CHAPTER 2

Stream Ciphers

RAINER A. RUEPPEL
R^3 Security Engineering

1 INTRODUCTION

Symmetric cryptosystems can be subdivided into block and stream ciphers. Block ciphers operate with a fixed transformation on large blocks of plaintext data; *stream ciphers* operate with a time-varying transformation on individual plaintext digits. Typically, a stream cipher consists of a keystream generator whose pseudo-random output sequence is added modulo 2 to the plaintext bits. A major goal in stream cipher design is to efficiently produce random-looking sequences, that is, sequences that as closely as possible resemble coin-tossing sequences. In general, one considers a sequence random if no patterns can be recognized in it, if no predictions can be made about it, and no simple description of it can be found. But if in fact the keystream can be generated efficiently, there certainly exists such a simple description. Nevertheless, the generator may produce "indistinguishable" sequences if no computations done on the keystream can reveal this simple description. The original key must be transformed in such a complicated way that it is computationally infeasible to recover the key. The level of randomness of a sequence can be defined in terms of the classes of computations which cannot detect statistical irregularities in it. Statistical tests of randomness emulate various simple computations encountered in practice, and check that the statistical properties of the sequence under investigation agree with those predicted if every sequence element was drawn from a uniform probability distribution.

We distinguish four principal approaches to the construction of stream cipher systems [102]. These approaches differ in their assumptions about the capabilities and opportunities of the cryptanalyst, in the definition of cryptanalytic success, and in the notion of security.

1. ***Information-theoretic approach.*** In the information-theoretic approach the cryptanalyst is assumed to have unlimited time and computing power. Cryptanalysis is the process of determining the message (or the particular key) given only the cryptogram and the a priori probabilities of various keys and messages. The secrecy system is considered broken when there is a "unique" solution to the cryptogram: one message with probability essentially unity while all others are practically zero. The assumption of an infinitely powerful adversary implies that the notion of security in the Shannon model is independent of the complexity of the encryption or decryption method. A cipher system is said to be ***perfectly secure*** if the plaintext and the ciphertext are statistically independent. That is, the cryptanalyst is no better off after observation of the cryptogram than he was before. A cipher system is said to ***be ideally secure*** if the cryptanalyst cannot find a unique solution for the plaintext, no matter how much ciphertext he is allowed to observe.

An interesting subproblem, called the local randomization problem, arises when the basic information-theoretic model is modified in such a way that the opponent's observations are assumed to be limited to less than a certain number of (not necessarily consecutive) ciphertext digits.

2. System-theoretic approach. The objective of the system-theoretic approach is to make sure that each new cryptosystem creates a difficult and previously unknown problem for the cryptanalyst. Breaking such a system is an "unglamorous" problem as compared to the cryptanalysis of a system based on some "famous" problem such as factoring or the discrete log. The designer's goal is to make sure that none of the fundamental cryptanalytic principles (such as substitution, divide and conquer, statistical analysis) are applicable to his system. To prevent cryptanalysis based on these fundamental principles, a set of general design criteria for keystream generators has evolved over time. Examples are period, linear complexity, statistical criteria, confusion, diffusion, and nonlinearity criteria for Boolean functions. Because the criteria are mainly of system-theoretic nature, we refer to this approach as the system-theoretic approach. A (stream) cipher system is designed in such a way that it directly satisfies the set of applicable design criteria. Doubtless this is today the most widely used practical design methodology.

3. Complexity-theoretic approach. In the complexity-theoretic approach all computations are parametrized by a security parameter, usually the key length, and an asymptotic analysis is carried out. Only algorithms whose running times can be expressed as polynomials in the size of the input are considered to be computationally feasible. Cryptanalysis is the process of (a) predicting a single digit of the keystream, or (b) distinguishing the keystream sequence from a truly random sequence. The designer's goal is to base his stream cipher system on, or make it equivalent to, some computationally infeasible problem. A keystream generator is defined to be ***perfect*** if it is (a) unpredictable, or (b) indistinguishable by all polynomial time statistical tests. Unfortunately, the perfect generator is a hypothetical device; it is not known if it exists. The proposed generators are all based on the assumed difficulty of one out of a few "famous" problems such as for instance, discrete logarithm, quadratic residuosity, and inverting the Rivest-Shamir-Adleman (RSA) encryption algorithm.

4. Randomized stream ciphers. The designer may, instead of aiming to ensure that the cryptanalytic process involves an infeasible work effort, try to ensure that the cryp-

tanalytic problem has an infeasible size. The objective is to increase the number of bits the cryptanalyst has to examine in the cryptanalytic process while keeping the secret key small. This can done by making use of a large publicly accessible random string in the encryption and decryption process. The key then specifies which parts of the large randomizer are to be used, whereas the opponent, not knowing the secret key, is forced to search through all the random data. The security of randomized stream cipher systems may be expressed by (a lower bound on) the average number of bits the cryptanalyst must examine before his chances of determining the key or the plaintext improve over pure guessing. Different interpretations of such a result are possible. The expected number of bittests is a lower bound on the number of steps that any algorithm breaking the system must perform, and thus leads to a notion of computational security. But the expected number of bittests is also a lower bound on the number of bits the opponent has to observe before his a posteriori probabilities of the various keys improve, and thus leads to a notion of information-theoretic security. These possible interpretations are the reason we have treated these stream cipher systems in a separate section.

In this survey, we will discuss and illustrate these four approaches. Many of the comments apply not only to stream ciphers but also to cryptosystems in general. Desmedt [24] recently looked at the dual problem of classifying cryptosystems according to the methods that have to be used to cryptanalyze them.

Often, linear feedback shift registers (LFSR) and linear congruential generators are recommended as pseudorandom sequence generators. But being linear devices they are usually easily cryptanalyzed no matter how many of the parameters are kept secret. Since we do not consider LFSRs and linear congruential generators as possible keystream generators (although they may be potential building blocks in a stream cipher design), and since there is a wealth of literature on their characteristic properties [43,60,130], we refrain from describing them in this survey.

As history has shown, most cryptanalytic successes were caused by human failure and poor administrative procedures, especially in the key management domain. In addition, a secure cipher design does not necessarily imply a secure implementation, as was pointed out explicitly by Desmedt [24]. It seems obvious that not only does one have to verify that the implemented cipher algorithm performs as specified in the design, but also that the assumptions of the design, such as physical security of the keys, are satisfied. There also remains the larger issue of verifying that no additional functionality in the implementation compromises security. A description of historic designs and failures or implementation issues is beyond the scope of this survey, where we want to concentrate on current research directions in stream cipher design,

First we give a classification of stream cipher systems and their different modes of operation.

1.1 Terminology and Modes of Operation

Automata theory and, in particular, finite-state machine theory are well suited to describe stream cipher systems and their different modes of operation. Let $\mathcal{X}$ denote the plaintext alphabet, $\mathcal{Y}$ the ciphertext alphabet, $\mathcal{Z}$ the keystream alphabet, $\mathcal{S}$ the state space (internal state) of the stream cipher, $\mathcal{K}$ the key space. Let x_i, y_i, z_i, and s_i denote the plaintext digit, ciphertext digit, keystream digit, and the internal state at time i. A key $k \in \mathcal{K}$ is selected according to a probability distribution P_K. Usually the key is

drawn according to the uniform probability distribution, but occasionally it may be infeasible to select the key completely randomly.

A general stream encryptor is described by the equations

$$s_{i+1} = F(k, s_i, x_i)$$
$$y_i = f(k, s_i, x_i)$$

where F is the next-state function and f is the output function. Typically

$$y_i = x_i + F(k_i, s_i)$$

a condition that is necessary and sufficient for the stream decryptor to operate without delay [66]. The sequence

$$\{z_i = f(k, s_i) : i \geq 1\}$$

is referred to as the keystream. To provide secure encryption the keystream must be as random as possible.

1.1.1 Synchronous stream ciphers and pseudorandom generators. Stream ciphers are commonly subdivided into synchronous and self-synchronizing systems. In a synchronous stream cipher the keystream is generated independently of the message stream (and the ciphertext stream). The corresponding device is referred to as keystream generator, running-key generator, or pseudorandom sequence generator. The operation of the keystream generator is governed by the two rules:

$$s_{i+1} = F(k, s_i)$$
$$z_i = f(k, s_i)$$

The initial state s_0 may be a function of the key k, and possibly, some randomization variable. The purpose of a keystream generator is to expand a short random key k into a long pseudorandom string z' $= z_1, \ldots, z_l$. This can be written compactly as

$$G: \mathcal{K} \rightarrow \mathcal{Z}^l$$
$$z^l = G(k)$$

which emphasizes the functional relationship between the key k and the keystream z^l. For a binary key k^n of length n and a binary keystream of length l the (n, l) sequence generator is a function from $\{0, 1)^n$ to $\{0, 1\}^l$ such as that $z^l = G(k^n)$.

In complexity theory a generator is defined asymptotically as an infinite class $\{G_n : n \geq 1\}$ of (n, l(n)) sequence generators, where l is a polynomial function of the index n and the computation time of each sequence generator is upperbounded by a polynomial function of n.

Synchronous stream ciphers can be further classified according to the mode they are operated in :

Counter mode [25]:

$$s_{i+1} = F(s_i)$$
$$z_i = f(k, s_i)$$

The next-state function does not depend on the key but is guaranteed to step through all (or most) of the state space. Examples for such F are ordinary counters and maximum-length LFSRs. The cryptographic strength necessarily resides in the output function f.

Output feedback (internal feedback) mode [15]:

$$s_{i+1} = F(k, s_i)$$
$$z_i = f(s_i)$$

The output function f does not depend on the key. Very often it simply consists of a l-bit predicate of the current state (for instance, the least significant bit or the parity). For a discussion of using the Data Encryption Standard (DES) in output feedback mode see [31]. Sometimes one sees a variant of this mode where the key k determines only the initial state.

$$s_0 = k$$
$$s_{i+1} = F(s_i)$$
$$z_i = f(s_i)$$

A synchronous stream cipher has no error propagation. The decryption of a distorted ciphertext digit only affects the corresponding plaintext digit. But a synchronous stream cipher (as its name indicates) requires perfect synchronization between sender and receiver. If a digit is gained or lost during transmission the receiver only obtains garbled data after the point of synchronization loss. To regain synchronization would typically require searching over the possible offsets between the sender's and the receiver's clock. In general, to "enter late" into an ongoing ciphertext sequence is only possible for the decrypter if he can determine the correct time instant registered by the encryptor's clock.

1.1.2 Self-synchronizing stream ciphers and scramblers. The most common mode of a self-synchronizing stream cipher is the ***cipher feedback mode:***

$$s_i = F(y_{i-1}, y_{i-2}, \ldots, y_{i-N})$$
$$z_i = f(k, s_i)$$

The state of the stream cipher is determined by the previous N ciphertext symbols. The stream encryptor employs feedback, but the corresponding finite-state machine inverse, the decryptor, uses feedforward. The cryptographic strength resides in the output function. Note that input (the ciphertext symbols) and output (the keystream) off are known to the cryptanalyst under a known plaintext attack. A self-synchronizing stream decryptor has, as its name indicates, the ability to automatically synchronize itself without knowledge of the encryptor's clock time. On reception of N correct consecutive channel digits its state becomes identical to the encryptor's state and synchronization is regained. A self-synchronizing stream cipher has limited error propagation. A distorted channel digit remains in the internal state of the stream decryptor for N consecutive plaintext digits (as it is shifted through the state). After reception of N consecutive undistorted channel digits the stream decryptor is able to decipher correctly.

A ***scrambler*** is also defined as a finite-state machine that, regardless of the initial state, converts a periodic input sequence into a periodic output sequence with generally higher, but at least the same, period [103]. Scramblers have a randomizing effect on the data patterns transmitted and thus can reduce the sensitivity of synchronization systems to specific periodic data patterns [127]. Typically scramblers are linear devices whose purpose is to facilitate synchronization. Nonlinear key-dependent scramblers (self-synchronizing stream ciphers) have the potential to combine secrecy with ease of synchronization. Some basic results and a case study are reported in [66].

Synchronous stream ciphers can be equipped with a self-synchronizing feature. This typically involves periodically checking the internal state of the decryptor or reinitializing the states of both the encryptor and the decryptor on some preagreed condition.

2 INFORMATION-THEORETIC APPROACH

In his landmark paper [109] Shannon developed the basic information-theoretic approach to secrecy systems. In the Shannon model the cryptanalyst is assumed to have unlimited time and computing power. He is restricted to a ciphertext-only attack, but as Diffie and Hellman [25] pointed out, a known segment of plaintext can be taken into account as added redundancy since it improves the a priori information of the cryptanalyst about the plaintext. The a posteriori knowledge consists of the (a posteriori) probabilities of the various messages and keys that may have produced the observed ciphertext. Cryptanalysis in the Shannon model is defined as the process of determining the message (or the particular key) given only the cryptogram and the a priori probabilities of various keys and messages. The secrecy system is considered broken when there is a "unique" solution to the cryptogram: one message with probability essentially unity while all others are practically zero. Note that the assumption of an infinitely powerful adversary implies that the notion of security in the Shannon model is independent of the complexity of the encryption or decryption method.

Let $X^n = X_1, \ldots, X_n$ denote an n-bit plaintext message, let $Y^n = Y_1, \ldots, Y_n$ denote the corresponding n-bit ciphertext message, and let K be the key drawn according to the probability distribution P_K. If $H(X^n)$ denotes the uncertainty about the plaintext and $H(X^n \mid Y^n)$ the conditional uncertainty after observation of Y^n, then the mutual information $I(X^n; Y^n) = H(X^n) - H(X^n \mid Y^n)$ between X^n and Y^n is a basic measure of security in the Shannon model. Three cases may occur:

1. $I(X^n; Y^n) = 0$ for all n; then X^n and Y^n are statistically independent, and the cryptosystem is said to be *perfectly secure*. The cryptanalyst cannot do better than guess the message according to the probability distribution of meaningful messages. In other words, the (a posteriori) probability distribution of decrypted messages $D(K, y^n)$ after observation of y^n with K chosen according to P_K is, for all y^n, identical to the (a priori) probability distribution over the message space. In a perfect cryptosystem the basic inequality $H(K) \geq H(X^n)$ holds. Thus, fresh key has to be supplied at a constant rate. Knowing a plaintext segment of a perfect cipher may reveal the subkey used to encrypt that plaintext segment but will in general not allow to sharpen the a posteriori probabilities of the remaining message segments. The one-time pad discussed below is a perfect cipher.

2. $0 < I(X^n; Y^n) < H(X^n)$ for large n; then there remains a residual uncertainty about the plaintext that cannot be resolved (not even with unlimited computing power). This condition implies that the key equivocation $H(K \mid Y^n) > 0$. Cryptosystems with a finite key for which $H(K \mid Y^n) > 0$ when $n \to \infty$ are called *ideally secure*. The cryptanalyst can sharpen the probability distribution of possible messages X^n, but he cannot find a unique solution for X". This does not necessarily mean that the system is secure; the residual uncertainty may only affect a small portion of the plaintext. To reach ideal security usually demands perfect data compression, or randomization of the plaintext [47]. Note that ideal security hinges on complete confidentiality of the plaintext.

3. $I(X^n; Y^n) \approx H(X^n)$ for large n; after a certain number of ciphertext bits there remains only one solution for the corresponding plaintext (and the applied key). This is typically the case with practical cryptosystems.

As a theoretical security index for cryptosystems Shannon introduced the ***unicity distance*** $n_u = \min \{n : H(K \mid Y^n) \approx 0\}$ which is the required amount of ciphertext before the cryptanalyst can, in principle, solve for the key. Once he knows the key he also knows the message. For typical cryptosystems $n_u \approx H(K)/(1 - h)$ where h denotes the information rate per plaintext bit, and $1 - h$ is the redundancy per plaintext bit. Note that both for perfect and ideal cryptosystems $n_u = \infty$, and hence it became commonplace to call perfect and ideal cryptosystems ***unconditionally secure*** [22,25]. For perfect cryptosystems this is a useful definition, but an ideal cryptosystem can become disastrously insecure in a known plaintext attack. Thus it seems inappropriate to call an ideal cryptosystem unconditionally secure.

One of the most famous cipher systems is doubtless the one-time pad (sometimes called the Vernam cipher) [109,118]. It received its name from the requirement that every part of the key be only used once.

One-time pad:

Input: message: a sequence of message bits x_i, $i = 1, 2, \ldots$
key: a sequence of independent and uniformly distributed bits k_i, $i = 1, 2, \ldots$

Output: ciphertext $y_i = x_i \oplus k_i$, $i = 1, 2, \ldots$, consisting of the bitwise exor of the message stream with the key sequence.

The one-time pad is a stream cipher in the sense that every message bit is transformed independently with a time-varying transformation. There is no keystream generator since the key is directly added to the message. The one-time pad is perfect, $I(X^n; Y^n) = 0$. Knowing part of the plaintext does not violate the statistical independence condition for the remaining segments. The number of key bits has to be larger than the number of message bits. The key generation, distribution, and management problem is enormous, so that it is only used for certain untypical applications such as two users sharing a hotline with high confidentiality requirements.

Equivalent to the unicity distance of a cipher system, one may define the ***unicity distance of a keystream generator*** as the number of keystream symbols that need to be observed in a known plaintext attack before the key can be uniquely determined. In a known plaintext attack there is no uncertainty about the plaintext, $h = 0$, and consequently $n_{G,u} \approx H(K)$.

2.1 local Randomization

Since the unicity distance of a keystream generator with a finite key K is about $n_{G,u} \approx H(K)$ bits, there is no security left in the Shannon sense once the opponent has acquired more than $n_{G,u}$ bits of keystream. An interesting subproblem arises when the Shannon model is modified in such a way that the opponent is given direct access to the keystream, but only to a limited number of e (not necessarily consecutive) bits of his choice. The problem of proving security under this additional constraint is called the

local randomization problem [73]. Note that e has to be smaller than $H(K)$. Schnorr [105] motivated the local randomization problem by presenting the following construction:

Schnorr's pseudorandom generator G = {G,} :

Input: the key (seed) k is a random function $f : I_m \rightarrow I_m$; size of description $m2^m$.

1. Set $y_i^0 = i$ for $i = 0, 1, \ldots, 2^{2m} - 1$.
2. For $j = 0, 1, 2,$ do

$$y_i^{j+1} = (R(y_i^j), L(y_i^j) \oplus f(R(y_i^j)))$$

(L and R mean left and right half of argument.)

Output $G_n(k)$: the sequence of y_i^3, $i = 0, 1, 2, \ldots, 2^{2m} - 1$.

Note that the key k has size $n = m2^m$ (the function description) and that $G,(k)$ has length $2m2^{2m}$. Thus, the generator stretches a seed of length n into roughly n^2 pseudo-random bits. Schnorr's generator is an application of the permutation function generator developed by Luby and Rackoff [63]. This permutation generator consists of an M-round DES-like structure, where in each round i a different (pseudo) random function f_i is applied. It is shown that three rounds suffice to prove perfectness (polynomial-time indistinguishability) of the resulting permutation generator, provided the functions f_i are indistinguishable.

Schnorr used the following asymptotic notion of security borrowed from complexity theory (see Section 3). A generator $G = \{G_n\}$ is provably locally randomized if it passes all (even those with unlimited time-bound) statistical tests $T = \{T_n\}$ that depend on at most $e(n)$ bits of $G_n(k)$. Schnorr claimed that his generator is provably locally randomized with $e(m) = 2^{m/3-(\log m)^2}$. In [101] it was shown that this is not true. In fact, a statistical test was given that efficiently distinguishes Schnorr's generator with only $e(m) = 4m$ bits of $G_n(k)$. This result even holds for an arbitrary number of internal rounds (instead of the proposed 3), or, for an odd number of internal rounds, when the seed is doubled and two functions f, g are used alternately. The flaw in Schnorr's construction was independently discovered by Ohnishi [88], and Zheng, Matsumoto, and Imai [129], who were studying the construction of pseudorandom permutations, as proposed by Luby and Rackoff [63]. Ohnishi discovered that, for an odd number of internal rounds, G can be distinguished for any palindromic arrangement of functions.

If Schnorr's generator is subjected to a system-theoretic analysis the result is disastrous. There exists a linear time attack that recovers the key from a segment of $m2^m + O(m)$ bits of keystream [101]. Note that this segment is only slightly larger than the unicity distance of G,. This attack reaches the performance limits of any attack, since simply reading the key from memory requires linear time and linear space in the size of the key. This result shows that, if the assumptions of the security model do not coincide with the practical application, security can break down.

Since the basic model is still the Shannon model it seems more natural to define security directly in information-theoretic terms. Maurer and Massey [73] used the following notion of security: an (n, l, e)-perfect local randomizer if every subset of e bits of the l output bits is a set of independent and

uniformly distributed (i.u.d.) binary random variables (provided the key bits are independent and uniformly distributed random variables). By interpreting an (n, o-sequence generator as the encoder of a binary block code with 2" codewords of length l, where the n information bits correspond to the key, they were able to use coding-theoretic arguments and tools. As a consequence, the problem of determining the maximum achievable degree of perfect local randomness of any linear (n, l) sequence generator is equivalent to the problem of determining the maximum achievable minimum distance d of a linear binary $[l, l - n, d]$ code. They showed that the encoder of an extended Reed–Solomon (RS) code over $GF(2^m)$ with e information symbols, codeword length 2^m, and design distance $2^m - e + 1$ is a linear $(me, m2^m, e)$-perfect local randomizer when the symbols are appropriately represented by m binary digits. Note that $e >$ n/log 1. The parameters of an extended RS code can be chosen to coincide with the parameters $n = m2^m$ and $l = 2m2^{2m}$ of Schnorr's generator. If the extended RS code is defined over $GF(2^{2m})$ with 2^{m-1} information symbols then the corresponding encoder is a $(m2^m, 2m2^{2m}, 2^{m-1})$-perfect local randomizer. This local randomizer not only achieves true local randomness instead of indistinguishability but also gives a degree of local randomness that is greater than the third power of what Schnorr hoped for. In [73] it was also shown that nonlinear perfect local randomizers based on nonlinear codes exist whose degree of local randomness surpasses the degree of the corresponding best linear local randomizer, but no constructions were given.

3 SYSTEM-THEORETIC APPROACH

The system-theoretic approach to stream cipher design is twofold. One objective is to develop methods and building blocks that have provable properties with respect to certain system-theoretic measures such as period, linear complexity, frequency distribution, and distance to linear structures. A second objective is to study cryptanalytic principles and to develop design rules that render attacks based on these principles impossible. Such fundamental cryptanalytic principles are, for instance, substitution and approximation (preferrably by linear components), divide and conquer (on the key-space), and exploitation of statistical deficiencies (such as intersymbol dependencies). To prevent cryptanalysis based on these fundamental principles a set of general design criteria for keystream generators [2,89,98,109] has evolved over time, for instance,

1. Long period, no repetitions
2. Linear complexity criteria: large linear complexity, linear complexity profile, local linear complexity, etc.
3. Statistical criteria such as ideal k-tuple distributions
4. Confusion: every keystream bit must be a complex transformation of all or most of the key bits
5. Diffusion: Redundancies in substructures must be dissipated into long-range statistics
6. Nonlinearity criteria for Boolean functions like mth-order correlation immunity, distance to linear functions, avalanche criterion, etc.

Any secure keystream generator must satisfy this set of general design criteria. In the system-theoretic approach, the ciphers are designed to directly satisfy the criteria (in

general, under the most favorable conditions for the cryptanalyst). Doubtless this is today the most widely used approach to practical security. The DES, although in its basic mode not a stream cipher, provides a good example of a system-theoretic design. With a finite key of only 56 bits (which is much smaller than the necessary keysizes for most other systems inluding public-key systems) the DES has shown considerable resistance to (public) cryptanalysis over the past 15 years.

But a major problem is that the design criteria only partially reflect the general cryptanalytic principles. It may happen that a generator provably satisfies all design criteria and yet is insecure. Or, in other words, the design criteria form a set of necessary but not sufficient conditions for the security of a keystream generator. The advantage of the system-theoretic approach is that each new cryptosystem creates a difficult and previously unknown problem for the cryptanalyst. Breaking such a system is an "unglamorous" problem as compared to the cryptanalysis of a system based on some "famous" problem such as factoring or the discrete log.

3.1 Transform Techniques

In this section we introduce some basic definitions and describe transform techniques that are useful in the analysis of sequences and cryptographic transformations, especially of Boolean functions.

