# Pseudorandom Bit Generators in Stream-Cipher Cryptography

**Kencheng Zeng, Chung-Huang Yang, Dah-Yea Wei, and T.R.N. Rao**
**University of Southwestern Louisiana**

The art of cryptography as a means for protecting private information against unauthorized access is as old as writing itself. Cryptography, indeed, is the only practical means for sending information over an insecure channel, be it telephone line, microwave, or satellite. The increasing use of electronic means of data communications, coupled with the growth of computer usage, has extended the need to protect information.

Stream ciphers play an especially important role in cryptographic practices — both diplomatic and military — that protect communications in the very high frequency domain. The central problem in stream-cipher cryptography, however, is the difficulty of generating a long unpredictable sequence of binary signals from a short and random key. Unpredictable sequences are desirable in cryptography because it is impossible, given a reasonable segment of its signals and computer resources, to find out more about them. Pseudorandom bit generators have been widely used to construct these sequences.

Considerable progress has been made in the design and analysis of pseudorandom bit generators over the last decade. The purpose of this article is to survey some of these developments.

**The information age lends new dimensions to the art of cryptography. Techniques for encryption, decryption, and fending off attacks from intruders provide the only protection of sensitive data.**

## Background

To provide the general background for our exposition, we begin by describing a cipher (see Figure 1). A cipher conceals the *plaintext M* by transforming it into a disguised form, called the *ciphertext C*, so that only the authorized receiver can transform it back to the original plaintext. The process of transforming plaintext into ciphertext is called *encryption* or *enciphering*, and the inverse transformation from ciphertext to plaintext is called *decryption* or *deciphering*.

To prevent the plaintext from being easily revealed by an unauthorized person, the sender must transform a given plaintext into a large variety of possible ciphertexts selected by a specific parameter. This parameter is called the *encryption key* $K_e$. The receiver then deciphers the ciphertext using the *decryption key* $K_d$. In a public-key cryptosystem, $K_e$ is made public while $K_d$ is kept secret; it is computationally infeasible to deduce $K_d$ from $K_e$. In a private-key cryptosystem, the sender and the receiver



0018-9162/91/0200-008$01.00 © 1991 IEEE

usually share a common key $K$ that is used for both enciphering and deciphering. $K$ is alterable and is always kept secret.

Private-key cryptosystems are further classified into block ciphers and stream ciphers. Block ciphers (see Figure 2) divide the plaintext into blocks and encipher each block independently. Under the control of a fixed key, different occurrences of a particular plaintext block will always be encrypted as the same ciphertext block. Thus, the block size has to be large enough to frustrate attacks from a cryptanalyst (an enemy that is attempting to obtain secret information) by analyzing the occurrence frequencies of various patterns among the cipher blocks.

In stream ciphers, on the other hand, the plaintext is encrypted on a bit-by-bit basis. In encrypting the dataflow to be transmitted, the key is fed into an algorithm called the pseudorandom bit generator to create a long sequence of binary signals. This "keystream" is then mixed with the plaintext sequence, usually by Exclusive-OR (XOR bit-wise modulo-2 addition) gates, to produce the ciphertext. Figure 3 illustrates this principle.

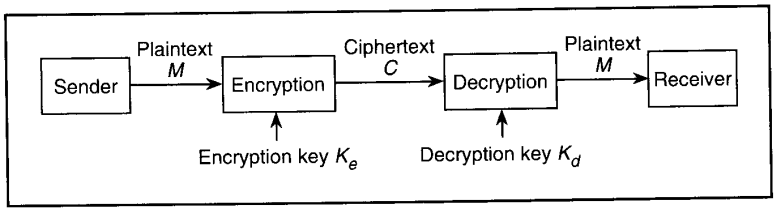


**Figure 1. A cipher.**

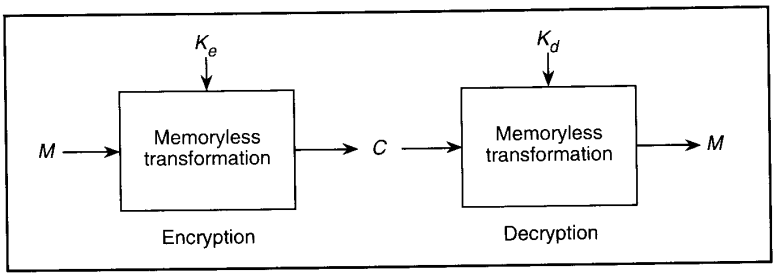


**Figure 2. A block cipher.**

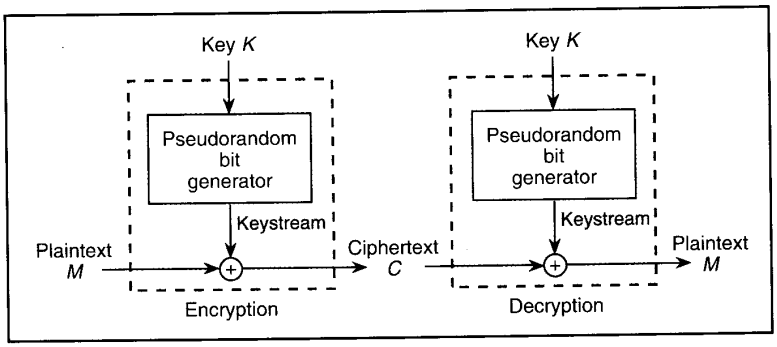


**Figure 3. A stream cipher.**

**The one-time pad.** Pseudorandom bit generators have a long history. In 1917, G. Vernam invented the remarkably simple *one-time pad* in which the secret key is a sequence of randomly generated bits (see Figure 4). The message of length $n$ is enciphered by XORing with the secret key of the same length to form the ciphertext, which is then deciphered by XORing with the same secret key transported to the legitimate receiver via a safe channel.

If the key $K$ is produced by a binary symmetric source such that the probability for any enciphering signal to be 1 is equal to 1/2 independently of the other signals, the one-time pad is perfectly secure against a *ciphertext-only attack* in which the cryptanalyst has no knowledge about the plaintext $M$. We point out in passing that in practice a certain amount of information about the plaintext $M$ is generally available and an attack made on the basis of this knowledge is called a *known-plaintext attack*.

However, the inconvenience of using one-time pads is evident. Since the twenties, many cipher systems have operated with an approximate version of the one-time pad that uses long sequences of pseudorandom bits generated from short secret keys. The bits appear to be random in the local sense, but they are in some way reproducible and hence are only pseudorandom in the large.

**Secure pseudorandom bit generators.** Generally speaking, we can formulate three requirements for *cryptographically secure* keystream generators:

(1) The period of the keystream must be large enough to accommodate the length of the transmitted message.
(2) The output bits must be easy to generate.

| | |
|---|---|
| Plaintext | $M$: 0110001111111101... |
| Secret key | $K$: 100110010001011... |
| Ciphertext | $C$: 111110101110110... |

**Figure 4. A one-time pad.**

(3) The output bits must be hard to predict. Given the generator and the first $n$ output bits, $a(0)$, $a(1)$, ... , $a(n-1)$, it should be computationally infeasible to predict the $(n+1)^{th}$ bit $a(n)$ in the sequence with better than a 50-50 chance. That is, given a portion of the output sequence, the cryptanalyst should not generate other bits forward or backward.

