Rainer A. Rueppel

Analysis and Design of Stream Ciphers

Springer-Verlag

Communications and Control Engineering Series

Rainer A. Rueppel

Analysis and Design of Stream Ciphers

With 53 Figures

Springer-Verlag
Berlin Heidelberg NewYork
London Paris Tokyo

Dr. RAINER A. RUEPPEL
Dept. of Electrical Engineering
and Computer Science
University of California, San Diego
EECS C-014
La Jolla, CA 92093
USA

ISBN-13:978-3-642-82867-6 e-ISBN-13:978-3-642-82865-2
DOI: 10.1007/978-3-642-82865-2

Library of Congress Cataloging in Publication Data.
Rueppel, Rainer,
Analysis and design of stream ciphers.
(Communications and control engineering series)
1. Ciphers. 2. Cryptography.
I. Title. II. Series.
Z104.R83 1986 652'.8 86-17663
ISBN-13:978-3-642-82867-6 (U.S.)

2161/3020-543210

Acknowledgments

This book would not exist without Professor Jim Massey, my former thesis advisor at the Swiss Federal Institute of Technology in Zurich. Most of the research results presented here were developed during my time as research associate with Jim. Working with him provided both inspiration and challenge at the same time. I can only hope that this book approaches the high standard of excellence exemplified by his own work.

I am indepted to Borer Electronics AG, Solothurn, Switzerland, supporting the preparation of this book. The cooperation originated in a joint project for the European Space Agency which was concerned about methods for encryption of spacecraft telemetry and telecommand links. Many of the new approaches presented here were inspired by the setup considered in this joint project. I wish to thank Hans-Peter Bader who worked with me on the ESA project for his thorough proof-reading of the manuscript. Special thanks are due to Karin Beyeler who efficiently mastered every hurdle in the preparation of the final manuscript, from deciphering my hand-writing, over fighting with text-processors and printers, to drawing the figures.

I also wish to thank Othmar Staffelbach, currently with Gretag, Regensdorf, for his detailed comments and helpful suggestions concerning my doctoral thesis.

Last but not least, I wish to thank my wife Ursula who was awaiting our first baby during the time I wrote my doctoral thesis und who is presently awaiting our second baby, for her continuing support and understanding.

Foreword

It is now a decade since the appearance of W. Diffie and M.E. Hellmann's startling paper, "New Directions in Cryptography". This paper not only established the new field of public-key cryptography but also awakened scientific interest in secret-key cryptography, a field that had been the almost exclusive domain of secret agencies and mathematical hobbyist. A number of excellent books on the science of cryptography have appeared since 1976. In the main, these books thoroughly treat both public-key systems and block ciphers (i.e. secret-key ciphers with no memory in the enciphering transformation) but give short shrift to stream ciphers (i.e., secret-key ciphers with memory in the enciphering transformation). Yet, stream ciphers, such as those implemented by rotor machines, have played a dominant role in past cryptographic practice, and, as far as I can determine, remain still the workhorses of commercial, military and diplomatic secrecy systems.

My own research interest in stream ciphers found a natural resonance in one of my doctoral students at the Swiss Federal Institute of Technology in Zurich, Rainer A. Rueppel. As Rainer was completing his dissertation in late 1984, the question arose as to where he should publish the many new results on stream ciphers that had sprung from his research. Because his work had been starkly fundamental, had spanned a wide area, and had been described in his dissertation with remarkable clarity, it seemed a shame to publish this work in fragments as is the usual practice. I thus asked Rainer to prepare a book for this series, one that would expand his dissertation to a broad treatment of stream ciphers. In short, I asked him to fill the yawning gap for a book that would give stream ciphers the thorough attention that their importance merits. Rainer accepted the challenge and produced this book that manages to incorporate many new results into a unified study of stream ciphers.

This is the first book in this Springer series devoted to cryptography, but others are already underway. I hope to make this series into an outlet for significant new treatises on cryptography. Rainer's book has set a high standard for those to follow.

James L. Massey

List of Illustrations

Table of Contents

1 Introduction

With the entry of our world into the information age, the problem of securing and authenticating data has become an inseparable aspect of modern computer and communication systems. More and more data is stored and transmitted by electronic means, and thereby gets exposed openly. The privacy of individuals as well as of organizations depends severely on the possibility of protecting communicated information from unauthorized disclosure and modification.

Cryptographic systems provide secrecy by use of transformations. At the sending site the plaintext message is, under control of the enciphering key, transformed into the ciphertext that is supposed to be unintelligible to any opponent without the secret deciphering key. At the legitimate receiver, the ciphertext is, under control of the secret deciphering key, retransformed into the original plaintext. Cryptographic systems are commonly classified into block and stream ciphers.

Block ciphers divide the plaintext into blocks, usually of fixed size, and operate on each block independently. Block ciphers are therefore simple substitution ciphers and must have large alphabets to prevent cryptanalysis by exhaustive search.

Stream ciphers divide the plaintext into characters and encipher each character with a time-varying function whose time-dependency is governed by the internal state of the stream cipher. After each character that is enciphered, the device changes state according to some rule. Therefore two occurrences of the same plaintext-character will usually not result in the same ciphertext character.

In this way, block and stream ciphers may be identified as cryptographic systems having one or more than one internal state, respectively. This classification may be seen in analogy to error correcting codes which are subdivided into block and convolutional codes.

In synchronous stream ciphers, the next state depends only on the previous state and not on the input, so that the progression of states is independent of the sequence of characters received. Such an enciphering is memoryless, but time varying. The output corresponding to a particular input depends only on the characters before it or after it.

One of the most remarkable of all ciphers is the one-time-pad where the plaintext message is added bit by bit (or in general character by character) to a nonrepeating random sequence of same length. The one-time-pad is also referred to as the Vernam cipher in honor of G. Vernam who developed the principle in 1917 (Kahn 67) for telegraph communications. The remarkable fact about the one-time-pad is its perfect security; assuming a ciphertext only attack, Shannon (Shan 49) proved that even with infinite computing resources the cryptanalyst could never separate the true plaintext from all other meaningful plaintexts. The disadvantage, of course, is the unlimited amount of key needed.

The appealing features of the one-time-pad suggested building synchronous stream ciphers which encipher the plaintext by use of a pseudo-random sequence, thereby removing the requirement of unlimited key. This pseudo-random sequence is, under control of a secret key, generated by a deterministic algorithm called the key stream generator. With a synchronous stream cipher, the known plaintext assumption is equivalent to giving the cryptanalyst access to the key stream. For the system to be secure, it must not be possible to predict any portion of the key stream with higher reliability than just random guessing, regardless of the number of key stream characters already observed. Linear automata produce sequences which are readily predicted and thus are cryptographically weak. They must be combined with nonlinear transformations in order to obtain the desired highly complex pseudo-random sequences.

This book is devoted to the nonlinear system aspect of synchronous stream ciphers, in particular to the generation of highly unpredictable pseudo-random sequences.

Chapter 2 discusses the principle of stream encryption and defines the key stream generator. The linear complexity of the produced key stream is identified as a design parameter of paramount importance. Nonlinear transformations are restricted to the feedforward part of the key stream generator in order to guarantee analyzability, and some properties of the nonlinear feedforward part are derived.

Chapter 3 provides a concise introduction into the algebraic tools suited for the analysis of nonlinear combinations of linear feedback shift registers. Apart from basic definitions, this chapter also contains results of independent interest on arithmetic operations in

extension fields as well as definitions which are not in common use but proved to be convenient in the nonlinear theory of periodic sequences.

In chapter 4, the linear complexity is shown to have valuable properties as a measure for the unpredictability, or equivalently the randomness, of finite sequences. A characterization of binary sequences of length n is obtained in terms of the expectation and the variance of the associated linear complexity. The growth of linear complexity with increasing sequence length n is described using a random walk. "Typical" random sequences are shown to have a "typical" linear complexity profile. For the practically interesting case of periodically repeating a finite random sequence, the expected linear complexity is calculated. The purpose of this chapter is to derive criteria which allow one to judge the unpredictability of deterministically generated random sequences.

Chapter 5 aims at providing a theoretical basis for analyzing nonlinear combinations of periodic sequences in terms of linear complexity of the produced output sequence. The algebraic normal form of GF(2)-functions is chosen as a reference. In the case of nonlinear operations on distinct phases of a maximal-length (m-)sequence, a matrix approach is given which allows complete analyzability, but is in general computational infeasible for cases of practical interest. Key has derived an upperbound on the linear complexity of the produced sequence which depends on the nonlinear order of the employed function and on the order of the recursion of the m-sequence. The probability of selecting a function whose associated output sequence exhibits a linear complexity substantially smaller than the upperbound is shown to tend to zero with increasing prime order of the recursion of the m-sequence. A large class of functions is exhibited for which the linear complexity of the produced sequence is lowerbound by $\binom{L}{k}$, where L denotes the prime order of the recursion of the m-sequence and where k denotes the nonlinear order of the function. In the case of nonlinear operations on sequences with distinct (but not necessarily irreducible) minimal polynomials, it is shown that the linear complexity of the produced output sequence is equal to the nonlinear combining function evaluated over the reals with the sequence arguments replaced by the associated linear complexities, provided the roots of each minimal polynomial are simple and lie in an extension field whose degree is relatively prime to the degree of the extension field containing each root of the other

minimal polynomials.

Chapter 6 discusses the effects of clocking linear feedback shift registers (LFSRs) at rates higher than the system clock. It is shown that such multiple clocking results in simulating a different LFSR from the one physically implemented. The simulated LFSR is completely determined by the original LFSR and the associated speed factor. A random sequence generator, suggested by a linear cipher problem, which employs multiple clocking is analyzed in detail.

In chapter 7, the 0/1 knapsack and related problems are investigated for their applicability in cryptography. It is shown how integer addition, when both the integers and their sum are represented in radix-2 form, may be computed completely in GF(2). The nonlinear order of the function producing the ith bit of the sum can be exactly quantified and is shown to grow exponentially fast with i. These results lead to a complete GF(2)-description of the knapsack when viewed as function from the set of binary N-tuples into the integers, parameterized by the weights.

In chapter 8, a running key generator for a stream cipher system is proposed in which a knapsack is applied to the state of a maximal-length linear feedback shift register. The sequences defined by the individual sum bits of the knapsack are analyzed and are shown to have cryptographically interesting properties. Some small scale simulations of the proposed knapsack stream cipher are included.

Chapter 9 discusses the effects of allowing memory in the nonlinear combining function. The combiner then becomes a finite-state-machine by itself. It is shown that such FSM-combiners can easily be made immune against the correlation attack, a threat to which many recently proposed key stream generators succumb. Real addition of r-ary sequences is shown to define such a correlation resistant structure with memory. When two m-sequences of relatively prime recursion order are added over the reals the linear complexity of the resulting sum sequence is shown in general to be equal or very close to the product of the periods of the input sequences.

The book is equipped with many illustrating examples which are designed to ease the understanding of the theoretical concepts and which are taken, whenever possible, from practical and realistic situations.

2 Stream Ciphers

For practical reasons (such as delay and ease of processing) the semi-infinite message stream is usually subdivided into fixed size entities which are enciphered serially. There are two fundamentally different approaches to how a sensible enciphering could be accomplished. When the enciphering transformation operates on each such message entity independently, then one speaks of a block cipher. Thus block ciphers are simple substitution ciphers and must have necessarily large alphabets to prevent cryptoanalysis by brute force. This is why the suggestive name "block" is used to designate the size of a message entity. Stream ciphers, in contrast, encipher each message entity with a time-varying function whose time dependency is governed by the internal state of the stream cipher. Since, for this enciphering principle, the message entities are not required to be large, one speaks of "character" to designate the size of a message entity. After each character is "enciphered" the stream cipher changes state according to some rule. Therefore two occurrences of the same plaintext character usually do not result in the same ciphertext character.

To obtain a clear distinction between block and stream ciphers, the criterion of memory can be used (see Fig. 2.1).

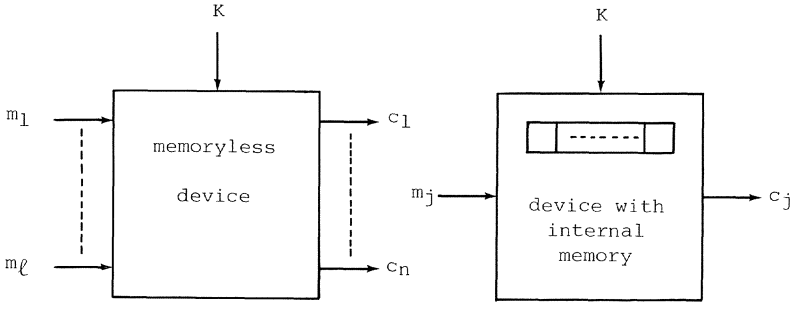


Fig. 2.1. The basic 2 enciphering principles.

A block cipher then specifies a memoryless device which transforms a message block $\underline{m} = [m_1,..,m_1]$ under control of a key K into a cipher-

text block $\underline{c} = [c_1,\ldots,c_n]$, where the message text alphabet and the ciphertext alphabet usually are identical. A stream cipher specifies a device with internal memory that transforms the jth digit m_j of the message stream into the jth digit c_j of the ciphertext stream by means of a function which depends on both the secret key K and the internal state of the stream cipher at time j.

In this way, cryptographic systems are classified into block and stream ciphers, in analogy to error correcting codes which are subdivided into block and convolutional codes. For a fixed key K, the block cipher will transform identical message blocks into identical ciphertext blocks. This allows the active wiretapper to insert, delete or replay previously recorded ciphertext blocks (unless some additional protocol is employed that controls the sequence of message blocks). Likewise block enciphering allows the passive wiretapper to search through the ciphertext for matches. This is a serious threat when fixed information entities (such as salary records) are treated as plaintext blocks. Conversely, since the stream cipher encrypts each character under a time varying function of the key, it prevents deletion, insertion or replay of ciphertext, as well as ciphertext searching. One may say that a stream cipher is inherently more secure than a block cipher because of the additional dimension offered by the use of memory. But note that a block cipher may always be augmented into a stream cipher by associating memory to it. Then the block cipher merely is used as a nonlinear function. For the most prominent block cipher at this time, the data encryption standard (DES), there exist many different proposals as to how its security could be increased by introducing memory into the transformation (see Denn 83 for a brief survey).

Stream ciphers may be further subdivided into synchronous and self-synchronizing systems. In synchronous stream ciphers, the next state depends only on the previous state and not on the input so that the succession of states is independent of the sequence of characters received. Consequently, the enciphering transformation is memoryless, but time-varying. But the device itself is not memoryless, it needs the internal memory to generate the necessary state sequence. It is natural therefore, in a synchronous stream cipher, to separate the enciphering transformation from the generation process of the time-varying parameter that controls the enciphering transformation (see Fig. 2.2).

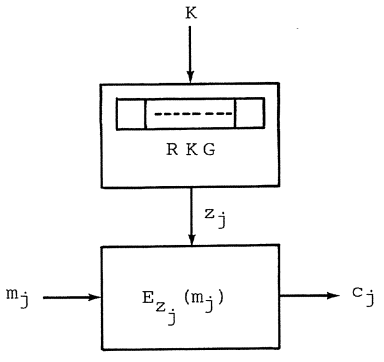


Fig. 2.2. A decomposed synchronous stream cipher

The sequence $\tilde{z} = z_0, z_1, \ldots$, which controls the enciphering, is called the key stream or running key. The deterministic automaton which produces the key stream from the actual key K and the internal state is called the running-key generator. Whenever the key K and the internal state are identical at the sender and receiver, the running keys necessarily are also identical, and deciphering is easily accomplished. One says that the running key generators at the sending and receiving site are synchronized with each other. Whenever the running key generators lose synchronism, deciphering becomes impossible, and means must be provided to reestablish synchronization.

In contrast, in self-synchronizing stream ciphers, the deciphering transformation has finite memory with respect to the influence of past characters so that an erroneous or lost ciphertext character causes only a fixed number of errors in the deciphered plaintext, after which again correct plaintext is produced.

Stream ciphers are in general difficult to analyze because of the internal memory and the utilized nonlinear transformations. This may again be seen in analogy to coding theory, where the majority of publications deal with block codes. In fact, Prof. Widman indicated at the Zurich Seminar 1984 on digital communications that the importance of stream ciphers is not appropriately reflected in the open literature, whereas only a small percentage of publications on cryptographic systems deal with stream ciphers, he said, stream ciphers make up the vast majority of applications.

2.1 Theoretical versus Practical Security

One of the most remarkable of all ciphers is the one-time-pad (also referred to as Vernam cipher (Kahn 67)) where the ciphertext is the bit by bit modulo-2 sum of the plaintext message and a nonrepeating random sequence of same length. Since in GF(2) addition and subtraction are the same, the deciphering is accomplished by applying the enciphering a second time. Fig. 2.3 illustrates the one-time-pad.

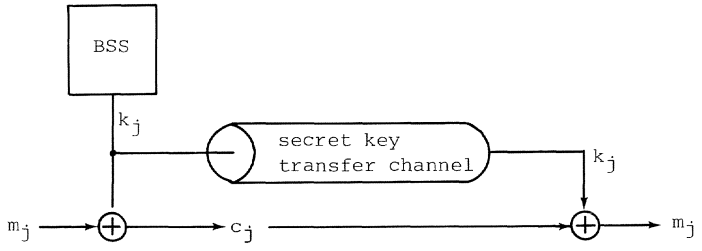


Fig. 2.3. Principle of one-time-pad

The basic principle of the one-time-pad is to statistically decouple ciphertext and plaintext by use of a truly random key sequence. The device which emits such a random sequence, i.e. a sequence where each bit is equally likely to be 0 or 1 independently of the preceding bits, is called a binary symmetric source (BSS). In short, a BSS puts out a fair coin tossing sequence. The remarkable fact about the one-time-pad is its perfect security. Assuming a ciphertext only attack, Shannon (Shan 49) proved that even with infinite computing resources the cryptanalyst could never separate the true plaintext from all other meaningful plaintexts. A cryptographic system is said to offer perfect secrecy (or is said to be unconditionally secure (Diff 79)), if the mutual information between the plaintext message and the associated ciphertext is zero, independently of the length of the message. Then intercepting the ciphertext does not convey any information whatsoever about the plaintext to the cryptanalyst, he is still confronted with the a priori probability distribution over the set of possible messages. The one-time-pad of Fig. 2.3 offers perfect secrecy since adding of a random key sequence to the plaintext stream causes the conditional probability of a ciphertext given the associated plaintext message to be independent of the plaintext. Via Bayes' rule, this implies that the conditional probability of a

plaintext message given the associated ciphertext is independent of the ciphertext. Hence the conditional uncertainty about the plaintext given the associated ciphertext is independent of the ciphertext, thereby assuring the desired statistical independence. It is obvious that perfect secrecy may only be obtained as long there are available as many random key bits as there are information bits in the plaintext stream. But this is the crucial point: the only way to reproduce the truly random key stream at the receiver is to prerecord it and to distribute the identical key stream tapes to the sender and receiver. Clearly such a system is generally impractical in light of both the key generation time and the key distribution problems. For this reason, the applicability of the one-time-pad has been mainly limited to espionage circles (or similar situations) where complete secrecy is of paramount importance. Nevertheless it is conceivable that in the future one-time-pad systems will be manufactured which will exploit the tremendous possibilities of new storage technologies (such as holography). The feasibility of such systems will depend on the availability of random processes as well as on the speed with which the random quantities may be measured and stored. Necessarily the key distribution in such systems would have to take place physically, since enciphering of the new key would consume as much old key as new key that is transmitted.

The operational disadvantages of the one-time-pad have led to the development of synchronous stream ciphers which encipher the plaintext in much the same way as the one-time-pad, but with a deterministically generated random sequence. Thus the running key generator in Fig. 2.2 has, controlled by the true key K, the task to simulate a random sequence which then is used to encipher the plaintext. The security of such a synchronous stream cipher now depends on the "randomness" of the key stream. Assuming a known plaintext attack, the cryptanalyst has full access to the running key. For the system to be secure, the key stream must be unpredictable: regardless of the number of key stream digits observed, the subsequent key stream digits must be no better predictable than by just randomly guessing them. This implies that it is unfeasible for the cryptanalyst to determine the true key as he would be able to reliably predict the key stream. A necessary requirement for unpredictability is a long period of the key stream. The period defines a linear recursion, thus knowing the value of the period and the first period of the key stream determines completely the remainder of the key stream. If the keystream also satisfies linear or nonlinear recursions of order

smaller than the period, then knowing the recursion and its initial state completely specifies the remainder of the key stream. Finding the shortest nonlinear feedback shift register able to produce a given sequence is in general an unfeasible task. But for finding the shortest linear feedback shift register able to produce a given sequence there exists an efficient synthesis procedure (Mass 69). Consequently, for the key stream to be unpredictable, it is necessary that its linear complexity (which is the length of this shortest linear feedback shift register able to produce the key stream) be large. A large linear complexity is a necessary but not sufficient condition to ensure the unpredictability of the key stream. For instance, the sequence $(0.....01)^{\infty}$ consisting of some zeros and an appended 1 which are periodically repeated, has maximum linear complexity, only the circulating shift register may produce it. But clearly this sequence is readily predicted by the allzero sequence and thus cryptographically of little use. Finally, unpredictability requires that, independent of the preceding digit, the next key stream digit appears to be drawn from a uniform distribution. Therefore the key stream necessarily must have uniform statistics, i.e. an equal distribution of single digits, of pairs, triples of digits, etc. The most fundamental parameter already listed for estimating the unpredictability of a key stream is the linear complexity. Since the period of a sequence always is at least as large as its linear complexity, a large linear complexity implies a large period. Typical random sequences exhibit a typical growth of linear complexity (compare chapter 4) with an increasing number of considered sequence digits. Thus when the key stream has such a typical linear complexity profile, it is likely to exhibit also uniform statistics. In contrast, uniform distribution properties in a periodic sequence do not imply a large linear complexity. For instance, maximal-length sequences which are also called pseudonoise (PN-)sequences for their excellent distribution properties, have minimum linear complexity with respect to the period length. Binary maximal-length sequences of period 2^L-1 are completely determined by knowing only 2L sequence bits.

Consequently, the criterion of a typical linear complexity profile with a large final value, supported by the requirement of uniform statistics, provides a sensible basis for estimating the unpredictability, and thus the randomness quality of the key stream.

2.2 The Key Stream Generator

In a synchronous stream cipher, the running key generator may in general be viewed as autonomous finite state machine (see Fig. 2.4).

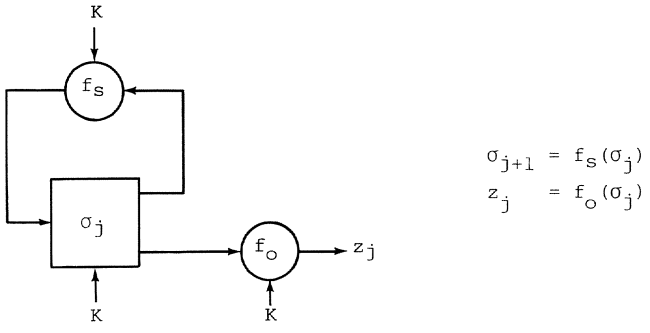


Fig. 2.4. The key stream generator as an autonomous finite state machine

The key stream generator as a finite state machine consists of an output alphabet and a state set, together with two functions and an initial state σ_0. The next state function f_s maps the current state σ_j into a new state σ_{j+1} from the state set, and the output function f_o maps the current state σ_j into an output symbol z_j from the output alphabet. The key K may determine the next state function and the output function as well as the initial state. The basic problem of key stream generator design in the context of finite state machines is to find next state functions f_s and output functions f_o which are guaranteed to produce a running key $\tilde{z}$ that satisfies the basic requirements of large period, large linear complexity and uniform distribution properties. The determination of these basic performance criteria, of course, implies the analyzability of the running key generator. The construction of secure running key generators necessarily implies the introduction of nonlinear transformations, which greatly complicates the indispensable analysis. There exist implicit and explicit methods for introducing nonlinear effects. For instance, letting one linear feedback shift register (LFSR) clock a second LFSR is an implicit way to generate nonlinear effects. Explicit methods directly apply nonlinear functions; in the autonomous automaton description of the running key generator (Fig. 2.4), the next state and the output mappings are candidates for nonlinear transformations. Unfortunately, the theory of autonomous au-

tomata whose change of state function is nonlinear (e.g. nonlinear feedback shift registers) is not very well developed so that a running key generator employing nonlinear feedback will only be very limitedly analyzable. Different is the situation when a linear autonomous automaton is combined with a nonlinear output mapping to be used as running key generator; such a structure is well within analytic reach. In this context, it is convenient to subdivide the running key generator into a driving part and a combining part. The driving part then governs the state sequence of the running key generator and is responsible for providing sequences of large periods and good short- and midterm statistics. For instance, the driving part could consist of a set of maximal-length LFSRs. In contrast, the combining part then has the task of significantly increasing the linear complexity of the key stream in order to make infeasible any linear attack (such as the Berlekamp-Massey LFSR synthesis algorithm), but without destroying the good distribution properties provided by the driving sequences. It is rewarding to allow the nonlinear combining function to contain memory (see Fig. 2.5).

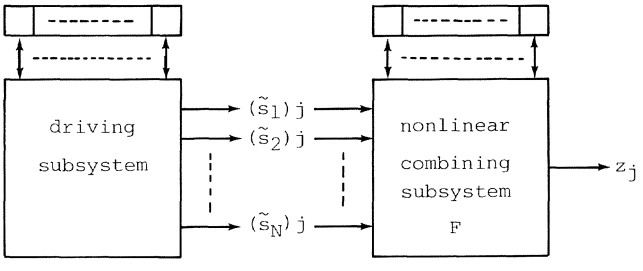
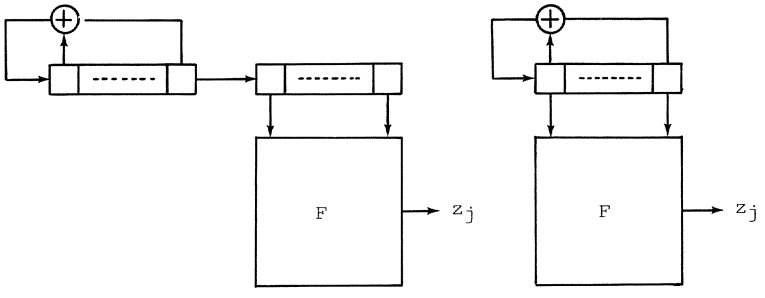


Fig. 2.5. The conceptual distinction between driving and nonlinear combining subsystem in a general key stream generator.

By allowing F to contain some memory, part of the total state memory of the running key generator is dedicated to the feedforward part which in fact becomes a finite state machine by itself. This greatly increases the number of options available for the nonlinear combining subsystem F. Often when F contains memory in the mathematical sense, it need not employ memory in the physical sense, since F may directly be applied to the state of the driving subsystem. This is illustrated (see Fig. 2.6) by the nonlinear state filter generator (which is investigated in great detail in section 5.1).



(a) explicit memory (b) implicit memory

Fig. 2.6. The state filter generator (a) explicitly showing the memory associated to the nonlinear combiner F, and (b) equivalent practical realization.

The described systematic approach to running key generator design motivates the search for nonlinear combining subsystems F which satisfy the following requirements:

(1) F transfers the statistical properties of the periodic driving sequences to the generated running key in the sense that, when the input sequences are true q-ary coin tossing sequences, so is the output sequence.

(2) F maximizes the period of the running key relative to the periods of the driving sequences. This offers the needed high potential for a large linear complexity of the key stream.

(3) F maximizes the linear complexity of the running key relative to the linear complexities of the driving sequences. Linear methods to reliably predict future key stream digits from any observed part of the key stream are made unfeasible.

(4) F does not leak; any modularizing attack (divide and conquer) directed towards the submodules of the driving subsystem of the key stream generator must fail.

(5) F is easy to implement and fast.

(6) If necessary, F is easily controlled by the key. (The key could control only the driving subsystem).

Key stream generators built by nonlinearly combining periodic sequences emitted from a linear driving subsystem are in correspondence to Shannon's two cryptographic principles (Shan 49) of "confusion" and "diffusion". This method employs confusion by the use of complicated nonlinear transformations, and employs diffusion by making each digit of the key affect many digits of the ciphertext (where we tacitly have assumed that the key governs the state sequence of the driving subsystem of the running key generator).

2.3 The Synchronization (Problem) of Stream Ciphers

Synchronous stream ciphers require, as their name indicates, perfect synchronism between the encrypting and the decrypting device. When digits are lost or added during transmission, then the keystream produced at the receiving site will be added with some offset to the cipherstream which effectively encrypts the cipherstream a second time. To reestablish synchronism at the receiver usually involves searching over all possible offsets that the keystream could have experienced, or notifying the sender that the cryptosystems should be reinitialized. In both cases, large portions of the transmitted data may be lost. This difficulty of reestablishing synchronization is considered to be the major disadvantage of synchronous stream ciphers. But this apparently weak point is simultaneously responsible for the frustration of almost all active attacks. Injection, deletion or replay of ciphertext cause immediate loss of synchronization. The only active attack left to the wiretapper is to selectively alter digits in the ciphertext, which corresponds to simulating worse channel conditions than there actually are. Infrequent active wiretapping of this sort will be corrected by the error control system and frequent changes of ciphertext digits will cause the error control system to reject the received message and to notify the receiver. Consequently, the synchronization "problem" of synchronous stream ciphers is one reason for their cryptographically appealing properties. There is a possibility of tradeoff between security and synchronization difficulty. When each key stream digit is derived from a fixed number of preceding ciphertext digits, then the stream cipher becomes self-synchronizing (see Fig. 2.7).

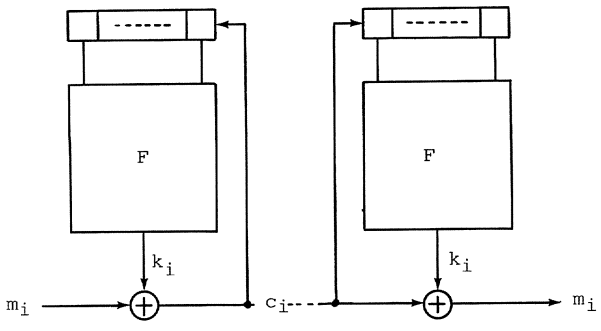


Fig. 2.7. Principle of a self-synchronizing stream cipher

The deciphering transformation has finite memory with respect to the influence of past digits so that an erroneous or lost ciphertext digit only propagates forward for M digits. The receiver resynchronizes automatically after M consecutive correct ciphertext digits have been received. Each key digit is functionally dependent on the initial state of the running key generator and on all the preceding ciphertext digits. This recursive procedure of generating the key stream diffuses the statistical properties of the plaintext over the ciphertext. As a consequence, self-synchronizing stream ciphers are nonperiodic if the message stream is nonperiodic (which is true in general). Self-synchronizing stream ciphers protect against ciphertext searching, but because of their self-synchronizing property they do not (or only slightly) protect against injections, deletions and replay of ciphertext. In general, self-synchronizing stream ciphers offer a lower level of security since arguments and function values of the key stream generating process are known to the cryptanalyst. (Note that the memory in Fig. 2.7 is filled with ciphertext digits). The real disadvantage of self-synchronizing stream ciphers, however, is their limited analyzability because of the dependence of the key stream on the message stream.

When a communications link employs some means for packet or frame synchronization (as is often the case in digital communications), then the synchronous stream cipher may be supplied with some sort of self-synchronizing property without decreasing the security level. Let $\sigma_0(i)$ be the initial state of the stream cipher at the beginning of frame i. When the initial state $\sigma_0(i+1)$ of the stream cipher at the beginning of frame i+1 is computed by some deterministic func-

tion g from $\sigma_0(i)$, i.e. $\sigma_0(i+1) = g(\sigma_0(i))$, then one may say that the stream cipher has the frame-self-synchronizing feature. When $\sigma_0(0)$ and the key are identical at the sending and receiving site, then independently of all intermediate synchronization losses, $\sigma_0(i)$ will be identical since the deterministic function g is computed locally both at the sender and at the receiver. For instance, advancing the state by a fixed offset larger than the frame length is a possible option for g. Another solution for g is given by $\sigma_0(i+1) = \overline{\sigma_1(i)}$, that is, the preceding initial state is advanced by one (or possibly more) positions and then complemented to form the subsequent initial state. Care must be taken to avoid that the allzero state results from complementing (since otherwise one frame will be transmitted in plain) and that a possible repetition of part of the key stream is unlikely.

3 Algebraic Tools

This chapter provides a concise introduction into algebraic tools suited for the analysis of nonlinear combinations of LFSRs. Many of the definitions stated may be either found in (Mass 84b) or some good text book on finite fields such as (Lidl 83). Nevertheless this chapter also contains results of independent interest and definitions which are not in common use but proved to be convenient in the nonlinear theory of periodic sequences. The purpose of this chapter is to collect the basic definitions.

3.1 Finite Fields and Polynomials

For a prime p, let F_p be the set [0,1,...,p-1] of integers. Then the algebraic system $\langle F_p, \oplus, \odot \rangle$ where the field operations are defined for a and b in F_p by

$$a \oplus b = R_p\,(a + b) \tag{3.1a}$$

$$a \odot b = R_p\,(a\ b) \tag{3.1b}$$

is called the Galois field of order p, denoted GF(p) or simply the finite field with p elements. $R_p(n)$ denotes the unique remainder r when the integer n is divided by the prime p. The operations on the rights side of (3.1) are integer addition and multiplications, respectively. The simplest, but perhaps most important, of all fields is GF(2), for which $F_2=\{0,1\}$ and the rules for multiplication and addition are those of "modulo-2 arithmetic". In GF(2), the elements 0 and 1 are called binary digits.

A polynomial over GF(p) in an indeterminate X is an expression of the form

$$a(X) = a_0 + a_1X + \ldots + a_nX^n \tag{3.2}$$

where n is a nonnegative integer and the coefficients a_i, $0 \leqslant i \leqslant n$, are elements of GF(p). If a(X) is a polynomial with a_n nonzero, then a_n and a_0 are called the leading coefficient and the constant term

of a(X), respectively, while n is called the degree of a(X), denoted deg(a(X)). If the leading coefficient of a(X) is 1, then a(X) is called a monic polynomial. The polynomial, all of whose coefficients are 0, is called the zero polynomial and is also denoted by 0. The context then must make clear whether 0 stands for the zero element of GF(p) or the zero polynomial. By way of convention, the degree of the zero polynomial is deg(0) = $-\infty$. Polynomials of degree ≤ 0 are called constant polynomials. Let a(X) and b(X) be two polynomials over GF(p) of degree n and m, respectively, where without loss of generality n is assumed to be at least as large as m. Define polynomial addition as

$$a(X) + b(X) = \sum_{i=0}^{n} (a_i \oplus b_i) X^i \tag{3.3}$$

and define polynomial multiplication as

$$a(X)b(X) = \sum_{k=0}^{n+m} c_k X^k \tag{3.4a}$$

where

$$c_k = \sum_{\substack{i+j=k \\ 0 \leq i \leq n \\ 0 \leq j \leq m}} a_i \odot b_j \quad . \tag{3.4b}$$

Then the set of all polynomials of finite degree over GF(p) when equipped with the operations of polynomial addition (3.3) and polynomial multiplication (3.4) forms a ring called the polynomial ring over GF(p) and denoted by GF(p)[X].

The operation of polynomial multiplication as defined in (3.4) allows one to introduce the concept of divisibility. The polynomial a(X) in GF(p)[X] divides the polynomial c(X) in GF(p)[X] if there exists a polynomial b(X) in GF(p)[X] such that c(X) = a(X)b(X). One also says that a(X) is a divisor of c(X), or that c(X) is a multiple of a(X), or that c(X) is divisible by a(X). The greatest common divisor of the polynomials a(X) and b(X) (at least one of which is not

the zero polynomial), denoted gcd(a(X),b(X)), is the monic polynomial of highest degree that divides both a(X) and b(X). If gcd(a(X), b(X)) = 1, then a(X) and b(X) are said to be relatively prime. The counterpart to the notion of the greatest common divisor is that of the least common multiple. For nonzero polynomials, a(X) and b(X) in GF(p)[X], there exists a uniquely determined monic polynomial, denoted lcm(a(X),b(X)), which is the polynomial of smallest degree divisible by both a(X) and b(X).

The polynomial property analogous to the integer property of "primeness" is "irreducibility". A polynomial p(X) in GF(p)[X] is said to be irreducible over GF(p) if p(X) has positive degree and p(X) = a(X)b(X) with a(X) and b(X) in GF(p)[X] implies that either a(X) or b(X) is a constant polynomial. Or in other words, a polynomial of positive degree is irreducible over GF(p) if it allows only trivial factorization. As primes p can be used to construct the Galois field of order p, irreducible polynomials p(X) can be used to construct the extension field of order p^m. Let p(X) in GF(p)[X] be irreducible over GF(p) of degree m. Then the algebraic system $\langle E,\oplus,\odot\rangle$ where E is the set of all polynomials in GF(p)[X] with degree less than m and where the field operations are defined for a(X) and b(X) in E by

$$a(X) \oplus b(X) = R_{p(X)}\,(a(X) + b(X)) \tag{3.5a}$$

$$a(X) \odot b(X) = R_{p(X)}\,(a(X)b(X)) \tag{3.5b}$$

is called the Galois field of order p^m, denoted $GF(p^m)$, or equivalently, the finite field with p^m elements. $R_{p(X)}(c(X))$ denotes the unique remainder polynomial r(X) when the polynomial c(X) is divided by the irreducible polynomial p(X). The operations on the right side of (3.5) are polynomial addition and multiplication, as defined in GF(p)[X]. It is common practice to write GF(q) to denote the finite field of q elements when one is dealing with properties that apply whether $q = p$ or $q = p^m$ with $m \geq 1$ for some prime p. If a(X) is in GF(q)[X], then replacement of the indeterminate X in a(X) by a fixed element of GF(q) yields a well-defined element of GF(q). This is known as the principle of substitution.

An element d in GF(q) is called a root (or a zero) of the polynomial a(X) in GF(q)[X] if a(d) = 0. This is equivalent to stating that (X-d) divides a(X). If a(X) is divisible by $(X-d)^k$, but not by $(X-d)^{k+1}$, then k is called the multiplicity of the root d. When k =

1, then d is called a simple root of a(X), and when $k \geq 2$, then d is called a multiple root of a(X). If a(X) is an irreducible polynomial in GF(q)[X] of degree larger than 1, it may not have any root in GF(q). Let a(X) in GF(q)[X] have positive degree and E be an extension field of GF(q). Then a(X) is said to split in E if a(X) can be written as a product of linear factors in E[X], that is, if there exist elements $\alpha_1, \alpha_2, \ldots, \alpha_n$ in E such that

$$a(X) = a_n(X - \alpha_1)(X - \alpha_2) \ldots (X - \alpha_n) \tag{3.6}$$

The field E is called the splitting field of a(X) over GF(q); E is the smallest field containing all the roots of a(X) in the sense that no proper subfield of E contains all the roots of a(X). If p(X) in GF(q)[X] is irreducible over GF(q) of degree m then its splitting field is $GF(q^m)$. Moreover, all its roots are simple and are given by the m distinct elements $\alpha, \alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{m-1}}$ of $GF(q^m)$. Regardless of whether α in $GF(q^m)$ is a root of an irreducible polynomial in GF(q)[X] of degree m or not it is customary to call the elements $\alpha, \alpha^q, \ldots, \alpha^{q^{m-1}}$ in $GF(q^m)$ the conjugates of α with respect to GF(q). Every element α in $GF(q^m)$ has associated a unique monic irreducible polynomial in GF(q)[X] of smallest degree whose root it is. This polynomial is called the minimum polynomial of α, denoted $m_\alpha(X)$. Consequently, the conjugates of α in $GF(q^m)$ are distinct if and only if the minimum polynomial of α over GF(q) has degree m. Otherwise, the degree k of this minimum polynomial $m_\alpha(X)$ is a proper divisor of m, and then the conjugates of α with respect to GF(q) are distinct elements $\alpha, \alpha^q, \ldots, \alpha^{q^{k-1}}$, each repeated m/k times. The distinct and simple roots $\alpha, \alpha^q, \ldots, \alpha^{q^{k-1}}$ of $m_\alpha(X)$ in GF(q)[X] are called the conjugate roots. We will introduce the notion of conjugate root sequence for the semi-infinite series of conjugates,

$$\{\alpha^{q^i}\}_{i=0}^{\infty} = \alpha, \alpha^q, \alpha^{q^2}, \ldots \tag{3.7}$$

Then the period of the conjugate root sequence is equal to the degree of the minimum polynomial of α over GF(q). The usefulness of the minimum polynomial lies in the fact that α is a root of an arbitrary polynomial a(X) in GF(q)[X] if and only if $m_\alpha(X)$ is a factor of a(X). We will call an element α in $GF(q^m)$ a quintessential element of $GF(q^m)$ when its conjugate root sequence has maximum period

m, or equivalently, when its minimum polynomial has degree m. The order of an element α in $GF(q^m)$ is defined to be the period of the semi-infinite sequence

$$\{\alpha^i\}_{i=0}^{\infty} = 1, \alpha, \alpha^2, \ldots \tag{3.8}$$

Since the nonzero elements of $GF(q^m)$ form a group of order q^m-1 under multiplication, it follows that the order of any nonzero element α in $GF(q^m)$ must divide q^m-1. A nonzero element α in $GF(q^m)$ is called a primitive element if its order has the maximum possible value, q^m-1. Because q and q^m-1 are relatively prime, all the conjugate roots of α must have the same order as α. In particular, when α in $GF(q^m)$ is a primitive element, then so are all its conjugates in $GF(q^m)$, and it is appropriate to call the associated minimum polynomial of α a primitive polynomial. In general, when p(X) in GF(q)[X] is a nonzero polynomial with $p(0) \neq 0$, then the least positive integer T for which p(X) divides X^T-1 is called the order of p(X), denoted by ord(p(X)). If $p(0) = 0$ then $p(X) = X^i g(X)$ with i being a positive integer and $g(0) \neq 0$; the order of p(X) then is defined to be the order of g(X). The order of a polynomial p(X) is sometimes also called the period of p(X) or the exponent of p(X). For irreducible polynomials p(X) in GF(q)[X] of degree m, the order of p(X) is equal to the order of any root α of p(X) in the multiplicative group of $GF(q^m)$. When α in $GF(q^m)$ is a primitive element, then every nonzero element β in $GF(q^m)$ may be expressed as a power of α. Since every nonzero element β in $GF(q^m)$ has associated a unique minimum polynomial $m_\beta(X)$ in GF(q)[X], the membership of β in a conjugate root set separates the nonzero elements of $GF(q^m)$ into disjoint sets. Consequently, the q^m-1 possible integer exponents of the primitive element α are separated according to the membership of α^k to a conjugate root set into the so-called cyclotomic cosets. The cyclotomic coset containing k consists of all the distinct integers of the form kq^i, $i \geq 0$, in the ring of integers mod q^m-1.

At the end of this subsection on finite fields and polynomials, we prove a fundamental property of the elementary field operations which is of some independent interest.

Proposition 3.1. Root Algebra

If α is an element of $GF(q^m)$-$GF(q)$ and β is an element of $GF(q^n)$-$GF(q)$, whose conjugate root sequences $\{\alpha^{q^i}\}_{i=0}^{\infty}$ and $\{\beta^{q^i}\}_{i=0}^{\infty}$ have period m' and n', respectively, where gcd $(m',n') = 1$, then $\gamma = \alpha * \beta$ has a conjugate root sequence $\{\gamma^{q^i}\}_{i=0}^{\infty}$ of period $m'n'$ for * being either addition or multiplication.

Proof:

$$\gamma^{q^{m'n'}} = (\alpha * \beta)^{q^{m'n'}} = \alpha^{q^{m'n'}} * \beta^{q^{m'n'}} = \alpha * \beta = \gamma \qquad (3.9)$$

(Note that for * denoting addition the above line holds because q is a power of the field characteristic p). Thus the period t of the conjugate root sequence of γ divides $m'n'$.

Because m' and n' are relatively prime, t may always be written as the product of two integers $t = t_1 t_2$ where t_1 divides m' and t_2 divides n'.

Since t divides $t_1 n'$ we obtain

$$(\alpha * \beta)^{q^{t_1 n'}} = \alpha * \beta \; . \qquad (3.10)$$

But it also holds that

$$(\alpha * \beta)^{q^{t_1 n'}} = \alpha^{q^{t_1 n'}} * \beta \; . \qquad (3.11)$$

(3.10) and (3.11) imply that m' divides $t_1 n'$; but since gcd(m,n) = 1, m' must be a divisor of t_1.

Since t divides $m' t_2$ we obtain

$$(\alpha * \beta)^{q^{m' t_2}} = \alpha * \beta \; . \qquad (3.12)$$

But it also holds that

$$(\alpha * \beta)^{q^{m't_2}} = \alpha * \beta^{q^{m't_2}} \qquad (3.13)$$

(3.12) and (3.13) imply that n' divides $m't_2$, hence n' must be a divisor of t_2.

Since $t = t_1t_2$ divides $m'n'$ but m' divides t_1 and n' divides t_2 we conclude that $t = m'n'$.

The Chinese Remainder Theorem and $\gcd(m',n') = 1$ assure that the set $\{\alpha^{q^i} * \beta^{q^i} : 0 \leq i < m',\ 0 \leq j < n'\}$ is a complete conjugate root set and that the associated minimum polynomial $m_\gamma(X)$ in GF(q)[X] is irreducible of degree $m'n'$. When α and β are quintessential elements of their fields $GF(q^m)$ and $GF(q^n)$, respectively, then $\gamma = \alpha * \beta$ is a quintessential element of $GF(q^{mn})$. We may also generalize proposition 3.1 to the case of iterated application of the operation *.

Corollary 3.2.

If α_j is an element of $GF(q^m)$-GF(q), $j = 1,\ldots,N$, whose conjugate root sequence $\{\alpha_j^{q^i}\}_{i=0}^{\infty}$ has period m'_j and $\gcd(m'_j,m'_k) = 1$ for all $j \neq k$, then $g = \alpha_1 * \alpha_2 * .. * \alpha_N$ has a conjugate root sequence $\{\gamma^{q^i}\}_{i=0}^{\infty}$ of period $m'_1m'_2\ldots m'_N$.

Proof: $\gamma_1 = \alpha_1 * \beta_2$ has a conjugate root sequence of period $m'_1m'_2$ by proposition 3.1. The same argument applies to $\gamma_2 = \gamma_1 * \alpha_3$ when $\gcd(m_1m_2,m_3) = 1$. Thus γ_2 has a conjugate root sequence of period $m'_1m'_2m'_3$. The Corollary follows now by induction.

3.2 Linear Feedback Shift Registers (LFSRs) and Sequences

One of the most useful devices in the generation of running keys is the linear feedback shift register (LFSR), the general form of which is shown in Fig. 3.1.

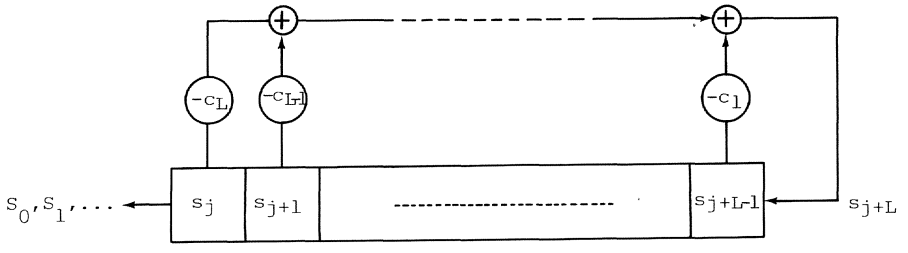


Fig. 3.1. A general Linear Feedback Shift Register

Both the feedback coefficients $c_1, c_2, \ldots, c_L$ and the output digits $s_0, s_1, s_2, \ldots$ are elements of GF(q). The number L of delay cells used for the generation of the output sequence is called the length of the LFSR, whereas an individual delay cell is referred to as a stage of the LFSR. The contents of the L stages is called the state of the LFSR. The L digits $s_0, s_1, \ldots, s_{L-1}$ initially loaded into the L stages specify the initial contents (or initial state) of the LFSR and can be arbitrarily selected. The feedback coefficients are commonly summarized in the feedback polynomial

$$C(D) = 1 + c_1 D + c_2 D^2 + \ldots c_L D^L \qquad (3.14)$$

which has degree at most L and in which the indeterminate is called D. Let denote $\tilde{s} = s_0, s_1, s_2, \ldots$ the semi-infinite output sequence produced by the LFSR. The digits s_j, $j \geq L$, of $\tilde{s}$ are determined by the LFSR and its initial contents. The recursion implemented by the LFSR is readily seen from Fig. 3.1 to be

$$s_{j+L} = -\sum_{i=1}^{L} c_i s_{j+L-i} \qquad j \geq 0 \qquad (3.15)$$

which is referred to as an L-th order linear recurrence relation.

The sequence $\tilde{s}$ is periodic if there exists a positive integer T such that $s_{j+T} = s_j$ for all $j \geq 0$. When $c_L \neq 0$, then the output sequence $\tilde{s}$ is periodic and the LFSR is called nonsingular. Every state of a nonsingular LFSR has a predecessor which consequently must be unique. This implies that all states of a nonsingular LFSR lie on closed cycles in its associated state-transition diagram. The sequence $\tilde{s}^t = s_t, s_{t+1}, s_{t+2}, \ldots$ is called the t-th phase shift of $\tilde{s}$, when $\tilde{s}$ is a periodic sequence, or equivalently, when the LFSR producing $\tilde{s}$ is nonsingular. It follows that all distinct phase shifts of $\tilde{s}$ correspond to all distinct states on a closed cycle in the state transition diagram of the associated LFSR. Therefore all distinct phase shifts of $\tilde{s}$ are termed cyclically equivalent. When two sequences $\tilde{r}$ and $\tilde{s}$ may be generated by the same nonsingular LFSR but have associated distinct state cycles of the LFSR, then r and s are called cyclically distinct sequences.

The D-transform S(D) of the semi-infinite output sequence $\tilde{s}$ over GF(q) of the LFSR is defined as

$$S(D) = s_0 + s_1 D^1 + s_2 D^2 + \ldots = \sum_{j=0}^{\infty} s_j D^j \quad . \tag{3.16}$$

S(D) is a formal power series in the indeterminate D introduced to store all the sequence digits in correct order. The adjective "formal" reflects the idea that convergence or divergence of the power series is irrelevant in this context. If the algebraic operations of addition and multiplication for formal power series are defined to be the natural extensions of the corresponding polynomial operations (3.3) and (3.4), then the set of all formal power series over GF(q), equipped with this addition and multiplication, forms a communitative ring with identity, called the ring of formal power series over GF(q) and denoted by GF(q)[[D]]. The polynomial ring GF(q)[D] is contained as a subring in GF(q)[[D]]. This allows one to form the product C(D)S(D) = P(D) between the feedback polynomial C(D) and the formal power series S(D) of the output sequence $\tilde{s}$ of the LFSR. By the LFSR recursion (3.15) all the coefficients p_i, $i \geq L$, are zero. Thus P(D) is a polynomial over GF(q) of degree less than L, and

$$S(D) = \frac{P(D)}{C(D)} \qquad \deg(P(D)) < L \tag{3.17}$$

which is known as the fundamental identity of formal power series of linear recurring sequences. The terms of $\tilde{s}$ may be computed by long division from the rational function (3.17). There are q^L choices for the coefficients of P(D) and q^L choices for the initial contents of the LFSR. Therefore one may say that the sequences producible by the LFSR of Fig. 3.1 are in one-to-one correspondence to the rational functions P(D)/C(D) with deg(P(D)) < L.

The D-transform is a useful instrument to describe the multiplexing of different sequences. When the digits s_j of $\tilde{s}$ are spread to the positions k + jN, where k denotes a fixed offset, then the D-transform $D^k S(D^N)$ exactly describes the corresponding positions in the multiplexed sequence.

3.3 Minimal Polynomial and Traces

A linear recurring sequence satisfies many linear recurrence relations. For instance, if the sequence $\tilde{s} = s_0, s_1, \ldots$ is periodic with period T, it satisfies for every positive integer k the linear recurrence relation $s_{j+kT} = s_j$ $(j \geq 0)$. The all zero sequence $\tilde{0}$ satisfies any linear recurrence relation as defined in (3.15). Therefore, when one is considering the set of all sequences that may be produced by an LFSR it is useful to distinguish between the linear recurrence relation of least possible order that may generate a given sequence in this set and the linear recurrence relation as implemented by the LFSR. The monic polynomial

$$c(X) = X^L + c_1 X^{L-1} + \ldots + c_{L-1} X + c_L \quad \text{in } GF(q)[X] \tag{3.18}$$

associated with the linear recurrence relation (3.15) or equivalently, to the LFSR of Fig. 3.1. is called the characteristic polynomial of the LFSR or the linear recurrence relation (3.15). Note that the feedback polynomial C(D) and c(X) are reciprocal polynomials, that is, $C(D) = D^L c(D^{-1})$. When a sequence $\tilde{s}$ is generated by an LFSR, the linear recurrence relation implemented by the LFSR need not be minimal in the sense that there is no linear recurrence relation of smaller order that could generate the sequence. Therefore any linear recurring sequence $\tilde{s}$ can be associated with its minimal polynomial

$m_{\tilde{s}}(X)$ in GF(q)[X] corresponding to the characteristic polynomial of the minimum order linear recurrence relation. $m_{\tilde{s}}(X)$ is monic and unique, and has the property that it divides the characteristic polynomial of any linear recurrence relation which is satisfied by $\tilde{s}$. The most feasible way of actually computing the minimal polynomial of $\tilde{s}$ is provided by the Berlekamp-Massey LFSR synthesis algorithm (Mass 69); it yields the reciprocal polynomial of $m_{\tilde{s}}(X)$. If $\tilde{s}$ is periodic, then any phase shift $\tilde{s}^t$ has the same minimal polynomial $m_{\tilde{s}}(X)$ as $\tilde{s}$; moreover, the period T of $\tilde{s}$ may be calculated from $m_{\tilde{s}}(X)$, the relationship being $T = \mathrm{ord}(m_{\tilde{s}}(S))$. If the characteristic polynomial c(X) in GF(q)[X] of an LFSR is monic and irreducible, then c(X) is the minimal polynomial of any nonzero sequence produced by the LFSR. When the minimal polynomial of a sequence $\tilde{s}$ is primitive in GF(q)[X] of degree L, then the period of $\tilde{s}$ assumes its maximum value q^L-1 for this degree, and $\tilde{s}$ is said to be a maximum-length sequence or a PN (Pseudo-Noise)-sequence. The terminology pseudo-noise sequence stems from the fact that every nonzero L-tuple appears exactly once in the period of such a sequence and thus results in almost equal distribution for all k-tuples ($k < L$) within a period.