3.1.1 Discrete Fourier transform and linear complexity. If α is a primitive N-th root of unity in any field IF, then the discrete Fourier transform (DFT) of the "time-domain" sequence $a^N = (a_0, a_1, \ldots, a_{N-1})$ with components from F is the "frequency-domain" sequence $A^N = (A_0, A_1, \ldots, A_{N-1})$ where

$$A_i = \sum_{j=0}^{N-1} a_j \alpha^{ij} \qquad i = 0, 1, \ldots, (N-1)$$

The inverse DFT relation is

$$a_j = \frac{1}{N^*} \sum_{i=0}^{N-1} A_i \alpha^{-ij} \qquad j = 0, 1, \ldots, (N-1)$$

where $N^* = N \bmod p$ if the characteristic of F is p and $N^* = N$ if the characteristic of F is 0.

We proceed to discuss the close relationship between the DFT and the linear complexity of a sequence, and we follow the treatment given in [104]. The computation of the ith component A_i of the DFT may be viewed as the inner product between a^N and the sequence $(\alpha^{0i}, a", \alpha^{2i}, \ldots, a^{(N-1)i})$. Therefore, it is possible to describe the DFT as a matrix transform

$$A^N = a^N \cdot F$$

where

$$F = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \alpha^1 & \alpha^2 & \ldots & \alpha^{N-1} \\ \vdots & & & & \\ 1 & \alpha^{N-1} & \alpha^{2(N-1)} & \ldots & \alpha^{(N-1)(N-1)} \end{pmatrix}$$

Analogously, the inverse DFT is given as

$$a^N = \frac{1}{N^*} A^N \cdot F^{-1}$$

where

$$F^{-1} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha^{-1} & \alpha^{-2} & \cdots & \alpha^{-(N-1)} \\ \vdots & & & & \\ 1 & \alpha^{-(N-1)} & \alpha^{-2(N-1)} & \cdots & \alpha^{-(N-1)(N-1)} \end{pmatrix}$$

Now define the circulant matrix of a^N [104], denoted $M(a^N)$, to be the matrix whose rows consist of the N cyclic left shifts of a^N.

$$M(a^N) = \begin{pmatrix} a_0 & a_1 & \cdots & a_{N-1} \\ a_1 & a_2 & \cdots & a_0 \\ & & & \\ a_{N-1} & a_0 & \cdots & a_{N-2} \end{pmatrix}$$

Let $L = \Lambda(\tilde{a})$ be the linear complexity of the sequence $\tilde{a} = a_0, a_1, a_2, \ldots$. Then, by the definition of the linear complexity, L is the smallest integer such that the ith row of $M(a^N)$ can be written as a linear combination of the previous rows. Hence, the rank of $M(a^N)$ is at least L. On the other hand, every row with index $L \leq i \leq N - 1$ can be written as a linear combination of previous rows (in fact, of the L previous rows). Hence, the rank of $M(a^N)$ is L. This establishes that the linear complexity of a periodic, semi-infinite sequence $\tilde{a} = (a^N)^\infty$ is equal to the rank of the circulant $M(a^N)$.

The circulant matrix $M(a^N)$ may be written as

$$M(a^N) = F^{-1} \cdot D_A \cdot F^{-1}$$

where F^{-1} is the inverse DFT matrix, and D_A is the N by N diagonal matrix whose diagonal entries are the elements of A^N. Under the assumption that a is a primitive Nth root of unity the matrices F and F^{-1} have full rank. Hence,

$$\text{rank } (M(a^N)) = \text{rank } (D_A)$$

But the rank of D_A is equal to the number of nonzero elements in A^N, that is, equal to $W_H(A^N)$. This establishes the first part of the following theorem which has been used implicitly by Blahut [7], but was first established by Massey in this explicit form [70]. The second part is proved analogously.

Blahut's theorem: ***The linear complexity of the periodic, semi-infinite "time-domain" sequence $\tilde{a} = (a^N)^\infty$ is equal to the Humming weight of the finite-length, "frequency-domain" sequence A^N, where A^N is the DFT of a^N,***

$$\Lambda((a^N)^\infty) = w_H(A^N)$$

Similarly, the linear complexity of the periodic, semi-infinite sequence $(A^N)^\infty$ is equal to the Humming weight of the finite-length sequence a^N, where a^N is the inverse DFT of A^N.

$$w_H(a^N) = \Lambda((A^N)^\infty)$$

Many linear complexity results about nonlinear combinations of shift register sequences have alternate proofs in the "frequency" domain.

3.1.2 Walsh transform and Boolean functions. The Walsh transform of a real-valued function $f : \mathbb{F}_2^n \to \mathbb{R}$ is defined as

$$F(w) = \sum f(x)(-1)^{\omega \cdot x}$$

where $\omega \cdot x$ denotes the dot product $\omega_1 x_1 \oplus \omega_2 x_2 \oplus \ldots \omega_n x_n$. The function f can be recovered by the inverse Walsh transform

$$f(x) = 2^{-n} \sum_{\omega}' F(\omega)(-1)^{\omega \cdot x}$$

Let the real-valued function $f : \mathbb{F}_2^n \to \mathbb{R}$ be specified by the naturally ordered vector

$$[f(x)] = [f(0), f(1), \ldots, f(2^n - 1)]$$

where $f(x) = f(x_1, x_2, \ldots, x_n)$ and $x = x_1 + x_2 2 + \ldots + x_n 2^{n-1}$. Analogously, let the Walsh transform $F : \mathbb{F}_2^n \to \mathbb{R}$ of the function f be given by the naturally ordered vector

$$[F(\omega)] = [F(0), F(1), \ldots, F(2^n - 1)]$$

Then, the Walsh transform may be represented as the matrix transform

$$[F(\omega)] = [f(x)] \cdot H_n$$

where H_n denotes the Hadamard matrix of order n. H_n is defined recursively by

$$H_0 = [1]$$

$$H_n = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes H_{n-1}$$

where $\otimes$ denotes the Kronecker product of matrices. Since $H_n^2 = 2^n I_n$, the inverse Walsh transform is given by

$$[f(x)] = 2^{-n}[F(\omega)] \cdot H_n$$

By treating the values of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ as the real numbers 0 and 1, one can define the Walsh transform of such Boolean f. Sometimes it is more convenient to work with the equivalent binary function $\hat{f} : \mathbb{F}_2^n \to \{-1, +1\}$ defined by

$$\hat{f}(x) = (-1)^{f(x)}$$

whose Walsh transform has the form

$$\hat{F}(\omega) = \sum_{x} (-1)^{f(x) \oplus \omega \cdot x}$$

The relationship between $\hat{F}(\omega)$ and F(o) is given by

$$\hat{F}(\omega) = 2^n\delta(\omega) - 2F(\omega)$$
$$F(\omega) = 2^{n-1}\delta(\omega) - \frac{1}{2}\hat{F}(\omega)$$

where $\delta(\omega) = 1$ for $\omega = 0$ and 0 otherwise.

In this survey, whenever we apply the Walsh transform to establish properties of the Boolean function f, we use the equivalent binary function $\hat{f}$ to compute $\hat{F}$.

3.1.3 Algebraic normal form transform. Let $f : \mathbb{F}_q^N \rightarrow \mathbb{F}_q$ be a $\mathbb{F}_q$ switching function. Then f may be written in algebraic normal form (ANF) as

$$f(\mathbf{x}) = \sum_{\mathbf{i}\in\mathbb{Z}_q^N} c_{\mathbf{i}}\mathbf{x}^{\mathbf{i}}$$

where $\mathbf{x} = (x_1, \ldots, x_N) \in \mathbb{F}_q^N$, $\mathbf{i} = (i_1, \ldots, i_N) \in \mathbb{Z}_q^N$, $c_{\mathbf{i}} \in \mathbb{F}_q$, and

$$\mathbf{x}^{\mathbf{i}} = \prod_{n=1}^{N} x_n^{i_n}$$

The order of a product term is defined as $\Sigma_{n=1}^{N} i_n$. The nonzero constant term is defined to have 0th order. The linear terms have order 1. The order of the function f is defined to be the maximum of the order of its product terms that have a nonzero coefficient.

Now consider the Boolean function: $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$. Let f be specified by the naturally ordered vector

$$[f(x)] = [f(0), f(1), \ldots, f(2^n-1)]$$

where $f(x) = f(x_1, x_2, \ldots, x_n)$ and $x = x_1 + x_2 2 + x_3 2^2 + \ldots x_n 2^{n-1}$.

Let the ANF of f be specified by the naturally ordered vector

$$[a_i] = [a_0, a_1, \ldots, a_{2^n-1}]$$

where $a_i = a(i_1, i_2, \ldots, i_n)$ and $i = i_1 + i_2 2 + i_3 2^2 + \ldots i_n 2^{n-1}$. Then the ANF transform is given by [50,98]

$$[a_i] = [f(x)]A_n$$

and the inverse ANF transform is

$$[f(x)] = [a_i]A_n$$

where A_n is defined recursively by

$$A_0 = [1]$$
$$A_n = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \otimes A_{n-1}$$

and $\otimes$ denote the Kronecker product of matrices. Note that A_n is an involution, i.e., that $A_n^2 = I$.

There is a fast way to iteratively evaluate the binary ANF transform [50,98]. Let $[f^1(x)]$ and $[f^2(x)]$ be the first and second half of the vector $[f(x)]$. Then

$$[a_i] = [f(x)] \cdot A_n = [[f^1(x)] \cdot A_{n-1}, ([f^1(x)] + [f^2(x)]) \cdot A_{n-1}]$$

which can be iterated until A, is reached.

The general ANF transform for GF(q)-functions has been described in [50].

3.2 Period and linear Complexity of Sequences

The linear complexity $\Lambda(s^l)$ of the sequence $s^l = s_0, s_1, \ldots, s_{l-1}$ is the length L of the shortest LFSR that can generate s^l when the first L digits of s^l are initially loaded into the register. Equivalently, the linear complexity $\Lambda(s^l)$ is defined to be the smallest nonnegative integer L such that there exists a linear recursion with fixed constants $c_1, c_2, \ldots, c_L$ satisfying

$$s_j + c_1 s_{j-1} + \ldots + c_L s_{j-L} = 0 \qquad L \leq j < l$$

The linear complexity is a very useful concept in the study of stream ciphers. As was pointed out by Massey [71] any sequence that can be generated by a finite-state machine (whether linear or nonlinear) over a finite field also has a finite linear complexity. Moreover, there exists an efficient algorithm, the Berlekamp-Massey algorithm [65], that computes the linear complexity of a given sequence s^l in time lL. This algorithm forms a universal attack for running key generators since it carries the potential of substituting any running key generator by its shortest linear equivalent. Therefore, a necessary condition that any running key generator must meet, is the requirement for a large linear complexity [89,98].

3.2.1 Random sequences. Define L_i to be the linear complexity of the subsequence $s^i = (s_0, s_1, \ldots, s_{i-1})$. Then the sequence $L, L_2, \ldots, L_l$ is called the ***linear complexity profile*** of s^l. For binary independent and uniformly distributed sequences L_i becomes a random variable. In [95] it is shown that

$$\mathrm{E}(L_i) = \frac{i}{2} + \frac{1}{4} + \frac{(-1)^i}{36} + O(i2^{-i})$$

$$\mathrm{Var}(L_i) = \frac{86}{81} + O(i2^{-i})$$

These results have been generalized by Smeets [114] to sequences over arbitrary finite fields $\mathbb{F}_q$. He showed that the variance of the linear complexity decreases with the size of the sequence alphabet; for large values of q the variance will be approximately l/q. Thus, the larger the size of $\mathbb{F}_q$ the closer the complexity profile will follow the line $i/2$.

The close relationship between the linear complexity profile and continued fractions was studied in [21,82,83,121]. Knowing that the linear complexity profile of random sequences will typically stay close to the $i/2$ line, one may reverse the problem and ask how random is a sequence whose linear complexity profile perfectly follows the $i/2$ line, that is, where $L_i = \lfloor (i+1)/2 \rfloor$ for $i \geq 1$. In [95] it was conjectured that the sequence $\tilde{s}$ which contains a 1 at the locations $i = 2^j - 1$ and is zero otherwise possesses such a perfect linear complexity profile. This conjecture was proved in [21] using the Euclidean algorithm. An adapted version of the proof was given in [69] using the

Berlekamp-Massey synthesis algorithm. In [120] a complete characterization of all binary sequences that possess a perfect linear complexity profile was derived. It was shown that a binary sequence $\tilde{s}$ has a perfect linear complexity profile if and only if $s_0 = 1$ and $s_{2i} = s_{2i-1} \oplus s_{i-1}$ for $i \geq 1$. A relatively short proof of this characterization can also be found in [83].

Instead of investigating the ensemble of all sequences of a given length i, one may consider how the linear complexity of a randomly chosen and then fixed sequence will behave as i varies. Niederreiter [84] has shown that $\lim_{i \to \infty} L_i / i = 1/2$ for almost all sequences over $\mathbb{F}_q$. He defined a sequence $\tilde{s}$ to possess a good linear complexity profile [85] if there exists a constant c (that may depend on the sequence) such that

$$\left| L_i - \frac{1}{2} \right| \leq c \operatorname{Log} i \qquad \text{for all } i \geq 1$$

where Log i = max{1, log i}, and showed that almost all $\mathbb{F}_q$ sequences possess such a good linear complexity profile.

Piper [89] pointed out that a keystream sequence should have an acceptable linear complexity profile for every starting point. Niederreiter showed [85] also that $\lim_{i \to \infty} L_i(\tilde{s}_m)/i = 1/2$ for all shifted versions $\tilde{s}_m = s_m, s_{m+1}, \ldots$, $m \geq 0$, of almost all sequences $\tilde{s}$ over $\mathbb{F}_q$. Naturally extending the notion of good linear complexity profile, he defined a sequence to possess a uniformly good linear complexity profile if there exists a constant c (that may depend on the sequence but not on the shift factor m) such that

$$\left| \Lambda_i(\tilde{s}_m) - \frac{1}{2} \right| \leq c \operatorname{Log} i \quad \text{for all } m \geq 0 \text{ and } i \geq 1$$

However, a uniformly good linear complexity profile is not a typical property of random sequences, since it was shown in [85] that almost no sequence has such a uniformly good linear complexity profile.

The linear complexity profile quite naturally suggests a statistical test: The linear complexity profile of well-designed running key generators is indistinguishable from the linear complexity profile of random sequences. An experimental study of the practical applicability of the linear complexity profile test was reported in [114].

3.2.2 Periodic sequences. Consider the periodic $\mathbb{F}_q$-sequence $\tilde{s} = (s^T)^\infty$ and assume that

$$s_i + c_1 s_{i-1} + \ldots + c_L s_{i-L} = 0 \qquad L \leq i$$

is the linear recursion of least possible order that can generate the sequence $\tilde{s}$. The polynomial

$$m_{\tilde{s}}(x) = x^L + c_1 x^{L-1} + \ldots + c_L$$

associated with the linear recursion is called the minimal polynomial of $\tilde{s}$. $m_{\tilde{s}}$ is monic and unique; it divides the characteristic polynomial of any other linear recursion that is satisfied by $\tilde{s}$. The sequence $\tilde{s}$ may be represented by a linear combination of fundamental solutions of the associated linear recursion, or equivalently, by a linear combination of the roots of the corresponding minimal polynomial, thereby yielding a direct "time-

domain solution" (see, e.g., [54,81,131]. Let $m_{\tilde{s}}(x)$ consist of K irreducible factors m,(x) with multiplicity e_k,

$$m_{\tilde{s}}(x) = \prod_{k=1}^{K} m_k^{e_k}(x)$$

Furthermore, let d_k be the degree and α_k be one of the roots of the kth irreducible factor. Then the ith digit of $\tilde{s}$ is uniquely determined by

$$s_i = \sum_{k=1}^{K} \sum_{j=0}^{d_k-1} \alpha_k^{iq^j} \sum_{l=0}^{e_k-1} A_{k,l}^{q^j} \binom{i+l}{l}$$

with $A_{k,l}^{q^j} \in GF(q^{d_k})$, $A_{k,e_k} \neq 0$ being determined by the initial terms of the sequence, and the binomial coefficients being computed modulo the characteristic p of the field $\mathbb{F}_q$.

If $m_{\tilde{s}}(x)$ is the minimal polynomial of the sequence $\tilde{s}$ then the least positive period T of $\tilde{s}$ is equal to the exponent (also called order or period) of $m_{\tilde{s}}(x)$. If T_k denotes the exponent of $m_k(x)$ or equivalently, the multiplicative order of the root α_k, then [4,60]

$$T = p^e \operatorname{lcm}(T_1, \ldots, T_k)$$

where e is the smallest integer with $p^e \geq \max\{e_1, \ldots, e_k\}$. When $\mathbb{F}_q$-sequences are combined then the two operations of termwise addition and termwise multiplication are of principal interest. Consider first the sum of two sequences $\tilde{z} = \tilde{x} + \tilde{y}$. In [45] the following bounds on the linear complexity of the sum sequence are given

$$\Lambda(\tilde{x}) + \Lambda(\tilde{y}) - 2\gcd(T_x, T_y) \leq \Lambda(\tilde{z}) \leq \Lambda(\tilde{x}) + \Lambda(\tilde{y})$$

Equivalently, one may prove that $\Lambda(\tilde{z}) = \Lambda(\tilde{x}) + \Lambda(\tilde{y})$ if $\gcd(T_x, T_y) = 1$ and $x - 1$ divides at most one of the two minimal polynomials $m_{\tilde{x}}, m_{\tilde{y}}$ [38]. *The* period of the sum sequence is implicitly bounded by the following divisibility condition [45,98].

$$\frac{\operatorname{lcm}(T_x, T_y)}{\gcd(T_x, T_y)} \mid T_z \mid \operatorname{lcm}(T_x, T_y)$$

whose right side is known at least since Selmer's work [106]. As a consequence, $T_z = T_x T_y$ if and only if the two periods are relatively prime [38].

3.2.3 Products of periodic sequences. Now consider the termwise product of two sequences (Herlestam [48] uses the term *Hadamard product* to distinguish the term-wise product of two sequences from the multiplication rule in the ring of formal power series).

$$\tilde{z} = \tilde{x} \wedge \tilde{y}$$
$$z_i = x_i \cdot y_i, \qquad i = 0, 1, 2, \ldots$$

It is a basic fact that the linear complexity of a product sequence can never exceed the product of the linear complexities of the sequences being multiplied [48,131]. Consequently, whenever the linear complexity of a product sequence satisfies this upperbound with equality, one says that this product sequence attains maximum linear complexity.

Considerable interest has been paid to the question of what conditions (in particular, what easily verfied conditions) have to be imposed on the minimal polynomials of the sequences such that the resulting product sequence will exhibit maximum linear complexity. At least since Selmer's work [106] it has been known that if the two minimal polynomials are irreducible and have relatively prime degree then the product sequence attains maximum linear complexity. In fact, Selmer provides a reference dated 1881 where this result appears without proof.

Herlestam [48,49] established the following necessary and sufficient condition on the roots of the involved minimal polynomials that would guarantee maximum linear complexity of the product sequence: $\Lambda(\tilde{z}) = \Lambda(\tilde{x}) \cdot \Lambda(\tilde{y})$ if and only if at most one of $m_{\tilde{x}}, m_{\tilde{y}}$ has multiple roots and the mutual product of their roots are all distinct. Unfortunately, this is not a readily verified condition.

Based on arithmetic properties of the periods T_x and T_y Rueppel and Staffelbach [99] developed the following condition: Let $m_{\tilde{x}}$ have no multiple roots and let $m_{\tilde{y}}$ *be* irreducible then the product sequence will attain maximum linear complexity if

$$\text{ord}(q) \bmod t_y = L_x$$

where $\text{ord}(q)$ mod t denotes the multiplicative order of q modulo t and t_y is defined as $T_y / \gcd(T_x, T_y)$. If both $m_{\tilde{x}}$ and $m_{\tilde{y}}$ are irreducible, then either $\text{ord}(q)$ mod $t_y = L_x$ or $\text{ord}(q)$ mod $t_x = L_y$ satisfies to guarantee maximum linear complexity. Golić [38] subsequently showed that this condition is the most general sufficiency condition expressed only in terms of the periods of the minimal polynomials. He also derived a second sufficiency condition (we refer to the original paper for the very involved formula) which is of some theoretical interest, since one of the two conditions has to be satisfied if the product sequence is to exhibit maximum linear complexity.

If it is easy to find divisors t of $q^L - 1$ that satisfy the condition $\text{ord}(q)$ mod $t = L$, it is also easy to achieve maximum linear complexity of the product sequence. One must ensure that the period of the first (irreducible) minimal polynomial contains t as a factor, and that the period of the second (not necessarily irreducible) minimal polynomial is relatively prime to f. Any integer $t > 1$ that divides $q^L - 1$ but is relatively prime to $q^i - 1$, $i < L$, is called a primitive factor of $q^L - 1$. The greatest such factor is denoted F_L. As an immediate consequence of this definition it follows that for any divisor t of F_L, the multiplicative order of q modulo t is L. Therefore maximum linear complexity is attained when t_y contains a primitive factor of $q^{L_y} - 1$. Interestingly enough, for all integers $q > 1$ and $L > 2$, every $q^L - 1$ contains a primitive factor, except for the single case $q = 2$ and $L = 6$. Moreover, the primitive factor F_L *can* actually be computed from the Lth cyclotomic polynomial. As a practical consequence, any product of N m-sequences will attain maximum linear complexity when the corresponding degrees are distinct and greater than 2 [99].

Finally, a less general but simpler condition in the sense that it only depends on the periods T_x and T_y is given in both [38,45]: the product sequence will attain maximum linear complexity if $\gcd(T_x, T_y) = 1$, (and provided the sequences are nonzero). The same condition also ensures maximum period of the product sequence, see [60, p. 435] or [106].

3.3 Functions of Periodic Sequences

From a practical viewpoint two situations are important: nonlinearly filtering the state of a LFSR (or, more generally, the state of any counting mechanism), and combining

the output sequences from several LFSRs. The former type of generator is referred to as a ***filter generator*** and the latter as a ***combination generator.*** It is obvious that the filter generator could be considered as a special case of the combination generator, where all the sequences combined satisfy the same recursion. But, since analysis and correlation attacks (see Section 3.4) are somewhat different, it is useful to consider each class separately.

Many of the published generator proposals (see Section 3.7) belong to one of the two classes. First, technology has provided some stimulus, since shift registers and combination components such as flip-flops, multiplexers, random access memories (RAMs), etc., were readily available. Second, the system-theoretic properties of these generators such as period, linear complexity, and statistics can frequently be analyzed by the feedforward nature of the nonlinearities. When considering these two classes of generators one may think alternatively of periodic sequences or LFSRs feeding the nonlinearities, since any periodic sequence has a finite and unique linear equivalent. A basic requirement for a generator is that it be unpredictable, which directly implies that the linear complexity of its output sequence should be large enough to prevent any attack based on the Berlekamp-Massey shift register synthesis algorithm [65].

There exists a canonical form for $\mathbb{F}_q$ switching functions, the so-called algebraic normal form [3], which directly reflects the sum of products property (see Section 3.1.3 for the description of the ANF). When a Galois switching function is applied to combine several $\mathbb{F}_q$ sequences, the resulting output sequence is formed by termwise combination of the sequence digits.

3.3.1 Filter generator. We proceed with the discussion of nonlinear combinations of different phases of one sequence. Consider the binary case $q = 2$.

Filter generator:

Input: parameters: one m-LFSR $< C(D), L >$
key: output function $f: \mathbb{F}_2^L \rightarrow \mathbb{F}_2$
initial state a_0 of the LFSR.

For $i = 1, 2, \ldots$ do

1. Shift LFSR (state transform F): $a_i = F(a_{i-1})$.
2. Compute $z_i = f(a_i)$

Output: the sequence of z_i, $i = 1, 2, \ldots$

Groth [44] concentrated on the use of second-order products which he applied to the stages of an LFSR with a primitive connection polynomial. No stage was allowed to be used more than once. To obtain reasonable statistics, Groth summed as many of the second-order products as possible. Higher-order nonlinearities were achieved by layering, that is, the sequences generated by summing second-order products were combined in the next layer by more second-order products, and so on. Groth was able to show the expected growth in linear complexity of the generated sequence as a function of the growing order of the nonlinearities. But he neglected to notice that maximum linear complexity is not always achieved. Hence, his results are only upper bounds on the true linear complexity.

It is well known that, for a shift register, or a counter, of period T and a sequence of period dividing T, there exists a function f that maps the counter into the sequence.

Consider a binary filter generator, its general linear equivalent will consist of all LFSRs whose characteristic polynomial divides $x^{2^L-1} - 1$. The $2^L - 1$ sequences that correspond to the $2^L - 1$ state bits of this general linear equivalent are linearly independent. It can also be shown [98] that the $2^L - 1$ sequences that correspond to the $2^L - 1$ coefficients in the ANF off are linearly independent. This suggests, at least in principle, a solution to the problem of generating functions that produce sequences of guaranteed linear complexity. First, pick the desired linear complexity by turning on state bits in the general linear equivalent, that is, by choosing a linear combination of the $2^L - 1$ basis sequences specified by the general linear equivalent. Then express the resulting sequence in terms of the ANF basis, which directly yields the required function f. But specifying the function f in general needs as much data as the first period of the sequence. Hence, the above procedure is impractical, and a more realistic goal is to find classes of functions with reasonable keyspace that generate large linear complexity when used in a filter generator.