The problem is this: On which basis can one draw the conclusion that the output signals of a certain given keystream generator are hard to predict? No universally applicable and practically checkable criteria have been developed to certify the security of bit generators. For that matter, no general theory of cryptanalysis is known to exist except for an ever-expanding arsenal

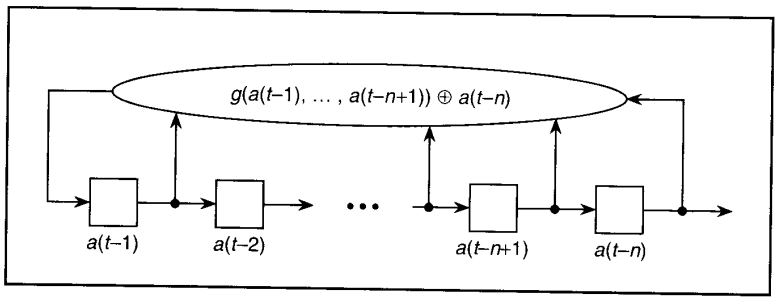


**Figure 5. The *n*-stage feedback shift register.**

of concrete attack methods elaborated for various practical purposes. Nevertheless, many systems have been proposed and then cracked in the history of cryptologic practices.

## Building blocks

Intuitively speaking, random means unpredictable. For a sequence to be random, the period of the sequence must be large and various patterns of a given length must be uniformly distributed over the sequence. Here we briefly describe some of the simpler methods for generating pseudorandom sequences. In particular, the linear feedback shift registers detailed in this section serve as building blocks for constructing the generators we discuss later.

**Linear congruence generators (LCGs).** This widely discussed method for generating pseudorandom numbers is based on recurrences of the form

$$X_{i+1} = aX_i + b \bmod m.$$

Here, $(a, b, m)$ are the parameters describing the generator and can be utilized as secret keys. $X_0$ is the seed. If the parameters are chosen judiciously, the numbers $X_i$ will not repeat until all integers of the interval $[0, m-1]$ have occurred. As an example, the sequence generated by $X_i = 5X_{i-1} + 3 \bmod 16$ with $X_0 = 1$ is

{ 1, 8, 11, 10, 5, 12, 15, 14, 9, 0, 3, 2, 13, 4, 7, 6, 1, 8, ... }.

Boyar[1] shows that sequences generated by LCGs are not cryptographically secure. Given a sufficiently long part of the sequence, one can reconstruct the parameters $m, a, b$. Truncated LCGs (using only some leading bits of $X_i$) also have been shown to be insecure. At the end of this presentation, we cite a survey on the cryptanalysis of some similar generators authored by Brickell and Odlyzko.

**Feedback shift registers (FSRs).** Strong cryptographic sequences can be designed on the basis of shift registers even when the feedback is linear. A feedback shift register consists of $n$ flip-flops and a feedback function that expresses each new element $a(t)$, when $t \geq n$, of the sequence in terms of the previously generated elements $a(t-n)$, $a(t-n+1), \ldots, a(t-1)$, as shown in Figure 5. For the state diagram of the shift register to consist of branchless cycles only, the feedback function must be nonsingular, that is, of the form

$$a(t) = g(a(t-1), a(t-2), \ldots, a(t-n+1)) \oplus a(t-n),$$

where $\oplus$ denotes the XOR operation and the element $a(t-n)$ figures explicitly on the right.

Depending on whether the function $g$ is linear (implementable with XORs alone) or not, the generator is called a linear feedback shift register (LFSR) or a nonlinear feedback shift register (NLFSR). Each individual storage element of the FSR is called a *stage*, and the binary signals $a(0)$, $a(1), a(2), \ldots, a(n-1)$ are loaded into them as initial data to generate the sequence.

The period of the sequence produced by an FSR depends both on the number of stages and on the details of the feedback connection. Clearly, the maximal period of a sequence that can be generated by an $n$-stage FSR with nonsingular feedback is $2^n$, the number of possible states an $n$-stage shift register may have.

**Nonlinear feedback shift registers.** The state diagram of a nonsingular FSR may have many small cycles, and the output sequence becomes insecure when the generator falls into one of them. A countermeasure is to design $n$-stage shift registers that generate sequences of the largest possible period $2^n$, the so-called *De Bruijn sequences* of degree $n$. The following is a De Bruijn sequence of period $2^4$, generated by the four-stage NLFSR of Figure 6.

1 0 1 1 0 0 1 0 1 0 0 0 0 1 1 1 ...

Rather than detailing NLFSRs, we summarize by saying that the number of possible $n$-stage De Bruijn sequences is as large as $2^{2^{n-1}-n}$ and these sequences have ideal pattern distribution. For example, every pattern of length 4 is produced exactly once by the generator in a period of work. But De Bruijn sequences constructible by known algorithms either are technically difficult to implement for fast generation or suffer from severe weaknesses related to their auto-correlational characteristics. For example, with an extensive class of sequences amenable to fast generation, the coincidence probability between $a(t)$ and $a(t-n)$ is much greater than 1/2. The opponent cryptanalyst can make good use of this feature.

**Linear feedback shift registers.** LFSR circuits have been in use for a long time for error-control coding, VLSI testing, spread-spectrum communications, etc. These circuits are also among the most important devices in building pseudorandom bit generators. They have feedback functions of the form

$$a(t) = c_1 a(t-1) \oplus c_2 a(t-2) \oplus \ldots \oplus c_{n-1} a(t-n+1) \oplus a(t-n),$$

with $c_i \in \{0, 1\}$. The feedback connection of an LFSR can be represented formally by the so-called *feedback polynomial*

$$f(x) = 1 + c_1 x + c_2 x^{n-2} + \ldots + c_{n-1} x^{n-1} + x^n$$

in the indeterminate $x$, which has no other meaning than as a mathematical symbol. The feedback polynomial decides the period and the statistical behavior of the output sequence. To avoid trivial output, the zero-state should be excluded from initial setting. For example, if a four-stage LFSR has the feedback polynomial

$$f(x) = 1 + x + x^2 + x^3 + x^4,$$

then, depending on the initial data we load into its stages, it generates one of the fol-

lowing three nontrivial sequences of period 5 with rather poor statistics:

*a*) 1 1 1 1 0 1 1 1 1 0 ... ,
*b*) 1 0 0 0 1 1 0 0 0 1 ... ,
*c*) 0 1 0 0 1 0 1 0 0 1 ... .

On the contrary, if the LFSR has the feedback polynomial

$$f(x) = 1 + x + x^4,$$

it generates a single nontrivial sequence of period 15 with the best statistics that a sequence of this length can possess:

1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 ... .

