,\mathrm{ord}(g_j)\cdots\mathrm{ord}(h_k) : 1 \le i \le s,\ 1 \le j \le t, \ldots, 1 \le k \le u\} \\
&\quad = \mathrm{lcm}\{\mathrm{ord}(f_i) : 1 \le i \le s\}\,\mathrm{lcm}\{\mathrm{ord}(g_j) : 1 \le j \le t\}\cdots\mathrm{lcm}\{\mathrm{ord}(h_k) : 1 \le k \le u\} \\
&\quad = \mathrm{ord}(m_\sigma)\,\mathrm{ord}(m_\tau)\cdots\mathrm{ord}(m_\upsilon) \\
&\quad = \mathrm{per}(\sigma)\,\mathrm{per}(\tau)\cdots\mathrm{per}(\upsilon) \\
&\quad = \left(2^S - 1\right)\left(2^T - 1\right)\cdots\left(2^U - 1\right).
\end{aligned}$$

To justify the fourth equality in the above argument we note that $\mathrm{ord}(f_i)$ divides $2^S - 1$, $\mathrm{ord}(g_j)$ divides $2^T - 1$, and $\mathrm{ord}(h_k)$ divides $2^U - 1$ for all $1 \le i \le s$, $1 \le j \le t$, and $1 \le k \le u$, and that the numbers $2^S - 1$, $2^T - 1$, $\ldots$, $2^U - 1$ are pairwise relatively prime according to Lemma 2. □

Example 4. Consider the 4-stage NLFSR with feedback function $F(x_0, x_1, x_2, x_3) = x_0 + x_1 + x_2 + x_1x_2$, and the 5-stage NLFSR defined by the feedback function $G(x_0, x_1, x_2, x_3, x_4) = x_0 + x_1 + x_3 + x_1x_3$. Using any nonzero initial state vector of $\mathbb{F}_2^4$, the first feedback shift register will produce a periodic binary sequence σ of

period $\mathrm{per}(\sigma) = 15$ and linear complexity $L(\sigma) = 14$. For instance, if we use the initial state vector $(0, 0, 0, 1)$, we get

$$\sigma = (0\,0\,0\,1\,0\,1\,1\,0\,1\,0\,0\,1\,1\,1\,1)^\infty$$

The minimal polynomial of σ is

$$m_\sigma(x) = x^{14} + x^{13} + \cdots + x + 1 = f_1(x)f_2(x)f_3(x)f_4(x),$$

where $f_1(x) = x^2 + x + 1$, $f_2(x) = x^4 + x^3 + x^2 + x + 1$, $f_3(x) = x^4 + x + 1$, $f_4 = x^4 + x^3 + 1$. Similiarly, if we initialize the second shift register with the nonzero vector $(0, 0, 0, 0, 1)$, it generates the periodic sequence

$$\tau = (0\,0\,0\,0\,1\,0\,1\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,0\,0\,1\,0\,0\,0\,1\,1\,1\,1\,1)^\infty$$

of period $\mathrm{per}(\tau) = 31$ and linear complexity $L(\tau) = 30$. The corresponding minimal polynomial is

$$m_\tau(x) = x^{30} + x^{29} + \cdots + x + 1 = g_1(x)g_2(x)g_3(x)g_4(x)g_5(x)g_6(x),$$

where $g_1, \ldots, g_6$ are the six irreducible (and primitive) polynomials in $\mathbb{F}_2[x]$ of degree 5. The product sequence $\sigma\tau$ has period $\mathrm{per}(\sigma\tau) = 15 \cdot 31 = 465$ and linear complexity $L(\sigma\tau) = 14 \cdot 30 = 420$. The canonical factorization of the minimal polynomial of $\sigma\tau$ consists of $4 \cdot 6 = 24$ irreducible binary polynomials. Of these polynomials six have degree 10 and order 93, six have degree 20 and order 155, and twelve have degree 20 and order 465.

The irreducible factors:

	g_1	g_2	g_3	g_4	g_5	g_6
f_1	$f_1 \vee g_1$	$f_1 \vee g_2$	$f_1 \vee g_3$	$f_1 \vee g_4$	$f_1 \vee g_5$	$f_1 \vee g_6$
f_2	$f_2 \vee g_1$	$f_2 \vee g_2$	$f_2 \vee g_3$	$f_2 \vee g_4$	$f_2 \vee g_5$	$f_2 \vee g_6$
f_3	$f_3 \vee g_1$	$f_3 \vee g_2$	$f_3 \vee g_3$	$f_3 \vee g_4$	$f_3 \vee g_5$	$f_3 \vee g_6$
f_4	$f_4 \vee g_1$	$f_4 \vee g_2$	$f_4 \vee g_3$	$f_4 \vee g_4$	$f_4 \vee g_5$	$f_4 \vee g_6$

The degrees of the irreducible factors:

	5	5	5	5	5	5
2	10	10	10	10	10	10
4	20	20	20	20	20	20
4	20	20	20	20	20	20
4	20	20	20	20	20	20

The orders of the irreducible factors:

	31	31	31	31	31	31
3	93	93	93	93	93	93
5	155	155	155	155	155	155
15	465	465	465	465	465	465
15	465	465	465	465	465	465

Note that there are exactly $\varphi(93)/10 = 6$ irreducible binary polynomials of degree 10 and order 93, $\varphi(155)/20 = 6$ irreducible binary polynomials of degree 20 and order 155, and $\varphi(465)/20 = 12$ irreducible binary polynomials of degree 20 and order 465 (compare [20, Theorem 3.5]). All these polynomials appear in the canonical factorization of $m_{\sigma\tau}$. This is a consequence of the fact that the sequences σ and τ have maximum linear complexities $L(\sigma) = 2^4 - 2 = 14$ and $L(\tau) = 2^5 - 2 = 30$, respectively. □

9 Conclusion

We proposed a new additive stream cipher called ACHTERBAHN. The keystream generator is based on eight binary primitive NLFSR's that can produce sequences of period $2^N - 1$, where N is the length of the shift register. The NLFSR's are endowed with configurable linear feedforward functions. The output sequences of the feedforward functions are combined by a balanced 4th order correlation immune Boolean function. The keystream generator is able to produce an ensemble of 2^{64} translation distinct binary sequences. Each sequence has a period larger than 2^{207} and linear complexity larger than 2^{85}. Due to fast hardware implementations of the underlying feedback shift registers, the keystream generator permits a high-speed implementation in which one byte of keystream is generated per clock cycle. The eight bits generated in each clock cycle can be used to encrypt all eight lines of a bus in real time.

It was this latter feature that gave the stream cipher its name. There is yet another more subtle reason for choosing the name Achterbahn for the proposed stream cipher. The reason has something to do with the with human psychology and human physiology. If people ride a state of the art roller coaster—the american expression for the german word Achterbahn—some become addicted to it while others get sick. We hope that the same will happen to the cryptographer who studies our stream cipher and to the cryptanalyst who aims to break it.

Acknowledgement

We would like to thank Gerd Dirscherl for fruitful discussions during the development process of the stream cipher.

References

[1] M.-L. Akkar and C. Giraud: An Implementation of DES and AES, Secure against Some Attacks, *CHES 2001*, (Ç. K. Koç, D. Naccache, and C. Paar,

eds.), Lecture Notes in Computer Science, vol. 2162, pp. 309–318, Springer-Verlag, 2001.

[2] L. Batina, J. Lano, N. Mentens, S. B. Örs, B. Preneel, I. Verbauwhede: Energy, Performance, Area versus Security Trade-offs for Stream Ciphers, *Workshop Records of SASCS – The State of the Art of Stream Ciphers* (Brugge, Belgium, 2004), pp. 302–310. Available at http://www.isg.rhul.ac.uk/research/projects/excrypt/stvl/sasc-record.zip

[3] S. R. Blackburn: A generalization of the discrete Fourier transform: determining the minimal polynomial of a periodic sequence, *IEEE Trans. Inform. Theory* **40**, 1702–1704 (1994).

[4] J. Blömer, J. G. Merchan, and V. Krummel: Provably Secure Masking of AES, *Selected Areas in Cryptography – SAC 2004*, Lecture Notes in Computer Science, vol. 3357, pp. 69–83, Springer-Verlag, 2004.

[5] N. G. deBruijn: A combinatorial problem, *Indag. Math.* **8**, 461–467 (1946).

[6] A. Canteaut: Open problems related to algebraic attacks on stream ciphers (invited talk), *Proc. of The International Workshop on Coding and Cryptography WCC'2005* (Bergen, Norway, 2005), P. Charpin and Ø. Ytrehus, eds., pp. 1-10.

[7] N. Courtois and W. Meier: Algebraic attacks on stream ciphers with linear feedback, *Advances in Cryptology – EUROCRYPT 2003* (E. Biham, ed.), Lecture Notes in Computer Science, vol. 2656, pp. 345–359, Springer-Verlag, 2003.

[8] Z.-D. Dai and J.-H. Yang: Linear complexity of periodically repeated random sequences, *Advances in Cryptology – EUROCRYPT '91* (D. W. Davies, ed.), Lecture Notes in Computer Science, vol. 547, pp. 168–175, Springer-Verlag, 1991.

[9] M. Feldhofer, S. Dominikus, and J. Wolkersdorfer: Strong Authentication for RFID Systems Using the AES Algorithm, *CHES 2004*, Lecture Notes in Computer Science, vol. 3156, pp. 357-370, Springer-Verlag, 2004.

[10] B. M. Gammel and R. Göttfert: Combining certain nonlinear feedback shift registers, *Workshop Record of SASC – The State of the Art of Stream Ciphers* (Brugge, Belgium, 2004), pp. 234–248. Available at http://www.isg.rhul.ac.uk/research/projects/excrypt/stvl/sasc-record.zip

[11] B. M. Gammel and R. Göttfert: Linear filtering of nonlinear shift register sequences, *Proc. of The International Workshop on Coding and Cryptography WCC'2005* (Bergen, Norway, 2005), P. Charpin and Ø. Ytrehus, eds., pp. 117-126.