Key [54] has shown that if f has order k then the linear complexity of $\tilde{z}$ is bounded from above by

$$\Lambda(\tilde{z}) \leq L_k = \sum_{j=1}^{k} \binom{L}{j}$$

But in the cryptographic application one is rather interested in generating sequences with guaranteed large mimimum linear complexity. For a subset of the class of bent functions, Kumar and Scholtz [57] were able to derive a lower bound that slightly exceeds

$$\Lambda(\tilde{z}) \geq \binom{L/2}{L/4} 2^{L/4}$$

where L is restricted to be a multiple of 4. Bernasconi and Gunther [5] and, independently, Rueppel [98] developed slightly different bounds whose combination yields the following lower bound: let $\tilde{z}$ be produced by any nonzero linear combination of N consecutive kth-order products $(k < L)$ of equidistant phases of the m-sequence $\tilde{s}$,

$$\tilde{z} = \sum_{j=0}^{N-1} c_j \tilde{s}^j \tilde{s}^{j+\delta} \cdots \tilde{s}^{j+(k-1)\delta}$$

where $\gcd(\delta, 2^L - 1) = 1$, then the linear complexity of $\tilde{z}$ has the lower bound

$$\Lambda(\tilde{z}) \geq \binom{L}{k} - (N - 1)$$

This lower bound remains valid when an arbitrary function f' is added to this sum of kth-order products, provided its order k' is smaller than k. As a practical consequence, one has at hand for implementation a large class of functions with guaranteed minimum linear complexity.

In [98] a probabilistic analysis is carried out for the case where a randomly selected kth-order function f is applied to the stages of a m-LFSR of prime length L. It is shown that the fraction P_n of filter generators with nonlinear order k producing

sequences of linear complexity L_k is lower bounded by

$$P_n \approx e^{-L_k/L2^L} > e^{-1/L}$$

It follows that the fraction of degenerate sequences approaches zero with increasing L.

3.3.2 Combination generator. Now consider the more general case where N periodic $\mathbb{F}_q$ sequences are combined in a memoryless output function $f : \mathbb{F}_q^N \rightarrow \mathbb{F}_q$. Restrict the algebraic normal form of f to powers $i \in \mathbb{Z}_2^N$.

Combination generator:

Input: parameters: N shift registers $< L_j, C_j(D) >$
nonlinear output function $f : \mathbb{F}_q^N \rightarrow \mathbb{F}_q$
key: N initial states $a_0^{(j)}$ of the N LFSRs.

For $i = 1, 2, \ldots$ do

1. For $j = 1, \ldots, N$ do
   shift $LFSR_j$.
   extract $a_i^{(j)}$.
2. $z_i = f(a_i^{(1)}, \ldots, a_i^{(N)})$

Output: the sequence of z_i, $i = 1, 2, \ldots$

Define the integer function $f^* : \mathbb{Z}^N \rightarrow \mathbb{Z}$ corresponding to f by

$$f^*(\mathbf{x}) = \sum c_{\mathbf{i}}^* \mathbf{x}^{\mathbf{i}}$$

where $c_{\mathbf{i}}^* = 0$ if $c_{\mathbf{i}} = 0$ and $c_{\mathbf{i}}^* = 1$ if $c_{\mathbf{i}} \neq 0$ [99]. From the discussion of sum and product sequences in the previous section it follows that the linear complexity and the period of the keystream produced by the combination generator are upperbounded by [48,99]

$$\Lambda(\bar{z}) \leq f^*(L_1, \ldots, L_N)$$
$$T_z \leq \mathrm{lcm}(T_1, \ldots, T_N)$$

Key [54] showed for the binary case that the upper bound can be reached when m-sequences of relatively prime degree are combined in f.

In [98] a set of conditions is developed each of which guarantees maximum linear complexity. The two that have most practical relevance are:

1. If the N $\mathbb{F}_q$–LFSRs employed in the combination generator are chosen to have primitive connection polynomials $C_j(D)$ and different lengths L_j, then the keystream will exhibit maximum linear complexity $\Lambda(\bar{z}) = f^*(L_1, \ldots, L_N)$.
2. If the N $\mathbb{F}_2$– ''LFSRs'' employed in the combination generator are chosen to be filter generators of pairwise relatively prime lengths L_j, then the keystream will exhibit maximum linear complexity $\Lambda(\bar{z}) = f^*(\Lambda_1, \ldots, \Lambda_N)$ where Λ_j denotes the linear complexity of the jth filter generator.

Note that with arrangement 2 one can generate sequences with a linear complexity that is exponential in the number of storage cells employed. In [99] a simple generator arrangement is given that produces with 150 memory cells a keystream whose linear complexity is at least 2^{146}.

In [38] it is shown that based on the pairwise coprimality of the periods (see Section 3.2.3) a lower bound on the linear complexity can be derived. Let the N GF(q)-sequences be nonzero and have relatively prime periods, then

$$\Lambda(\tilde{z}) \geq f^*(\Lambda_1 - 1, \ldots, \Lambda_N - 1)$$

where Λ_j represents the degree of the jth minimal polynomial.

Now consider the power function $f: \mathbb{F}_q \rightarrow \mathbb{F}_q$, $q = p^m$, defined by $f(x) = x^e$ where e is fixed and lies between $0 \leq e \leq q - 1$. Herlestam [48,49] showed that the linear complexity of $\tilde{z} = f(\tilde{x}) = \tilde{x}^e$ (computed termwise) is upperbounded by

$$\Lambda(\tilde{x}^e) \leq \prod_k \binom{\Lambda(\tilde{x}) + e_k - 1}{e_k}$$

where e_k are the digits of the p-ary expansion $e = e_0 + e_1 p + \ldots + e_{m-1}p^{m-1}$. Brynielsson [12] proved that the upper bound is reached if $\tilde{x}$ is a maximum-length sequence of degree $\Lambda(\tilde{x}) = L$. When the characteristic $p = 2$, the formula becomes particularly easy,

$$\Lambda(\tilde{x}^e) = L^{W_H(e)}$$

where $W_H(e)$ denotes the Hamming weight of e. This result generalizes to the case of an arbitrary polynomial $f(x) = \sum_{i=0}^{q-1} a_i x^i$. In [12] it is shown that for $p = 2$ and a maximum-length sequence $\tilde{x}$ of degree L,

$$\Lambda(f(\tilde{x})) = \sum_{i:a_i \neq 0} L^{W_H(i)}$$

Chan and Games [18] studied a similar problem. Let $\tilde{s}$ be a maximum-length sequence of degree L with digits from $\mathbb{F}_q$, q odd, and let $f: \mathbb{F}_q \rightarrow \mathbb{F}_2$ be an arbitrary mapping (the nonlinear feedforward function). Consider the binary sequence $\tilde{z} = f(\tilde{s})$. Chan and Games showed that the resulting sequences have linear complexity

$$\Lambda((\tilde{z})) = \frac{q^L - 1}{q - 1} \Lambda(\tilde{u})$$

where $\tilde{u}$ denotes the binary sequence obtained by applying the defining mapping f to a listing of the nonzero elements of $\mathbb{F}_q$ according to the powers of some primitive element. Because the sequences can be viewed as being obtained from finite geometries, they are termed binary geometric sequences. In [19] the autocorrelation and the cross-correlation functions of such binary geometric sequences is studied, including the case q even.

Now consider an arbitrary function $f: \mathbb{F}_q^2 \rightarrow \mathbb{F}_q$ whose ANF is a polynominal in two variables:

$$f(x, y) = \sum_{(i,j) \in \mathbb{Z}_q^2} a_{ij} x^i y^j$$

Brynielsson also proved [12] that for $p = 2$, $\tilde{x}$ and $\tilde{y}$ maximum-length sequences of degree L_1 and L_2, respectively, the linear complexity of $\tilde{z} = f(\tilde{x}, \tilde{y})$ satisfies

$$\Lambda(f(\tilde{x}, \tilde{y})) = \sum_{(i,j):a_{ij}\neq 0} L_1^{W_H(i)} L_2^{W_H(j)}$$

The filter and the combination generator are special classes of keystream generators. We refer the reader to Section 3.7 for a list of specific generators and the corresponding system-theoretic analysis.

3.4 Correlation Attacks

Correlation attacks have been proposed for two principal keystream generator arrangements, the combination generator consisting of a set of shift registers and a nonlinear output function f [8,77,112,128], and the filter generator consisting of a shift register with nonlinear state filter f, [30,111] (see Section 3.3). Interestingly, a structure identical to the combination generator was used in an interplanetary ranging system that was developed at the Jet Propulsion Laboratory (JPL) under the guidance of S. W. Golomb in the late 1950s [42,116]. The objective then was to design a Boolean combiner for shift register sequences of short pairwise relatively prime periods to produce a sequence whose period was the product of the component periods and which was highly correlated with each component sequence to facilitate the calculation of range. This ranging problem is virtually dual to the cryptographic problem, where one desires to combine a set of shift register sequences in such a way that the resulting sequence has least possible correlation with the component sequences. Blaser and Heinzmann [8] noticed that correlation methods could be used as a means to attack combination generators with "poorly" chosen combiners f (i.e., combiners that leak information about the component sequences). Their work in the cryptographic setting was independent of the much earlier work at the JPL. Siegenthaler [110] then developed the basic correlation attack (described in Section 3.4.1), thereby spurring much research activity. A first objective was to improve the efficiency of the algorithms and to enlarge their applicability. The general importance of correlation-immunity as a design criterion for combiners was recognized.

The applicability of the correlation attacks depends first on the choice of the output function f, and second on the parameters of the LFSRs.

3.4.1 Correlation attacks on combination generators. Consider a binary combination generator (see Section 3.3) and assume that $f : \mathbb{F}_2^N \to \mathbb{F}_2$ leaks information about the jth shift register sequence $\tilde{a}^{(j)}$ into the keystream $\tilde{z}$. Modeling the input to f as sequences of independent and uniformly distributed binary random variables, and taking into account the memoryless nature of f, one may compute the probability $p_j = P(f(A_1, \ldots, A_N) = A_j)$; p_j approximates the probability that the keystream digit coincides with a digit from the shift register sequence $\tilde{a}^{(j)}$. To isolate the effect of the jth LFSR on the keystream, one can model the rest of the combination generator as a binary symmetric channel (BSC) with error probability $1 - p_j$. Thus, conceptually, one considers the keystream as a distorted version of the shift register sequence $\tilde{a}^{(j)}$.

The problem now reduces to finding the correct phase $a_0^{(j)}$ from a finite string of keystream z^n and from the redundancy contained in $\tilde{a}^{(j)}$ (the linear relations governing

the behavior of $\tilde{a}^{(j)}$). Siegenthaler [112] developed and analyzed the (basic) correlation attack (here denoted as algorithm A). It is assumed that the cryptanalyst is in possession of a complete description of the combination generator, except for the key.

Algorithm A: Basic correlation attack on combination generator [112]

Input: description of combination generator $< \{L_i, C_i(D)\}, f >$
segment of keystream z^n of length n

For $j = 1, 2, \ldots, N$ do

1. Compute leakage (correlation) probability $p_j = P(z = a^{(j)})$ from function f assuming that N BSS's form the input to f.
   If $p_j = \frac{1}{2}$ abort attempt on LFSR_j and continue with j = j + 1.
2. Correlation phase: for $d = 1$ up to at most $2^{L_j} - 1$ do
   a. Compute cross-correlation function

$$C_{\tilde{a}^{(j)},\tilde{z}}(d) = \frac{1}{n} \sum_{i=1}^{n} (-1)^{z_i}(-1)^{a_{i+d}^{(j)}}$$

   b. If $C_{\tilde{a}^{(j)},\tilde{z}}(d)$ is greater than some suitable threshold T go to the verification step.
   c. Verify d by correlating an additional keystream segment $z'^{n'}$ with the corresponding segment from $\tilde{a}_d^{(j)}$.
   d. If test is successful assume d_j is correct phase and proceed with j + 1st LFSR, otherwise proceed with $d = d + 1$.

Ouput: the set of initial states $\{a_0^{(j)}\}$ for those LFSRs that leak information into the keystream.

For the attack to be successful a certain amount of keystream is needed and a suitable threshold T has to be chosen. Define $P_f = P(C_{\tilde{a}^{(j)},\tilde{z}}(d) \geq T \mid \text{incorrect phase})$ to be the probability of a false alarm. Define $P_m = P(C_{\tilde{a}^{(j)},\tilde{z}}(d) < T \mid \text{correct phase})$ to be the probability of a miss. P_m can be chosen freely, for example, $P_m = 0.01$, according to the risk one is willing to undergo in missing the correct phase. To minimize the number of verification steps in algorithm A choose $P_f = 2^{-L}$. The threshold T and the needed number n of keystream digits then follow from these choices [112]. For example, for a leakage probability $p = 0.75$, a probability $P_m = 0.01$ of missing the correct phase, a shift register length $L = 41$ at most $n = 355$ digits of keystream are needed with a threshold $T = 0.394$. A natural method is to use the maximum likelihood test to minimize $P_m + 2^L P_f$ which is approximately the probability of error [13]. Then, for the given values, the result is a threshold $T = 0.408$ with $P_m = 0.022$ and $P_f = 0.017 \cdot 2^{-41}$. Another possibility is to consider the k most correlated phases. An approximation for the probability that the correct one is among the selected ones can be found in [14]. For the given values, for $n = 300$ digits of keystream one needs to look at the 1000 largest correlation values to have a success probability 0.98 of finding the correct one.

The computational effort to run the algorithm A is an average of $\sum_{j=1}^{N} 2^{L_j}$ (provided f leaks) which compares favorably with the effort $\prod_{j=1}^{N} 2^{L_j}$ required by an exhaustive search. Such attacks that sequentially solve for the individual subkeys residing in

the shift registers are also referred to as "divide and conquer" attacks. A well-designed generator must not succumb to divide and conquer attacks. Algorithm A is only applicable when the shift register lengths L_j are small, and when the nonlinear combiner f leaks information about individual input variables. For $L_j \geq 50$ algorithm A becomes infeasible.

Algorithm A attacks the **subkey** residing in LFSR_j by exhausting all its possible phases. But not all phases are equally likely; for instance, by the very fact that there is correlation between $\tilde{a}^{(j)}$ and $\tilde{z}$, phases of $\tilde{a}^{(j)}$ that have smaller Hamming distance to $\tilde{z}$ are more likely than phases with high Hamming distance. One improvement [77] over algorithm A that may be pursued at this point is to try to select those digits in z^n that have high probability of being undistorted copies of the corresponding digits in a^n, and correlate with all phases that are close to those highly reliable digits. The process of estimating which digits may be unaltered is based on the linear recursion satisfied by $\tilde{a}^{(j)}$. Let t denote the number of taps of the LFSR that produces $\tilde{a}^{(j)}$. Then every sequence digit a, participates exactly $t + 1$ times in the linear recursion implemented by this LFSR:

$$a_j = c_1 a_{j-1} \oplus c_2 a_{j-2} \oplus \ldots \oplus c_L a_{j-L}$$

namely, at those positions where $c_k \neq 0$, inclusively at $c_0 = 1$. More linear relations involving a_i can be found by allowing polynomial multiples of the feedback polynomial $C(D)$. In particular,

$$(C(D))^{2^t} = C(D^{2^t})$$

a transformation that preserves the number of taps t. This property is important since the feasibility of the correlation attack strongly depends on a small number of taps. Suppose that a total of m linear relations is at disposal. A fixed digit a in the sequence $\tilde{a}^{(j)}$ must satisfy all m relations,

$$LR_k = a \oplus A_k = 0 \qquad k = 1, \ldots, m$$

where each A_k is the sum of t different terms of $\tilde{a}^{(j)}$. After passing $\tilde{a}^{(j)}$ through the BSC with error probability $1 - p$ the linear relations become distorted:

$$LR_k = z \oplus Z_k \qquad k = 1, \ldots, m$$

where now each Z_k is the sum of t different terms of $\tilde{z}$ and not necessarily sums to zero with z. For the considered digit z count the number h of linear relations LR_k that are satisfied. Conditioned on the event that h linear relations are satisfied the a priori probability $p = P(z = a)$ can be modified into the a posteriori probability p^*. When $z = a$ then a higher number h of satisfied relations is to be expected than for the case $z \neq a$. Or alternatively, the probability distribution of h given $z = a$ can be distinguished from the probability distribution of h given $z \neq a$ if sufficient keystream is available. Consequently, the basic correlation algorithm can be improved by adding an estimation phase [77] that first determines a set of highly reliable digits with the aim of reducing the work of the computationally expensive correlation phase.

Algorithm B: Estimation/correlation attack on combination generator [77]

Input: description of combination generator $< \{L_j, C_j(D)\}, f >$
n digits of keystream z^n

For $j = 1$ to N do

1. Compute leakage (correlation) probability $p_j = P(z = a_j)$ from function f assuming that N BSS's form the input to f.
   If $p_j = \frac{1}{2}$ abort attempt on $\mathbf{LFSR}_j$ and continue with $j = j + 1$.
2. Estimation phase: find set of highly reliable digits in z^n
   a. Compute expected number m of $t + 1$ digit relations available to test z^n

$$m = (t_j + 1) \log \left(\frac{n}{2L_j} \right)$$

   b. Select $h \leq m$ in such a way that at least L_j digits of z^n are expected to satisfy h or more linear relations.
   c. For any digit of z^n compute a posteriori correlation probability p_j^* given that h out of m relations for z^n are satisfied. Select those linearly independent L_j digits that have highest probability p_j^* as a reference guess I_0 for a_j^n.
3. Correlation phase: detect and correct remaining errors
   a. Compute the corresponding initial state $a_0^{(j)}$ and test its correctness by correlation of the resulting LFSR-sequence $(a^{(j)})^n$ with z^n.
   b. If the correlation indicates that the initial state $a_0^{(j)}$ is incorrect, systematically modify the initial guess by adding correction patterns of Hamming weight 1, **2** . . to I_0 and return to the verification step until the correct initial state $a_0^{(j)}$ is' found.

Output: the set of initial states $a_0^{(j)}$, $j = 1, \ldots, N$ for those LFSRs that leak information to the output of the keystream generator.

The computational effort of this algorithm is dominated by the correlation phase. The more errors were made in the estimation phase, the more initial points a_0 have to be tested in the correlation phase. In the extreme, the algorithm degenerates to an exhaustive search over all initial phases of the LFSR under investigation; but still a slight improvement over algorithm A remains by starting the correlation with the most probable initial states, instead of a random initial state. In general, algorithm B has computational complexity $O(2^{cL})$ with $0 \leq c \leq 1$; c depends on the leakage probability p, the number t of taps, and on the ratio n/L. Algorithm B is successful if a long enough segment of keystream z^n is available, if the number t of taps is small, $t \leq 10$, and if the LFSR substantially leaks into the keystream, $p > 0.6$.

On the other hand, a clear limit to algorithm B exists: when $t > 16$ and $p \leq 0.75$ then no improvement over exhaustive search is gained. As a consequence, algorithm B suggests concrete design rules for combination generators:

1. Choose f such that it is kth-order correlation-immune (see Section 3.5, below).
2. Choose long LFSRs ($L \geq 100$) with connection polynomials that have many taps ($t \geq 10$).

In algorithm B only the digits having high probability of being correct were selected and processed. In addition, one can also make use of digits that have a high

probability of being distorted. Another improvement over algorithm A that may be pursued [77,128] is to try to correct the errors in the distorted "codeword" z^n to get back the undistorted "codeword" a^n thereby recovering the information symbols a^L (the initial state); in this second approach computation of the cross-correlation function is no longer necessary. In [128] a method is proposed where those digits in z^n are complemented that satisfy less than a certain number of parity-check equations (linear relations). This threshold is set to about $m/2$ when m is the total number of parity-check equations. After the correction step the procedure is iterated. This approach bears similarity to Gallager's iterated decoding algorithm [32].

In [77] another algorithm is proposed based on the idea of iterating the process of assigning a posteriori probabilities p^* several times before complementation of those digits whose probability of being correct is below a suitable threshold. This threshold is chosen to yield optimal correction effect. Then the whole process is restarted with the original probability p reassigned to every digit. In the end the algorithm should converge to the correct solution a^n. The behavior of these iterative algorithms has not been fully analyzed, one problem being that the correction step introduces statistical dependencies in the remaining error pattern. But in terms of applicability they result in similar design rules as algorithm B. We refer the reader to [77,128] for a full account.

3.4.2 Correlation attacks on filter generators. Consider a binary filter generator (see Section 3.3). To extend a given keystream segment z^n systematically, the cryptanalyst may try to determine the used key directly, or he may try to determine an equivalent system that produces the same keystream. A specific filter generator could always be simulated by a period $(2^L - 1)$ counter and a suitably chosen output function. Note that this argument implies that any m-LFSR of length L with appropriately chosen output function is able to generate the keystream. Another approach to determining an equivalent system could be the linear equivalent generator for the keystream. In both approaches the cryptanalyst does not make use of his knowledge of the connection polynomial used in the original system. Taking into account the connection polynomial the original system can be simulated by a combination generator with m-LFSRs all possessing the same known primitive connection polynomial and a suitably chosen nonlinear combining function g. Such equivalent systems always exist; when k denotes the number of taps for f then put simply the k phases tapped by f into separate shift registers and set $g = f$ to obtain a trivial equivalent system. Occasionally, it may happen that $m < k$. Consider the periodic cross-correlation function.

$$C_{\tilde{a},\tilde{z}}(d) = \frac{1}{T}\sum_{i=0}^{T-1}(-1)^{z_i}(-1)^{a_{i-d}}$$

between the driving register sequence $\tilde{s}$ and the keystream $\tilde{z}$. In [11 1] an analytic expression for $C_{\tilde{a},\tilde{z}}$ is derived involving the Walsh transform $\hat{F}$ off(x) = (- 1)$^{f(x)}$

$$C_{\tilde{a},\tilde{z}}(d) = 2^{-k}\left(1 + \frac{1}{T}\right)\hat{F}(v(d)) - \frac{\hat{f}(0)}{T}$$

where v(d) denotes the linear combination of the L stage sequences that produces the dth shift $\tilde{s}(d)$ of $\tilde{s}$. Inspection of the above expression shows that a peak in the-cross-correlation function can only occur if $|\hat{F}(v(d))|$ is large. But the Walsh transform $F(v(d))$

is zero for all linear combinations $v(d)$ that select a state sequence that is not tapped by f. Thus, there are at most $2^{l} - 1$ peaks in the cross-correlation function. Of these at most k are at positions d_i such that the corresponding sequences $\tilde{s}(d_i)$ are linearly independent. In principle one could directly compute the filter function f by inverting the Walsh transform F whose values are known from the cross-correlation function $C_{\tilde{a},\tilde{z}}$. But in practice this is infeasible since it would require to know and computationally handle a full period of keystream. Thus, in practice one must compute the cross-correlation with a short segment of n bits of keystream which introduces a certain amount of noise. Depending on the segment length n, the number of taps k, and the properties of the filter function f it may be that the relevant peaks of the cross-correlation $C_{\tilde{a},\tilde{z}}$ disappear in the noise. The following algorithm suggested in [111] attempts to construct an equivalent generator from a set of peaks observed in the cross-correlation whose positions correspond to linearly independent sequences $\tilde{s}(d_i)$.

Algorithm: Cryptanalyst's representation of filter generator [111]

Input: description $< C(D), L >$
keystream segment z^n of length n

1. Compute cross-correlation function for all d between $0 \leq d \leq 2^L - 2$.

$$C_{\tilde{a},\tilde{z}}(d) = \frac{1}{n} \sum_{i=0}^{n-1} (-1)^{z_i}(-1)^{a_{i-d}}$$

2. Keep a list with the L strongest values of $C_{\tilde{a},\tilde{z}}(d)$.
3. Select a basis $\tilde{a}(d_1), \ldots, \tilde{a}(d_m)$ of dimension $m \leq n$ for the sequences $\tilde{a}(d_1)$, $\tilde{a}(d_2)$, . . . corresponding to the positions $d_1, d_2, \ldots$ of the observed peaks.
4. Compute g from the m input sequences $\tilde{a}(d_1), \ldots, \tilde{a}(d_m)$ and the corresponding output sequence $\tilde{z}$.

Output: the set of m linearly independent initial states $\{a_0^{(d_j)}\}$ and the function $g : \mathbb{F}_2^m \rightarrow \mathbb{F}_2$.