In general, to guarantee the largest possible period $2^n - 1$ for the output sequence, the feedback polynomial $f(x)$ of the LFSR should be *primitive*. This means $f(x)$ should be chosen so that the least positive integer $T$ that makes $X^T - 1$ divisible by $f(x)$ will be $T = 2^n - 1$. Ready algorithms exist for testing the primitiveness of polynomials. The number of primitive polynomials of degree $n$ is

$$\frac{\phi(2^n - 1)}{n},$$

where $\phi(x)$, known as the Euler function, denotes the number of positive integers less than the integer $x$ and relatively prime to it. A sequence generated by an LFSR with a primitive feedback polynomial is called a maximal-length LFSR sequence, or simply an $m$-sequence.

Maximality of the period simultaneously guarantees good statistics. It is easy to show that for any integer $0 < k \leq n$ and any $k$-pattern **a**, the probability for a segment **x** of length $k$, cut down randomly from the $m$-sequence, to be of pattern **a** is

$$prob(\mathbf{x} = \mathbf{a}) \doteq \begin{cases} 1/2^k + 1/2^{n+k}, & \text{if } \mathbf{a} \neq 0, \\ 1/2^k - 1/2^n, & \text{if } \mathbf{a} = 0. \end{cases}$$

It can also be deduced that for any integer $0 < q < 2^n - 1$, the coincidence probability between the signals $a(t)$, $a(t+q)$ is

$$prob(a(t) = a(t+q)) \doteq 1/2 - 1/2^n,$$

meaning that an $m$-sequence also has ideal auto-correlational characteristics.

**The linear complexity issue.** However, in spite of the large choice of primitive feedback polynomials in addition to large periods and ideal statistics, $m$-sequences cannot be used as keystreams without undergoing further cryptographic transformations. In fact, the key of secrecy (initial state and feedback connection) of an $n$-stage LFSR can be determined from just $2n$ successive bits of the output sequence. To see this, let us assume that the segment

$$a(t),\ a(t+1),\ \ldots,\ a(t+n-1),\ a(t+n),\ \ldots,\ a(t+2n-2),\ a(t+2n-1)$$

is given. Then we can express each of the $n$ bits $a(t+n), \ldots, a(t+2n-1)$ in terms of the $n$ bits immediately to the left of it as

$$\begin{aligned} a(t)c_0 \oplus a(t+1)c_1 \oplus \ldots \oplus a(t+n-1)c_{n-1} &= a(t+n), \\ a(t+1)c_0 \oplus a(t+2)c_1 \oplus \ldots \oplus a(t+n)c_{n-1} &= a(t+n+1), \\ &\ldots \\ a(t+n-1)c_0 \oplus a(t+n)c_1 \oplus \ldots \oplus a(t+2n-2)c_{n-1} &= a(t+2n-1). \end{aligned}$$

The feedback constants can then be determined by solving the displayed system of linear algebraic equations in the unknowns

$$c_1, \ldots, c_{n-1}.$$

In general, every periodic sequence **a** can be produced by a nonsingular LFSR. An efficient synthesis procedure[2] exists for finding the feedback polynomial of the shortest LFSR that will generate the sequence. The length of such an LFSR is called the *linear complexity* of the sequence **a**, and we denote it as $LC(\mathbf{a})$. Consequently, to design a generator suitable for cryptographic usage, one must guarantee a large-enough key-independent lower bound to the linear complexity of the sequences it generates.

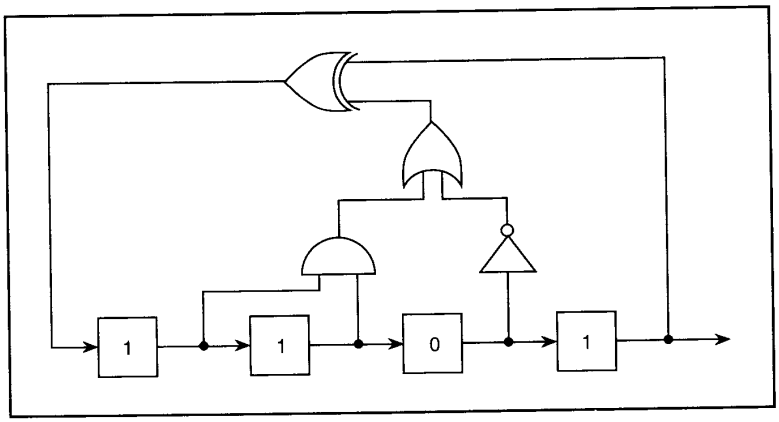


**Figure 6. A four-stage De Bruijn sequence generator.**

## Attacking keystream generators

Many of the publicly proposed keystream generators have been cracked. We remind the reader that cracking keystream generators amounts to the same thing as conducting known-plaintext attacks on stream ciphers. Secure communication should resist such attacks. When cracking something, we either crack it to pieces or crack it until flaws appear. With keystream generators, this means we either design a method to reveal the key of secrecy or show that certain parts of it turn out to be redundant when looked at from the angles of various attacks. We use the word *crack* in both senses because discovery and removal of *key redundancies* help to improve the cryptographic quality of newly designed generators.

We review three simple methods that will be needed to explain the next section of this article. The first two methods are of a rather general nature, reviewed here only to show how certain features of the generator (like statistical dependency and latent linearity) may influence the strength of the output sequence. The third method, however, is very concrete and often leads to thorough solutions to the problems.

**The Siegenthaler correlation attack.**[3] In many generators, the finally formed keystream **b** is obtained by combining the output sequences $\mathbf{a}_i$, $1 \leq i \leq k$ of several systems. Stronger or weaker statistical dependency may exist between **b** and each $\mathbf{a}_i$. To utilize these dependencies, Siegenthaler developed a general key-identification

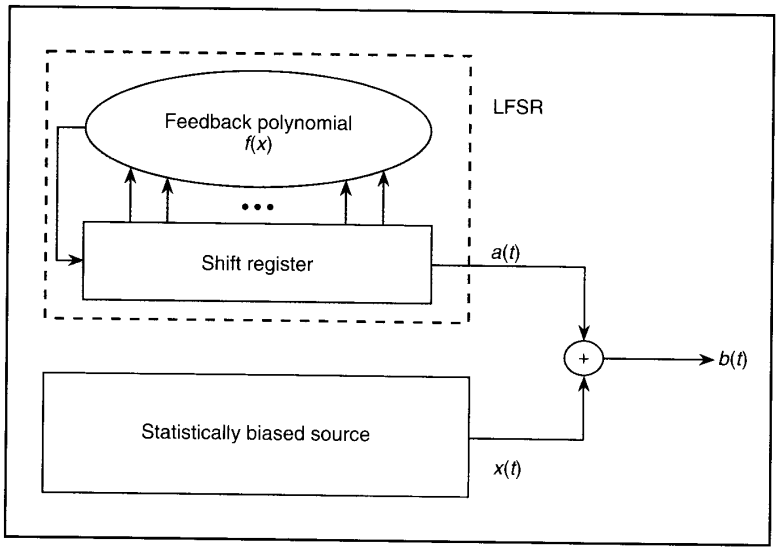


**Figure 7. Model used in the linear syndrome attack.**

criterion for conducting divide-and-conquer attacks on such systems. The attacks are addressed to the individual constituent sequences $\mathbf{a}_i$ on a separate basis and are carried out through exhaustive searching.