When the minimal polynomial of $\tilde{s}$ is irreducible in GF(q), then there is a handy description for the jth term of $\tilde{s}$ yielding the direct "time-domain solution". Suppose the characteristic polynomial c(X) in GF(q)[X] of degree L is irreducible and is identical with the minimal polynomial of $\tilde{s}$. Let α in $GF(q^L)$ be a root of c(X) (which implies that α is a quintessential element of $GF(q^L)$). Then for every coefficient A in $GF(q^L)$, the sequence $\tilde{s}^*$ defined by

$$s^*_j = A\alpha^j \qquad j = 0,1,2,... \tag{3.19}$$

is a solution of the homogeneous linear recursion (3.15) associated with the minimal polynomial c(X), as can be seen by direct substitution. However, the digits of $\tilde{s}^*$ lie in the extension field $GF(q^L)$ rather than in GF(q) as required for $\tilde{s}$. To obtain a solution with digits only from GF(q), one introduces the trace function $T_L(\beta)$ which maps any element β in $GF(q^L)$ into the groundfield GF(q) in the following way

$$T_L(\beta) = \beta + \beta^q + ... + \beta^{q^{L-1}} \quad . \tag{3.20}$$

The trace is a linear function with respect to the "scalar" field GF(q), i.e. for a_1 and a_2 in GF(q) and for β_1 and β_2 in $GF(q^L)$,

$$T_L(a_1\beta_1 + a_2\beta_2) = a_1T_L(\beta_1) + a_2T_L(\beta_2) \ . \tag{3.21}$$

The linearity of the trace function assures that the GF(q) sequence $\tilde{s}$ defined by

$$s_j = T_L(A\alpha^j) \qquad j = 0,1,2,... \tag{3.22}$$

is a solution of the linear recursion (3.15). Since α is a quintessential element of $GF(q^L)$, each choice of the coefficient A gives a different sequence $\tilde{s}$. Consequently (3.22) provides q^L solutions to the linear recursion (3.15). But there are only q^L choices for the initial contents of the LFSR with characteristic polynomial c(X). Thus there is a one-to-one correspondence between coefficients A in $GF(q^L)$ and initial contents of the LFSR, and (3.22) allows one to describe every GF(q)-solution of the linear recursion (3.15). The coefficient A may be found by solving the system of linear equations defined by (3.22) and the initial contents of the LFSR. An equivalent way for determining A is provided by residue theory. Since A is the residue of the rational form P(D)/C(D) of S(D) at $D = \alpha^{-1}$, A may be computed as follows:

$$A = \frac{-P(D)}{D\,C'(D)} \ \Big|_{\ D = \alpha^{-1}} \tag{3.23}$$

where $C'(D) = c_1 + 2c_2D + 3c_3D^2 + \ldots + L\,c_LD^{L-1}$ is the formal derivative of C(D) and where C(D) still is assumed to be irreducible over GF(q). Care must be taken when (for whatever reason) the expression (3.22) is evaluated with α in $GF(q^L)$ not being a quintessential element. Then the minimum polynomial of α in $GF(q^L)$, which is identical with the minimal polynomial of the corresponding sequence $\tilde{s}$ (as defined by (3.22)), has a degree L' which is a strict divisor of L. Consequently, there are only $q^{L'}-1$ nontrivial solutions for the recursion defined by the minimal polynomial $m_{\tilde{s}}(X)$, but q^L-1 nontrivial choices for the coefficient A in (3.22). By symmetry the A's can be partitioned into equivalence classes according to the sequence they cause.

$$\begin{aligned}
\mathcal{A}_0 &= \{A : T_L(A\alpha^j) = \tilde{0}\} \\
&\vdots \\
\mathcal{A}_i &= \{A : T_L(A\alpha^j) = \tilde{s}_i\} \\
&\vdots \\
\mathcal{A}_{q^{L'}-1} &= \{A : T_L(A\alpha^j) = \tilde{s}_{q^{L'}-1}\}
\end{aligned} \tag{3.24}$$

Each class contains exactly $q^{L-L'}$ distinct elements A from $GF(q^L)$. The "zero-class" $\mathcal{A}_0$ deserves special attention; apart from the trivial choice A = 0, there are $q^{L-L'}-1$ nonzero choices for A in $GF(q^L)$ which will cause the sequence $\tilde{s}$ to be the allzero sequence $\tilde{0}$. The impact of this observation is that the sequence $\tilde{s}$ as defined by the trace function (3.22) with A and α both being nonzero elements of $GF(q^L)$ is only guaranteed to be nontrivial if α is a quintessential element of $GF(q^L)$.

When the minimal polynomial $m_{\tilde{s}}(X)$ in GF(q)[X] of a GF(q)-sequence $\tilde{s}$ has only simple roots, i.e., when it factors into distinct monic irreducible factors, then a description equivalent to (3.22) may be found. Let P(D)/C(D) be the rational form of S(D), and let $C_i(D)$, i = 1,...,N be the N distinct irreducible factors of C(D). Apply the partial fraction expansion

$$S(D) = \frac{P(D)}{C(D)} = \sum_{i=1}^{N} \frac{P_i(D)}{C_i(D)} \tag{3.25}$$

which corresponds to decomposing $\tilde{s}$ into a sum of sequences $\tilde{s}_i$, each $\tilde{s}_i$ having irreducible minimal polynomial $m_{\tilde{s}_i}(X)$ over GF(q) of degree L_i, i.e.

$$\tilde{s} = \sum_{i=1}^{N} \tilde{s}_i \ . \tag{3.26}$$

When α_i in $GF(q^{L_i})$ denotes a root of the minimal polynomial of $\tilde{s}_i$ and A_i in $GF(q^{L_i})$ gives the initial state of $\tilde{s}_i$, then (3.22) implies

$$s_j = \sum_{i=1}^{N} T_{L_i}(A_i \alpha_i^{\,j}) \qquad j = 0,1,... \tag{3.27}$$

When a sequence $\tilde{s}$ over GF(q) can be described using the trace function, then phase shifts of $\tilde{s}$ are particularly easy to express. Let $\tilde{s}^t$ be the t-th phase shift of $\tilde{s}$ which is defined by (3.27). Then $\tilde{s}^t$ is given by

$$s_{t+j} = \sum_{i=1}^{N} T_{L_i}(A_i \alpha_i{}^t \alpha_i{}^j) \qquad j = 0,1,... \qquad (3.28)$$

This concludes the short survey on algebraic tools suited for the anahlysis of nonlinear combinations of LFSRs.

4 Random Sequences and Linear Complexity

Stream ciphers utilize deterministically generated "random" sequences to encipher the message stream. Since the running key generator is a finite state machine, the key stream necessarily is (ultimately) periodic. Thus the best one can hope for is to make the first period of a periodic key stream resemble the output of a binary symmetric source (BSS). A BSS is a device which puts out with equal probability a zero or a one independently of the previous output bits, or in other words, a BSS realizes a fair coin tossing experiment. (Note that we have tacitly assumed the sequences under investigation to be defined over GF(2)). The period of the key stream necessarily is a finite quantity. Thus we are confronted with the problem of characterizing the randomness of a finite sequence. But how can this be done in light of the fact that every finite output sequence of a BSS is equally likely? It seems difficult to define adequately the concept of randomness (in a mathematical sense) for finite sequences. Still, nearly everyone would agree that something like a "typical" output sequence of a BSS exists. A finite coin tossing sequence, for example, would "typically" exhibit a balanced distribution of single bits, pairs, triples, etc. of bits, and long runs of one symbol would be very rare. This in contrast to infinite coin tossing sequences, where local nonrandomness is sure to occur. D.E. Knuth (Knut 81) discusses various concepts of randomness for infinite sequences and gives a short description of how randomness of a finite sequence could be defined. By the above typicality-argument, one is led naturally to the criterion of distribution properties. A finite sequence of length T may be called "random" if every binary k-tuple for all k smaller than some upperbound (e.g. logT) appears about equally often. The "randomness postulates" of S. Golomb (Golo 67) based on this definition have gained widespread popularity (especially in the cryptographic community). Golomb proposed the following three requirements to measure the randomness of a periodic binary sequence. First, the disparity between zeros and ones within one period of the sequence does not exceed 1. Second, in every period, $(1/2^i)$th of the total number of runs has length i, as long as there are at least 2 runs of length i. Third, the periodic autocorrelation function is two-valued. Every sequence which satisfied these three randomness requirements was called by Golomb a pseudo-noise (PN) sequence. But although Golomb called his requirements "randomness postulates", they do not define a general measure

of randomness for finite sequences. These "randomness postulates" rather describe almost exclusively the sequences which have a primitive minimal polynomial (since they have maximum possible period, they are also called maximum-length sequences or m-sequences). But this means that the so-called PN-sequences are highly predictable, if L denotes the degree of the primitive minimal polynomial of the PN-sequence under investigation, then only 2L bits of the sequence suffice to specify completely the remainder of the period of length 2^L-1. Clearly the idea of randomness also reflects the impossibility of predicting the next digit of a sequence from all the previous ones. An interesting approach to a definition of randomness of finite sequences based on this concept of unpredictability was taken by R. Solomonov (Solo 64) and A. Kolmogorov (Kolm 65). They characterized the "patternlessness" of a finite sequence by the length of the shortest Turing machine program that could generate the sequence. Patternlessness may be equated with unpredictability or randomness. This concept was further developed by P. Martin-Loef (Mart 66). A different approach to evaluating the complexity of finite sequences was given by A. Lempel and J. Ziv (Lemp 76). Instead of an abstract model of computation such as a Turing machine one could directly use a linear feedback shift register (LFSR) model and measure the (linear) unpredictability of a sequence (finite or periodic) by the length of the shortest LFSR which is able to generate the given sequence. This approach is particularly appealing since there exists an efficient synthesis procedure (the Berlekamp-Massey LFSR synthesis algorithm (Mass 69)) for finding the shortest LFSR which generates a given sequence. This length is also referred to as the linear complexity associated to the sequence. The following sequence obtained by the author in 31 trials with a fair swiss coin may serve as an illustration for the concept of linear complexity as measure of randomness (or linear unpredictability).

$$\tilde{s} = (10001111010000110111110100010100)^{\infty} \qquad (4.1)$$

In Fig. 4.1, we compare the dynamic behaviour of the linear complexity of the periodically repeated swiss coin sequence (4.1) to that of a PN-sequence of period 31. $\Lambda(s^n)$ denotes the linear complexity of the first n digit subsequence of $\tilde{s}$.

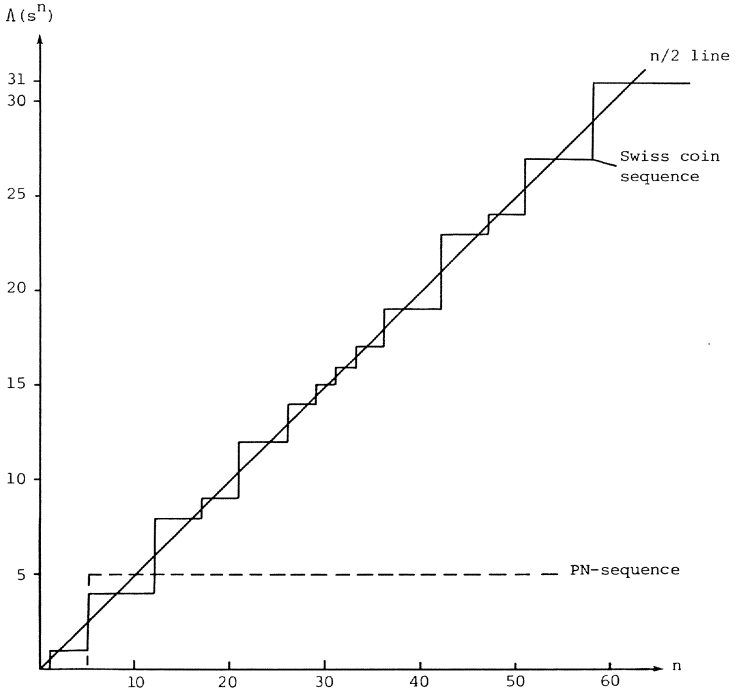


Fig. 4.1. Linear complexity profiles of the swiss coin sequence (4.1) and the PN-sequence generated by $\langle 5, 1+D^2+D^5 \rangle$ and initial state [0,0,0,0,1]

The linear complexity of the swiss coin sequence (4.1) grows approximately as n/2, where n denotes the number of processed bits, and stops at 31 which is the period of the sequence (4.1). Thus only the circulating shift register of length L = 31 is able to generate the swiss coin sequence. Conversely, the so-called PN-sequence of period 31 has a linear complexity of only 5 and is highly predictable. But note, a high linear complexity alone does not guarantee good randomness properties. As an example consider the sequence built by 30 consecutive 0's and an appended 1 which is periodically repeated. This sequence can also only be generated by the circulating shift register of length 31, but does not exhibit any randomness properties whatsoever. This could be seen in the associated linear complexity profile, in which the linear complexity remains at 0 until the 1 appears at the 31st position which causes the linear complexity to jump from 0 to 31 in one swoop. Consequently, we expect a "typical" random sequence to have associated a "typical" linear complexity profile closely following the n/2 line.

Let $s^n = s_0, s_1, \ldots, s_{n-1}$ denote a sequence of n independent and uniformly distributed binary random variables, and let $\Lambda(s^n)$ be the associated linear complexity. Our primary interest is in $N_n(L)$, the number of sequences of length n with linear complexity $\Lambda(s^n) = L$. Consider the basic recursion from $\Lambda(s^{n-1})$ to $\Lambda(s^n)$. The difference between the nth binary random variable s_{n-1} and the nth digit generated by the minimal-length LFSR which is able to generate s^{n-1} is called the next discrepancy δ_{n-1}. If the LFSR of length $\Lambda(s^{n-1})$ which generates s^{n-1} also generates s^n, then $\delta_{n-1} = 0$ and the linear complexity does not change. Conversely, if the LFSR of length $\Lambda(s^{n-1})$ which generates s^{n-1} fails to generate s^n, then $\delta_{n-1} = 1$ and the linear complexity increases when $\Lambda(s^{n-1})$ is smaller than n/2. The recursion describing the length change is basic to the LFSR-synthesis procedure (Mass 69):

$$\delta_{n-1} = 0 \qquad \Lambda(s^n) = \Lambda(s^{n-1}) \tag{4.2a}$$

$$\delta_{n-1} = 1 \quad \begin{cases} \Lambda(s^n) = \Lambda(s^{n-1}) & \text{if } \Lambda(s^{n-1}) \geq \frac{n}{2} \\ \Lambda(s^n) = n - \Lambda(s^{n-1}) & \text{if } \Lambda(s^{n-1}) < \frac{n}{2} . \end{cases} \tag{4.2b}$$

Note that the linear complexity does not change (regardless of the value of the discrepancy) when $\Lambda(s^{n-1}) \geq \frac{n}{2}$. It is illuminating to represent graphically the linear complexity recursion (4.2) (see Fig. 4.2).

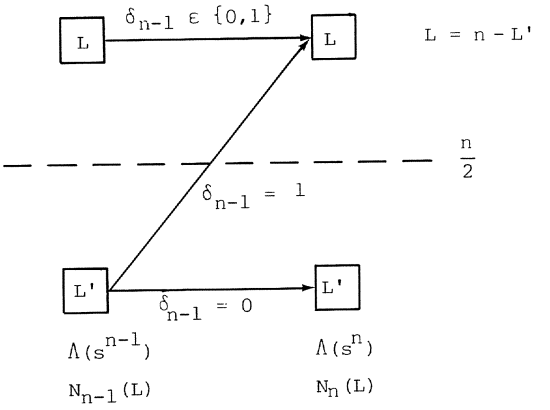


Fig. 4.2. Graphical illustration of the linear complexity growth process

From the diagram in Fig. 4.2, we may now directly read off the recursion for $N_n(L)$. If $\Lambda(s^{n-1}) = L' < \frac{n}{2}$, then $N_n(L') = N_{n-1}(L')$ since only one choice for s_{n-1} causes $\delta_{n-1} = 0$. The second choice for s_{n-1} causes $\delta_{n-1} = 1$ and thus transfers $N_{n-1}(L')$ sequences to the new complexity $L = n-L'$. If $\Lambda(s^{n-1}) = L > \frac{n}{2}$, then $\Lambda(s^n) = L$ (irrespective of δ_{n-1}) and $2N_{n-1}(L)$ sequences contribute to $N_n(L)$. The only exception to the sketched process in Fig. 4.2 occurs when n is even and $L = \frac{n}{2}$. In this case no path from $\Lambda(s_{n-1}) = L' < \frac{n}{2}$ may lead to $\Lambda(s^n) = L = \frac{n}{2}$, since $L = \frac{n}{2} = n-L'$ would require $L' = \frac{n}{2}$ which contradicts the assumption. We can now write the recursion for $N_n(L)$, the number of sequences of length n with linear complexity L, as

$$N_n(L) = \begin{cases} 2N_{n-1}(L) + N_{n-1}(n-L) & n \geqslant L > \frac{n}{2} \quad (4.3a) \\ 2N_{n-1}(L) & L = \frac{n}{2} \quad (4.3b) \\ N_{n-1}(L) & \frac{n}{2} > L \geqslant 0 \; . \quad (4.3c) \end{cases}$$

The initial conditions for the recursion (4.3) are $N_1(0) = N_1(1) = 1$. At any length n the total number of sequences is 2^n. In table 4.1, the values of $N_n(L)$ are listed for all positive $n \leqslant 10$.

L \ n	1	2	3	4	5	6	7	8	9	10
0	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	2	2	2	2	2
2		1	4	8	8	8	8	8	8	8
3			1	4	16	32	32	32	32	32
4				1	4	16	64	128	128	128
5					1	4	16	64	256	512
6						1	4	16	64	256
7							1	4	16	64
8								1	4	16
9									1	4
10										1

Table 4.1. Values of $N_n(L)$ for n = 1, ..., 10

The general form of $N_n(L)$ is easily guessed from table 4.1.

$$N_n(L) = \begin{cases} 2^{\min\{2n-2L,\,2L-1\}} & n \geq L > 0 \qquad (4.4a) \\ 1 & n > L = 0 \qquad (4.4b) \end{cases}$$

To show that this solution is correct, we first prove that the solution satisfies the recursion (4.3) for all $n > 1$.

Suppose $n \geq L > n/2$, then $N_n(L) = 2^{2n-2L}$, $N_{n-1}(L) = 2^{2n-2-2L}$ and $N_{n-1}(n-L) = 2^{2n-2L-1}$, since $n \leq 2L$ implies $2(n-L) < n-1$. These values satisfy recursion (4.3a) for all $n > 1$, as can be seen by substitution.

Suppose $L = n/2$, then $N_n(L) = 2^{2L-1}$ and $N_{n-1}(L) = 2^{2L-2}$, which satisfy recursion (4.3b) for all even $n > 1$.

Suppose $n/2 > L > 0$, then $N_n(L) = N_{n-1}(L) = 2^{2L-1}$ and the recursion (4.3c) is trivially satisfied for all $n > 1$.

By taking into account the initial conditions $N_1(0) = N_1(1) = 1$ the solution (4.4) is seen to yield the correct values for $n = 2$. Thus (4.4) is the solution to the recursion (4.3). We summarize the result in the following proposition.

Proposition 4.1. Distribution of $N_n(L)$

The number $N_n(L)$ of binary sequences $s^n = s_0, s_1, \ldots, s_{n-1}$ of length n having linear complexity exactly L is

$$N_n(L) = \begin{cases} 2^{\min\{2n-2L,\,2L-1\}} & n \geq L > 0 \\ 1 & n > L = 0 \,. \end{cases}$$

The form of $N_n(L)$ for the general case of q-ary sequences may be found in (Gust 76) where the objective of that author was to evaluate the performance of the Berlekamp-Massey LFSR synthesis algorithm. Our interest is in characterizing a "typical" random sequence by means of the associated linear complexity. Proposition 4.1 tells us that the vast majority of the possible binary sequences of length n will have linear complexity close to n/2. A quantity of independent interest, related to $N_n(L)$, is the number of semi-infinite sequences of linear complexity L or less, which we denote by N_L. For finite $L > 0$, Proposition 4.1 gives $N_\infty(L) = 2^{2L-1}$. Thus

$$N_L = 1 + \sum_{j=1}^{L} 2^{2j-1} \tag{4.5}$$

where the added 1 accounts for the allzero sequence, which has linear complexity L = 0. Evaluating the finite geometric series (4.5) yields

$$N_L = \frac{2}{3}\, 2^{2L} + \frac{1}{3} \ . \tag{4.6}$$

When we consider the tree corresponding to the set of all binary semi-infinite sequences, then at depth 2L every sequence of linear complexity L or less is characterized by the fact that the associated LFSR which may produce the sequence is unique. Hence the significance of (4.6) is that almost exactly 2/3 of all sequences of length 2L may be generated with an LFSR of length L or less. Both proposition 4.1 and the above argument on N_L suggest that any sequence of n randomly selected binary digits will "typically" have a linear complexity close to n/2. To obtain a precise characterization, we may compute the expected linear complexity of a sequence s^n of n independent binary random variables $s_0, s_1, \ldots, s_{n-1}$ (as emitted from a BSS).

$$E[\Lambda(s^n)] = \sum_{b^n} \Lambda(b^n)\, P(b^n) \tag{4.7}$$

where b^n denotes a particular realization of the coin tossing sequence s^n. Since each b^n is equally likely, the probability $P(s^n = b^n)$ is 2^{-n}. Therefore

$$E[\Lambda(s^n)] = 2^{-n} \sum_{b^n} \Lambda(b^n) = 2^{-n} L^*(n) \tag{4.8}$$

where we have introduced the symbol $L^*(n)$ for $2^n\, E[\Lambda(s^n)]$. The set of all b^n may be subdivided into equivalence classes according to the associated linear complexity. Thus we may rewrite the sum $L^*(n)$ in (4.8) as

$$L^*(n) = \sum_{L=1}^{n} \sum_{\{b^n : \Lambda(b^n) = L\}} L \ . \tag{4.9}$$

The Lth equivalent class is easily identified to contain $N_n(L)$ elements. Thus

$$L^*(n) = \sum_{L=1}^{n} L\, N_n(L) \ . \tag{4.10}$$

Replacing $N_n(L)$ by the solution given in proposition (4.1), we obtain

$$L^*(n) = \sum_{L=1}^{n} L 2^{\min\{2n-2L,\,2L-1\}} \tag{4.11}$$

which may be subdivided into two sums according to the dominance of 2n-2L or 2L-1, which results in

$$L^*(n) = \sum_{L=1}^{\lfloor n/2 \rfloor} L 2^{2L-1} \sum_{L=\left\lceil \frac{n+1}{2} \right\rceil}^{n} L 2^{2n-2L} \ . \tag{4.12}$$

It is now possible to obtain a closed form expression for the finite sum in (4.12) by applying standard analytical methods. We illustrate the principle by evaluating

$$\sum_{j=1}^{m} j 2^{2j-1} \ . \tag{4.13}$$

First, we introduce a dummy variable I raised to the (j-1)st power,

$$\sum_{j=1}^{m} j\, I^{j-1}\, 2^{2j-1} \ .$$

Now we integrate the sum with respect to I,

$$\sum_{j=1}^{m} I^{j}\, 2^{2j-1} \ .$$

This is an ordinary geometric series whose sum is given by

$$2I\,\frac{I^m 2^{2m}-1}{I\,2^2-1} \quad .$$

Differentiating this sum and setting I = 1, we obtain as the closed form solution for (4.13)

$$\sum_{j=1}^{m} j2^{2j-1} = \frac{(m+1)}{3}\,2^{2m+1} - \frac{2}{9}\,(2^{2m+1}-1) \quad . \tag{4.14}$$

Because of the floor- and ceiling-functions in (4.12), it is convenient to distinguish between even and odd n. Let $L^*_e(n)$ and $L^*_0(n)$ denote the function L*(n) evaluated at even n and at odd n, respectively. Then by applying the standard techniques, as explained in the derivation of (4.14), to the individual sums in (4.12), we obtain for even n

$$L^*_e(n) = \{2^n(\tfrac{n}{3} - \tfrac{2}{9} + \tfrac{2}{9}\,2^{-n})\} + \{2^n(\tfrac{n}{6} + \tfrac{4}{9} - 2^{-n}(\tfrac{n}{3} + \tfrac{4}{9}))\} \tag{4.15}$$

where the brackets {} enclose the values of the two distinct sums in (4.20).

In the case of odd n, we similarly obtain

$$L^*_0(n) = \{2^n(\tfrac{n}{6} - \tfrac{5}{18} + \tfrac{2}{9}\,2^{-n})\} + \{2^n(\tfrac{n}{3} + \tfrac{5}{9} - 2^{-n}(\tfrac{n}{3} + \tfrac{4}{9}))\}. \tag{4.16}$$

Now it is straightforward to combine (4.8), (4.15) and (4.16) to obtain the desired expected linear complexity $E[\Lambda(s^n)]$. We summarize the result in the following proposition.

Proposition 4.2. $E[\Lambda(s^n)]$

The expected linear complexity of a sequence $s^n = s_0, s_1, \ldots, s_{n-1}$ of n independent and uniformly distributed binary random variables is given by

$$E[\Lambda(s^n)] = \frac{n}{2} + \frac{4+R_2(n)}{18} - 2^{-n}\left(\frac{n}{3} + \frac{2}{9}\right) \qquad (4.17)$$

where $R_2(n)$ denotes the remainder when n is divided by 2.

Proposition 4.2. confirms our suspicion that the linear complexity of a randomly selected sequence s^n can be expected close to n/2. Nevertheless, it is surprising how very close to half the sequence length that the expected linear complexity actually lies. For large values of n,

$$E[\Lambda(s^n)] \cong \frac{n}{2} + \frac{4+R_2(n)}{18} \qquad n >> 1 \qquad (4.18)$$

which differs from n/2 by only an offset of 2/9 in the case of even n or 5/18 in the case of odd n. Besides the expectation, the variance of the linear complexity is a second key parameter suited for characterizing "typical" random sequences. The variance is defined as

$$\begin{aligned} Var[\Lambda(s^n)] &= E[\{\Lambda(s^n) - E[\Lambda(s^n)]\}^2] \\ &= E[\Lambda^2(s^n)] - E[\Lambda(s^n)]^2 \quad . \end{aligned} \qquad (4.19)$$

Following the same approach as for the derivation of $E[\Lambda(s^n)]$, the second moment $E[\Lambda^2(s^n)]$ is found to be (compare 4.12)

$$L^{2*}(n) = E[\Lambda^2(s^n)]2^n = \sum_{L=1}^{\lfloor n/2 \rfloor} L^2 2^{2L-1} + \sum_{L=\left\lceil \frac{n+1}{2} \right\rceil} L^2 2^{2n-2L} \qquad (4.20)$$

We apply again the standard technique of integration and differentation of the finite sums in (4.20) to obtain a closed form expression for $L^{2*}(n)$.

For analytical convenience, let $L_e^{2*}(n)$ and $L_o^{2*}(n)$ denote the function $L^{2*}(n)$ evaluated at even and odd n, respectively. We indicate the two distinct sums in (4.20) by enclosing them with brackets {}. In the case of even n, we obtain

$$L_e^{2*}(n) = \{2^{n+1} \left(\frac{1}{12} n^2 - \frac{1}{9} n + \frac{5}{27}\right) - \frac{10}{27}\}$$
$$+ \{2^n \left(\frac{1}{12} n^2 + \frac{4}{9} n + \frac{20}{27}\right) - \left(\frac{1}{3} n^2 + \frac{8}{9} n + \frac{20}{27}\right)\} \qquad (4.21)$$

In the case of odd n, we obtain

$$L_o^{2*}(n) = \{2^n \left(\frac{1}{12} n^2 - \frac{5}{18} n + \frac{41}{108}\right) - \frac{10}{27}\}$$
$$+ \left(2^n \left(\frac{1}{6} n^2 + \frac{5}{9} n + \frac{41}{54}\right) - \left(\frac{1}{3} n^2 + \frac{8}{9} n + \frac{20}{27}\right\} . \qquad (4.22)$$

Now it is straightforward to combine (4.20), (4.21), and (4.22) to obtain the desired closed form expression for the second moment of the linear complexity for all positive n:

$$E[\Lambda^2(s^n)] = \frac{1}{4} n^2 + \frac{4+R_2(n)}{18} + \frac{40+R_2(n)}{36}$$
$$- 2^{-n} \left(\frac{1}{3} n^2 + \frac{8}{9} n + \frac{10}{9}\right) \qquad (4.23)$$

where $R_2(n)$ denotes the remainder when n is divided by 2. Finally, the first moment of the linear complexity (as shown in proposition 4.2) together with the second moment as displayed in (4.23), allow the calculation of $Var[\Lambda(s^n)]$, via (4.19). We summarize the result in the following proposition.

Proposition 4.3. $\mathrm{Var}[\Lambda(s^n)]$

The variance of the linear complexity of a sequence $s^n = s_0, s_1, \ldots, s_{n-1}$ of n independent and uniformly distributed binary random variables is given by

$$\mathrm{Var}[\Lambda(s^n)] = \frac{86}{81} - 2^{-n}\left(\frac{14-R_2(n)}{27}\, n + \frac{82-2R_2(n)}{81}\right) - 2^{-2n}\left(\frac{1}{9}\, n^2 + \frac{4}{27}\, n + \frac{4}{81}\right) \qquad (4.24)$$

where $R_2(n)$ denotes the remainder when n is divided by 2. Moreover,

$$\lim_{n\to\infty} \mathrm{Var}[\Lambda(s^n)] = \frac{86}{81}\ . \qquad (4.25)$$

The variance is a measure of spread. If the variance is small then large deviations of the random variable under consideration from its mean are improbable. One might have expected that the spread of the linear complexity grows with increasing length n of the investigated sequence. Note that $\Lambda(s^n)$ may assume more and more values with increasing n. The interesting implication of proposition 4.3 is that the spread of the linear complexity $\Lambda(s^n)$ is virtually independent of the sequence length n. Regardless of how many sequence bits are processed, the fraction of sequences centered around the mean is virtually constant. We may make these intuitive statements more precise by invoking Chebyshev's inequality (Fell 68), which implies that, for any $k > 0$, the probability that the linear complexity of a random sequence s^n differs by an amount larger or equal than k from its mean is bounded from above by the variance of the linear complexity divided by k^2. Thus, for all n,

$$P\{\,|\,\Lambda(s^n) - E[\Lambda(s^n)\,|\,\geqslant k\} \leqslant \frac{\mathrm{Var}[\Lambda(s^n)]}{k^2} \qquad (4.26)$$

Suppose k = 10, then, for sufficiently large n, Chebychev's inequality provides a bound of $(86/81)10^{-2} = 0.0106$. Consequently, at least 99 % of all random sequences s^n have a linear complexity within the range $(n/2) \pm 10$. This is a surprisingly sharp characterization of random sequences by means of their associated linear comple-

xity. Moreover, Chebychev's inequality is known to yield fairly loose bounds in individual applications because of its universality, so we may expect an even closer scattering of the linear complexities around the mean.

A different approach which could help to characterize random sequences is to consider the growth process of the linear complexity as a special kind of random walk. In this interpretation $\Lambda(s^n)$ gives the "position" of the "particle" at time n. We may define the n/2-line as the "origin" of the "particle", since at any time the expected location of the "particle" is about n/2 (compare proposition 4.2). Typically the "particle" would depart from the n/2-line to some position below the n/2-line, then jump above the n/2-line and walk back to the n/2 line. Fig. 4.3 illustrates such a typical section of the linear complexity profile of a binary sequence.

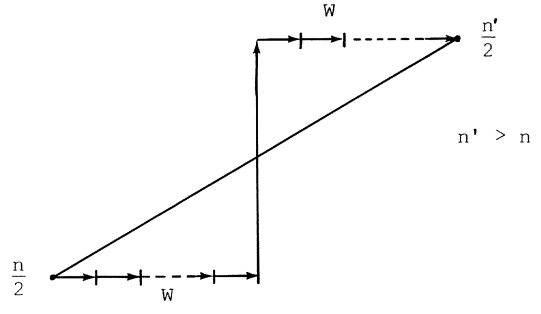


Fig. 4.3. A typical random walk segment of $\Lambda(s^n)$

Compare also the linear complexity profile of the swiss coin sequence (4.1) depicted in Fig. 4.1. The recursion (4.2) describing the growth of linear complexity forces $\Lambda(s^n)$ to retain its value, whenever that value is greater than n/2, until $\Lambda(s^{n'}) = n'/2$. From this point on, a change in linear complexity could occur at every step. In case of such a change, the jump of $\Lambda(s^n)$ is symmetrical with respect to the n/2-line, i.e. the "particle" $\Lambda(s^n)$ jumps from L to (n+1)-L. Without loss of essential generality, assume that $\Lambda(s^n) = n/2$. (Note that every nonzero sequence crosses at least once the n/2-line). Then the next jump will occur at time n+k, that is, after k time units, if

$$\delta_n = \delta_{n+1} = \ldots = \delta_{n+k-2} = 0 \; ; \; \delta_{n+k-1} = 1 \qquad (4.27)$$

causing the new linear complexity to be

$$\Lambda(s^{n+k}) = (n+k) - \Lambda(s^n) \tag{4.28}$$

By the fact that the s_i are independent and fair coin tosses, the probability that the event (4.27) occurs is 2^{-k}. Let W be the random variable denoting the number of time units until the next length change occurs, given that at time n $\Lambda(s^n) = n/2$. The above observations then imply

$$E[W] = \sum_{k=1}^{\infty} k\, 2^{-k} = \sum_{k=0}^{\infty} 2^{-k} = 2 \ . \tag{4.29}$$

Thus, for the "particle" $\Lambda(s^n)$, the average return time to the origin (the n/2-line) will be 2E[W] = 4; and the average jump height will be $E[\Delta L] = E[W]$, since $\Delta L = (n + W - (n/2)) - (n/2) = W$. The results obtained from the random walk interpretation of the linear complexity profile are summarized in the following proposition, where we have also generalized to an arbitrary starting point $\Lambda(s^n) = L$ to cover all possible sequences.

Proposition 4.4. Random walk setup

If $\tilde{s} = s_0, s_1, \ldots$ denotes a sequence of independent and uniformly distributed binary random variables and if $\Lambda(s^n) = L$, then the average number of sequence bits that have to be processed until the next length change occurs is given by

$$E[W \mid \Lambda(s^n) = L] = \begin{cases} 2 & \text{if } L \leq \frac{n}{2} \\ 2+2L-n & \text{if } L > \frac{n}{2} \end{cases} \tag{4.30}$$

Moreover, the average length change is

$$E[\Delta L \mid \Lambda(s^n) = L] = \begin{cases} 2 & \text{if } L \geq \frac{n}{2} \\ n-2L+2 & \text{if } L < \frac{n}{2} \end{cases} \tag{4.31}$$

The import of proposition 4.4. is that it provides information about the details of the linear complexity profile of random sequences.

Proposition 4.4 tells us that the linear complexity profile of a random sequence will look like an irregular staircase with an average step length of 4 time units and an average step height of 2 linear complexity units. A good illustration of this "typical" growth process is given by the linear complexity profile of the swiss coin sequence depicted in Fig. 4.1.

The various characterizations of binary random sequences by means of the associated linear complexity (as described in proposition 4.1 - 4.4) might now suggest that we have only to put a "channel" of sufficient size around the n/2-line to separate the random looking sequences from the nonrandom looking sequences. But obviously enough, the probability that a random sequence $\Lambda(s^n)$ will leave this fictitious channel at least once goes to 1 as n goes to infinity. It is not even true that the sequences whose linear complexity profile stays very close to the n/2 line will always exhibit good statistical properties. An interesting example is provided by the sequence y whose terms are defined as

$$y_n = \begin{cases} 1 & \text{if } n = 2^j - 1 \qquad j = 0,1,2... \\ 0 & \text{otherwise} \end{cases} \tag{4.32}$$

The sequence $\tilde{y}$ is highly "nonrandom" (the fraction of 1's in y^n is about $\log_2 n \,/n$), yet it has a linear complexity profile following the n/2-line as closely as is possible (see Fig. 4.4).

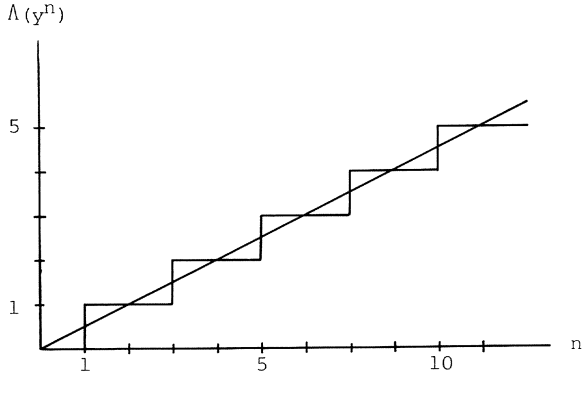


Fig. 4.4. The perfect staircase profile associated to the sequence (4.32)

The proof that $\Lambda(y^n) = \lfloor (n+1)/2 \rfloor$, $n \geq 1$ was first accomplished by Z. Dai (Dai 85) utilizing the Euclidean algorithm. We present here an adapted version (Mass 85) which bases on the Berlekamp-Massey LFSR-synthesis algorithm. Suppose the LSFR-synthesis algorithm responds with the sequence of discrepancies δ = 1, 0, 1, 0, 1, 0, ... when fed with $\tilde{y}$. Then clearly $\Lambda(y^n) = \lfloor (n+1)/2 \rfloor$. Since $\delta_n = 0$, for odd n, the connection polynomial associated to y^n will not be modified at odd n.

Let $C_k(D)$ denote the connection polynomial found by the LFSR-synthesis algorithm at time n = 2k. Then the action of the synthesis algorithm can be summarized in the following recursion

$$C_k(D) = C_{k-1}(D) + D^2 C_{k-2}(D) \tag{4.33}$$

If this recursion is now treated as a fixed coefficient linear recursion over the field F(D) of rational functions over GF(2), it may be solved by standard analytical techniques. The characteristic polynomial of (4.33) is $X^2 + X + D^2$. Let β and β' be roots of this polynomial; then $\beta' + \beta = 1$ or, equivalently, $\beta' = 1 + \beta$. Then the general solution may be written as

$$C_k(D) = A_1 \beta^k + A_2 (1+\beta)^k \tag{4.34}$$

where A_1, A_2 are elements (as well as is β) of the algebraic extension of F(D) defined by the characteristic polynomial $X_2 + X + D^2$. Taking into account the initial conditions $C_0(D) = 1$, $C_1(D) = 1 + D$ we arrive at

$$C_k(D) = (D+\beta)\beta^k + (1+D+\beta)\,(1+\beta)^k \tag{4.35}$$

Consider the solution at powers of 2, $k = 2^i$,

$$C_{2^i}(D) = 1 + D + \beta + \beta^{2^i}$$

where $\beta + \beta^{2^i}$ must be some polynomial in D. Noting that

$$\beta^{2^i} = (\beta + D^2)^{2^{i-1}} = \beta^{2^{i-1}} + D^{2^i}$$

a simple induction leads to

$$\beta^{2^i} = \beta + \sum_{j=1}^{i} D^{2^j}$$

Hence

$$C_{2^i}(D) = 1 + \sum_{j=0}^{i} D^{2^j} \qquad (4.36)$$

Let $P_k(D)$ denote the polynomial corresponding to the n = 2k initial digits of $\tilde{y}$, that is $P_k(D)/C_k(D)$ coincides with Y(D) for the first 2k digits. By the LFSR-synthesis, $P_k(D)$ must satisfy exactly the same recursion as $C_k(D)$,

$$P_k(D) = P_{k-1}(D) + D^2 P_{k-2}(D) \qquad (4.37)$$

but with different initial conditions: $P_0(D) = 0$, $P_1(D) = 1$. Hence the general solution for $P_k(D)$ follows as

$$P_k(D) = \beta^k + (1+\beta)^k \qquad (4.38)$$

Evaluated at powers of 2, $k=2^i$, we obtain $P_{2^i}(D) = 1$. Now if $\tilde{\delta}$ = 101010.. is in one-to-one correspondence with $\tilde{y}=110^1 10^3 10^7 1...$ then the first 2^{i+1} terms of Y(D) must coincide with the first 2^{i+1} terms of $P_{2^i}(D)/C_{2^i}(D)$, for any $i \geq 0$. Hence it must hold

$$\frac{P_{2^i}(D)}{C_{2^i}(D)} = \frac{1}{1+\sum_{j=0}^{i} D^{2^j}} = \sum_{j=0}^{i+1} D^{2^j-1} + D^{2^{i+1}} (1 + ...)$$

which indeed is true, as can be seen by multiplying both sides with $C_{2^i}(D)$. This completes the proof of proposition 4.5.

Proposition 4.5.

The sequence $y = 110^{1}10^{3}10^{7}1\ldots$ (or as defined in (4.32)) has associated the discrepancy sequence $\delta = 101010\ldots$, and therefore the linear complexity of any subsequence y^n, $n \geq 0$, is given by

$$\Lambda(y^n) = \left\lfloor \frac{n+1}{2} \right\rfloor \tag{4.39}$$

Note that $\tilde{y} = 110^{1}10^{3}10^{7}1\ldots$ also satisfies the relation $y_i^2 = y_{2i+1}$, all $i \geq 0$; any sequence with this property is called a delayed-decimation/square (DDS-)sequence (Mass 85). Another example of a DDS-sequence is the syndrome sequence of a binary primitive BCH code.

The example depicted in fig. 4.4 suggests that too regular linear complexity profiles are incompatible with the randomness properties of the associated sequences. But note that the sequence $\tilde{y}$ as defined in (4.32) is not the only sequence with this perfect staircase profile. Whenever $\Lambda(s^n) > \frac{n}{2}$ then, independent of the choice for y_n, $\Lambda(y^{n+1})$ will be equal to $\Lambda(y^n)$. This indicates that there exist in fact many sequences which have associated the perfect staircase profil shown in Fig. 4.4. And undoubtedly, there will be some among them with good statistical properties. But remember that the perfect staircase profil would indeed pass randomness tests based on the expectation of linear complexity (proposition 4.2 and 4.3), but it never would pass a randomness test based on the random walk setup (proposition 4.4). Hence with the knowledge acquired so far on the linear complexity profile of random sequences, we would not accept as "random" a sequence with such a regular profile as that shown in Fig. 4.4.

From the practical standpoint in good stream cipher design, one important question remains to be answered. A deterministically generated key stream must necessarily be (ultimately) periodic. Thus, the question of what the linear complexity profile of a periodically repeated random bit string will look like is of considerable practical interest. Let $z^T = z_0, z_1, \ldots, z_{T-1}$ denote the first period of the semi-infinite sequence $\tilde{z}$, and assume z^T to be selected according to a fair coin tossing experiment. Then from the preceding analysis we may immediately deduce that $E[\Lambda(\tilde{z})]$ is at least $T/2$, since that

result holds for the finite random sequence z^T. On the other hand z^T could be put into a pure cycling shift register of length T to produce $\tilde{z}$. Thus $\tilde{z}$ certainly satisfies the recursion $z_{T+j} = z_j$, which implies that $E[\Lambda(\tilde{z})]$ is at most T. But how likely is it that $\tilde{z}$ satisfies a linear recursion of order lower than T? And how would the linear complexity profile change from that point on where the first bits of z^T are repeated? Intuitively, one would expect the linear complexity to grow to close to the period length T, since the recursion which produces the second half of z^T from the first half is unlikely to have any similarities to the recursion that produces the first half of z^T from the second half (which is required by the periodic repetition). Now let Z*(D) denote the polynomial associated with the first period z^T of $\tilde{z}$. Then

$$Z(D) = \frac{Z^*(D)}{1+D^T} \tag{4.40}$$

Z*(D) may be interpreted as the polynomial associated with the initial state of a circulating shift register. The question of the expected linear complexity of $\tilde{z}$ now corresponds to asking for the expected degree m of the denominator polynomial in (4.40) after reduction by $\gcd(Z^*(D), 1+D^T)$. To every choice of Z*(D), there is a unique partial fraction expansion

$$Z(D) = \sum_{i=1}^{n} \sum_{k=1}^{m_i} \frac{P_{ik}(D)}{[C_i(D)]^k} \tag{4.41}$$

where $C_i(D)$, i=1,..,n, are the irreducible factors of $1 + D^T$ and m_i, $i = 1, \ldots, n$ are their multiplicities, and where $\deg(P_{ik}(D)) < \deg(C_i(D))$. Suppose now that the binary coefficients of the numerator polynomials $P_{ik}(D)$ are chosen independently from a uniform distribution. This induces a uniform probability distribution over the set of possible initial periods z^T, (or equivalently, over the set of possible Z*(D)), since there exists a unique correspondence between initial periods Z*(D) and the choice of numerator polynomials in the partial fraction expansion (4.34). But a uniform probability measure over all z^T implies that each digit z_j, j=0,...,T-1, is an independent and uniformly distributed binary random variable. We conclude that the expected linear complexity of z may equivalently be computed as the expected degree of the minimal polynomial of z

given that all coefficients of the numerator polynomials $P_{ik}(D)$ are chosen independently from a uniform distribution. Unfortunately, there appears to be no simple solution to this problem since the irreducible factors $C_i(D)$ of $1+D^T$, as well as their multiplicities strongly depend on the value of T. We will demonstrate the solution for 2 extreme cases thereby obtaining results of some significance for the general case. Suppose first that T is equal to 2^n-1 with n a prime. Then the partial fraction expansion (4.34) takes on the special form

$$Z(D) = \frac{Z^*(D)}{1+D^{2^n-1}} = \frac{A}{1+D} + \sum_{i=1}^{M} \frac{P_i(D)}{C_i(D)} \tag{4.42}$$

where each $C_i(D)$ has prime degree n, and thus the number of such factors is $M = (2^n-2)/n$. When we randomly select A and the coefficients of $P_i(D)$, i=1,...,M, then the probability that A and $P_i(D)$ are zero is 2^{-1} and 2^{-n}, respectively. Therefore

$$P_k = P(\Lambda(\tilde{z}) = 2^n - 1 - kn) = P(\Lambda(\tilde{z}) = 2^n - 2 - kn)$$

$$= \frac{1}{2} \binom{M}{k} (1 - 2^{-n})^{M-k} (2^{-n})^k \ . \tag{4.43}$$

With the same approximations as used in proposition 5.8, we obtain for large prime n and small k

$$P_k \simeq \frac{1}{2k!n^k} e^{-\frac{1}{n}} \tag{4.44}$$

By considering the two choices of 2^n-1 and 2^n-2 for the linear complexity we may provide a rough lowerbound on the expected linear complexity of $\tilde{z}$,

$$E[\Lambda(\tilde{z})] \geq (2^n-1)P_o + (2^n-2)P_o \geq \simeq e^{-\frac{1}{n}} (2^n-\tfrac{3}{2}) \tag{4.45}$$

The significance of the bound (4.45) lies in the fact that, as n increases, it approaches the period T, thereby showing that the linear complexity of z can be expected to be very close to the period length for all prime n. A much better estimate of the actual $E[\Lambda(\tilde{z})]$ may be obtained when more than just the two largest choices for $\Lambda(\tilde{z})$, with their corresponding probabilities P_k as computed in

(4.44) are taken into account. When T is chosen odd, then the minimal polynomial of $\tilde{z}$ does not contain any repeated factors (which is equivalent to saying that the minimal polynomial of $\tilde{z}$ has only simple roots). The other extreme may be found when the period length T is chosen to be a power of 2, i.e. $T = 2^n$. Then there exists only one root, namely 1, which occurs with multiplicity 2^n, and

$$Z(D) = \frac{Z^*(D)}{1+D^{2^n}} = \frac{Z^*(D)}{(1+D)^{2^n}} \quad . \tag{4.46}$$

Then the partial fraction expansion (4.41) takes on the special form

$$Z(D) = \sum_{i=1}^{2^n} \frac{A_i}{(1+D)^i} \tag{4.47}$$

When all the binary coefficients A_i are drawn independently from a uniform distribution, then half the sequences $\tilde{z}$ will have linear complexity 2^n, one forth of the $\tilde{z}$ will have linear complexity 2^n-1, one eighth will have $\Lambda(\tilde{z}) = 2^n-2$, and so on. Thus the probability distribution induced on $\Lambda(\tilde{z})$ is given by

$$P(\Lambda(\tilde{z}) = L) = 2^{L-2^n-1} \qquad L = 1,\ldots,2^n \tag{4.48}$$

With the help of this probability distribution, it is now easy to compute the expected linear complexity

$$E[\Lambda(\tilde{z})] = \sum_{L=1}^{2^n} L \cdot 2^{L-2^n-1} = 2^{-2^n-1} \sum_{L=1}^{2^n} L2^L \quad . \tag{4.49}$$

Invoking the integration/differentiation technique for sums (as demonstrated in the derivation of (4.14)) results in

$$E[\Lambda(\tilde{z})] = 2^n - 1 + 2^{-2^n} \quad .$$

This result is summarized in the following proposition.

Proposition 4.6. Periodic repetition of random sequence

If the semi-infinite sequence $\tilde{z}$ is generated by periodically repeating a sequence $z^T = z_0,\ldots,z_{T-1}$ of T independent and uniformly distributed binary random variables, i.e. $\tilde{z} = z^T, z^T, \ldots$, and if $T = 2^n$, then the expected linear complexity of $\tilde{z}$ is

$$E[\Lambda(\tilde{z})] = 2^n - 1 + 2^{-2^n} \tag{4.50}$$

The two investigated cases of periodically repeating a finite sequence of random bits are extreme in the sense that, for a period $T = 2^n-1$, the minimal polynomial of $\tilde{z}$ is sure to contain only simple roots whose number then equals the linear complexity of $\tilde{z}$, and, for a period $T = 2^n$, the minimal polynomial of $\tilde{z}$ is sure to contain only one root whose multiplicity then equals the linear complexity of $\tilde{z}$. For both choices of the period we were able to show that the expected linear complexity is almost equal to the period length.

Recapitulating, we may say that the linear complexity of a sequence provides a good measure of its unpredictability, especially when the growth process of the linear complexity with respect to the number of considered sequence bits (which was termed the linear complexity profile) is taken into account. For true random sequences of length n, the expected linear complexity was shown to be about n/2. Moreover, the vast majority of these sequences were shown to have associated a linear complexity very close to n/2. The dynamic characterization of random sequences by means of linear complexity results in an average linear complexity increase of 2 after an average number of 4 considered sequence digits. When a random sequence of length $T = 2^n$ ($n \geq 0$) or $T = 2^n-1$ (n prime) T is periodically repeated, then the expected linear complexity is close to the period length T and the associated linear complexity profile is not distinguishable from the linear complexity profile of a true random sequence up to T digits. Heuristic arguments suggest that the expected linear complexity will in general be close to the period length T and that in fact the associated linear complexity profile will not be distinguishable from the linear complexity profile of a true ran-

dom sequence even up to 2T digits. (Compare also the swiss coin sequence example displayed in Fig. 4.1). We conclude that a good random sequence generator should have linear complexity close to the period length, and also a linear complexity profile which follows closely, but "irregularly", the n/2-line (where n denotes the number of sequence digits) thereby exhibiting average step lengths and step heights of 4 and 2, respectively.

5 Nonlinear Theory of Periodic Sequences

Running key generators employ nonlinear transformations in order to achieve high unpredictability of the generated key stream.

A useful measure of unpredictability, or equivalently, randomness of a sequence is provided by the associated linear complexity (see chapter 4). Thus there is a need for analyzing nonlinear combinations of (periodic) sequences in terms of linear complexity as well as in terms of period, statistics, leakage etc. (But analyzability is not always appreciated as basic requirement. We even heard designers of very complex key stream generators say that their cryptodevice must be secure because it is too complex ever to be analyzed). In the sequel we will develop some theoretical tools which allow to calculate or bound the linear complexity of nonlinear combinations of periodic sequences in cryptographically interesting cases.

The simplest possible nonlinear transformation is the product of two binary digits, which is also called the and-function in switching theory. To see this, let x_1, x_2 be two binary variables and let f denote their product, i.e.

$$f(x_1,x_2) = x_1x_2 \tag{5.1}$$

where we use juxtaposition to denote multiplication in GF(2). Then for binary variables x_1', x_2'

$$\begin{aligned} f(x_1+x_1', x_2+x_2') &= (x_1+x_1')(x_2+x_2') \\ &\neq x_1x_2+x_1'x_2' = f(x_1,x_2) + f(x_1',x_2') \ . \end{aligned} \tag{5.2}$$

Thus the sum of the function values is not equal to the function of the sum of the arguments or, in other words, the product of two binary digits is a nonlinear function. When we apply this and-function to the outputs of two distinct LFSRs we obtain the termwise product of the two shift register sequences. We will stipulate for the reminder of this chapter that whenever we speak of a product of two or more sequences we mean the termwise product of these sequences. Herlestam (Herl 82) uses the terminology "Hadamard product" to distinguish the product of two sequences from the multiplication

rule in the ring of formal power series. The concept of products of binary variables (or binary sequences) naturally extends to the case of a general boolean function of the arguments.

There exists a canonical form for boolean functions, the so-called algebraic normal form, which directly reflects the sum of products property. Let $\underline{x} \in \{0,1\}^N$ be a binary vector of dimension N and let $f(\underline{x})$ be an arbitrary nonlinear function of the components of $\underline{x}$, then f can be written out in algebraic normal form as

$$\begin{aligned} f(\underline{x}) = a_0 &+ a_1 x_1 + \ldots a_N x_N \\ &+ a_{12} x_1 x_2 + \ldots + a_{N-1,N}\, x_{N-1}\, x_N \\ &+ a_{123} x_1 x_2 x_3 + \ldots + a_{N-2,N-1,N}\, x_{N-2}\, x_{N-1}\, x_N \\ &\vdots \\ &\vdots \\ &+ a_{123\ldots N}\, x_1 x_2 \ldots x_N \end{aligned} \qquad (5.3)$$

A product of n variables is said to be an n-th order product. For example, $x_1 x_2 x_3$ is a third order product. The constant 1 is said to be a 0-th order product. The first order products are also called linear terms. The order of the function f is defined to be the maximum of the order of its product terms, that appear in (5.3) with the coefficients equal to 1. By way of convention, the zero function is defined to have order $-\infty$. Fig. 5.1 illustrates the definitions by means of an example.

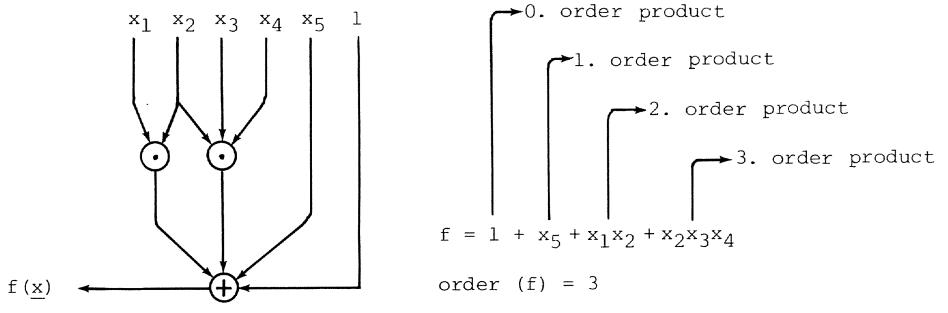


Fig. 5.1. Example of a nonlinear function of binary variables in algebraic normal form.

Instead of just looking at one specific argument vector $\underline{x}$ and one function value $f(\underline{x})$, we may think of the $\underline{x}$'s as representing binary sequences either from different LFSR's or from different stages of one LFSR. The algebraic normal form provides a fairly general reference system for which it is possible to set up a nonlinear theory of sequences. As we shall see later, the algebraic normal form preserves the relationship of growing nonlinearity (or equivalently, growing order of f) to growing complexity of the resulting output sequence in an almost ideal sense. But as we shall also see, we must distinguish the cases of taking as arguments shifted versions of the same sequence and of taking as arguments completely different sequences. The justification of the algebraic normal form as a useful base of reference will be given in the subsequent chapters when we apply the nonlinear theory associated to the algebraic normal form in the analysis of various types of stream ciphers.

We have used so far the word "nonlinear" exclusively in conjunction with products. This might give a wrong idea of "nonlinearity"; whether an operation is nonlinar or not strongly depends on the number system we are working in. Consider for example integer addition of two binary numbers, $S = x_1 + x_2$, where $x_1, x_2 \in \{0,1\}$ and S denotes their integer sum. Over the integers, addition clearly is a linear operation. But suppose we want to describe the integer addition of two binary variables in GF(2) - this is possible, when we take the binary representation of the sums as the result. Fig. 5.2 shows the difference.

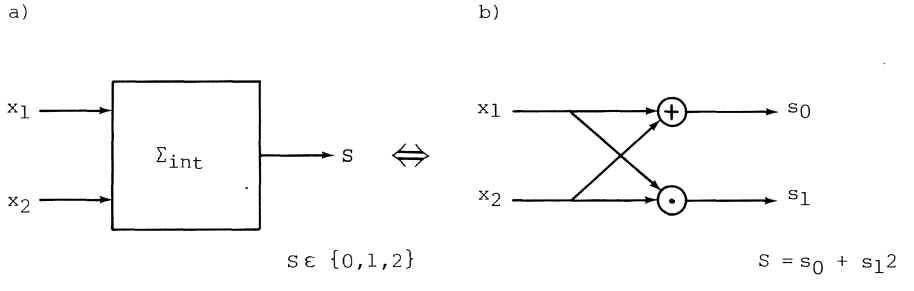


Fig. 5.2. Description of integer sum of two binary variables over the integers (a) and over GF(2) (b).

This illustrates that integer addition when realized in GF(2) is a nonlinear operation, whereas integer addition for itself is linear. So we must always remember the number system we are presently using when we classify operations or transformations as being linear or nonlinear. This fact will have special importance when we are ready to discuss the knapsack and related stream ciphers.