It is an open problem when the above attack will be successful, since for any other m-LFSR of length L there exists a corresponding function f' that generates the same keystream as the original arrangement [98]. Thus, one might think that correlation with any maximum-length shift register of length L would produce another usable equivalent system. However, practice has shown that the method works only for small functions (small number of taps n) with pronounced correlation between output and linear combinations of inputs. Since the above algorithm involves correlation with a significant part of the LFSR period, it is only feasible when $L < 50$ [111].

3.5 Nonlinearity Criteria

The analysis of Boolean functions with respect to their suitability as cryptographic transformations is based on the following model: Let $f, g : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ be Boolean functions and let X be a random variable that takes on values $x \in \mathbb{F}_2^n$ with uniform probability 2^{-n}. We are interested in describing the statistical dependencies of the random

variables Z = $f(X)$ and Y = g(X) in terms of their producing functions f and g. The cross-correlation function off and g is defined as

$$C_{fg}(t) = \sum_x (-1)^{f(x)}(-1)^{g(x \oplus t)}$$

Note that the correlation coefficient $c(f,$ g), as defined in [78], satisfies

$$c(f, g) = 2^{-n}C_{fg}(0)$$

On the other hand,

$$c(f, g) = P(Y = Z) - P(Y \neq Z)$$

Hence, it follows that

$$P(Y = Z) = \frac{1}{2} + \frac{c(f, g)}{2}$$

Suppose g is a (nonconstant) linear function g(x) = $\omega \cdot$ x = $\omega_1 x_1 \oplus \ldots \oplus \omega_n x_n$. Such linear functions will be denoted $L,(x)$. Then

$$c(f, L_\omega) = 2^{-n} \sum_x (-1)^{f(x)}(-1)^{\omega x} = 2^{-n}\hat{F}(\omega)$$

It is easy to see that Z = $f(X)$ and $L,(X)$ are statistically independent if and only if $c(f, L,)$ = 0, or equivalently, if and only if $F(o)$ = 0. To prevent divide and conquer correlation attacks on combination generators, Siegenthaler [110] introduced the notion of correlation-immunity. A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is defined to be mth order correlation-immune if Z = $f(X)$ is statistically independent of any subset $X_{i_1}, X_{i_2}, \ldots, X_{i_m}$ of m input variables. Xiao and Massey [124] showed that this condition is equivalent to the condition that Z = $f(X)$ is statistically independent of the sum

$$\omega_1 X_1 \oplus \omega_2 X_2 \oplus \ldots \oplus \omega_n X_n$$

for every choice of $(\omega_1, \ldots, \omega_n) \in \mathbb{F}_2^n$ such that $W_H(\omega) \leq m$. The proof of the above condition was subsequently simplified by Brynielsson [14] using the following argument. Consider the Walsh transform of the conditional probability distribution $p(x \mid z)$ for a subset $X_1, \ldots, X_m$ of m binary variables

$$\begin{aligned}\sum_{x \in \mathbb{F}_2^m} p(x \mid z)(-1)^{\omega \cdot x} &= E[(-1)^{\omega \cdot X} \mid Z = z] \\ &= E[(-1)^{\omega \cdot X}] \\ &= \sum_{x \in \mathbb{F}_2^m} p(x)(-1)^{\omega \cdot x}\end{aligned}$$

Hence, $p(x \mid$ z) and $p(x)$ are identical since their Walsh transforms are identical. Using the correlation coefficient $c(f, L,)$ it follows that a Boolean function f is mth-order correlation-immune if and only if

$$\hat{F}(\omega) = 0 \qquad (\forall \omega : 1 \leq W_H(\omega) \leq m)$$

This characterization of correlation-immune functions in terms of their Walsh transforms was developed by Xiao and Massey [124].

Siegenthaler [110] showed that there exists a trade-off between the attainable nonlinear order k and the attainable level of correlation immunity m. A memoryless mth-order correlation-immune function, for $1 \leq m < n$, can attain at most nonlinear order $n - m$. For balanced f the attainable nonlinear order is $n - m - 1$ unless $m = n - 1$. Xiao and Massey slightly extended these results by showing that for any mth-order correlation-immune function all coefficients of $(n - m)$th-order product terms in the algebraic normal form off must be equal, that is, they either all are present or all vanish. When f is required to be balanced, then they all vanish, as was already noted [110]. Since the ANF transform and the Walsh transform are both linear, their composition is a linear map which directly computes the algebraic normal form off from the Walsh coefficients. The upper bound on the attainable order of an mth-order correlation-immune function then follows from the vanishing Walsh coefficients and the structure of the Walsh-to-ANF-transform.

Siegenthaler addressed the problem of constructing balanced output mth order correlation immune combining functions with maximum nonlinear order $n - m - 1$. His recursive procedure is based on the fact that two mth-order correlation-immune functions f_1 and f_2 of n variables can be combined into a mth-order correlation-immune function f of $n + 1$ variables,

$$f(x_1, \ldots, x_{n+1}) = x_{n+1} f_1(x_1, \ldots, x_n) \oplus (x_n \oplus 1) f_2(x_1, \ldots, x_n)$$

If the product terms in f_1 and f_2 do not coincide completely, then the nonlinear order of f is equal to the larger of the two orders of f_1 and f_2 plus 1.

In [98] it was shown that the Walsh transform can be used to find the best affine approximation of a given Boolean function f. As can be seen from the correlation coefficient $c(f, L_\omega)$ this problem reduces to finding the largest Walsh transform coefficient $F(\omega)$. If $L_\omega = \omega \cdot X$ correlates positively with $Y = f(X)$ then $\omega \cdot x$ is the best approximation, otherwise $\omega \cdot x \oplus 1$ is the best approximation. There is an interesting connection to coding theory. Finding the best affine approximation to a function f is formally equivalent to maximum likelihood decoding, when the codewords are the 2^{n+1} affine function descriptions, and the information bits are the $n + 1$ coefficients of the affine function. In fact, this code is known as a Reed-Muller code of order 1 [64, p. 373].

The result was also extended [98] to solve the problem of finding the best affine approximation to a set $\mathcal{F} = \{f_1, f_2, \ldots, f_M\}$ of Boolean functions of n input variables. For instance, in a keystream generator part of the key may be devoted to selecting a specific nonlinear combiner from a set of suitable ones. In a first step compute the Walsh transforms $\hat{F}_i(\omega)$ of the M functions. Then sum the Walsh transforms componentwise $\hat{F}_{\mathcal{F}}(\omega) = \sum_{i=1}^{M} \hat{F}_i(\omega)$ (it is assumed that the f_i are selected with equal probability in the application). Find that nonzero ω that maximizes $|\hat{F}_{\mathcal{F}}(\omega)|$ and denote it by ω_a. If $\hat{F}_{\mathcal{F}}(\omega_a)$ is positive then $L_{\omega_a}(x) = \omega_a \cdot x$ is the best linear approximation, otherwise $1 \oplus \omega_a \cdot x$ is the best affine approximation.

Application of the above approximation technique to the S-boxes used in the DES showed that (1) linearizing the S-box mappings typically results in success rates of 75% up to 87.5% (a balanced Boolean function of four variables always has an affine approximation with success probability $p \geq \frac{3}{4}$, see below), with slightly better results for the inverse mappings; (2) approximating the four S-box mappings jointly by a single affine transformation may produce high success rates, occasionally even up to 81%.

The objective in the design of correlation-immune functions is to prevent correlation with linear functions L_ω, especially those for which $W_H(\omega)$ is small. Now consider

the total correlation of a Boolean function f with the set of all linear functions [78], defined as

$$C^2(f) = \sum_{\omega} c^2(f, L_\omega) = 2^{-2n} \sum_{\omega} \hat{F}^2(\omega)$$
$$= 2^{-n} \sum_{x} f^2(x)$$
$$= 1$$

where we have made use of Parseval's theorem. The total correlation is independent of the function f, in fact it is constant. Hence, there are limits to how much correlation-immunity can be designed into a Boolean function f. The larger the subset of linear functions L_ω for which we require the correlation coefficient to vanish, the stronger will be the correlation to the remaining linear functions. In the extreme, maximum correlation-immunity of order $n - 1$ is only satisfied by the (linear) function $x_1 \oplus x_2 \oplus \ldots \oplus x_n$ (or its complement), thereby introducing maximum correlation to the sum of all variables.

In [97] it was investigated how memory can help to address the problem of designing correlation-immune combiners. It was shown that finite-state combiners can decouple the requirements for correlation immunity and nonlinearity. Combiners of the form

$$z_i = \sum_{j=1}^{n} x_{j,i} \oplus g(\sigma_{i-1})$$
$$\sigma_i = F(\sigma_{i-1}, \mathbf{x}_{i-1})$$

where σ_i denotes the internal state of the combiner at time i, attain maximum order $n - 1$ of correlation immunity. The functions F and g can be designed to satisfy any suitable nonlinearity criterion, such as maximum nonlinear order, without being restricted by the desired order of correlation immunity. In particular, 1 bit of memory is sufficient to decouple the requirements. It is also shown that the (bitserial) integer addition of n integers inherently defines a maximum-order correlation-immune finite-state combiner. The input sequences can be interpreted as binary representations of integer numbers starting with the least significant bit first. Computing the ith bit of the sum requires all the ith bits of the input sequences and the carry from lower positions of the sum. Keeping the carry information requires some finite memory.

In [79] the total correlation of finite-state combiners with 1 bit of memory is investigated. Since the output digit z_i of the finite-state combiner at time i depends on at most the digits $x_{j,0}, x_{j,1}, \ldots, x_{j,i}$ for $1 \leq j \leq n$, there must be correlation to linear functions of the form

$$\sum_{k=0}^{i} \sum_{j=1}^{n} \omega_{j,k} x_{j,k}$$

There are $2^{n(i+1)}$ such functions. Now consider a general combiner with 1 bit of memory with balanced output and next-state functions $f_0, f_s : \mathbb{F}_2^{n+1} \rightarrow \mathbb{F}_2$

$$z_i = f_0(x_{1,i}, \ldots, x_{n,i}, \sigma_{i-1})$$

$$\sigma_i = f_s(x_{1,i}, \ldots, x_{n,i}, \sigma_{i-1})$$

Let a = $(\omega_0, \omega_1, \ldots, \omega_{n-1}, 0)$ and $\beta = (\omega_0, \omega_1, \ldots, \omega_{n-1}, 1)$ denote linear combinations that explicitly exclude and include the state σ. Decompose the total correlation $C^2(f_0)$ into

$$C^2(f_0) = \sum_{\alpha} c^2(f_0, L_\alpha) \sum_{\beta} c^2(f_0, L_\beta)$$
$$= C_0^2(f_0) + C_1^2(f_0) = 1$$

Equivalently, $C^2(f_s)$ may be decomposed into $C_0^2(f_s) + C_1^2(f_s)$. Then it is proved in [79] that the total correlation of the output digit z_i of the 1-bit combiner with the $N = 2^{n(m+1)}$, $1 \le m \le i$, linear functions of the form

$$L_{\omega,m} = \sum_{k=i-m}^{i} \sum_{j=1}^{n} \omega_{j,k} x_{j,k}$$

satisfies

$$\sum_{n=1}^{N} c_n^2 = C_0^2(f_0) + C_1^2(f_0)[1 - (C_1^2(f_s))^m]$$

Note that this total correlation converges to 1, except in the singular case $C_0(f_s) = 0$ where the limit is $C_0^2(f_0)$. Hence, in the nonsingular case the total correlation is independent of the functions f_0, f_s employed in the 1-bit combiner. This generalizes the corresponding result for (memoryless) combining functions. When LFSR-sequences are used to drive the 1-bit combiner then nonzero correlation to functions $L_{\omega,m}$ leads to nonzero correlation to sums $\Sigma_{j=1}^{n} s_j$ of LFSR-sequences $s_j = \Sigma_{k=i-m}^{i} \omega_{j,k} x_{j,k}$ where s_j is a different phase of the jth LFSR. The best one can do in this situation is to design the 1-bit combiner to be maximum order correlation-immune [98]. This condition corresponds to requiring that for all nonvanishing correlation coefficients there is no correlation to a sum of less than N LFSR-sequences [79]. The summation combiner [97] for any number n of input sequences is an example of such a maximum-order correlation-immune finite-state combiner.

Meier and Staffelbach recently [78] developed a framework for nonlinearity criteria of Boolean functions. They classified nonlinearity criteria, such as nonlinear order, correlation immunity, and best affine approximation, with respect to their suitability for cryptographic design. They argued that a function has to be considered weak whenever it can be turned into a cryptographically weak function by means of simple (e.g., affine) transformations, and that a criterion is only useful if it is invariant under such a group of transformations. The distance to sets of cryptographically weak functions appears to be a fundamentally useful measure to express and analyze nonlinearity criteria.

The *distance* between two binary functions f and g is defined as

$$d(f, g) = |\{x \in \{0, 1\}^n : f(x) \ne g(x)\}|$$

that is, as the Hamming distance between the two function tables. The distance of f to a linear function L_ω is closely related to the correlation coefficient $c(f, L_\omega)$

$$d(f, L_\omega) = 2^{n-1}(1 - c(f, L_\omega))$$

and may equivalently be expressed in terms of the Walsh transform $\hat{F}(\omega)$

$$d(f, L_\omega) = 2^{n-1} - \frac{1}{2}\hat{F}(\omega)$$

The maximum distance to an affine function is 2^{n-1}, since $d(f, \omega \cdot x) = d$ implies $d(f, \omega \cdot x \oplus 1) = 2^n - d$. Consequently, a Boolean function f is mth-order correlation-immune if its distance $d(f, L_\omega) = 2^{n-1}$ for all linear functions $L_\omega(x) = \omega \cdot x$ with $W_H(\omega) \leq m$. But as with the total correlation it is impossible to achieve maximum distance to all linear functions. Let the distance $d(f, S)$ of f to a set of functions S be defined as

$$d(f, S) = \min_{g \in S} \{d(f, g)\}$$

that is, the distance between f and the nearest neighbor off in S. As a consequence, the distance off to the set $\mathcal{A}$ of affine functions is given as

$$d(f, \mathcal{A}) = 2^{n-1} - \frac{1}{2}\max_{\omega} \{|\hat{F}(\omega)|\}$$

Hence, the distance $d(f, \mathcal{A})$ may be upperbounded if we can give a lower bound on $\hat{F}^2(\omega)$. Using Parseval's theorem one finds

$$\sum_{\omega} \hat{F}^2(\omega) = 2^{2n}$$

Therefore, there exists an ω for which $\hat{F}^2(\omega) \geq 2^n$. We conclude that for even n and any Boolean function f

$$d(f, \mathcal{A}) \leq 2^{n-1} - 2^{\frac{n}{2}-1}$$

Similarly, it may be shown that for even n and any balanced Boolean function f

$$d(f, \mathcal{A}) \leq 2^{n-1} - 2^{\frac{n}{2}-1} - 2$$

It was shown in [78] that $d(f, \mathcal{A})$ satisfies the invariance property, that is, $d(f, \mathcal{A})$ remains invariant under the group of all affine transformations. Another set of potentially weak functions is the set $\mathcal{L}$ of functions with linear structures. A Boolean function f has a ***linear structure*** [20, 28] at $a \in \mathbb{F}_2^n$ if there exists $ab \in \mathbb{F}_2^n$ such that for all $x \in \mathbb{F}_2^n$

$$f(x) = f(x \oplus a) \quad \text{or} \quad f(x) \neq f(x \oplus a)$$

Affine functions have a linear structure for all $a \neq 0$. But a function with a linear structure is not necessarily affine. Consider as an example the nonlinear function $f(x) = x_1x_2 \oplus x_2 \oplus x_2x_3$ which has a linear structure for $a = (1, 0, 1)$. In [78] it was proved that

$$d(f, \mathcal{L}) \leq 2^{n-2}$$

and it was shown that $d(f, \mathcal{L})$ also satisfies the invariance property, that is, $d(f, \mathcal{L})$ remains invariant under the group of all affine transformations.

Motivated by the definition of functions with linear structures, Meier and Staffelbach [78] defined a function to ***be perfect nonlinear*** (with respect to linear structures) if

$$(\forall a \neq 0) \qquad |\{x \in \{0, 1\}^n : f(x) = f(x \oplus a)\}| = |\{x \in \{0, 1\}^n : f(x) \neq f(x \oplus a)\}| = \frac{1}{2}2^n$$

that is, if for every nonzero vector a the values $f(x)$ and $f(x \oplus a)$ agree for exactly half the arguments x and disagree for the other half. Equivalently, f is perfect nonlinear if the autocorrelation function $C_f(a)$ satisfies

$$C_f(a) = \sum \hat{f}(x \oplus a)\hat{f}(x) = 0 \qquad (\forall a \neq 0)$$

Invoking the Wiener-Khintchine theorem (which states that $C_f(a)$ and the spectrum $\hat{F}^2(\omega)$ are Walsh transform pairs) yields the alternative characterization of perfect nonlinear functions in the transform domain

$$\hat{F}^2(\omega) = 2^n$$

Therefore, a ± l-valued Boolean function is perfect nonlinear if and only if $|\hat{F}(\omega)| = 2^{n/2}$ for all ω. But this is exactly the defining property of ***bent*** functions as introduced by Rothaus [93] in combinatorial theory. It is shown that the class of perfect nonlinear (bent) functions is optimum with respect to both distances; for an even number of arguments n, the perfect nonlinear (bent) functions simultaneously have maximum distance $2^{n-1} - 2^{n/2-1}$ to all affine functions as well as maximum distance 2^{n-2} to all functions with linear structure. Rothaus shows [93] that bent functions only exist for an even number n of arguments, and that their nonlinear order is bounded by n/2. He gives explicit constructions for bent functions:

1. Let $n = 2m$ and g be an arbitrary function; then functions of the form

$$f(x_1, \ldots, x_n) = g(x_1, \ldots, x_m) \oplus x_1x_{m+1} \oplus \ldots \oplus x_mx_n$$

are bent.

2. Let $x = (x_1, \ldots, x_n)$ and let $a(x)$, $b(x)$ and $c(x)$ be bent functions such that $a(x) \oplus b(x) \oplus c(x)$ is also bent. Then the function

$$f(x, x_{n+1}, x_{n+2}) = a(x)b(x) \oplus a(x)c(x) \oplus b(x)c(x) \oplus [a(x) \oplus b(x)]x_{n+1} \oplus [a(x) \oplus c(x)]x_{n+2} \oplus x_{n+1}x_{n+2}$$

is also a bent function.

A more general construction for Boolean bent functions was given by Maiorana (see [27,86]). Let $g : \mathbb{F}_2^n \to \mathbb{F}_2$ be any function and let $\pi : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be any bijective transformation. Then the function $f : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2$ defined by

$$f(x_1, x_2) = \pi(x_1) \cdot x_2 \oplus g(x_1)$$

is a binary bent function. An alternative characterization of bent functions may be given in terms of combinatorial structures called difference sets [27]. A function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is bent if and only if

The set $S_1 = \{x : f(x) = 1\}$ is a difference set in $\mathbb{F}_2^n$ with parameters $(2^n, 2^{n-1} \pm 2^{\frac{n}{2}-1}, 2^{n-2} \pm 2^{\frac{n}{2}-1})$. (That is, there are $2^{n-1} \pm 2^{\frac{n}{2}-1}$ elements in S_1 and each nonzero $x \in S_1$ can be expressed in $2^{n-2} \pm 2^{\frac{n}{2}-1}$ ways as a difference $x = y \oplus z$ of elements $y, z \in S_1$.)

Constructions of difference sets have been known for a long time. In [76] a construction is given which is equivalent to Maiorana's construction. Generalized bent functions from $\mathbb{Z}_q^n$ to $\mathbb{Z}_q$, q a positive integer, are investigated by Kumar, Scholtz, and Welch [58]. Subsequently, Nyberg [86] considered the cryptographic properties of such generalized bent functions. Nyberg extended the notion of perfect nonlinearity to the q-ary case: a function $f : \mathbb{Z}_q^n \to \mathbb{Z}_q$ is perfect nonlinear if for all nonzero $a \in \mathbb{Z}_q^n$ and $b \in \mathbb{Z}_q$,

$$|\{x : f(x) = f(x + a) + b\}| = q^{n-1}$$

Clearly a perfect nonlinear function is bent. But a bent function is perfect nonlinear, only if q is prime. It can be shown that bent (perfect nonlinear) functions $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ exist for every odd prime p and every positive integer n. Nyberg [86] considers in particular the value distributions of p-ary bent functions, p prime, and their Hamming distances to the set of affine functions. The construction of (Boolean) bent functions has intrigued researchers for quite some time. The number of Boolean bent functions remains an open problem. In [126] a list of known constructions is given. In [91] a construction based on concatenation of rows of the Hadamard matrix H_m is given which was subsequently shown to be equivalent to Maiorana's construction (Nyberg).

Webster and Tavares [122] introduced the notion of ***strict avalanche criterion*** (SAC) when they studied S-boxes (see also [23]). A Boolean function satisfies the strict avalanche criterion if any output bit changes with probability exactly $\frac{1}{2}$ on complementation of one input bit. That is, if

$$(\forall a : W_H(a) = 1) \qquad P(f(X) = f(X \oplus a)) = \frac{1}{2}$$

Equivalently, the function f satisfies the SAC if the autocorrelation function vanishes at all arguments of Hamming weight 1

$$(\forall a : W_H(a) = 1) \qquad C_f(a) = 0$$

By the Wiener–Khintchine theorem the inverse Walsh transform of $\hat{F}^2(\omega)$ must satisfy

$$(\forall a : W_H(a) = 1) \qquad \sum_{\omega} \hat{F}^2(\omega)(-1)^{\omega \cdot a} = 0$$

This is exactly Forré's spectral characterization of Boolean functions satisfying the SAC [29]. Forré then extended the basic notion by defining the SAC of order m. A function f satisfies the SAC of order m [29,61] if and only if any partial function obtained from f by keeping m of its inputs constant satisfies the SAC (for any choice of the positions and of the values of the constant bits). In [61] it is proved that there are 2^{n+1} functions of n variables that satisfy the SAC of maximum order $n - 2$.

Perfect nonlinear (bent) functions also satisfy the strict avalanche criterion. In fact, the definition of the SAC corresponds to the definition of perfect nonlinearity restricted to offset vectors a of Hamming weight 1. Thus perfect nonlinearity is a stron-

ger requirement than the SAC. In [91] a construction for all 2^{n+1} functions that satisfy the SAC of order $n - 2$ is given. Noting the similarity in the definitions of perfect nonlinearity and SAC, Preneel et al. [91] subsequently proposed a nonlinearity criterion that encompasses botn. They defined a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ to satisfy the ***propagation criterion of degree k*** if

$$(\forall a : 1 \leq W_H(a) \leq k) \qquad P(f(X) = f(X \oplus a)) = \frac{1}{2}$$

Let $PC(k)$ denote the class of functions satisfying the propagation criterion of degree k. Equivalently, f is in $PC(k)$ if

$$(\forall a : 1 \leq W_H(a) \leq k) \qquad C_f(a) = 0$$

or, by invoking the Wiener–Khintchine theorem, if

$$\sum_{\omega : 1 \leq W_H(a) \leq k} \hat{F}^2(\omega)(-1)^{\omega \cdot a} = 0$$

Note that for Boolean functions SAC corresponds to $PC(1)$ and that perfect nonlinearity (bentness) corresponds to $PC(n)$.

There are some consequences for the design of keystream generators. For a nonlinearly filtered LFSR, the keystream is correlated to at most 2" phases of the same LFSR with a constant sum $\Sigma c^2 = 1$. Thus, zero correlation to some phases necessarily implies higher correlation to other phases. The best one can do is to uniformly minimize the cross-correlation, which is done by choosing f as close as possible to a perfect nonlinear function. For a memoryless nonlinear combiner of n sequences from N different LFSRs, the keystream is correlated to at most 2" linear combinations of phases from N shift registers. A divide and conquer attack is possible when there is a correlation with a linear combination of phases from less than N shift registers. Thus, the best one can do [78,97] is to choose f to be maximum-order correlation-immune, such that there is only correlation with sums of LFSR phases involving all N shift registers. The remaining unavoidable correlation can be uniformly minimized. When an application demands balanced functions, Meier and Staffelbach [78] recommend to randomly select a bent function (if there is one) and to complement an arbitrary set of $2^{(n/2)-1}$ ones (zeros) in the function table. The resulting balanced function is called nearly perfect. When other design criteria have to be met in an application, they recommend to systematically generate a random perfect nonlinear function, and then to search for the closest function also satisfying those other criteria.