We review this first attack method briefly to emphasize the extreme importance — both to the cryptographer who designs and to the cryptanalyst who attacks — of considering the statistical interdependencies between various constituent parts of the keystream generator. The third method we discuss illustrates the point further.

**The linear consistency attack.** In contrast to the statistical test of Siegenthaler, an algebraic test oriented toward utilizing the linearity latent in various keystream generators is given by Zeng, Yang, and Rao.[4] The method is formulated on the basis of a precise estimation of the consistency probability of a system of linear algebraic equations $A\mathbf{x} = \mathbf{b}$ with a random $m \times n$ coefficient matrix $A$, $m > n$, and fixed nonzero vector **b.**

In many keystream generators, one can often single out a comparatively small subkey $K_1$ from the whole key of secrecy $K$ and set up a system $A\mathbf{x} = \mathbf{b}$ of linear algebraic equations such that

- the coefficient matrix $A$ depends solely upon the subkey $K_1$;
- the right-side vector **b** is determined by a certain segment of the output sequence;
- if the subkey $K_1$ is specified incorrectly, the corresponding system $A\mathbf{x} = \mathbf{b}$ will be, with a probability of nearly 1, inconsistent. When such a system is found to be consistent, one can generally deduce from its solution the remaining part $K - K_1$ of the key $K$.

When this is the case, one can apply exhaustive searching to determine the subkey $K_1$, with the consistency of the corresponding linear system as the key identification criterion. The success of such an attack means that the $|K| - |K_1|$ bits of the remaining part of the key $K$ in fact do not contribute substantially to the cryptographic strength of the system as a whole. A certain form of key redundancy has been discovered in the system.

**The linear syndrome attack.**[5,6] Since exhaustive searching has been applied in both of the discussed methods, we can regard them as different ways to discover key redundancy in various generators — rather than as practical algorithms for recovering the plaintext. On the other hand, if the statistical dependency between the sequences **b** and **a** is strong enough, effective analytic algorithms can be designed for this purpose (see Figure 7). Let us examine in some detail a third approach that we call the *linear syndrome* attack.

Suppose the source sequence $\mathbf{x} = \{x(t)\}$ in Figure 7 is statistically biased and contains more zeros than ones. The coincidence probability between the known signal $b(t)$ and the unknown signal $a(t)$ will be

$$p = prob(a(t) = b(t)) = 1/2 + \varepsilon,\ \varepsilon > 0.$$

We call the positive number $\varepsilon$ the coincidence advantage between $a(t)$ and $b(t)$. We try to recover **a** from **b** by incrementing this advantage. For this purpose, we assume that the feedback polynomial $f(x)$ is known to the cryptanalyst but the current state of the LFSR is unknown. We consider a set of trinomial multiples

$$g(x) = 1 + x^l + x^k,\quad k > l > 0$$

of $f(x)$. Corresponding to each of these trinomials, we can form three new signals called the syndromes:

$$\sigma_1(t) = b(t) \oplus b(t+l) \oplus b(t+k),$$
$$\sigma_2(t) = b(t-l) \oplus b(t) \oplus b(t+k-l),$$
$$\sigma_3(t) = b(t-k) \oplus b(t-k+l) \oplus b(t).$$

It can be shown that the coincidence advantage between each $\sigma_i(t)$ and $a(t)$ will be $2\varepsilon^2$. The new coincidence advantage turns out to be smaller than the advantage $\varepsilon$ we start with, but it is preferable to the old one in that we can try to amplify it by considering the joint effect of larger and larger sets of syndromes. Discovery and amplification of advantages is a principle of supreme importance in cryptanalysis.

Concretely speaking, if we consider a set of $2m + 1$ syndromes and revise the sequence **b** according to the rule of majority decision, namely, by putting

$$b'(t) = \bar{b}(t)$$

if at least $m + 1$ syndromes are 1, and $b'(t) = b(t)$ if otherwise, it can be shown that the coincidence advantage $\varepsilon_m$ between $a(t)$ and $b'(t)$ will tend to 1/2 as $m$ increases indefinitely. This means we can count on recovering the sequence **a**, and hence also the sequence **x**, from the sequence **b** by increasing the number of syndromes we use.

A more practical approach, however, is to make iterated revisions, using a suitably fixed number of syndromes computed according to fixed formulas. In particular, when $\varepsilon$ is large enough, say $\varepsilon = 0.25, 0.375$, or the like, it is easy to recover **a** from a segment of **b** of length linear in the size $n$ of the attacked LFSR. The computational expense needed for doing this is also linear in $n$. We see later that not one generator will collapse under this attack.

## Some design techniques

Numerous techniques that can be used to guarantee the unpredictability of the keystreams have been published in the open literature. We cannot discuss them in detail here. Instead, we are interested in seeing how some of these attempts fail or how their weakness is exposed under the attacks previously described. Only three of the simplest techniques will be reviewed, namely, that of nonlinear feed-forward transformation, step control, and multiclocking.

**Nonlinear feed-forward transformation.** Many keystream generators are based on combining two or more generators by using a nonlinear function. The problem is how to choose a function strong enough to serve the purpose. We illustrate this by analyzing a couple of simple examples.

*The Geffe generator.*[7] One of the simplest ways of combining LFSRs is to employ a two-to-one multiplexer, as shown in Figure 8. The output signal of the generator at the moment $t$ is

$$b(t) = a_1(t)a_3(t) \oplus \overline{a_1(t)}a_2(t)$$
$$= a_2(t) \oplus a_1(t)(a_2(t) \oplus a_3(t))$$

or, similarly,

$$b(t) = a_3(t) \oplus \overline{a_1(t)}(a_2(t) \oplus a_3(t)).$$

If the primitive feedback polynomials of LFSR-1, LFSR-2, and LFSR-3 have degrees $n_1$, $n_2$, and $n_3$, respectively, the generator will have linear complexity $LC = n_3n_1 + (n_1 + 1)n_2$ and period

$$T = lcm\left(2^{n_1}-1, 2^{n_2}-1, 2^{n_3}-1\right).$$

The weakness of such a generator comes from the fact that we have the coincidence probability

$$p = Prob(b(t) = a_2(t))$$
$$= Prob(a_1(t) = 0) + Prob(a_1(t) = 1) \cdot Prob(a_3(t) = a_2(t))$$
$$= 1/2 + 1/4.$$

The coincidence probability between $b(t)$ and $a_3(t)$ can be estimated similarly. Therefore, if the feedback polynomials of the LFSRs are all known, and are primitive trinomials of degrees not exceeding $n$, the system can be easily cracked under an attack carried out with the help of the linear syndrome method. The current states of all three constituent LFSRs can be recovered on a captured segment of length $N = 37\,n$ of the output sequence. The computation expense needed amounts to only 896 $n$-bit operations.