[12] C. Gehrmann, M. Näslud: ECRYPT: Yearly Report on Algorithms and Keysizes (2004), 1 March 2004, Revision 1.0. Available at http://www.ecrypt.eu.org/documents.html

[13] R. Göttfert: *Produkte von Schieberegisterfolgen*, Ph.D. Thesis, Univ. of Vienna, 1993.

[14] J. H. Hoch and A. Shamir: Fault analysis of stream ciphers, *CHES 2004* (M. Joye and J.-J. Quisquater, eds.) Lecture Notes in Computer Science, vol. 3156, pp. 240–253, Springer-Verlag, 2004.

[15] C. J. A. Jansen: *Investigations On Nonlinear Streamcipher Systems: Construction and Evaluation Methods*, Ph.D. Thesis, Technical Univ. of Delft, Delft, 1989.

[16] P. Kitsos, N. Sklavos, K. Papadomanolakis, O. Koufopavlou: Hardware Implementation of Bluetooth Security, *IEEE Pervasive Computing*, **2**(1), 21-29 (2003).

[17] P. Kitsos, G. Kostopoulos, N. Sklavos, O. Koufopavlou: Hardware Implementation of the RC4 Stream Cipher, *Proc. of the 46th IEEE Midwest Symposium on Circuits and Systems*, pp. 27-30, Cairo, Egypt, 2003.

[18] A fast cryptographic checksum algorithm based on stream ciphers, *Advances in Cryptology – AUSCRYPT '92* (J. Seberry and Y. Zheng, eds.), Lecture Notes in Computer Science, vol. 718, pp. 339–348, Springer-Verlag, 1993.

[19] D. Laksov: Linear recurring sequences over finite fields, *Math. Scand.* **16**, 181–196 (1965).

[20] R. Lidl and H. Niederreiter: *Finite Fields*, Encyclopedia of Mathematics and Its Applications, vol. 20, Addison-Wesley, Reading, Mass., 1983. (Now Cambridge Univ. Press.)

[21] R. J. McEliece: *Finite Fields for Computer Scientists and Engineers*, Kluwer, Boston, 1987.

[22] W. Meidl and H. Niederreiter: On the expected value of the linear complexity and the k-error linear complexity of periodic sequences, *IEEE Trans. Inform. Theory* **48**, 2817–2825 (2002).

[23] W. Meier and O. Staffelbach: Fast correlation attacks on certain stream ciphers, *J. Cryptology* **1**, 159–176 (1989).

[24] S. Morioka and A. Satoh: An Optimized S-Box Circuit Architecture for Low Power AES Design, *CHES 2002*, Lecture Notes in Computer Science, vol. 2523, pp.172-186, Springer-Verlag, 2002.

[25] H. Niederreiter: Distribution properties of feedback shift register sequences, *Problems Control Inform. Theory* **15**, 19–34 (1986).

[26] H. Niederreiter: Cryptology—The mathematical theory of data security, *Prospects of Mathematical Science* (T. Mitsui, K. Nagasaka, and T. Kano, eds.), pp. 189–209, World Sci. Pub., Singapore, 1988.

[27] H. Niederreiter: *Random Number Generation and Quasi-Monte Carlo Methods*, CBMS-NFS regional conference series in applied mathematics, vol. 63, SIAM, 1992.

[28] J. M. Rabaey: *Digital Integrated Circuits*, Prentice Hall, 1996.

[29] R. A. Rueppel: Linear complexity and random sequences, *Advances in Cryptology – EUROCRYPT '85* (F. Pichler, ed.), Lecture Notes in Computer Science, vol. 219, pp. 167–188, Springer-Verlag, 1985.

[30] A. Satoh, S. Morioka, K. Takano, and S. Munetoh: A Compact Rijndael Hardware Architecture with S-Box Optimization, *Advances in Cryptology – ASIACRYPT 2001*, (C. Boyd, ed.), Lecture Notes in Computer Science, vol. 2248, pp. 239-254, Springer-Verlag, 2001.

[31] E. S. Selmer: *Linear Recurrence Relations over Finite Fields*, Department of Mathematics, Univ. of Bergen, 1966.

[32] A. Shamir, J. Patarin, N. Courtois, and A. Klimov: Efficient algorithms for solving overdefined systems of multivariate polynomial equations, *Advances in Cryptology – EUROCRYPT 2000* (B. Preneel, ed.), Lecture Notes in Computer Science, vol. 1807, pp. 392–407, Springer-Verlag, 2000.

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

[34] P. Voigtländer: Entwicklung einer Hardwarearchitektur für einen AES-Coprozessor, Ph.D.Thesis, Hochschule für Technik, Wirtschaft und Kultur Leipzig, Leipzig, Germany, 2003.

[35] E. A. Walker: Non-linear recursive sequences, *Can. J. Math.* **11**, 370–378 (1959).

[36] N. Zierler and W. H. Mills: Products of linear recurring sequences, *J. Algebra* **27**, 147-157 (1973).