3.6 Clock-Controlled Shift Registers

One technique that has attracted designers of keystream generators ever since is to irregularly clock certain parts of the generator in order to achieve nonlinear effects. Recently, Gollmann and Chambers published a comprehensive survey on clock-controlled shift registers [41]. We distinguish two classes of clock control techniques applicable to shift registers: forward clock control and feedback clock control.

3.6.1 Forward clock control. Basic forward clock control refers to the situation where one regularly clocked shift register is used to control the clock of another shift register.

Basic clock-controlled generator:

Input: parameters: a control register $< L_1, C_1(D) >$ with period T_1
a generating register $< L_2, C_2(D) >$ with period T_2
mapping $f : \mathbb{F}_2^{L_1} \rightarrow \mathbb{Z}_{T_2}$
key: initial states x_0, y_0 of the two registers
and, possibly, the function f

For $i = 1, 2, \ldots$ do

1. Clock LFSR, once and extract $f(\mathbf{x}_i)$.
2. Clock $\mathbf{LFSR}_2$ $f(\mathbf{x}_i)$ times and extract resulting output $y_{\sigma(i)}$, where $\sigma(i)$ denotes the accumulated sum of all clock pulses that have reached $\mathbf{LFSR}_2$ at time i.
3. Assign $z_i = y_{\sigma(i)}$

Output: the sequence of z_i, i = 1, 2, . . .

The operation of this generator is governed by

$$z_i = y_{\sigma(i)} \qquad \sigma(i) = \sum_{k=1}^{i} f(x_k)$$

Depending on the realization of the function f one distinguishes different types of forward clock-controlled generators. If $f(\mathbf{x}_i) = x_i$, that is, if the current output bit of the control register directly is fed to the clock input of the generating register, the generator is called a stop-and-go generator [6,119]. This generator has poor cryptologic and statistical properties. Whenever the keystream bit changes value from time i to time i + 1 one knows that the control register has emitted a 1. Moreover, the generator has strong intersymbol dependency since half the time (depending on the number of zeros in the control sequence) the previous keystream symbol is copied to the next position. The stop-and-go generator can be improved by simply adding a 1 to the control sequence, that is, $f(\mathbf{x}_i) = 1 + x_i$, thereby transforming the generator into a step-once-twice generator. For the binary rate multiplier generator [16] f takes on the form

$$f(\mathbf{x}_i) = 1 + \sum_{j=0}^{n-1} x_{i-j} 2^j \qquad n \leq L_1$$

An early investigation of the period of clock-controlled generators was done by Tretter [117]. The control register produces a constant number of ones and zeros counted over its period. Consequently, the number of clock pulses that reach the generating register during one period of the control register is constant. Define

$$S = \sigma(T_1) = \sum_{k=1}^{T_1} f(\mathbf{x}_k)$$

After $\mathrm{lcm}(S, T_2)$ clock pulses have been applied to the generating register both registers are back in their starting positions. Hence, the generator is periodic with T, $(\mathrm{lcm}(S, T_2)/S) = T_1 T_2 / \gcd(S, T_2)$. If $\gcd(S, T_2) = 1$ then it can be shown that the overall period reaches its maximum value $T = T_1 T_2$. For the remaining discussion of forward clock control it is assumed that the condition $\gcd(S, T_2) = 1$ is always satisfied.

An early investigation of the linear complexity achievable with clock controlled generators was done by Nyffeler [87]. An upper bound on the linear complexity of the keystream can be derived as follows. Decimate the keystream by T_1, that is, consider the set of sequences

$$\{z_{j+kT_1}\}_k = \{y_{\sigma(j+kT_1)}\}_k = \{y_{\sigma(j)+kS}\}_k$$

for $1 \le j \le T_1$. Each of these T_1 sequences is an S-decimation of the regular output sequence $\tilde{y}$ of the generating register. Let $C_s(D)$ denote the connection polynomial that can produce all these S-decimations. Then the polynomial $C_s(D^{T_1})$ can produce the keystream for all choices of parameters and keys. Hence, the linear complexity of the keystream is upperbounded by $\Lambda(\tilde{z}) \le L_2T_1$. For the stop-and-go generator the upper bound is reached if the control register produces an m-sequence, that is, if $T_1 = 2^{L_1} - 1$, and if $\gcd(L_1, L_2) = 1$ [6].

One way of proving that the upper bound is reached is to prove that the generating polynomial of the keystream $C_s(D^{T_1})$ is irreducible. Suppose $C_S(D)$ of degree L is irreducible and has order T. If, for any prime factor p of T_1, holds that p divides T but not $(2^L - 1)/T$ then $C_s(D^{T_1})$ is irreducible. In particular, the degree of $C_s(D^{T_1})$ is LT_1. Versions of this result can be found in [16,87,107,113,119]. For instance, the stop-and-go generator and the step-once-twice generator achieve maximum linear complexity if both the control and the generating register produce the same m-sequence of period $T = 2^L - 1$ [119], since then the keystream has period T^2 and linear complexity TL. For the special case $T_1 = 2^n$ Günther derived a "tight" lower bound [46] $\Lambda(\tilde{z}) > \frac{1}{2}L_2T_1$ which holds provided $C_2(D)$ is irreducible. When the generating register of the step-once-twice generator produces an m-sequence then it can be shown that all l-tuples of length $l \le (L_2 + 1)/2$ appear in the output sequence with the same frequency as in the original m-sequence $\tilde{y}$. For delays bounded by $T_2/2$ the magnitude of the out-of-phase autocorrelation function is bounded by $1/T_2$.

Golić and Zivković [37] consider the following generalization of the basic clock-controlled generator. Let the generating register be a m-LFSR, and let the control register be a pure cycling register of length T_c that is randomly filled with digits $0 \le d_i \le 2^{L_2} - 1$. At time i the digit d_i specifies how many steps the generating register has to take. The sequence $\tilde{d}$ is called the difference decimation sequence. It is shown that maximum linear complexity L_2T_c of the output sequence can be reached only if the multiplicative order of 2 modulo $T_2/\gcd(S, T_2)$ is equal to L_2. Furthermore, when the difference decimation sequence is assumed to be chosen at random and uniformly, a lowerbound on the probability that a decimated m-sequence has maximum linear complexity T_cL_2 is established. This bound can be made arbitrarily close to 1 for appropriate choices of the parameters T_c and L_2.

To improve the bad statistics of the stop-and-go generator while maintaining maximum speed Gunther proposed the alternating step generator [46]. This generator consists essentially of two stop-and-go generators that share the same control register; the corresponding output sequences are then added to produce the keystream.

Alternating step generator:

Input: parameters: a control register $< L_c, C_c(D) >$ with period T_c
a pair of generating registers $< L_1, C_1(D), L_2, C_2(D) >$
key: initial states $\mathbf{x}_0$, $\mathbf{y}_0^{(1)}$, $\mathbf{y}_0^{(2)}$ of the three registers

For $i = 1, 2, \ldots$ do

1. Clock $LFSR_1$ once and produce x_i
2. If $x_i = 1$ then shift $LFSR_1$ and produce $y^{(1)}_{\sigma(i)}$
   If $x_i = 0$ then shift $LFSR_2$ and produce $y^{(2)}_{i-\sigma(i)}$
3. Set $z_i = y^{(1)}_{\sigma(i)} \oplus y^{(2)}_{i-\sigma(i)}$

Output: the sequence of z_i, $i = 1, 2, \ldots$

For the analysis in [46] the control register is assumed to produce a de Bruijn sequence of period $T_c = 2^k$. If $C_1(D)$ and $C_2(D)$ are different and irreducible, and if $\gcd(T_1, T_2) = 1$ then the period of the keystream is maximal, $T = 2^k T_1 T_2$, and the linear complexity of the keystream is bounded, $(L_1 + L_2)2^{k-1} < \Lambda(\vec{z}) \leq (L_1 + L_2)2^k$. More generally, Günther has also shown that the conditions $\gcd(S, T_1) = \gcd(T_c - S, T_2) = \gcd(T_1, T_2) = 1$ and $T_1, T_2 \geq 2$ are sufficient to guarantee a maximal period $T = T_c T_1 T_2$. Using a result from [46] that states the sum of two sequences $\vec{x} \oplus \vec{x}'$ has linear complexity at least $L + L' - 2\gcd(T, T')$, one obtains the lower bound $\Lambda(\vec{z}) \geq (L_1 + L_2 - 2)T_c$ for the general case.

If the two generating registers regularly produce m-sequences, then the frequency of all tuples of length $l \leq \min\{L_1, L_2\}$ is 2^{-l} up to a deviation of order $O(1/2^{L_1 - l}) + O(1/2^{L_2 - l})$ that is, these frequencies are close to ideal. A similar result holds for the autocorrelation function when the delays are restricted to values $|\tau| \leq T_c - 1$.

It was pointed out in [46] that the alternating step generator succumbs to a divide and conquer attack with respect to the control register. With a keystream segment of about $4L_c$ a search through all phases of the control register will reveal the correct one. With knowledge of the state of the control register the rest of the key is easily recovered. Thus, assuming that all registers have comparable lengths, the effective keysize of the alternating generator is at most the third root of the number of possible keys.

In a natural extension, the basic clock-controlled generator may be used to control the clock of a third shift register, its output may then control the clock of a fourth shift register, and so forth. This arrangement is referred to as a cascade generator [17,39,55,117,119].

Cascade generator:

Input: parameters: a basic shift register $< L, C(D) >$ with period T_0
the number N of stages, each consisting of a basic shift register
key: initial states $\mathbf{s}_0^{(n)}$, $n = 1 \ldots N$

For $i = 1, 2, \ldots$ do

1. Shift stage 1 and produce $y_i^{(1)}$.
2. For $n = 2 \ldots N$ do
   Shift the nth stage $y_i^{(n-1)}$ times and produce

$$y_i^{(n)} = y_i^{(n-1)} \oplus s^{(n)}_{\sigma_{n-1}(i)}$$

$$\sigma_{n-1}(i) = \sum_{k=1}^{i} y_i^{(k-1)}$$

Alternatively, one may shift the nth stage $y_i^{(n-1)} + 1$ times

3. Set $z_i = y_i^{(N)}$

Output: the sequence of z_i, $i = 1, 2, \ldots$

Two arrangements have been analyzed: the m-sequence cascade [41,55,117,119] and the p-cycle cascade [17,39,40]. In the m-sequence cascade the basic shift register produces a maximum-length sequence of period $T_0 = 2^L - 1$. For each stage a different primitive polynomial may be chosen. It can be shown that the immediate output sequence of the shift register in the nth stage has period T_0^n and linear complexity LT_0^{n-1}. The keystream then is the sum of all the immediate outputs and has period T_0^N and linear complexity $1 + L(T_0 + \ldots + T_0^{N-1})$ that is greater than LT_0^{N-1}.

In the p-cycle cascade the basic shift register consists of a pure cycling register of prime length p. If one assumes that each register is loaded with a state of even parity then the minimal polynomial of a ***p-cycle*** register can be at most $C_p(D) = 1 + D + \ldots + D^{p-1}$. Now $C_p(D)$ is irreducible if and only if 2 is a primitive element in GF(p). Such primes are referred to as 2-primes [17]. Artin has conjectured [56] that about 37% of the primes are 2-primes. If p^2 does not divide $2^{p-1} - 1$ then the linear complexity of the p-cycle cascade is p^N, exactly identical to the period. If there is a register whose state has odd parity it can be shown that the linear complexity can at most decrease by one, that is, $p^N - 1$. In [40] a curious effect is described: Suppose one knows the sequence produced by the basic shift register but not the corresponding stage phases, then it is possible to unravel the correct phases stage by stage. This effect has been termed the ***lock-in*** effect. For the p-cycle cascade the effective key space is reduced from $(2^p - 2)^n$ to $((2^p - 2)/p)^n$ which may be serious when p is small.

3.6.2 Feedback clock control. In [100] the idea of a shift register that controls its own clock is developed. Whenever the output symbol is a zero, d clock pulses are applied to the LFSR, and, in case the output symbol is a 1, k clock pulses are applied to the LFSR. The effect on the output sequence is that of an irregular decimation; correspondingly, the resulting sequence is termed a [d, k]-self-decimated sequence.

[d, k]-Self-decimation generator:

Input: parameters: a shift register $< L, C(D) >$ with period T_0
stepping rule $f : \{0,1\} \rightarrow \mathbb{Z}_{T_0}$
key: initial state $\mathbf{y}_0$

For $i = 0, 1, 2, \ldots$ do

1. Extract $z_i = y_{\sigma(i)}$

2. Apply $f(y_{\sigma(i)})$ clock pulses to LFSR and update a(i)

$$\sigma(i + 1) = \sigma(i) + f(y_{\sigma(i)})$$

Output: the sequence of z_i, $i = 0, 1, 2, \ldots$

Clearly the [d, k]-self-decimated sequence generator is a singular device, that is, the state diagram will in general contain (one or more) cycles and tails. Depending

on the initial state there may be a preperiod in the self-decimated sequence. Assume that the shift register produces a maximum-length sequence, that is, $T_0 = 2^L - 1$. Then there are 2^{L-2} states with no predecessor in the state diagram provided only that $d \neq k$. This result immediately establishes a universal upper bound on the period of the generator, $T \leq (3/4)(2^L - 1)$. If d is a unit modulo T_0 then the general self-decimation rule can be separated, $[d, k] = [d][1, k']$, into a constant decimation followed by a $[1, k']$-decimation. It can be shown that the state sequence of the self-decimation generator has period

$$T_L = \left\lfloor \frac{2}{3}(2^L - 1) \right\rfloor$$

for all self-decimation rules $[d, k] = g[1, 2] \bmod (2^L - 1)$ with $\gcd(g, 2^L - 1) = 1$. A similar result holds for $[d, k] = g[1, 2^{L-1}] \bmod (2^L - 1)$ with $\gcd(g, 2^L - 1) = 1$. The distribution of zeros and ones in a self-decimated sequence is balanced and the same is true for l-tuples of short lengths. Experimental data on the linear complexity of $[1, 2]$-self-decimated sequences is consistent with the lower bound 2^{L-1} but no proof of this bound exists.

It is obvious that the self-decimation generator could not be used directly as a keystream generator since any of its output digits directly reveals a state bit, and, when the stepping rule is known, the position of the next state bit to be copied to the output sequence. A simple improvement suggested by **Günther** is to use different elements of the shift register sequence for the feedback clock control and the output. If both d and $k \geq 2$ then

$$z_i = y_{\sigma(i)+1} \qquad \sigma(i+1) = \sigma(i) + f(y_{\sigma(i)})$$

will not reveal the position of the next keystream digit.

Chambers and Gollmann [17] suggested a modification of the self-decimation generator which allowed them to find cases in which the output sequence achieves maximum linear complexity. Let, as before, $z_i = y_{\sigma(i)}$ and clock the shift register regularly unless the condition

$$y_{\sigma(i)}, \ldots, y_{\sigma(i)-(z-1)} = \underline{0}$$

is satisfied. Then step twice. For $L - z$ even, it can be shown that the self-decimation generator has period $T = 2^L - 1 - (2/3)(2^{L-z} - 1)$. An exhaustive search was carried out to find pairs (L, z) that result in 2-prime values for the period T. If T is a 2-prime then $C_p(D) = 1 + D + \ldots + D^{T-1}$ is irreducible and the output sequence has linear complexity T or $T - 1$. Such a modified self-decimation generator could then be used as basic stage in a p-cycle cascade, since for large values of p pure cycling registers become unfeasible.

3.7 Generators

In this section we describe a collection of generators that have been proposed and published in the open literature. Those generators that utilize clock control such as

1. Basic clock-controlled generator
2. Alternating step generator

3. Cascade generator
4. Self-decimation generator

are described separately in Section 3.6. Some of the following generators are specific implementations of the generic arrangements described in Section 3.3.

3.7.1 Geffe's generator. In 1973 [34] Geffe proposed as a basic building block a 3-LFSR keystream generator which employed the nonlinear combining function $f : \mathbb{F}_2^3 \rightarrow \mathbb{F}_2$ defined by

$$f(x_1, x_2, x_3) = x_3 \oplus x_1x_2 \oplus x_2x_3$$

The lengths L_j and the feedback polynomials $C_j(D)$ are parameters of choice, subject to the constraints that the lengths are pairwise relatively prime and the feedback polynomials are primitive.

Geffe's generator:

Input: parameters: 3 LFSRs $< L_j, C_j(D) >$
key: initial states $\mathbf{a}_{j0}$ of the three LFSRs.

For $i = 1, 2, \ldots$ do

1. For $j = 1, \ldots, 3$ do
   shift LFSR_j.
   extract a_{ji}.
2. $z_i = a_{3i} \oplus a_{1i}a_{2i} \oplus a_{2i}a_{3i}$

Output: the sequence of z_i, $i = 1, 2, \ldots$

The system-theoretic analysis shows that the overall period $T = T_1T_2T_3$ is maximized. The linear complexity has value $\Lambda(\bar{z}) = L_3 + L_1L_2 + L_2L_3$. Although the combination principle yields good statistical results and large linear complexity it is cryptographically weak. The function f leaks information about the states of LFSR 1 and 3 into the keystream, since $P(f(X) = X_1) = P(f(X) = X_3) = \frac{3}{4}$. Later it was noted that the function $f' : \mathbb{F}_2^3 \rightarrow \mathbb{F}_2$ defined by

$$f'(x_1, x_2, x_3) = x_1x_2 \oplus x_1x_3 \oplus x_2x_3$$

obtained by a slight modification off, exhibits equally good distribution properties, but allows one to generate sequences with higher linear complexity. For a discussion of this function, see Section 3.7.4, below.

3.7.2 Pless generator. The proposal of this generator [90] appears to be technology-driven, that is, the availability of the *J-K* flip-flop circuits seems to have stimulated the construction. In algebraic normal form, a *J-K* flip-flop implements the function

$$y_j = x_j^{(1)} \oplus y_{j-1}(1 \oplus x_j^{(1)} \oplus x_j^{(2)})$$

where y_j denotes the internal state at time j. Pless noted that knowledge of the current output bit of the *J-K* flip-flop identifies both the source and the value of the next output bit. In order to remedy this weakness Pless proposed to suppress alternate bits of the output sequence.

The generator consists of eight driving LFSRs and four *J-K* flip-flops each of which act as a nonlinear combiner for a separate pair of LFSRs. The four flip-flop output sequences are decimated by 4 and then interleaved to yield the keystream. The lengths L_j and the feedback polynomials C,(D) are parameters of choice.

Pless generator:

Input: parameters: 8 LFSRs $< L_j, C_j(D) >$,
key: initial states $\mathbf{a}_0^{(1)}, \ldots, \mathbf{a}_0^{(8)}$ of the 8 LFSRs.

For $i = 0, 1, 2, \ldots$ do

1. Fork= 1,. . . , 4 do
   a. Shift each LFSR
   b. Compute kth *J-K* flip-flop function for corresponding pair of LFSRs

$$y_{4i}^{(k)} = a_{4i}^{(2k-1)} \oplus y_{4i-1}^{(k)}(1 \oplus a_{4i}^{(2k-1)} \oplus a_{4i}^{(2k)})$$

   c. Collect four keystream bits as $z_{4i+k} = y_{4i}^{(k)}$

Output: the sequence of z_l, 1 = 1, 2, . . .

When the periods of the eight LFSRs are chosen to be pairwise relatively prime then the overall period of the keystream is their product. Rubin [94] subsequently noticed that a "divide and conquer" strategy can be used to attack each of the four subgenerators independently, and he developed a known-plaintext attack which is successful with as little as 15 characters. Moreover, and more fundamentally, the *J-K* flip-flop as a combination principle is cryptographically weak. The information-theoretic analysis off under the assumption of an unknown and random state bit and independent and uniformly distributed binary input variables yields a nonzero mutual information $I(Z;$ X,) = $I(Z; X_2)$ = 0.189 bits for either of the two input variables. Consequently, this generator succumbs to attacks as described in Section 3.4.

3.7.3 Multiplexer generator. The proposal of this generator [51,52] appears to be technology-driven, that is, the availability of multiplexer circuits seems to have stimulated the construction. The generator consists of two driving LFSRs and a multiplexer as nonlinear combiner f. The multiplexer, controlled by the state of LFSR, , selects at each time instant one stage of LFSR,. The content of this stage then forms the current term of the keystream. The number h of control inputs to the multiplexer must satisfy $1 \leq h \leq L$, and $2^h \leq L$,. *The* lengths L_j, the feedback polynomials $C_j(D)$, and h are parameters of choice.

Multiplexer generator:

Input: parameters: 2 LFSRs $< L_j, C_j(D) >$,
h and control vector j = $(j_0, j_1, \ldots, j_{h-1})$ such that
$0 \leq j_0 \leq j_1 < \ldots < j_{h-1} \leq L_1$.
key: initial states $\mathbf{s}_0^{(1)}, \mathbf{s}_0^{(2)}$ of the 2 LFSRs.

For $i = 1, 2, \ldots$ do

1. Shift LFSR, and $\mathbf{LFSR}_2$
2. Compute the integer

$$a_i = \sum_{k=0}^{h-1} 2^k s_i^{(1)}(j_k)$$

3. Extract

$$z_i = s_i^{(2)}(\theta(a_i))$$

where θ is an invertible mapping from $\{0, 1, \ldots, 2^h - 1\}$ to $\{0, 1, \ldots, L_2 - 1\}$

Output: the sequence of z_i, $i = 1, 2, \ldots$

Assume that the two connection polynomials are primitive, and that the lengths are relatively prime, that is, gcd $(L_1, L_2) = 1$. Then it can be proved [51] that the period assumes the maximum value $T = (2^{L_1} - 1)(2^{L_2} - 1)$. The linear complexity can be upperbounded by

$$\Lambda(\vec{z}) \leq L_2\left(1 + \sum_{i=1}^{h} \binom{L_1}{i}\right)$$

if $2 \leq h < L_1 - 1$ with equality if all h stages are spaced at equal intervals. If $h = L_1 - 1$ or $h = L_1$ then $\Lambda(\vec{z}) = L_2(2^{L_1} - 1)$. It is also shown [52] that the periodic autocorrelation function $C(\tau) = -1/(2^{L_2} - 1)$ for most out-of-phase values of τ, $1 \leq \tau \leq T - 1$.

3.7.4 Threshold generator. This generator was proposed in 1984 [11] as a simple and efficient way of obtaining keystreams with guaranteed large linear complexity while maintaining good statistical properties. The generator consists of a set of M driving LFSRs and a nonlinear combination rule f. The number M of LFSRs, the lengths L_j, and the feedback polynomials $C_j(D)$ are parameters of choice, subject to the constraints that the lengths are pairwise relatively prime and the feedback polynomials are primitive.

Threshold generator:

Input: parameters: M LFSRs $< L_j, C_j(D) >$
key: initial state $\mathbf{a}_{10}, \ldots, \mathbf{a}_{M0}$ of the M LFSRs.

For $i = 1, 2, \ldots$ do

1. For $j = 1, \ldots, M$ do
   Shift $\mathbf{LFSR}_j$.
   Extract a_{ji}.
2. Compute the integer sum of the current output bits and decide

$$z_i = \begin{cases} 1 & \text{if } \left(\sum_{j=1}^{M} a_{ji}\right) > \dfrac{M}{2} \\ 0 & \text{otherwise} \end{cases}$$

Output: the sequence of z_i, $i = 1, 2, \ldots$

The output sequence will only be balanced if the number M of shift registers employed is odd. Consider the case $M = 3$, then the output mapping f can equivalently be written in algebraic normal form:

$$z_i = a_{1i}a_{2i} \oplus a_{1i}a_{3i} \oplus a_{2i}a_{3i}$$

which is the mod 2 sum of all second-order products. The system-theoretic analysis shows that the overall period $T = T_1T_2T_3$ is maximized. The linear complexity has value $\Lambda(\vec{z}) = L_1L_2 + L_1L_3 + L_2L_3$. Although the combination principle yields good statistics and large linear complexity it is cryptographically weak. The information-theoretic analysis off under the assumption of independent and uniformly distributed binary input variables yields a nonzero mutual information $I(Z; X_j)$ of 0.189 bits for every j. As a consequence, there is positive correlation between the keystream and every input sequence.

3.7.5 Inner product generator. In [67] it is shown that perfect linear cipher systems exist. The linear encryption transformation has the form

$$y_i = F_k(x_i, \ldots, x_{i-M}) = x_i + \sum_{j=1}^{M} c_j(i, k)x_{i-j}$$

where the coefficients $c_j(i, k)$ depend on both the time instant i and the key k. If the plaintext is restricted never to contain M consecutive zeros, a perfect linear cipher (where plaintext and ciphertext are statistically independent) can be obtained with two digits of key per plaintext digit. The construction makes use of the M-fold inner product of M consecutive plaintext digits with M consecutive key digits shifted at double speed. This perfect linear cipher suggests that the inner product of two LFSRs that are operated at different speeds may provide an interesting sequence generator. The lengths L_1, L_2, and the speed factors d_1, d_2 are parameters of choice.