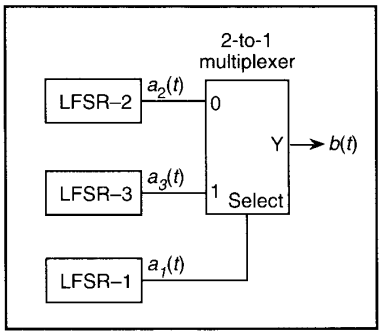


**Figure 8. The Geffe generator.**

*The Jennings generator.*[8] This scheme also uses the multiplexer to combine two LFSRs of lengths $l$ and $n$. Figure 9 shows how this scheme is structured. The literature reports that similar schemes have been recommended by the European Broadcasting Union as standards for scrambling television broadcasts.

The generator produces the output signals $c(t)$, $t \geq 0$, in the following way: Fix a positive integer $h \leq \min(l, \lfloor \log_2 n \rfloor)$ and a tap pattern

$$0 \leq i_0 < i_1 < \ldots < i_{h-1} \leq l-1$$

on LFSR-1. For every moment $t \geq 0$, form the number

$$u(t) = a(t+i_0) + a(t+i_1)\,2 + \ldots + a(t+i_{h-1})2^{h-1}$$

and transform it into

$$\theta(u(t)) = s_0(t) + s_1(t)2 + \ldots + s_{k-1}(t)2^{k-1},$$
$$k = \lceil \log_2 n \rceil,$$

by an injective mapping $\theta$: $\{0, 1, \ldots, 2^h-1\} \rightarrow \{0, 1, \ldots, n-1\}$. If we assume the primitive feedback polynomials of LFSR-1, LFSR-2 are known, then the mapping $\theta$ — together with the initial states $K_1$, $K_2$ of the two LFSRs — form the key of secrecy in the system. The output signal is defined by

$$c(t) = b\left[t + \theta(u(t))\right].$$

If $(l, n) = 1$, the output sequence has period $(2^l-1)(2^n-1)$ and linear complexity

$$LC \leq n\left(1 + \sum_{i=1}^{h} C_i^l\right),$$

with equality if the tap positions are spaced at equal intervals.

A closer look at the definition of $c(t)$ reveals that, irrespective of what $\theta$ may be, this signal can be expressed as a linear combination of the $b(t)$'s with coefficients dependent mainly on $K_1$. This is the Achilles' heel to which the linear consistency attack can be applied. It has been shown[4] that if the feedback polynomials are known, the Jennings generator can be cracked on an output segment of length $N \geq l + n2^h$ by $2^{l+h}$ consistency tests with $K_1$ alone as the objective of exhaustive searching. Though the family of possible mappings $\theta$ is dramatically large, the contribution of this key to the cryptographic strength of the system as a whole is negligible. The same can be said of the subkey $K_2$.

**Step control.** Besides subjecting the LFSR sequences to various nonlinear feed-

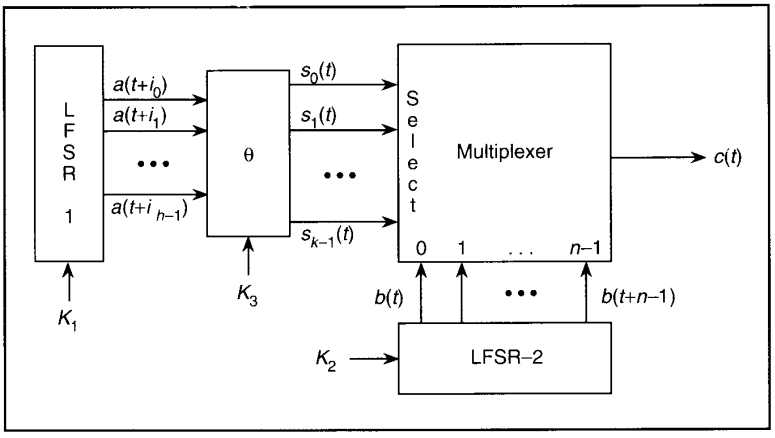


**Figure 9. The Jennings generator.**

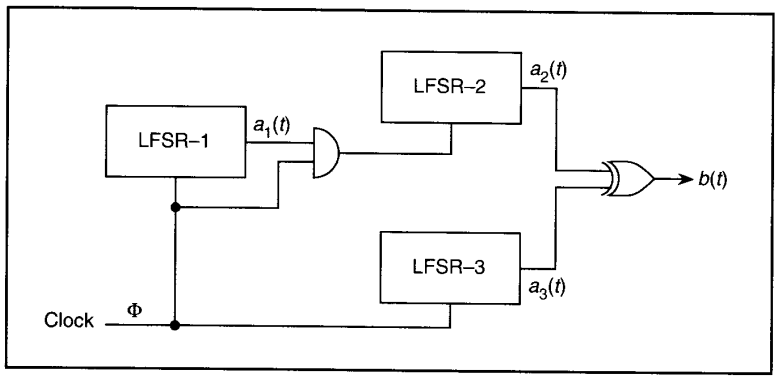


**Figure 10. The Beth-Piper stop-and-go generator.**

forward transformations, an important strengthening measure is to control the clock pulses to the LFSR(s) under consideration with the help of a step-controlling sequence. There are various ways for realizing step control of LFSRs.

*The Beth-Piper stop-and-go generator.*[9] As shown in Figure 10, the clock input to LFSR-2 is controlled by the output of LFSR-1 so that LFSR-2 can change its state at time $t$ only if $a_1(t-1) = 1$. Under suitable conditions imposed on the degrees $n_1, n_2, n_3$ of the three constituent LFSRs, the output sequence will have linear complexity

$$LC = (2^{n_1}-1)n_2 + n_3$$

and period

$$T = (2^{n_1}-1)(2^{n_2}-1)(2^{n_3}-1).$$

Although the output sequence has dramatically large linear complexity, the security of this generator is very poor. Let $a_2'(t)$ denote the output signal of the free-running LFSR-2 (that is, clocked in the normal way) at the moment $t$. Then we shall have the coincidence probability

$$\begin{aligned} p &= Prob\left[b(t)\oplus b(t+1) = a_3(t)\oplus a_3(t+1)\right] \\ &= Prob\,(a_1(t)=0) + Prob\,(a_1(t)=1) \\ &\quad \cdot Prob\,(a_2'(t) = a_2'(t+1)) \\ &= 1/2 + 1/4. \end{aligned}$$

Thus, if the feedback polynomials of LFSR-1 and LFSR-3 are known, we can apply the linear syndrome method to recover first the sequence $\mathbf{a}_3$ from $\mathbf{b}$, then the sequence $\mathbf{a}_1$ from $\mathbf{a}_2$. It is even unnecessary to assume the feedback polynomial of LFSR-2 is known.