Three fundamental papers have appeared which discuss the problem of products of sequences. Zierler and Mills (Zier 73) were basically interested in finding that LFSR which could produce every sequence resulting from multiplying every possible output sequence of not necessarily distinct LFSRs. This approach essentially gives an upper bound on the linear complexity of any product sequence and thus goes from the cryptographer's viewpoint in the wrong direction. Nevertheless in some specific cases, precise linear complexity results for some product sequences may be obtained. Groth (Grot 71) concentrated on the use of 2nd 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, he summed as many of the 2nd order products as possible. Higher order nonlinearities were achieved, by layering, i.e., to the sequence generated by summing 2nd order products, some 2nd order products were again applied, 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 implied in this paper that the growth is completely deterministic, or in other words, that the linear complexity of the generated sequence can be easily controlled. As we shall see in this chapter, this is not the case; degeneracies in the linear complexities of the generated sequences may occur, and in general it is extremely difficult to lowerbound (or guarantee) the linear complexity of the sequences produced by nonlinearly filtering the state of an LFSR. Key (Key 76) investigated in his very readable and fundamental paper both the nonlinear filtering of a single LFSR and the nonlinear combination of the outputs of several distinct LFSRs. He (as did Groth) limited himself to consider only those sequences which are available at the stages of an LFSR, in particular he allowed only phase differences of at most the length of the LFSR. But when (as we will suggest in the next chapter) the speed of the LFSR is taken as an additional parameter, this theory must be extended to allow arbitrary phase differences. But then even the result of Key's which might be considered as his most solid one, namely, that a 2nd order product of 2 distinct

phases of the same sequence never degenerates, is no longer true. Key comments as follows on possible degeneracies: "Exceptions of this sort are few and present neither a serious theoretical difficulty nor a practical limitation". But unless the cryptographer knows at least the probability of a severe degeneracy, he will probably not be reassured by such a claim. Key also implies in his paper that the linear complexity of the produced sequences can be deterministically chosen. The situation when nonlinear combinations of output sequences of several distinct LFSR's are considered is completely different. Here many cases can be found where the linear complexity of the produced sequences can be guaranteed. For this reason, the nonlinear combiner is also sometimes referred to as a security amplifier function. Key shows that the product of 2 quintessential elements from distinct extension fields of relatively prime degrees is a quintessential element of the smallest extension field containing both of the former extension fields. This suffices to determine the linear complexity in some interesting cases, but not when the input sequences to the nonlinear combiner have reducible minimal polynomials. Herlestam (Herl 82) investigated also the product of two periodic sequences (which he called the Hadamard product) and derived necessary and sufficient conditions on the roots of the two associated minimal polynomials for the resulting linear complexity of the product sequence to be maximum.

5.1 Nonlinear Operations on Phases of a Sequence with Irreducible Minimal Poynomial

We begin by considering (see Fig. 5.3) an LFSR to whose stages a nonlinear function f is applied. We will call this also the case of a nonlinearly filtered LFSR. For the remainder of this section, we will assume that the driving LFSR produces a binary sequence.

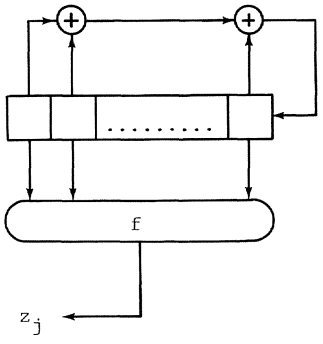


Fig. 5.3. An LFSR filtered by the general nonlinear function f

An LFSR of length L is said to be a maximal-length LFSR if, when started in a non-zero state, it generates a PN-sequence. One could say that PN-sequences exhibit the least complexity, since L digits completely suffice to specify the maximum possible period of 2^L-1 digits. These maximal-length sequences were called Pseudo-Noise (PN)-sequences since it appeared they showed noise-like statistics. But from the discussion of the linear complexity profile of random sequences (chapter 4) we know that the vast majority of truly random strings when repeated periodically would have linear complexity close to the period length. Consequently, the first period of such a PN sequence would be extremely unlikely as the output of a truly random source. Nevertheless, since the PN-sequences have maximum period with minimum hardware they provide the potential for the largest increase in linear complexity. Thus the best one can hope for with a generator of the type depicted in Fig. 5.3 is to generate sequences of linear complexity 2^L-1 with an LFSR of only L stages and a suitable nonlinear function f. Let σ_0 denote the initial state of the LFSR and let $\underline{z} = [z_0, z_1, \ldots, z_{2^L-2}]$ be the vector of bits produced by f in the first period of the LFSR. Since the state of the LFSR is the input to the nonlinear function f, we obtain the following relationship

$$
\begin{aligned}
z_0 &= f(\sigma_0) \\
z_1 &= f(\sigma_1) \\
&\vdots \\
z_{2^L-2} &= f(\sigma_{2^L-2})
\end{aligned}
\qquad (5.4)
$$

where σ_i is the successor of state σ_{i-1} in the LFSR.

When the LFSR has a primitive connection polynomial and is started in a nonzero state σ_0, then all 2^L-1 consecutive states $\sigma_0,\ldots,\sigma_{2^L-2}$ are distinct and nonzero. To uniquely specify a function f of L binary variables, we must assign to every distinct L-dimensional input vector its associated function value. Thus there is a unique correspondence between $\underline{z}$ and the function f with respect to the initial state σ_0 except for the value that f assigns to the zero state.

There is a subtle difficulty associated with the function value of the all-zero state. Since this state never appears within the cycle of the LFSR, the constant term in the algebraic normal form of f is undetermined by (5.4). Consequently, there are in fact two functions associated with every vector $\underline{z} \in \{0,1\}^{2^L-1}$. But to avoid ambiguities, f(0,...,0) will be defined to be zero.

Lemma 5.1.

> Given a maximal-length LFSR of length L with inital state $\sigma_0 \neq \underline{0}$, then for any binary vector $\underline{z} \in \{0,1\}^{2^L-1}$ there exists a unique corresponding function f which may produce this vector $\underline{z}$ as the first period of the output sequence z when applied to the stages of the LFSR.

When the connection polynomial of the LFSR is irreducible but not primitive, that is, when the state cycle has a period T which is a proper divisor of 2^L-1, then we clearly can generate any sequence $\tilde{z}$ periodic in T by choice of an appropriate f. But now the function f is no longer unique, since only those arguments appearing during the state cycle contribute to the function definition. To uniquely define f, we have to specify the desired output sequence $\tilde{z}$ for every possible state cycle. There are $(2^L-1)/T$ cyclically distinct output

sequences of period T, each associated to a state cycle of the LFSR. The initial state σ_0 of the LFSR determines the state cycle and thus also the particular sequence $\tilde{z}$ associated with it.

The impact of Lemma 5.1 is twofold. First, it tells us that no matter what vector $\underline{z} \in \{0,1\}^{2^L-1}$ we specify as initial period of the output sequence $\tilde{\underline{z}}$, there exists a function f for any initial state of the maximal-length LFSR which may generate this sequence. Recall that there are 2^{2^L-1} different sequences of period 2^L-1 or a divisor of 2^L-1. Thus f allows us in particular to generate all those periodic sequences which have a linear complexity close to the period length. And second, Lemma 5.1 promises an exponential growth in linear complexity of the produced sequences. With an LFSR of length L and a nonlinear function f whose order is also bound by L, most of the generated sequences will exhibit a linear complexity close to 2^L-1. Note that Lemma 5.1 only proved the existence of the desired function f and did not say how it may be obtained or what may be the possible relationship between a particular f and the linear complexity of the output sequence.

Let $\tilde{s}$ be the PN-sequence generated by the maximal-length LFSR. At each stage of the LFSR we will observe a different phase of $\tilde{s}$. So let $\tilde{s}_i$ denote the shifted version of $\tilde{s}$ observable at state i, i = 1, ..., L (in particular $\tilde{s} = \tilde{s}_1$) and let $\underline{s}_i \in \{0,1\}^{2^L-1}$ be the binary vector corresponding to the first period of $\tilde{s}_i$. Then using (5.3), we may write for the first period $\underline{z}$ of the produced output sequence $\tilde{z}$ (assuming that $f(0,\ldots,0) = 0$)

$$\underline{z} = a_1\underline{s}_1 + \ldots + a_L\underline{s}_L + a_{12}\underline{s}_1\underline{s}_2 + \ldots + a_{12\ldots L}\underline{s}_1\underline{s}_2\ldots\underline{s}_L \qquad (5.5)$$

$\underline{z}$ is element of a (2^L-1)-dimensional vector space and there are exactly 2^L-1 coefficients to be specified in f. Thus since any $\underline{z} \in \{0,1\}^{2^L-1}$ can be obtained by the linear combination (5.5), the 2^L-1 vectors $\underline{s}_1,\ldots,\underline{s}_L,\underline{s}_1\underline{s}_2,\ldots,\underline{s}_1\underline{s}_2\underline{s}_3,\ldots,\underline{s}_1\underline{s}_2\ldots\underline{s}_L$ are linearly independent. We may summarize this result in the following Lemma.

Lemma 5.2.

> When a general nonlinear function f is applied to the stages of a maximal-length LFSR then the 2^L-1 sequences corresponding to the isolated product terms in the algebraic normal form of f are linearly independent. Moreover they form a basis for the vector space of sequences with period 2^L-1.

Each coefficient in the algebraic normal form of f switches on (or off) a basis sequence for the considered vector space, thus the algebraic normal form of f provides a natural means to describe the resulting output sequences.

Clearly one can also generate any sequence of period 2^L-1 by taking a pure cycling shift register of length 2^L-1 and filling it with the first period of the desired sequence. In this alternative representation the 2^L-1 distinct unit vectors $[1,0,\ldots,0]$, $[0,1,0,\ldots,0]$, $[0,..,0,1]$ form the underlying basis which allows one to completely specify any vector $\underline{z} \in \{0,1\}^{2^L-1}$.

The pure cycling shift register of length 2^L-1 has connection polynomial

$$D^{2^L-1} + 1 = \prod_i C_i(D) \tag{5.6}$$

which factors into all the irreducible polynomials $C_i(d)$ of degree L_i' where L_i' divides L except the polynomial D. Let Z(D) be the D-transform of the sequence $\tilde{z}$. Then we may expand Z(D) into partial fractions as follows

$$Z(D) = \frac{Z^*(D)}{1+D^{2^L-1}} = \sum_i \frac{P_i(D)}{C_i(D)} \; ; \; \deg(Pi(D)) < \deg(Ci(D)) \tag{5.7}$$

where Z*(D) denotes the D-transform of the first period of $\tilde{z}$, or equivalently, the initial contents of the pure cycling shift register. Equation (5.7) expresses Z(D) as the sum of sequences from shorter LFSR's, the i-th of which has connection polynomial $C_i(D)$. The total length of these component LFSRs is 2^L-1.

Thus, the 2^L-1 state variables that define the initial contents of these component LFSRs can be used to specify a third basis for the vector space of initial period vectors $\underline{z}$ in the manner that each sequence in the basis is the output of one of the component LFSR's when it has a single 1 in its initial contents.

Let us make an example to illustrate the three bases encountered so far. Suppose the LFSR has length L = 4 and we choose the primitive polynomial $C(D) = 1 + D + D^4$ as connection polynomial. Let the initial state be [0111] which corresponds to the characteristic phase of the LFSR. Then the simple nonlinear generator arrangement from Fig. 5.3 becomes

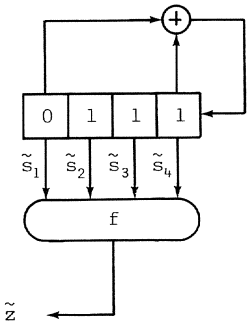


Fig. 5.4. The $<4,1+D+D^4>$ maximal-length LFSR filtered by a nonlinear function f.

Lemma 5.2. states that the sequences corresponding to the isolated product terms in the algebraic normal form are linearly independent. For illustration we will list their first period vector, which contains all the information necessary to specify the associated sequence.

Single product in ANF(f)	First period vector produced by single product term
1	0 1 1 1 1 0 1 0 1 1 0 0 1 0 0
2	1 1 1 1 0 1 0 1 1 0 0 1 0 0 0
3	1 1 1 0 1 0 1 1 0 0 1 0 0 0 1
4	1 1 0 1 0 1 1 0 0 1 0 0 0 1 1
12	0 1 1 1 0 0 0 0 1 0 0 0 0 0 0
13	0 1 1 0 1 0 1 0 0 0 0 0 0 0 0
14	0 1 0 1 0 0 1 0 0 1 0 0 0 0 0
23	1 1 1 0 0 0 0 1 0 0 0 0 0 0 0
24	1 1 1 0 0 0 0 1 0 0 0 0 0 0 0
34	1 1 0 0 0 0 1 0 0 0 0 0 0 0 1
123	0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
124	0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
134	0 1 0 0 0 0 1 0 0 0 0 0 0 0 0
234	1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1234	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

Table 5.1. The basis vectors for the 15-dimensional vector space as generated by single product terms of the algebraic normal form (ANF) of f (e.g. 124 corresponds to the product term $\underline{s}_1\underline{s}_2\underline{s}_4$).

Therefore any 15-dimensional vector $\underline{z}$ may be expressed as

$$\underline{z} = a_1\underline{s}_1 + \ldots + a_4\underline{s}_4 + a_{12}\underline{s}_1\underline{s}_2 + \ldots + a_{1234}\underline{s}_1\underline{s}_2\underline{s}_3\underline{s}_4 \qquad (5.8)$$

or written in matrix form as

$$\underline{z} = p^t\underline{a} \qquad (5.9)$$

where P denotes the matrix whose rows are formed by all the single product vectors (as they stand in table 5.1) and $\underline{a}$ denotes the coefficient vector of the algebraic normal form of f with the succession of the coefficient a_i defined lexicographically (compare table 5.1).

The second basis encountered (which is the obvious one) consists of all 15 unit vectors $\underline{e}_1^t = [1,0,\ldots,0]$, ..., $\underline{e}_{15}^t = [0,\ldots,0,1]$.

Consequently

$$\underline{z} = b_1\underline{e}_1 + \ldots + b_{15}\underline{e}_{15} \tag{5.10}$$

and in matrix form

$$\underline{z} = I\,\underline{b} \tag{5.11}$$

where I denotes the identity matrix and $\underline{b}$ is just the 15-dimensional vector $\underline{z}$. This second basis corresponds to the pure cycling LFSR of length 15 where the ith initial state bit is set to 1 when the coefficient b_i is 1.

The third basis provides some additional insight into the structural properties of sequences having period 15. $D^{15}+1$ factors into the irreducible polynomials

$$\begin{aligned}
&1 + D\\
&1 + D + D^2\\
&1 + D + D^4\\
&1 + D^3 + D^4\\
&1 + D + D^2 + D^3 + D^4.
\end{aligned} \tag{5.12}$$

Thus we may decompose the pure cycling shift register of length 15 into 5 distinct LFSRs having the above 5 irreducible connection polynomials:

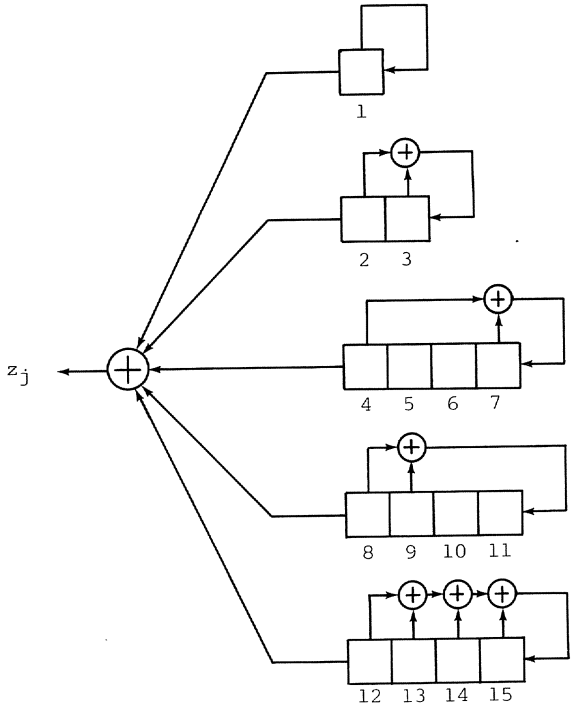


Fig. 5.5. The general decomposed linear equivalent of the pure cycling shift register of length 15.

The natural choice of the basis in the general decomposed linear equivalent are the 15 state variables. We may now list the 15 distinct and linearly independent sequences produced by a single 1 at any of the 15 possible state locations.

State bit r_i	Inital period vector $\underline{d}_i^t$ produced when $r_i = 1$ $r_j = 0$, $j \neq i$.
1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2	1 0 1 1 0 1 1 0 1 1 0 1 1 0 1
3	0 1 1 0 1 1 0 1 1 0 1 1 0 1 1
4	1 0 0 0 1 1 1 1 0 1 0 1 1 0 0
5	0 1 0 0 0 1 1 1 1 0 1 0 1 1 0
6	0 0 1 0 0 0 1 1 1 1 0 1 0 1 1
7	0 0 0 1 1 1 1 0 1 0 1 1 0 0 1
8	1 0 0 0 1 0 0 1 1 0 1 0 1 1 1
9	0 1 0 0 1 1 0 1 0 1 1 1 1 0 0
10	0 0 1 0 0 1 1 0 1 0 1 1 1 1 0
11	0 0 0 1 0 0 1 1 0 1 0 1 1 1 1
12	1 0 0 0 1 1 0 0 0 1 1 0 0 0 1
13	0 1 0 0 1 0 1 0 0 1 0 1 0 0 1
14	0 0 1 0 1 0 0 1 0 1 0 0 1 0 1
15	0 0 0 1 1 0 0 0 1 1 0 0 0 1 1

Table 5.2. The basis vectors $\underline{d}_1^t, \ldots, \underline{d}_{15}^t$ for the 15-dimensional vector space as generated by single state bits of 1 in the general decomposed linear equivalent.

Therefore any 15-dimensional vector can be expressed as

$$\underline{z} = r_1\underline{d}_1 + \ldots + r_{15}\underline{d}_{15} \tag{5.13}$$

or written in matrix form as

$$\underline{z} = D^t\underline{r} \tag{5.14}$$

where D denotes the matrix whose rows are formed by the basis vectors $\underline{d}_1^t, \ldots, \underline{d}_{15}^t$ (as they stand in table 5.2) and $\underline{r}$ denotes the initial loading of the 15 state cells in the general decomposed linear equivalent.

The "algebraic normal form basis" allows to calculate the specific nonlinear function which produces a given sequence $\tilde{z}$ of period 2^L-1. The "general decomposed linear equivalent basis" allows us to determine the linear complexity of the sequence $\tilde{z}$, and in particular, to find which of the available polynomials contribute to the production of $\tilde{z}$. To illustrate these conversions, we randomly selected with a fair Swiss coin the initial period of the following sequence of period 15,

$$\tilde{z} = (010111011000110)^{\infty} . \tag{5.15}$$

The first period $\underline{z} \in \{0,1\}^{15}$ of $\tilde{z}$ specifies the remainder of the semi-infinite sequence. To determine the decomposed linear equivalent, we may compute, using (5.14)

$$\underline{r} = (D^t)^{-1}\underline{z} \tag{5.16}$$

which results in the initial state vector

$$\underline{r} = [000010011001101] . \tag{5.17}$$

This allows us to draw the decomposed linear equivalent for the "Swiss coin sequence" (5.15).

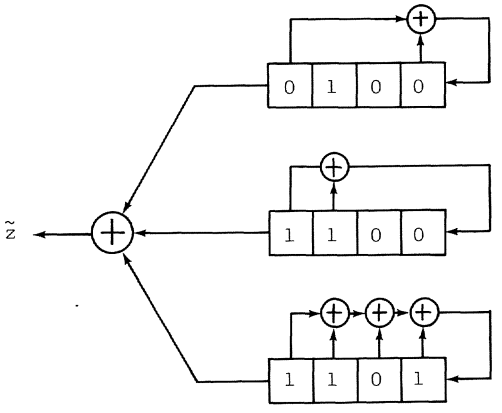


Fig. 5.6. The decomposed linear equivalent associated to the "Swiss coin sequence" $\tilde{z}$ given in (5.15).

The 2 LFSRs with connection polynomials $1+D$ and $1+D+D^2$ do not contribute to the generation of $\tilde{z}$, thus the linear complexity of the

"Swiss coin sequence" (5.15) is only 12. Note that this is again a confirmation of the principles developed on random sequences (see chapter 4); the linear complexity of a periodically repeated random string is highly likely to be close to the period length.

There would have been a much easier way to determine directly the linear complexity of $\tilde{z}$, we could have simply applied the Massey-Berlekamp Shift Register Synthesis Algorithm (Mass 69) to obtain the linear equivalent associated to $\tilde{z}$, which is shown in Fig. 5.7.

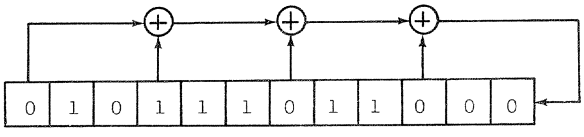


Fig. 5.7. Linear equivalent associated to the "Swiss coin sequence" (5.15).

The connection polynomial of the linear equivalent generator corresponds to the product the contributing irreducible connection polynomials in the decomposed linear equivalent generator. One could have simply generated the "Swiss coin sequence" by loading its first period into a pure cycling shift register of length 15, but this would not have properly reflected the structural properties of $\tilde{z}$.

We now turn to the remaining problem of what particular nonlinear function f could have produced (together with the driving maximal-length LFSR in Fig. 5.4) the "Swiss coin sequence" (5.15). We may solve for the coefficient vector $\underline{a}$ of f, using (5.9), as follows:

$$\underline{a} = (P^t)^{-1} \cdot \underline{z} \quad . \tag{5.18}$$

Note that the matrices P^t and D^t may always be inverted since they have rank 2^L-1, or equivalently, their columns define a basis for the (2^L-1)-dimensional vector space. Carrying out (5.18) with the "Swiss coin sequence" (5.15) results in

$$\underline{a} = [100101010101110] \tag{5.19}$$

which allows us to write the algebraic normal form of f as

$$f(x_1,x_2,x_3,x_4) =$$

$$x_1 + x_4 + x_1x_3 + x_2x_3 + x_3x_4 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4 \ . \qquad (5.20)$$

This f, applied to the stages of the driving LFSR, is shown in Fig. 5.8.

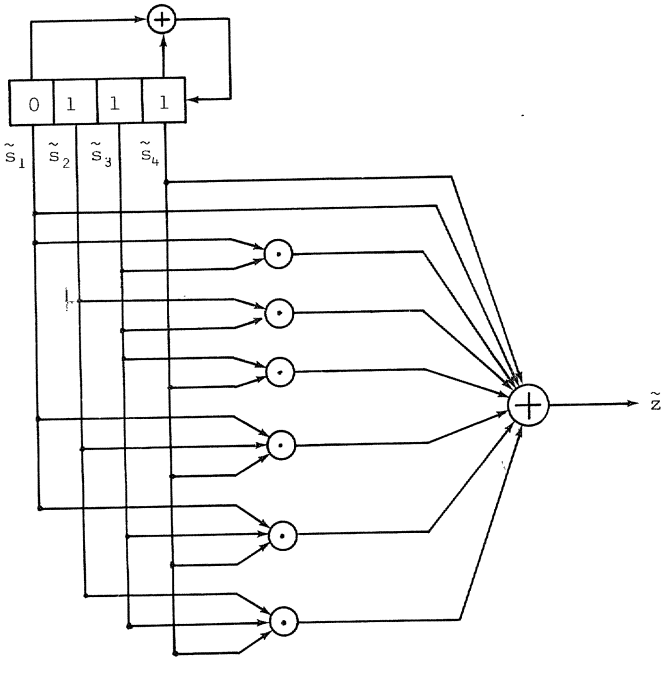


Fig. 5.8. The nonlinear generator associated to the "Swiss coin sequence" (5.15)

So far, we have seen that it is (in principle) possible initially to specify a sequence whose period divides 2^L-1 and then to derive the associated nonlinear generator and the associated decomposed linear equivalent. But we do not thereby have any control over the linear complexity of the sequence $\tilde{z}$, since we select $\tilde{z}$ prior to the analysis. What the cryptographer would like to possess is a method which allows him to specify the linear complexity of a sequence and then returns to him a nonlinear function that produces a sequence of the specified linear complexity. It might not be obvious, but we have the tools already at hand to do this. Combining (5.14) and (5.18) we obtain

$$\underline{a} = (P^t)^{-1} D^t \underline{r} = (D \cdot P^{-1})^t \underline{r} \ . \qquad (5.21)$$

Hence, we may specify an arbitrary initial state for the general decomposed linear equivalent, and, after multiplication with the matrix $(DP^{-1})^t$, we may read off the nonlinear function which produces this sequence when applied to the chosen maximal-length LFSR. In particular it is possible to make any available polynomial $C_i(D)$ contribute to the linear complexity of the produced sequence by loading the associated LFSR with a nonzero state. Since each row in DP^{-1} corresponds to a nonlinear function which maps the driving maximal-length LFSR into itself or into any other available LFSR with irreducible connection polynomial of degree that divides L, we may call these functions also transformer functions. Before we illustrate the concept of a transformer function, let us write out the matrix DP^{-1} which exhibits some particularly interesting properties:

activated polynomial	state bit	coefficient vector $\underline{a}_i^t$ of the algebraic normal form of f associated to state bit r_i														
$C_i(D)$	r_i	1	2	3	4	12	13	14	23	24	34	123	124	134	234	1234
1+D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$1+D+D^2$	2	1	1	0	0	1	1	0	1	0	1	0	0	0	0	0
	3	0	1	1	1	0	0	1	1	1	1	0	0	0	0	0
$1+D+D^4$	4	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
	6	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
	7	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
$1+D^3+D^4$	8	1	0	1	1	0	1	0	0	1	1	1	1	1	1	0
	9	1	1	1	0	0	1	0	1	0	1	1	0	0	0	0
	10	1	1	1	1	1	0	0	0	1	0	1	1	0	0	0
	11	1	1	0	1	0	1	1	0	0	0	1	1	1	0	0
$1+D+D^2+D^3+D^4$	12	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0
	13	0	1	0	0	1	1	1	1	1	1	0	0	0	0	0
	14	1	0	0	0	1	0	0	1	0	1	0	0	0	0	0
	15	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0

Table 5.3. The basis functions $\underline{a}_1^t, .. \underline{a}_{15}^t$ for the 15 dimensional vector space, or equivalently, the matrix DP^{-1}.

Comments:

(1) The particular form of the 15 by 15 matrix DP^{-1} shown in table 5.3 depends on the driving maximal-length LFSR $<4,1+D+D^4>$ and its initial state [0 1 1 1].

(2) r_i denotes the state bit in the general decomposed linear equivalent which is set to one by the corresponding function $\underline{a}_i$. Thus, when the nonlinear function f defined by the coefficient vector $\underline{a}_i$ is applied to the driving LFSR, then the same periodic sequence results as when the general decomposed linear equivalent is started with state bit $r_i = 1$ as the only non-zero bit.

(3) The activated polynomial is the connection polynomial of the linear equivalent of the sequence produced by $\underline{a}_i$ together with the driving maximal-length LFSR.

With the help of table 5.3. it is now easy to specify a particular transformer. Suppose we select row 12 of DP^{-1}, then the nonlinear generator defined by the driving LFSR and the nonlinear function $\underline{a}_{12}$ simulates the behaviour of $<4,1+D+D^2+D^3+D^4>$ with initial state [1000] (see Fig. 5.9).

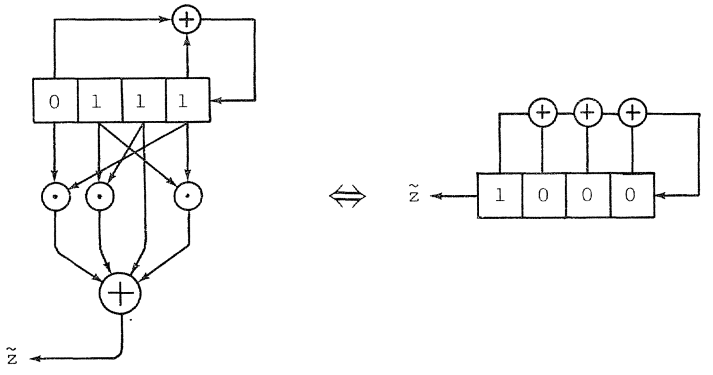


Fig. 5.9. The nonlinear generator simulating the LFSR $<4,1+D+D^2+D^3+D^4>$

In exactly the same way we may simulate any single LFSR of the general decomposed linear equivalent (Fig. 5.5) having any desired initial state. All we need is to specify the corresponding initial state vector $\underline{r}$, then to multiply by $(DP^{-1})^t$, and we obtain the co-

efficient vector a defining the nonlinear function f which is able, when applied to the driving LFSR's stages, to simulate the desired LFSR. We have thus solved, at least in principle, the cryptographer's problem of finding nonlinear functions f which produce sequences of guaranteed linear complexity. With the help of table 5.3. we may in fact generate sequences $\tilde{z}$ having any linear complexity between 0 and 15. The reason for this versatility is easily seen from the general decomposed linear equivalent in Fig. 5.5. There is one connection polynomial of degree 1, one of degree 2, and three of degree 4. It is possible to represent any number (i.e. any linear complexity) between 0 and 15 as the degree of a product of these polynomials. But in general not every linear complexity between 0 and 2^L-1 can be obtained. Take for example the case where L is a prime, then all the distinct irreducible polynomials which divide $1+D^{2^L-1}$, have either degree L or 1. Since 1+D is the only irreducible polynomial of degree 1 which divides $1+D^{2^L-1}$, all the remaining irreducible polynomials must have degree L. Consequently the linear complexities which may be generated by nonlinearly filtering a maximal-length LFSR of prime length L have the form kL or kL+1, $0 \leqslant k \leqslant (2^L-2)/L$.

Since the matrices in the discussed synthesis procedure have dimensions 2^L-1 by 2^L-1, even for small L the matrix approach becomes unfeasible. Let us return for a moment to table 5.3; the matrix DP^{-1} exhibits a very peculiar structure. At the right side of the dashed line we find only zeros, but the right entries of the matrix correspond to a higher order nonlinearity than the left entries. Thus the dashed line explains how "much" nonlinearity is needed to activate any of the LFSRs in the general decomposed linear equivalent. With the driving LFSR $\langle 4,1+D+D^4\rangle$ we only need linear logic to activate $C_3(D) = 1+D+D^4$ (as is obvious), we need 2nd-order logic to activate $C_2(D) = 1+D+D^2$ and $C_5(D) = 1+D+D^2+D^3+D^4$, we need 3rd-order logic to activate $C_4(D) = 1+D^3+D^4$, and finally, $C_1(D) = 1+D$ can only be activated by 4th-order logic. Therefore the key property of the nonlinear function f which relates to the linear complexity of the produced sequences $\tilde{z}$ must be the order of f. For suppose we randomly select a 2nd order function f and apply it to the driving LFSR $\langle 4,1+D+D^4\rangle$. Then we know with certainty that $C_1(D) = 1+D$ and $C_4(D) = 1+D^3+D^4$ may not contribute to the linear complexity of the produced sequence. But we expect that $C_2(D) = 1+D+D^2$, $C_3(D) = 1+D+D^4$, and $C_5(D) = 1+D+D^2+D^3+D^4$ are contained in the linear equivalent gene-

rator. The crux is that the order of f does not contain enough information to guarantee that $C_2(D)$, $C_3(D)$ and $C_5(D)$ contribute to the linear complexity of the produced sequence. For instance the transformer function which simulated $C_2(D)$ has 2nd order, as well as the transformer function which simulates $C_5(D)$. Thus when the order of f is our main parameter in the synthesis of sequences with controllable linear complexity, we must be aware that a fundamental uncertainty about the actual linear complexity produced remains, which can only be resolved by acquiring additional structural information about f or the generated sequence. The best we can hope for in this situation is to make the probability arbitrarily small that the linear complexities of the generated sequences differ by a large amount from the expected value. Certainly the most desirable situation would be a procedure which could synthesize easily classes of functions with a guaranteed associated linear complexity. But this might be an unrealistic idea, since even the description of one of the basis functions (or transformer functions) for the (2^L-1)-dimensional vector space requires as much data as the first period of the sequence itself.

We conclude the discussion of the structural properties of nonlinear functions f by 2 more simple results.

Lemma 5.3.

The Lth-order product of all L stages of an LFSR with primitive connection polynomial (and nonzero state) always produces a sequence with maximum linear complexity 2^L-1.

Proof The driving LFSR generates a PN-sequence. Thus exactly once on its state cycle of length 2^L-1 the driving LFSR experiences the all-one state, which is the only state which contributes a one to the output. Consequently only the pure cycling shift register of length 2^L-1 can produce this sequence.

From the cryptographic viewpoint the Lth-order product is useless although the resulting sequence $\tilde{z} = (0\ldots010\ldots0)^\infty$ has maximum possible linear complexity. But the sequence $\tilde{z}$ does not at all possess

the typical linear complexity profile of a random sequence as discussed in the preceding chapter. $\tilde{z} = (0...010...0)^{\infty}$ is in fact highly predictable: by approximating it by the all-zero sequence $\tilde{0}$ (which has linear complexity 0), that is, by simply ignoring it, we can predict $\tilde{z}$ except for just one bit position within the period. This suggests also that nonlinear transformations which only utilize high-order products are cryptographically dangerous. Although the resulting sequences have the desired high linear complexities, they can in general be predicted more reliably. But note that these sequences would never pass randomness tests based on the linear complexity profile. We conclude that it is advantageous to use nonlinear transformation which incorporate many products of any order up to some maximum order.

The properties exhibited by the nonlinear filter function f also depend strongly on the particular connection polynomial of the driving LFSR. An interesting question is whether there exist some structural relationship between distinct (connection polynomial/nonlinear function)-pairs. To investigate this question let us generalize the concept of nonlinearily filtering the stages of an LFSR with primitive connection polynomial to nonlinearily combining arbitrary but distinct phases $\tilde{s}$, $\tilde{s}^{t_1}$, $\tilde{s}^{t_2}$, ..., $\tilde{s}^{t_n}$ ($0 < t1 < t2 < ... < tn <$ period of $\tilde{s}$) of the sequence $\tilde{s}$, whose minimal polynomial $m_{\tilde{s}}(X) \in GF(2)[X]$ is irreducible and has degree L. Then for any nonlinear function f of n+1 variables, let the resulting sequence

$$\tilde{z} = f(\tilde{s}, \tilde{s}^{t_1}, \ldots \tilde{s}^{t_n}) \tag{5.22}$$

be defined by

$$z_j = f(s_j, s_{j+t_1}, \ldots, s_{j+t_n}) \ . \tag{5.23}$$

It is known that LFSRs with reciprocal connection polynomials produce sets of sequences that are the same in reverse order. Let $\tilde{r}$ be the reverse of $\tilde{s}$, i.e. $s_j = r_{T-1-j}$, when T denotes the period of $\tilde{s}$. Substituting s_{j+t_i} by $r_{T-1-j-t_i}$ in (5.23) yields

$$z_j = f(r_{T-1-j}, r_{T-1-j-t_1}, \ldots, r_{T-1-j-t_n}). \tag{5.24}$$

Now reversing the order of $\tilde{z}$ by replacing j by T-1-j we obtain

$$z_{T-1-j} = f(r_j, r_{j-t_1}, r_{j-t_2}, \ldots, r_{j-t_n}) \tag{5.25}$$

from which it directly follows that

$$z_{T-1-j+t_n} = f(r_{j+t_n}, r_{j+t_n-t_1}, \ldots, r_j) \tag{5.26}$$

Note that the arguments of f now are the sequences $\tilde{r}^{t_n}$, $\tilde{r}^{t_n-t_1}$, ..., $\tilde{r}$ which means that the original phases are symmetrically changed, i.e. t_i has been replaced by t_n-t_i. Since $\tilde{z}$ and its reverse have reciprocal minimal polynomials, they have same linear complexity. Therefore we have proven the following proposition:

Proposition 5.4. Symmetry Property

Let $\tilde{s}$, $\tilde{s}^{t_1}$, ..., $\tilde{s}^{t_n}$ be distinct phases ($0 < t_1 < t_2 < \ldots t_n <$ period of $\tilde{s}$) of the sequence $\tilde{s}$ whose minimal polynomial $m_{\tilde{s}}(X) \in GF(2)[X]$ is irreducible and has degree L.
Let $\tilde{r}$ be the reverse of $\tilde{s}$, i.e. $s_j = r_{T-1-j}$ where T is the period of $\tilde{s}$. Then for any function f, the sequence $\tilde{z}$ defined by

$$z_j = f(s_j, s_{j+t_1}, \ldots, s_{j+t_n}) \tag{5.27}$$

and the sequence $\tilde{y}$ defined by

$$y_j = f(r_{j+t_n}, r_{j+t_n-t_1}, \ldots, r_j) \tag{5.28}$$

where in (5.28) the phases of the arguments are symmetrically changed (i.e. t_i is replaced by t_n-t_i, i=1,...,n), are also reverses of one another thus have the same linear complexity.

Proposition 5.4 implies that all results derived for a nonlinearly filtered LFSR whose connection polynomial is irreducible also hold for the configuration, where the connection polynomial is replaced by its reciprocal (which corresponds to symmetrically changing the feedback taps) and the nonlinear filter function is replaced by its "reciprocal" (which corresponds to symmetrically changing the function taps). Let us illustrate the symmetry principle by means of an example. The standard driving LFSR used in this section was

$<4,1+D+D^4>$, its reciprocal is $<4,1+D^3+D^4>$; let the nonlinear function f be the single product $x_1x_3x_4$, its "reciprocal" then is $x_1x_2x_4$.

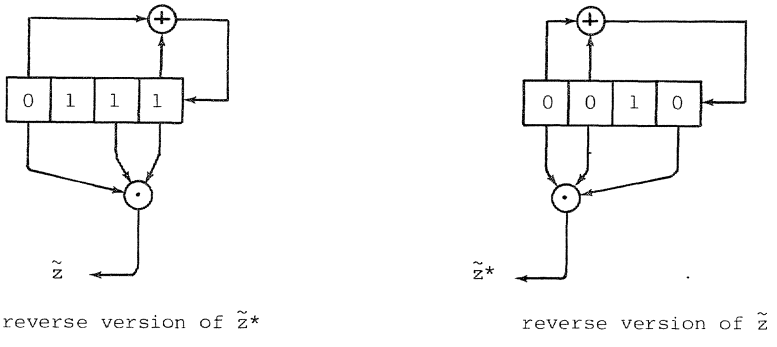


Fig. 5.10. Reciprocal nonlinear generator pair, each one producing the reversed version of the other's output sequence

It is well known from the work of Groth (Grot 71), Key (Key 76) and others that the order of a nonlinear function plays a key role in the determination of the linear complexity of the produced sequences, but that there remains a fundamental uncertainty about the actual linear complexity, since the order of f does not always determine the linear complexity.

The following result appears to be generally known, but does not appear explicitly in the literature.

General upperbound on the linear complexity of nonlinearly filtered PN-sequences.

Let f be any kth-order function of k distinct phases $\tilde{s}^{t_1}$, $\tilde{s}^{t_2}$, ..., $\tilde{s}^{t_k}$ of a maximum-length sequence $\tilde{s}$.

$$\tilde{z} = f(\tilde{s}^{t_1}, \ldots, \tilde{s}^{t_k}) \quad . \tag{5.29}$$

Then, the linear complexity of $\tilde{z}$ is upperbound by

$$\Lambda(\tilde{z}) \leq \sum_{i=1}^{k} \binom{L}{i} \tag{5.30}$$

Proof: Each $\tilde{s}^{t_i}$, by the fundamental shift-and-add property of PN-sequences, is a non-zero linear combination of $\tilde{s}$, $\tilde{s}^1$, ..., $\tilde{s}^{L-1}$. Thus, (5.29) is equivalent to a different nonlinear function of order at most k operating on $\tilde{s}$, $\tilde{s}^1$, ..., $\tilde{s}^{L-1}$. The bound (5.30) follows immediately from the work of Key (Key 76).

But in the cryptographic application one is rather interested in generating sequences with a guaranteed large minimum linear complexity. For the class of bent sequences, Kumar and Scholtz (Kuma 83) were able to derive a lowerbound which slightly exceeds $\binom{L/2}{L/4} 2^{L/4}$, where L (denoting the length of a m-LFSR) is restricted to be a multiple of 4. In the sequel we will derive a general lowerbound of $\binom{L}{k}$ on the linear complexity of nonlinearly filtered m-sequences which holds for a broad and simple class of functions, and is not constrained in the length L of the m-LFSR or the nonlinear order k of the function employed. We start by examining the product of k phases of a m-sequence.

Proposition 5.5. Root presence test for the product of distinct phases of a PN-sequence

Let $\tilde{s}^{t_1}$, $\tilde{s}^{t_2}$, ..., $\tilde{s}^{t_k}$ be be k distinct phases of the same sequence $\tilde{s}$ whose minimal polynomial $m_{\tilde{s}}(X) \in GF(2)[X]$ is primitive and has degree L. Let $\alpha \in GF(q^L)$ denote a root of $m_{\tilde{s}}(X)$, and let $\tilde{z}$ be the product of the given k distinct phases, i.e.

$$\tilde{z} = \tilde{s}^{t_1} \tilde{s}^{t_2} \ldots \tilde{s}^{t_k} = \prod_{i=1}^{k} \tilde{s}^{t_i} \; . \tag{5.31}$$

Let $W_2(i)$ denote the Hamming weight of the radix-2 form of the integer i. Then α^e with $W_2(e) = k$ is a root of the minimal polynomial of $\tilde{z}$ if and only if the determinant

$$A_e = \begin{vmatrix} \alpha^{t_1 2^{e_1}} & \cdots & \alpha^{t_k 2^{e_1}} \\ \alpha^{t_1 2^{e_2}} & \cdots & \alpha^{t_k 2^{e_2}} \\ \vdots & & \vdots \\ \alpha^{t_1 2^{e_k}} & \cdots & \alpha^{t_k 2^{e_k}} \end{vmatrix} \neq 0 \qquad (5.32)$$

where $e = 2^{e_1} + 2^{e_2} + \ldots + 2^{e_k}$, $0 \leq e_1 < e_2 < \ldots < e_k < L$.

Remarks:

Since the roots are always grouped in sets of conjugates, it is sufficient in applications of proposition 5.5 to consider those exponents e which are coset leaders in the ring of integers modulo (2^L-1). There are $\binom{L}{k}$ distinct roots α^e whose exponents have weight $W_2(e) = k$.

Proof: Assume without loss of generality that $\tilde{s}$ is in its characteristic phase, i.e.

$$s_j = T_L(\alpha^j) = \alpha^j + \alpha^{2j} + \ldots + \alpha^{2^{L-1}j} . \qquad (5.33)$$

Then by the shift property of sequences with irreducible minimal polynomial (compare section 3.4.)

$$(\tilde{s}^t)_j = T_L(\alpha^t \alpha^j) \qquad (5.34)$$

and we obtain for the jth bit of $\tilde{z}$

$$z_j = \prod_{i=1}^{k} T_L(\alpha^{t_i} \alpha^j)$$

$$\neq \prod_{i=1}^{k} (\alpha^{t_i} \alpha^j + \alpha^{2t_i} \alpha^{2j} + \ldots + \alpha^{2^{L-1}t_i} \alpha^{2^{L-1}j}) \qquad (5.35)$$

which can be decomposed into one sum with roots α^e whose exponents have weight $W_2(e) < k$ and into a second sum with roots α^e whose exponents have weight $W_2(e) = k$. Denoting the first sum as y_j, we have

$$z_j = y_j + \sum_{\{e:W_2(e)=k\}} A_e \alpha^{ej} \tag{5.36}$$

where A_e denotes the resulting coefficient of α^e. In order to obtain an exponent e with binary weight k from k powers of 2, they all must be different, that is $e = 2^{e_1} + 2^{e_2} + .. + 2^{e_k}$, with $0 \leq e_1 < e_2 < \ldots < e_k < L$. Since there are $\binom{L}{k}$ ways in which k distinct powers of 2 may be selected from L available, the sum contains $\binom{L}{k}$ terms.

A specific exponent $e = 2^{e_1} + 2^{e_2} + \ldots + 2^{e_k}$, $0 \leq e_1 < e_2 < \ldots < e_k < L$ can be obtained in k! different ways corresponding to the k! permutations of the k distinct powers of 2. In other words, there are k! choices of distinct $m_1, .., m_k$ such that $e = 2^{m_1} + 2^{m_2} + \ldots + 2^{m_k}$. This in turn corresponds to the number of ways in which one can select the k distinct powers of 2 each from a different trace in (5.35). Thus the resulting coefficient of α^{ej} in (5.36) will be

$$A_e = \sum_{\underline{m} \varepsilon P_e} (\alpha^{t_1})^{2^{m_1}} (\alpha^{t_2})^{2^{m_2}} \ldots (\alpha^{t_k})^{2^{m_k}} \tag{5.37}$$

where P_e is the set of all permutations of $(e_1, \ldots, e_k)$. Taking into account that addition and subtraction coincide over any finite field of characteristic 2, it follows that (5.37) computes the k by k determinant as defined in (5.32). We conclude that just when the determinant (5.32) is nonzero for a specific exponent e, the root α^e contributes to the linear complexity of $\tilde{z}$.

The root presence test for the product of distinct phases of a PN-sequence is in fact not so complicated as it might look at first glance. Since $m_{\tilde{z}}(X)$ necessarily contains only simple roots, the

number of roots of $m_{\tilde{z}}(X)$ is equal to the linear complexity of $\tilde{z}$ or, equivalently, to the length of the linear equivalent generator (the connection polynomial of the linear equivalent generator is in fact the reciprocal of the minimal polynomial $m_{\tilde{z}}(X)$ of the product sequence $\tilde{z}$). Proposition 5.5 now essentially provides a test for determining which of $\binom{L}{k}$ distinct roots α^e, which may contribute to the linear complexity of $\tilde{z}$, actually do so contribute. Since these roots always appear in sets of conjugates, we only have to test whether one in each conjugate set is a root of $m_{\tilde{z}}(X)$, for instance we need only test the coefficients of those α^e whose exponents are coset leaders in the ring of integers modulo 2^L-1. Let us illustrate the test by an example. Assume the PN-sequence $\tilde{s}$ has minimal polynomial $m_{\tilde{s}} = X^4+X^3+1$. Then

$$s_j = T_4(\alpha^j) = \alpha^j + \alpha^{2j} + \alpha^{4j} + \alpha^{8j} \ . \tag{5.38}$$

Consider the 2nd order product $\tilde{z} = \tilde{s}\cdot\tilde{s}^2$. In $GF(2^4)$ only α^3 and α^5 have exponents that are coset leaders and that have binary weight 2. Applying the test (5.32), we have

$$A_3 = \begin{vmatrix} 1 & \alpha^2 \\ 1 & \alpha^4 \end{vmatrix} = \alpha^4+\alpha^2 \neq 0 \qquad A_5 = \begin{vmatrix} 1 & \alpha^2 \\ 1 & \alpha^8 \end{vmatrix} = \alpha^8+\alpha^2 \neq 0 \tag{5.39}$$

and we thus find that $X^2 + X + 1$ (the minimumpolynomial of α^5) and $X^4 + X^3 + X^2 + X + 1$ (the minimumpolynomial of α^3) are both factors of $m_{\tilde{z}}(X)$.

One might think that with 2nd-order products the linear complexity of $\tilde{z}$ is always bounded from below by $\binom{L}{2}$ since the determinant (5.32) apparently results in the sum of 2 distinct elements of $GF(2^L)$ for all α^e to be considered and thus can never be zero. But this is not true as we illustrate by means of an example. Let the PN-sequence $\tilde{s}$ be the same as in (5.38) and consider the 2nd-order product $\tilde{z} = \tilde{s}\cdot\tilde{s}^5$. Then test (5.30) will result in

$$A_3 = \begin{vmatrix} 1 & \alpha^5 \\ 1 & \alpha^{10} \end{vmatrix} \neq 0 \qquad A_5 = \begin{vmatrix} 1 & \alpha^5 \\ 1 & \alpha^5 \end{vmatrix} = 0 \ . \tag{5.40}$$

We find that $X^2 + X + 1$ (the minimum polynomial of α^5) is <u>not</u> a factor of $m_{\tilde{z}}(X)$, and thus the roots α^5, α^{10} do not contribute to the linear complexity of $\tilde{z}$. In the literature the notion of <u>degeneracy</u>

has been adopted for such a case where the linear complexity of the nonlinearily produced sequence $\tilde{z}$ is not maximal or, equivalently, where at least one of the possible factors of $m_{\tilde{z}}(X)$ is missing. In his paper, Key considered only phase shifts of at most the LFSR-length L. With this restriction added in, it is true that a single 2nd-order product will never produce a degeneracy. But as we have seen in the above example, when arbitrary phase shifts are allowed, even the simple 2nd-order product cannot be guaranteed not to result in a degeneracy.

Proposition 5.5 can be generalized to a root presence test for the product of distinct phases of a sequence $\tilde{s}$ whose minimal polynomial $m_{\tilde{s}}(X)$ is irreducible, but nonprimitive. We express the roots of $m_{\tilde{s}}(X)$ as some power α^d of a primitive element $\alpha \in GF(2^L)$. Let the binary weight of the exponent be $W_2(d) = m$. Then the 2nd-order product introduces possibly all roots α^e, whose exponent e has binary weight $W_2(e) = 2m$ and whose order is at most the order of α^d; i.e. $ord(\alpha^e) \leq ord(\alpha^d)$. To these roots α^e the test (5.32) may be applied to determine their occurrence in the minimal polynomial $m_{\tilde{z}}(X)$ of the product sequence $\tilde{z}$. For a kth-order product everything repeats with roots α^e, whose exponent e has binary weight $W_2(e) = km$ and whose order is smaller than the order of α^d. But note that we do not have access to elements of $GF(2^L)$ whose order is larger than the order of α^d, in particular we do not have access to $\phi(2^L-1)$ primitive elements, where ϕ denotes the Euler totient function. Thus from the cryptographic viewpoint it might be a waste to start with a nonprimitive but irreducible minimal polynomial $m_{\tilde{s}}(X)$ since much of the linear complexity potential gets lost. On the other hand, when the order (or period) of $m_{\tilde{s}}(X)$ is large enough, this waste does not count and the cryptanalyst may well be confused by the different possible cycles, since he has usually been confronted only with PN-sequences.

Proposition 5.5 told us only how we could test the minimal polynomial of the product sequence for the presence of all roots α^e whose exponents have binary weight $W_2(e)$ equal to the order of the product. But knowing the test condition (5.32) enables us to find classes of products or functions which guarantee that the determinant in (5.32) is not zero. This permits us to establish a lower-bound on the linear complexity of the product sequence $\tilde{z}$.

Corollary 5.6. Product of equidistant phases

When $\tilde{s}^t$, $\tilde{s}^{t+\delta}$, ..., $\tilde{s}^{t+(k-1)\delta}$, $1 \leq k \leq L$, are equidistant phases of the same PN-sequence $\tilde{s}$ and $\gcd(\delta, 2^L-1)=1$, then all $\binom{L}{k}$ distinct elements $\alpha^e \in GF(2^L)$ whose exponents e have binary weight $W_2(e) = k$ are guaranteed to be roots of the minimal polynomial $m_{\tilde{z}}(X)$ of the product $\tilde{z}$ of all k distinct phases. Thus, the linear complexity of the product sequence $\tilde{z}$ is lowerbound by

$$\Lambda(\tilde{z}) \geq \binom{L}{k} \tag{5.41}$$

Proof: Assume without loss of essential generality that t, t+δ, ..., t+(k-1)δ = 0,δ,...,(k-1)δ, that is

$$\tilde{z} = \tilde{s}\ \tilde{s}^{\delta} \ \ldots\ \tilde{s}^{(k-1)\delta} \ . \tag{5.42}$$

Then the coefficient determinant (5.32) becomes a Vandermonde determinant ΔV whose value is given by

$$A_e = \Delta V(\alpha^{\delta 2^{e_1}}, \ \ldots, \ \alpha^{\delta 2^{e_k}}) = \prod_{n=2}^{k} \prod_{j=1}^{n-1} (\alpha^{\delta 2^{e_n}} - \alpha^{\delta 2^{e_j}}) \tag{5.43}$$

The Vandermonde determinant is zero if and only if $\delta 2^{e_n} = \delta 2^{e_j}$, for at least one pair $n \neq j$. But this cannot happen since e has a unique decomposition $2^{e_1} + .. + 2^{e_k}$ with $0 \leq e_1 < e_2 < \ldots < e_k < L$, and since, when $\gcd(\delta, 2^L-1) = 1$, multiplication by δ, results in a permutation of all the elements in the ring of integers modulo 2^L-1. Therefore all the $\binom{L}{k}$ distinct elements $\alpha^e \in GF(2^L)$ are roots of $m_{\tilde{z}}(X)$ and thus contribute to the linear complexity of $\tilde{z}$.
(Note that this result does not depend on the particular primitive minimal polynomial $m_{\tilde{s}}(X)$).

In the original version of this corollary (Ruep 84), the bound (5.41) was proved for adjacent phases of $\tilde{s}$. While analyzing a special structure of running-key generator, Bernasconi and Günther independently discovered the same bound (Bern 85) and generalized it to encompass the case of equidistant phases.

It follows from Corollary 5.6 that the designer of a nonlinear feedforward generator of the type considered in this section (see Fig. 5.3) could take an arbitrary nonlinear function f of order k-1 (or less) and apply it to the stages of a primitive LFSR. Then he could choose a kth order product of successive stages of the primitive LFSR and add it to the output of f. The linear complexity of the produced sequence $\tilde{z}$ is guaranteed to be at least $\binom{L}{k}$ (which can be a very large number for practical parameters such as L = 100 and k = 50). But a word of caution must be added: at this point the lowerbound (5.41) is based completely on a single high-order product, whose influence on the produced sequence is very infrequent. But our possibilities to frustrate the cryptanalyst are not yet exhausted. A cryptographically more interesting class of provably complex functions is the subject of the next Corollary.

Corollary 5.7. Linear combinations of products

Let $\tilde{s}^t\tilde{s}^{t+\delta} \ldots \tilde{s}^{t+(k-1)\delta}$ be a kth order product of k, k < L, equidistant phases of a m-sequence $\tilde{s}$, where $\gcd(\delta, 2^L-1) = 1$. Let $\tilde{z}$ be produced by any nonzero linear combination of N consecutive such products, that is

$$\tilde{z} = \sum_{i=0}^{N-1} c_i \tilde{s}^i \tilde{s}^{i+\delta} \ldots \tilde{s}^{i+(k-1)\delta} \qquad (5.44)$$

where N is positive and not all c_i are zero. Then the linear complexity of $\tilde{z}$ is lowerbounded by

$$\Lambda(\tilde{z}) \geq \binom{L}{k} - (N-1) \qquad (5.45)$$

Proof: A single kth-order product of the form $\tilde{z}^{(i)} = \tilde{s}^i \tilde{s}^{i+\delta} \ldots \tilde{s}^{i+(k-1)\delta}$, with δ such that $\gcd(\delta, 2^L-1)=1$, was shown in Corollary 5.6 to result in a nonzero Vandermonde determinant as coefficient for each root α^e whose exponent has binary weight $W_2(e) = k$. Analogously to (5.36) we can write for the jth bit of the ith single kth-order product sequence $\tilde{z}^{(i)}$,

$$z_j^{(i)} = y_j^{(i)} + \sum_{\{e:W_2(e)=k\}} A_e^{(i)} \alpha^{ej} \tag{5.46}$$

where $y_j^{(i)}$ accounts for the effects of the roots where exponents have binary weight smaller than k. Let ΔV denote the Vandermonde determinant as defined in (5.43) and note that $A_e^{(0)} = A_e = \Delta V$. Moreover $A_e^{(i)} = \alpha^{ie_m} \Delta V$, since in every row m of the determinant defined by $A_e^{(i)}$ a factor of $\alpha^{i2^{e_m}}$ can be pulled out. Therefore it follows that

$$z_j = y_j + \sum_{i=0}^{N-1} c_i \sum_{\{e:W_2(e)=k\}} \alpha^{ie} \Delta V \alpha^{ej} \tag{5.47}$$

where y_j accounts for the accumulated effects of roots with exponents of binary weight less than k. Rearranging sums in (5.47) results in

$$z_j = y_j + \sum_{\{e:W_2(e)=k\}} \alpha^{ej} \Delta V \sum_{i=0}^{N-1} c_i \alpha^{ie} \tag{5.48}$$

Since the Vandermonde determinant does not vanish we conclude that α^e is a root of the minimal polynomial of $\tilde{z}$ (i.e. it contributes to the linear complexity of $\tilde{z}$) if and only if

$$c_0 + c_1\alpha^e + \ldots + c_{N-1}(\alpha^e)^{N-1} \neq 0$$

that is, if and only if α^e is not a root of the polynomial $c_0 + c_1X + c_2X^2 + \ldots + c_{N-1}X^{N-1}$. But this polynomial can have at most N-1 roots.

Note that in the case of $k < L$, $N \leq L$, and L prime, none of the roots under consideration can vanish, since they cannot possibly be root of a polynomial of degree less than L, and hence the slightly tighter bound (5.41) is seem to hold in this case.