Inner product generator:

Input: parameters: two LFSRs $< L_j, C_j(D) >$ and two speed factors d_1, d_2
key: initial states $\mathbf{a}_0^{(1)}, \mathbf{a}_0^{(2)}$ of the registers.

For $i = 1, 2, \ldots$ do

1. Shift LFSR, d_1 times.
2. Shift LFSR, d_2 times.
3. Compute the inner product of the 2 LFSR states

$$z_i = \sum_{k=1}^{min\{L_1, L_2\}} \mathbf{a}_{1k} \cdot \mathbf{a}_{2k}$$

Output: the sequence of z_i, $i = 1, 2, \ldots$

The system-theoretic analysis shows [67] that the output sequence will have linear complexity $\Lambda(\vec{z}) = L_1L_2$ provided $C_1(D)$ and $C_2(D)$ are irreducible, $\gcd(L_1, L_2) = 1$, $\gcd(d_1, T_1) = \gcd(d_2, T_2) = 1$, where T_1 and T_2 denote the periods of the original shift registers, and the shift registers are initially loaded with nonzero contents. The period T of $\vec{z}$ is lowerbounded by $T_z \geq T_1T_2/(q - 1)$. For the analysis of the global randomness properties, assume that $C_1(D)$ and $C_2(D)$ are primitive polynomials over $GF(2)$, and that $L_2 > L_1$. Then it can be proved that the number of zeros within a period is $(2^{L_2} - 1)(2^{L_1-1} - 1)$ and the difference between the number of ones and the number of zeros relative to the period length is $1/(2^{L_1} - 1)$. The inner product combination enforces "almost" ideal distribution properties of the output sequence. This generator hence also carries the potential of synthesizing LFSRs with irreducible but nonprimitive characteristic polynomials that possess a state cycle with good global randomness properties. The described generator may be cascaded in a natural way: combine the output sequence $\vec{z}$ in a second inner product with a third LFSR of length L_3; since $\vec{z}$ has no identifiable speed factor attached to it, it is sufficient to choose d_3 greater than 1.

3.7.6 Wolfram's cellular automaton generator. At Crypto'85 Wolfram [123] proposed to use a binary one-dimensional cellular automaton with nonlinear next-state function as sequence generator. A one-dimensional cellular automaton consists of a (possibly infinite) line of sites with values $a_i \in \mathbb{Z}_n$. These values are updated in parallel (synchronously) in discrete time steps according to a fixed rule of the form

$$a'_k = \Phi(a_{k-r}, a_{k-r+1}, \ldots, a_{k+r})$$

where r denotes the span of input arguments to Φ. Any practical implementation of a cellular automaton must contain a finite number of sites N. These are typically arranged in a circular register with periodic boundary conditions, that is, the cell indexes of the arguments of next-state rule Φ are computed mod N. In [123] the next-state update rule

$$a'_k = a_{k-1} \oplus (a_k \vee a_{k+1})$$

is singled out as the most promising among all three-argument rules Φ.

Wolfram's cellular automaton generator:

Input: parameters: number of cells N
(and possibly the next-state function f)
key: initial state $\mathbf{a(0)} = (a_0(0), a_1(0), \ldots, a_{N-1}(0))$.

For $i = 1, 2, \ldots$ do

1. Update cellular automaton using the rule

$$a_k(i) = a_{k-1}(i - 1) \oplus (a_k(i - 1) \vee a_{k+1}(i - 1))$$

for $k = 0, 1, \ldots, N - 1$, with the cell indexes being computed mod N.

2. Extract keystream from any single site k

$$z_i = a_k(i)$$

Output: the sequence of z_i, $i = 1, 2, \ldots$

For infinite N it is shown in [123] that all length S spatial sequences and all length T temporal sequences occur with equal probabilities, provided the probability distribution of initial configurations is uniform. Thus, the corresponding entropies are maximal. But the deterministic nature of the above cellular automaton rule implies that only certain space-time patterns of values can occur. In fact, knowing the temporal sequences of two adjacent sites allows the spatial reconstruction of the cellular automaton configuration. Similarly, all the site values in a particular space-time patch are completely determined by the values that appear on its upper, left, and right boundaries. It is shown that the entropy contained in the cellular automaton configuration per time unit is at most 1.2 bits. Hence knowledge of the time sequences of values of about 1.2 sites suffice in principle to determine the values of all other sites.

In addition to studying the global properties of cellular automata evolution, one may also look at the effects caused by local changes in the cellular automaton configuration. The form of the cellular automaton rule implies that a change in one site propagates to the right with 1 bit per time unit. Empirical measurements indicate that the rate of information propagation to the left is about $\frac{1}{4}$ bit per application of the cellular automaton rule [123].

Regarding the finite-size behavior, one may examine the state transition diagram of the N site circular cellular automaton. Typically the state transition diagram consists of a set of cycles, fed by trees representing transients. For the above generator it is shown that the number of states that have two (branch nodes) or zero predecessors (root nodes) tends to zero asymptotically with N. For large N, the state transition diagrams appear to be increasingly dominated by a single cycle. The actual maximal cycle lengths II, were computed for $N < 55$ [123]; for the determined range Π_N can be approximated by

$$\log_2 \Pi_N \approx 0.61(N + 1)$$

Note that for a random next-state mapping one would expect cycles of average length $2^{N/2}$. Empirical results indicate that all 2^S possible length S keystreams can be generated from any of the 2^N keys (initial states of the size N cellular automaton) provided $N \geq S + 2$. Finally, a battery of statistical tests was applied to the generator, which all showed no significant deviations from randomness.

3.7.7 $1/p$ generator. In [10] the l/p generator is proposed and analyzed. It is based on the state transformation $F(x) = bx$ mod N of a linear congruential generator. In the same paper the quadratic congruential generator whose state transformation is $F(x) = x^2$ mod N is analyzed and compared to the l/p generator. Let $b > 1$ denote a fixed base. The seed for the $1/p$ generator consists of an integer p that is relatively prime to b, and a random $x_0 \in Z_p^*$. Basically, the l/p generator expands the seed x_0/p to the base b, that is, it produces the sequence of b-ary quotient digits when x_0 is divided by p to the base b. In contrast, the linear congruential generator produces the sequence of b-ary remainder digits when x_0 is divided by p to the base b.

$1/p$ generator:

Input: parameters: base $b > 1$
key: integer p with $\gcd(p, b) = 1$, a random $x_0 \in Z_p^*$.

For $i = 1, 2, \ldots$ do

1. Compute next-state function

$$x_i = F(x_{i-1}) = bx_{i-1} \bmod p$$

2. Compute output function

$$z_i = f(x_{i-1}) = bx_{i-1} \operatorname{div} p$$

Output: the sequence of z_i, $i = 1, 2, \ldots$

The analysis in [10] shows that the period of the keystream reaches its maximum $T = p - 1$ for p prime and b a primitive root mod p. For this subclass of generators interesting statistical properties result. The corresponding keystreams can be proved to be generalized de Bruijn sequences of period $p - 1$. A generalized de Bruijn sequence has the property that every b-ary string of $|p| - 1$ digits appears at least once, and every b-ary string of $|p|$ digits appears at most once in a given period of the sequence ($|p|$ denotes the length of the b-ary expansion of p). These sequences resemble the maximum-length shift register sequences. Hence, it is not surprising that knowledge of the base b and of $k = \lceil \log_b(2p^2) \rceil$ digits $z_{m+1}, \ldots, z_{m+k}$ of the keystream is sufficient to predict the generator in both forward and backward direction. This is done by applying the continued fraction expansion algorithm to the fraction $z_{m+1} \ldots, z_{m+k}/b^k$. The convergent is guaranteed to be x_m/p if $1/b^k \leq 1/2p^2$. Since $\gcd(b, p) = 1$ it is also easy to extend the sequence in backward direction by computing $x_{i-1} = b^{-1}x_i \bmod p$.

3.7.8 Summation generator. In [97] the suitability of integer addition as a combining function is investigated. Let the input sequences be binary expansions of some integers (possibly infinite, if the sequences are assumed to be semi-infinite). The sum function $f : \mathbb{Z}^N \rightarrow \mathbb{Z}$ defined by $z = \Sigma_{i=1}^{N} x_i$ can be computed in bit-serial fashion starting with the least significant bit. The investigation was motivated by the fact that integer addition is highly nonlinear, when considered over $\mathbb{F}_2$, and at the same time provides the maximum order of correlation immunity. The following generator [97] is a simple application:

Summation generator:

Input: parameters: N LFSRs $< L_j, C_j(D) >$
key: initial states of the N LFSRs and carry C_0

For $i = 1, 2, \ldots$ do

1. Step each shift register once to produce $x_{1i}, x_{2i}, \ldots, x_{Ni}$.
2. Compute the integer sum

$$S_i = \sum_{k=1}^{N} x_{ki} + C_{i-1}$$

3. Set

$$z_i = S_i \bmod 2$$

$$C_i = \left\lfloor \frac{S_i}{2} \right\rfloor$$

Output: the sequence z_i, $i = 1, 2, \ldots$

The analysis in [97] shows that the real sum of N periodic sequences with r-ary digits is ultimately periodic with period $T = \prod T_i$ if the periods T_i of the individual sequences are **pairwise** relatively prime. When two binary m-sequences of relatively prime degrees are added over the reals then the sum sequence exhibits linear complexity close to its period length, $\Lambda(\tilde{z}) \leq (2^{L_1} - 1)(2^{L_2} - 1)$. If the input vectors are independent and uniformly distributed then the keystream will also consist of i.u.d. random variables. The real sum directly provides a $(N - 1)$**th-order** correlation-immune combination principle, which is the maximum order of immunity possible.

Subsequently, in [79] the correlation analysis of memoryless functions was extended to combiners with 1 bit of memory (see also Section 3.5, above). The summation generator with two input sequences is such a l-bit combiner **and** it was shown that there is positive correlation with $x_{1,i} \oplus x_{1,i-1} \oplus x_{2,i}$ (which is in accordance with result that the summation generator is maximum-order correlation-immune when fed with shift register sequences). In [115] the probability distribution of the carry for addition of random integers is analyzed. It is proved that the carry is balanced for even N and biased for odd N and that the bias converges to 0 as N tends to ∞.

3.7.9 Knapsack generator. In [96] the suitability of the knapsack as a nonlinear transformation is investigated. The knapsack problem, also known as a subset sum problem, is an NP-complete problem [33]. A problem instance consists of a finite set of L positive integer-valued weights and a positive integer S. The problem is to decide if there is a **subset** of the weights that sums exactly to S. For a given set of weights the knapsack may be viewed as a function from the selection vector $x = x_1, \ldots, x_L$ to the integer S which is the sum of those weights selected by x. The difficulty of the knapsack problem motivated its application as a nonlinear function in a filter generator [96]. Note that unlike the NP-complete problem the weights are kept secret in this application.

Knapsack generator:

Input: parameters: LFSR $< L,$ C(D) $>,$ modulus Q
key: L knapsack weights $w_1, \ldots, w_L$ of size N bits each
initial state $\mathbf{x}_0 = x_{10}, \ldots, x_{L0}$ of the LFSR

For $i = 1, 2, \ldots$ do

1. Step LFSR to produce next state
2. Compute knapsack sum

$$S_i = \sum_{k=1}^{L} x_{ki} w_i \mod Q$$

3. Extract some bits of S_i to form Z_i

Output: the sequence Z_i, $i = 1, 2, \ldots$

The knapsack sum sequence is periodic with period $2^L - 1$ which implies that the jth bit sequence $\tilde{s}_j$ also is periodic with period $2^L - 1$. The analysis in [98] shows that,

if Q = 2^N then the jth bit sequence $\tilde{s}_j$ of the knapsack sum sequence $\tilde{S}$ has linear complexity

$$\Lambda(\tilde{s}_j) \leq \sum_{k=1}^{2^j} \binom{L}{k} \qquad j < \lceil \log L \rceil$$

$$\Lambda(\tilde{s}_j) \leq \sum_{k=1}^{L} \binom{L}{k} = 2^L - 1 \qquad j \geq \lceil \log L \rceil$$

which shows that the log L least significant bits of a mod 2^L knapsack are cryptographically weaker than the higher-order bits. The effective keysize for output stage j is at least L log L bits for all j $\geq \lceil \log L \rceil$.

4 COMPLEXITY-THEORETIC APPROACH

The basis of the complexity-theoretic approach is the notion of computationally accessible information. If two random variables are statistically independent then there is no way to compute information about the second from observation of the first. But even if there is complete statistical dependence (as for ciphertext-plaintext pairs in some public key cryptosystems) it may not be computationally accessible.

In the complexity-theoretic model all computations are parametrized by a security parameter, usually the key length, and an asymptotic analysis is carried out. Only algorithms whose running times can be expressed as polynomials in the size of the input are considered to be computationally feasible. The opponent is assumed to be limited to polynomial time attacks. Cryptanalysis is the process of (1) predicting a single digit of the keystream, or (2) distinguishing the keystream sequence from a truly random sequence. A keystream generator is defined to be ***perfect*** if it is (1) unpredictable, or (2) indistinguishable by all polynomial-time statistical tests. But such a perfect generator is a hypothetical device; it is not known whether it exists. The proposed generators are all based on the assumed difficulty of some "famous" problem like, for instance, the discrete log [9], quadratic residuosity [10], and inverting the RivestShamir-Adleman (RSA) encryption algorithm, [1,80]. Proving the perfectness of these generators would require superpolynomial lower bounds on the complexity of the associated problems. Since this has been impossible, researchers have resorted to heuristic arguments, called intractability hypotheses, to prove that a generator is "perfect." We will reserve the word ***perfect*** (without quotation marks) for the ideal device, which is not known to exist, and will use "perfect" (with quotation marks) whenever a generator is "perfect" modulo some intractability hypothesis. The broader question also remains of how much evidence is provided by an asymptotic analysis, inasmuch as all implementations of cryptosystems necessarily have a finite size.

4.1 Basic Notions and Concepts

A ***bit generator*** G is a sequence {G, : $n \geq 1$} of polynomial time algorithms G,. Each $G_n : \{0,1\}^n \rightarrow \{0,1\}^l$ stretches a random key x^n of length n into a pseudorandom sequence z^l of length $l(n)$, where l is a polynomial function of n. Let $z^l = G_n(x^n)$ denote the keystream produced by G_n on input x^n.

Let $\mu_{R,l}$ denote the uniform probability distribution on the set of 1 bit sequences, that is, $\mu_{R,l}(s^l) = 2^{-l}$. Correspondingly, let $\mu_{G,l(n)}$ denote the probability distribution of keystream sequences z^l as generated by G_n for randomly selected keys (that is, for keys chosen according to $\mu_{R,n}$). The probability of a distinguished pseudorandom sequence z^l then is

$$\mu_{G,l}(z^l) = 2^{-n} \cdot \#\{x^n : G_n(x^n) = z^l\}$$

Let μ_G be the sequence of probability distributions $\{\mu_{G,l(n)} : n \geq 1\}$ as produced by G ; μ_G is said to be induced by G.

A practical security requirement for a keystream generator is its unpredictability. Given a segment z^i of i bits it must be unfeasible to extend the sequence beyond i. Otherwise, a captured segment of keystream would allow one to successfully decrypt a portion of the ciphertext without knowledge of the key. The concept of unpredictability can be formalized through the notion of a next-bit test (or predictor) [9,125].

A ***predictor (next-bir*** rest) C = {C, : $n \geq 1$} is a sequence of probabilistic polynomial size circuits C_n with $i_n < l(n)$ input gates and one binary output gate. Loosely speaking, a predictor C is capable of predicting a pseudorandom generator G if the fraction of times C_n's output bit b agrees with z_{i+1} is "significantly" different from. $\frac{1}{2}$. More precisely, a predictor C is said to predict a pseudorandom generator G (G is said to fail the next-bit test C) if there exists a polynomial $P(n)$ such that for infinitely many n

$$Pr(C_n(z^i) = z_{i+1}) \geq \frac{1}{2} + \frac{1}{P(n)}$$

where z^l is drawn according to $\mu_{G,l(n)}$.

Conversely, G is said to pass C if for all but a finite number of n and for all polynomials $P(n)$,

$$Pr(C_n(z^i) = z_{i+1}) < \frac{1}{2} + \frac{1}{P(n)}$$

G is said to be unpredictable if for all next-bit tests C, for all but a finite number of n, for all polynomials $P(n)$, and for all $i <$ l(n)

$$Pr(C_n(z^i) = z_{i+1}) < \frac{1}{2} + \frac{1}{P(n)}$$

A keystream generator attempts to efficiently simulate randomness. If the pseudorandom sequences it generates were efficiently distinguishable from purely random sequences one could not claim that the generator simulates randomness. On the other hand, the generator's output can always be distinguished from purely random sequences by simply trying all the keys and comparing the resulting pseudorandom sequences with the sequence at hand. But this method takes exponential time and is considered unfeasible. The ability of distinguishing a pseudorandom sequence from a random sequence seems to be the most fundamental step preceding any other step in analyzing the pseudorandom sequence. The concept of indistinguishability can be formalized through the notion of a statistical test [125].

A ***statistical test*** $T = \{T_n : n \geq 1\}$ for the bit generator is a sequence of probabilistic polynomial-size circuits T_n with I(n) input gates and one binary output gate.

Loosely speaking, a test T is able to distinguish a pseudorandom generator G if the fraction of times it puts out a 1 when it is confronted with sequences r^l drawn uniformly from $\mu_{R,l}$ is "significantly" different from the fraction of times it puts out a 1 when it is confronted with sequences z^l drawn according to the generator's output distribution $\mu_{G,l(n)}$. More formally, a test T is said to distinguish G if there exists a polynomial P(n) such that for infinitely many n

$$|p_n^{T,G} - p_n^{T,R}| \geq \frac{1}{P(n)}$$

where $p_n^{T,G}$ and $p_n^{T,R}$ denote the probabilities that T emits a 1 on input of sequences drawn according to $\mu_{G,l(n)}$ and $\mu_{R,l}$.

Conversely, a generator G is said to pass the statistical test T if for all polynomials $P(n)$ and all but a finite number of n

$$|p_n^{T,G} - p_n^{T,R}| < \frac{1}{P(n)}$$

Yao succeeded in linking together the notions of predictor and statistical test, proving their equivalence [125].

Theorem: *A bit generator G passes all next-bit tests C if and only if it passes all statistical tests* T.

One then defines: A bit generator G is ***perfect*** if it passes all polynomial size statistical tests T.

Based on Yao's result, it is (at least in principle) possible to establish that a generator is perfect by proving that there is no polynomial-size predictor for the resulting pseudorandom sequences. But unfortunately, none of the proposed generators could be proved to be perfect. Indeed, it is not known if perfect generators exist.

Motivated by Yao's result, Blum and Micali [9] developed a general scheme to construct bit generators. Letf $= \{f_n : X_n \rightarrow X,\}$ be a one-way permutation to be used as next-state function for the generator. Let $B = \{B, : X_n \rightarrow \{0, 1\}\}$ be a binary predicate with domain X_n to be used as output function. Randomly select an element $x \in X_n$ as seed, iterate f_n on x, and output $z_i = B_n(f_n^i(x))$ for $1 \leq i \leq l(n)$. If B is an unpredictable predicate for f then the keystream z^l is unpredictable (by next-bit tests) to the left, and by Yao's result, is indistinguishable by any statistical test T (in particular, it is also unpredictable to the right). Concrete applications of this scheme are:

1. Blum-Micali generator [9]

$$f_n : Z_p^* \ni x \mapsto y = \alpha^x \mod p \in Z_p^*$$

$$B_n : Z_p^* \ni y \mapsto \text{half}_p(x) \in \{0, 1\}.$$

2. Quadratic residue generator [10]

$$f_n : QR_N \ni x \mapsto y = x^2 \mod N \in QR_N$$

$$B_n : QR_N \ni y \mapsto \text{lsb}(x) \in \{0, 1\}.$$

3. RSA generator [1]

$$f_n : Z_N^* \ni x \mapsto y = x^e \mod N \in Z_N^*$$

$$B_n : Z_N^* \ni y \mapsto \mathrm{lsb}(x) \in \{0, 1\}.$$

These proposals are all based on the assumed difficulty of some "famous" problem such as the discrete log, quadratic residuosity, and inverting RSA. To prove the "perfectness" of each generator then makes it necessary to introduce heuristic intractability hypotheses with respect to the difficulty of the underlying problem. In the sequel these generators will be discussed in detail.

4.2 Generators

4.2.1 Shamir's pseudorandom number generator. The first proposal to use RSA in a pseudorandom number generator is due to Shamir [108]. His interest was in determining if for a given ciphertext Y the additional knowledge of an arbitrary number of decryptions mod N under various secret exponents $d_i = 1/e_i \mod \Phi(N)$ helps to find the decryption mod N under a specific secret exponent $d = 1/e \mod \Phi(N)$. Let $e_1, e_2, \ldots, e_l$ be a fixed sequence of public key exponents such that all the e_i are pairwise relatively prime. Choose an RSA-modulus $N = pq$ of size n such that all the e_i are relatively prime to $\Phi(N)$ and choose a public seed S from Z_N. The secret key of the generator consists only of the factorization $\{p, q\}$ of N.

Shamir's generator [108]:

Input: parameters: the modulus N,
the (public) seed S randomly selected from Z_N
the sequence $e_1, \ldots, e_l$,
key: the factorization p, q.

1. For $i = 1, \ldots, l$ decrypt S under the secret exponents $d_i = (1/e_i) \mod \Phi(N)$

$$Z_i = S^{1/e_i} \mod N$$

Output: the sequence of Z_i, $i = 1, \ldots, l$

Note that this generator puts out whole n-bit numbers, not only bits. In [108] the following result is proved.

Theorem: *There exists a fixed polynomial* P(n) *such that any circuit* C_n *which, when given as input a modulus N of size* n *and a seed* $S \in Z_N$, *is able to predict Z, from a number of known roots* $Z_2, \ldots, Z_l$ *for at least a fraction* S(n) *of the instances, can be transformed into another circuit* A, *of size at most* $|C_n| + P(n)$ *that inverts RSA on input N, S for at least a fraction* S(n) *of the instances.*

The proof exhibits the following circuit $A_{,,}$ and shows that it will recover $X = Y^{1/e} \mod N$ with the help of C,.

Circuit $A_{,,}$ *:* simulates $D_N(Y) \mod N$

Input: Y, N

1. Define $S = Y^{e_2 \cdots e_l} \bmod N$ and $E = \prod e_i$; compute for $i = 2, \ldots, l$

$$Z_i = S^{1/e_i} = Y^{\left(\frac{E}{e_1 e_i}\right)} \bmod N$$

by successive exponentiations.

2. Call C_n with input N, S, $Z_2, \ldots, Z_l$ and retrieve Z_1.
3. Use Euclid's extended gcd-algorithm to compute

$$\gcd\left(\frac{E}{e_1}, \ldots, \frac{E}{e_l}\right) = a_1\frac{E}{e_1} + \ldots + a_l\frac{E}{e_l} = 1$$

4. Since $Z_i = X^{E/e_i} \bmod N$ multiply

$$Z_1^{a_1} Z_2^{a_2} \ldots Z_l^{a_l} = (X^{E/e_1})^{a_1} (X^{E/e_2})^{a_2} \ldots (X^{E/e_l})^{a_l} = X$$

Output: $X = Y^{1/e_1} \bmod N$

If C_n is successful for a fraction $\delta(n)$ of the instances of size n so must be A_n, since the transformation $S = Y^{E/e_1} \bmod N$ is a permutation. Thus, predicting Shamir's pseudo-random number generator is provably equivalent to inverting RSA. If the RSA-function is almost everywhere secure, the random number generator must also be almost everywhere secure. But in reality, when RSA is used for encrypting plaintext messages, the distribution of ciphertexts will not be uniform, and consequently, the success rates may be quite different. In [9] it was pointed out that the numbers of Shamir's generator could be unpredictable as a whole yet they could have a special form. For instance, individual bits could be heavily biased or predictable with high probability; as an illustration let N have the form $2^n + k$, then for uniform Z_i the most significant bit will be 1 with probability k/N and 0 with probability $2^n/N$.