*The Gollmann cascade.*[10] Figure 11 shows this strengthened version of the Beth-Piper generator. It consists of a series of $l$ LFSRs with primitive feedback polynomials of the same degree $n$. $\mathrm{LFSR}_i$ is clock-controlled jointly by all $\mathrm{LFSR}_j$, $j<i$, in the same way as in the Beth-Piper scheme. The output sequence has been shown to have period $T = (2^n-1)^l$ and a dramatically large linear complexity $LC \geq n(2^n-1)^{l-1}$. The proposer of this scheme is himself aware of a certain weakness related to it. Efforts are being made to estimate its influence on the security of the keystream thus generated.

*Bilateral stop-and-go control.*[11] A less clumsy scheme is based on the observation that if a binary sequence $\mathbf{b}$ has an odd prime period $T$, its linear complexity $LC(\mathbf{b})$ will be bounded from below by the order $Ord_T(2)$ of the number 2 modulo $T$, that is,

$$LC(\mathbf{b}) \geq Ord_T(2).$$

A sequence with a prime period is desirable because one can subject it to various further cryptographic transformations without influencing the established lower bound to its linear complexity, provided the transformations do not render it into a constant sequence. However, if $T = 2^n - 1$ happens to be one of the Mersenne primes, then we shall have $Ord_T(2) = n$. So $T$ should be chosen to be a prime beyond the progression $2^n - 1$. Sequences with periods of the form $q \cdot 2^n - 1$ with odd $q \geq 3$ have been constructed[11] from a pair of $n$-stage LFSRs.

Figure 12 shows the logic diagram for a sequence generator with period $5 \cdot 2^{n-2} - 1$, where both $n$-stage LFSRs are started in some nonzero states. Step control is carried out in the following manner:

(1) if $(a(t+n-1), a(t+n-2)) = (0, 1)$, then the clock pulse to LFSR-B is to be blocked;

(2) if $(b(t+n-1), b(t+n-2)) = (0, 1)$, but $(a(t+n-1), a(t+n-2)) \neq (0,1)$, then the clock pulse to LFSR-A is to be blocked.

The state diagram of such a generator consists of $3 \cdot 2^{n-2} - 1$ branched cycles of length $5 \cdot 2^{n-2} - 1$, each carrying a number of branches of length one. The values of $n$ that make $5 \cdot 2^{n-2} - 1$ prime can be determined by the help of the Lucas-Lehmer criterion or by the algorithm proposed in the cited paper.[11] The linear complexity of the out-

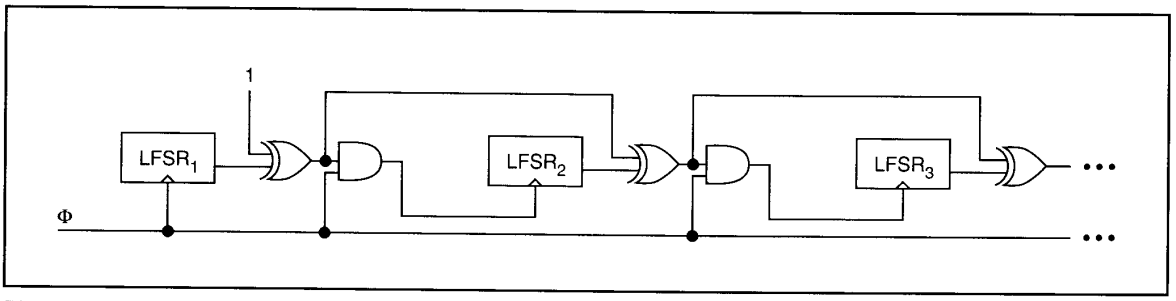


**Figure 11. The Gollmann cascade stop-and-go generator.**

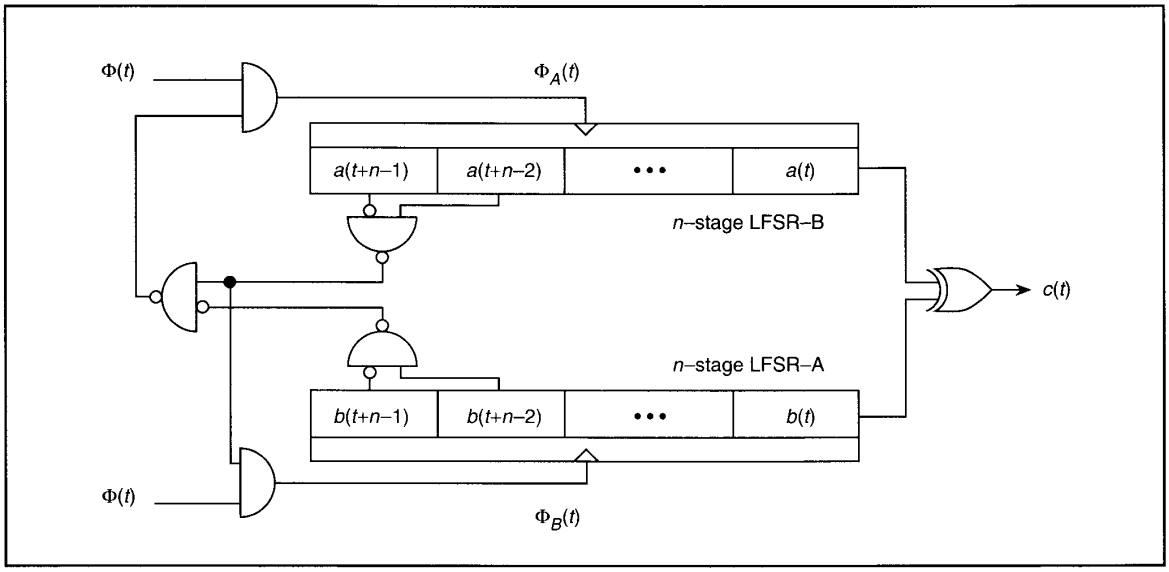


**Figure 12. A pseudorandom bit generator based on bilateral step control.**

put sequence has been shown to be of an order of magnitude approximately equal to that of the period. No evident key redundancy has been observed in this system.

**Multiclock systems.** The techniques discussed up to this point assume that the LFSR under consideration operate at the same bit rate as the final output. Some researchers[12-14] propose that the LFSRs be operated at rates higher than the output rate. Although several cycles of system clock will be needed for the generation of a single output bit, multiclocking, when used judiciously, opens new possibilities for designing interesting generators.

*The Massey-Rueppel multispeed generator.*[12] This generator utilizes two LFSRs clocked at different speeds. As illustrated in Figure 13, LFSR-2 of degree $n$ is clocked at a speed $d \geq 2$ times as fast as LFSR-1 of degree $l$, $n \geq l$, and the output signal $c(t)$ is produced according to

$$c(t) = \sum_{i=0}^{l-1} a(t+i)\, b(dt+i).$$

The speed factor $d$ is variable and is used as a part of the key of secrecy.