The foregoing analysis also implies that adding an arbitrary function f' of linearly independent phases of $\tilde{s}$ does not affect the bound (5.45) as long as the nonlinear order of f' is smaller than k. We finally arrive at the following guideline: choose the nonlinar filter function f you want to apply to a maximum-length LFSR with L stages as

$$\tilde{z} = \sum_{i=0}^{N-1} c_i \tilde{s}^i \tilde{s}^{i+\delta} \cdots \tilde{s}^{i+(k-1)\delta} + f'(\tilde{s}^0, \tilde{s}^1, \ldots, \tilde{s}^{L-1}) \qquad (5.49)$$

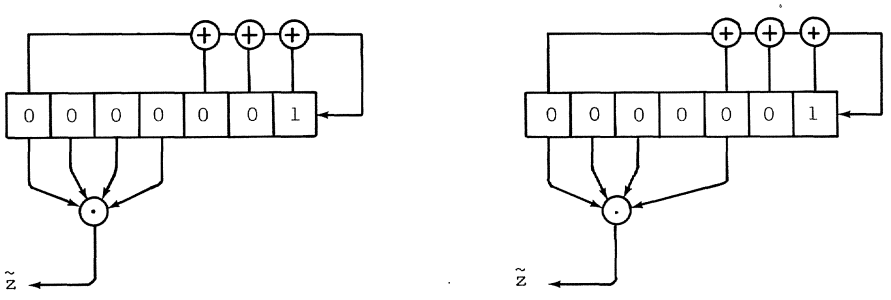
with $\gcd(\delta, 2^L-1)=1$ and $\mathrm{ord}(f') < k$, then the linaer complexity of the resulting output sequence is at least $\binom{L}{k} - (N-1)$. In terms of realization the first part of (5.49) does not look particularly convenient, since it may involve widely spaced phases of $\tilde{s}$. But note, we can always express any function of any number of phases of $\tilde{s}$ by an equivalent function of L consecutive phases. Interestingly enough, this even means that the linear combination of products (as in (5.44)), when reduced to L consecutive phases, will in general exhibit also products of any order smaller than k.

We want to remark that all the derived results hold as well for GF(q)-sequences as long as GF(q) has characteristic 2. Comparing the above bound with the Kumar and Scholz bound for bent sequences (Kuma 83) shows that for L = 24 (which is the largest value in their table) the linear complexity of bent sequences is bounded from below by 76'864 whereas the linear complexity of the above class of sequences is lower bounded by $2{,}7 \cdot 10^6$ (when k is chosen to be 12).

In the preceding analysis, we separated those elements $\alpha^e \in GF(2^L)$ whose exponents had binary weight $W_2(e)$ equal to the order k of the product from the remaining elements $\alpha^e \in GF(2^L)$ whose exponents had $W_2(e) < k$. In the ring of integers modulo (2^L-1) an element j can be written uniquely as a sum of k not necessarily distinct powers of 2 if and only if $W_2(j) = k$ (of course the powers are distinct if $W_2(j) = k$). For instance if k = 3 and L = 4 then the element $7 = 2^2 + 2^1 + 2^0$ is uniquely representable since $W_2(7) = 3$; but $5 = 2^1 + 2^1 + 2^0 \equiv 2^3 + 2^3 + 2^0 \pmod{15}$ has 2 representations since $W_2(5) = 2 < k = 3$. This unique representation of the exponents e with $W_2(e) = k$ is in fact the fundamental reason for being able to analyze their presence or absence in the minimal polynomial $m_{\tilde{z}}(X)$ of the product se-

quence $\tilde{z}$. By induction on k it is clear that also all elements α^e with $W_2(e) < k$ are possible candidates for being roots of $m_{\tilde{z}}(X)$. Unfortunately, since their exponents e are not uniquely representable by powers of 2, there is no equivalent analysis guaranteeing their absence or presence in $m_{\tilde{z}}(X)$.

We want to illustrate that only slight modifications in the nonlinear filter function may cause the linear complexity of the produced sequence to severely degenerate. The following example is taken from (Link 84).



(a) $\Lambda(\tilde{z}) = 98$ (b) $\Lambda(\tilde{z}) = 63$

Fig. 5.11. The LFSR $<7, D^7+D^3+D^2+D+1>$ driving two distinct 4th order products

The single 4th-order product $f = x_1x_2x_3x_4$, when applied to the stages of the primitive LFSR of Fig. 5.11, produces a sequence $\tilde{z}$ whose linear complexity 98 is maximum (it satisfies the upperbound (5.30) with equality). But if we connect stage 5 of the same LFSR instead of stage 4 into the 4th-order product, i.e. $f = x_1x_2x_3x_5$ the produced sequence will exhibit heavy degeneracies and have linear complexity only 63. In the example we could even have generated a sequence z whose linear complexity is 63 with the function $f = x_1x_2x_3$ whose order is 3.

In all the preceding analysis we have seen that the case of a nonlinearly filtered LFSR is extremely difficult to handle mathematically. Heavy degeneracies in the linear complexity of the produced sequences may occur and seem almost unpredictable. This "unpredictability" of the resulting linear complexity may have caused many designers of pseudo-random sequence generators to refuse to use the discussed type of nonlinear generator. To know that the linear com-

plexity of the produced sequence is at least $\binom{L}{k}$ for a kth-order function might be a sufficient level of security for some cryptographers to use the nonlinear state-filter-type of generator. Instead of a deterministic viewpoint let us now take a probabilistic viewpoint. The best we then can hope for is to find classes of functions which produce nondegenerate sequences with probability arbitrarily close to one. To investigate this problem let us assume that a randomly selected kth-order nonlinear function f is applied to the stages of a primitive LFSR of length L. Define

$$L_k = \sum_{i=1}^{k} \binom{L}{i} \tag{5.50}$$

to be the maximum linear complexity reachable with the kth-order filter function f. L_k also corresponds to the number of coefficients that must be selected to completely specify f. Each of these coefficients is randomly set to 0 or 1 with equal probability. The formal power series Z(D) of any sequence produceable by the linear equivalent of length L_k can be expressed as

$$Z(D) = \frac{P(D)}{C(D)} \quad \text{with} \quad \begin{array}{l} \deg\,[P(D)] < L_k - 1. \\ \deg\,[C(D)] = L_k \end{array} \tag{5.51}$$

The irreducible factors of C(D) all have degree L or a divisor of L, since the roots contributing to S(D) all lie in the Galois field $GF(2^L)$. Now expand (5.51) into partial fractions,

$$Z(D) = \frac{P_1(D)}{C_1(D)} + \frac{P_2(D)}{C_2(D)} + \ldots + \frac{P_r(D)}{C_r(D)} \tag{5.52}$$

where $C_i(D)$ are the irreducible factors of C(D), and $\deg[P_i(D)] < \deg[C_i(D)]$, i = 1, .., r.

We immediately see that the degree of the linear equivalent will only be maximum if all the numerators $P_i(D)$, i=1,..,r, are different from zero.

Consequently, we can calculate the number of nondegenerate sequences as the total number of non-zero choices for all the polynomials $P_i(D)$. This approach allows us also to give an accurate estimate of

the fraction of nondegenerate sequences produced by a nonlinearly filtered LFSR of prime length (the case that the cryptographer is always the most interested in).

Proposition 5.8.:

If a nonlinear function of order k, k < L, is applied to the stages of an LFSR with primitive connection polynomial of prime degree L, and

$$L_k = \sum_{i=1}^{k} \binom{L}{i}$$

then the fraction P_n of nondegenerate sequences produced by the nonlinearly filtered LFSR is lowerbound by

$$P_n \simeq e^{-\left(\frac{L_m}{L \cdot 2^L}\right)} > e^{-\frac{1}{L}} . \qquad (5.53)$$

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

Proof: If L is prime, then every element except 1 of $GF(2^L)$ has associated a minimum polynomial of degree exactly L. Consequently, all the irreducible factors $C_i(D)$ of the linear equivalent C(D) have degree L. There are L_k/L such distinct factors. For every factor there are 2^L-1 different ways in which the corresponding numerator polynomial $P_i(D)$ can be made nonzero.

Thus the fraction P_n of nondegenerate sequences is

$$P_n = \left[\frac{2^L-1}{2^L}\right]^{L_k/L} = \left[1 - \frac{1}{2^L}\right]^{L_k/L} = \left[1 - \frac{1}{2^L}\right]^{2^L \cdot \frac{L_k}{L \cdot 2^L}} . \qquad (5.54)$$

For large n, $(1-1/n)^n \to e^{-1}$. Applying this approximation and noting that L_k cannot be greater than 2^L-1 yields

$$P_n \simeq e^{-\left(\frac{L_k}{L \cdot 2^L}\right)} > e^{-\frac{1}{L}} . \qquad (5.55)$$

The fraction of degenerate sequences is given by

$$1 - P_n < 1 - e^{-\frac{1}{L}} .$$

Letting L go to infinity shows that the fraction of degenerate sequences tends to zero.

Notice that P_n directly corresponds to the probability that a randomly chosen nonlinear function produces a nondegenerate sequence. As an example, consider an LFSR of length L = 53 with primitive connection polynomial. Suppose the nonlinear function is chosen randomly and has order about half the length, say k = 26. Then $L_k \simeq 2^{L-1}$ and the probability that the linear equivalent does not degenerate is about

$$e^{-\frac{1}{2L}} \simeq .9906 .$$

If the randomly chosen function has order close to L, then the probability of obtaining nondegenerate complexity is

$$e^{-\frac{1}{L}} \simeq .981$$

which is also the lowerbound on the fraction of nondegenerate sequences independently of the order k of the nonlinear function. To illustrate the rareness of a "dangerous" degeneracy, let us also calculate the fraction of nonlinear functions resulting in a degeneracy of just one factor of degree L or, in other words, whose associated linear equivalent has length L_k-L. There are L_k/L different ways how such a single degeneracy could happen. Thus the fraction of single degeneracy sequences is

$$\frac{L_k}{L} \cdot \frac{1}{2^L} \left(1 - \frac{1}{2^L}\right)^{\frac{L_k}{L} - 1} \simeq \frac{L_k}{L} \cdot \frac{1}{2^L} e^{-\left(\frac{L_k - L}{L \cdot 2^L}\right)}$$

$$\simeq \frac{L_k}{L} \cdot \frac{1}{2^L} e^{-\left(\frac{L_k}{L 2^L}\right)} \qquad (5.56)$$

Using the numbers of the above example for $L_k \simeq 2^{L-1}$ the fraction of nonlinear functions resulting in none or a single degeneracy is about

$$e^{-\frac{1}{2L}} + \frac{1}{2L} \cdot e^{-\frac{1}{2L}} = .999956 .$$

Or equivalently, the probability that a degeneracy of more than one factor in the linear equivalent occurs is as small as $4.4 \cdot 10^{-5}$.

Proposition 5.9 tells us that we can make the probability of a degenerate linear equivalent arbitrarily small by choosing L prime and sufficiently large. It is also obvious from the partial fraction expansion argument that almost all occuring degeneracies will be negligibly small.

Instead of guaranteeing maximum linear complexity, the designer of a nonlinear state-filter generator may define the desired security level in terms of the probability that a degeneracy of larger than some tolerated amount results in the linear equivalent. Independent of how small he chooses this probability he will always find L and k satisfying the security level. But the most important result is in fact that no longer specific classes of functions have to be considered. Any truly randomly selected function will be sufficient. This means that there is no restriction at all on the set of allowed functions, which is intuitively pleasing both from the realization viewpoint (no special care has to be taken to exclude "insecure" functions) as from the security viewpoint (the cryptanalyst may be more frustrated by the size of the keyspace than he is delighted by the small probability of taking advantage of a considerable degeneracy).

5.2 Nonlinear Operations on Sequences with Distinct Minimal Polynomials

Consider the general type of running-key generator depicted in Fig. 5.12, where the output sequences of M LFSRs are taken as arguments of a single nonlinear combining function F whose output then forms the running key z.

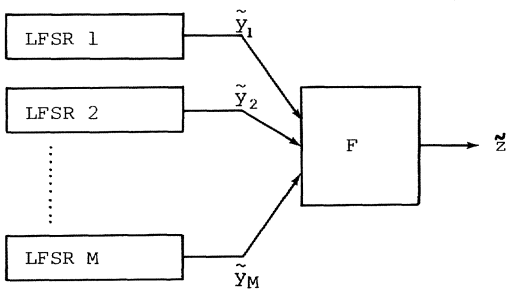


Fig. 5.12. A commonly used running-key generator structure

Considerable interest has been paid to the problem of controlling the linear complexity of sequences that result from nonlinear combinations, in particular products, of LFSR-sequences (Selm 66, Key 76, Herl 82). 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 (Zier 73, Herl 82). Consequently, whenever the linear complexity of a product sequence satisfies this upperbound with equality, one says that this product sequence attains maximum linear complexity. Our main interest is in the following question: what conditions (in particular, what easily verified conditions) have to be imposed on the LFSRs and/or the nonlinear function F such that the resulting key stream will exhibit maximum linear complexity? It has been known for a long time that the product of two GF(q)-sequences with irreducible minimal polynomials of degree m and n, respectively, attains maximum linear complexity mn, if m and n are relatively prime ((Selm 66), (Key 76); in fact Selmer provides a reference dated in 1881 where this result is stated without proof). As a practical consequence, most of the running key generators of the type depicted in Fig. 5.12 are built according to this condition; the driving LFSRs usually have primitive connection polynomials, and their lengths are chosen to be pairwise relatively prime. Only recently, Rueppel and Staffelbach (Ruep 86) were able to

show that neither irreducibility of the involved minimal polynomials nor relative primeness of their degrees is required to attain maximum linear complexity, thereby greatly enhancing the number of available design options (at the end of this section we will review some of their results). Note that the model of Fig. 5.12 does not exclude the case where the driving LFSRs are replaced by nonlinearly filtered LFSRs (see section 5.1): we could simply imagine their linear equivalents being the driving LFSRs. We will only be concerned with memoryless mappings F; examples of such memoryless combiners of distinct LFSRs can be found in (Geff 73) and (Brüe 84). Finally, whenever it is possible to switch from sequences over GF(2) to sequences over GF(q) without further complication, we will do so. But our main emphasis will be on binary sequences.

Often designers of nonlinear combiner generators require, in addition to the pairwise primality of the lengths of the driving LFSRs, that also the periods be relatively prime in order to obtain maximum running key period. We will shortly show that this additional restriction is essentially superfluous; in fact, in the binary case both conditions are equivalent.

Lemma 5.9. Period (Order) Criterion

For any integers $q > 1$, $m > 0$, $n > 0$

$$\gcd(q^m-1, q^n-1) = q^{\gcd(m,n)} - 1$$

Proof: (Knut 81, P. 252)

Let $m > n$ and apply the division algorithm to obtain $m = pn + r$. Then

$$q^m - 1 = q^{pn+r} - 1 = q^{pn+r} - q^r + q^r - 1 = q^r(q^{pn} - 1) + q^r - 1$$

Therefore

$$q^m - 1 \bmod q^n - 1 = q^{m \bmod n} - 1$$

where a mod b denotes the remainder when a is divided by b. The Lemma follows now directly from Euclid's algorithm.

Suppose two sequences $\tilde{r}$ and $\tilde{s}$ of period T_1 and T_2 are being added or multiplied termwise. Clearly the resulting sequence $\tilde{z}$ (whose period we denote by T) is periodic in T_1T_2. Therefore the least period T of $\tilde{z}$ can be written as $T = d_1d_2$ where d_1 divides T_1 and d_2 divides T_2. It follows that $\tilde{z}$ must be periodic in d_1T_2 which implies that T_1 divides d_1T_2. Finally, dividing by the $\gcd(T_1,T_2)$ tells us that $T_1/\gcd(T_1,T_2)$ is a divisor of d_1. Similarly, one may find that $T_2/\gcd(T_1,T_2)$ is a divisor of d_2. Hence, without having made any assumption about the minimal polynomials of the sequences to be combined we have shown that

$$\frac{T_1T_2}{(\gcd(T_1,T_2))^2} \leqslant T \leqslant T_1T_2$$

This may be combined with Lemma 5.9 to produce the following corollary.

Corollary 5.10. to Period Criterion

Let $\tilde{r}$ and $\tilde{s}$ be nonzero sequences over GF(q) with irreducible minimal polynomials $m_{\tilde{r}}(X)$ and $m_{\tilde{s}}(X)$ ε GF(q)[X] whose degrees m and n are relatively prime. Let T_1 and T_2 be the orders of $m_{\tilde{r}}(X)$ and $m_{\tilde{s}}(X)$.

If the sequences are added or multiplied termwise, then the period T of the resulting sequence $\tilde{z}$ is lowerbound by

$$T \geqslant \frac{T_1 . T_2}{(q-1)^2} . \qquad (5.57)$$

In particular, when q = 2

$$T = T1 \cdot T2 . \qquad (5.58)$$

The bound (5.57) may be tightened to $T \geqslant T_1 T_2/(q-1)$ when only termwise addition of the sequences is considered. Since, by assumption, $m_{\tilde{r}}(X)$ and $m_{\tilde{s}}(X)$ are relatively prime, the minimal polynomial of the sumsequence $\tilde{z}$ is equal to the product of $m_{\tilde{r}}(X)$ and $m_{\tilde{s}}(X)$. Hence, the order of $m_{\tilde{z}}(X)$ is equal to the least common multiple of T_1 and T_2, which in turn is at least $T_1T_2/(q-1)$.

Corollary 5.10 may easily be extended to the case where $m_{\tilde{r}}(X)$ and $m_{\tilde{s}}(X)$ comprise irreducible factors of degree m or a divisor of m and of degree n or a divisor of n, respectively. We see in fact that in almost all cryptographically interesting cases a simple condition on the degrees of the involved minimal polynomials (or their factors) is sufficient to guarantee the desired product of periods.

One of our basic tools in the analysis of the effects induced by the nonlinear combiner will again be the trace operator. It is a very handy instrument for the description of sequences with irreducible minimal polynomials. But before we start dealing with sequences, we will present a trace identity which is of some independent interest.

Lemma 5.11. Trace Identity

If α and β are elements of $GF(q^m)$ and $GF(q^n)$, respectively, where gcd(m,n) = 1, then

$$T_m(\alpha)\, T_n(\beta) = T_{mn}(\alpha\beta) \ . \tag{5.59}$$

Proof: Note that $GF(q^m)$ and $GF(q^n)$ are both subfields of $GF(q^{mn})$ so that $\alpha\beta$ is well-defined in $GF(q^{mn})$. Then

$$T_{mn}(\alpha\beta) = \sum_{i=0}^{mn-1} \alpha^{q^i}\, \beta^{q^i} \ . \tag{5.60}$$

Since $\alpha \in GF(q^m)$ and $\beta \in GF(q^n)$, it holds that

$$\alpha^{q^i} = \alpha^{q^{i \bmod m}} \tag{5.61a}$$

$$\beta^{q^i} = \beta^{q^{i \bmod n}} \tag{5.61b}$$

where "i mod M" denotes the remainder when i is divided by M. Because gcd(m,n) = 1, the Chinese remainder theorem implies that (i mod m, i mod n) takes on each pair (j,k) with $0 \leq j < m$ and $0 \leq k < n$ exactly once as i ranges from 0 to nm-1. Thus (5.60) and (5.61) imply

$$T_{mn}(\alpha\beta) = \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} \alpha^{q^j} \beta^{q^k}$$

$$= \sum_{j=0}^{m-1} \alpha^{q^j} \sum_{k=0}^{n-1} \beta^{q^k} \qquad (5.62)$$

which we now recognize to be the desired identity.

Suppose now α and β are quintessential elements of $GF(q^m)$ and $GF(q^n)$, respectively. Then the root algebra proposition 3.1 assures that $\alpha\beta$ is a quintessential element of $GF(q^{mn})$ when gcd(m,n) = 1. The same result could have been obtained from the above trace identity and the fact that the period of the conjugate root sequence of $\alpha\beta$ is the least positive integer t such that $(\alpha\beta)^{q^t} = \alpha\beta$, (in fact base the root algebra proposition and Lemma 5.11 on the same properties of extension fields). If α and β are quintessential elements of their fields, they have associated irreducible minimum polynomials over GF(q) of degree m and n respectively. Consequently, the minimum polynomial of $\alpha\beta$ over GF(q) is irreducible of degree mn. This result was also proved in (Key 76). Now let us extend the argument to sequences. Assume the sequences $\tilde{r}$ and $\tilde{s}$ to have irreducible minimal polynomials of degree m and n, respectively. Then the jth digit of the corresponding sequences may be expressed as follows using the trace mapping

$$r_j = T_m(A\alpha^j) \quad A \neq 0 \text{ in } GF(q^m) \qquad (5.63a)$$

$$s_j = T_n(B\beta^j) \quad B \neq 0 \text{ in } GF(q^n) \qquad (5.63b)$$

where A and B define the starting phases of the corresponding sequences. Note that $A\alpha^j$ and $B\beta^j$ are well-defined elements of $GF(q^m)$ and $GF(q^n)$, respectively. Therefore, we may directly apply the trace identity (5.59) to the sequence digits as defined by (5.63). Then, if gcd(m,n) = 1,

$$z_j = r_j s_j = T_m(A\alpha^j)\, T_n(B\beta^j) = T_{mn}(AB(\alpha\beta)^j) \; . \qquad (5.64)$$

Since $\alpha\beta$ is a quintessential element of $GF(q^{mn})$ and is also a root of the minimal polynomial of the product sequence, one has the following result.

Lemma 5.12. Simple product

Let $\tilde{r}$ and $\tilde{s}$ be sequences with irreducible minimal polynomials $m_{\tilde{r}}(X)$ and $m_{\tilde{s}}(X)$ over GF(q) of degree m and n, respectively. If gcd(m,n) = 1, then the product sequence $\tilde{z} = \tilde{r}\tilde{s}$ has irreducible minimal polynomial $m_{\tilde{z}}(X)$ over GF(q) of degree mn.

Moreover, Corollary 5.10 also assures that the period of the product sequence $\tilde{z}$ is at least the product of the periods of $\tilde{r}$ and $\tilde{s}$ divided by $(q-1)^2$.

Lemma 5.12 can also be derived using Zierler and Mill's (Zier 73) or Herlestam's (Herl 82) approach. In what follows, we want to mention some subtleties involved with the trace description of sequences. In the proof of Lemma 5.12, we departed from the minimal polynomials of the considered sequences. Let us now directly start from the trace description (5.64) of the product. Note that (5.64) also holds when α or β are elements of proper subfields $GF(q^{m'}) \subset GF(q^m)$ or $GF(q^{n'}) \subset GF(q^n)$, respectively. According to the properties of the roots α, β and the coefficients A, B we may distinguish 3 cases:

(1) $\tilde{z} = \tilde{r}\tilde{s}$ has irreducible minimal polynomial $m_{\tilde{z}}(X)$ over GF(q) of degree mn if and only if α and β are quintessential elements and A and B are nonzero elements of their corresponding fields. This is the case investigated in Lemma 5.13. Note that if m and n are distinct primes, it is automatically guaranteed that α and β are quintessential elements.

(2) $\tilde{z} = \tilde{r}\tilde{s}$ has irreducible minimal polynomial of degree m'n', where m' divides m and n' divides n, when α and β lie in the subfields $GF(q^{m'})$ and $GF(q^{n'})$, respectively, provided neither A belongs to the class $\mathcal{A}_0$ (the set of A's causing $\tilde{r}$ to be the allzero sequence, compare (3.24)) nor B belongs to $\mathcal{B}_0$ (the set of B's causing $\tilde{s} = \tilde{0}$).

(3) $\tilde{z} = \tilde{r}\tilde{s}$ is the allzero sequence (despite nonzero coefficient A ε $GF(q^m)$ and B ε $GF(q^n)$) whenever α and β lie in the subfields $GF(q^{m'})$ and $GF(q^{n'})$, respectively, and A belongs to the class $\mathcal{A}_0$ or B belongs to $\mathcal{B}_0$.
Note that the minimal polynomial of $\alpha\beta$ nevertheless is irreducible of degree m'n'.

In general we may say that whenever A and B do not belong to the zero classes $\mathcal{A}_0$ and $\mathcal{B}_0$, respectively, then $\tilde{z} = \tilde{r}\tilde{s}$ has minimal polynomial equal to the minimum polynomial of $\alpha\beta$. Let us make an example to illustrate the 3 cases which may occur.

Suppose γ is a primitive element of $GF(2^4)$ and has minimum polynomial $m_\gamma(X) = X^4 + X + 1$. So we may represent any element α in $GF(2^4)$ as a power of γ. Suppose further that β is a primitive element of $GF(2^3)$ and has minimum polynomial $m_\beta(X) = X^3 + X + 1$. Then

$$z_j = r_j s_j = T_4(A\alpha^j)\, T_3(B\beta^j) \tag{5.65}$$

has (1) Irreducible minimal polynomial
$m_{\tilde{z}}(X) = X^{12} + X^9 + X^5 + X^4 + X^3 + X + 1$
if $\alpha \in \{\gamma, \gamma^2, \gamma^4, \gamma^8\}$ and A and B are nonzero elements of their fields.

(2) Irreducible minimal polynomial
$m_{\tilde{z}}(X) = X^6 + X^4 + X^2 + X + 1$ if $\alpha \in \{\gamma^5, \gamma^{10}\}$ and $A \notin \{\gamma^5, \gamma^{10}, 1, 0\}$.

(3) $\tilde{z}$ is the allzero sequence if $\alpha \in \{\gamma^5, \gamma^{10}\}$ and $A \in \{\gamma^5, \gamma^{10}, 1, 0\}$.

This example illustrates that care has to be taken when one is working with the trace description. Simply selecting α from $GF(q^m) - GF(q)$ and β from $GF(q^n) - GF(q)$ and associating to them in the trace description any nonzero elements A and B from the corresponding fields does not in general guarantee that the allzero sequence is avoided. But when we add in the constraint that m and n be distinct primes, then $\alpha\beta$ will be a quintessential element of $GF(q^{mn})$ and any nonzero coefficients from the corresponding fields will guarantee that $\tilde{z}$ has maximum linear complexity mn.

In the preparatory work done so far we considered only sequences with irreducible polynomials. Let us now generalize the theory in order to cover also reducible minimal polynomials. Here one case is of special interest, namely the case where the minimal polynomial of the sequence under consideration factors into distinct irreducible polynomials of the same degree or a divisor of this degree. This precisely happens when the considered sequence is generated by a

nonlinear state-filter generator (see section 5.1). Our intention is to cover this cryptographically interesting situation by the following basic result.

Proposition 5.13. Multiplication Rule for Sequences with Distinct Minimal Polynomials

Let $\tilde{r}$ be a sequence with minimal polynomial $m_{\tilde{r}}(X)$ over GF(q) of degree M, whose roots α are all simple and lie in $GF(q^m)$ - GF(q) (which is equivalent to stating that $m_{\tilde{r}}(X)$ divides $x^{q^m-1}-1$ and contains no linear factors).

Similarly, let $\tilde{s}$ be a sequence with minimal polynomial $m_{\tilde{s}}(X)$ over GF(q) of degree N, whose roots β are all simple and lie in $GF(q^n)$ - GF(q) (which is equivalent to stating that $m_{\tilde{s}}(X)$ divides $x^{q^n-1}-1$ and contains no linear factors).

If gcd(m,n) = 1 (and provided $\alpha'\alpha^{-1} \notin GF(q)$ for any distinct roots α and α' of $m_{\tilde{r}}(X)$ and $\beta'\beta^{-1} \notin GF(q)$ for any distinct roots β and β' of $m_{\tilde{s}}(X)$) then the minimal polynomial $m_{\tilde{z}}(X)$ of the product sequence $\tilde{z} = \tilde{r}\tilde{s}$ has MN simple roots γ in $GF(q^{mn})$ - $\{GF(q^m) \cup GF(q^n)\}$.

Proof: Let R(D) and S(D) be the formal power series associated to $\tilde{r}$ and $\tilde{s}$, respectively. Expand R(D) and S(D) into partial fractions according to the irreducible factors of the connection polynomial of the associated linear equivalent (note that the minimal polynomial of a periodic sequence and the connection polynomial of the associated linear equivalent are reciprocal to each other).

$$R(D) = \sum_i \frac{P_i(D)}{F_i(D)} \quad \deg[P_i(D)] < m_i = \deg[F_i(D)] \tag{5.66a}$$

$$R(D) = \sum_k \frac{Q_k(D)}{G_k(D)} \quad \deg[Q_k(D)] < n_k = \deg[G_k(D)] \tag{5.66b}$$

where $m_i = \deg[F_i(D)]$ is either equal to m or divides m for all i, since all the roots α of $m_{\tilde{r}}(X)$ are assumed to be in $GF(q^m)$. Similarly $n_k = \deg[G_k(D)]$ is either equal to n or divides n for all k, since all the roots β are assumed to lie in $GF(q^n)$. Let $\tilde{r}_i$ ($\tilde{s}_k$) be the sequence associated to the ith (kth) partial fraction $R_i(D) = P_i(D)/F_i(D)$ ($S_k(D) = Q_k(D)/G_k(D)$). We may then write

$$\tilde{r} = \sum_i \tilde{r}_i \tag{5.67a}$$

$$\tilde{s} = \sum_k \tilde{s}_k \tag{5.67b}$$

and obtain for the product sequence $\tilde{z}$

$$\tilde{z} = \tilde{r}\,\tilde{s} = \sum_i \sum_k \tilde{r}_i \tilde{s}_k = \sum_i \sum_k \tilde{z}_{ik} \,. \tag{5.68}$$

Since $\gcd(m,n) = 1$ implies $\gcd(m_i,n_k) = 1$ for all $m_i | m$ and all $n_k | n$, Lemma 5.12 assures that $\tilde{z}_{ik}$ has irreducible minimal polynomial of degree $m_i n_k$. According to (5.64) we may express the jth bit of $\tilde{z}_{ik}$ using the trace mapping

$$(\tilde{z}_{ik})_j = T_{m_i n_k}(A_i B_k (\alpha_i \beta_k)^j) \tag{5.69}$$

for suitable coefficients A_i in $GF(q^m)$ and B_k in $GF(q^n)$. Combining (5.68) and (5.69), we conclude that the degree of $m_{\tilde{z}}(X)$ is $\sum\sum_{ik} m_i n_k = MN$ if and only if all root products $\alpha_i \beta_k$ are different and thus have distinct irreducible minimum polynomials. Now suppose for $\alpha \neq \alpha'$ in $GF(q^m)$ and $\beta \neq \beta'$ in $GF(q^n)$ that $\alpha\beta = \alpha'\beta'$ which implies that $\alpha(\alpha')^{-1} = \beta'\beta^{-1}$. But since $GF(q^m) \cap GF(q^n) = GF(q)$, $\alpha(\alpha')^{-1}$ must lie in the groundfield GF(q). Therefore $\alpha\beta = \alpha'\beta'$ if and only if $\alpha = c\alpha'$ and $\beta' = c\beta$ with c being a nonzero scalar from GF(q), which proves the proposition.

Note that the extra added condition that $\alpha'\alpha^{-1}$ and $\beta'\beta^{-1}$ must not lie in the groundfield GF(q) for any $\alpha \neq \alpha'$ in GF(q^m) - GF(q) and any $\beta \neq \beta'$ in GF(q^n) - GF(q) has no significance when we are working with sequences over GF(2). The only nonzero scalar in GF(2) is 1, so there are no distinct $\alpha \neq \alpha'$ such that $\alpha = 1\alpha'$. Let us first illustrate Proposition 5.14 by means of an example over GF(2).

Suppose

$$r_j = T_3(\alpha_1^j)$$

where α_1 is a root of $m_{\tilde{r}}(X) = X^3 + X^2 + 1$. Suppose further

$$s_j = T_4(\beta_1^j) + T_2(\beta_2^j)$$

where β_1 is a root of $m_{\tilde{s}_1}(X) = X^4 + X^3 + 1$ and
β_2 is a root of $m_{\tilde{s}_2}(X) = X^2 + X + 1$

Consequently, $m_{\tilde{s}}(X) = X^6 + X^3 + X^2 + X + 1$.

Then

$$z_1 = r_j s_j = T_{12}((\alpha_1\beta_1)^j) + T_6((\alpha_1\beta_2)^j)$$

Where $\alpha_1\beta_1$ is a root of $m_{\tilde{z}_{11}}(X) = X^{12} + X^{11} + X^9 + X^8 + X^7 + X^3 + 1$
and $\alpha_1\beta_2$ is a root of $m_{\tilde{z}_{12}}(X) = X^6 + X^5 + X^4 + X^2 + 1$

which results in

$$m_{\tilde{z}}(X) = X^{18} + X^{14} + X^{12} + X^{11} + X^{10} + X^9 + X^6 + X^4 + X^3 + X^2 + 1.$$

Fig. 5.13 illustrates the example in terms of the linear equivalents.

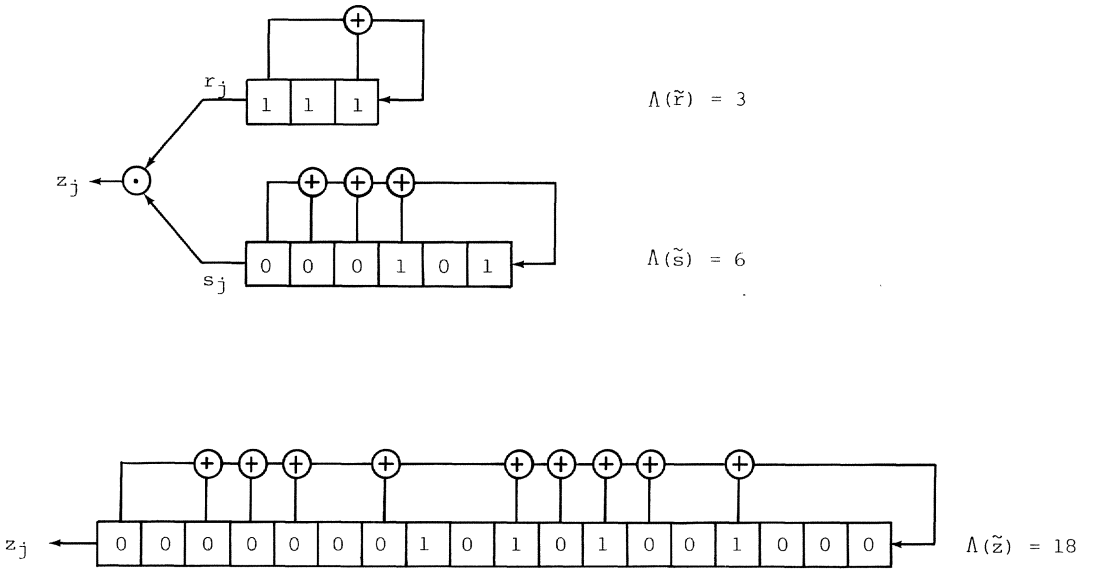


Fig. 5.13. Product of 2 sequences of GF(2) and associated linear equivalents

Proposition 5.13 tells us that the product sequence $\tilde{z}$ is guaranteed to attain maximum linear complexity $\Lambda(\tilde{z}) = \Lambda(\check{r})\Lambda(\check{s})$ if the roots of $m_{\check{r}}(X)$ and of $m_{\check{s}}(X)$ all lie in the true extension fields $GF(q^m) - GF(q)$ and $GF(q^n) - GF(q)$, and if $\gcd(m,n) = 1$. Note that this does not mean that the degrees of $m_{\check{r}}(X)$ and $m_{\check{s}}(X)$ are relatively prime (as the above example also illustrates).

The root multiple condition (no root may be a simple scalar multiple of another root in the minimal polynomial of the considered sequence) needs some further comment.

The best way to discuss this constraint is again by means of an example. Let us consider sequences over GF(3).

Suppose

$$r_j = T_2(\alpha_1^j) + T_2(\alpha_2^j)$$

where α_1 is a root of $m_{\tilde{r}_1}(X) = X^2 + X + 2$ and
α_2 is a root of $m_{\tilde{r}_2}(X) = X^2 + 2X + 2$.

Note that both $m_{\tilde{r}_1}(X)$ and $m_{\tilde{r}_2}(X)$ are primitive polynomials over GF(3) and that $m_{\tilde{r}}(X) = X^4 + 1$.

Suppose further

$$s_j = T_3(\beta_1^j) + T_3(\beta_2^j)$$

where β_1 is a root of $m_{\tilde{s}_1}(X) = X^3 + 2X + 1$ and
β_2 is a root of $m_{\tilde{s}_2}(X) = X^3 + 2X + 2$.

Note that $m_{\tilde{s}_1}(X)$ is primitive and that $m_{\tilde{s}}(X) = X^6 + X^4 + X^2 + 2$.

$$\begin{aligned} \text{Then} \quad z_j = r_j s_j &= T_6((\alpha_1\beta_1)^j) + T_6((\alpha_1\beta_2)^j) \\ &\quad + T_6((\alpha_2\beta_1)^j) + T_6((\alpha_2\beta_2)^j) \\ &= T_6(2(\alpha\ \beta)^j) + T_6(2(2\alpha\beta)^j) \end{aligned}$$

where $(\alpha\beta) = (\alpha_1\beta_1) = (\alpha_2\beta_2)$
is a root of $m_{\tilde{z}_{11}}(X) = m_{\tilde{z}_{22}}(X) = X^6 + 2X^3 + X^2 + X + 2$

and $(2\alpha\beta) = (\alpha_1\beta_2) = (\alpha_2\beta_1)$
is a root of $m_{\tilde{z}_{12}}(X) = m_{\tilde{z}_{21}}(X) = X^6 + X^3 + X^2 + 2X + 2$

which finally results in $m_{\tilde{z}}(X) = X^{12} + 2X^8 + 1$. Fig. 5.14 illustrates the example in terms of the associated linear equivalents.

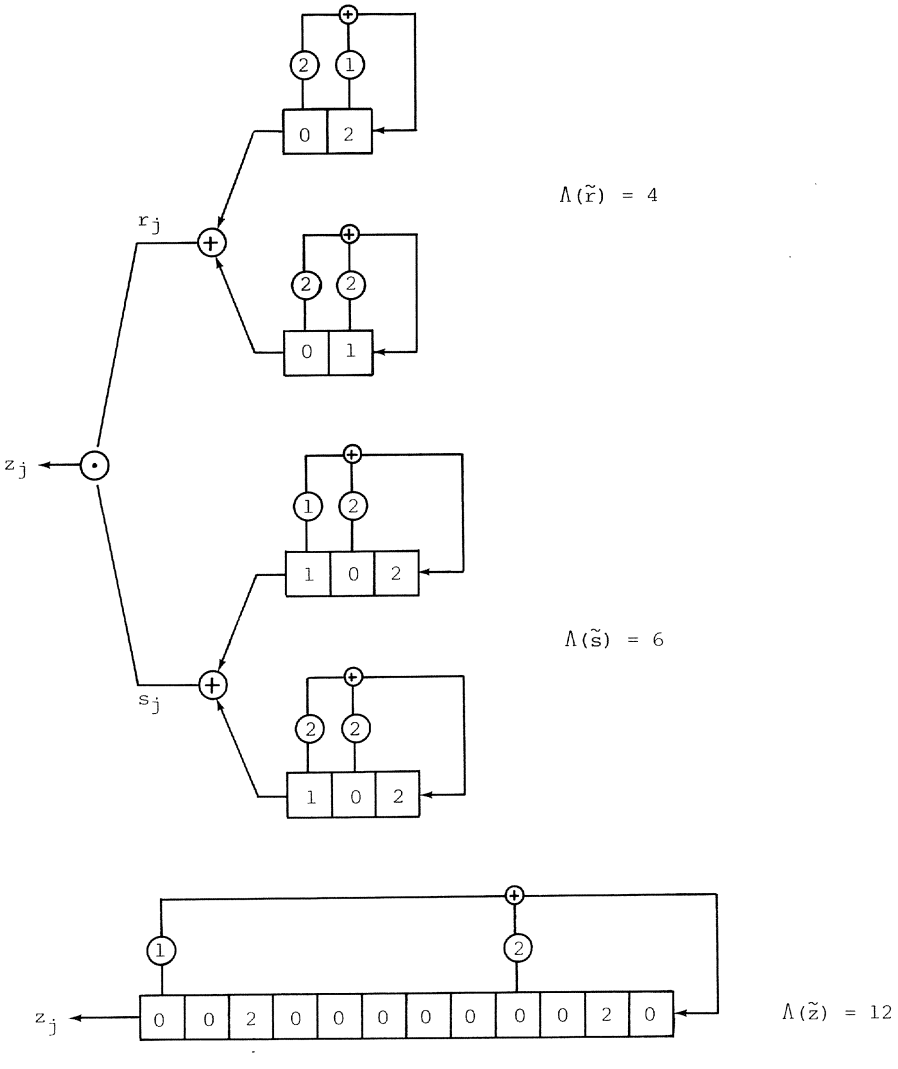


Fig. 5.14. Product of 2 sequences over GF(3) and associated linear equivalents.

Despite the fact that both factors of $m_{\tilde{r}}(X)$ and one factor of $m_{\tilde{s}}(X)$ are distinct and primitive polynomials over GF(3) and despite gcd(2,3) = 1, the linear complexity of the product sequence, $\Lambda(\tilde{z}) = 12 \neq \Lambda(\tilde{r})\Lambda(\tilde{s}) = 24$. Half of the attainable linear complexity is lost in the example, since the root multiple constraint has not been obeyed. For $\alpha_1\alpha_2^{-1} = \beta_1\beta_2^{-1} = 2$ which is a scalar from the groundfield GF(3).

With the framework provided in Proposition 5.13 it is now easy to generalize the theory to cover the case of a product of more than 2 sequences with distinct minimal polynomials.

Corollary 5.14. to the multiplication rule for sequences with distinct minimal polynomials

Let $\tilde{s}_i$, i = 1,.., N, be sequences over GF(q) with minimal polynomial $m_{\tilde{s}}(X)$ of degree M_i. Let $m_{\tilde{s}_i}(X)$ have only simple roots α_{in} from $GF(q^{m_i}) - GF(q)$, i = 1, .., N, n = 1, ..., M_i, none of which is a scalar multiple of another root of $m_{\tilde{s}_i}(X)$.

If $\gcd(m_i, m_j) = 1$ for all $i \neq j$, then

$$\tilde{z} = \prod_{i=1}^{N} \tilde{s}_i \qquad (5.70)$$

has minimal polynomial $m_{\tilde{z}}(X)$ of degree $M = \prod_{i=1}^{N} M_i$ whose roots are all simple and lie in $GF(q^m) - \{\bigcup_k GF(q^k)\}$ where $m = \prod_{i=1}^{N} m_i$ and k runs through all (N-1)st-order products of m_i's.

Proof: If $\tilde{y}_1 = \tilde{s}_1\tilde{s}_2$ then $m_{\tilde{y}_1}(X)$ has degree M_1M_2 and all its roots are simple and lie in $GF(q^{m_1m_2}) - \{GF(q^{m_1}) \cup GF(q^{m_2})\}$ according to Proposition 5.13. Since $\gcd(m_3, m_1m_2) = 1$, $\tilde{y}_2 = \tilde{y}_1\tilde{s}_3$, has minimal polynomial $m_{\tilde{y}_2}(X)$ of degree $(M_1M_2)M_3$, all of whose roots are simple and lie in $GF(q^{(m_1m_2)m_3}) - \{GF(q^{m_1m_2}) \cup GF(q^{m_1m_3}) \cup GF(q^{m_2m_3})\}$ again by Propostion 5.13. The Corollary now follows by induction.

Now that we have covered also a Nth-order product of sequences with distinct minimal polynomials (of the special type considered here) we are ready for the case the designer is most interested in. He wants to know what linear complexity the running key will exhibit when he applies an arbitrary mapping F to the N periodic sequences he has available for manipulation. In the following Corollary, we will slightly generalize the algebraic normal form of a function F in the sense that the coefficients of the product terms may take on values from the groundfield GF(q) (and not only from GF(2) as defined in switching theory).

Corollary 5.15. to the multiplication rule for sequences with distinct minimal polynomials

Let

$$F(x_1,\ldots,x_N) = a_0 + \Sigma a_i x_i + \Sigma a_{ij} x_i x_j + \ldots + a_{12..N} x_1 x_2 \cdot\cdot x_N$$
$$a_i, a_{ij}, \ldots \varepsilon GF(q) \qquad (5.71)$$

be a nonlinear function over GF(q). Let $\tilde{s}_i$, i = 1, ..., N, be sequences over GF(q) with minimal polynomial $m_{\tilde{s}_i}(X)$ of degree M_i, all of whose roots are simple from $GF(q^{m_i}) - GF(q)$ not being a scalar multiple of some other root of $m_{\tilde{s}_i}(X)$.

If $\gcd(m_i, m_j) = 1$, for all $i \neq j$, then

$$\tilde{z} = F(\tilde{s}_1, \ldots, \tilde{s}_N) \qquad (5.72)$$

has minimal polynomial $m_{\tilde{z}}(X)$ of degree

$$M = F'(M_1, \ldots, M_N) \tag{5.73}$$

where $F'(x_1,\ldots,x_N)$ is defined as in (5.71) with a_i', $a_{i'j}$, ... being 1 if a_i, a_{ij}, ... are nonzero, and zero otherwise and where F' is evaluated over the integers and not in GF(q) as is F to produce the keystream $\tilde{z}$. All the roots of $m_{\tilde{z}}(X)$ are simple and lie in $GF(q^m)$ - GF(q) where $m = \prod_{i=1}^{N} m_i$

Remark:

In unpublished work, Herlestam has given a result formally identical to (5.73) in order to describe the length of the linear equivalent able to produce every sequence from a given set. In contrast, our interest is in the linear complexity of a single sequence.

Proof: Corollary 5.14 assures that any kth-order product of sequences $\tilde{s}_1, \ldots, \tilde{s}_k$, has, under the conditions specified in the Corollary, associated a minimal polynomial of degree

$$M_k' = \prod_{i=1}^{k} M_1 .$$

Multiplication of a periodic sequence of GF(q) by a non-zero constant does not change the recursion of the sequence and thus does not affect the linear complexity of the sequence nor the roots of the associated minimal polynomial. Since $\gcd(m_i,m_j) = 1$ for all $i \neq j$, no two distinct products can possibly result in the same irreducible factors of the associated minimal polynomial. Thus by summing over all product terms in (5.71) with the sequences $\tilde{s}_i$, i=1, ..., N being the arguments, we obtain as minimal polynomial $m_{\tilde{z}}(X)$ for the running key $\tilde{z}$ the product of all the minimal polynomials associated to the individual product terms being present (in general it is the least common multiple of all these minimal polynomials). Consequently, the degree M of $m_{\tilde{z}}(X)$ is equal to the sum of the degrees M_k' of the minimal polynomials associated to the individual product terms being present in F, which proves the Corollary.

The relation (5.73) is an extremely handy design means; in the binary case it allows to calculate the linear complexity of the running key $\tilde{z}$ by simply substituting the arguments of F by the corresponding linear complexities and by evaluating F over he integers. The constraints we have imposed in order to obtain this guaranteeable linear complexity are that each of the input sequences $\tilde{s}_i$ must have a minimal polynomial $m_{\tilde{s}_i}(X)$ all of whose irreducible factors are distinct of degree m_i' dividing m_i and that $\gcd(m_i,m_j) = 1$, all $i \neq j$. Since we may always find, for any number of distinct irreducible polynomials, a common splitting field the first constraint does not really limit the design considerations. But clearly the second constraint that the fields $GF(q^m)$ contributing the roots to the individual minimal polynomials $m_{\tilde{s}_i}(X)$ may pairwise only have the ground-field GF(q) in common plays an important role. It assures that every distinct product of $\tilde{s}_i$'s generates roots in a distinct part of $GF(q^m)$ and thus is in fact responsible for the predictability of the linear complexity of the running key $\tilde{z}$.

Recently, Rueppel and Staffelbach (Ruep 86) were able to show that, in order to let the resulting product sequence achieve maximum linear complexity, the roots of the minimal polynomials of the involved sequences need not come from finite fields of pairwise relatively prime degrees. Their approach concentrated on the order of the roots rather than on the degree, or the order, of the finite fields which contain the roots. We state their main theorem without proof.

Theorem 5.16.

Let $\tilde{a}$ be a GF(q)-sequence of period T_a with minimal polynomial $m_{\tilde{a}}(X)$ of degree L all of whose irreducible factors are distinct. Let $\tilde{b}$ be a GF(q)-sequence of period T_b with irreducible minimal polynomial $m_{\tilde{b}}(X)$ of degree M. If $\tilde{a}$ and $\tilde{b}$ are multiplied termwise, then the resulting product sequence $\tilde{z}$ will have maximum linear complexity LM if

$$\text{ord}(q) \bmod t_b = M \tag{5.74}$$

where ord(q) mod t denotes the multiplicative order of q modulo t, and where t_b is defined as $T_b/\gcd(T_a,T_b)$.

If both $m_{\tilde{a}}(X)$ and $m_{\tilde{b}}(X)$ are irreducible, then theorem 5.16 implies that either

$$\text{ord}(q) \bmod t_a = L$$

(where $t_a = T_a/\gcd(T_a,T_b)$), or

$$\text{ord}(q) \bmod t_b = M$$

sufficies to guarantee maximum linear complexity of the product sequence $\tilde{z}$.

We illustrate theorem 5.16 by means of an example. $2^{11}-1$ factors into the prime factors 23 and 89. It is easily verified that

$$\text{ord}(2) \bmod 23 = 11$$

and

$$\text{ord}(2) \bmod 89 = 11$$

Hence, if one sequence has a primitive minimal polynomial of degree 11 and the other sequence has irreducible minimal polynomial of degree 11 and either period 23 or 89, then according to theorem 5.16 the product sequence will attain maximum linear complexity 121, despite the roots of both minimal polynomials are taken from the same Galois-field.

What about the practicality of condition (5.74)? If it is easy to find divisors t of q^m-1 which satisfy the condition ord(q) mod t = m, it is also easy to achieve maximum linear complexity of a product sequence. One has only to ensure that the period of the irreducible minimal polynomial contains t as a factor, and the period of the second (not necessarily irreducible) minimal polynomial is relatively prime to t. The solution to the posed problem of finding divisors t of q^m-1 with the property ord(q) mod t = m, and moreover to the problem of finding polynomials whose periods contain t as a factor, is astonishingly simple. But before we can give the answer we have to introduce the notion of a primitive factor. Any integer $t > 1$ which divides q^m-1 but is relatively prime to q^i-1 for $i < m$, is called a primitive factor of q^m-1. The greatest such primitive factor is denoted by F_m. AS an immediate consequence of this definition it follows that for any divisor t of F_m, the multiplicative order of q modulo t is m. Therefore, a sufficient condition for maximum line-

ar complexity of a product of two sequences (as defined in theorem 5.16) is that t_b contains a primitive factor of q^M-1. Interestingly enough it happens that almost all q^m-1 contain such a primitive factor. Note that for instance the factors of $2^6-1 = 3^2 7$ have already appeared in $2^2-1 = 3$ and in $2^3-1 = 7$ so that 2^6-1 has no primitive factor. The real surprising fact is that this early exception is the only exception. To be more precise, one can prove that, for all integers $q > 1$ and $m > 2$, every q^m-1 contains a primitive factor, except for the single case $q = 2$ and $m = 6$. Moreover, the greatest primitive factor F_m can actually be computed for every q^m-1. Let $Q_m(X)$ denote the m-th cyclotomic polynomial, that is, the polynomial all of whose roots are all the primitive m-th roots of unity and whose leading coefficient is one. $Q_m(X)$ has integer coefficients, is irreducible and has degree $\phi(m)$, where ϕ denotes Euler's function. Then Berl 68, P. 91)

$$X^m-1 = \prod_{d|m} Q_d(X) \tag{5.75}$$

is the complete polynomial factorization of X^m-1. Substituting q for X leads to a partial factorization of the number q^m-1,

$$q^m-1 = \prod_{d|m} Q_d(q) = Q_m(q) \prod_{\substack{d|m \\ d<m}} Q_d(q) \tag{5.76}$$

It follows that any primitive factor of q^m-1 must be contained in $Q_m(q)$. Again surprising, in most of the cases the greatest primitive factor is just equal to $Q_m(q)$. More specificly (Carm 14, Bril 83, Ruep 86),

$$F_m = \begin{cases} Q_m(q) & \text{if } Q_m(q) \text{ and } m \text{ are relatively prime} \\ Q_m(q)/p & \text{if there exists a common prime factor } p \text{ of } Q_m(q) \text{ and } m \text{ (there can be only one such factor).} \end{cases}$$

For illustration purposes we give in tables 5.4 and 5.5 the numerical values of the primitive factors of q^m-1 for $m < 50$ and $q = 2$ and 3 (an extensive list of complete factorizations of numbers of the form b^m-1 and b^m+1 can be found in Bril 83).

m	F_m	m	F_m
2	3	26	2731
3	7	27	262657
4	5	28	29 · 113
5	31	29	233 · 1103 · 2089
6	no primitive factor	30	331
7	127	31	2147483647
8	17	32	65537
9	73	33	599479
10	11	34	43691
11	23 · 89	35	71 · 122921
12	13	36	37 · 109
13	8191	37	233 · 616318177
14	43	38	174763
15	151	39	79 · 121369
16	257	40	61681
17	131071	41	13367 · 164511353
18	19	42	5419
19	524287	43	431 · 9719 · 2099063
20	41	44	397 · 2113
21	337	45	631 · 23311
22	683	46	2796203
23	47 · 178481	47	2351 · 4513 · 13264529
24	241	48	97 · 673
25	601 · 1801	49	4432676798593

Table 5.4. Greatest primitive factors of 2^m-1 for $1 < m < 50$ and their factorization

m	F_m	m	F_m
2	1	26	398581
3	13	27	109 · 433 · 8209
4	5	28	29 · 16493
5	11^2	29	59 · 28537 · 20381028
6	7	30	31 · 171
7	1093	31	683 · 102673 · 4404047
8	41	32	21523361
9	757	33	2413941289
10	61	34	103 · 307 · 1021
11	23 · 3851	35	71 · 2664097031
12	73	36	530713
13	797161	37	13097927 · 17189128703
14	547	38	2851 · 101917
15	4561	39	313 · 6553 · 7333
16	17 · 193	40	42521761
17	1871 · 34511	41	83 · 2526913 · 86950696619
18	19 · 37	42	43 · 2269
19	1597 · 363889	43	431 · 380808546861411923
20	1181	44	5501 · 570461
21	368089	45	181 · 1621 · 927001
22	67 · 661	46	23535794707
23	47 · 1001523179	47	1223 · 21997 · 5112662 · 96656723
24	6481	48	97 · 557 · 769
25	8951 · 391151	49	491 · 4019 · 8233 · 51157 · 131713

Table 5.5. Greatest primitive factors of 3^m-1 for $1 < m < 50$ and their factorization

As a practical consequence of the existence of these characteristic factors theorem 5.16 guarantees that the product of the output sequences of any N maximum-length LFSRs of different lengths (greater than 2) will exhibit maximum linear complexity (provided of course the LFSRs are started in a nonzero state). Moreover, proceeding along the same lines as in Corollary 5.15, if N maximum-length GF(q)-LFSRs all of whose lengths are different and greater than 2 are combined in a nonlinear function F (as defined in (5.71)) then the output sequence produced by F is guaranteed to have maximum linear complexity. And this maximum linear complexity is easily computed as the real value of F when the arguments of F are replaced by the corresponding LFSR-lengths and when the nonzero coefficients of F are replaced by 1 (compare (5.73)). For example, if the four binary m-LFSRs of lengths 10, 12, 15 and 16 are combined in the function $F = x_3 + x_1x_2 + x_2x_3x_4 + x_1x_2x_3x_4$, then the linear complexity of the resulting output sequence will be 15 + 10 12 + 12 15 16 + 10 12 15 16 = 31'815. For many years (or even decades) key stream generators of the type depicted in Fig. 5.12 were built with the restriction that the lengths of the driving m-LFSRs had to be pairwise relatively prime which rendered design and implementation more difficult. The above results show that this restriction essentially is superfluous, it can be replaced by the much weaker restriction of distinctness of the employed m-LFSR lengths. But any lengths convenient (e.g. multiples of 8, etc.) may be chosen for implementation.

5.3 Correlation-Immunity of Memoryless Combining Functions

It is illustrative to start out with a practical example. In 1973 Geffe (Geff 73) proposed as basic building block a 3-LFSR running key generator which employed the nonlinear function F_1, (here displayed in its algebraic normal form),

$$F_1(x_0,x_1,x_2) = x_2 + x_0x_1 + x_1x_2 \quad \text{in GF(2)} \tag{5.77}$$

to combine the 3 LFSR output sequences. Later it was noted that the following function F_2

$$F_2(x_0,x_1,x_2) = x_0x_1 + x_0x_2 + x_1x_2 \quad \text{in GF(2)}, \tag{5.78}$$

obtained by a slight modification of F_1, exhibits equally good distribution properties, but allows to generate sequences with a higher linear complexity. Therefore, mapping F_2 was termed "improved Geffe". In 1984, Brüer proposed the so-called "threshold mapping" (Brüe 84) which, in the case of 3 LFSRs, reduces to mapping F_2. Essentially, the threshold mapping puts out the most significant bit of the real sum of its input bits and therefore requires an odd number of input variables in order to achieve a balanced output distribution. With the help of Corollary 5.15 it is possible to determine the linear complexity of the output sequence of either F_1 or F_2 for specified input sequences. Suppose, for instance, F_1 and F_2 are fed by sequences $\tilde{y}_0$, $\tilde{y}_1$ and $\tilde{y}_2$ whose linear complexities are 15, 21 and 28, respectively (here it is assumed that the roots of $m_{\tilde{y}_0}(X)$ are from $GF(2^5)$ - GF(2), that the roots of $m_{\tilde{y}_1}(X)$ are from $GF(2^6)$ - GF(2), and that the roots of $m_{\tilde{y}_2}(X)$ are from $GF(2^7)$ - GF(2); note that 15 = 5 + 5 + 5, 21 = 3 + 6 + 6 + 6, 28 = 7 + 7 + 7 + 7, and that the degrees of the extension fields are pairwise relatively prime). Then the Geffe mapping F_1 would produce a sequence of linear complexity $F_1(15,21,28) = 931$ and the improved Geffe- or threshold mapping would produce a sequence of linear complexity $F_2(15,21,28) = 1323$.