4.2.2 Blum-Micali generator. Let p be an odd prime and let a be a generator for Z_p^*, the ring of units modulo p. The discrete logarithm of an element $y \in Z_p^*$ with respect to a, denoted as index$_a$(y), is defined as the unique integer $0 \le x \le p - 2$ such that $y = a^x \bmod p$. The discrete logarithm problem with input p, a, y consists of finding index$_a$(y). In general, no efficient algorithm is known that can solve the discrete logarithm problem. If $y \in QR_p$, that is, y is a quadratic residue modulo p, then index$_a$(y) = $2t$ for some integer $t < (p-1)/2$. The two square roots of $y \in QR_p$ then are $a^t \bmod p$, which is called the principal square root, and $a^{t+(p-1)/2} \bmod p$, which is called the nonprincipal square root. For each p and a an element $x \in Z_p^*$ is a principal square root if and only if index$_a$(x) $< (p-1)/2$. Blum and Micali [9] showed that the existence of an efficient algorithm to decide if a given element is a principal square root with respect to p and a implies the existence of an efficient algorithm to compute the discrete logarithm. Define the binary predicate

$$\text{half}_p(x) = \begin{cases} 1 & \text{if } x < \dfrac{p-1}{2} \\ 0 & \text{otherwise} \end{cases}$$

Blum-Micali generator [9]:

Input: parameters: (p, a)
key: a randomly selected seed $x_1 \in Z_p^*$.

1. For $i = 1, \ldots, l$ do
   a. $z_i = \text{half}_p(x_i)$
   b. $x_{i+1} \leftarrow \alpha^{x_i} \bmod p$

Output: the sequence $z_1, z_2, \ldots, z_l$

The security of the index generator is based on the difficulty of computing discrete logarithms modulo p. In [9] it is shown:

Theorem: *If there is a probabilistic polynomial-size circuit that, when given as input* p, α, *and* $y = \alpha^x \bmod p$, *guesses* $\text{half}_p(x)$ *for at least a fraction* $1/P'(n)$ *of the primes* p *of size* n *with advantage* $\geq 1/2 + 1/P(n)$ *then, for any polynomial* Q(n), *there is another circuit that, when given as input* p, α, y *computes* $\text{index}_{p,\alpha}(y)$ *for at least a fraction* $1/P'(n)$ *of the primes* p *of size* n *with advantage* $\geq 1 - 1/Q(n)$.

If there is a predictor that, when given as input the keystream $z_1, \ldots, z_l$, is able to guess z_0 with probability $\geq 1/2 + \varepsilon(n)$, then it is possible to determine the predicate $\text{half}_p(x)$ from p, α, $y = \alpha^x \bmod p$ with the same probability of success. Use y as the seed for the index generator and produce the sequence $z_1, \ldots, z_l$; call the predictor to obtain z_0 which equals $\text{half}_p(x)$ with probability $\geq 1/2 + \varepsilon(n)$. By the above theorem, the existence of such a predictor with advantage $\geq 1/2 + 1/P(n)$ would then imply that the discrete logarithm problem could be solved in probabilistic polynomial time. To prove the "perfectness" of the index generator Blum and Micali introduced the following intractability hypothesis [9, 56].

Discrete logarithm assumption: For any polynomials P and P', any polynomial size circuit C, and all but a finite number of $n \geq 1$, the fraction of primes p of size n for which C is able to solve the discrete logarithm problem with probability $\geq 1 -$ l/P(n) is upperbounded by l/P'(n).

This hypothesis is true if and only if the index generator is unpredictable (to the left). As a last step, apply Yao's theorem [125] to conclude that the index generator is indistinguishable by probabilistic polynomial time statistical tests, and thus is "perfect."

The index generator is only one concrete implementation of a general scheme also proposed in [9]. Subsequently it was shown [53,62] that the discrete exponentiation function has $\log\log p$ simultaneous secure bits. This result implies a modified index generator with $\log\log p$ bits per each iteration of the discrete exponentiation.

4.2.3 RSA generators. The basic RSA generator [1] is another implementation of the general scheme proposed in [9], with $f(x) = x^e \bmod N$ as one-way permutation, and $B(x) = \text{lsb}(x)$ as unpredictable predicate for f. Let e be an integer ≤ 3. Let N be an integer of size |n| which is the product of two primes p, q, such that gcd(e, $\phi(N)$) = 1. The RSA generator accepts as input the triplet (N, e, x).

RSA generator:

Input: parameters: (N, e)
key: a randomly selected seed $x \in Z_N^*$.

1. $x_0 \leftarrow x$
2. For $i = 1, \ldots, l$ do
   a. compute $x_i \leftarrow x_{i-1}^e \mod N$
   b. extract $z_i \leftarrow \text{lsb}(x_i)$

Output: the sequence $\{z_i\}_1^l$

The security of this generator is based on the difficulty of inverting RSA. The following theorem is proved in [1].

Theorem: ***Every probabilistic algorithm which, when given as input*** $y = x^e \bmod N$, e, and, ***N, is able to guess the least significant bit of x in expected time*** T,(n) ***with probability at least*** $\frac{1}{2}$ + r(n) ***can be transformed into an algorithm that inverts RSA ciphertexts in expected time*** T(n) ***in*** $O(\varepsilon^{-8}(n)n^3$ T,(n)).

If $T_B(n)$ is a polynomial function of n so is $T(n)$ and RSA ciphertexts can be inverted in polynomial time. The ability of predicting z_0 from the keystream sequence $z_1, \ldots, z$, implies the ability of deciding $\text{lsb}_N(x)$ from $y = x^e \bmod N$. To see this, generate in a first step $z_1, \ldots, z_l$ using the RSA generator with y as x_1. In a second step call the predictor with $z_1, \ldots, z_l$ as input and obtain a bit b that equals z_0 with probability $1/2 + \varepsilon(n)$. Since z_0 equals $\text{lsb}_N(x_0)$ the probability that b equals $\text{lsb}_N(x)$ is also $1/2 + \varepsilon(n)$. The existence of a predictor with advantage $\varepsilon(n) = 1/P(n)$ would imply a polynomial-time algorithm for inverting RSA. Under the assumed intractability of inverting RSA there cannot be a probabilistic polynomial time algorithm that guesses the least significant bit of the plaintext x with advantage $1/P(n)$, which implies that there cannot be a polynomial time statistical test T that is able to distinguish the RSA generator. Thus, the RSA generator is "perfect."

Every probabilistic algorithm that, when given for input $x^e \bmod N$, N, and e, can guess the least significant bit of x with probability at least $1/2 + \varepsilon(n)$ is equivalent to a statistical test T that is able to distinguish the following distributions

1. The uniform distribution on [1, N]
2. The distribution of $x^e \bmod N$ for random, even $x \in [1, N]$

with advantage r(n). Instead of assuming that there is no polynomial-time algorithm that can invert a non-negligible fraction of RSA instances, one may directly start with the hypothesis that the above two distributions, for randomly selected RSA-moduli N of size n, are indistinguishable by polynomial-time statistical tests. A general result on the construction of pseudorandom functions presented in [36] then allows one to conclude that the iterated application of $x^e \bmod N$ in the RSA generator yields a keystream z^l which is also indistinguishable by polynomial-time statistical tests and thus, that the RSA generator is "perfect." It was shown in [1] that the arguments extend to the log n least significant bits of x. These bits are indistinguishable if RSA is assumed to be secure. Hence, without losing "perfectness" log n bits could be extracted per modular exponentiation in the RSA generator.

Generating 1 bit (or log n bits) of keystream per modular exponentiation is too slow for most applications. To overcome this drawback Micali and Schnorr [80] introduced the following much stronger hypothesis:

RSA hypothesis [80]: Let e $\geq$ 3 be an odd integer. For random moduli N of size n (which are the product of two primes each having size n/2) such that gcd(e, $\phi(N)$) = 1 and all M proportional to $N^{2/e}$ the following distributions on [I, N] are indistinguishable by polynomial-time statistical tests:

1. The uniform distribution on [1, N]
2. The distribution induced by x^e mod N for random $x \in$ [1, M]

As before the general result on the construction of pseudorandom functions presented in [36] now allows one to conclude that the iterated application of x^e mod N in the RSA generator yields a keystream z^l which is also indistinguishable by polynomial-time statistical tests and thus, that the modified RSA generator is "perfect." But now a fraction (e − 2)/e of the bits can be put out per exponentiation, and only a fraction 2/e is needed for the next iteration. This efficiency improvement is a simple consequence of the changed hypothesis.

4.2.4 Quadratic residue generator. Let N be the product of two distinct odd primes p and q. An element y $\in Z_N^*$ is called a quadratic residue modulo N if y = x^2 mod N for some $x \in Z_N^*$. Denote the set of quadratic residues modulo N by QR_N. Every element y $\in QR_N$ has exactly four square roots. If $p = q = 3$ mod N then exactly one of these four square roots belongs to QR_N. As a consequence, the mapping

$$QR_N \ni x \mapsto x^2 \mod N \in QR_N$$

is one-to-one and onto, and has associated the inverse mapping

$$QR_N \ni y \mapsto \sqrt{y} \mod N \in QR_N.$$

Rabin has shown [92] that factoring N and computing square roots are equivalent problems in the sense that an efficient algorithm for one of these problems implies an efficient algorithm for the other problem.

Quadratic residue generator [10]:

Input: parameters: modulus N of size n
key: a randomly selected $x_1 \in QR_N$.

For $i = 1, 2, \ldots, l$ do

1. $z_i = \text{lsb}(x_i)$
2. $x_{i+1} = x_i^2 \mod N$

Output: the sequence $z_1, z_2, \ldots, z_l$.

If $p = q = 3$ mod 4 then exactly half the elements of Z_N^* have Jacobi symbol + 1, and the other half have Jacobi symbol − 1. Denote the corresponding sets by $Z_N^*(+1)$ and $Z_N^*(-1)$. None of the elements of $Z_N^*(-1)$ and exactly half the elements of $Z_N^*(+1)$ are quadratic residues modulo N. The quadratic residuosity problem with input N and $x \in Z_N^*(+1)$ consists of deciding if x is a quadratic residue modulo N. It is an open

problem to find an efficient procedure that solves the quadratic residuosity problem. The security of the quadratic residue generator is based on the difficulty of deciding quadratic residuosity. In [10] it is shown that deciding quadratic residuosity can be efficiently reduced to determining the least significant bit of $x = \sqrt{y}$ mod N on input N and y $\in Z_N^*(+1)$. Combining this result with a result by Goldwasser and Micah [35] yields the following theorem [10]:

Theorem: *If there is a probabilistic polynomial-size circuit which, when given as input N and y* $\in QR_N$, *determines the least significant bit of* $x = \sqrt{y}$ *mod N for at least a fraction* l/P'(n) *of the moduli N of size* n *with advantage* $\geq 1/2$ + l/P(n) *then, for any polynomial* Q(n), *there is another circuit that, when given as input N and* $x \in Z_N^*$ (+ 1), *determines quadratic residuosity for at least a fraction* l/P'(n) *of the moduli N of size* n *with advantage* $\geq 1 - 1/Q(n)$.

If there is a predictor that, when given as input the keystream $z_1, \ldots, z_l$, is able to guess z_0 with probability $\geq \frac{1}{2} + \varepsilon(n)$, then it is possible to determine the lsb of $x = \sqrt{y}$ mod N with the same probability of success. Use y as the seed and generate the sequence $z_1, \ldots, z_l$, call the predictor to obtain z_0 which equals the lsb of x with probability $\geq \frac{1}{2} + \varepsilon(n)$. By the above theorem, the existence of such a predictor with advantage $\geq \frac{1}{2} + 1/P(n)$ would then imply that the quadratic residuosity problem could be solved in probabilistic polynomial time. To prove "perfectness" of the quadratic residue generator Blum, Blum, and Shub introduced the following intractability hypothesis [10,56].

Quadratic residuosity assumption: For any polynomials P and P', any circuit C, and all but a finite number of $n \geq 1$, the fraction of moduli N of size n for which C is able to determine quadratic residuosity with probability $\geq 1 - 1/P(n)$ is upper-bounded by l/P'(n).

This hypothesis is true if and only if the quadratic residue generator is unpredictable (to the left). As a last step, apply Yao's theorem [125] to conclude that the quadratic residue generator is indistinguishable by probabilistic polynomial time statistical tests, and thus is "perfect."

5 RANDOMIZED STREAM CIPHERS

In general, it is difficult to prove lower bounds on the computational effort to solve all, or almost all, instances of a given problem. As illustrated in the complexity-theoretic approach one typically has to resort to heuristic arguments about the computational difficulty of a problem (called intractability hypotheses). A different approach is to focus on the size of the cryptanalytic problem, instead of the work effort. The objective is to increase the number of bits the cryptanalyst has to examine in the cryptanalytic process while keeping the secret key small. This can be done by making use of a large publicly accessible random string in the encryption and decryption process. The key then specifies which parts of the large randomizer are to be used, whereas the opponent, not knowing the secret key, is forced to search through all the random data.

The security level of randomized stream cipher systems may be expressed by the average number of bits the cryptanalyst has to examine before his chances of determining

the key or the plaintext improve over pure guessing. The design objective is to establish a provable lower bound on the expected number of **bittests** the cryptanalyst must perform in order to have non-negligible probability of success. Different interpretations of such a result are possible. The expected number of **bittests** is a lower bound on the number of steps that any algorithm breaking the system must perform, and thus leads to a notion of computational security. But the expected number of **bittests** is also a lower bound on the number of bits the opponent has to observe before his a posteriori probabilities of the various keys or messages improve, and thus leads to a notion of information-theoretic security. These possible interpretations are the reason why we have treated randomized stream cipher systems in this section separately.

The following probabilistic cipher was proposed by Diffie [26] in unpublished work (see also [72]):

Diffie's randomized stream cipher:

Input: message $x = x_1, x_2, \ldots$
key k: a random n-bit string.

1. Flip 2" random sequences $r_1, r_2, \ldots, r_{2^n}$.
2. Use the kth random string r_k as the one-time pad to encrypt x.

Output: send $y = x \oplus r_k$ and $r_1, r_2, \ldots, r_{2^n}$ over 2" + 1 telephone lines.

In this randomized stream cipher the plaintext is encrypted with a one-time pad. The corresponding one-time pad is sent publicly over a telephone line, but disguised by a huge number of unrelated random strings also sent publicly. The cryptanalyst has no choice but to examine the random sequences one at a time until he finds the correct one-time pad. Thus, any attack must examine an expected number of bits which is in $O(2^n)$. It appears that a comparable security level is achievable if, instead of 2" only n random strings are sent, and the key is used to specify a linear combination of those random strings.

Massey and Ingemarsson [68] proposed a different randomized stream cipher which, for reasons that soon will become clear, they called the Rip van Winkle cipher.

Rip van Winkle cipher:

Input: plaintext sequence $\tilde{x} = x_1, x_2, \ldots$
key: a random n-bit number, $0 \leq k < 2^n$.

1. Flip a random k-bit preamble $r_1, r_2, \ldots, r_k$
2. Flip a random keystream $\tilde{z} = z_1, z_2, \ldots$.
3. For $i = 1, 2, \ldots$ do
   a. Encrypt the plaintext with the random keystream into the ciphertext sequence $\tilde{y}^{(1)}$:

$$y_i^{(1)} = x_i \oplus z_i$$

   b. Form a second sequence $\tilde{y}^{(2)}$ by concatenating the random preamble with the keystream:

$$y_i^{(2)} = \begin{cases} r_i & 1 \leq i \leq k \\ z_{k-i} & k < i \end{cases}$$

4. Send alternatingly a bit of $\tilde{y}^{(1)}$ and $\tilde{y}^{(2)}$.

Output: the sequence of bitpairs $(y_i^{(1)}, y_i^{(2)})$, $i = 1, 2, \ldots$

$y_i^{(1)}$ contains the encrypted message and $y_i^{(2)}$ contains the keystream only disguised by a random preamble $r_1, \ldots, r_k$ of unknown length k. In order to decrypt, the receiver simply has to wait k time units until the random preamble has passed by. For the Rip van Winkle cipher a lower bound on the expected number of bittests of any attack can be proved [68]:

Theorem: *Any algorithm that wants to determine the key* k *in a known plaintext attack (with probability at least* $\delta \geq 2^{-n}$ *of being correct) satisfies*

$$E(B) \geq 2^{n/2}\sqrt{1 - 2^{-n}\lfloor\delta^{-1}\rfloor}\,(1 + 2^{-n}(\lfloor\delta^{-1}\rfloor - 1))$$

where B *denotes the number of bittests.*

Thus, by randomizing the ciphertext it is possible to guarantee that any opponent has to spend an exponential effort in the size of the key: about $2^{n/2}$ bittests are necessary before the opponent can sharpen the probability distribution of possible key values. Unfortunately, there is a trade-off since the receiver also has to wait until an exponential number of bits, on the average until 2" bits of ciphertext have arrived before he can start decryption. In Massey's words, "One can easily guarantee that the enemy cryptanalyst will need thousands of years to break the cipher, if one is willing to wait millions of years to read the plaintext." The Rip van Winkle cipher is completely impractical.

Motivated by the ideas contained in the Rip van Winkle cipher, Maurer [75] developed a randomized stream cipher for which one can prove that the enemy obtains no information in Shannon's sense about the plaintext with probability close to 1 unless he accesses an infeasible number of bits (performs an infeasible computation). This approach provides an information-theoretic notion of security under a computational restriction of the opponent (whereas typically, information-theoretic security implies that the opponent has infinite computing power).

Maurer's randomized stream cipher [75]:

Input: plaintext $x^N = (x_1, \ldots, x_N) \in \mathbb{F}_2^N$
public randomizer $R[s, t]$, $1 \leq s \leq S$, $0 \leq t \leq T - 1$
key $k^S = (k_1, \ldots k_S) \in \mathbb{Z}_T^S$

1. Compute keystream

$$z_i = \sum_{s=1}^{S} R[s, (k_s + i - 1) \mod T] \mod 2 \quad 1 \leq i \leq N$$

2. Encrypt

$$y^N = x^N \oplus z^N$$

Output: ciphertext y^N.

The enemy attacking this system may have knowledge of the plaintext statistics and may also have some other a priori information about the plaintext. This situation is modeled by introducing an additional random variable V, jointly distributed with the plaintext according to $P(X^N, V)$, that collects all a priori information that the enemy may have about the plaintext. The enemy is allowed to use an arbitrary (possibly probabilistic) sequential strategy for selecting the addresses $A_i = (B_i, C_i)$ of the randomizer bits to be examined. Let $O_i = R[B_i, C_i]$ be the observed value of the randomizer bit at the address A_i in the ith step. The total information available to the enemy before execution of the ith step in his attack is the cryptogram Y^N, the values O^{i-1} of all previously examined randomizer bits together with the corresponding addresses A^{i-1}, and the a priori information V. Thus, the enemy's strategy is completely specified by the sequence of conditional probability distributions $P(A_i \mid Y^N V A^{i-1} O^{i-1})$, $i \geq 1$. For the described model Maurer was able to prove [75].

Theorem: *There exists an event $\mathscr{E}$ such that, for all joint probability distributions* $P(X^N, V)$ *and for all strategies of examining bits* O^M *at addresses* A^M *of the randomizer* R

$$I(X^N; Y^N A^M O^M \mid V, \mathscr{E}) = 0 \quad \text{and} \quad P(\mathscr{E}) \geq 1 - N\delta^S$$

where $\delta = M/ST$ *is the fraction of randomizer bits examined.*

The theorem states that if the event $\mathscr{E}$ occurs, then the enemy's total observation (Y^N, A^M, O^M) gives no additional information about the plaintext X^N beyond what he already knew before he started the attack. The chance that the enemy can learn something new about the plaintext is given by the probability that the event $\mathscr{E}$ does not occur. This probability is upperbounded by $P(\bar{\mathscr{E}}) < N\delta^S$. Hence, to have a substantial chance to obtain new information about the plaintext, the enemy is forced to examine a substantial fraction of the randomizer bits.

ACKNOWLEDGMENTS

First of all, I would like to thank Jim Massey for his continuous encouragement and help. His friendship deeply influenced my research, and my life.

Three recent workshops have greatly inspired me and have contributed to the present form of this manuscript. In October 1989, Jim Massey and Hansjiirg Mey organized the Monte Verità Seminar on "Future Directions in Cryptography" in Ascona, Switzerland. In September 1989, Andy Odlyzko, Claus Schnorr, and Adi Shamir organized a Workshop on Cryptography in Oberwolfach, Germany. And early in 1989, Thomas Beth and Fred Piper organized a Workshop on Stream Ciphers in Karlsruhe, Germany. I would like to thank the organizers for their invitations, and the participants for the many stimulating discussions.

I would like to thank Lennart Brynielsson, Bill Chambers, Yvo Desmedt, Dieter Gollmann, Christoph Gunther, Ueli Maurer, Harald Niederreiter, Kaisa Nyberg, Andy Odlyzko, Andrea Sgarro, Thomas Siegenthaler, Gus Simmons, and Othmar Staffelbach whose valuable comments and helpful discussions greatly improved the paper.

Finally, I would like to thank Crypto AG for their support in the initial phase of preparing this paper.

REFERENCES

[1] W. Alexi, B. Chor, 0. Goldreich, and C. I? Schnorr, "RSA and Rabin functions: Certain parts are as hard as the whole," ***SIAM J. Comput.,*** vol. 17, pp. 194-209, April 1988.

[2] H. Beker and F. Piper, ***Cipher Systems: the Protection of Communications,*** London: Northwood Books, 1982.

[3] B. Benjauthrit and I. S. Reed, "Galois switching functions and their applications," ***IEEE Trans. Comput.,*** vol. C-25, pp. 78-86, Jan. 1976.

[4] E. R. Berlekamp, ***Algebraic Coding Theory,*** New York: McGraw-Hill, 1968.

[5] J. Bernasconi and C. G. Gunther, "Analysis of a nonlinear feedforward logic for binary sequence generators," ***BBC Tech. Rep.,*** 1985.

[6] T. Beth and F. Piper, "The stop-and-go generator,'' in ***Lecture Notes in Computer Science 209; Advances in Cryptology: Proc. Eurocrypt*** '84, T. Beth, N. Cot, and I. Ingemarsson, Eds., Paris, France, April 9-11, 1984, pp. 88-92. Berlin: Springer-Verlag, 1985.

[7] R. E. Blahut, "Transform techniques for error-control codes," ***IBM J. Res. Develop.*** vol. 23, pp. 299-315, 1979.

[8] W. Blaser and I? Heinzmann, "New cryptographic device with high security using public key distribution," ***Proc. IEEE Student Paper Contest 1979-80,*** pp. 145-153, 1982.

[9] M. Blum and S. Micah, "How to generate cryptographically strong sequences of pseudo-random bits," ***SIAM J. Comput.,*** vol. 13, pp. 850-864, 1984.

[10] L. Blum, M. Blum, and M. Shub, '' 'A simple unpredictable pseudo-random number generator," ***SIAM J. Comput.,*** vol. 15, pp. 364-383, 1986.

[11] J. 0. Bruer, "On pseudo random sequences as crypto generators," in ***Proc. Int. Zurich Seminar on Digital Communication,*** Switzerland, 1984.

[12] L. Brynielsson, "On the linear complexity of combined shift register sequences ,'' in ***Lecture Notes in Computer Science 219; Advances*** in **Cryptology: Proc. Eurocrypt** '85, F. Pichler, Ed., Linz, Austria, April 1985, pp. 156-166. Berlin: Springer-Verlag, 1986.

[13] L. Brynielsson, "Wie man den richtigen Schlüssel in einem Heuhaufen findet," ***Kryptologie Aufbauseminar J.,*** Kepler Universität, Linz, Austria, 1987.

[14] L. Brynielsson, "Below the unicity distance," Workshop on Stream Ciphers, Karlsruhe, Germany 1989.

[15] C. M. Campbell, "Design and specification of cryptographic capabilities," ***IEEE Commun. Soc. Mug.,*** vol. 16, pp. 15-19, 1978.

[16] W. G. Chambers and S. M. Jennings, "Linear equivalence of certain BRM shift-register sequences," ***Electron. Lett.,*** vol. 20, Nov. 1984.

[17] W. G. Chambers and D. Gollmann, "Generators for sequences with near-maximal linear equivalence," ***IEE Proc. E.,*** vol. 135, pp. 67-69, 1988.

[18] A. H. Chan and R. A. Games, "On the linear span of binary sequences obtained from finite geometries," in ***Lecture Notes in Computer Science 263; Advances in Cryptology: Proc. Crypto*** '86, A. M. Odlyzko, Ed., Santa Barbara, CA, Aug. 11-15, 1986, pp. 405-417. Berlin: Springer-Verlag, 1987.

[19] A. H. Chan, M. Goresky, and A. Klapper, "Correlation functions of geometric sequences ,'' ***Proc. Eurocrypt*** 90, I. Damgård, Ed., Springer Verlag (in press).

[20] D. Chaum and J. H. Evertse, "Cryptanalysis of DES with a reduced number of rounds," in *Lecture Notes in Computer Science 218; Advances in Cryptology: Proc. Crypto* '85, H. C. Williams, Ed., Santa Barbara, CA, Aug. 18-22, 1985, pp. 192-211. Berlin: Springer-Verlag, 1986.