If LFSR-1 and LFSR-2 have primitive feedback polynomials and $(l, n) = 1$, $(d, 2^n-1) = 1$, then the output sequence would have a linear complexity $LC = ln$ with period $T = (2^l-1)(2^n-1)$ and would possess excellent statistical properties.

Again, the bilinear character of the definition of the signal $c(t)$ for fixed $d$ leaves space for the linear consistency attack to be applied here. It has been shown[4] that if the feedback polynomials are known, the cryptanalyst can determine the speed factor $d$ and the initial states of both LFSRs by $(d_{max}-1)(2^l-1)$ consistency tests applied to an output segment of length $N \geq l + n + \log_2 d_{max}$.

*Self-decimation of* m-*sequences.*[13,14] In 1987, Rueppel proposed a sequence generator using a single LFSR.[13] Figure 14 illustrates the arrangement. An LFSR with a primitive feedback polynomial is used to clock-control itself in the following way. When the output signal is 0, $d$ clock pulses

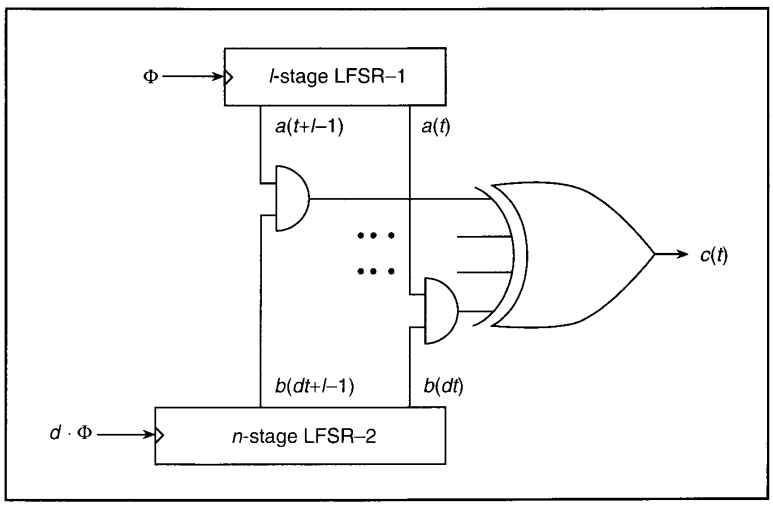


**Figure 13. Massey-Rueppel's multispeed generator.**

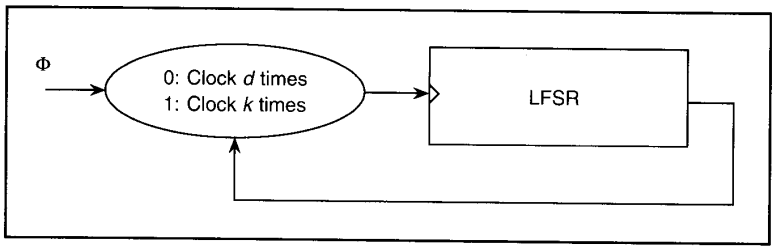


**Figure 14. Rueppel's self-decimated sequence.**

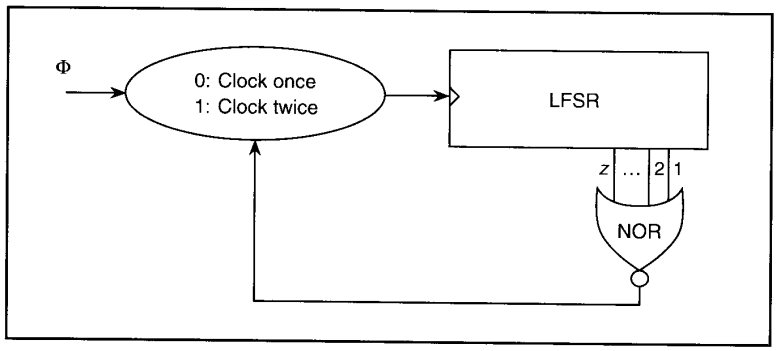


**Figure 15. Gollmann's self-decimated sequence.**

are applied to the LFSR; otherwise, $k$ clock pulses are applied. The effect is that some of the signals of the $m$-sequence generated by the LFSR are skipped or decimated in forming the output sequence, according to a pseudorandom rule determined by the $m$-sequence itself. As it stands, Rueppel's scheme is not a very serious generator for cryptographic purposes.

The scheme illustrated in Figure 15 was proposed by Chambers and Gollmann.[14] By suitably choosing the number $n$ of stages in the LFSR and the number $z$ of input ends to the transferring NOR function, the authors succeeded in producing a binary sequence of prime period $p$ and linear complexity equal to $p-1$ or $p$. But this is only algebra! With the increase of the number $z$, the resulting decimated sequence differs from the undisturbed $m$-sequence less and less.

Although the schemes proposed seem to be unsuccessful, the idea of self-decimation is intrinsically interesting. Much remains to be done before a qualified generator can be proposed on the basis of this idea.

We conclude with a couple of remarks concerning the design of qualified keystream generators. First, there are unlimited mathematical and technical resources for constructing keystream generators, but there is no generally checkable criterion for justifying the usability of the systems we design. Therefore, the only practical way to certify a generator for use in communication protection is to subject it to merciless mocking attacks. The examples of design failure considered here should be accepted as evidences against any kind of blind optimism in this aspect. As a much more convincing argument, we need only remark that an oil spill can only spoil the environment in certain localities, while a single case of cryptographic failure can turn everything into turmoil! The history of cryptologic practice contains examples of the latter kind.

Second, we would like to reemphasize that large linear complexity is only one of the necessary conditions that a qualified keystream generator must satisfy. There are many good methods for guaranteeing large lower bounds to it. Some of the recently published research in stream-cipher cryptography misguidedly concentrates on such issues as large periods and large linear complexity alone. We think that discovery and liquidation of strong statistical dependencies and systematic latent algebraic relations — including the linear ones — are also important problems to consider because they often lead to key redundancy in the systems we design. ■

## Acknowledgments

This research was supported by the Board of Regents of Louisiana Grant 86-USL(2)-127-03.

## References

1. J. Boyar, "Inferring Sequences Produced by Pseudorandom Number Generators," *J. ACM*, Vol. 36, No. 1, Jan. 1989, pp. 129-141.

2. J.L. Massey, "Shift-Register Synthesis and BCH Decoding," *IEEE Trans. Information Theory*, Vol. IT-15, No. 1, Jan. 1969, pp. 122-127.

3. T. Siegenthaler, "Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications," *IEEE Trans. Information Theory*, Vol. IT-31, No. 5, Sept. 1984, pp. 776-780.