At this point it is illuminating to examine the table form of function F_2 (5.78).

x_0	x_1	x_2	$F_2(x_0,x_1,x_2)$
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	1
0	1	1	1
1	1	1	1

Table 5.6. Mapping defined by the boolean function $F_2 = x_0x_1 + x_0x_2 + x_1x_2$

From table 5.6 it is apparent that if the three-dimensional input vectors to F_2 all are equally likely (that is, if the input sequences to F_2 have ideal statistics), then the function output will have equal probability of being a 1 or a 0. But suppose somebody reveals the value of x_0, then, since the value of F_2 agrees in 3/4 of all possible cases with x_0 (as can be seen from table 5.6) the uncertainty about the output of F_2 has been reduced considerably. Equivalently, knowing the output of F_2 reduces considerably the uncertainty about x_0 (in fact, since F_2 is symmetric, this effect is true for each of the three input variables). This "leakage" of information can be exploited as follows: assume that F_2 is driven by three LFSR-sequences, then when we multiply part of the key stream with part of the sequence $\tilde{x}_0$ term by term, and we hit the correct phase of $\tilde{x}_0$, we will expect at 3/8 of the positions a "1" instead of the usual fraction 1/4 which occurs when two random sequences are multiplied together. Consequently, correlating the key stream with all possible phase shifts of one of the input sequences $\tilde{x}_i$ (which is assumed to be generated by a fixed and known LFSR) will reveal the true phase of $\tilde{x}_i$ and thus the state of its generating LFSR. Such deficiencies in the chosen function F allow in general to subdivide the attack into M subattacks against each of the M subgenerators, which often decreases the necessary work sufficiently to make an attack feasible. Blaser and Heinzmann (Blas 79) were the first to point out this possibility which was then fully developed by Siegenthal (Sieg 84a). Siegenthaler proposed the information-theoretic model displayed in Fig. 5.15 in order to investigate the statistical dependencies introduced by the nonlinear combiner.

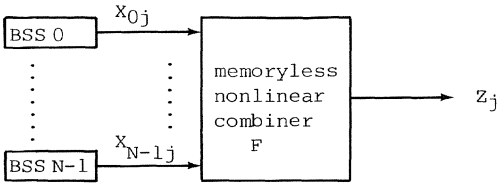


Fig. 5.15. Information-theoretic model used to define correlation-immunity (BSS = Binary Symmetric Source)

The input sequences to the nonlinear combiner are assumed to be sequences of independent and uniformly distributed binary random variables. This eliminates statistical effects introduced by nonideal sources such as LFSRs. A memoryless nonlinear function F is termed correlation-immune of oder m (Sieg 84b) if the output of F and any m input variables, considered jointly, are statistically independent. For a memoryless combiner time is immaterial, since at any time the output only depends on the current input variables. This allows us to drop the time index in the subsequent analysis. Therefore, a memoryless nonlinear function F is m-th order correlation-immune if the mutual information between the output variable and any subset of m input variables is zero, that is, if

$$I(Z; X_{i_1} X_{i_2} \ldots X_{i_m}) = 0$$
$$0 \leq i_1 < i_2 < \ldots < i_m \leq N-1 \qquad (5.79)$$

If a nonlinear combiner is m-th order correlation-immune it is not possible to correlate on any combination of m input sequences. Unfortunately, one has to pay a prize for correlation-immunity in terms of reduced nonlinear order. Siegenthaler (Sieg 84b, see also Xiao and Massey (Xiao 85)) showed that there exists a tradeoff between the attainable nonlinear order k and the attainable level of correlation-immunity m,

$$k + m \leq N \qquad (5.80)$$

where N is the number of input variables to F. When the random variable Z is required to be uniformly distributed, as is often the case in cryptographic applications, the tradeoff reads as

$$k + m \leq N \quad \text{for } m = 0 \text{ or } m = N-1$$
$$k + m \leq N-1 \quad \text{for } 1 \leq m \leq N-2 \qquad (5.81)$$

Thus, the more correlation-immunity, the smaller the nonlinear order of F and consequently the smaller the linear complexity of the running key, and vice versa. Fig. 5.16 graphically illustrates the situation.

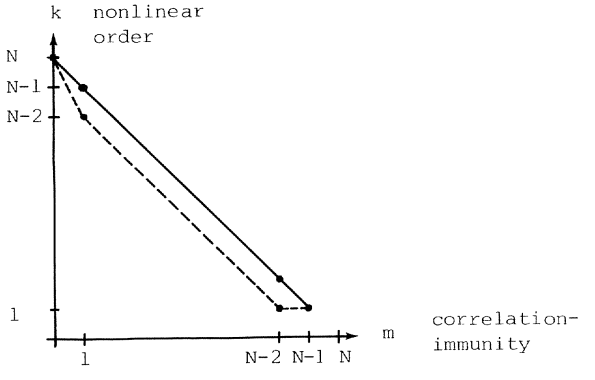


Fig. 5.16. Attainable region of k + m for (a) arbitrarily distributed Z (——), (b) uniformly distributed Z (--)

Maximum order of correlation-immunity is N-1 and corresponds to the GF(2)-sum of all input variables. If one demands minimal correlation-immunity of order 1 together with a balanced output distribution then the nonlinear order of F has to be reduced to at most N-2. This explains directly why with N = 3 input variables (as in the 2 exemplatory functions (5.77) and (5.78)) the memoryless function F either is correlation-immune or nonlinear, but not both. Moreover, functions which satisfy the tradeoff (5.80) or (5.81) with equality are in general difficult to find.

Very different is the situation when the restriction is dropped that the nonlinear combiner be memoryless. Then, as will be shown in Chapter 9, it is possible to obtain combiners, which exhibit at the same time maximum correlation-immunity and maximum nonlinear order. In fact, one bit of memory is sufficient to eliminate completely the tradeoff (5.80) or (5.81). Moreover, the nonlinear part of the overall combiner is freed from any restriction and thus can be implemented at any choice.

A very compact and illustrative description of the amount of correlation-immunity of a boolean function F can be obtained in terms of its Walsh transform (Xiao 85). In general, F defines a mapping from the vector space $GF(2)^N$ of binary N-tuples into GF(2). We may write $F(\underline{x})$ for the function value evaluated at the n-dimensional input vector $\underline{x} = (x_0, x_1, \ldots, x_{N-1})$; equivalently, and more compactly, we may write F(x), where $x = x_0 + x_1 2 + \ldots + x_{N-1} 2^{N-1}$ denotes the integer corresponding to the binary vector $\underline{x}$. Similarly, define $\underline{\omega} = (\omega_0, \omega_1, \ldots, \omega_{N-1})$ as vector in $GF(2)^N$ and $\omega = \omega_0 + \omega_1 2 + \ldots \omega_{N-1} 2^{N-1}$ as its corresponding integer. The Walsh transform of any real-valued function F over $GF(2)^N$ (as which we may trat our boolean function) is defined as

$$S_F(\omega) = \sum_{x=0}^{2^N-1} F(x)(-1)^{\underline{x}\cdot\underline{\omega}} \qquad (5.82)$$

where $\underline{x}\cdot\underline{\omega} = x_0\omega_0 + x_1\omega_1 + \ldots + x_{N-1}\omega_{N-1}$ is to be interpreted as the integer corresponding to the GF(2)-valued dot product of the two binary vectors. the function F(x) can be recovered by the inverse Walsh transform

$$F(x) = 2^{-N} \sum_{\omega=0}^{2^N-1} S_F(\omega)(-1)^{\underline{x}\cdot\underline{\omega}} \qquad (5.83)$$

Fig. 5.17 illustrates the Walsh transform pair for the exemplatory function F_2 described in table 5.6.

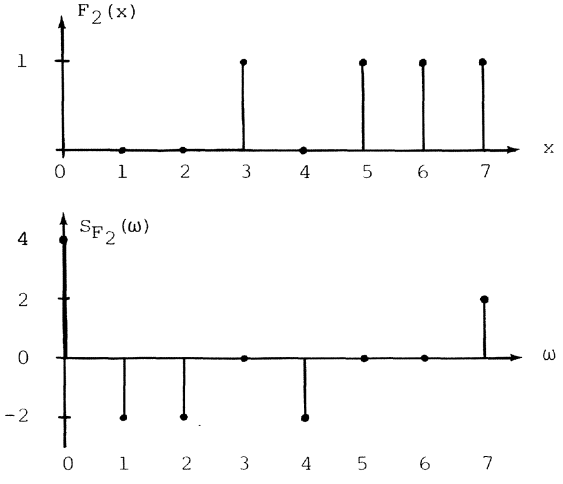


Fig. 5.17. The mapping F_2 (known as "improved Geffe" or "threshold mapping for 3 input variables") and its Walsh transform.

In the information-theoretic context of Fig. 5.15 we may say that a boolean function F creates statistical dependencies between subsets of its input variables and its output. Clearly, knowing the complete input vector to F does not leave any uncertainty about the output of F. But are there individual input variables or small subsets of input variables which reduce the uncertainty about the output of F? Or in other words, is there a nonzero mutual information between any subset $X_{i_1}, \ldots, X_{i_m}$ of input variables and the output variable Z of F? At this point a lemma due to Xiao and Massey (Xiao 85) is useful.

Lemma 5.16.

The discrete random variable Z is independent of the m independent and uniformly distributed binary random variables $X_1, \ldots, X_m$ if and only if Z is independent of the GF(2)-sum $c_1X_1 + c_2X_2 + \ldots + c_mX_m$ for every choice of $c_1, c_2, \ldots, c_m$ not all zeroes, in GF(2).

The lemma implies that the mutual information between the output variable Z of the boolean function F and any m input variables $X_{i_1}, \ldots, X_{i_m}$ considered jointly, i.e. $I(Z; X_{i_1}, \ldots, X_{i_m})$, is zero if and only if the mutual information between Z and any nonzero linear combination of the m input variables, i.e. $I(Z; c_1X_{i_1} + \ldots + c_mX_{i_m})$, is zero.

For instance, Z is independent of X_1X_2 if and only if Z is independent of X_1, of X_2 and of their sum $X_1 + X_2$.

Note that $\underline{\omega} \cdot \underline{x}$ in the Walsh transform (5.80) and in its inverse (5.83) defines a linear combination of the input variables for every choice of ω. It is convenient to introduce the integer-valued functions

$$N_{ab}(\omega) = \#\{x : z = F(x) = a \text{ and } \underline{\omega}\cdot\underline{x} = b\} \tag{5.84}$$

where a, b are in {0,1}, and where #{} denotes the cardinality of the indicated set. It follows from (5.82) that

$$S_F(\omega) = N_{10}(\omega) - N_{11}(\omega) \tag{5.85}$$

Now let us consider the conditional probability that $\underline{\omega}\cdot\underline{X}$ is equal to b given that Z = a. Assume that $\underline{\omega}$ is nonzero; we obtain

$$P(\underline{\omega}\cdot\underline{X} = b|Z = a) = \frac{P(\underline{\omega}\cdot\underline{X}=b, Z=a)}{P(Z=a)} = p_a^{-1}2^{-N}N_{ab}(\omega) \tag{5.86}$$

where p_a denotes the (nonzero) probability that Z is equal to a. Using the fact that

$$P(\underline{\omega}\cdot\underline{X} = 0|Z=1) + P(\underline{\omega}\cdot\underline{X} = 1|Z=1) = 1 \tag{5.87a}$$

$$P(\underline{\omega}\cdot\underline{X} = 0|Z=1) - P(\underline{\omega}\cdot\underline{X} = 1|Z=1) = p_1^{-1}2^{-N}(N_{10}(\omega) - N_{11}(\omega))$$

$$= p_1^{-1}2^{-N}S_F(\omega) \tag{5.87b}$$

leads to the solution

$$P(\underline{\omega}\cdot\underline{X} = 0|Z=1) = \frac{1}{2} + p_1^{-1}2^{-N-1}S_F(\omega) \tag{5.88a}$$

$$P(\underline{\omega}\cdot\underline{X} = 1|Z=1) = \frac{1}{2} - p_1^{-1}2^{-N-1}S_F(\omega) \tag{5.88b}$$

Note that for uniformly distributed random variables $X_0, \ldots, X_{N-1}$ it must hold

$$N_{0b}(\omega) + N_{1b}(\omega) = 2^{N-1} \tag{5.89}$$

Therefore, equivalently to (5.88), we obtain

$$P(\underline{\omega}\cdot\underline{X} = 0|Z=0) = \frac{1}{2} + p_0^{-1}2^{-N-1}S_F(\omega) \tag{5.90a}$$

$$P(\underline{\omega}\cdot\underline{X} = 1|Z=0) = \frac{1}{2} - p_0^{-1}2^{-N-1}S_F(\omega) \tag{5.90b}$$

Using (5.88) and (5.90) in

$$I(Z;\ \underline{\omega}\cdot\underline{X}) = H(\underline{\omega}\cdot\underline{X}) - H(\underline{\omega}\cdot\underline{X}|Z) \tag{5.91}$$

where $H(\underline{\omega}\cdot\underline{X})$ is 1 bit for any nonzero $\underline{\omega}\cdot\underline{X}$, proves the following proposition.

Proposition 5.17.

Let $X_0, X_1, \ldots, X_{N-1}$ be the N independent and uniformly distributed arguments of the boolean function F, whose output forms the random variable Z. Then for any $\underline{\omega} \neq \underline{0}$.

$$I(Z;\ \underline{\omega}\cdot\underline{X}) = 1 - p_0 h\left(\frac{1}{2} - \frac{S_F(\omega)}{p_0\ 2^{N+1}}\right) - p_1 h\left(\frac{1}{2} - \frac{S_F(\omega)}{p_1\ 2^{N+1}}\right) \tag{5.92}$$

where $h(p) = -p \log p - (1-p)\log(1-p)$ denotes the binary entropy function, and where p_a is $P(Z=a)$.

Moreover, when Z is uniformly distributed then

$$I(Z; \underline{\omega}\cdot\underline{X}) = 1 - h(\tfrac{1}{2} - 2^{-N}S_F(\omega)) \qquad (5.93)$$

It is interesting to observe that proposition 5.17 solves the problem of best linear (affine) approximation of any given boolean function. An affine function can be written as $b + c_0X_0 + c_1X_1 + \ldots + c_{N-1}X_{N-1}$, that is, a linear combination of the arguments plus a constant. By the best linear (affine) approximation of a given (generally nonlinear) boolean function F we mean that linear (affine) function whose value agrees with the given function F for the largest number of distinct arguments $x_0, x_1, \ldots, x_{N-1}$. The binary entropy function is monotonically decreasing for arguments $\frac{1}{2} + \alpha$, where $0 \leq |\alpha| \leq \frac{1}{2}$. Hence, according to (5.92) and (5.93) the largest $S_F(\omega)$ in magnitude specifies that linear combination of input variables $\underline{\omega}\cdot\underline{X}$ which maximizes the mutual information between the function output Z and $\underline{\omega}\cdot\underline{X}$. And according to (5.88) and (5.90) the sign of $S_F(\omega)$ determines when the value of the linear combination $\underline{\omega}\cdot\underline{X}$ has to be inverted to give the most agreement with the function output Z. The rule for finding the best linear (affine) approximation of a given boolean function F is: (1) compute the Walsh transform $S_F(\omega)$ of F, (2) find that nonzero ω which maximizes $|S_F(\omega)|$ and denote it by ω_m, (3) if $S_F(\omega_m)$ is negative then $\underline{\omega}_m\cdot\underline{X}$ is the best linear approximation, if $S_F(\omega_m)$ is positive then $1 + \underline{\omega}_m\cdot\underline{X}$ is the best linear approximation.

For example, in the Walsh transform $S_{F_2}(\omega)$ in Fig. 5.17 there are 4 condidate ω (1,2,4,7) which maximize $|S_{F_2}(\omega)|$. So, according to the rule above $1 + X_0$, $1 + X_1$, $1 + X_2$ or $X_0 + X_1 + X_2$ all are best affine approximations to $F_2 = X_0X_1 + X_0X_2 + X_1X_2$. The probability of agreement can be computed from (5.88b) or (5.90a), and is in this example equal to 3/4.

Now suppose that the nonlinear function F is not fixed but rather can be chosen from a set $\mathcal{F} = \{f_1, f_2, \ldots, f_M\}$ of boolean functions of N input variables. For instance, in a running key generator part of the key may be devoted to selecting a specific nonlinear combiner

from a set of suitable ones. The set $\mathcal{F}$ of suitable functions would be predetermined in order to satisfy minimal requirements such as a balanced output distribution, a certain level of correlation-immunity, and a certain nonlinear order. For the subsequent analysis we will make the following assumptions: (1) F denotes the random variable indicating which of the functions in $\mathcal{F}$ is chosen; (2) F is uniformly distributed, that is, $P(F=f_i) = 1/M$; (3) each f_i from $\mathcal{F}$ has the same output distribution, that is, in its table form each f_i has the same number of "1" entries, and therefore also the same number of "0" entries. Condition (3) assures that it is not possible, by simply monitoring the output distribution, to determine which function f_i has been chosen to produce Z. Define, corresponding to (5.84), for every f_i, $1 \leq i \leq M$, the integer-valued functions

$$N_{ab}^{(i)}(\omega) = \#\{x : z = f_i(x) = a \text{ and } \underline{\omega}\cdot\underline{x} = b\} \tag{5.94}$$

where a,b are in {0,1}. Consider for $\underline{\omega} \neq \underline{0}$ the joint probability

$$\begin{aligned} P(\underline{\omega}\cdot X=b;\ Z=a) &= \sum_{i=1}^{M} P(\underline{\omega}\cdot\underline{X}=b,\ Z=a|F=f_i)\ P(F=f_i) \\ &= \frac{1}{M}\, 2^{-N} \sum_{i=1}^{M} N_{ab}^{(i)}(\omega) \end{aligned} \tag{5.95}$$

Since we assumed that $P(Z = a|F = f_i) = P(Z = a) = p_a$ we obtain by combining Bayes' rule together with (5.95)

$$P(\underline{\omega}\cdot\underline{X}=b|Z=a) = \frac{1}{p_a M 2^N} \sum_{i=1}^{M} N_{ab}^{(i)}(\omega) \tag{5.96}$$

Using the law of total probability together with (5.96) implies

$$P(\underline{\omega}\cdot\underline{X}=0|Z=1) + P(\underline{\omega}\cdot\underline{X}=1|Z=1) = 1 \tag{5.97a}$$

$$P(\underline{\omega}\cdot\underline{X}=0|Z=1) - P(\underline{\omega}\cdot\underline{X}=1|Z=1) = \frac{1}{p_1 M 2^N}\left(\sum_{i=1}^{M} N_{10}^{(i)}(\omega) - \sum_{i=1}^{M} N_{11}^{(i)}(\omega)\right)$$

$$= \frac{1}{p_1 M 2^N}\sum_{i=1}^{M} S_{f_i}(\omega) \tag{5.97b}$$

which leads to the solution

$$P(\underline{\omega}\cdot\underline{X}=0|Z=1) = \frac{1}{2} + \frac{1}{p_1 M 2^N}\sum_{i=1}^{M} S_{f_i}(\omega) \tag{5.98a}$$

$$P(\underline{\omega}\cdot\underline{X}=1|Z=1) = \frac{1}{2} - \frac{1}{p_1 M 2^N}\sum_{i=1}^{M} S_{f_i}(\omega) \tag{5.98b}$$

From this point on the procedure is identical as in formulas (5.89) to (5.91). We therefore skip this part and immediately state the Corollary to Proposition 5.17.

Corollary 5.18.

Let $Z = F_i(X_0, X_1, \ldots, X_{N-1})$, where $X_0, X_1, \ldots, X_{N-1}$ are N independent and uniformly distributed binary random variables, where F_i is a randomly selected boolean function from a set $\mathcal{F} = \{f_1, f_2, \ldots, f_M\}$ all of whose members have the identical output distribution, and where Z denotes the output random variable. Then for any $\underline{\omega} \neq \underline{0}$

$$I(Z;\omega\cdot X) =$$

$$1 - p_0 h\left(\frac{1}{2} - \frac{\sum_{i=1}^{M} S_{f_i}(\omega)}{p_0 M 2^N}\right) - p_1 h\left(\frac{1}{2} - \frac{\sum_{i=1}^{M} S_{f_i}(\omega)}{p_1 M 2^N}\right) \tag{5.99}$$

where $p_a = P(Z=a) = P(Z=a|F=f_i)$.

Moreover, when Z is uniformly distributed,

$$I(Z;\underline{\omega}\cdot\underline{X}) = 1 - h\left(\frac{1}{2} - 2^{-N+1}M^{-1}\sum_{i=1}^{M} S_{f_i}(\omega)\right) \qquad (5.100)$$

As did proposition 5.17 for a single function, corollary 5.18 solves the problem of best linear (affine) approximation for a given set of boolean functions, Comparing (5.99) to (5.92) and (5.100) to (5.93) we notice that the mutual information between Z and $\underline{\omega}\cdot\underline{X}$ is maximized at that ω where the sum of the Walsh transform coefficients is maximum in magnitude. Thus, to find the best linear approximation for a set of functions we have to compute the Walsh transform for each function in the set, sum the Walsh transforms componentwise (over the reals), and proceed with this Walsh transform sum in exactly the same way as we did for a single function.

We will now illustrate the usefulness of Walsh transforms by analyzing and approximating some functions encountered in the Data Encryption Standard (DES) (NBS 77). As example shall serve the S-box S_2, whose mappings $y = S_2(c,x)$ are specified as follows

c \ x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	14	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

(5.101)

In each S-box there are 4 invertible mappings from $GF(2)^4$ to $GF(2)^4$; the 4 input bits specify one out of 16 columns, and 2 additional control bits specify one out of 4 rows (corresponding to the 4 invertible mappings); the table entry y at this point specifies the integer corresponding to the 4 output bits (y_0, y_1, y_2, y_3). For instance, let the control bit vector be (1,1) = 3 and let the input vector be (1,1,0,1) = 11, then the output vector is $S_2(3,11) = 12 = (0,0,1,1)$.

Let $S_2^{-1}(c,y)$ denote the inverse mapping of $S_2(c,x)$, that is, $S_2^{-1}(c, S_2(c,x)) = x$. Then $S_2^{-1}(c,y)$ is given as

c \ y	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	1	10	6	7	14	4	9	2	8	15	5	12	11	3	0
1	9	10	5	0	2	15	12	3	6	13	11	14	8	1	7	4
2	0	7	14	13	5	8	11	2	9	12	4	3	10	6	1	15
3	12	3	7	4	6	13	9	10	1	15	2	8	11	0	14	5

(5.102)

For the moment we will concentrate on the mapping specified by the last row in S_2^{-1}, that is, $S_2^{-1}(3,y)$. $S_2^{-1}(3,y)$ defines 4 boolean functions $f_0, \ldots, f_3$ from $GF(2)^4$ to GF(2), one for each bit of x. Let $x_i = f_i(\underline{y})$, $i = 0, \ldots, 3$, where $\underline{y} = (y_0, y_1, y_2, y_3)$ is the binary vector corresponding to the integer $y = y_0 + y_1 2 + y_2 2^2 + y_3 2^3$. Then, for every f_i we may compute the Walsh transform and find a best linear (affine) approximation. We renounce displaying the 4 Walsh transforms and directly give the resulting optimal linear (affine) approximations

$$
\begin{aligned}
x_0 &= y_0 + y_1 + y_3 \\
x_1 &= y_0 + y_1 + y_2 \\
x_2 &= y_1 + y_2 + y_3 \\
x_3 &= y_0 + y_2 + y_3 + 1
\end{aligned}
\qquad (5.103)
$$

Converting the system (5.103) of affine GF(2)-equations into the integer representation of (5.101) and (5.102) we obtain

y	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x	8	(3)	15	(4)	(6)	(13)	1	(10)	5	14	(2)	9	(11)	(0)	12	7

(5.104)

(5.104) gives the best linear approximation of $S_2^{-1}(3,y)$. Most remarkable, half of the linear outputs coincide correctly with the images of the original nonlinear mapping (these numbers are circled), and in the remaining half of the linear outputs only 1 bit (out of 4) is wrong at a time. Thus, in 56 out of 64 cases the output bit of the original nonlinear mapping $s_2^{-1}(3,y)$ is correctly computed by the best linear (affine) approximation (5.103) or (5.104), or, in other words, the probability that the linear approximation delivers the correct output bit is as high as 87.5 %.

Applying the same technique directly to the feedforward mapping $S_2(3,x)$ results in a success rate of 75 %, that is in 3/4 of the cases the linear (affine) approximation delivers the correct output bit. These figures are typical for all the S-Box mappings implemented in DES, we found that for all S-Boxes for feedforward and inverted mappings the successrate of the best linear (affine) approximation typically lies between 75 % and 87.5 % (with slightly better results for the inverted mappings).

Instead of directly applying the Walsh approximation technique to the feedforward mapping $S_2(3,x)$ we may also try to work back from the highly successful linear approximation of the inverted mapping $S_2^{-1}(3,x)$. Straightforward inversion of (5.104) results in

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
y	(13)	6	(10)	(1)	(3)	8	(4)	15	0	11	(7)	(12)	14	(5)	9	2

(5.105)

leaving the correct matches (circled numbers) untouched. But interestingly enough, all the other images now have exactly 3 bit errors, i.e. 1 is mapped into 6 instead of 8, 5 is mapped into 8 instead of 15, etc. Thus, the mapping (5.105) would have a successrate of only 62.5 %. Yet if we complement the bit vectors corresponding to the uncircled numbers in (5.105), that is, 6 is changed into 9, 8 into 7, 15 into 0, etc., then the resulting mapping will have a successrate as high as 92.5 %. Note that selectively complementing half the images of a linear mapping makes the overall mapping nonlinear.

Now let us consider a different problem. Suppose we want to approximate the most significant bit of $S_2^{-1}(c,y)$ (5.102) regardless of which of the four invertible mappings is selected by the control bits. Let $x_3^{(i)} = f_3^{(i)}(y)$, $i = 0,\ldots,3$, where $x_3^{(i)}$ is the most significant bit of $S_2^{-1}(i,y)$. From (5.102) we can easily compute $x_3^{(i)}$, $i = 0,\ldots,3$, as follows:

y	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$x_3^{(0)}$	1	0	1	0	0	1	0	1	0	1	1	0	1	1	0	0
$x_3^{(1)}$	1	1	0	0	0	1	1	0	0	1	1	1	1	0	0	0
$x_3^{(2)}$	0	0	1	1	0	1	1	0	1	1	0	0	1	0	0	1
$x_3^{(3)}$	1	0	0	0	0	1	1	1	0	1	0	1	1	0	1	0

(5.106)

Our goal is to find the best linear approximation to the set of boolean functions $\{f_3^{(0)}, f_3^{(1)}, f_3^{(2)}, f_3^{(3)}\}$. Note that these mappings all have the same uniform output distribution (as can easily be checked from (5.106)). We further assume that the control bits (c_0, c_1) are independent and uniformly distributed.

According to corollary 5.18, we may find the best linear approximation to the whole set of functions $\{f_3^{(i)}\}$ by computing the componentwise real sum of the Walsh transforms of each function in the set. The resulting sum reads as

ω	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\sum_i S_{f_3^{(i)}}(\omega)$	32	0	4	-4	0	0	0	0	0	0	-4	-4	-4	(12)	4	(12)

(5.107)

There are two nonzero ω which maximize the magnitude of the combined Walsh transform coefficient (see circled numbers). We randomly select $\omega = 15$ which corresponds to the linear combination $\underline{\omega} \cdot \underline{y} = y_0 + y_1 + y_2 + y_3$. Since the sign of the combined Walsh transform coefficient is positive at $\omega = 15$, we must complement $y_0 + y_1 + y_2 + y_3$. Thus the best linear (affine) approximation simultaneously to the set of 4 boolean functions given in (5.106) is

$$x_3 = y_0 + y_1 + y_2 + y_3 + 1 \tag{5.108}$$

The function (5.108) is the best linear (affine) choice when we do not know which mapping $f_3^{(i)}$ has been selected by the control bits. Independent of the control bits it will deliver in 44 out of 64 cases the correct bit x_3, or, in other words, its successrate is 68.75 %.

Applying the same technique to S-Box S_5 yields a remarkable successrate. Suppose, as before, we want to find the best linear (affine) approximation for the most significant input bit x_3 as function of the output bits (y_0, y_1, y_2, y_3), irrespectively of the value of the control bits. The Walsh transform procedure indicates that also for S_5 the function (5.108) is the optimum linear choice. But now the linear mapping will deliver the correct bit x_3 in 52 out of 64 cases or, in other words, its successrate is as high as 81.25 %

The usefulness of the optimum linear approximation technique is not yet exhausted. Let us leave cryptography for the moment for a short detour into the realm of coding theory. Consider a code with N information bits and a codewordlength of 2^N bits. $(2)^{2^N}$ bits are needed to completely specify a boolean function of N variables. Among the set of these $(2)^{2^N}$ boolean functions there are 2^N strictly linear functions of N variables. If we now choose these 2^N linear functions in table form description as our codewordset, then the Walsh transform will enable us to decode optimally. Any errors which occurred during the transmission of the codeword will in general change the linear table form into a nonlinear function description. Applying the Walsh transform to the received codeword will tell us what is the best linear (affine) approximation to the received function description which is by definition the optimum decoding rule. Note that we implicitly assumed that the information bits are independent and uniformly distributed, and that the all-zero codeword can be handled separately (it defines the all-zero linear combination which was excluded in proposition 5.17). But the Walsh transform gives in fact not only the best linear but also the best affine approximation to a given boolean function. So one may pack an additional information bit into the code, keeping the codewordlength the same, since there are 2^{N+1} affine functions of N variables. For example, one can construct a (8,4)-code with minimum distance 4 by using as codewordset all 2^4 affine function description of length 8 (that is, all functions of the form $c_0 + c_1x_1 + c_2x_2 + c_3x_3$, where the information bits in our code interpretation are represented by the c_i, i = 1, ..., 4). Using the Walsh transform decoding procedure one can correct one error and detect two errors.

The above described code is known as Reed-Muller code of order one (MacW 83, p. 373). In the context of coding theory the term "Hadamard Transform" is used instead of "Walsh Transform", but the maximum likelihood decoding rule is formally equivalent to the optimum linear (affine) approximation technique developed in this section.

Now let us turn back to the general problem of characterizing the correlation-immunity of boolean functions. From (5.92) and (5.93) in proposition 5.17 it follows that Z is statistically independent of $\underline{\omega}\cdot\underline{X}$ if $S_F(\omega)$ is zero; conversely, if the mutual information $I(Z;\underline{\omega}\cdot\underline{X})$ is positive then $S_F(\omega)$ is nonzero. This directly leads to the cha-

racterization of the level of correlation-immunity inherent in a boolean function F in terms of the Walsh transform of F. The following result was pointed out by Xiao and Massey (Xiao 85).

Corollary 5.19.

A boolean function F is correlation-immune of order m if and only if

$$S_F(\omega) = 0 \qquad \text{for } 1 \leq W_2(\omega) \leq m$$

where $W_2(\omega)$ denotes the binary weight of ω.

Correlation-immunity of order m means there is no mutual information between Z and any subset of m input variables, considered jointly. This in turn implies, by virtue of Lemma 5.16, that there is no mutual information between Z and any nonzero linear combination of m or fewer input variables, and thus proves corollary 5.19.

The table form and the algebraic normal form (ANF) are both useful ways of representing a boolean function F. So far we used whichever form proved convenient,; now we will demonstrate how one form may be converted into the other. Usually the ANF of a boolean function F is given as in (5.3) where a kth-order product term is written as

$$a_{i_1 i_2 \cdots i_k} x_{i_1} x_{i_2} \cdots x_{i_k}$$

In what follows we will simplify notation and write for the same product term

$$a_i x_{i_1} x_{i_2} \cdots x_{i_k} \tag{5.109}$$

where

$$i = 2^{i_1} + 2^{i_2} + \ldots + 2^{i_k} \quad \text{and} \quad 0 \leq i_1 < i_2 < \ldots < i_k \leq N-1$$

and we will denote the constant term by a_0. Then, for instance, the ANF of a boolean function of 3 variables is written as

$$F(x_0,x_1,x_2) = a_0 + a_1x_0 + a_2x_1 + a_3x_0x_1 + a_4x_2 + a_5x_0x_2 + a_6x_1x_2 + a_7x_0x_1x_2 \qquad (5.110)$$

For convenience, let again x be the integer corresponding to the binary vector $\underline{x} = (x_0, x_1, \ldots, x_{N-1})$, that is, $x = x_0 + x_1 2 + \ldots + x_{N-1}2^{N-1}$, and let i have binary representation $\underline{i} = (i_0, i_1, \ldots, i_{N-1})$, that is, $i = i_0 + i_1 2 + \ldots + i_{N-1}2^{N-1}$. Notice that (5.110) implies that $F(0) = a_0$, $F(1) = a_0 + a_1$, $F(2) = a_0 + a_2$, $F(3) = a_0 + a_1 + a_2 + a_3$, and so on. It is easily verified that generally in order to compute F(X), all those a_i have to be summed (mod2) whose index $\underline{i}$ has a zero component at the same positions as $\underline{x}$. Thus, the rule for converting the ANF of a boolean function into its table form may be stated as follows,

$$F(x) = \sum_{\{i:\underline{i}\wedge\overline{\underline{x}}=0\}} a_i \qquad \text{(mod 2)} \qquad (5.111)$$

where $\underline{i} \wedge \overline{\underline{x}}$ means the componentwise logical and operation between $\underline{i}$ and the complement of $\underline{x}$. Since in GF(2) addition and subtraction are the same, it follows from (5.11) that

$$a_0 = F(0)$$
$$a_1 = F(1) - a_0 = F(1) + F(0)$$
$$a_2 = F(2) - a_0 = F(2) + F(0)$$
$$a_3 = F(3) - a_0 - a_1 - a_2 = F(3) + F(2) + F(1) + F(0), \quad \text{etc.}$$

And interestingly enough, the conversion from table form to ANF is formally equivalent to (5.11),

$$a_i = \sum_{\{x:\underline{x}\wedge\overline{\underline{i}}=\underline{0}\}} F(x) \qquad \text{(mod 2)} \qquad (5.112)$$

Consequently, the same procedure can be used to implement either conversion. We graphically illustrate the conversion procedure for a boolean function of 3 variables whose description requires 8 bits in either the ANF or the table form.

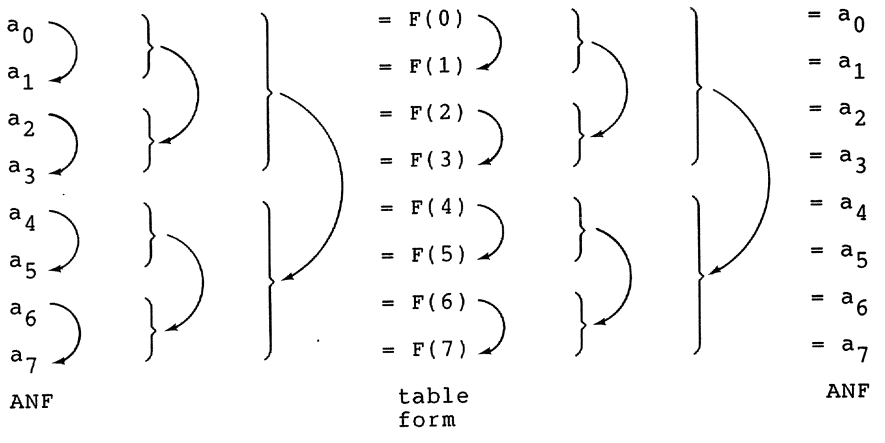
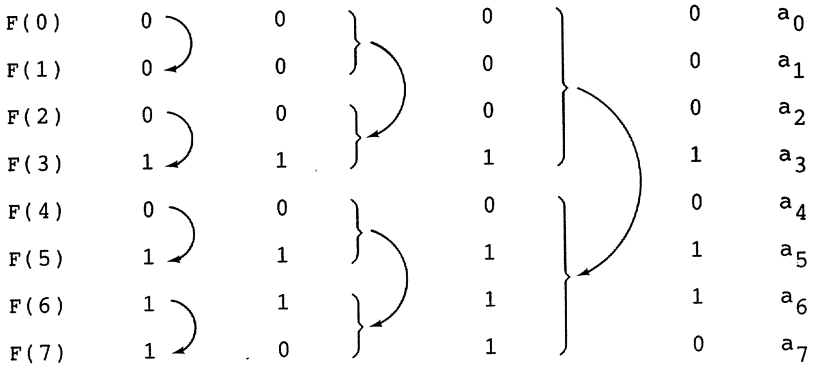


Fig. 5.18. The same conversion procedure can be used for conversion from ANF to table form of a boolean function as for conversion from table form to ANF. Each arrow indicates a GF(2) vector addition.

As an example let us convert F_2 as displayed in its table form in table 5.6 into its ANF.



The result is $F_2(x_0,x_1,x_2)$ as can be checked in (5.78).

Since the Walsh transform provides even another complete description of a boolean function F, we may ask about the direct relationship of $S_F(\omega)$ to the ANF of F. Combining (5.112) with the definition (5.83) of the inverse Walsh transform we obtain

$$a_i = \sum_{\{x : x \wedge \underline{i} = 0\}} 2^{-N} \sum_{\omega=0}^{2^N-1} S_F(\omega)(-1)^{\underline{x}\cdot\underline{\omega}} \qquad (\text{mod } 2)$$

Rearranging the sums yields

$$a_i = 2^{-N} \sum_{\omega=0}^{2^N-1} S_F(\omega) \sum_{\{x : x \wedge \underline{i} = 0\}} (-1)^{\underline{x}\cdot\underline{\omega}} \qquad (\text{mod } 2)$$

The second sum takes on the value $2^{W_2(i)}$ (this is the number of x which satisfy $\underline{x} \wedge \underline{i} = \underline{0}$) whenever ω is such that $\underline{\omega} \wedge \underline{i} = \underline{0}$, and is zero otherwise. Therefore, we may write

$$a_i = 2^{W_2(i)-N} \sum_{\{\omega : \underline{\omega} \wedge \underline{i} = \underline{0}\}} S_F(\omega) \qquad (\text{mod } 2) \qquad (5.113)$$

where $W_2(i)$ denotes the binary weight of i. Relation (5.113) is the desired conversion formula from the Walsh transform to the ANF directly, and was first proved in (Xiao 85). It can be used to determine the effects of correlation-immunity on the structure of the ANF. Corollary 5.19 states that for a mth-order correlation-immune function F the Walsh transform $S_F(\omega)$ must be zero whenever $1 \leq W_2(\omega) \leq m$. The summing condition $\underline{\omega} \wedge \underline{i} = \underline{0}$ in (5.113) implies $W_2(\omega) + W_2(i) \leq N$. Thus, if $W_2(i) \geq N-m$ then $W_2(\omega) \leq m$, and the only nonvanishing term in (5.113) will be $S_F(0)$. Hence, for the ANF of a mth-order correlation-immune function it holds that

$$a_i = 2^{W_2(i)-N} S_F(0) \quad (\text{mod } 2) \quad \text{if } W_2(i) \geq N-m. \qquad (5.114)$$

When F is required to provide a uniform output distribution then $S_F(0) = 2^{N-1}$ (since $S_F(0)$ directly counts the number of ones in the table form of F), and (5.114) becomes

$$a_i = \begin{cases} 0 & \text{if } W_2(i) \geq N-m > 1 \\ 1 & \text{if } W_2(i) \geq N-m = 1 \end{cases} \qquad (5.115)$$

Among the nonzero coefficients a_i those whose index has the largest binary weight $W_2(i)$ determine the nonlinear order k of F, in fact, k is equal to the largest $W_2(i)$ given that a_i is nonzero. Thus (5.115) imposes an upperbound on the nonlinear order k when the desired order of correlation immunity is m,

$k + m < N$ if $m < N-1$

$k + m \leqslant N$ if $m = N-1$

This is the tradeoff encountered in (5.81) and first derived by Siegenthaler (Sieg 84b).

In chapter 9 we will show that correlation-immunity and nonlinear order need not form a tradeoff when the combiner F is allowed to contain memory.

5.4 Summary and Conclusions

It was emphasized that a good running key generator should always be an analyzable one. As reference system the algebraic normal form of a nonlinear binary function (or its equivalent over GF(q)) was chosen and the basic figure of merit in the analysis has been the linear complexity of the produced key stream. The theory presented makes possible the analysis and construction of feedforward generators of the type depicted in Fig. 5.19 in terms of the canonical system provided by the algebraic normal forms of the nonlinear functions involved.

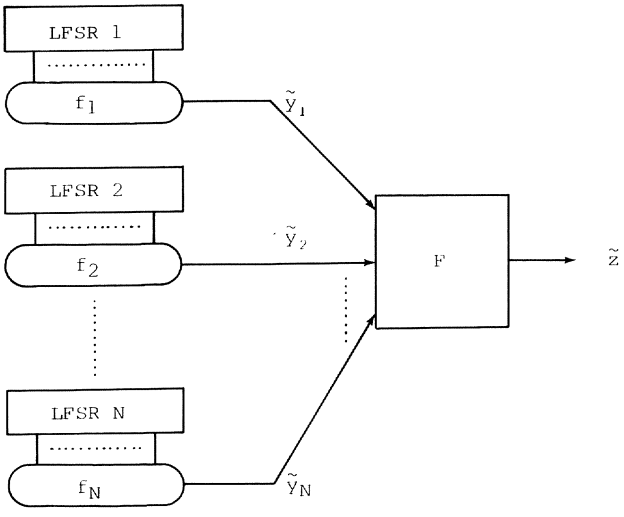


Fig. 5.19. General analyzable nonlinear feedforward keystream generator

The modular approach allowed to separate the effects introduced by the nonlinear transformations.

(I) The nonlinear state filter f

Here we are given a set of sequences observable at the distinct stages of a LFSR with irreducible connection polynomial (or in general a set of distinct phases of one sequence with irreducible minimal polynomial) and the problem posed is to

find classes of functions which when applied to the stages of the driving LFSR produce a sequence of predictable linear complexity. This problem can in principle be solved. We showed a

(1) conceptual matrix approach which allowed complete description but is infeasible in practice because of the amount of data that has to be considered. This approach was included for the additional insight it provided into the structural properties of nonlinear state filters.

The key parameter of f is its order. There is a close connection between growing order k of the nonlinear state filter f and growing linear complexity $\Lambda(\tilde{y})$ of the produced output sequence $\tilde{y}$.

(2) When the information about the nonlinear state filter f is reduced to only specifying its order k, it can be proven (Key 76) that

$$\Lambda(\tilde{y}) \leq \sum_{i=1}^{k} \binom{L}{i} = L_k \qquad (5.116)$$

where L is the length of the driving LFSR over GF(2) whose connection polynomial is assumed to be primitive.

The bound (5.116) indicates that one may obtain about exponential increase in linear complexity with linearly growing order of f. Note that when k = L

$$\Lambda(\tilde{y}) \leq \sum_{i=1}^{L} \binom{L}{i} = 2^L - 1. \qquad (5.117)$$

Since the bound (5.116) is in most cases satisfied with equality, a deviation of the actual linear complexity from the expected value given by the bound is called a degeneracy. It is of course unpleasant for the cryptographer only to have an

upperbound on the complexity of a nonlinearly generated sequence, in particular when no minimal complexity can be guaranteed. One part of our analysis was concerned with providing such a lowerbound for a class of functions as broad as possible.

(3) Let the driving LFSR of length L have a primitive connection polynomial defined over a field GF(q) of characteristic 2. If the function f is chosen such that

$$\tilde{y} = \sum_{i=0}^{N-1} c_i \tilde{s}^i \tilde{s}^{i+\delta} \ldots \tilde{s}^{i+(k-1)\delta} + f'(\tilde{s}^0, \tilde{s}^1, \ldots, \tilde{s}^{L-1}) \qquad (5.118)$$

where every kth-order product comprises only equidistant phases of the driving m-sequence $\tilde{s}$ where the distance δ between the phases is relatively prime to the period of $\tilde{s}$, and where the order of f' is assumed to be smaller than k, then the linear complexity of the produced sequence is at least

$$\Lambda(\tilde{y}) \geq \binom{L}{k} - (N-1) \qquad (5.119)$$

If L is prime and N is smaller or equal to L then the lowerbound (5.119) becomes

$$\Lambda(\tilde{y}) \geq \binom{L}{k} \qquad (5.120)$$

The lowerbound (5.119) might give a sufficient level of security if L is chosen large enough or if the state filter generator is used as a building block in a bigger system.

When on the other hand, the nonlinear function f is selected completely randomly, severe degeneracies may happen, and the lowerbound (5.119) does not apply. In order to assess to potential danger inherent in a random selection of f we need to know the likelyhood of a degeneracy to occur.

(4) Let f be a randomly selected kth-order nonlinear function to be applied to the stages of an LFSR over GF(2) of prime length L with primitive connection polynomial. Then the probability that the produced sequence $\tilde{y}$ is nondegenerate is lowerbounded by

$$P\left[\Lambda(\tilde{y}) = L_k\right] \simeq \exp\left[-\frac{L_k}{L\ 2^L}\right] > \exp\left[-\frac{1}{L}\right] \qquad (5.121)$$

Consequently the probability of any degeneracy happening goes to zero with increasing L. The probability of a single irreducible factor of degree L vanishing in the linear equivalent is approximately

$$P\left[\Lambda(\tilde{y}) = L_k - L\right] \simeq \frac{L_k}{L\ 2^L} \exp\left[-\frac{L_k}{L\ 2^L}\right]$$

$$\simeq \frac{1}{L} \exp\left[-\frac{1}{L}\right] \text{ for k close to L.} \qquad (5.122)$$

(Note that we neglected the linear factor X + 1 which may occur in $m_{\tilde{y}}(X)$ when k = L; its vanishing was not considered to be a degeneracy).

Even with a driving LFSR of as small a length as 5, only about 1,5 % of the 4th-order functions would result in a degeneracy of more than just one irreducible factor of degree 5 in the minimal polynomial associated to the produced sequence $\tilde{y}$. We conclude that for large L the probability of a severe degeneracy happening is virtually zero and thus may be neglected. Perhaps this is the meaning that Key (Key 76) had in mind when he wrote: "Exceptions of this sort are few and present neither a theoretical difficulty nor a practical limitation".

(II) The nonlinear combiner F

Here the situation is quite different compared to the state filter: we are given a function F and the problem posed is to find classes of periodic sequences which, when taken as arguments of F, produce a sequence of predictable linear complexity. This problem seems to be inherently easier to solve. At least there exist some simple constructive rules as to how the periodic input sequences may be chosen to obtain complete analyzability.

(1) For any binary input sequences $\tilde{y}_1$, .., $\tilde{y}_N$ to F, as produced by nonlinear state filter generators, the linear complexity of the running-key $\tilde{z}$ is completely determined by the algebraic normal form of F and the linear complexities of the input sequences as

$$\Lambda(\tilde{z}) = F(\Lambda(\tilde{y}_1), \ldots, \Lambda(\tilde{y}_N)) \qquad (5.123)$$

evaluated over the integers, if the connection polynomials of the driving LFSRs are irreducible and of pairwise relatively prime degree.

(2) Recently, Rueppel and Staffelbach (Ruep 86) showed that, if N maximum-length GF(q)-LFSRs, all of whose lengths are different and greater than two, are combined in a nonlinear function F (as defined in (5.71)) then the output sequence is guaranteed to have maximum linear complexity. And this maximum linear complexity is easily computed as the real value of F with the arguments of F being replaces by the corresponding LFSR-lengths, and with the nonzero coefficients of F being replaced by 1. The reason for this result lies in the remarkable fact that every q^m-1, for all integers $q > 1$ and $m > 2$ (except for the single case $q = 2$ and $m = 6$) contains a primitive factor, that is, a factor which is not contained in any q^i-1, $i < m$.

A possible attack on a nonlinear combiner F, is the correlation attack (Blas 79, Sieg 84a). When F is not properly chosen the correlation of the running-key $\tilde{z}$ with one of the subgenerator sequences $\tilde{y}_i$ provides the information necessary to determine the phase of the subgenerator.

(3) A memoryless F can be made mth-order correlation-immune (Sieg 84b) (such that it is not possible to correlate on any combination of m subgenerators). But this reduces the order of F and thus also the attainable linear complexity. The order k of F is limited by

$$k + m < N \qquad 1 \leq m \leq N-2 \tag{5.124}$$

when N denotes the number of arguments in F. Xiao and Massey (Xiao 85) gave an interesting interpretation of the correlation-immunity of a memoryless function F in terms of the Walsh transform of F.

(4) The trade-off (5.124) can be avoided by allowing memory in the nonlinear combining function F. In fact, as is shown in chapter 9, one bit of memory suffices to obtain maximum order N-1 of correlation-immunity together with maximum nonlinear order N of F.

A completely different problem is to find a good linear (affine) approximation of any given boolean function.

(5) We have shown that the Walsh transform solves this problem optimally. When $S_F(\omega)$ denotes the Walsh transform of F, then that nonzero ω which maximizes $|S_F(\omega)|$ specifies the optimal linear combination of input variables, and the sign of this maximum Walsh transform coefficient specifies the constant term in the linear (affine) approximation.

Loosely speaking, the nonlinear combiner F provides the means of increasing the linear complexity by generating new roots in bigger fields, whereas the nonlinear state filter provides the means of utilizing the elements available in the field defined by the irreducible connection polynomials of the driving LFSRs. As illustration for an analyzable running-key generator of the type depicted in Fig. 5.19 we may continue the example given at the beginning of section 5.3 by showing its realization.

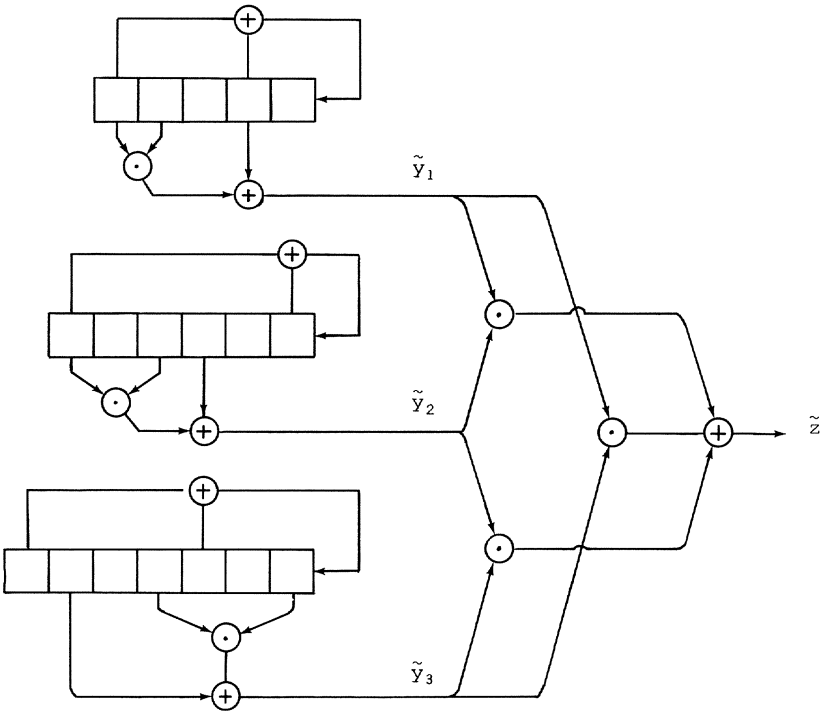


Fig. 5.20. Possible realization of the example given at the beginning of section 5.3 where $\Lambda(y_1)=15$, $\Lambda(y_2)=21$, $\Lambda(y_3)=28$ and the resulting $\Lambda(z)=1323$

6 Multiple Speed: An Additional Parameter in Secure Sequence Generation

The classical means to drive key stream generators are linear feedback shift registers (LFSRs). Usually the feedback taps of these driving LFSRs are predetermined (e.g. hardwired) and the key resides in the initial contents of the LFSRs and possibly in the nonlinear combinations of these LFSRs. Additional flexibility and security could be provided if the feedback connections of the driving LFSRs were also controlled by the key. To the cryptanalyst it is clearly of advantage to know the exact recursion the driving sequences have to obey; he is then confronted with the problem of determining the actual state of the LFSRs (in the last chapter we saw that exactly this complete knowledge about the driving LFSRs made the correlation attack feasible when the nonlinear combining function was chosen carelessly). But to control the feedback connections is in general a difficult task. A driving LFSR usually has a primitive or irreducible connection polynomial such that all nonzero initial states result in the same linear complexity of the produced sequence. So first one has to find some primitive or irreducible polynomials of degree equal to the desired length of the driving LFSR. Then the coefficients of the possible feedback polynomials have to be stored which might consume a considerable amount of storage. And finally some means have to be provided to physically alter or reprogram the feedback connections. The described process of controlling the feedback taps is so inconvenient that designers usually dismiss it, despite the desirable effects of having part of the key residing in the feedback connections. But fortunately today's advanced technology makes possible a very elegant way of controlling the feedback polynomials, simultaneously avoiding the described difficulties. The key idea lies in clocking each of the driving LFSRs at a constant multiple of the system clock (Fig. 6.1).

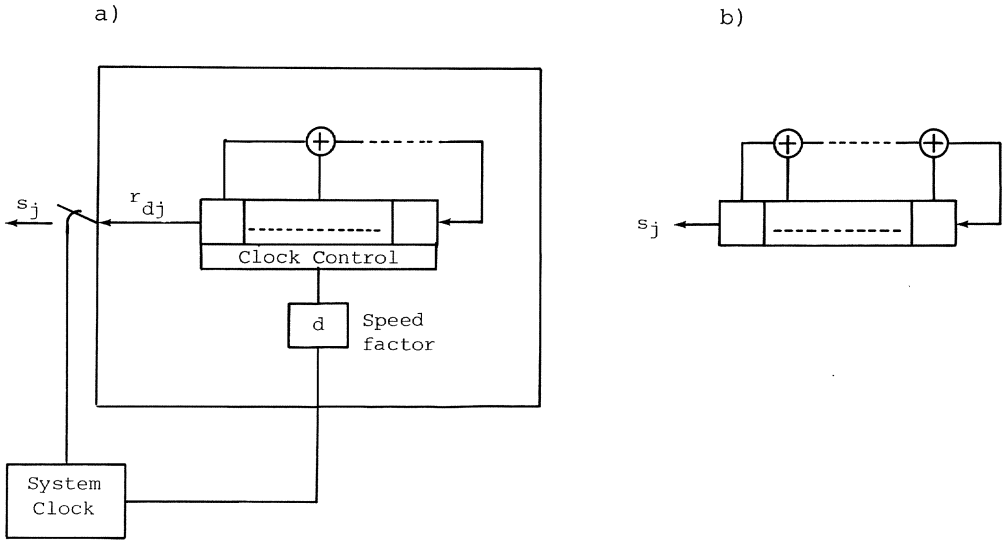


Fig. 6.1. Conceptual configuration of a d-fold clocked LFSR with fixed feedback connections (a) and the mathematically equivalent LFSR as observed from the outside.

We will call the actually implemented LFSR as shown in Fig. 6.1 the original LFSR, it is completely described by its length L and its connection polynomial C(D). For the original LFSR, time is passing d times faster as for the surrounding system, d will be called the speed factor. When the d-fold clocked original LFSR is observed only at the system clock times it will behave like a different LFSR which we will call the simulated LFSR (Fig. 6.1). The interesting question now is in what way the properties of the simulated LFSR relate to those of the original LFSR. For the moment suppose the speed factor d can be chosen such that the simulated LFSR has an irreducible or primitive connection polynomial. Then we do not even have to know the actual feedback connections simulated. As long as on both sides the same key is used, the same LFSR is simulated and hence the same sequence generated. We also do not need additional storage except for holding the speed factor d, and, maybe most important, we do not have to alter the feedback connections in any way. Given the original LFSR, the choice of the speed factor completely determines the imaginary taps of the simulated LFSR. But, there is a trade-off between speed of the overall system and the versatility of the speed factors. Since the driving LFSRs are clocked at an integer multiple d of the system clock, the choice of the maximum speed factor to-

gether with the achievable speeds by current technology limit the maximum speed of the overall system. Therefore the suggested method may be of limited use in high-speed applications. Interestingly enough, in the early days of circuit integration, the researchers were concerned about finding combinations of LFSRs which were able to produce high-speed sequences at low clock rates of the participating LFSRs (Lemp 70). The idea behind the speed factor goes just in the opposite direction. Today's technology provides fast and cheap integrated circuits, so we might as well sacrifice some of the speed to gain additional security and flexibility. Another cryptographic application of clock manipulation is the so-called clock controlled LFSR, where one LFSR is used to provide the clock for a second one (Nyff 75, Beth 84, Goll 84). There the main objective is in generating sequences of large linear complexity.