[21] Zong-duo Dai, "Proof of Rueppel's linear complexity conjecture," *IEEE Trans. Inform. Theory,* vol. 32, pp. 440-443, May 1986.

[22] D. E. Denning, *Cryptography and Data Security,* Reading, MA: Addison-Wesley, 1983.

[23] Y. Desmedt, J. J. Quisquater, and M. Davio, "Dependence of output on input of DES: Small avalanche characteristics," in *Lecture Notes in Computer Science 196; Advances in Cryptology: Proc. Crypto* '84, G. R. Blakley and D. Chaum, Eds., Santa Barbara, CA, Aug. 19-22, 1984, pp. 359-376. Berlin: Springer-Verlag, 1985.

[24] Y. G. Desmedt, "Cryptanalysis of conventional and public key cryptosystems," *Proc. SPRCI'89,* Rome, Nov. 23-24, 1989.

[25] W. Diffie and M. Hellman, "Privacy and authentication: An introduction to cryptography," *Proc. IEEE,* vol. 67, pp. 397-427, 1979.

[26] W. Diffie, Private communication, July 1984.

[27] J. F. Dillon, "Elementary Hadamard difference sets," *Proc. 6th Southeastern Conf. Combinatorics, Graph Theory, and Computing,* Boca Raton, FL, pp. 237-249, 1975; in Congressus Numerantium No. XIV, Utilitas Math., Winnipeg, Manitoba, 1975.

[28] J. H. Evertse, "Linear structures in block cyphers," in *Lecture Notes in Computer Science 304; Advances in Cryptology: Proc. Eurocrypt* '87, D. Chaum and W. L. Price, Eds., Amsterdam, The Netherlands, April 13-15, 1987, pp. 249-266. Berlin: Springer-Verlag, 1988.

[29] R. Forré, "The strict avalanche criterion: Spectral properties of boolean functions and an extended definition," in *Lecture Notes in Computer Science 403; Advances in Cryptology: Proc. Crypto* '88, S. Goldwasser, Ed., Santa Barbara, CA, Aug. 21-25, 1987, pp. 450-468. Berlin: Springer-Verlag, 1990.

[30] R. Forré, "A fast correlation attack on nonlinearly feedforward filtered shift-register sequences," in *Lecture Notes in Computer Science 434; Advances in Cryptology; Proc. Eurocrypt* '89, J.-J. Quisquater and J. Vandewalle, Eds., Houthalen, Belgium, April 10-23, 1989, pp. 586-595. Berlin: Springer-Verlag, 1990.

[31] J. Gait, "A new nonlinear pseudorandom number generator," *IEEE Trans. Software Eng.,* vols. S E3, no. 5, pp. 359-363, Sept. 1977.

[32] R. G. Gallager, "Low-density parity-check codes," Cambridge, MA: MIT Press 1963.

[33] M. R. Garey and D. S. Johnson, *Computers and Intractability, New* York: W. H. Freeman, 1979.

[34] P. R. Geffe, "How to protect data with ciphers that are really hard to break," *Electronics,* Jan. 4, 1973.

[35] S. Goldwasser and S. Micali, "Probabilistic encryption and how to play mental poker keeping secret all partial information," *J. Comput. Sys. Sci.,* vol. 28, no. 2, Apr. 1984.

[36] 0. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *J. ACM,* vol. 33, no. 4, pp. 792-807, 1986.

[37] J. Golić and M. V. Zivković, "On the linear complexity of nonuniformly decimated pn-sequences," ***IEEE Trans. Inform. Theory,*** vol 34, pp. 1077-1079, Sept. 1988.

[38] J. D. Golić, "On the linear complexity of functions of periodic GF(q)-sequences," ***IEEE Trans. Inform. Theory,*** vol. IT-35, pp. 69-75, Jan. 1989.

[39] D. Gollman, "Pseudo random properties of cascade connections of clock controlled shift registers," in ***Lecture Notes in Computer Science 209; Advances in Cryptology: Proc. Eurocrypt*** '84, T. Beth, N. Cot, and I. Ingemarsson, Eds., Paris, France, April 9-11, 1984, pp. 93-98. Berlin: Springer-Verlag, 1985.

[40] D. Gollman and W. G. Chambers, "Lock-in effect in cascades of clock-controlled shift-registers," in ***Lecture Notes in Computer Science 330; Advances*** in ***Cryptology: Proc. Eurocrypt*** '88, C. G. Gunther, Ed., Davos, Switzerland, May 25-27, 1988, pp. 331-343. Berlin: Springer-Verlag, 1988.

[41] D. Gollmann and W. G. Chambers, "Clock-controlled shift registers: A review," ***IEEE J. Selected Areas Commun., vol. 7,*** pp. 525-533, May 1989.

[42] S. W. Golomb, "Deep space range measurements," Jet Propulsion Laboratory, Pasadena, CA Research Summary, No. 36-1, 1960.

[43] S. W. Golomb, ***Shift Register Sequences,*** San Francisco: Holden Day, 1967.

[44] E. J. Groth, "Generation of binary sequences with controllable complexity," ***IEEE Trans. Inform. Theory,*** vol. IT- 17, no. 3, May 1971.

[45] C. G. Gunther, "On some properties of the sum of two pseudorandom sequences," paper presented at Eurocrypt'86, Linköping, Sweden, May 20-22, 1986.

[46] C. G. Gunther, "Alternating step generators controlled by de Bruijn sequences," in ***Lecture Notes in Computer Science 304; Advances in Cryptology: Proc. Eurocrypt'87,*** D. Chaum and W. L. Price, Eds., Amsterdam, The Netherlands, April 13-15, 1987, pp. 5-14. Berlin: Springer-Verlag, 1988.

[47] C. G. Gunther, "A universal algorithm for homophonic coding," in ***Lecture Notes in Computer Science 330; Advances in Cryptology: Proc. Eurocrypt'88,*** C. G. Gunther, Ed., Davos, Switzerland, May 25-27, 1988, pp. 405-414. Berlin: Springer-Verlag, 1988.

[48] T. Herlestam, "On the complexity of functions of linear shift register sequences," ***ht. Symp. Inform. Theory,*** Les Arc, France, 1982.

[49] T. Herlestam, "On functions of linear shift register sequences," in ***Lecture Notes in Computer Science 219; Advances in Cryptology: Proc. Eurocrypt'85,*** F. Pilcher, Ed., Linz, Austria, April 1985, pp. 119-129. Berlin: Springer-Verlag, 1986.

[50] C. J. Jansen, Investigations on nonlinear streamcipher systems: Construction and evaluation methods, Ph.D. thesis, Eindhoven University of Technology, The Netherlands, 1989.

[51] S. M. Jennings, "Multiplexed sequences: Some properties of the minimum polynomial," in ***Lecture Notes in Computer Science 149; Cryptography: Proc. Workshop Cryptography,*** T. Beth, Ed., Burg Feuerstein, Germany, March 29-April 2, 1982, pp. 189-206. Berlin: Springer-Verlag, 1983.

[52] S. M. Jennings, "Autocorrelation function of the multiplexed sequence," ***IEE Proc.***, vol. 131, no. 2, pp. 169-172, Apr. 1984.

[53] B. Kaliski, A pseudo random bit generator based on elliptic logarithms, M. Sc. thesis, Massachusetts Institute of Technology, 1987.

[54] E. L. Key, "An analysis of the structure and complexity of nonlinear binary sequence generators," ***IEEE Trans. Inform. Theory,*** vol. IT-22, no. 6, pp. 732-763, Nov. 1976.

[55] K. Kjeldsen and E. Andresen, "Some randomness properties of cascaded sequences," ***IEEE Trans. Inform. Theory,*** vol. IT-26, pp. 227-232, March 1980.

[56] E. Kranakis, ***Primality and Cryptography,*** Stuttgart: Teubner, Wiley, 1986.

[57] P. V. Kumar and R. A. Scholtz, "Bounds on the linear span of bent sequences," ***IEEE Trans. Inform. Theory,*** vol. IT-29, pp. 854-862, Nov. 1983.

[58] F! V. Kumar, R. A. Scholtz, and L. R. Welch, "Generalized bent functions and their properties," ***J. Combinatorial Theory,*** Ser. A 40, pp. 90-107, 1985.

[59] A. Lempel and M. Cohn, "Maximal families of bent sequences," ***IEEE Trans. Inform. Theory,*** vol. IT-28, pp. 865-868, Nov. 1982.

[60] R. Lidl and H. Niederreiter, "Finite Fields," in ***Encyclopedia of Mathematics and Its Applications, Vol. 20,*** Reading, MA: Addison-Wesley, 1983.

[61] S. Lloyd, "Counting functions satisfying a higher order strict avalanche criterion," in ***Lecture Notes in Computer Science 434; Advances in Cryptology; Proc. Eurocrypt'89,*** J.-J. Quisquater and J. Vandewalle, Eds., Houthalen, Belgium, April 10-23, 1989, pp. 63-74. Berlin: Springer-Verlag, 1990.

[62] D. L. Long and A. Wigderson, "How discrete is the discrete log?" in ***Proc. 15th ACM Symposium on Theory of Computation,*** Apr. 1983.

[63] M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions," ***SIAM J. Comput. vol.*** 17, pp. 373-386, 1988.

[64] F. J. MacWilliams and N. J. A. Sloane, "The theory of error correcting codes," Amsterdam: North-Holland, 1977.

[65] J. L. Massey, "Shift-register synthesis and BCH decoding," ***IEEE Trans. Inform. Theory,*** vol. IT-15, pp. 122-127, Jan. 1969.

[66] J. L. Massey, A. Gubser, A. Fischer, P. Hochstrasser, B. Huber, and R. Sutter, "A self-synchronizing digital scrambler for cryptographic protection of data," in ***Proceedings of International Zurich Seminar,*** March, 1984.

[67] J. L. Massey and R. A. Rueppel, "Linear ciphers and random sequence generators with multiple clocks,'' in ***Lecture Notes in Computer Science 209; Advances in Cryptology: Proc. Eurocrypt'84,*** T. Beth, N. Cot, and I. Ingemarsson, Eds., Paris, France, April 9-11, 1984, pp. 74-87. Berlin: Springer-Verlag, 1985.

[68] J. L. Massey and I. Ingemarsson, "The Rip van Winkle cipher-a simple and provably computationally secure cipher with a finite key," in ***Abstracts of Papers. IEEE Int. Symp. Inform. Theory,*** Brighton, England, June 24-28, 1985.

[69] J. L. Massey, "Delayed-decimation/square sequences," ***Proc. 2nd Joint Swedish-Soviet Workshop on Information Theory,*** Gränna, Sweden, Apr. 14-19, 1985.

[70] M. Z. Wong and J. L. Massey, "The characterization of all binary sequences with perfect linear complexity profiles," in ***Abstracts of Papers,*** *Eurocrypt'86*, Linköping, Sweden, May 20-22, 1986, pp. 3-4A–3-4B.

[71] J. L. Massey, "Cryptography and System Theory," ***Proc. 24th Allerton Conf. Commun., Control, Comput.,*** Oct. 1-3, 1986.

[72] J. L. Massey, "Probabilistic encipherment," ***Elektrotechnik und Maschinenbau,*** vol. 104, no. 12, Dec. 1986.

[73] U. Maurer and J. L. Massey, "Perfect local randomness in pseudo-random sequences," in ***Lecture Notes in Computer Science 435; Advances in Cryptology: Proc. Crypto'89,*** G. Brassard, Ed., Santa Barbara, CA, Aug. 20-24. 1989, pp. 110–112. Berlin: Springer-Verlag, 1990.

[74] J. L. Massey, "Applied digital information theory," Lecture Notes, Swiss Federal Institute of Technology, Zurich.

[75] U. Maurer, "A provable-secure strongly-randomized cipher," in ***Lecture Notes in Computer Science 473; Advances in Cryptology: Proc. Eurocrypt'90,*** I. Damgård, Ed., Aarhus, Denmark, May 21-24. 1990, pp. 361-373. Berlin: Springer-Verlag.

[76] R. L. McFarland, ''A family difference sets in non-cyclic groups," ***J. Combinatorial Theory,*** Ser. A 15, pp. l-10, 1973.

[77] W. Meier and 0. Staffelbach, "Fast correlation attacks on stream ciphers," ***J. Cryptol.,*** vol. 1, no. 3, pp. 159-176, 1989.

[78] W. Meier and 0. Staffelbach, "Nonlinearity criteria for cryptographic functions," in ***Lecture Notes in Computer Science 434; Advances in Cryptology; Proc. Eurocrypt'89,*** J.-J. Quisquater and J. Vandewalle, Eds., Houthalen, Belgium, April 10-23, 1989, pp. 549-562. Berlin: Springer-Verlag, 1990.

[79] W. Meier and 0. Staffelbach, "Correlation properties of combiners with memory in stream ciphers," in ***Lecture Notes in Computer Science 473; Advances in Cryptology: Proc. Eurocrypt'90,*** I. Damgård, Ed., Aarhus, Denmark, May 21-24. 1990, pp. 204-213. Berlin: Springer-Verlag.

[80] S. Micali and C. P. Schnorr, "Efficient, perfect random number generators," preprint, Massachusetts Institute of Technology, University of Frankfurt, 1988

[81] L. M. Milne-Thomson, "The calculus of finite differences," London: Macmillan and Co., 195 1.

[82] H. Niederreiter, "Continued fractions for formal power series, pseudorandom numbers, and linear complexity of sequences," contributions to General Algebra 5, Proc. Conf. Salzburg, Teubner, Stuttgart, 1986.

[83] H. Niederreiter, "Sequences with almost perfect linear complexity profile," in Lecture ***Notes in Computer Science 304; Advances in Cryptology: Proc. Eurocrypt'87,*** D. Chaum and W. L. Price, Eds., Amsterdam, The Netherlands, April 13-15, 1987, pp. 37-51. Berlin: Springer-Verlag, 1988.

[84] H. Niederreiter, "Probabilistic theory of linear complexity," in ***Lecture Notes in Computer Science 330; Advances in Cryptology: Proc. Eurocrypt'88, C. G.*** Gunther, Ed., Davos, Switzerland, May 25-27, 1988, pp. 191-209. Berlin: Springer-Verlag, 1988.

[85] H. Niederreiter, "Keystream sequences with a good linear complexity profile for every starting point," in ***Lecture Notes in Computer Science 434; Advances in Cryptology; Proc. Eurocrypt'89,*** J.-J. Quisquater and J. Vandewalle, Eds., Houthalen, Belgium, April 10-23, 1989, pp. 523-532. Berlin: Springer-Verlag, 1990.

[86] K. Nyberg, "Construction of bent functions and difference sets," in ***Lecture Notes in Computer Science 473;Advances in Cryptology:Proc. Eurocrypt'90***, I. Damgård, Ed., Aarhus, Denmark, May 21–24. 1990, pp. 151-160. Berlin: Springer-Verlag.

[87] I? Nyffeler, Binäre Automaten und ihre linearen Rekursionen, Ph.D. thesis, University of Berne, 1975.

[88] Y. Ohnishi, A study on data security, Master thesis (in Japanese), Tohuku University, Japan, 1988.

[89] F. Piper, "Stream ciphers," ***Elektrotechnik und Maschinenbau,*** vol. 104, no. 12, pp. 564-568, 1987.

[90] V. S. Pless, "Encryption schemes for computer confidentiality," ***IEEE Trans. Comput.,*** vol. C-26, pp. 1133-l 136, Nov. 1977.

[91] B. Preneel, W. Van Leekwijck, L. Van Linden, R. Govaerts, and J. Vandewalle, "Propagation characteristics of boolean functions," in ***Lecture Notes in Com-***

puter Science 473; Advances in Cryptology: Proc. Eurocrypt'90, I. Damgård, Ed., Aarhus, Denmark, May 21-24. 1990, pp. 161-173. Berlin: Springer-Verlag.

[92] M. 0. Rabin, "Digital signatures and public-key functions as intractable as factorization," Massachusetts Institute of Technology Laboratory for Computer Science, TR-212, 1979.

[93] 0. S. Rothaus, "On bent functions," *J. Combinatorial Theory,* **vol. 20,** pp. **300-305,** 1976.

[94] F. Rubin "Decrypting a stream cipher based on *J-K* flip-flops," *IEEE Trans Comput.,* vol. C-28, no. 7, pp. 483-487, July 1979.

[95] R. A. Rueppel, "Linear complexity and random sequences," in *Lecture Notes in Computer Science 219; Advances in Cryptology: Proc. Eurocrypt'85*, F. Pilcher, Ed., Linz, Austria, April 1985, pp. 167-188. Berlin: Springer-Verlag, 1986.

[96] R. A. Rueppel and J. L. Massey, "The knapsack as a nonlinear function," *IEEE Int. Symp. Inform. Theory,* Brighton, UK, May 1985.

[97] R. A. Rueppel, "Correlation immunity and the summation combiner," in *Lecture Notes in Computer Science 218; Advances in Cryptology: Proc. Crypto'85,* H. C. Williams Ed., Santa Barbara, CA, Aug. 18-22, 1985, pp. 260-272. Berlin: Springer-Verlag, 1986.

[98] R. A. Rueppel, *Analysis and Design of Stream Ciphers,* Berlin: Springer-Verlag, 1986.

[99] R. A. Rueppel and 0. Staffelbach, "Products of sequences with maximum linear complexity," *IEEE Trans. Inform. Theory,* vol. IT-33, no. 1, pp. 124–131, Jan. 1987.

[100] R. A. Rueppel, "When shift registers clock themselves," in *Lecture Notes in Computer Science 304; Advances in Cryptology: Proc. Eurocrypt'87,* D. Chaum and W. L. Price, Eds., Amsterdam, The Netherlands, April 13-15, 1987, pp. 53-64. Berlin: Springer-Verlag, 1988.

[101] R. A. Rueppel, "On the security of Schnorr's pseudo random sequence generator," in *Lecture Notes in Computer Science 434; Advances in Cryptology; Proc. Eurocrypt'89,* J.-J. Quisquater and J. Vandewalle, Eds., Houthalen, Belgium, April 10-23, 1989, pp. 423-428. Berlin: Springer-Verlag, 1990.

[102] R. A. Rueppel, "Security models and notions for stream ciphers," *Proc. 2nd IMA Conf. Cryptography and Coding,* Cirencester, England, Dec. 1989.

[103] J. E. Savage, Some simple self-synchronizing digital data scramblers, *Bell Sys. Tech. J.,* vol. 46, no. 2, pp. 449-487, Feb. 1967.

[104] T. Schaub, A linear complexity approach to cyclic codes, Ph.D. thesis, Swiss Federal Institute of Technology, Zurich, 1988.

[105] C. P. Schnorr, "On the construction of random number generators and random function generators," in *Lecture Notes in Computer Science 330; Advances in Cryptology: Proc. Eurocrypt'88*, C. G. Gunther, Ed., Davos, Switzerland, May 25-27, 1988, pp. 225-232. Berlin: Springer-Verlag, 1988.

[106] E. S. Selmer, Linear recurrence relations over finite fields, Lecture Notes, University of Bergen, Bergen, Norway, 1966.

[107] J. A. Serret, "Cours d'algèbre supérieure," Tome II, p. 154, Gauthier-Villars, Paris, 1886.

[108] A. Shamir, "On the generation of cryptographically strong pseudo-random sequences," *8th Int. Colloquium on Automata, Languages, and Programming,* Lecture Notes in Computer Science 62, Springer Verlag, 198 1.

[109] C. E. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech. J.*, vol. 28, pp. 656-715, Oct. 1949.

[110] T. Siegenthaler, "Correlation-immunity of nonlinear combining functions for cryptographic applications," ***IEEE Trans. Inform. Theory,*** vol. IT-30, pp. 776-780, Oct. 1984.

[111] T. Siegenthaler, "Cryptanalyst's representation of nonlinearity filtered ml-sequences," in ***Lecture Notes in Computer Science 219; Advances in Cryptology: Proc. Eurocrypt'85,*** F. Pilcher, Ed., Linz, Austria, April 1985, pp. 103-110. Berlin: Springer-Verlag, 1986.

[112] T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only," ***IEEE Trans. Comput.,*** vol. C-34, pp. 81-85, Jan. 1985.

[113] B. Smeets, "A note on sequences generated by clock-controlled shift registers," in ***Lecture Notes in Computer Science 219; Advances in Cryptology: Proc. Eurocrypt'85,*** F. Pilcher, Ed., Linz, Austria, April 1985, pp. 40-42. Berlin: Springer-Verlag, 1986.

[114] B. Smeets, "The linear complexity profile and experimental results on a randomness test of sequences over the field F_q," ***IEEE Int. Symp. Inform. Theory,*** Kobe, Japan, June 19-24, 1988.

[115] 0. Staffelbach and W. Meier, "Cryptographic significance of the carry for ciphers based on integer addition," in ***Advances in Cryptology, Proc. Crypto 90, S.*** Vanstone, Ed. (in press).

[116] R. C. Titsworth, "Optimal ranging codes," ***IEEE Trans. Space Electron. Telemetry,*** pp. 19-30, March 1964.

[117] S. A. Tretter, "Properties of PN^2 sequences," ***IEEE Trans. Inform. Theory,*** vol. IT-20, pp. 295-297, March 1974.

[118] G. S. Vernam, "Cipher printing telegraph systems for secret wire and radio telegraphic communications," ***J. Amer. Inst. Elec. Eng.,*** vol. 45, pp. 109-115, 1926.

[119] R. Vogel, "On the linear complexity of cascaded sequences," in ***Lecture Notes in Computer Science 209; Advances in Cryptology: Proc. Eurocrypt'84,*** T. Beth, N. Cot, and I. Ingemarsson, Eds., Paris, France, April 9-11, 1984, pp. 99-109. Berlin: Springer-Verlag, 1985.

[120] M. Z. Wang and J. L. Massey, "The characteristics of all binary sequences with perfect linear complexity profiles," paper presented at Eurocrypt'86, Linkoping, Sweden, May 20-22, 1986.

[121] M. Wang, "Linear complexity profiles and continued fractions," in ***Lecture Notes in Computer Science 434; Advances in Cryptology; Proc. Eurocrypt'89,*** J.-J. Quisquater and J. Vandewalle, Eds., Houthalen, Belgium, April 10-23, 1989, pp. 571-585. Berlin: Springer-Verlag, 1990.

[122] A. F. Webster and S. E. Tavares, "On the design of S-boxes," in ***Lecture Notes in Computer Science 218; Advances in Cryptology: Proc. Crypto'85,*** H. C. Williams, Ed., Santa Barbara, CA, Aug. 18-22, 1985, pp. 523-534. Berlin: Springer-Verlag, 1986.

[123] S. Wolfram, "Cryptography with cellular automata," in ***Lecture Notes in Computer Science 218; Advances in Cryptology: Proc. Crypto'85,*** H. C. Williams, Ed., Santa Barbara, CA, Aug. 18-22, 1985, pp. 429-432. Berlin: Springer-Verlag, 1986.

[124] G. Z. Xiao and J. L. Massey, "A spectral characterization of correlation-immune functions," ***IEEE Trans. Inform. Theory,*** vol. 34, no. 3, pp. 569–571, May 1988.

[125] A. C. Yao, "Theory and applications of trapdoor functions," ***Proc. 25th IEEE Symp. Foundations Comput. Sci., New*** York, 1982.

[126] R. Yarlagadda and J. E. Hershey, "Analysis and synthesis of bent sequences," *Proc. IEE,* vol. 136, pt. E., pp. 112-123, March 1989.

[127] L. E. Zegers, "Common bandwidth transmission of data signals and wide-band pseudonoise synchronization waveforms," *Philips Res. Reports Suppl.*, no. 4, 1972.

[128] K. Zeng and M. Huang, "On the linear syndrome method in cryptanalysis," in *Lecture Notes in Computer Science 403; Advances in Cryptology: Proc. Crypto'88*, S. Goldwasser, Ed., Santa Barbara, CA, Aug. 21-25, 1987, pp. 469-478. Berlin: Springer-Verlag, 1990.

[129] Y. Zheng, T. Matsumoto, and H. Imai, "Impossibility and optimality results on constructing pseudorandom permutations," in *Lecture Notes in Computer Science 434; Advances in Cryptology; Proc. Eurocrypt'89,* J.-J. Quisquater and J. Vandewalle, Eds., Houthalen, Belgium, April 10-23, 1989, pp. 412-422. Berlin: Springer-Verlag, 1990.

[130] N. Zierler, "Linear recurring sequences," *J. Soc. Indust. Appl. Math.,* vol. 7, no. 1, pp. 31-48, March 1959.

[131] N. Zierler and W. H. Mills, "Products of linear recurring sequences," *J. Algebra,* vol. 27, no. 1, pp. 147-157, Oct. 1973.