4. K.C. Zeng, C.H. Yang, and T.R.N. Rao, "On the Linear Consistency Test (LCT) in Cryptanalysis with Applications," *Proc. Crypto '89*, Springer-Verlag Lecture Notes in Computer Science, No. 435, Springer-Verlag, New York, 1989, pp. 164-174.

5. K.C. Zeng and M.Q. Huang, "On the Linear Syndrome Method in Cryptanalysis," *Proc. Crypto '88*, Springer-Verlag Lecture Notes in Computer Science, No. 403, New York, 1988, pp. 469-478.

6. K.C. Zeng, C.H. Yang, and T.R.N. Rao, "An Improved Linear Syndrome Algorithm in Cryptanalysis with Applications," to appear in *Proc. Crypto '90*, Springer-Verlag Lecture Notes in Computer Science, New York.

7. P.R. Geffe, "How to Protect Data with Ciphers that Are Really Hard to Break," *Electronics*, Vol. 46, No. 1, Jan. 1973, pp. 99-101.

8. S.M. Jennings, "Multiplexed Sequences: Some Properties of the Minimum Polynomial," *Proc. Workshop on Cryptography*, No. 149, Springer-Verlag Lecture Notes in Computer Science, New York, 1982, pp. 189-206.

9. T. Beth and F.C. Piper, "The Stop-and-Go Generator," *Proc. Eurocrypt '84*, Springer-Verlag Lecture Notes in Computer Science, No. 209, New York, 1984, pp. 88-92.

10. W.G. Chambers and D. Gollmann, "Lock-in Effect in Cascades of Clock-Controlled Shift-Registers," *Proc. Eurocrypt '88*, Springer-Verlag Lecture Notes in Computer Science, No. 330, 1988, pp. 331-343.

11. K.C. Zeng, C.H. Yang, and T.R.N. Rao, "Large Primes in Stream-Cipher Cryptography," *Proc. Auscrypt '90*, Springer-Verlag Lecture Notes in Computer Science, No. 453, New York, 1990, pp. 194-205.

12. J.L. Massey and R.A. Rueppel, "Linear Ciphers and Random-sequence Generators with Multiple Clocks," *Proc. Eurocrypt '84*, Springer-Verlag Lecture Notes in Computer Science, No. 209, New York, 1984, pp. 74-87.

13. R.A. Rueppel, "When Shift Registers Clock Themselves," *Proc. Eurocrypt '87*, Springer-Verlag Lecture Notes in Computer Science, No. 304, New York, 1987, pp. 53-64.

14. W.G. Chambers and D. Gollmann, "Generators for Sequences with Near-Maximal Linear Equivalence," *IEE Proceedings*, Vol. 135, Pt. E., No. 1, Jan. 1988, pp. 67-69.

## Further reading

Arazi, B., "On the Synthesis of De Bruijn Sequences," *Information and Control*, Vol. 49; No. 2, May 1981, pp. 81-90.

Beker, H., and F. Piper, "Cipher Systems," John Wiley & Sons, New York, 1982.

Brickell, E.F., and A.M. Odlyzko, "Cryptanalysis: A Survey of Recent Results," *Proc. IEEE*, Vol. 76, No. 5, May 1988, pp. 578-593.

Chambers, W.G., "Clock-Controlled Shift Registers in Binary Sequence Generators," *IEE Proceedings*, Vol. 135, Pt. E., No. 1, Jan. 1988, pp. 17-24.

Desmedt, Y.G., "What Happened with Knapsack Cryptographic Schemes?" *NATO ASI on Performance Limits in Communication Theory and Practice*, Il Ciocco, Italy, 1986.

Gollmann, D., and W.G. Chambers, "Clock-Controlled Shift Registers: A Review," *IEEE J. on Selected Areas in Comm.*, Vol. 7, No. 4, May 1989, pp. 525-533.

Golomb, S.W., *Shift Register Sequences*, rev. ed., Aegean Park Press, Laguna Hills, Calif., 1982.

Kahn, D., *The Codebreakers*, MacMillan, New York, 1967.

Knuth, D.E., *The Art of Computer Programming; Vol. 2, Seminumerical Algorithms*, 2nd ed., Addison-Wesley, Reading, Mass., 1981.

Lempel, A., "On a Homomorphism of the De Bruijn Graph and its Applications to the Design of Feedback Shift Registers," *IEEE Trans. Computers*, Vol. C-19, No. 12, Dec. 1970, pp. 1,204-1,209.

Meier, W., and O. Staffelback, "Fast Correlation Attacks on Certain Stream Ciphers," *J. Cryptology*, Vol. 1, 1989, pp. 159-176.

Piper, F., "Recent Developments in Cryptography," *NATO ASI on Performance Limits in Communication Theory and Practice*, Il Ciocco, Italy, 1986.

Rueppel, R.A., "Analysis and Design of Stream Ciphers," Springer-Verlag, New York, 1986.

Shannon, C.E., "Communication Theory of Secrecy Systems," *Bell Systems Tech. J.*, Vol. 28, No. 4, Oct. 1949, pp. 656-715.

**Kencheng Zeng** has been a professor at the University of Science and Technology of China, Beijing, since 1978 and a visiting professor at the Center for Advanced Computer Studies, University of Southwestern Louisiana, since 1989. His areas of scientific interest include algebra, number theory, and the cryptology-related problems of mathematics.

Zeng was graduated from the National Qinghua University Department of Mathematics in 1950 .

**Chung-Huang Yang**'s current research interests include cryptology, VLSI design, and computer graphics.

Yang holds a BS degree in electrical engineering from the National Cheng-Kung University and the MS and PhD degrees in computer engineering from the University of Southwestern Louisiana. He is a member of the IEEE Computer Society, the ACM, and the International Association for Cryptologic Research.

**Dah-Yea Wei** is a doctoral candidate in computer science at the Center for Advanced Computer Studies, University of Southwestern Louisiana. His research interests include cryptography and data security, coding theory, and parallel processing.

Wei received the BE degree in information engineering from Feng Chia University in 1983 and the MS degree in computer and information sciences from the University of Alabama in 1987.

**T.R.N. Rao** (also known as N. Rao Thammavaram) holds the chair of Z.L. Loflin Professor of Computer Science at the Center for Advanced Computer Studies at the University of Southwestern Louisiana. He previously taught at the University of Maryland and Southern Methodist University and served as a Fulbright fellow in India during 1986-87.

Rao holds the BSc in physics from Andhra University and the DIISc from the Indian Institute of Science, Bangalore. He also received the MS and PhD degrees in electrical engineering from the University of Michigan in 1961 and 1964, respectively. Rao received a fellow of IEEE award in 1984 and is an IEEE Computer Society distinguished visitor and an ACM lecturer. He is the author of over 75 papers and several books, and has received numerous grants from federal agencies for his work in cryptography.

The authors can be contacted at the Center for Advanced Computer Studies, University of Southwestern Louisiana, PO Box 44330, Lafayette, LA 70504-4330.