6.1 The Simulated Linear Feedback Shift Register

Our interest is in analyzing the effects of the speed factor d on the behavior of the simulated LFSR. Let the original LFSR whose length is denoted by L, have irreducible connection polynomial C(D) in GF(q)[X], and let $r = r_0, r_1, r_2, \ldots$ be the sequence of digits from GF(q) produced by the original LFSR at the time instants of the high-speed clock. Suppose for illustration the chosen speed factor d is equal to 3. Then the output sequence r of the original LFSR and the sequence s observable at the system clock times (compare Fig. 6.1) are related as follows:

high-speed clock times	r_0	r_1	r_2	r_3	r_4	r_5	r_6	r_7	...
	↓			↓			↓		
system clock times	s_0			s_1			s_2		...

Thus clocking an LFSR d times faster than the surrounding system and taking an output symbol from the LFSR only at the system clock times corresponds to decimating the LFSR sequence by d. We have therefore identified the relation between the sequence $\tilde{s}$ generated by the simulated LFSR and the sequence $\tilde{r}$ produced by the original LFSR as the d-th decimation

$$s_j = r_{dj} \qquad j = 0,1,2,... \tag{6.1a}$$

or in sequence notation

$$\tilde{s} = \tilde{r}[d] \tag{6.1b}$$

The d-th decimation of a sequence is conveniently treated using the trace operator. So let c(X) denote the characteristic polynomial associated to the original LFSR, and assume for analytic convenience that c(X) is monic. Note that c(X) and C(D) are reciprocal polynomials, i.e. $c(D) = D^L C(D^{-1})$, in particular c(X) is irreducible in GF(q)[X] and has degree L. The roots of c(X) then lie in the extension field $GF(q^L)$. Let α be a root of c(X) and let $T_L(\beta)$ be the trace operator which maps $GF(q^L)$ into GF(q). Then the jth digit of the sequence $\tilde{r}$ produced by the original LFSR is conventiently described as

$$r_j = T_L(A\alpha^j) \qquad j = 0,1,2,... \tag{6.2}$$

where A denotes a coefficient from $GF(q^L)$ corresponding to the initial state of the original LFSR. Using the trace description of $\tilde{r}$ the effect of the d-th decimation is readily incorporated

$$s_j = r_{dj} = T_L(A\alpha^{dj}) \ . \tag{6.3}$$

It is illuminating to write

$$s_j = T_L(A\beta^j) \tag{6.4}$$

where $\beta = \alpha^d$. Consequently the sequence s satisfies the linear recursion associated to the minimum polynomial of β in $GF(q^L)$. The period T of the original LFSR is the smallest positive integer t such that c(X) divides X^T-1. Equivalently, T is the multiplicative order of α in $GF(q^L)$ and thus $\alpha, \alpha^2, \ldots, \alpha^{T-1}, \alpha^T = 1$ are the T distinct roots of X^T-1. By proper choice of d ($1 \leq d \leq T$), it follows that $\beta = \alpha^d$ can be selected as any root of X^T-1 and hence as a root of any monic irreducible polynomial that divides X^T-1. This subdivides the possible d's into equivalence classes according to the irreducible polynomials whose roots are the elements of the form $\beta = \alpha^d$. The multiplicative order of β in $GF(q^L)$ will be $T^* = T/\gcd(d,T)$ which also determines the degree L^* of the minimum polynomial

of β. For L^* is the smallest positive integer l such that $\beta^{q^l} = \beta$; but exponents of β are calculated modulo T^*, hence L^* is equal to the multiplicative order of q in Z_{T^*}, the ring of integers modulo T^*. Finally, since the minimum polynomial of $\beta = \alpha^d$ in $GF(q^L)$ is (according to (6.4)) identical to the characteristic polynomial $c^*(X)$ of the simulated LFSR, we have proven the following proposition which is a generalization of known results on the decimation of maximal-length sequences (Zier 59, Sarw 80).

Proposition 6.1. The simulated LFSR

Let $\tilde{r}$ be the sequence produced by the original LFSR whose characteristic polynomial c(X) is irreducible in GF(q) of degree L. Let α be a root of c(X) and let T be the period of c(X).
Let $\tilde{s}$ be the sequence resulting from sampling $\tilde{r}$ at intervals of d clock cycles, i.e. $\tilde{s} = \tilde{r}[d]$.
Then the simulated LFSR, that is, the LFSR which could directly produce s, has the following properties:

(1) the characteristic polynomial $c^*(X)$ of the simulated LFSR is the minimum polynomial of α^d in $GF(q^L)$

(2) the period T^* of $c^*(X)$ is equal to $T/\gcd(d,T)$

(3) the degree L^* of $c^*(X)$ is equal to the multiplicative order of q in Z_{T^*}.

Moreover, all d in C_k, where

$$C_k = \{k, kq, kq^2, \ldots\} \mod T$$

denotes the cyclotomic coset of k modulo T, result in the same simulated LFSR, except for different initial contents. Finally, every sequence producible by the simulated LFSR is equal to $\tilde{r}[d]$ for some choice of the initial contents of the original LFSR.

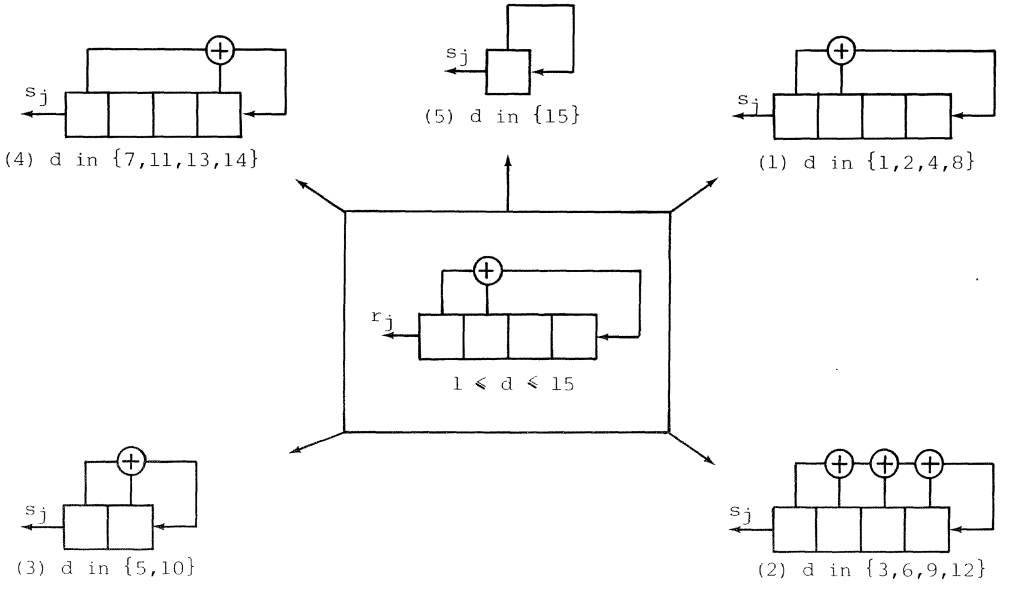


Fig. 6.2. The original LFSR with $c(X) = X^4 + X + 1$ shown in the box and the simulated LFSRs (1) - (5) as occurring by use of the corresponding speed factors.

For d = 3, T* = 5 = 15/gcd(3,15), and L* = 4 since the multiplicative order of 2 in Z_5 is 4.

For d = 5, T* = 3 = 15/gcd(5,15), and L* = 2 since the multiplicative order of 2 in Z_3 is 2.

Note that care has to be taken when the simulated LFSR has characteristic polynomial c*(X) of degree L* smaller than L. Then there might occur cases where the all-zero sequence is produced instead of a sequence with minimal polynomial c*(X). Let for instance $\tilde{r}$ be determined by (under the same assumptions as in the above example)

$$r_j = T_4(\alpha^5 \alpha^j) \tag{6.6}$$

When d is in 5,10 , the all-zero sequence will result instead of sequence $\tilde{s}$ with minimal polynomial $c^*(X) = X^2 + X + 1$. The reason being that α^5, as well as α^{10}, belong to the set of coefficients A that force $T_4(A\alpha^{5j})$ to be zero for all $j \geq 0$.

Proposition 6.1 provided a complete characterisation of the output sequence $\tilde{s}$ of the simulated LFSR in terms of the original LFSR and the speed factor d. But often in running-key generators also the individual stages of the LFSRs are interconnected. So let us investigate the relationship between the sequences observable in adjacent stages of the simulated LFSR. Let $\tilde{r}^k = r_k, r_{k+1}, r_{k+2}, \ldots$ denote the k-th phase of the sequence $\tilde{r} = r_0, r_1, r_2, \ldots$ produced by the original LFSR. It follows from (6.2) that

$$(\tilde{r}^k)_j = r_{j+k} = T_L(A\alpha^k\alpha^j) \tag{6.7}$$

so that the phase shift k can be read off by comparing the coefficients of α^j in the trace descriptions. When the original LFSR is clocked d times faster than the surrounding logic, then $\tilde{r}^k[d]$ is precisely the sequence observable at the kth stage of the simulated LFSR. Combining (6.3) and (6.7) we obtain

$$\begin{aligned}(\tilde{r}^k[d])_j = r_{dj+k} &= T_L(A\alpha^k\alpha^{dj}) \\ &= T_L(A\alpha^k\beta^j)\end{aligned} \tag{6.8}$$

where we have again replaced α^d by β (note that $\tilde{r}[d] = \tilde{r}^0[d]$). In general, the sequences $\tilde{r}^k[d]$, for k = 0, .., L-1, will not be phase shifts of one another; rather, they will be "cyclically distinct" sequences producible by the simulated LFSR. However, when gcd(d,T) = 1, so that the simulated and the original LFSRs have the same period, the sequences $\tilde{r}^k[d]$ will be phase shifts of one another. To see this, we note that gcd(d,T) = 1 implies that d has a multiplicative inverse e modulo T, i.e. there exists an integer e ($1 \leqslant e \leqslant T$) such that

$$\beta^e = \alpha^{de} = \alpha^{QT+1} = \alpha \tag{6.9}$$

and hence

$$de = QT + 1. \tag{6.10}$$

In this case, we can rewrite (6.8) as

$$(\tilde{r}^k[d])_j = T_L(A\beta^{ke}\beta^j) = (\tilde{s}^{ke})_j \tag{6.11}$$

which we recognize to be the ke-th phase shift of $\tilde{s}$. This proves the following Proposition.

Proposition 6.2.

Let $\tilde{r} = r_0, r_1, r_2, \ldots$ be a nonzero sequence of period T produced by the original LFSR whose characteristic polynomial is irreducible over GF(q) and has degree L. When the speed factor d is chosen such that gcd(d,T) = 1 then the sequence $\tilde{r}^k[d]$ observable at the k-th stage ($1 \leq k \leq L$) of the simulated LFSR is the e-th phase of the sequence $\tilde{r}^{k-1}[d]$ observable at the adjacent (k-1)st stage, where e ($1 \leq e < T$) denotes the multiplicative inverse of d modulo T.

When an LFSR is operated at normal speed there are only available L consecutive phases of the output sequence. To get access to phases outside this window requires under normal speed some additional hardware, i.e. adding two distinct shifts of the same sequence produces a third sequence with identical minimal polynomial, but of generally unknown phase. The practical import of Proposition 6.2 is that simulating an LFSR by multiple-clocking of another LFSR gives simultaneous access to widely separated phases of the sequence produced by the simulated LFSR. The speedfactor d can be used to adjust the phase difference between sequences observable at adjacent stages of the simulated LFSR.

It is known that L consecutive phases of a sequence with minimal polynomial of degree L are linearly independent. An interesting question is, whether the widely separated phase shifts $\tilde{s}$, $\tilde{s}^e$, $\tilde{s}^{2e}$, ..., $\tilde{s}^{(L-1)e}$ observable at the stages of the simulated LFSR (as described in Proposition 6.2) still are linearly independent. Assume the sequences $\tilde{s}^{ke}$, k = 0, ..., L-1, are linearly dependent. Then there would exist a_k, k = 0, ..., L-1, in GF(q) not all zero such that

$$\sum_k a_k (\tilde{s}^{ke})_j = 0 \qquad \text{not all } a_k = 0 \tag{6.12}$$

which using (6.11) can be rewritten as

$$\sum_k a_k T_L(A\beta^{ke}\beta^j) = 0 \qquad \text{not all } a_k = 0 \ . \tag{6.13}$$

Making use of the linearity of the trace operator we obtain

$$T_L(A\beta^j \sum_k a_k \beta^{ke}) = 0 \qquad \text{not all } a_k = 0 \tag{6.14}$$

which would imply that

$$\sum_k a_k \beta^{ke} = 0 \qquad \text{not all } a_k = 0 \tag{6.15}$$

Hence β^e would be the root of a nonzero polynomial over GF(q) with degree less than L. But this is impossible since $\beta^e = \alpha$ has a minimum polynomial of degree L. Thus we have proved.

Proposition 6.3.

> The sequences $\tilde{s}^{ke} = \tilde{r}^k[d]$, k = 0, ..., (L-1), observable at the stages of the simulated LFSR are linearly independent (under the assumptions made in Proposision 6.1).

The practical import of Proposition 6.3 is that no flexibility is lost when an LFSR is simulated by multiple-clocking of another LFSR, rather than directly implemented. Still any sequence producible by the simulated LFSR can be obtained by linear combinations of the contents of the actual LFSR being simulated.

6.2 A Random Number Generator Suggested by a Linear Cipher Problem

In (Mass 84a) it is shown that perfect linear cipher systems exist. Suppose the enciphering transformation F_k operating on the current and the previous M plaintext symbols (collected in $\underline{x}_j = [x_j, x_{j-1}, \ldots, x_{j-M}]$) produces the current ciphertext symbols y_j in the following way:

$$y_j = F_K(\underline{x}_j) = x_j + \sum_{i=1}^{M} c_i(j,K)x_{j-i} \qquad j = 0,1,2,\ldots \tag{6.16}$$

where the coefficients $c_i(j,K)$ depend on both the time instant j and the key K. (Note that all digits and operations are assumed to be in GF(q) and that the initial conditions $\underline{x}_0$ required for (6.16) may be chosen arbitrarily). F_K is a linear transformation as we note first

$$F_K(\underline{x}_j) + F_K(\underline{x}'_j) = x_j + x'_j + \sum_{i=1}^{M} c_i(j,K)[x_{j-i} + x'_{j-i}]$$

$$= F_K(\underline{x}_j + \underline{x}'_j) \tag{6.17}$$

and second

$$a\,F_K(\underline{x}_j) = ax_j + \sum_{i=1}^{M} c_i(j,K)ax_{j-i} = F_K(a\underline{x}_j) \tag{6.18}$$

for any scalar a from GF(q). A perfect cipher system is one in which for each allowable plaintext sequence $\tilde{x}$ every digit of the ciphertext sequence is statistically independent and identically distributed over the ensemble of randomly chosen keys. Since in the considered linear cipher, M consecutive zeroes in the plaintext sequence $\tilde{x}$ force the next ciphertext digit y_j to be equal to the next plaintext digit x_j (compare (6.16)) we see that perfect secrecy is only possible when the plaintext sequence is restricted never to contain M consecutive zeroes. To actually obtain perfect secrecy one may simply choose the coefficients $c_i(j,K)$ independently at random

from a uniform distribution over GF(q). The plaintext restriction then guarantees that at least one of the independent "key digits" appears with a nonzero multiplier in (6.16). But this would require M "key digits" to encipher just one plaintext symbol. The interesting question still open at the time is: What is the least amount of key (measured in digits of key per plaintext digit) required for a perfect linear cipher as described in the foregoing. In (Mass 84a), a specific linear cipher system is shown (see Fig. 6.3) that achieves perfect secrecy using two digits of key per plaintext digit, and it is conjectured that this much key is also necessary for all $M \geq 2$.

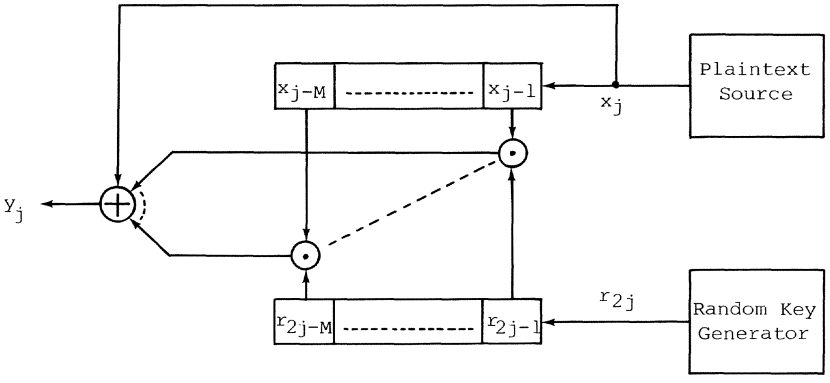


Fig. 6.3. A perfect linear cipher system, using a conjectured minimum of two random key digits per plaintext digit.

It is known since the work of Shannon (Shan 49) that the least amount of key required for perfect secrecy in any type of cipher system is one key digit per plaintext digit. It is illuminating to compare the simple one-time pad system where the random key sequence $\tilde{z}$ is directly added to the plaintext sequence $\tilde{x}$, to the linear cipher. The one-time pad is perfect, requires one key bit z_j per plaintext bit x_j, but is nonlinear since first

$$F_z(x_j) + F_z(x'_j) = x_j + z_j + x'_j + z_j$$
$$\neq x_j + x'_j + z_j = f_z(x_j + x'_j) \qquad (6.19)$$

and second

$$aF_z(x_j) = ax_j + az_j \neq ax_j + z_j = F_z(ax_j) \qquad (6.20)$$

Note that the linearity conditions can only be hypothetically evaluated on both the linear cipher system and the one-time pad system since different plaintexts are always enciphered under different parts of the random key sequence.

6.2.1. The Random Sequence Generator

A perfect secrecy system is of course an ideal random number generator since the ciphertext symbols are statistically independent and uniformly distributed random variables. This suggests that the basic structure in Fig. 6.3 may be of use in random sequence generation. For this purpose, it is natural to replace the plaintext source of Fig. 6.3 by an M-stage LFSR started in some nonzero state, as this automatically enforces the "plaintext restriction" (never are M consecutive zeroes are produced) as well as introduces some element of pseudo-randomness. It is a natural next step to approximate the random key generator of Fig. 6.3 by a second LFSR of length L ($L \geq M$) also started in some nonzero state but clocked at a speed 2 times, or more general, d times that of the "plaintext" LFSR. The additive component x_j, whose significance was in easing the deciphering process, may be considered to be optional. Finally, to be completely general, we may allow the "plaintext"-LFSR also to have associated its own speed factor d_2, whereas we call d_1 the speed factor of the lower LFSR. The resulting random sequence generator is shown in Fig. 6.4. Such a device might be used as a random number generator or as a key stream generator in a conventional stream cipher.

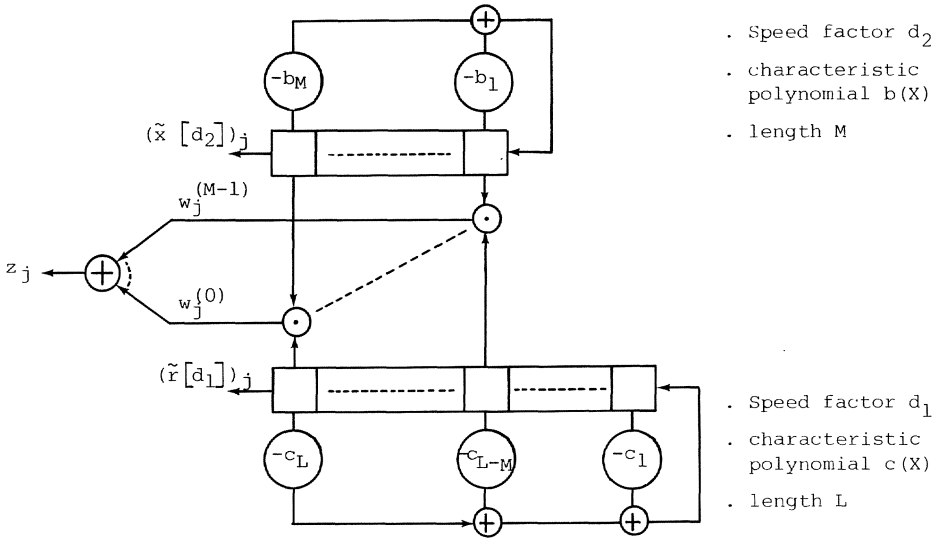


Fig. 6.4. The random sequence generator suggested by the linear cipher problem, employing multiple speed LFSRs.

6.2.2. Analysis of the Random Sequence Generator

With the tools developed on both the treatment of nonlinear combinations of periodic sequences and the treatment of multiply clocked LFSRs it is easy to determine the linear complexity, and period of the produced random sequence $\tilde{z}$. Both LFSRs are clocked according to their own speed factors. To guarantee that the randomizing effect induced by shifting the random key sequence at double speed in the perfect linear cipher is not lost in the suggested random sequence generator, we must require that the two speed factors be different. Consider the kth input sequence $\tilde{w}^k$ to the adder forming $\tilde{z}$; $\tilde{w}^k$ is the product of the sequences observable at the kth stage of both simulated LFSRs. Therefore

$$(\tilde{w}^k)_j = (\tilde{r}^k[d_1])_j \, (\tilde{x}^k[d_2])_j \, . \qquad (6.21)$$

Now suppose that the characteristic polynomials c(X) and b(X) of the two LFSRs in Fig. 6.4 are irreducible over GF(q), that α is a root of c(X) and β is a root of b(X), that α^{d_1} has the same multiplicative order T_1 in $GF(q^L)$ as α and that β^{d_2} has the same multiplicative order T_2 in $GF(q^M)$ as β. The imposed conditions merely guarantee that the simulated sequences in either LFSR have maximum period and maximum linear complexity with respect to the original LFSRs implemented. Since we are interested in the cryptographic applicability of the random sequence generator, we wish to use sequences with long periods and pseudo-randomness properties as well as keep the attainable linear complexity as high as possible. With the assumptions made, we can write for $\tilde{w}^k$ according to (6.8) using the trace operator

$$(\tilde{w}^k)_j = T_L(A\alpha^k(\alpha^{d_1})^j)\ T_M(B\beta^k(\beta^{d_2})^j) \tag{6.22}$$

where A and B are nonzero elements of $GF(q^L)$ and $GF(q^M)$ if, as we now assume, the initial states of the LFSRs are both nonzero. When gcd(L,M) = 1, the trace identity Lemma 5.11 implies that

$$(\tilde{w}^k)_j = T_{LM}(AB\alpha^k\beta^k(\alpha^{d_1}\beta^{d_2})^j) \tag{6.23}$$

and moreover, the product Lemma 5.12 assures that $\tilde{w}^k$ is a sequence with irreducible minimal polynomial of degree LM, and hence that the linear complexity of $\tilde{w}^k$ is

$$\Lambda(\tilde{w}^k) = LM\ . \tag{6.24}$$

In fact, we see from (6.23) that each sequence $\tilde{w}^k$, k = 0, .., (M-1), has the same irreducible minimal polynomial of degree LM. To prove that also their sum

$$\tilde{z} = \sum_{k=0}^{M-1} \tilde{w}^k \tag{6.25}$$

has the same linear complexity, it remains only to show that $\tilde{z}$ is not the all-zero sequence $\tilde{0}$. The linearity of the trace operator implies together with (6.23) that

$$z_j = T_{LM}(AB(\alpha^{d_1}\beta^{d_2})^j \sum_{k=0}^{M-1} (\alpha\beta)^k) \qquad (6.26)$$

which may result in the all-zero sequence if and only if $\alpha\beta$ is a root of the polynomial $1 + X + \ldots + X^{M-2} + X^{M-1}$ over GF(q). But this cannot be the case; the root algebra proposition 3.1 shows that the minimum polynomial of $\alpha\beta$ over GF(q) has degree LM. Since $\tilde{z}$ has the same irreducible minimal polynomial as has each of the sequences $\tilde{w}^k$, also the periods must coincide. Corollary 5.10 shows that the period of $\tilde{w}^k$ is at least $(T_1T_2)/(q-1)^2$ and hence is also the period of $\tilde{z}$. In particular when q = 2, the period of $\tilde{z}$ is equal to the product of T_1 and T_2. We may summarize the analytical results obtained in the following Proposition.

Proposition 6.4. Linear Complexity and Period of the Random Sequence Generator of Fig. 6.4.

When the original LFSRs employed in the random sequence generator have irreducible connection polynomials whose degrees L and M are relatively prime, and when the speed factors $d_1 \neq d_2$ are chosen such that $\gcd(d_1,T_1) = \gcd(d_2,T_2) = 1$, where T_1 and T_2 denote the periods of the original LFSRs then the output sequence $\tilde{z}$ will have linear complexity

$$\Lambda(\tilde{z}) = LM \qquad (6.27)$$

as will also each of the input sequences $\tilde{w}^k$, k = 0, ..., (M-1), to the adder that forms $\tilde{z}$ (provided of course that the original LFSRs are initially loaded with nonzero contents). Moreover, the period T of $\tilde{z}$ is lowerbound by

$$T(\tilde{z}) \geq \frac{T_1T_2}{(q-1)} . \qquad (6.28)$$

Note that the nonlinear combination of the 2 LFSRs, which can be viewed as the "inner product" of the states up to length M of the smaller LFSR, has memory M. It may seem strange that we were able to analyze the random sequence generator with the tools developed for

memoryless nonlinear combinations (see chapter 5). But it is often feasible to extend the results to the analysis of specific nonlinear combinations with memory. One might ask now the legitimate question: why bother considering such a complicated nonlinear combination as the "inner product" of the two LFSRs? Just the product of the two output sequences, or any of the $\tilde{w}^k$ feeding the adder in Fig. 6.4 would have exactly the same linear complexity and the same period as the actual output sequence $\tilde{z}$. Thus we would save a lot of hardware by replacing the "inner product" of the 2 states by a single product of two corresponding stages. The answer lies in the statistical properties of the produced output sequences. Suppose q = 2, then $\tilde{w}^k$ will exhibit a gross imbalance of 0's and 1's; with ideal binary random sequence generators one would expect the product to be 0 in 3/4 of the cases. With the actually implemented LFSRs there might occur some other short term "nonrandom" features in addition to the imbalance. On the other hand, the perfect linear cipher argument suggests that "almost" perfect distribution properties should result from replacing the plaintext source and the random key generator by pseudo-random sequence generators provided the plaintext restriction of no M consecutive 0's is obeyed. To analyze the statistical properties of the random sequence generator of Fig. 6.4 let us assume that both LFSRs have primitive characteristic polynomials and that q = 2. The "plaintext" LFSR all of whose stages are tapped to participate in the "inner product" may be identified to have a counter/selector function. Every nonzero binary vector of length M appearsss exactly once as state in a period of this LFSR and thereby enables the corresponding linear combination of sequences observable at the tapped stages of the lower LFSR. Proposition 6.3 assures that all sequences $\tilde{r}^k[d_1]$, k = 0, ..., (M-1), observable at the first M stages of the lower LFSR are linearly independent. Thus every distinct nonzero linear combination of these sequences will result in a nonzero distinct sequence. Since we assumed that $\tilde{r}[d_1]$ is a maximum-length sequence, it follows from proposition 6.2 that

$$\tilde{r}^k[d_1] = (\tilde{r}[d_1])^{ke_1} \qquad k = 0, \ldots, M-1 \tag{6.29}$$

is the ke_1-th phase of $\tilde{r}[d_1]$, where e_1 ($1 \leq e_1 < 2^L-1$) is the multiplicative inverse of d_1 modulo 2^L-1, and thus $\tilde{r}^k[d_1]$ is also a maximum-length sequence with identical minimal polynomial. The vector space of maximum-length sequences with identical minimal polynomials is closed under shifts, which implies that every nonzero linear com-

bination of linearly independent phases $\tilde{r}^k[d_1]$ is again a distinct phase of $\tilde{r}[d_1]$. We may summarize the above considerations in the following statement: When both connection polynomials are primitive and q = 2, then at time instants $j + n(2^M-1)$ the random sequence generator of Fig. 6.4 emits the $(j+n(2^M-1))$-th bit of that phase of $\tilde{r}[d_1]$ determined by the state of the "plaintext" LFSR at time j. Consequently, within the period $(2^M-1)(2^L-1)$ of the random sequence generator, 2^M-1 distinct phases of the maximum-length sequence $\tilde{r}[d_1]$ have been interleaved in such a way that each bit of each phase of $\tilde{r}[d_1]$ has been put out exactly once. The described viewpoint of considering the "plaintext"-LFSR to be a mere selector of distinct phases of the maximum-length sequence produced by the lower LFSR allows one to easily derive the exact distribution of zeroes and ones in the output sequence $\tilde{z}$ of the random sequence generator. Each maximum-length sequence of period 2^L-1 contains exactly 2^{L-1} ones and $2^{L-1}-1$ zeroes in one period. The interleaving argument now implies the following proposition.

Proposition 6.5. Bit distribution of the random sequence generator of Fig. 6.4.

When the random sequence generator described in Proposition 6.4 is further restricted to employ LFSRs of length L and M (L > M) with primitive connection polynomials over GF(2), then the number of 1's and 0's within one period $(2^M-1)(2^L-1)$ of the output sequence $\tilde{z}$ is

$$Nr(1) = (2^M-1)2^{L-1} \tag{6.30a}$$

$$Nr(0) = (2^M-1)(2^{L-1}-1) \tag{6.30b}$$

and the difference between Nr(1) and Nr(0) relative to the period length is

$$\frac{Nr(1) - Nr(0)}{(2^M-1)(2^L-1)} = \frac{1}{2^L-1} \tag{6.31}$$

It is intuitively pleasing to see the correspondence between Proposition 6.5 and the linear cipher system depicted in Fig. 6.3. In the linear cipher system the ideal random key generator was the cause for perfect secrecy, whereas in the suggested random sequence gene-

rator of Fig. 6.4 the lower LFSR (which replaced the ideal random key generator) now is responsible for the distribution properties of the output sequence. Proposition 6.5 tells us that the imbalance of 0's and 1's in the output sequence $\tilde{z}$ can be made arbitrarily small by increasing the length L of the lower LFSR, or equivalently, by improving the pseudo-randomness properties of the sequence produced by the lower LFSR.

The special viewpoint taken to derive the bit distribution of the random sequence generator is of some independent interest. We therefore want to give an illustration of the underlying interleaving process. Let the "plaintext"-LFSR which we identified in its counting/selecting function have primitive connection polynomial $C(D) = 1 + D^2 + D^3$ and speed factor $d_2 = 1$. Let the lower LFSR be unspecified except for simulating an LFSR with primitive connection polynomial of degree $L > 3$ and relatively prime to 3. The "inner product" then connects to the first three stages of the lower LFSR, where the sequences $\tilde{r}[d_1]$, $\tilde{r}^1[d_1]$ and $\tilde{r}^2[d_1]$ are observable. Each state of the selecting LFSR enables one particular linear combination of said sequences:

state of selecting LFSR	enabled linear combination	shorthand notation
100	$\tilde{r}[d_1]$	$\tilde{r}_1$
010	$\tilde{r}^1[d_1]$	$\tilde{r}_2$
110	$\tilde{r}[d_1] + \tilde{r}^1[d_1]$	$\tilde{r}_3$
001	$\tilde{r}^2[d_1]$	$\tilde{r}_4$
101	$\tilde{r}[d_1] + \tilde{r}^2[d_1]$	$\tilde{r}_5$
011	$\tilde{r}^1[d_1] + \tilde{r}^2[d_1]$	$\tilde{r}_6$
111	$\tilde{r}[d_1] + \tilde{r}^1[d_1] + \tilde{r}^2[d_1]$	$\tilde{r}_7$

All the sequences $\tilde{r}_i$, $i = 1, \ldots, 7$ are distinct nonzero phases of $\tilde{r}[d_1]$. The selecting LFSR with connection polynomial $C(D) = 1 + D^2 + D^3$ initially loaded with [100] will produce the periodic state sequence (1,4,2,5,6,7,3), where we used the integer representation of the binary statevector. The interleaving process evolves as follows: at time j the jth bit of sequence $\tilde{r}_k$ is connected to the output where k denotes the integer corresponding to the state of the selecting LFSR:

$$k = \begin{matrix} 1 & 4 & 2 & 5 & 6 & 7 & 3 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \end{matrix}$$

$$\tilde{z} = (\tilde{r}_1)_0, (\tilde{r}_4)_1, (\tilde{r}_2)_2, (\tilde{r}_5)_3, (\tilde{r}_6)_4, (\tilde{r}_7)_5, (\tilde{r}_3)_6, (\tilde{r}_1)_7, \ldots$$

Consequently, since $\gcd(2^3-1, 2^L-1) = 1$, (2^3-1) maximum-length sequences of period (2^L-1) are interleaved to result in an overall period of $(2^3-1)(2^L-1)$.

We must mention that, for $q > 2$, statistical results are not so equally easily obtained, the reason being that the periods necessarily must have a common factor $(q-1)$ (see Corollary 5.10). This shows that in general only every $(q-1)$st digit of any sequence $\tilde{r}_i$ produceable by a linear combination of stages of the lower LFSR will appear as digit in the output sequence $\tilde{z}$. Thus one must be able to describe the statistical properties of $(\tilde{r}[d_1])[q^M-1]$, the (q^M-1)st decimation of the d_1-th decimation of the maximum-length sequence $\tilde{r}$, in order to derive some statistical properties of the output sequence $\tilde{z}$.

6.2.3. Extensions and Comments

One could of course combine the output sequence $\tilde{z}$ of the random sequence generator of Fig. 6.4 in a second "inner product" with a third LFSR of length N. Since $\tilde{z}$ has no identifiable speed factor attached to it, it is sufficient to choose the speed factor d_3 of the third LFSR greater than 1. If $\gcd(LM,N) = 1$ then sequences of linear complexity LMN and period $(q^L-1)(q^M-1)(q^N-1)/(q-1)^2$ are obtainable. This process could be iterated as many times as desired. The interesting fact about the "inner product" combination as employed in the discussed random sequence generator is that it automatically enforces "almost" ideal distribution properties of the output sequence (the "almost" being caused by the pseudo-randomness of the utilized LFSR-sequences). It seems as if the presented method of producing random sequences also carries the potential of synthesizing LFSRs with irreducible but nonprimitive characteristic polynomials which possess a state cycle exhibiting "almost" ideal statistical properties (a state on the mentioned state cycle is synthe-

sized at the same time). This is remarkable since pseudo-randomness is usually tied to maximum-length sequences, that is sequences with a primitive minimal polynomial.

Multiple-clocking of LFSRs provides additional flexibility and security in cryptographic applications. Different feedback connections may be simulated without physically changing the feedback taps. The uncertainty about the recursion actually used in a random sequence generator renders more difficult the work of a cryptanalyst. In particular, multiple clocking also provides the means to discourage the cryptanalyst from trying a correlation attack (as described in chapter 5.3) on nonlinearly combined LFSRs, since he would have to search through all recursions possibly in use. The limits of applicability of speed factors lie between technologically attainable clock rates and desired system clock rates. When the speed factors in a running key generator are put under control of a secret key, such multiple-clocking gives an added "dimension" to secure sequence generator design.

7 The Knapsack as a Nonlinear Function

The classical knapsack problem (also referred to as subset problem or 0/1 knapsack problem) is the following: given a set of N weights $w_1, \ldots, w_N$ and a "test" weight T, determine whether there exists a subset of weights which exactly sums to T. More formally, the 0/1 knapsack defines a function from the set of binary N-tuples into the nonnegative integers:

$$F_w: \{0,1\}^N \rightarrow Z^+ \tag{7.1a}$$

with

$$S = F_w(\underline{x}) = x_1 w_1 + \ldots + x_N w_N = \sum_{j=1}^{N} x_j w_j \; . \tag{7.1b}$$

The knapsack weights $w_1, \ldots, w_N$ are positive integer-valued coefficients and S denotes the partial sum (or subset sum) obtained by summing those weights which are selected by the binary vector $\underline{x}$. In its modular version, the knapsack equation (7.1) is defined modulo a given modulus Q. Note that both the modular and the nonmodular knapsack function are identical if the modulus Q is chosen to be larger than the sum of all N weights. The knapsack function is in general neither onto, i.e. there exist integers S in the range of $F_w(\underline{x})$ not being obtainable by any subset of the weights, nor one-to-one, i.e. there exist integers S in the range of $F_w(\underline{x})$ such that more than one $\underline{x} \in \{0,1\}^N$ will result in the same partial sum S. For example, in the knapsack $\underline{w}$ = [14, 28, 56, 82, 90, 132, 197, 284, 341, 455], the integers S = 515, 516 and 517 have three, no, and a unique solution, respectively. (The solutions are $515 = w_1+w_2+w_6+w_9 = w_2+w_3+w_5+w_9 = w_1+w_4+w_5+w_6+w_7$, and $517 = w_1+w_2+w_3+w_5+w_6+w_7$). Complexity theory tries to classify problems according to their inherent difficulty (see e.g. Gare 79). A problem is called tractable if there exists a deterministic automaton (e.g. some realistic model of computation such as the Turing machine) which solves each instance of the problem in time at most p(n), where n is the input size needed to describe the problem instance and p(.) is a polynomial defined over R. The class P consists of all tractable problems, that is, all problems solvable in polynomial time on a deterministic computer. In order to classify also intractable (or "hard") problems, complexity

theory has introduced the hypothetical device of a nondeterministic automaton. A nondeterministic automaton is best thought of as a computer with unlimited parallelism. The class NP consists of all problems which can be solved in polynomial time on a nondeterministic automaton. To solve some problems in NP on a deterministic computer seems to require exponential time. As the size of the input increases, the solution of these problems becomes infeasible even on the fastest computers. Note that NP contains P since every problem solvable in polynomial time on a deterministic automaton may clearly be solved in polynomial time on a nondeterministic automaton. The big question in complexity theory is whether NP is a strict superset of P or not. Although intuitively it seems that many problems in NP are much harder than the problems in P, no one has yet proved that P $\neq$ NP. To give some theoretical support to the above question, complexity theory has collected the subset of "hardest" problems in NP into the class NP-complete. Their significance resides in the fact that, if only one problem in NP-complete could be solved deterministically in polynomial time, then all of them could, and consequently NP would be equal to P. More and more problems have been shown to be in NP-complete. The longer none of these problems is found to be solvable in polynomial time on a deterministic computer, the stronger grows the suspicion that NP must be a strict superset of P. The knapsack problem as stated in the beginning of this chapter belongs to the class of NP-complete problems (Gare 79). Note that there exist problems beyond NP-complete which require even on a nondeterministic automaton truly exponential time for their solution.

7.1 The Significance of the Knapsack for Secrecy Systems

In their 1976 paper, Diffie and Hellman (Diff 76) speculated that the theory of NP-completeness could be applied to the design of strong cryptosystems. They suggested basing a cryptosystem in such a way on an NP-complete problem that breaking the cryptosystem would correspond to solving the NP-complete problem. Problems being computationally more difficult than the problems in NP are not suitable for application in cryptography since encoding and decoding functions must be easy to compute, i.e. they must be tractable. Thus the best a designer of a cryptosystem can hope for is to make breaking the system equivalent to solving an NP-complete problem. Nevertheless care has to be taken since complexity theory deals only with worst-case measures, and thus does not guarantee a strong cipher in all situations.

The knapsack problem, as formulated in complexity theory, is an existence problem: given a positive integer T, does here exist a subset of the given weights whose sum S is T. This problem is difficult to apply in cryptography; one rather wants to use the knapsack as a one-way function. A function f(x) is loosely defined to be a one-way function when it is easy to compute f(x) for any x in the domain of f, while, for "almost" all y in the range of f, it is very hard to find an x such that y = f(x). The "almost" has to be included, since any algorithm can quickly evaluate f (provided f is public) for some arguments x and tabulate the corresponding function values y = f(x); whenever a y occurs that has been pretabulated, the corresponding argument x is easily read off the table. Note that the above definition does not require the one-way function f to be invertible. When the knapsack $F_w(\underline{x})$ is going to be applied as a one-way function in a cryptosystem, then the "easy" way from the domain into the range will be an enciphering transformation. Consequently the cryptanalyst is confronted with the hopefully intractable problem of finding an $\underline{x}$ such that $S = F_w(\underline{x})$. But in its use as a one-way function, the knapsack presents a slightly different problem (as compared to the complexity theory formulation): given a partial sum S, find one (or all) combinations of weights which sum to S, or in other words, given that a solution $\underline{x}$ exists to the knapsack equality (7.1), find it (or all of them). Here the question of complexity theory is already answered. Intuitively, one would say, to decide whether a solution to the knapsack problem exists or not, is

"easier" than to actually find the solution given that it exists. We can set this feeling on a theoretical foundation by proving the following result:

Proposition 7.1.

To find a solution of the 0/1 knapsack problem, given that it exists, is NP-complete.

Proof: Let π_e denote the problem of determining the existence of a solution and let π_s denote the corresponding problem of finding a solution, given that it exists. Let further A_e and A_s be the algorithms for checking the existence of a solution and for actually finding the solution.

We will show that $\pi_s \in P$ (the class of problems solvable in polynomial time on a deterministic computer) if and only if P = NP (the class of problems solvable in polynomial time on a nondeterministic computer).

Suppose now P = NP. Then A_e needs polynomial time to check the existence of a solution. Suppose further that a solution exists. A_s can now be set up as follows, using A_e.

A_s:

(1) $A_e(S-w_1) = x_1 = \begin{cases} 1, & \text{if } S-w_1 \text{ is a true partial sum} \\ 0 & \text{otherwise} \end{cases}$

(2) $A_e(S-x_1w_1-w_2) = x_2$

which can be repeated N times until a valid solution is found.

It follows that A_s needs only polynomial time, since it consists of N applications of a polynomial time algorithm. Thus π_s is in P when P = NP.

Suppose conversely that $\pi_s \in P$. Then there exists an algorithm A_s which computes the solution in polynomial time T(n) given that it exists. Thus $A_s(S)$ will return the correct solution (when it exists) within time T(n). $A_s(S)$ will return a

wrong solution or no solution in time T(n) when it does not exist (checking a solution can be done in polynomial time, too).

Hence $\pi_e \in P$ and P = NP, since π_e is NP-complete. Consequently π_s belongs to the class of NP-complete problems.

Proposition 7.1 guarantees that the knowledge about the existence of a subset of weights summing to the observed partial sum S does not decrease the inherent difficulty in the problem. From the complexity theory viewpoint, it is equivalent to decide for an arbitrary positive integer S whether it represents a true partial sum, or to find the actual weights that sum to a given true partial sum. The time to find a solution vector $\underline{x}$ such that $S = F_w(\underline{x})$ is believed to be exponential in the number of weights N. The best presently known algorithms to enumerate all possible solutions to a given partial sum S all have time complexity $0(2^{N/2})$. Horowitz and Sahni (Horo 74) published a 2-table search algorithm which based on dynamic programming principles and required $0(2^{N/2})$ memory. Schroeppel and Shamir (Schr 79) were able to lower the storage requirements to $0(2^{N/4})$ by generating on line the 2 basic tables from smaller tables. Most recently Karnin (Karn 84) derived an algorithm requiring $0(2^{N/6})$ processors and memory cells. In order to apply the algorithm of Horowitz and Sahni also to modular knapsacks, the algorithm has to be modified since it makes strong use of the ordering of partial sums. A simple extension capable of finding the solutions of a modular knapsack transformation could be the following: (a) Compute the basic 2 tables as if there was no modulos Q, (b) apply the basic algorithm to the N partial sums S, S+Q, ..., S+(N-1)Q. Note that S, if not reduced modulo Q, is bounded from above by NQ because every weight is drawn from the ring of integers modulo Q. Thus solving modular knapsacks has at most slightly increased time complexity $0(N2^{N/2})$ and same storage complexity, as compared to solving nonmodular knapsacks. But the time complexity only provides an upperbound on the time required to solve any instance of the knapsack problem. There certainly exist instances which may be solved in much less time (e.g. the superincreasing knapsacks have a time complexity 0(N)).

Apart from the knapsack function there are other apparent one-way functions used in cryptography. The most prominent examples being

factoring of integers which is the basis of the RSA public-key system (Rive 78) and taking logarithms which is the basis of the Diffie-Hellman public-key distribution system (Diff 76) as well as of the Pohling-Hellmann cipher (Pohl 78) and of the OM-lock (Mass 83). But neither of these is related to an NP-complete problem. Hence we may assert that the knapsack function is the closest to a provable one-way function that is known at the present. Moreover, from the operational viewpoint it offers the advantages of being an appealingly simple and very fast transformation. Consequently one should think that the knapsack problem is an ideal source for cryptographic applications. But interestingly enough, the knapsack has perhaps the worst "reputation" of all presently known one-way functions. Why is this so? First, some researchers claim that the knapsack is an inherently weak function because it is linear, and linearity is always said to be the curse of the cryptographer. Indeed the knapsack is linear when we define the underlying number system to be the integers. To see this, let $\underline{x}^{(1)}$ and $\underline{x}^{(2)}$ be two N-dimensional binary selection vectors, and let S_1 and S_2 be the corresponding partial sums,

$$S_1 = \sum_{i=1}^{N} x_i^{(1)} w_i = F_w(\underline{x}^{(1)})$$

$$S_2 = \sum_{i=1}^{N} x_i^{(2)} w_i = F_w(\underline{x}^{(2)}) \qquad x_i^{(1)},\ x_i^{(2)} \in \{0,1\} \qquad (7.2)$$

Then the sum of the two "ciphertexts" is equal to the "ciphertext" of the sum of the 2 "plaintexts", i.e.

$$S_1 + S_2 = F_w(\underline{x}^{(1)}) + F_w(\underline{x}^{(2)}) = F_w(\underline{x}^{(1)} + \underline{x}^{(2)}) = F_w(\underline{x}^{(3)}) \qquad (7.3)$$

where $x_i^{(3)}$ now are drawn from the alphabet $\{0,1,2\}$. Thus when we allow the input vectors $\underline{x}$ to contain positive integer components, and when we define their addition to be integer addition, then the knapsack function will be a linear mapping in the sense that it obeys the principle of superposition.

Nevertheless the knapsack realizes also a nonlinear function. Suppose now the vectors $\underline{x}^{(1)}$ and $\underline{x}^{(2)}$ are drawn from the vector space of binary N-tuples. Then, in general

$$S_1 + S_2 = F_w(\underline{x}^{(1)}) + F_w(\underline{x}^{(2)}) \neq F_w(\underline{x}^{(1)} \oplus \underline{x}^{(2)}) = F_w(\underline{x}^{(3)}) \qquad (7.4)$$

where we have indicated addition in $[GF(2)]^N$ by $\oplus$. Clearly (7.4) tells us that the knapsack is a nonlinear mapping. We may conclude that the number system or field from which the selection variables x_i are considered to be taken plays a key role in the determination of the (non)-linearity of the knapsack. In fact, the incompatibility of integer addition as realized by the knapsack and the field operations as defined for the input variables, may be used to introduce a fast and easily implemented nonlinear function (see Fig. 7.1).

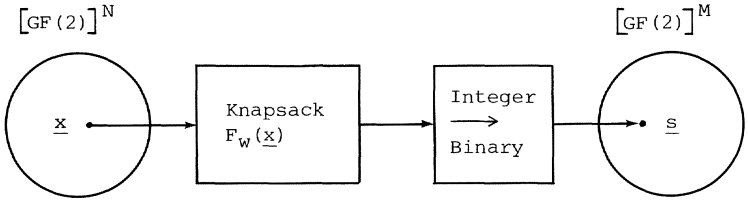


Fig. 7.1. The knapsack as nonlinear mapping between binary vector spaces

The main reason for the extremely bad repute of the knapsack is probably best understood from the history of public-key cryptography. To make the knapsack applicable to public key cryptography it has to be restricted in two ways. First, only if the knapsack function is one-to-one, is unique deciphering guaranteed (it seems also possible to allow non-injective mappings and to separate the correct solution from the others on basis of redundancy; but nobody known to the author has ever investigated such a system). Second, to make possible an easy inversion of the knapsack function for the legitimate users, some secret "trapdoor" has to be inserted into the one-way function allowing a short-cut solution with the secret deciphering key. A trapdoor one-way function may be considered as a family of invertible functions $\{f_z\}$ indexed by a parameter z (the trapdoor) such that for known z it is easy to find simple and fast enciphering

and deciphering algorithms that easily compute f_z and f_z^{-1}, but if only the enciphering algorithm is known it is very difficult to compute $f_z^{-1}(y)$ for almost all $y = f_z(x)$. The two restrictions imposed on the knapsack function have the disadvantage that they destroy its provable complexity. It is not known whether the isolated class of one-to-one knapsacks is inherently easier to solve, nor is it known whether determining the trapdoor may be made an NP-complete problem. Once the trapdoor is found, inverting the trapdoor knapsack is a polynomial time problem.

In their original paper, Merkle and Hellman (Merk 78) suggested to implant a trapdoor by starting from an easy one-to-one knapsack (e.g. a superincreasing knapsack) and then to conceal the easy knapsack by a simple modular transformation. Let $\underline{b} = [b_1, \ldots, b_N]$ be the superincreasing knapsack, i.e.

$$b_j > \sum_{i=1}^{j-1} b_i \ . \tag{7.5}$$

The modular transformation suggested by Merkle and Hellman consists of multiplying the easy knapsack weights by a multiplier A and reducing the result modulo M to destroy the superincreasing structure of the knapsack weights,

$$w_j = R_M(Ab_j) \qquad j = 1, \ldots, N \ . \tag{7.6}$$

The modulus M is required to be larger than the sum of all "easy" weights and A is chosen relatively prime to M. These conditions automatically enforce the transformed (e.g. presumably hard) knapsack to be one-to-one. Merkle and Hellman's proposal stimulated many research efforts, most of them directed towards breaking their trapdoor knapsack scheme. Shamir (Sham 82) was the first to present a polynomial-time attack. But it is in fact not too surprising that the Merkle-Hellman cryptosystem was broken. For the trapdoor is based on a linear principle. To see this, let $F_b(\underline{x})$ be the easy knapsack function employing the superincreasing weights. Now the knapsack scheme works since

$$A\ F_b(\underline{x}) \equiv F_b(A\underline{x}) \pmod{M} \tag{7.7}$$

which is the homogeneity condition for linear systems. It was also found that for cryptanalysis it is not necessary to determine the exact trapdoor b, A and M employed in the cryptosystem; usually there exist many b', A', M' allowing proper reconstruction of the plaintext x. It is interesting to note that even after 6 years of research the specialists do not agree precisely where the weakness of the trapdoor knapsack scheme stems from. The tendency nowadays seems to be to hold the knapsack itself responsible for all the flaws, and to excuse the modular transformation from guilt (Bric 83). The basic argument is that modular randomizers have proven to be secure in other public-key systems (e.g. the RSA system). But this argument neglects that modular exponentiation as used in the RSA system is a nonlinear operation whereas the modular scaling of the weights may be considered to be a linear operation. In our opinion, it is not appropriate to condemn the knapsack in general. A further indication that the considered types of knapsacks do not reflect the general properties of the knapsack is given by Brickell and Simmons (Bric 83). They were able to upperbound the probability that a randomly selected set of N weights would form a superincreasing sequence by

$$P_s < 2^{-\binom{N}{2}} \tag{7.8}$$

and the probability that a randomly selected set of N weights would be images under a single modular transformation of a superincreasing sequence by

$$P_{1M} < 2^{-\binom{N}{2}} \left(\sum_{i=1}^{n} w_i \right)^2 . \tag{7.9}$$

These probabilities show that the cryptographically considered classes of knapsack functions are in fact arbitrarily small compared to the total set of N weight knapsacks. But with the passage of time the knapsack and the modular transformation have seemed to become inseparable in the minds of people. A good illustration of this is provided by Desmedt et al. (Desm 84), they simply term useless for cryptography all those knapsacks which cannot be obtained by a countable number of generalized modular transforms (in particular, this would exclude all non-injective knapsacks from being of any use

in cryptography). But certainly, the applicability of the knapsack is neither bound by the modular transformation nor by the one-to-one requirement. Cryptography has a clear need for true one-way functions: Massey and the author (Ruep 84b) showed the application of the noninvertible knapsack for authentication purposes. In chapter 8 we will show how a noninvertible knapsack provides flexible and powerful means to generate keystreams of high linear complexity. It may also be the case that the knapsack will experience a renaissance in two-key cryptography when there are found better ways to insert the trapdoor information. The preceding discussion of the trapdoor knapsack should underline our belief that much of the bad repute of the knapsack function is due to the particular way it has been used in cryptography.

Our objective in considering the knapsack as a true one-way function is two-fold. First, we want to exploit its provable complexity (i.e. NP-completeness) to assure the computational infeasibility of finding input vectors $\underline{x}$ that could have resulted in a given partial sum S. Second, we are interested in the potential of the knapsack as a highly nonlinear but flexible and easily realizable transformation. In its use as a true one-way function, the knapsack is no longer required to be public. Thus it might be the case that some or all the weights belong to the key.

If we drop the assumption of given weights, interesting classes of problems arise. First, suppose it is possible to observe the outputs S of a knapsack system without having information about the weights or about the corresponding input vectors $\underline{x}$. Is it possible to determine with this little information the set of weights employed in the knapsack system (or some equivalent set of weights)? And if it is possible, what are the conditions?

Before we can answer these questions we have to give some precise notions.

The weights $w_1, w_2, \ldots w_N$ are called <u>linearly independent</u> if

$$\sum_{j=1}^{N} c_j w_j = 0 \qquad \text{implies } c_j = 0 \text{ for } j = 1, 2, \ldots, N \tag{7.10}$$

where the c_j are elements of the alphabet $\{-1,0,+1\}$. Note that this definition also holds for the modular knapsack when the addition is taken modulo Q.

This definition can immediately be used to prove the following Lemma:

Lemma 7.2.

The knapsack function F is one-to-one if and only if the weights are linearly independent.

Proof Suppose F is one-to-one. Then

$$\sum a_j w_j \neq \sum b_j w_j \qquad \text{all } \underline{a} \neq \underline{b} \text{ in } \{0,1\}^N$$

$$\rightarrow \sum (a_j - b_j)\, w_j \neq 0 \qquad \text{all } \underline{a} \neq \underline{b} \text{ in } \{0,1\}^N$$

$$\rightarrow \sum c_j w_j \neq 0 \qquad \text{all } \underline{c} \text{ in } \{-1,0,+1\}^N \ \underline{c} \neq \underline{0}$$

Suppose conversely that the weights are linearly dependent.

Then

$$\sum_j c_j w_j = 0 \qquad \underline{c} \text{ in } \{-1,0,+1\}^N \qquad \underline{c} \neq \underline{0}$$

Let

$$a_j = \begin{cases} 1 \text{ if } c_j = 1 \\ 0 \text{ otherwise} \end{cases} \qquad b_j = \begin{cases} 1 \text{ if } c_j = -1 \\ 0 \text{ otherwise} \end{cases}$$

$$\rightarrow \sum a_j w_j = \sum b_j w_j \qquad \text{all } \underline{a} \neq \underline{b} \text{ in } \{0,1\}^N$$

→ Knapsack is not one-to-one.

Note that the same result also holds for the modular knapsack.

If the non-modular knapsack is one-to-one, it gives in a certain sense always the most information about the weights, since for a given partial sum there is exactly one single combination of weights which could have produced this sum. Thus it is not surprising that we are able to prove the following result for one-to-one knapsacks.

Proposition 7.3.

> If the knapsack is non-modular and one-to-one, then the set of weights $\{w_1,\ldots,w_N\}$ is uniquely determined by the set of all partial sums.

Since we are not only able to prove the above proposition but can also give an algorithm which actually determines the weights, we will first state the algorithm and then prove by induction that the obtained set of weights is unique.

Weight Determining Algorithm

Let the knapsack be non-modular and one-to-one. Assume that the weights are ordered according to their size: $w_1 < w_2 < \ldots w_N$.

Define $A(n_1,n_2,..,n_i) = \{n_1,n_2,..,n_i,n_1+n_2,..,n_1+n_2+..+n_i\}$ to be the set of all non-zero partial sums of i intergers $n_1,..,n_i$.

Then the algorithm whose flowgraph is given below will solve for the knapsack weights. $S_1 = A(w_1,\ldots,w_N)$ denotes the initially available set of all distinct partial sums.

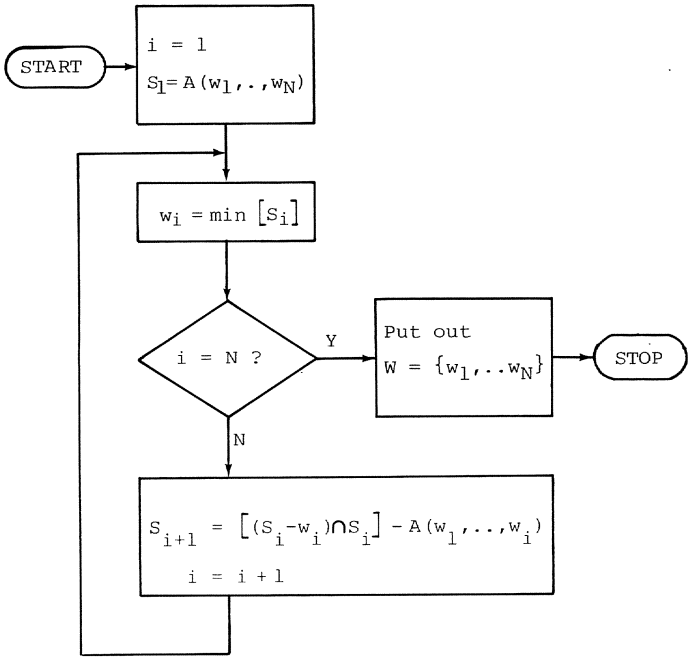


Fig. 7.2. Flowgraph of weight determining algorithm

This algorithm requires $0(N2^N)$ time, since it steps N times through the loop, and each time it has to process sets of size $0(2^N)$. Note that it is not clear if the weight determining problem still is in the class of NP-complete problems or if it is in the harder class of truly exponential problems. For suppose we guess a set of weights and want to check whether it produces the given set of partial sums. On any deterministic computer, this verification seems to require exponential time, which would mean that the problem is even more difficult than any problem in NP.

Proof of Propostion 7.3.

Suppose the unique weights $w_1 < w_2 < \ldots < w_i$ are already determined. We will show that then by the above algorithm the next weight w_{i+1} is uniquely determined, if the knapsack is one-to-one.

(1) No other element than a true partial sum is candidate for the next weight. Assume that $S_i \subset S_1$.

$$S_{i+1} = [(S_i - w_i) \cap S_i] - A(w_1,..,w_i) \qquad (7.11)$$

The intersection $I = (S_i - w_i) \cap S_i$ clearly is a subset of S_i

$$S_{i+1} = I - A(w_1,..,w_i) = I \cap A^c(w_1,..,w_i)$$
$$\rightarrow S_{i+1} \subset I \subset S_i \subset S_1$$

(2) S_{i+1} contains all partial sums made up from weights $w_{i+1},..,w_N$, in particular w_{i+1} is contained in S_{i+1}.

Assume $A(w_i,..,w_N) \subset S_i$. Then

$$[(S_i-w_i)\cap Si] \supset [A(w_i,..,w_N)-w_i)\cap A(w_i,..,w_N)] \supset A(w_{i+1},..,w_N)$$

since

$$A(w_i,..,w_N) = \{w_i, w_i+A(w_{i+1},..,w_N),\ A(w_{i+1},..,w_N)\}$$

Thus

$$A(w_{i+1},..,w_N) \subset S_{i+1} \text{ if } A(w_1,..,w_i) \cap A(w_{i+1},..,w_N) = 0$$

that is if the knapsack is one-to-one.

(3) The smallest element in S_{i+1} is w_{i+1} and it is unique.

$S_{i+1} \subset S_1 - A(w_1,..,w_i)$ by recursive rule (7.11)

Assume there is an element

$a < w_{i+1}$ contained in $S_1 - A(w_1,..,w_i)$

No element w_k, $k = i+1,..,N$ can have contributed to a, hence $a \varepsilon A(w_1,..,w_i)$ which is a contradiction.

And since the weights are linearly independent over $\{-1,0,+1\}$ there can only be one smallest element w_{i+1}.

To complete the inductive proof we state the initial conditions:

$$(1)\quad S_2 \subset S_1 \qquad \text{since } S_1 \subset S_1 \qquad (7.12)$$

$$(2)\quad A(w_2,..,w_N) \subset S_2 \qquad \text{since } A(w_1,..,w_N) = S_1 \qquad (7.13)$$

$$(3)\quad \min[S_1\} = w_1 \qquad \text{since } w_1 < w_2 <...< w_N \qquad (7.14)$$

Note that $A(w_1,..,w_i)$ in the basic recursion (7.11) which is needed to clean the set S_{i+1} of residual elements, could be replaced by the smaller set $w_i,\ w_i + A(w_1,..,w_{i-1})$ since S_i cannot possibly contain any element in $A(w_1,..,w_{i-1})$ by the same recursion (7.11).

Let us illustrate the weight determining algorithm (WDA) by means of some little examples:

(1) Let the given set of partial sums S_1 be

$$S_1 = \{1,5,6,7,8,12,13\} \rightarrow \min S_1 = w_1 = 1$$

$$(S_1-1) \cap S_1 = \{5,6,7,12\} = S_2 \rightarrow \min S_2 = w_2 = 5$$

$$(S_2-5) \cap S_2 = \{7\} = S_3 \rightarrow \min S_3 = w_3 = 7$$

Since there was no linear dependency among the weights involving coefficients c_j of absolute value greater than one (compare (7.10)), it was never necessary to clear S_{i+1} of residual elements by subtracting $A(w_1,...,w_i)$.

(2) Let S_1 contain

$$S_1 = \{1,2,3,4,5,6,7\} \rightarrow \min S_1 = w_1 = 1$$

$$S_2 = [(S_1-1) \cap S_1] - A(1)$$
$$= \{1,2,3,4,5,6\} - \{1\} \rightarrow \min S_2 = w_2 = 2$$

$$S_3 = [(S_2-2) \cap S_2] - A(1,2)$$
$$= \{1,2,3,4\} - \{1,2,3\} \rightarrow \min S_3 = s_3 = 4$$

Here linear dependencies among the weights not covered by the one-to-one requirement (7.10) (e.g. $2w_2 - w_4 = 0$) made necessary the residual clearing.

(3) Let S_1 contain

$$S_1 = \{1,3,4,5,7,8,10,11,12,14,15\} \rightarrow \min S_1 = w_1 = 1$$

$$S_2 = [(S_1-1) \cap S_1] - A(1)$$
$$= \{3,4,7,10,11,14\} - \{1\} \rightarrow \min S_2 = w_2 = 3$$

$$S_3 = [(S_2-3) \cap S_2] - A(1,3)$$
$$= \{4,7,11\} - \{1,3,4\} \rightarrow \min S_3 = w_3 = 7$$

$$S_4 = [(S_3-7) \cap S_3] - A(1,3,7) = 0$$

This example illustrates that the WDA does not find the correct set of weights when the knapsack is not one-to-one. But note, if the residual clean by $A(w_1, \ldots, w_i)$ is dropped, then the WDA will find the correct set of weights in the above example despite the non-one-to-oneness of the knapsack.

The weight determining algorithm shown in Fig. 7.2 has the property that it completely eliminates in step i the influence of w_i, e.g. S_{i+1} will not contain any partial sum to which w_i has contributed. The basic WDA may be simplified if at step i only those partial sums are eliminated which are possibly smaller than w_{i+1}. Under the assumption that the weights are ordered in increasing size, i.e. $w_1 < w_2 < \ldots < w_N$, and that the weights are linearly independent, only partial sums from $A(w_1, \ldots, w_i)$ could be of smaller value than w_{i+1}.

Consequently, the central recursion (7.11) could be replaced by

$$S_{i+1} = S_i - \{w_i, w_i + A(w_1,\ldots,w_{i-1})\} \qquad (7.15)$$

thus defining the modified weight determining algorithm. Although S_{i+1} still contains partial sums that incorporate weights $w_j < w_{i+1}$, it is guaranteed by (7.15) that all partial sums smaller than w_{i+1}

are eliminated. The modified WDA has the same time complexity $0(N2^N)$ as the basic WDA, but in the application it will be inherently faster. The main difference between the two versions is that the modified WDA will always fail when the knapsack is not one-to-one, whereas the basic WDA with the residual clearing removed will also find the weights for many non-one-to-one knapsacks.

At this place we have to make an important remark. Although the above algorithm can determine the true set of weights $\{w_1, w_2, \ldots, w_N\}$ employed in the knapsack system, it cannot possibly determine the actual weight vector $\underline{w} = [w'_1, \ldots, w'_N]$. The reason being that no information about the positions of the weights is contained in the partial sums. Thus from the partial sums we can only determine the actual knapsack system within a random permutation on the positions.

In most application of the knapsack as a true one-way function, it is not necessary or desirable to publish the weights. This is the reason why we considered the question of how much information about the weights is leaking out when we are able to continuously monitor the output of the knapsack. Probably, a knapsack with secret weights can be kept much smaller in the size for a given amount of intractability. Then the weight determining algorithms may become feasible. But possibly the situation arises where some of the input vectors $\underline{x}$ and their corresponding partial sums S are available. We may call this the unknown weight, but known input problem.

The knapsack equation (7.1) may also be written as the innerproduct of the binary input vector $\underline{x}$ and the weight vector $\underline{w}$ with the operation defined over the integers:

$$S = \sum_{j=1}^{N} x_j w_j = \underline{x}^t \underline{w} \tag{7.16}$$

where the vectors are taken in column representation.

The problem of determining the unknown $\underline{w}$ from N partial sums $S_1, \ldots, S_N$ and their corresponding input vectors $\underline{x}$ belongs to linear algebra. We can form a system of linear equations.

$$\begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ \vdots \\ S_N \end{bmatrix} = \begin{bmatrix} \underline{x}_1^t \\ \underline{x}_2^t \\ \vdots \\ \vdots \\ \underline{x}_N^t \end{bmatrix} \quad \underline{w} \leftrightarrow \underline{S} = X\underline{w} \tag{7.17}$$

which has a unique solution $\underline{w}$ if the N binary vectors $\underline{x}_1, \ldots, \underline{x}_N$ are linearly independent over the integers, or equivalently, if X has full rank N. To actually obtain the solution requires about $0(N^3)$ computations.

In the modular case, equation (7.16) is defined over the ring of integers modulo Q:

$$S \equiv x^t \underline{w} \pmod{Q} \tag{7.18}$$

and the system of linear equations (7.17) turns into a system of linear congruences,

$$\underline{S} \equiv X\ \underline{w} \pmod{Q} \ . \tag{7.19}$$

It is possible to transform (7.14) to a system of linear equations by introducing N new variables $\underline{v} = [v_1,\ldots,v_N]$

$$\underline{S} \equiv X\ \underline{w} - QI\underline{v} \tag{7.20}$$

where I denotes the identity matrix. If X has full rank N, then the dimension of the solution space is N and there exists an infinite number of solutions to (7.20). Clearly we are interested in $R_Q(\underline{w}) = [R_Q(w_1),\ldots,R_Q(w_N)]$, the vector of the remainders of the weights modulo Q, since this allows us to replace the true knapsack vector $\underline{w}$ by the equivalent vector $R_Q(\underline{w})$ which produces the same partial sums modulo Q as the original one.

The system of linear congruences (7.19) is defined over Z_Q, the ring of integers modulo Q. Thus it has a unique solution $R_Q(\underline{w})$ whenever X has rank N over Z_Q, or equivalently, whenever det(X) is relatively prime to Q (that is, whenever det(X) is a unit of Z_Q.

Consequently, if one wants to keep secret the weights of a knapsack, whether modular or not, one also has to hide the binary input vector $\underline{x}$ to the knapsack. Otherwise only M linearly independent vectors $\underline{x}_1$, .., $\underline{x}_N$ are needed to solve uniquely for the weights and their positions in the knapsack system.

We want to conclude this commentary section by summarizing that, with current knowledge, the knapsack function comes the closest to a provably complex one-way function known. The knapsack belongs to the class of NP-complete problems which are the "hardest" one-way functions available. The best a cryptographer can hope for is to make breaking his cryptosystem equivalent to solving an NP-complete problem. Yet the cryptographer must be aware how much "one-wayness" he may expect from a practical knapsack system. Suppose, for example, a 0/1 knapsack system is limited to N = 50 weights of 50 bits each. To describe the instance of the knapsack problem n = 2500 bits are needed. The best presently known algorithms need exponential time $0(2^{N/2})$ in the number of weights, which results in $T(n) \tilde{\sim} 2^{25}$ time for the considered knapsack problem. It follows that the actual running time of the algorithm is only $T(n) = n^{2 \cdot 2}$ for n = 2500. But computing a partial sum requires N=50 operations which needs time $n^{0 \cdot 5}$ for n = 2500. The reason for the not very large difference between the time to compute a partial sum from the weights and the time to compute a solution from a partial sum is due to the restriction of the knapsack to a practical size.

7.2 Addition is a Cryptographically Useful Function

The knapsack possesses the potential of realizing a nonlinear function with respect to GF(2) (as mentioned in the commentary section 7.1). If this function has desirable features such as large order (i.e. high degree of nonlinearity) or good distribution properties it could possibly be applied as nonlinear state filter or a nonlinear combiner (compare chapter 5). This idea deserves highest attention since first, the knapsack is a very simple and fast transformation (i.e. integer addition), but second, it is also a very flexible transformation (the weights are at our disposal). Thus, the problem of implementing nonlinear functions could be solved very easily - by using integer addition. But for the moment these remarks are speculations; before we can draw any conclusions we must find a way of describing integer addition in terms of GF(2) (Ruep 85a). Let us first consider the following subproblem which will open the door to the general answer.

Proposition 7.4.

Let S be the integer sum of N binary variables $b_1, b_2, .., b_N$

$$S = \sum_{i=1}^{N} b_i \tag{7.21}$$

and let

$$S = s_0 + s_1 2 + s_2 2^2 + \ldots + s_r 2^r \tag{7.22}$$

be the binary representation of S.

Then

$$s_i = \overline{\|}^{2^i}(b_1, b_2, \ldots, b_N) \tag{7.23}$$

where

$$\overline{\|}^{k}(b_1, \ldots, b_N) = \sum_{1 \leq i_1 < \ldots < i_k \leq N} b_{i_1} b_{i_2} \ldots b_{i_k} \tag{7.24}$$

denotes the GF(2)-sum of all distinct kth-order products of N binary variables.

Proof: Lucas' theorem (Berl 68) tells us that

$$\binom{A}{C} \equiv \binom{a_0}{c_0} \binom{a_1}{c_1} \ldots \binom{a_r}{c_r} \pmod 2 \tag{7.25}$$

where A and C are integers with binary representations $a_0 + a_1 2 + \ldots + a_r 2^r$ and $c_0 + c_1 2 + \ldots + c_r 2^r$, respectively.

Consequently,

$$\binom{S}{2^i} \equiv s_i \pmod 2 \tag{7.26}$$

There are $\binom{N}{k}$ ways to choose a subset of k variables from the total set of N variables $\{b_1, \ldots, b_N\}$.
Thus $\pi^k(b_1, \ldots, b_N$ denotes the sum of $\binom{N}{k}$ product terms each product having order k.

When exactly S of the N binary variables $b_1, \ldots, b_N$ take on the value 1, then $\binom{S}{k}$ of the product terms contribute a 1 to $\pi^k(b_1, \ldots, b_N)$. Therefore,

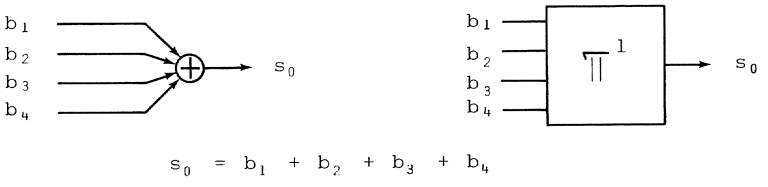
$$\overline{\|}^{2^i}(b_1, \ldots, b_N) = s_i \text{ in GF(2).} \tag{7.27}$$

N binary variables can sum to at most N. Hence their sum is always representable by $\lfloor \log N \rfloor + 1$ bits (or equivalently $r = \lfloor \log N \rfloor$ in (7.22)). Proposition 7.4 shows now the binary representation of the integer sum of N binary variables can be computed directly in GF(2) (using only addition and multiplication in GF(2)).

To illustrate this principle, we make an example.

Suppose N = 4, then 3 bits suffice to represent the integer sum of $b_1, \ldots, b_4$.

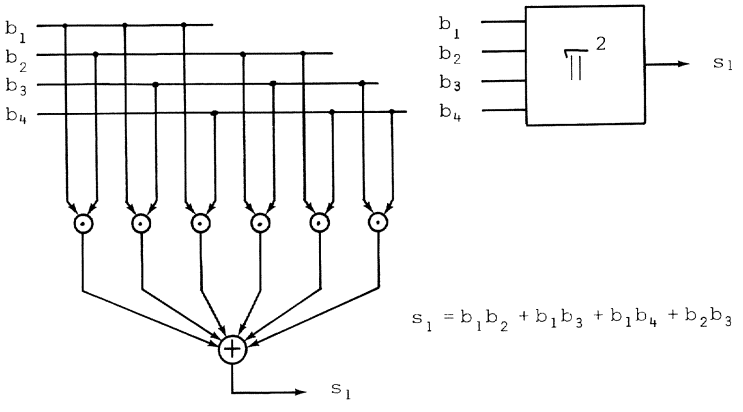
The least-significant bit of the sum, s_0, is the modulo-2 sum of the b_i.



$$s_0 = b_1 + b_2 + b_3 + b_4$$

Fig. 7.3. Least significant bit function

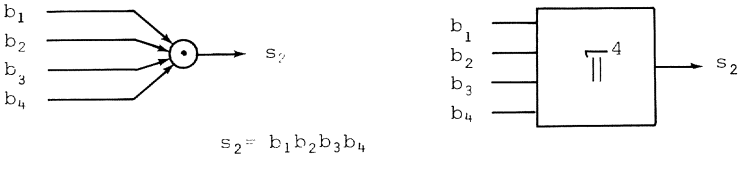
The second-least-significant bit, s_1, is the mod-2 sum of all second-order products of the b_i.



$$s_1 = b_1b_2 + b_1b_3 + b_1b_4 + b_2b_3 + b_2b_4 + b_3b_4$$

Fig. 7.4. Second-least significant bit function

The most significant bit, s_2, is the mod 2 sum of all 4th-order products of the b_i; with 4 variables there is only one 4th-order product.



$$s_2 = b_1b_2b_3b_4$$

Fig. 7.5. Most significant bit function

s_2 can be 1 if and only if all the b_i are 1, but in this case s_0 and s_1 are 0, as is easily seen by substituting the values into the formulas. (Note that the π-boxes realize the functions as defined by (7.24)).

The generalization of proposition 7.4 to the sum of N integers B_1, ...,B_N is now obvious and can be done straightforwardly. Let the ith integer B_i have the binary representation

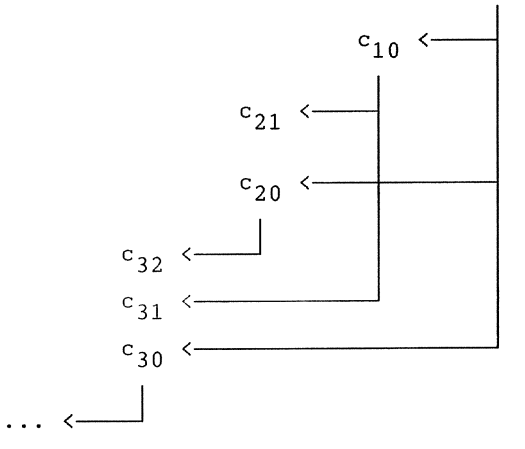
$$b_{i0} + b_{i1}2 + b_{i2}2^2 + \ldots + b_{ir}2^r \qquad i = 1,\ldots,N \tag{7.28}$$

and let S be the sum of N integers,

$$S = \sum_{i=1}^{N} B_i \ . \tag{7.29}$$

Write down the binary vectors corresponding to the N integers in consecutive rows of an array.

$$\begin{array}{lcccccc}
B_1 & \leftrightarrow & b_{1r} & \ldots & b_{13} & b_{12} & b_{11} & b_{10} \\
B_2 & \leftrightarrow & b_{2r} & \ldots & b_{23} & b_{22} & b_{21} & b_{20} \\
\vdots & & & & & & & \vdots \\
B_M & \leftrightarrow & b_{Nr} & \ldots & b_{N3} & b_{N2} & b_{N1} & b_{N0}
\end{array}$$



The addition of the least significant bits b_{i0}, i=1,...,N, directly corresponds to the problem treated in proposition 7.4. The higher order bits of this sum must now be treated as carries c_{10}, c_{20}, ... to the more significant bits. The sum of the jth significant bits then must comprise all the carries produced by the sums of the kth significant bits, k < j, which affect the jth column $\underline{b}_j$. We obtain for the jth bit s_j of the sum S,

$$s_j = \overline{\|}^{1}(\underline{b}_j) + \sum_{k=0}^{j-1} c_{jk} \qquad j = 0,\ldots,r+\lceil \log N \rceil \tag{7.30}$$

with the carry c_{jk} from the kth bit sum being recursively defined as

$$c_{jk} = \overline{\|}^{2^{j-k}}(\underline{b}_k, c_{k0}, \ldots, c_{k(k-1)}). \tag{7.31}$$

There are at most $\lceil \log N \rceil$ carry bits which have to be taken into account in (7.30) and (7.31), since the integer sum of an N-bit column $\underline{b}_j$ plus the maximum integer carry from the previous column $\underline{b}_{j-1}$ is always less than 2N.

To illustrate the influence of the carries, we compute the sum $S = B_1+B_2$ of 2 integers in GF(2)-arithmetic. Let B_1, B_2 be representable by 3 bits (e.g. $B_i = b_{i0} + b_{i1}2 + b_{i2}2^2$, i = 1,2). Then S is representable by 4 bits and log2 = 1 carry bits contribute from one bit sum to the next. Fig. 7.6 shows the complete block diagram with the π^k-functions defined according to (7.24).

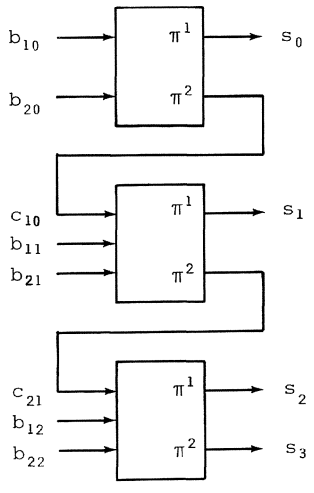


Fig. 7.6. GF(2)-description of the integer sum of two 3-bit integers

We note that every sum bit s_j (except for the least-significant bit s_0) depends through the carries in a nonlinear fashion on all the input bits of order lower than j. Thus integer addition is in fact a highly nonlinear operation (when defined in GF(2) (Ruep 85)). This may sound counterintuitive, but it is caused by the incompatibility of the two number systems. With this background developed on the nonlinear nature of addition with respect to GF(2), it is now easy to find the equivalent GF(2)-description of the knapsack.

7.3 The Knapsack in GF(2)-Arithmetic

To extend the integer sum to a knapsack we have to multiply each integer, w_i, by a binary variable x_i

$$S = \sum_{i=1}^{N} x_i w_i \tag{7.32}$$

The final expression for the jth bit of the partial sum (7.32) then becomes, using (7.30) and (7.31),

$$s_j = \overbrace{\|}^{1}(w_{1j}x_1, \ldots, w_{Nj}x_N) + \sum_{k=0}^{j-1} c_{jk}(\underline{x}) \tag{7.33}$$

where

$$c_{jk}(\underline{x}) = \overbrace{\|}^{2^{j-k}}(w_{1k}x_1,\ldots,x_{Nk}x_N,c_{k0}(\underline{x}),\ldots,c_{k,k-1}(\underline{x})). \tag{7.34}$$

Instead of proceeding with the above (awkward) formalism, let us illustrate the principle by means of an example. The following block diagram shows the operation of a 4 weight knapsack whose modulus is chosen to be 16.

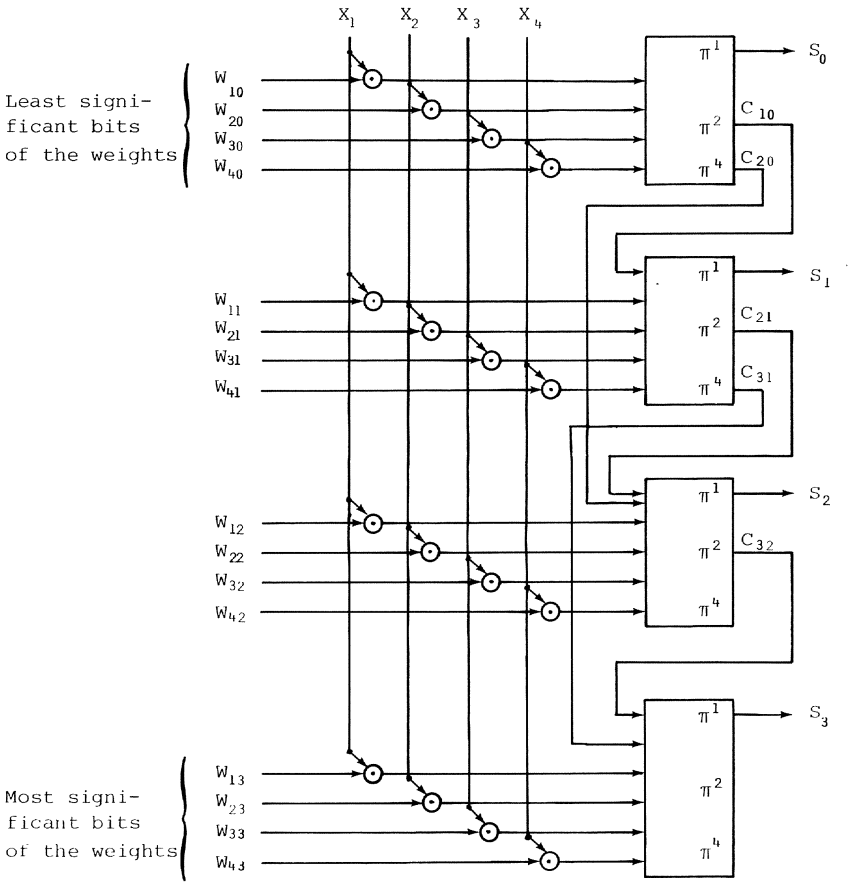


Fig. 7.7. Complete GF(2)-description of an N=4 weight, modulus Q=16 knapsack

In the knapsack application, the integer weights are no longer variables as they were in the pure integer sum problem. Now the weights can be considered as parameters; they are chosen once and for all. The variables of the knapsack problem are the binary selectors $x_1, x_2, .. x_N$. We may represent this by writing

$$s_j = f_{j,K}(\underline{x}) \tag{7.35}$$

for the boolean function evaluating the jth bit of the partial sum S, where K denotes the knapsack weights $w_1, w_2, .., w_N$ and $\underline{x} = [x_1, ..., x_N]$ is the binary selection vector.

From Fig. 7.7, it is apparent that the order of $f_{j,K}(\underline{x})$ increases with j. We would like to determine the exact way in which the order of $f_{j,K}(\underline{x})$ depends on j, when the knapsack is nonmodular (or when the modulus is a power of 2, which simply means that the overflow bits of S are dropped). The highest order term in $f_{j,K}(\underline{x})$ for suitable weights will be

$$c_{j0} = \overbrace{\| \; _K \; (\underline{x})}^{2^j} \tag{7.36}$$

as long as $2^j \leq N$, or $j \leq \lfloor \log N \rfloor$. Now suppose $c_{j,j-1}$ has order 2^j-1, then the product $c_{j0}c_{j,j-1}$ will have order $2^{j+1}-1$ when $j < \lfloor \log N \rfloor$ since both functions $c_{j0}(\underline{x})$ and $c_{j,j-1}(\underline{x})$ are symmetric in $\underline{x}$.

But

$$f_{j+1,K}(\underline{x}) = \overbrace{\| \; _K \; (\underline{x})}^{1} + \sum_{n=0}^{j} c_{j+1,n}(\underline{x}) \tag{7.37}$$

and $c_{j+1,j}$ contains the product $c_{j0}(\underline{x})c_{j,j-1}(\underline{x})$ and hence has order $2^{j+1}-1$ (when $j < \lfloor \log N \rfloor$). Since $c_{21}(\underline{x})$ has order 3, we conclude that $c_{j,j-1}$ has order 2^j-1 for $1 < j < \lfloor \log N \rfloor$. This proves the following proposition.

Proposition 7.5.

Let the knapsack equation

$$S = \sum_{i=1}^{N} x_i w_i$$

be defined over the integers, or over the ring of integers modulo a power of 2. When the knapsack equation is transformed into GF(2)-arithmetic, then the function $f_{j,K}(\underline{x})$ producing the jth bit s_j of the binary representation of the partial sum S has order

$$\text{ord}[f_{j,K}(\underline{x})] \leq \min\{2^j, N\} \; . \tag{7.38}$$

The simulation results reported in chapter 8 for the knapsack stream cipher give a strong indication that the bound (7.38) for randomly chosen weights is typically quite tight.

As an illustration, we compute the bound (7.38) for the orders of the binary functions producing the jth output bit of a knapsack with N=29 weights and a modulus $Q=2^{29}$.

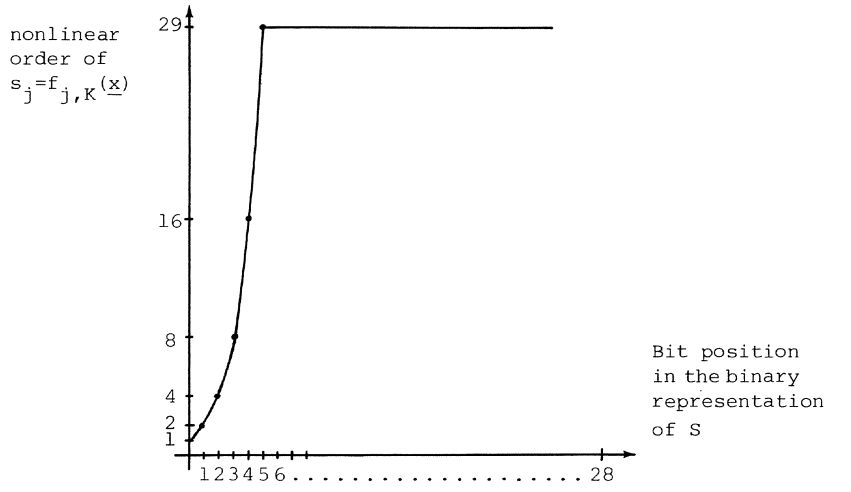


Fig. 7.8. The maximum nonlinear order of $s_j=f_{j,K}(\underline{x})$ as function of the position in $S = s_0 + s_1 2 + \ldots + s_{28}2^{28}$ when N=29 and $Q=2^{29}$

For a modulus which is not a power of 2, we cannot limit the influence of the knapsack weights on the generation of s_j to those weight bits w_{in}, $1 \leq i \leq N$, $n \leq j$, which have the same or smaller significance as s_j. In this case, we expect every function $f_{j,K}(\underline{x})$ to have maximum order - a suspicion which was confirmed by simulation results (see chapter 8.4).

8 The Hard Knapsack Stream Cipher

The combination of nonlinear functions with LFSRs provides a good means for generating running keys (see chapter 5). The major problem is the difficulty of implementing complex and truly random nonlinear functions. What we would like to have is a simple and fast implementation of a large set of highly nonlinear functions which is readily indexed by the key. Moreover, any function randomly chosen from the large available set should, with virtual certainty, produce a sequence with good distribution properties. As we have seen in the preceding chapter, the knapsack provides an ideal instrument to meet these requirements, except perhaps for a slightly reduced operational speed when compared to direct or table implementations. The maximum order of most sum-bit functions $f_{j,K}(\underline{x})$ assures that large linear complexity may be attained. The knapsack also distributes good statistics in the sense that equally distributed selector variables $x_1,\ldots,x_N$ will, with virtual certainty, result in equally distributed bits s_j of the binary representation of the partial sum S when the modulus is a power of 2 (or close to a power of 2). In the original knapsack problem the weights are public. When the knapsack is used to realize a nonlinear mapping, there is no reason to give this information away; giving it away can only make the attack easier. From the theoretical viewpoint, it is also appealing that the amount of key introduced by keeping the weights secret greatly increases the unicity distance. The weights directly specify the coefficients of the realized nonlinear functions. Thus to allow a truly random key choice, the knapsack should have sufficient size. This suggests an application of the knapsack as a nonlinear state filter (Ruep 85a).

8.1 System Description

The key idea is to replace the nonlinear feedforward function which is applied to the individual stages of the LFSR and which we do not know how to realize efficiently) by a knapsack (which we know how to realize efficiently) whose input vector is taken to be the state of the LFSR and whose output, the integer partial sum, is converted to its binary representation to form the key stream. Fig. 8.1 illustrates the principle.

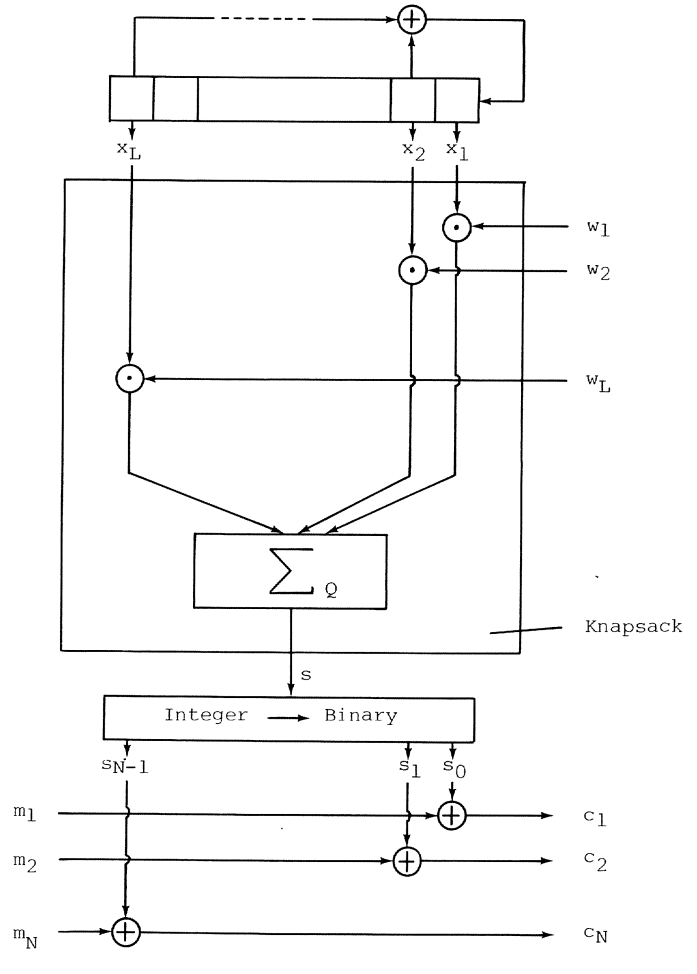


Fig. 8.1. Conceptual Knapsack Stream Cipher

With one operation of the knapsack, $N = \lceil \log Q \rceil$ simultaneous key stream bits are produced, which can be used to encipher a block of N message bits. Nevertheless, the device is a true stream cipher which enciphers every N message bits under a different block of the key-stream. To use a knapsack as feedforward transformation is particularly appealing since it is a provably complex one-way function. We must choose the modulus of the knapsack to be a power of 2 or close to a power of 2, say $Q = 2^N-1$, for the key stream bits to be equally distributed. If we do not, then not all binary N-tuples may occur, and thus the resulting distribution will be biased.

8.2 Analysis of the Knapsack Stream Cipher

In the GF(2) description of a knapsack, we have N binary input variables $x_1,..,x_N$ and M binary output variables $s_0,s_1,..s_{M-1}$. Note that the variable s_j is the output of stage j of the knapsack stream cipher as shown in Fig. 8.1. For the rest of this section, assume that M=N or, equivalently, that the modulus $Q=2^N$, since this results in the maximum number of evenly distributed output variables. Moreover, when the modulus is a power of 2, no division is required to compute the remainder of the partial sum S modulo Q, which increases the achievable speed. Let us first concentrate on the sequence s_{jn}, n = 0,1,2,.., observable at the jth output stage when the driving LFSR and the knapsack are operated in synchronism. For most randomly selected knapsacks, (7.38) will hold with equality or near-equality, so we can use (7.38) as a tight upperbound on the order of the actually chosen output function at stage j. We may now combine the knowledge on the linear complexity of sequences produced by non-linearly filtered LFSRs (see section 5.1) with proposition 7.5 to obtain:

Property 8.1. Linear complexity of the sequence $\tilde{s}_j$ produced at the jth output stage of the knapsack stream cipher.

Let the LFSR which drives the knapsack stream cipher have length L and primitive connection polynomial. Let the knapsack have L weights and be computed modulo 2^L.

Then $\tilde{s}_j$ will have linear complexity

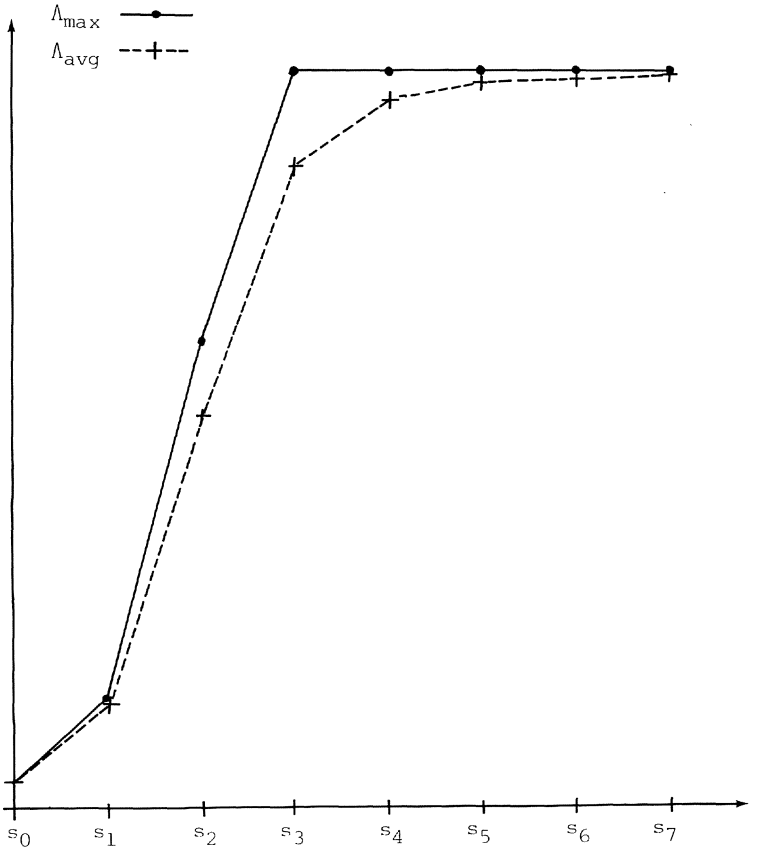
$$\Lambda(\tilde{s}_j) \leq \sum_{i=1}^{2^j} \binom{L}{i} \qquad j < \lceil \log L \rceil \tag{8.1}$$

$$\Lambda(\tilde{s}_j) \leq \sum_{i=1}^{L} \binom{L}{i} = 2^L - 1 \qquad j \geq \lceil \log L \rceil \tag{8.2}$$

where equality or near-equality will hold for most the randomly selected knapsacks.

Property 8.1 shows that the sequences observable at all but the first $\lceil \log L \rceil$ output stages of the knapsack stream cipher will have linear complexity close to the period length of the LFSR, which is the best we can hope for. It is unfeasible to attack such a sequence by means of a linear equivalent when L is about 50 (which corresponds to a linear complexity of about 10^{15}). Property 8.1 shows also that the $\lceil \log L \rceil$ least significant bits of a mod 2^L knapsack are cryptographically weaker than the higher order bits. It is even possible to quantify exactly their weakness in terms of the linear complexity of the produced output sequences, or equivalently, in terms of the order of the functions producing the output sequences. We conclude that the knapsack provides a virtually ideal structure for generating highly nonlinear functions which is, in addition, particularly easy to implement.

To illustrate the above assertions we compare in Fig. 8.2 the theoretically obtainable upperbound for the linear complexity of $\tilde{s}_j$ with the experimentally obtained average linear complexity of $\tilde{s}_j$ for L = 8 (that is, the m-LFSR has length 8, the knapsack has 8 weights and is computed modulo 2^8). In the experiment a randomly chosen knapsack was appended to the LFSR with primitive connection polynomial $C(D) = D^8 + D^6 + D^5 + D^4 + 1$, and the linear complexity was computed for each $\tilde{s}_j$, j = 0, ..., 7. The experiment was repeated 100 times to yield the average linear complexity displayed.



Λ_{max}	8	36	162	255	255	255	255	255
Λ_{avg}	8	35	135	219	241	249	249	251

Fig. 8.2. Theoretical upperbound Λ_{max} (according to (8.1), (8.2)) and experimental average Λ_{avg} for $\Lambda(\tilde{s}_j)$ when L = 8.

Although the presented example is very small, it confirms the assertions made in property 8.1.

Key's upperbound (5.30) states that, if a nonlinear filter function f of order k is applied to a maximum-length LFSR, the linear complexity of the resulting output sequence $\tilde{s}$ is bounded from above by

$$\Lambda(\tilde{s}) \leq \sum_{i=1}^{k} \binom{L}{i}$$

where L denotes the length of the LFSR. But this conversely implies that, if $\Lambda(\tilde{s})$ is found to lie in the range

$$\sum_{i=1}^{k-1} \binom{L}{i} < \Lambda(\tilde{s}) \leq \sum_{i=1}^{k} \binom{L}{i}$$

then the nonlinear order of f must be at least k. Applying this reasoning to the data from the above experiment we obtain the following average lower bound for the nonlinear order of $f_{j,K}$.

j	0	1	2	3	4	5	6	7
$E[ordf_{j,K}] \geq$	1	1.09	3.63	5.55	6.38	6.81	6.93	7.19
ord_{max}	1	2	4	8	8	8	8	8

These results are in good correspondence with the theoretical results obtained in chapter 7.3 (compare Proposition 7.5).

A question that the cryptographer is especially interested in is the effective key size. The key determines the initial state of the driving LFSR and the complete set of weights. Since it is possible to generate any sequence of length 2^L-1 by applying the corresponding nonlinear function to the stages of a length L LFSR with arbitrary primitive connection polynomial and arbitrary state, the initial state of the LFSR loses significance as a key paramenter. Thus the effective key size is determined by the number of different nonlinear functions which can be specified by choice of the knapsack weights. But note that this does not mean that the initial state of

the LFSR also loses its cryptographic significance in the operation of the stream cipher. Once the weights are determined, the state allows one to jump between arbitrary points within the period, and thus provides in this sense a valuable operational parameter. Let us now return to the question of the number of different nonlinear output functions $f_{j,K}(\underline{x})$ which can be specified by choice of the knapsack weights.

Recall from (7.33) that

$$f_{j,K}(\underline{x}) = \overline{\|}^{1}(w_{1j}x_1, \ldots, w_{Nj}x_N) + \sum_{n=0}^{j-1} c_{jn}(\underline{x})$$

with $c_{jn}(\underline{x})$ being recursively defined as

$$c_{jn}(\underline{x}) = \overline{\|}^{2^{j-n}}(w_{1n}x_1,\ldots,w_{Nn}x_N,c_{n0}(\underline{x}),\ldots,c_{n,n-1}(\underline{x})).$$

Because of the recursive definition of the function defining $c_{jn}(\underline{x})$, it is in general difficult to determine the exact number $Nr(f_{j,K})$ of distinct such functions. But to find a lowerbound on $Nr(f_{j,K})$ we may extract those parts of the function which have immediate and independent influence. The number of distinct choices for the function defining $c_{jn}(\underline{x})$ is clearly lowerbound by number of distinct choices for $\pi^{2^{j-n}}(w_{1n}x_1,..,w_{Nn}x_N)$; therefore $Nr(f_{j,K})$ is lowerbounded by the number of distinct choices for

$$\overline{\|}^{1}(w_{1j}x_1,\ldots,w_{Nj}x_N) + \overline{\|}^{2}(w_{1,j-1}x_1,\ldots,w_{N,j-1}x_N) + \ldots$$

$$+ \overline{\|}^{2^{\lfloor \log N \rfloor}}(w_{1,j-\lfloor \log N \rfloor}x_1,\ldots,w_{N,j-\lfloor \log N \rfloor}x_N). \qquad (8.3)$$

The linear terms in (8.3) provide 2^N different functions; the second order terms in (8.3) provide $2^N-\binom{N}{0}\binom{N}{1}$ different functions, since at least two weight bits must be nonzero to produce a second order contribution to the overall function (8.3). In general the terms of order 2^n in (8.3), $0 \leq n \leq \lfloor \log N \rfloor$, provide

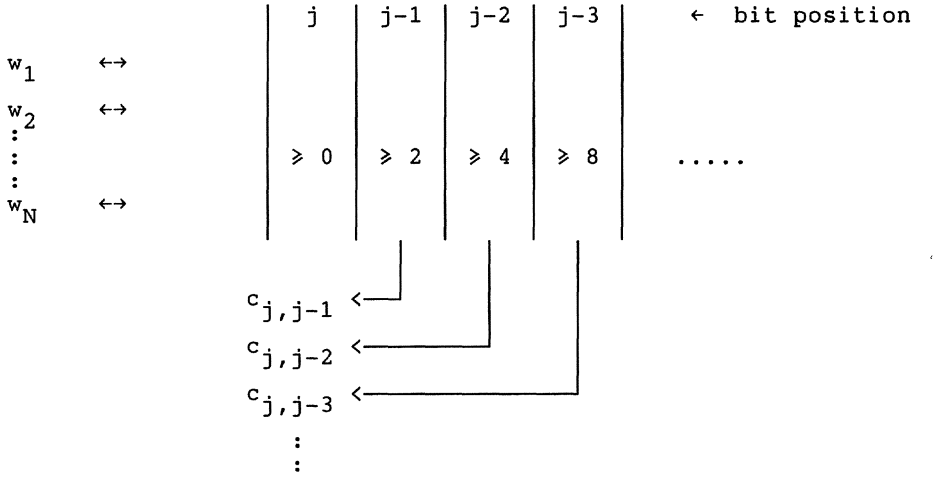
$$2^N - \sum_{i=0}^{2^n} \binom{N}{i} \tag{8.4}$$

different functions, since at least 2^n weight bits must be different from zero to produce a 2^nth-order contribution to the overall function.

Thus a lowerbound for $Nr(f_{j,K})$ is given by

$$Nr(f_{j,K}) \geq 2^N \prod_{n=1}^{\lfloor \log N \rfloor} \left[2^N = \sum_{i=0}^{2^n-1} \binom{N}{i} \right] \qquad j \geq \lfloor \log N \rfloor \ . \tag{8.5}$$

The bound (8.5) may be illustrated graphically as follows:



The column of weight bits $\underline{w}_{j-1}$ at position j-1 will directly contribute a carry $c_{j,j-1}$ to s_j when its Hamming weight is at least 2, and in general the column $\underline{w}_{j-n}$, $n \leq \lfloor \log N \rfloor$, will directly contribute a carry $c_{j,j-n}$, when its Hamming weight is at least 2^n. To get a handier but less tight bound than (8.5), we notice that

$$2^N - \sum_{i=0}^{2^n} \binom{N}{i} > 2^{N-2^n} \qquad n > 0 \tag{8.6}$$

which corresponds to the number of choices left when we consider the 2^n necessary nonzero bits in column $\underline{w}_{j-n}$ to be fixed.

Therefore

$$Nr(f_{j,K}) > 2^N \prod_{n=1}^{\lfloor \log N \rfloor} 2^{N-2^n}$$

$$= 2^{N(\lfloor \log N \rfloor + 1)} \; 2^{-\left(\sum_{n=1}^{\lfloor \log N \rfloor} 2^n \right)}$$

$$\geqslant 2^{N(\lfloor \log N \rfloor + 1)} \; 2^{-2^{\lfloor \log N \rfloor + 1}}$$

$$\geqslant 2^{N(\lfloor \log N \rfloor + 1)} \; 2^{-2N}$$

$$= 2^{N(\lfloor \log N \rfloor - 1)} \; . \qquad (8.7)$$

This proves the following proposition.

Proposition 8.2. Effective Keysize

> Let the knapsack employed in the knapsack stream cipher have N weights and modulus 2^N. Then the effective keysize for output stage i, $\lfloor \log N \rfloor \leqslant i \leqslant N$, is at least $N(\lfloor \log N \rfloor - 1)$ or, equivalently, by choice of the key at least $2^{N(\lfloor \log N \rfloor - 1)}$ different functions are specified for stage i, $\lfloor \log N \rfloor \leqslant i \leqslant N$.

The above proposition tells us that it is not much easier to attack a single output stage than to attack the complete knapsack stream cipher. As an example, take N=53; then the number of available functions for output stage i, $i \geqslant 5$, is at least $6.6 \cdot 10^{63}$, whereas the number of different knapsacks is 10^{845}. But for comparison, the number of distinct boolean functions of 53 variables is about $10^{10^{15}}$.

It is worth noting that the subset of available functions comprises, by the nature of the knapsack, with few exceptions only functions having good distribution properties. Thus we obtain what is desirable from the cryptographic viewpoint, namely limiting the set of all possible functions to a subset of functions with good distribution properties.

So far we have investigated the properties of the sequences observable at individual output stages. But these sequences are interleaved to produce the effective key stream. We might now ask about the linear complexity of the effective key stream. When sequences are interleaved, it is convenient to use their formal power series representation. Let $Z(D)$ be the formal power series associated with the binary sequence $\tilde{z} = z_0, z_1, z_2, \ldots$. When the sequence $\tilde{z}$ can be generated by an LFSR with connection polynomial $C(D)$, then $Z(D)$ can be expressed as fraction $Z(D) = P(D)/C(D)$ where the degree of $P(D)$ is less than the degree of $C(D)$.

From property 8.1, we know that the linear complexity of the jth output sequence $\tilde{s}_j$ is close to the period length 2^L-1 of the driving LFSR. Let $C_j(D)$ be the connection polynomial of the linear equivalent associated with $\tilde{s}_j$. Then

$$S_j(D) = \frac{P_j(D)}{C_j(D)} \qquad \deg[P_j(D)] < \deg[C_j(D)]. \tag{8.8}$$

Since, for generation of the key stream $\tilde{z}$, the bits of $\tilde{s}_j$ appear in positions $j+kN$ of $\tilde{z}$, we may describe that part of the key stream produced by $\tilde{s}_j$ by

$$D^j S_j(D^N) = D^j \frac{P_j(D^N)}{C_j(D^N)} . \tag{8.9}$$

Consequently the keystream $\tilde{z}$ has the formal power series

$$Z(D) = \sum_{j=0}^{N-1} D^j \frac{P_j(D^N)}{C_j(D^N)} . \tag{8.10}$$

When we combine these partial fractions to one fraction $Z(D) = P(D)/C(D)$, then the connection polynomial $C(D)$ of the linear equivalent of the keystream $\tilde{z}$ will be the least common multiple of all $C_j(D^N)$

$$C(D) = \text{lcm}\{C_1(D^N),\ldots,C_N(D^N)\} \tag{8.11}$$

$$\simeq (D^N)^{2^L-1}-1 \tag{8.12}$$

where the last step follows from the fact that each of the $C_j(D)$ divides $D^{2^L-1}-1$ and $\deg[C_j(D)]$ is close to 2^L-1. Thus (8.12) is virtually certain to hold. Let us summarize.

Property 8.3.

The linear complexity of the keystream produced by the knapsack stream cipher (depicted in Fig. 8.1) is upperbounded as

$$\Lambda(\tilde{z}) \leq N(2^L-1)$$

where near equality is virtually certain to hold (note that the linear complexity of $\tilde{z}$ could not be larger than $N(2^L-1)$ since this is the period of the keystream).

This concludes the theoretical analysis of the knapsack stream cipher.

8.3 Conclusions and Design Considerations

With the knapsack, it is possible to easily implement, and control a vast number of good nonlinear functions. The random choice of the knapsack weights leads automatically to functions which guarantee excellent distribution properties for the produced binary sequences. The inherent high order of the implemented functions ensures a linear complexity of the produced sequences close to the period length of the driving LFSR. Consequently, the key stream is virtually certain to have linear complexity nearly equal to its period. Because the large amount of key which affects the generation of the sequence observable at the ith output stage (which is in correspondence to Shannon's principle of diffusion (Shan 49)), it is infeasable to modularize the attack into single-stage attacks. Nevertheless, it might be possible that a correlation attack on a single stage output sequence would provide enough information to allow an ultimate solution for the corresponding function coefficients. But when the driving LFSR has reasonable length, say $L \simeq 50$, then the correlation attacker will be frustrated for two reasons. To acquire the necessary information generally requires correlation over substantially the whole period of the single stage output sequence, which is 2^L-1 when the driving LFSR has primitive connection polynomial. In addition the output bits of all stages are interleaved such that the effective monitoring time is N times larger. Now even if the first step of the correlation attack were feasible, there would remain the task of determining or approximating the actual function implemented by the knapsack. In section 7.1, we showed how a knapsack whose weights and input vectors are kept secret could (in principle) be attacked when the knapsack is one-to-one and all the partial sums are available. The probability that a randomly selected knapsack with N weights and a modulus $Q = 2^N$ is one-to-one is $Q!/Q^Q$, which is vanishingly small. For $N \simeq 50$ the weight determining algorithm of section 7.1 is far beyond applicability. This small probability of selecting a one-to-one knapsack also implies that a bad key choice (i.e. a superincreasing or modularly transformable knapsack) is virtually impossible to occur. So it is unnecessary to check a randomly selected key for weaknesses. We saw in the analysis that the least-significant $\lceil \log N \rceil$ output bits of a knapsack with N weights and a modulus which is a power of 2 are weaker than the higher order output bits. This suggests that the $\lceil \log N \rceil$ least significant output bits of the proposed knapsack stream cipher should be deleted. When the modulus is not a power of 2, but close to one,

say $Q=2^N-1$, then we may still expect the distribution properties of the produced sequences to be good, whereas the weakness of the least significant bits is remedied (compare the simulation results in section 8.4). But clearly this complicates the hardware involved and decreases the speed. So far we are guaranteed that neither attacks on the basis of linear equivalents nor attacks which try to discover the particular nonlinear functions employed will be feasible. But special care has to be taken that nothing leaks out about the current state of the driving LFSR. For, with known state, the problem of determining the weights reduces to the problem of solving a system of linear congruences. Thus neither the normal operation of the stream cipher nor the protocols which use the state (e.g. synchronization procedures) can permit an enemy to deduce any information about the current state. But when the knapsack stream cipher is operated as illustrated in Fig. 8.1, then any attacker knows that, by the cyclic shifting of the driving LFSR, N-1 of the binary input variables which contribute to the current partial sum will also contribute to the next partial sum of the knapsack. Thus it might be wise to destroy this one bit shift property of the input vector **x** to the knapsack. One way of accomplishing this is to introduce a variable speed factor d for the driving LFSR (as discussed in chapter 6). If the required speed of the stream cipher does not allow multiple clocking then the one-bit shift-property could be destroyed by selectively complementing the state of the driving LFSR before it is used as input vector to the knapsack.

8.4 Simulation Results of Small Scale Knapsack Stream Ciphers

In this subsection, we show the results of some simulations performed to confirm the heuristic analysis. In every simulation, a primitive connection polynomial and a randomly chosen selective complement were fixed. Then a knapsack was randomly chosen, and both the linear complexity and the distribution of single bits and pairs of bits was computed for every output sequence $\tilde{s}_j$. This was repeated for 100 different randomly chosen knapsacks. Then the average and standard deviation of both the linear complexity and the frequencies of bits and pairs of bits was computed. The simulation results are in excellent agreement with the theoretical heuristic predictions.

List of symbols:

C(D)	connection polynomial of driving LFSR
SC	selective complement
Q	modulus of the knapsack
$\overline{LC}$	average linear complexity
LC*	theoretical upperbound for linear complexity (Note that the selective complement can cause the function $f_{j,K}(\underline{x})$ to map $\underline{x} = \underline{0}$ into 1 which increases the upperbound (8.1) by 1 for $j < \log L$).
σ_{LC}	standard deviation of linear complexity
$\overline{P_{\underline{u}}}$	averaged frequency of occurrence of the pattern $\underline{u}$
$\sigma_{\underline{u}}$	associated standard deviation

Comments:

(1) Simulations 1 and 2 show that when the modulus of the knapsack is a power of two then the output sequences $\tilde{s}_j$, $j < \lceil \log L \rceil$ exhibit a significantly smaller linear complexity than the remaining output sequences.

(2) Simulation 3 shows that when the modulus of the knapsack is not a power of two then all the output sequences may be expected to attain close to maximum linear complexity.

Simulation 1

C(D) : $D^6 + D^5 + 1$
SC : 111001
Q : 2^6

	$\tilde{S}_5$	$\tilde{S}_4$	$\tilde{S}_3$	$\tilde{S}_2$	$\tilde{S}_1$	$\tilde{S}_0$
$\overline{LC}$	59.8	57.8	53.9	42.2	19	6.24
LC*	63	63	63	56	21	6
σ_{LC}	3.94	6.26	11.1	13.1	5.85	1.36
$\overline{P_0}$	.506	.518	.515	.507	.516	.52
$\overline{P_1}$	.494	.482	.485	.493	.484	.48
$\overline{P_{00}}$	.255	.274	.267	.256	.269	.276
$\overline{P_{01}}=\overline{P_{10}}$	.251	.244	.248	.250	.247	.244
$\overline{P_{11}}$	.243	.238	.237	.243	.237	.236
$\sigma_0=\sigma_1$	.0264	.0574	.0391	.0458	.0557	.0619
σ_{00}	.034	.0711	.044	.0553	.0782	.0932
$\sigma_{01}=\sigma_{10}$	.018	.0223	.0178	.0229	.0303	.0314
σ_{11}	.0298	.0504	.0419	.0468	.0437	.0308

Simulation 2

C(D) : $D^7 + D^4 + 1$
SC : 0000000
Q : 2^7

	$\tilde{s}_6$	$\tilde{s}_5$	$\tilde{s}_4$	$\tilde{s}_3$	$\tilde{s}_2$	$\tilde{s}_1$	$\tilde{s}_0$
$\overline{LC}$	123.93	123.44	120.9	110.63	74.76	25.9	6.93
LC*	127	127	127	127	98	28	7
σ_{LC}	4.76	6.36	10.82	18.44	22.57	6.06	0.7
min{LC}	99	92	63	28	28	7	0
$\overline{P_0}$	.509	.502	.5	.498	.503	.505	.501
$\overline{P_1}$	.496	.498	.5	.502	.497	.495	.499
$\overline{P_{00}}$	.251	.254	.247	.246	.254	.255	.252
$\overline{P_{01}}=\overline{P_{10}}$	.253	.249	.253	.252	.25	.249	.249
$\overline{P_{11}}$	.243	.249	.248	.25	.247	.246	.249
$\sigma_1=\sigma_0$	.03	.0348	.04	.0356	.0385	.0418	.0501
σ_{00}	.0355	.04	.0437	.0392	.0488	.0519	.0752
$\sigma_{01}=\sigma_{10}$	.0184	.0156	.0233	.019	.0232	.0187	.0251
σ_{11}	.0384	.0362	.0486	.0414	.0407	.0388	.0251

Simulation 3

LFSR : $D^7 + D^4 + 1$
SC : 0110111
Q : 2^7-1

	$\tilde{S}_6$	$\tilde{S}_5$	$\tilde{S}_4$	$\tilde{S}_3$	$\tilde{S}_2$	$\tilde{S}_1$	$\tilde{S}_0$
$\overline{LC}$	125.06	123.89	125.17	124.89	124.08	124.94	125.01
LC*	127	127	127	127	127	127	127
σ_{LC}	3.94	10.03	5.02	3.78	10.32	3.09	2.84
min{LC}	98	29	98	99	29	112	119
$\overline{P_0}$	.515	.508	.512	.507	.498	.507	.501
$\overline{P_1}$	.485	.492	.488	.493	.502	.493	.499
$\overline{P_{00}}$	.262	.262	.259	.255	.25	.26	.255
$\overline{P_{01}}=\overline{P_{10}}$	.252	.246	.253	.253	.248	.247	.246
$\overline{P_{11}}$	.233	.247	.235	.24	.255	.246	.253
$\sigma_0=$ 1	.0494	.0386	.0441	.0464	.0519	.04	.049
σ_{00}	.0519	.0424	.0591	.0539	.0582	.0461	.0437
$\sigma_{01}=\sigma_{10}$	.0177	.023	.0239	.0215	.0259	.0239	.0232
σ_{11}	.0531	.0474	.0393	.0483	.0578	.0469	.0631

9 Nonlinear Combining Functions with Memory

With the use of memory, the nonlinear combining function becomes a nonlinear finite state machine by itself which greatly increases the number of options available. We shall investigate how memory affects the catalogue of requirements to be imposed on the nonlinear combining subsystem (chapter 2.2).

9.1 Correlation-Immunity

A general nonlinear combining function must not "leak", that is, it must not be possible for the cryptanalyst to determine the subkey residing in one or more of the driving submodules form knowledge of the key stream and the structure of the running key generator ("divide and conquer"). Several authors (Sieg 84b, Xiao 85) investigated the correlation-immunity of nonlinear combiners, but always under the assumption that the combiner is memoryless. See section 5.3 for a detailled discussion of correlation-immunity of memoryless combiners. In 1984 the effects of internal memory on the correlation-immunity of nonlinear combiners were addressed by Rueppel (Ruep 84a). He showed that for a finite-state-machine combiner it was possible to obtain maximum correlation immunity together with maximum nonlinear order. Thus, the undesired tradeoff (5.80) could be broken through the use of memory. Subsequently some refinements and generalizations (Ruep 85b, Sieg 85) were added.

The behaviour of a finite-state-machine (FSM) is basically described by two functions: an output function f_0 operating on the internal state S_j and on the input X_j to produce the output Z_j, and a next-state function f_s operating on the internal state S_j and on the input X_j to produce the next state S_{j+1}, that is

$$Z_j = f_0(X_j, S_j) \tag{9.1a}$$

$$S_{j+1} = f_s(X_j, S_j) \tag{9.1b}$$

where every variable usually has its own alphabet. Representation (9.1) is called a Mealy-Automaton. For the FSM-combiner we shall assume that all variables consist of components from the same q-ary alphabet. Let the FSM-combiner be driven by N sequences, so that $\underline{X}_j = (X_{1j}, X_{2j}, \ldots, X_{Nj})$, let it put out a single digit Z_j at a time, and let $\underline{S}_j$ be a vector of arbitrary length. In order to investigate the statistical dependencies introduced by the FSM-combiner itself (and not by the sources which feed it) we shall further assume that the input sequences to the combiner are sequences of independent and uniformly distributed q-ary random variables. Fig. 9.1 shows the resulting model.

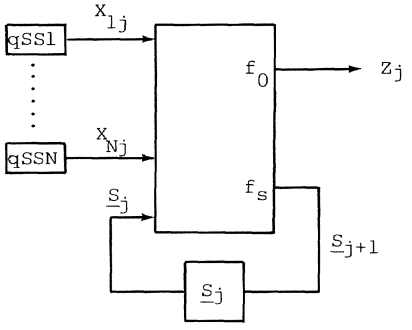


Fig. 9.1. Information-theoretic model of an FSM-combiner used to define correlation-immunity (qSS = q-ary symmetric source)

At this point it is illustrative to consider a practical example. Pless (Ples 77) proposed in 1977 a running-key generator which contains as basic building block a 2-LFSR subgenerator. In this subgenerator a J-K Flip-Flop acts as the nonlinear element which combines the two binary LFSR sequences. A J-K Flip-Flop defines a one-bit FSM whose state just contains the previous output bit, and whose output and next-state functions therefore coincide. Its behaviour is completely described in the GF(2)-equations

$$Z_j = X_{1j} + S_j(1 + X_{1j} + X_{2j}) \tag{9.2a}$$

$$S_{j+1} = Z_j \tag{9.2b}$$

Fig. 9.2 shows the logic equivalent.

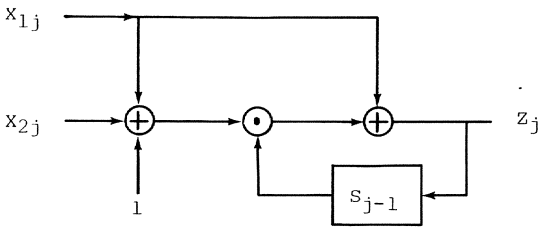


Fig. 9.2. FSM-equivalent of a J-K Flip-Flop

The output function f_0 as in (9.2a) is a memoryless mapping, so we can apply the techniques developed in section 5.3 to investigate the correlation-immunity of f_0. Fig. 9.3 displays the Walsh transform of f_0, where the ordering was chosen to be (X1j, X2j, Sj), or equivalently, where the function argument was thought of as the integer $X_{1j}2^0 + X_{2j}2^1 + S_j2^2$.

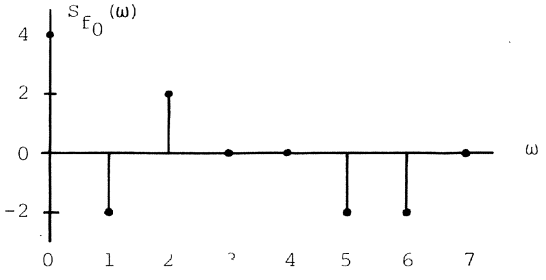


Fig. 9.3. Walsh transform of the boolean mapping defined by (9.2a)

The peaks in the Walsh transform at $\omega = 1$ and $\omega = 2$ tell us that the output bit Z_j is neither independent of X_{1j} nor of X_{2j}. Using (5.90a) we calculate for the probability that Z_j coincides with X_{1j} a value of 3/4, and for the probability that Z_j coincides with X_{2j} a value of 1/4. On the other hand, since $S_{f_0}(4)$ is zero, Z_j is independent of the internal state S_j which is merely the prior output bit Z_{j-1}. Consequently, if a J-K Flip-Flop is fed by two binary symmetric sources it will produce a sequence of independent and uniformly distributed random variables (as desired), but this output

sequence will exhibit a strong correlation with either input sequence. This fact was noted and exploited by Rubin (Rubi 79). This example illustrates that the output function f_0 plays an important role in the determination of the correlation-immunity of an FSM-combiner.

The natural extension of Siegenthaler's definition (Sieg 84a) of correlation-immunity for memoryless combiners is to say that an FSM-combiner is correlation-immune of order m if the mutual information between the output sequence and any subset of m input sequences is zero, that is if

$$I(Z^j; X_{i_1}^j, X_{i_2}^j, \ldots, X_{i_m}^j) = 0 \qquad j > 0 \tag{9.3}$$

$$1 \leqslant i_1 < i_2 < \ldots < i_m \leqslant N$$

where the notation Y^j stands for the sequence $Y_1, Y_2, \ldots, Y_j$ of length j. (Note that for notational convenience in this section we choose to let the sequences start with Y_1 rather than Y_0). In the memoryless model (Fig. 5.15) it is obvious that the output sequence must be a sequence of statistically independent random variables. Not so here; as Siegenthaler commented himself (Sieg 85), definition (9.3) allows to construct FSM-combiners which are mth-order correlation-immune but exhibit strong dependencies among the output digits. Consider for instance the binary output function.

$$Z_j = (1 + Z_{j-1}) \sum_{i=1}^{N} X_{ij} \tag{9.4}$$

When Z_{j-1} is 1 then Z_j must be 0 regardless of the input digits, when Z_{j-1} is 0 then Z_j equals to the sum of all current input variables. Thus the output function (9.4) is maximally correlation-immune (of order N-1), but is of no use in random sequence generation. In a running-key generator it must not be possible to reliably guess the next key bit regardless of how many prior key bits have been observed. Thus in order to make the discussion of correlation-immunity meaningful to the application of random sequence generation

we shall require that $\tilde{z}$ forms a sequence of independent and uniformly (or at least identically) distributed random variables. Under this constraint the definition (9.3) is equivalent to the following alternative definition of correlation-immunity for FSM-combiners:

$$I(Z_j;\ Z^{j-1},\ X^j_{i_1},\ \ldots,\ X^j_{i_m}) = 0 \qquad j > 0$$
$$1 \leqslant i_1 < \ldots < i_m \leqslant N \tag{9.5}$$

Definition (9.5) is intuitively pleasing: knowing all prior output digits and knowing (or guessing) jointly any m input sequences does not provide any information whatsoever on the next output digit. To prove the equivalence of definitions (9.3) and (9.5) when (9.3) is equipped with the constraint that Z^j must form a sequence of i.i.d. random variables, let m = 1 (only to avoid unnecessary notational akwardnesses, the proof for arbitrary m is identical), and decompose (9.3) in the following way:

$$I(Z^j;\ X^j) = I(Z^{j-1};\ X^j) + I(Z_j;\ X^j|Z^{j-1}) = 0$$

In a causal system $I(Z^{j-1};\ X^j) = I(Z^{j-1};\ X^{j-1})$.

Therefore

$$I(Z^j;\ X^j) = \sum_{k=1}^{j} I(Z_k;\ X^k|Z^{k-1}) = 0.$$

Mutual informations are always greater or equal to zero; hence it must hold that

$$I(Z_k;X^k|Z^{k-1}) = H(Z_k|Z^{k-1}) - H(Z_k|X^k,Z^{k-1}) = 0$$

Taking into account that $\tilde{z}$ forms an i.i.d. sequence, i.e. $H(Z_k|Z^{k-1}) = H(Z_k)$, we arrive at

$$I(Z_k;X^k|Z^{k-1}) = I(Z_k;Z^{k-1}X^k) = 0 \qquad k > 0$$

which establishes the equivalence.

Since the internal state $\underline{S}_j$ represents the complete information that is retained in the FSM-combiner from all previous input and output variables, we have the inequality

$$H(Z_j|Z^{j-1},X_{i_1}^j,\ldots,X_{i_m}^j) \geq H(Z_j|\underline{S}_j,X_{i_1 j},\ldots,X_{i_m j}) \tag{9.6}$$

Thus, if we require that Z_j is statistically independent of the state $\underline{S}_j$ and any m current input variables, considered jointly i.e. that

$$H(Z_j|\underline{S}_j,X_{i_1 j},\ldots,X_{i_m j}) = H(Z_j) \tag{9.7}$$

then (9.6) implies that the FSM-combiner is correlation-immune of order m according to definition (9.5). Now let the output function f_0 of the FSM-combiner have the following general form

$$Z_j = \sum_{i=1}^{N} X_{ij} + g(\underline{S}_j) \qquad j > 0 \tag{9.8}$$

where the current input variables $X_{1j},\ldots,X_{Nj}$ are summed and added modulo q to an arbitrary memoryless function g of the state $\underline{S}_j$. The state $\underline{S}_j$ represents the complete information that is retained in the FSM-combiner from all previous input and output variables. Suppose we know the complete history of the FSM-combiner and all but one, say X_{ij}, of the current input variables. We then may rewrite (9.8) as

$$Z_j = X_{ij} + Y_j \tag{9.9}$$

where Y_j summarizes our knowledge about the device. The fact that X_{ij} is drawn independently of Y_j from a uniform distribution implies that Z_j is also uniformly distributed. This can also be seen from the fact that (9.9) corresponds to sending Y_j through a memoryless q-ary symmetric channel with capacity zero, thereby ensuring that Z_j is uniformly distributed and independent of Y_j. Hence, any FSM-combiner which employs an output function of the form (9.8) is (N-1)st-order correlation-immune, this is in fact the maximum order of immunity possible. Moreover, note that we made no assumption about the

next-state function f_s nor the function g in (9.8). Therefore, those functions may be chosen independently in order to maximize the nonlinear effects of the FSM-combiner. In particular, one bit of memory suffices to realize a binary FSM-combiner with maximum correlation-immunity N-1 and with maximum nonlinear order N (with respect to the input variables). For this case the FSM equations (9.1) may be written as

$$z_j = \sum_{i=1}^{N} x_{ij} + s_j \qquad \text{in GF(2)} \qquad (9.10a)$$

$$s_{j+1} = f_s(x_{1j},\ldots,x_{Nj},s_j) \qquad \text{in GF(2)} \qquad (9.10b)$$

Fig. 9.4 shows the corresponding structure of the FSM-combiner.

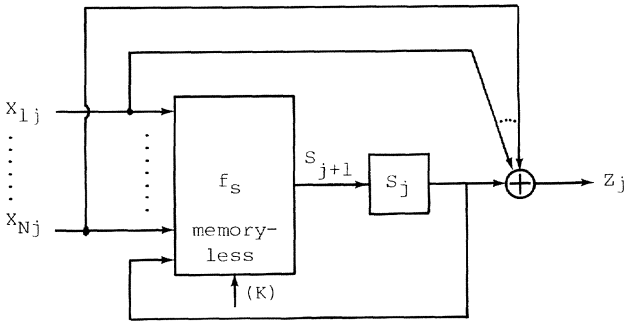


Fig. 9.4. 1-Bit memory FSM-combiner of maximum correlation-immunity N-1 allowing maximum nonlinear order in f.

If the next state s_{j+1} in (9.10b) is computed independently of the previous state then the FSM of Fig. 9.4 is said to have finite input memory (it can be realized with a pure feedforward structure).

The reason why (9.8) and (9.10) are so powerful is that the introduced memory allowed to separate the tasks of the combiner. The output function f_0 is responsible for the level of correlation-immunity and the balanced distribution of the output, whereas the next-state function f_s is responsible for the level of nonlinearity. Moreover,

since f_s is not restriced in any way, it can be chosen to be easily implemented and readily indexed by the key (if desired). In the memoryless case, all these tasks had to be burdened on a single mapping which obviously resulted in a conflict, as the tradeoff (5.80) indicates (compare Fig .5.16). In light of the above results it would be a definite disadvantage to require the nonlinear combiner to be memoryless.

The provable correlation-immunity of FSM-combiners in the case of truly random input sequences suggests that these structures are also secure against correlation attacks when the input sequences are sensibly chosen pseudo-random sequences, e.g. maximal-length sequences. In contrast, consider a brute force attack against the key stream generator depicted in Fig. 9.4 under the assumption that the N driving subgenerators are LFSRs and that the key resides in the initial contents of the LFSRs and the storage element. Since z^j, x_1^j, ..., x_{N-1}^j and S_1 completely specify x_N^j, a brute force attack can always be reduced to guessing S_1 and the initial contents of the N-1 smallest LFSRs. Thus, there is a slight sacrifice in resistance against a brute force attack when one uses the combiner of Fig. 9.4 in this manner.

9.2 The Summation Principle

Integer addition, when considered as a function of variables in GF(2), is a highly nonlinear operation (chapter 7.2). The knapsack stream cipher (chapter 8) illustrates one way of using integer addition in a cryptographic system. But there is a much more direct way to apply integer addition in the generation of sequences with large linear complexities. in Fig. 7.6, we showed the GF(2)-description of the real sum of two 3-bit integers. There essentially the same logic network (each π-box corresponds to a 3-bit real adder) is used for each bit of the sum. It is apparent that the sum of these two 3-bit integers (or in general n-bit integers) can be implemented bit-serially by time-sharing one π-box (Fig. 9.5).

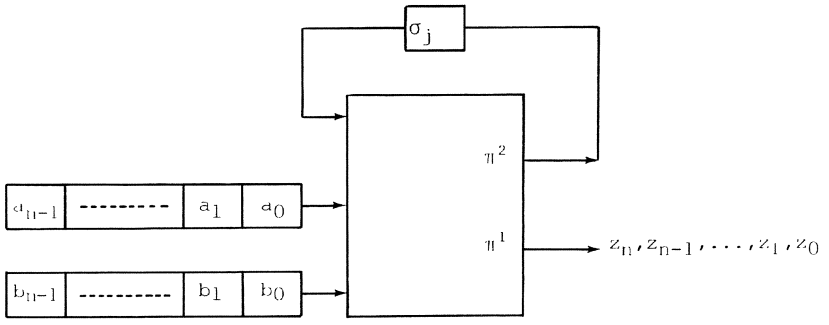


Fig. 9.5. Time-sharing of a 3-bit adder to produce bit-serially the real sum of two n-bit integers

When the two input shift registers in Fig. 9.5 are initially loaded with the binary representation (least significant bit first) of the 2 integers and when the feedback memory cell is initially zero, then after (n+1) clock cycles the (n+1) bits corresponding to the binary representation of the real sum will have appeared serially at the output. The real adder of Fig. 9.5 defines a finite state machine whose output and next state functions, according to definition (7.24), are given by

$$z_j = \pi^1(a_j, b_j, \sigma_j) = a_j + b_j + \sigma_j \qquad (9.11a)$$

$$\sigma_{j+1} = \pi^2(a_j, b_j, \sigma_j) = a_j b_j + (a_j + b_j)\,\sigma_j \qquad (9.11b)$$

Surprisingly enough, the real adder directly realizes the correlation immune structure displayed in Fig. 9.4. Note that π^1 defines the GF(2)-sum of the input variables and thus accounts for the correlation immunity, while π^2 defines the GF(2)-sum of all second-order products of the input variables and thus implements a memoryless nonlinear mapping. The memorycell is used to hold the "carry" bit from the (j-1)th to the jth position of the sum and carries all the nonlinear influence of the less significant bits. These observations suggest that real addition will be useful as nonlinear combining function in running key generation. The simplest running-key generator based on this summation principle may be obtained by adding two infinite integers whose binary representations are periodic sequences generated by suitable LFSRs.

To determine the period of the output sequence of a nonlinear FSM (as defined by the real adder in Fig. 9.5) when the inputs are periodic is in general a difficult problem. But in the case of real addition, these analytic difficulties may be avoided by considering the sequences as rational fractions. Let $\tilde{s} = s_0, s_1, \ldots,$ be a sequence of period T with digits from a finite alphabet $A_r = \{0, 1, \ldots, r-1\}$. Then we may uniquely identify the sequence $\tilde{s}$ with a rational fraction s

$$s = 0, \overline{s_{T-1}\, s_{T-2} \cdots s_1 s_0} \tag{9.12}$$

where the bar indicates that this part of the rational number is repeated periodically, and where we have assumed that the sequence $\tilde{s}$ starts with the least significant bits first. (Note that one could also identify $\tilde{s}$ with the rational fraction $s = 0,\overline{s_0 s_1 \ldots s_{T-2} s_{T-1}}$ where the initial sequence digits are assumed to have the highest significance). s defines a rational number, smaller than 1, whose value is given by

$$s = s_{T-1}r^{-1} + s_{T-2}r^{-2} + \ldots + s_0 r^{-T} + s_{T-1}r^{-T-1} + \ldots$$

$$= (s_{T-1}r^{-1} + s_{T-2}r^{-2} + \ldots + s_0 r^{-T})\,(1 + r^{-T} + r^{-2R} + \ldots)$$

$$= \frac{\sum_{i=0}^{T} s_{T-j}r^{T-j}}{r^T - 1} = \frac{p}{q} \qquad (9.13)$$

where we have reduced the rational fraction s to its lowest terms. As an example, consider the binary sequence $(011)^{\infty}$ which, when expressed as a rational fraction (9.13), results in $(1\cdot 2^2 + 1\cdot 2)/2^3-1) = 6/7$. On the other hand when 6/7 is expanded as a binary fraction one obtains

```
110  ÷  111 = 0,110...
1100
 111
 1010
  111
  110
    :
```

Since the remainder 110 = 6 again occurs, the division process must repeat from that point on.

Equation (9.13) shows that q divides r^T-1 which implies that $r^T = 1$ modulo q. Hence the order of r modulo q, denoted by $\mathrm{ord}_q(r)$, exists and divides the period T. On the other hand,

$$r^{\mathrm{ord}_q(r)}\, s = (kq + 1)s = kp + s \qquad (9.14)$$

where k is an integer. Thus, after $\mathrm{ord}_q(r)$ digits, the process by which the r-ary expansion of s is constructed repeats, which implies that T divides $\mathrm{ord}_q(r)$. We conclude that the period T of the r-ary sequence s̃ is equal to $\mathrm{ord}_q(r)$ when the rational fraction associated with s̃ has value s = p/q.

Let the sequences ã and b̃ have periods T_1 and T_2, respectively, such that gcd $(T_1,T_2) = 1$. According to (9.13), both sequences have associated rational fractions a and b,

$$a = \frac{A}{r^{T_1}-1} = \frac{p_1}{q_1} \tag{9.15a}$$

$$b = \frac{B}{r^{T_2}-1} = \frac{p_2}{q_2} \tag{9.15b}$$

where gcd (p_1,q_1) = gcd(p_2,q_2) = 1. The real sum of the sequences directly corresponds to the real sum of the rational fractions. Therefore

$$a + b = \frac{p_1}{q_1} + \frac{p_2}{q_2} = \frac{p_1q_2+p_2q_1}{q_1q_2} = c + \frac{n}{q_1q_2} \tag{9.16}$$

where we identify $n/(q_1q_2)$ as the rational fraction representing the real sum sequence $\tilde{z} = \tilde{a} + \tilde{b}$ and C, which is either 0 or 1, as the carry digit from one period of the sum sequence to the next. We note that gcd(n,q_1q_2) = 1 because gcd(q_1,q_2) = gcd(p_1,q_1) = gcd(p_2,q_2) = 1, and that gcd(r,q_1q_2) = 1 because gcd$(r,r^{T_1}-1)$ = gcd$(r,r^{T_2}-1)$ = 1. Thus the period of the r-ary expansion of n/q_1q_2 equals the period of the sum sequence $\tilde{z}$. The period T of the real sum sequence $\tilde{z}$, therefore, is given by

$$T = \operatorname{ord}_{q_1q_2}(r) \ . \tag{9.17}$$

Since q_1 and q_2 are relatively prime, the chinese remainder theorem (Star 81) implies that

$$T = \operatorname{ord}_{q_1}(r) \operatorname{ord}_{q_2}(r) = T_1T^2. \tag{9.18}$$

This concludes the proof of

Proposition 9.1. Period of real sum sequence

> Let $\tilde{a}$ and $\tilde{b}$ be two periodic sequences with digits from the same alphabet $A_r = \{0,1,\ldots,r-1\}$ whose periods are T_1 and T_2, respectively. When $\tilde{z}$ denotes the real sum of $\tilde{a}$ and $\tilde{b}$ expressed in radix r form and if gcd(T_1,T_2) = 1, then $\tilde{z}$ has period T_1T_2.

For example, suppose we want to add as integers the binary sequences $\tilde{a} = (10)^{\infty}$ and $\tilde{b} = (110)^{\infty}$ of periods 2 and 3, respectively. By considering the sequences as expansions of rational fractions we obtain

$$a = \frac{1}{3} \Rightarrow T_1 = \mathrm{ord}_3(2) = 2$$

$$b = \frac{3}{7} \Rightarrow T_2 = \mathrm{ord}_7(2) = 3$$

Then, because gcd(2,3) = 1,

$$z = \frac{1}{3} + \frac{3}{7} = \frac{16}{21} \Rightarrow T = \mathrm{ord}_{21}(2) = 6$$

and 16/21 expanded as binary fraction results in $0,\overline{110000}$ which corresponds to the sum sequence $\tilde{z} = (000011)^{\infty}$ of period T = 6. Since there is no overflow bit from one period of $\tilde{z}$ to the next, we could have obtained the real sum sequence directly by adding $T_1T_2 = 6$ bits from $\tilde{a}$ and $\tilde{b}$, writing the initial bits of each sequence as least significant bits of the integers to be added,

$$\begin{array}{l} 010101 \\ \underline{011011} \\ 110000 \end{array}$$

An interesting feature of considering the sequences as rational fractions with least significant bits leading is that the "transient phase" of the real sum sequence is moved to infinity and that the state of the summation combiner at the beginning of a new period of the sum sequence is directly given by the integer part of the sum of of the rational fractions (compare (9.16).

We may generalize proposition 9.1 to the sum of N periodic sequences.

Corollary 9.2

> Let $\tilde{x}_i$ (i = 1,...,N) be a periodic sequence with r-ary digits whose period is T_i. When $\tilde{z}$ denotes the integer sum of the $\tilde{x}_i$ and if $\gcd(T_i,T_j) = 1$, $i \neq j$, then $\tilde{z}$ is ultimately periodic with period

$$T = \prod_{i=1}^{N} T_i \ . \qquad (9.19)$$

Proof: For N = 2, proposition 9.1 ensures that the period of the real sum sequence $\tilde{z}_1 = \tilde{x}_1 + \tilde{x}_2$ will be T_1T_2. Now add a third sequence $\tilde{x}_3$ to $\tilde{z}_1$. Because T_3 is relatively prime to T_1T_2, proposition 9.1 guarantees that the period of $\tilde{z}_2 = \tilde{z}_1 + \tilde{x}_3$ will be $T_1T_2T_3$.

The corollary now follows by induction.

When the real adder of Fig. 9.5 is fed by two PN-sequence generators whose lengths are relatively prime, then, by proposition 5.9 (period criterion) and by proposition 9.1, the output sequence will have ultimate period equal to the product of the periods of the PN-sequences. In order to judge the applicability of real summation in complex sequence generation, we shall analyze the linear complexity of the real sum of two PN-sequences $\tilde{a}$ and $\tilde{b}$. Let L_1 and L_2 be the degrees of the corresponding primitive minimal polynomials and assume that $\gcd(L_1,L_2) = 1$, which implies that the real sum sequence $\tilde{z}$ has period $T = (2^{L_1}-1)(2^{L_2}-1)$. The real adder as depicted in Fig. 9.5 defines a finite state machine (FSM) whose output and next-state functions are given by (9.11). An FSM is said to have finite input memory M if M is the least integer such that the output digit at time j may be expressed as a function of the input variables at times $j-M,\ldots,j-1,j$. Clearly the FSM as described by (9.11) has in general infinite input memory. But whenever the input sequences to the real adder produce a pair of zeros or ones, then the state of the adder FSM is set to a value independent of the preceding states and input values. In particular, when periodic input sequences are used which produce at least a common pair of zeros or ones within the period of the output sequence, then the input memory M will be finite with respect to the particular driving sequences. For the moment we shall assume that the input memory M is fixed at some value m. This allows to convert the feedback structure of the nonlinear combiner (9.11) into a feedforward structure of input memory m. From the feedforward function f_m, it then is possible to calculate the associated linear complexity Λ_m of the output sequence.

Suppose we set M = 0, then (9.11) implies that the output sequence $\tilde{z}$ equals the GF(2)-sum of the PN-sequences $\tilde{a}$ and $\tilde{b}$, and thus has linear complexity $\Lambda_0 = L_1 + L_2$.

Suppose M=1, then (9.11) implies that

$$z_j = f_1(\tilde{a},\tilde{b}) = a_j + b_j + a_{j-1}b_{j-1}. \tag{9.20}$$

Lemma 5.12 ensures that the product $\tilde{a}\tilde{b}$ of the two maximum-length sequences has irreducible minimal polynomial of degree L_1L_2. It follows that $\Lambda_1 = L_1 + L_2 + L_1L_2$. From now on, we will neglect for analytic convenience the linear part of the nonlinear feedforward function f_m and we will redefine $\Lambda_0 = 0$ and $\Lambda_1 = L_1L_2$.

With M=2 (and neglected linear terms), (9.11) results in

$$z_j = f_2(\tilde{a},\tilde{b}) = a_{j-1}b_{j-1} + a_{j-1}a_{j-2}b_{j-2} + b_{j-1}a_{j-2}b_{j-2}. \tag{9.21}$$

Each third order product in (9.21) includes a product of two distinct phases of one maximum-length sequence.

Let α in $GF(2^{L_1})$ and β in $GF(2^{L_2})$ be a root of the primitive minimal polynomial $m_{\tilde{a}}(X)$ and $m_{\tilde{b}}(X)$, respectively. Then the product $\tilde{a}^{-1}\tilde{a}^{-2}$ has an associated minimal polynomial all of whose roots α^{e_1} have exponents e_1 of binary weight $W_2(e_1) \leq 2$ (compare (Key 76) and proposition 5.5). This implies that the linear complexity of $\tilde{a}^{-1}\tilde{a}^{-2}$ is at most $L_1 + \binom{L_1}{2}$. From proposition 5.13 then follows that $\tilde{a}^{-1}\tilde{a}^{-2}\tilde{b}^{-2}$ has linear complexity at most $L_2(L_1 + \binom{L_1}{2})$, which corresponds to the number of distinct elements $\alpha^{e_1}\beta^{e_2}$ in $GF(2^{L_1L_2})$ with $W_2(e_1) \leq 2$ and $W_2(e_2) = 1$. We conclude that

$$\Lambda_2(\tilde{z}) \leq L_1L_2 + L_1\binom{L_2}{2} + \binom{L_1}{2} L2 \ . \tag{9.22}$$

In general, when the memory M is extended from m-1 to m then 2^{m-1} new product terms of order m+1, with k, $1 \leq k \leq m$, distinct phases from the first PN-sequence $\tilde{a}$ and with m+1-k distinct phases from the second PN-sequence $\tilde{b}$ are obtained,

$$f_m - f_{m-1} = \tilde{a}^{-m}\tilde{b}^{-m} \sum_{\underline{i}\varepsilon\{0,1\}^m} (i_1\tilde{a}^{-1}+\bar{i}_1\tilde{b}^{-1}) \ldots (i_{m-1}\tilde{a}^{m-1}+\overline{i_{m-1}}\tilde{b}^{m-1}) \qquad (9.23)$$

where the sum ranges over all binary m-tuples $(i_1,\ldots,i_m)$ and $\bar{i}_k$ denotes the complement of i_k. The set R_m of distinct elements in $GF(2^{L_1L_2})$ that are possible roots of the minimal polynomial of $\tilde{z}$ as produced by f_m is given by

$$R_m = \{\alpha^{e_1}\beta^{e_2} : W_2(e_1) \leq k,\ W_2(e_2) \leq m+1-k,\ 1 \leq k \leq m\}. \qquad (9.24)$$

The only elements in R_m not also contained in R_{m-1} are

$$R_m - R_{m-1} = \{\alpha^{e_1}\beta^{e_2} : W_2(e_1) = k,\ W_2(e_2) = m+1-k,\ 1 \leq k \leq m\}. \qquad (9.25)$$

The number of these proper elements of R_m is

$$\sum_{k=1}^{m} \binom{L_1}{k} \binom{L_2}{m+1-k} \qquad (9.26)$$

which implies that the linear complexity of the sequence $\tilde{z}$ produced by the real adder equation (9.11), but with fixed memory M, is upperbounded as

$$\Lambda_M(\tilde{z}) \leq \sum_{m=1}^{M} \sum_{k=1}^{m} \binom{L_1}{k} \binom{L_2}{m+1-k} . \qquad (9.27)$$

The bound (9.27) increases with M and reaches its maximum value at $M = L_1 + L_2 - 1$. Consequently,

$$\Lambda_M(\tilde{z}) \leq (2^{L_1}-1)\,(2^{L_2}-1) \qquad M \geq L_1 + L_2 - 1 \qquad (9.28)$$

which implies the intuitively pleasing result that the linear complexity of the real sum sequence is at most equal to the product of the periods of the two PN-sequences. Thus, if the input memory is at

least $L_1 + L_2 - 1$, we will be virtually certain that the real sum sequence has maximum possible linear complexity (equal to its own period). A closer look at the real adder equation (9.11) reveals that the output sequence $\tilde{z}$ may equivalently be expressed in terms of the bit-by-bit GF(2)-sum $s_j = a_j + b_j$ and the GF(2)-product $p_j = a_j b_j$ of the PN-sequences (Maur 84)

$$z_j = s_j + \sum_{m=1}^{M} p_{j-m} \prod_{k=1}^{m-1} s_{j-k} \cdot \tag{9.29}$$

From (9.29) it follows that the input memory M cannot be larger than $L_1 + L_2 + 1$, since the maximum number of consecutive 1's in the sum sequence $\tilde{s}$ is $L_1 + L_2$. On the other hand, assume that $p_j = a_j b_j = 1$. Then in the states of the LFSRs at time j there remain $L_1 + L_2 - 2$ free variables $a_{j+1}, \ldots, a_{j+L_1-1}, b_{j+1}, \ldots, b_{j+L_2-1}$, which can for most maximum-length recursions be chosen such that the successive digits $s_{j+1}, \ldots, s_{j+L_1+L_2-2}$ of the sum sequence will all be one. Thus the memory M is virtually certain to be at least $L_1 + L_2 - 1$. We summarize

Property 9.3

> Let $\tilde{a}$ and $\tilde{b}$ be two binary PN-sequences whose primitive minimal polynomials have relatively prime degrees L_1 and L_2. When $\tilde{a}$ and $\tilde{b}$ are added over the reals (as shown in Fig. 9.5 and as defined in (9.11)), then the real sum sequence $\tilde{z}$ exhibits linear complexity close to its period length,
>
> $$\Lambda(\tilde{z}) \leq (2^{L_1}-1)(2^{L_2}-1) \tag{9.30}$$
>
> with near equality.

In table 9.1 we shall display some simulation results (Maur 84) which confirm that the bound (9.30) is extremely tight. In fact, no serious degeneracy was ever found, which suggests that integer addition is an inherently good nonlinear function.

N	L_1	L_2	L_3	T	$\Lambda(\tilde{z})$
2	3	4		105	$\geqslant$ 100
	3	5		217	$\geqslant$ 208
	4	5		465	$\geqslant$ 455
3	2	3	5	651	$\geqslant$ 641

Table 9.1. Small-scale simulations giving evidence that the bound (9.30) is very tight (for explanation of the table see text below)

In table 9.1, the column labeled N gives the number of m-sequences added over the reals; the columns labeled L_1, L_2 and L_3 give the degrees of the minimal polynomials of the m-sequences which were added; the column labeled T gives the period of the sum sequences; the column labeled $\Lambda(\tilde{z})$ displays the smallest linear complexity obtained for all possible combinations of different primitive minimal polynomials of the mentioned degrees. For instance, the first row tells us that each of the different m-sequences of degree 3 (there are 2) was separately added to each of the different m-sequences of degree 4 (there are 2), and never was a linear complexity of smaller than 100 obtained.

9.3 Summary and Conclusions

We have shown that for an FSM-combiner there exists no tradeoff between correlation-immunity and nonlinear order. The memory in the FSM-combiner allows to decouple the requirements in such a way that the output function is responsible for the level of correlation-immunity and the balanced distribution of the output, whereas the next-state function is responsible for the level of nonlinearity. Thus, maximum correlation-immunity of order N-1 is basically for free, and moreover, since the next state function is not restricted in any way, it can be chosen to be easily implemented and readily indexed by the key. As an exemplatory structure we have shown in Fig. 9.4 a one-bit-memory FSM-combiner which can satisfy all of the above requirements. In (9.8) we have given a general form of the output function which will always result in maximum correlation-immunity of the FSM-combiner and a balanced output distribution (under the assumptions made in section 9.1). When the input sequences to the correlation-immune combiner of Fig. 9.4 are generated by LFSRs and when it is assumed that the key resides only in the initial contents of these LFSRs (or equivalently, the nonlinear part f_s is assumed to be fixed and known), then a brute force attack may be reduced to guessing the initial contents of the memory cell and all but the longest LFSR. This does not decrease significantly the security of the proposed structure except when only two LFSRs are used as driving subgenerators.

Although it seems counterintuitive, integer (real) addition is extremely nonlinear when viewed over GF(2). The results in section 9.2 show that given two integers whose binary representations have very low (linear) complexity then their real sum may have very high (linear) complexity. This of course depends whether use was made of the nonlinear potential of real addition. Suppose, for example, we add the two integers whose binary representations are the sequences 0101... and 1010... each having linear complexity 2. The result is the all-1 sequence of linear complexity 1. Note that in this case the real sum and the mod-2 sum of the 2 sequences are identical and, in fact, never a nonlinear contribution through a carry occurred.

We have also shown that real addition of r-ary sequences (which can be implemented digit-serially) inherently provides the maximally correlation-immune structure depicted in Fig. 9.4. Moreover, when N

r-ary sequences, whose periods are pairwise relatively prime, are added over the reals, then the resulting r-ary sum sequence will have period equal to the product of the periods of the N sequences. When two binary PN-sequences, whose minimal polynomials have relatively prime degrees, are added over the reals, then the linear complexity of the sum sequence attains with virtual certainty its maximum possible value, the period of the sum sequence. Finally, real addition is easily implemented and fast. This suggests that the following general structure is useful in secure sequence generation.

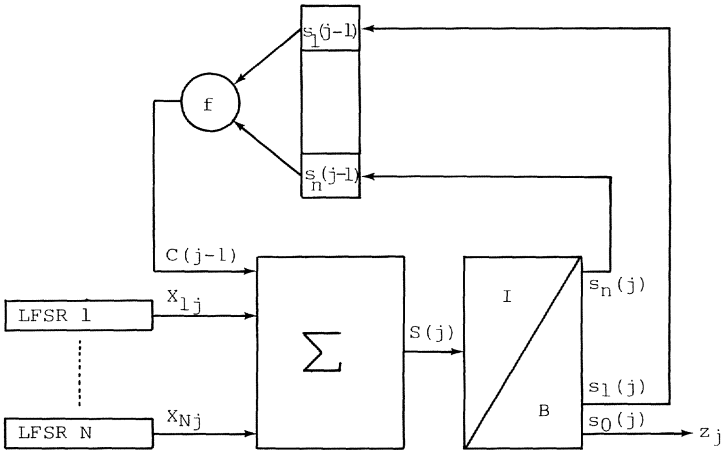


Fig. 9.6. General running-key generator basing on the real summation principle

Here the N current input digits from the driving LFSRs are added over the reals together with a feedback value C(j-1) which may depend on all but the current input values. In the integer/binary converter, the least significant digit is emitted as the current key stream digit. The remaining digits of the binary representation of S(j) are stored in the feedback memory on which an arbitrary function f may operate to produce the feedback value C(j) to be added at time j+1.

The general summation combiner (as depicted in Fig. 9.6) is capable of satisfying all the requirements that we demanded of the nonlinear combining subsystem (see chapter 2.2) except for the last one: the real sum of N sequences is a fixed function and can therefore not be

controlled by part of the key. But there is an interesting way how the key can be added to the summation combiner without affecting the nature of real addition (Fig. 9.7).

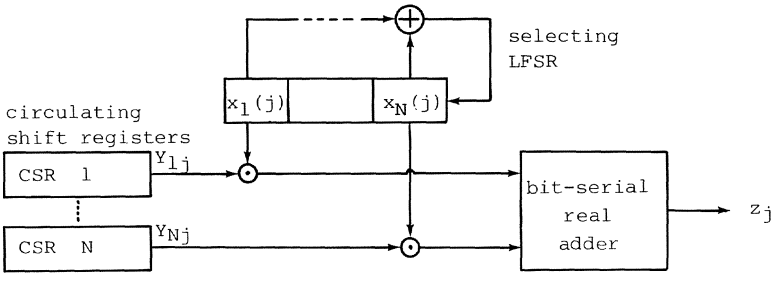


Fig. 9.7. A structure equivalent to the knapsack stream cipher (see chapter 8)

Let the circulating shift registers, whose contents are determined by the key, all have equal length N and let the selecting LFSR have a primitive connection polynomial of degree N. Suppose now, for each state of the selecting LFSR, the circulating shift registers are shifted N times (such that they are again in their starting phase) and the real adder has computed bit-serially N bits of the output sequence. If at this time instant, before the selecting LFSR is shifted to the next state, the feedback memory of the real adder is cleared, then the structure of Fig. 9.7 realizes the bit-serial implementation of the knapsack stream cipher (chapter 8). The weights of the knapsack are easily seen to sit in the circulating shift registers, and the selecting LFSR just enables the corresponding subset of weights to be summed. The N weights define N functions operating on the state of the selecting LFSR, each producing an individual bit in the binary representation of the knapsack sum (compare section 7.3). Thus, when the weights are added to the key, they allow one to index a large set of distinct functions, which simultaneously guarantees a large unicity distance. The appealing fact about the structure 9.7 is that it combines the cryptographically interesting properties of real addition with the provable complexity of the knapsack problem. It is apparent that the weak bits of the knapsack stream cipher stem only from the clearing of the feedback memory associated with the real adder. Therefore, the weakness of

the least significant bits of the knapsack sum is easily remedied by storing the nonlinear contents of the feedback memory after computation of a knapsack sum. This corresponds to always adding the overflow bits of the preceding knapsack sum to the least significant bits of the subsequent knapsack sum. With this feature, the resulting cipher may be called a knapsack stream cipher with "wrap-around carry". But in fact the operation of this device can no longer be described rigorously in terms of a knapsack, since the key stream bits are no longer broken up into natural "blocks" corresponding to knapsack sums. This implies that its security cannot be tied directly to the complexity of the knapsack problem, although it seems that the security has been increased by strengthening the weak bits of the knapsack sum.

Summarizing, we may say that memory in the nonlinear combining function has the potential of dramatically increasing the security of key stream generators.

Literature References

Berl 68 E.R. Berlekamp, "Algebraic Coding Theory", McGraw-Hill, New York, 1968.

Bern 85 J. Bernasconi, C.G. Günther, "Analysis of a Nonlinear Feedforward Logic for Binary Sequence Generators", submitted to IEEE Trans. on Info. Theory.

Beth 84 T. Beth, F.C. Piper, "The Stop-and-Go Generator", Proceedings of Eurocrypto 84, Paris, France, 1984.

Blas 79 W. Blaser, P. Heinzmann, "New Cryptographic Device with High Security Using Public Key Distribution", Proceedings of IEEE Student Paper Contests 1979 - 1980, P. 145 - 153, 1982.

Bric 83 E.F. Brickell, G.J. Simmons, "A Status Report on Knapsack Based Public Key Cryptosystems", Congress Numerantium, Vol. 37, 1983.

Bril 83 J. Brilhart, D.H. Lehner, J.L. Selbridge, "Factorization of $b^n \pm 1$, b = 2, 3, 5, 6, 7, 10, 11, 12 up to high powers", Contemporary Mathematics, vol. 22, 1983.

Brüe 84 J.O. Brüer, "On Pseudo Random Sequences as Crypto Generators", Proceedings of Int. Zurich Sem. on Digital Comm., Zurich, Switzerland, 1984.

Carm 14 R.D. Carmichael, "On the Numerical Factors of the Arithmetic Forms $\alpha^n \pm \beta^n$", Annals of Mathematics, Vol. 15 (1913-14), pp. 30-70.

Dai 85 Zong-duo Dai, "Proof of Rueppel's Linear Complexity Conjecture", submitted for publication to the IEEE Trans. on Info. Theory.

Denn 83 D.E. Denning, "Cryptography and Data Security", Addision-Wesley, 1983.

Desm 84 Y.G. Desmedt, H.P. Vandewalle, R.J.M. Govaerts, "A Critical Analysis of the Security of Knapsack Public-Key Algorithms", IEEE Trans. on Info. Th., Vol. IT-30, July 1984.

Diff 76 W. Diffie, M.E. Hellman, "New Directions in Cryptography", IEEE Trans. on Info. Th., Vol. IT-22, Nov. 1976.

Diff 79 W. Diffie, M.E. Hellman, "Privacy and Authentication: An Introduction to Cryptography", Proc. IEEE, Vol. 67, March 1979.

Fell 68 W. Feller, "An Introduction to Probability Theory and its applications", Vol. 1, John Wiley, 1968.

Gare 79 M.R. Garey, D.S. Johnson, "Computers and Intractability, A Guide to the Theory of NP-Completeness", W.H. Freeman and Co., San Francisco, Calif., 1979.

Geff 73 P.R. Geffe, "How to Protect Data with Ciphers that are Really Hard to Break", Electronics, Jan. 4, 1973.

Goll 84 D. Gollman, "Pseudo Random Properties of Cascade Connections of Clock Controlled Shift Registers", Proceedings of Eurocrypt 84, Paris, France, 1984.

Golo 67 S.W. Golomb, "Shift Register Sequences", Holden-Day, San Francisco, Calif., 1967.

Grot 71 E.J. Groth, "Generation of Binary Sequences with Controllable Complexity", IEEE Trans. on Info. Theory, Vol. IT-17, May 1971.

Gust 76 F.G. Gustavson, "Analysis of the Berlekamp-Massey Linear Feedback Shift-Register Synthesis Algorithm", IBM J. Res. Develop., 1976.

Herl 82 T. Herlestam, "On the Complexity of Functions of Linear Shift Register Sequences", Int. Symp. on Info. Th., Les Arc, France, 1982.

Horo 74 E. Horowitz, S. Sahni, "Computing Partitions with Applications to the Knapsack Problem", J. ACM, Vol. 21, April 1984.

Kahn 67 D. Kahn, "The Codebreakers", Macmillan Co., New York, 1967.

Karn 84 E.D. Karnin, "A Parallel Algorithm for the Knapsack Problem", IEEE Trans. on Comp., Vol. C-33, May 1984.

Key 76 E.L. Key, "An Analysis of the Structure and Complexity of Nonlinear Binary Sequence Generators", IEEE Trans. on Info. Theory, Vol. IT-22, Nov. 1976.

Knut 81 D.E. Knuth, "The Art of Computer Programming, Vol. 2: Seminumerical Algorithms" Addison-Wesley, 1981.

Kolm 65 A.N. Kolmogorov, "Three Approaches to the Quantitative Definition of Information", Probl. Inform. Transmission, Vol. 1, 1965.

Kuma 83 P.V. Kumar, R.A. Scholtz, "Bounds on the Linear Span of Bent Sequences", IEEE Trans. on Info., IT-29, Nov. 1983.

Lemp 70 A. Lempel, "High Speed Generation of Maximal Length Sequences", IEEE Trans. on Computers, Vol. C-19, Feb. 1970.

Lemp 76 A. Lempel, J. Ziv, "On the Complexity of Finite Sequences", IEEE Trans. on Info. Theory, IT-22, Jan. 1976.

Lidl 83 R. Lidl, H. Niederreiter, "Finite Fields", Encyclopedia of Mathematics and its Applications, Vol. 20, Addison-Wesley, 1983.

Link 84 P. Linke, "Nonlinearly Filtered Shift Registers for Cryptographic Applications", Semester Project, Swiss Federal Institute of Technology, Zürich, Feb. 1984.

MacW 83 F.J. MacWilliams, N.J.A. Sloane, "The Theory of Error-Correcting Codes", North-Holland, 1983.

Mart 66 P. Martin-Loef, "The Definition of Random Sequences", Information and Control, Vol. 9, 602-619, 1966.

Mass 69 J.L. Massey, "Shift-Register Synthesis and BCH Decoding", IEEE Trans. on Info. Theory, Vol. IT-15, Jan. 1969.

Mass 83 J.L. Massey, "A New Multiplication Method for GF(2^m) and its Application in Public Key Cryptography", presented at Eurocrypt 83, Udine, Italy, March 1983.

Mass 84a J.L. Massey, R.A. Rueppel, "Linear Ciphers and Random Sequence Generators with Multiple Clocks", presented at Eurocrypt 84, Paris, France, April 9-11, 1984.

Mass 84b J.L. Massey, "Applied Digital Information Theory", Lecture Notes, Swiss Federal Institute of Technology, Zurich, Switzerland, 1984.

Mass 85 J.L. Massey, "Delayed-Decimation/Square Sequences", Proceedings 2nd Joint Swedish Soviet Workshop on Information Theory, Gr{nna, Sweden, April 14-19, 1985, pp. 118-123.

Maur 84 U. Maurer, R. Viscardi, "Running Key Generators with Memory in the Nonlinear Combining Function", Diploma Project, Swiss Federal Institute of Technology Z}rich, Dec. 1984.

Merk 78 R.C. Merkle, M. Hellman, "Hiding Information and Signatures in Trapdoor Knapsacks", IEEE Trans. on Info. Th., Vol. IT-24, Sept. 1978.

NBS 77 "Data Encryption Standard", FIPS PUB 46, National Bureau of Standards, Washington, D.C., 1977.

Nyff 75 P. Nyffeler, "Binaere Automaten und ihre Linearen Rekursionen", Ph. D. Thesis, University of Bern, 1975.

Ples 77 V.S. Pless, "Encryption Schemes for Computer Confidentiality", IEEE Trans. on Computers, Vol. C-26, Nov. 1977.

Pohl 78 S.C. Pohlig, M.E. Hellman, "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Signifiance", IEEE Trans. on Info. Theory, Vol. IT-24, Jan. 1978.

Rive 78 R. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Comm. of the ACM, Vol. 21, Feb. 1978.

Rubi 79 F. Rubin, "Decrypting a Stream Cipher Based on J-K Flip-Flops", IEEE Trans. on Computers, Vol. C-28, July 1979.

Ruep 84a R.A. Rueppel, "New Approaches to Stream Ciphers", Ph.D. Thesis, Swiss Federal Institute of Technology, Zürich, 1984.

Ruep 84b R.A. Rueppel, "Telecommand Uplink Signature System", to be included in "Methods and Standards for Encryption of Spacecraft Telemetry and Telecommand Link", Final Report (to appear as ESA Guideline), ESTEC Contract 5815/84/NL/BI, 1984.

Ruep 85a R.A. Rueppel, J.L. Massey, "The Knapsack as a Nonlinear Function", IEEE International Symposium on Information Theory, 1985, Brighton, UK.

Ruep 85b R.A. Rueppel, "Correlation Immunity and the Summation Generator", Proceedings of Crypto 85, Santa Barbara, August 18-22, 1985.

Ruep 86 R.A. Rueppel, O. Staffelbach, "Products of Sequences with Maximum Linear Complexity", to appear in IEEE Trans. on Info. Theory.

Sarw 80 D.V. Sarwate, M.B. Pursley, "Crosscorrelation Properties of Pseudorandom and Related Sequences", Proceedings of the IEEE, Vol. 68, May 1980.

Schr 79 R. Schroeppel, A Shamir, " A $TS^2 = O(2^n)$ Time/Space Tradeoff for Certain NP-Complete Problems", Proc. 20th IEEE Symp. on the Foundations of Computer Science, Oct. 1979.

Selm 66 E.S. Selmer, "Linear Recurrence Relations over Finite Fields", Dept. of Math., University of Bergen, Norway, 1966.

Sham 82 A. Shamir, "A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem", Proc. 23rd Annual Symp. on Found. of Comp. Sci., 1982.

Shan 49 C.E. Shannon, "Communication Theory of Secrecy Systems", Bell Syst. Tech. J., Vol. 28, Oct. 1949.

Sieg 84a T. Siegenthaler, "Decrypting a Class of Stream Ciphers Using Ciphertext only", IEEE Trans. on Computers, Vol. C-33, 1984.

Sieg 84b T. Siegenthaler, "Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications", IEEE Trans. on Info. Theory, Vol. IT-31, to appear.

Sieg 85 T. Siegenthaler, "Design of Combiners to Prevent Divide and Conquer Attacks", Proceedings of Crypto 85, Santa Barbara, August 18-22, 1985.

Solo 64 R.J. Solomonov, "A Formal Theory of Inductive Inference", Part I, Inform. Control 7, 1964.

Star 81 H.M. Stark, "An Introduction to Number Theory", MIT Press, 1981.

Xiao 85 Xiao Guo-Zhen, J.L. Massey, "A Spectral Characterization of Correlation-Immune Combining Functions", to appear in IEEE Trans. on Info. Theory.

Zier 59 N. Zierler, "Linear Recurring Sequences", J. Soc. Indust. Appl. Math., Vol. 7, 1959.

Zier 73 N. Zierler, W.H. Mills, "Products of Linear Recurring Sequences", J. Algebra, Vol. 27, 1973.

Glossary

ordered according to earliest occurrence in text.

m_j	plaintext character
c_j	ciphertext character
z_j	derived key letter, running key
K	key
E	enciphering transformation
σ_j	internal state of running key generator or LFSR
LFSR	linear feedback shift register
GF(q)	finite field (Galois field) of order q
$R_a(b)$	remainder of a after division by b
GF(q)[X]	polynomial ring over GF(q)
gcd	greatest common divisor
lcm	least common multiple
a(X)	polynomial in the indeterminate X
E	extension field of GF(q)
$m_\alpha(X)$	minimum polynomial of α over GF(q)
α	element in some extension field E
C(D)	connection polynomial of an LFSR
$\tilde{s}=s_0,s_1,s_2,..$	semi-infinite sequence of digits
$\tilde{s}^t=s_t,s_{t+1},..$	t-th phase of $\tilde{s}$

$S(D)$	D-transform of $\tilde{s}$
$c(X)$	characteristic polynomial of an LFSR or of a linear recursion
$m_{\tilde{s}}(X)$	minimal polynomial of a sequence $\tilde{s}$
$T_L(\alpha)$	trace function mapping α in $GF(q^L)$ into $GF(q)$
$s^n=s_0,..,s_{n-1}$	finite sequence of n digits
$\Lambda(s^n)$	linear complexity of s^n
$N_n(L)$	number of sequences of length n with linear complexity L
δ_n	next discrepancy in LFSR-synthesis
$E[\Lambda(s^n)]$	expected linear complexity of a random sequence
$\lceil x \rceil$	ceiling function: the integer greater or equal to x
$\lfloor x \rfloor$	floor function: the integer smaller or equal to x
$Var[\Lambda(s^n)]$	variance of the linear complexity of a random sequence
T	period
f	nonlinear function (mostly used for state filter)
$\tilde{z}$	running key
A_e	coefficient of α^{ej} in the time domain solution of a linear recursion
$W_2(e)$	Hamming weight of the binary representation of e
ΔV	Vandermonde determinant
F	nonlinear function (mostly used for a combiner)

N	usually number or arguments of F (also number of weights in knapsack)
$\phi(.)$	Euler's totient function
$Q_m(X)$	m-th cyclotomic polynomial
F_m	greatest primitive factor of q^m-1
Z, X_i, Y_i	random variables
$I(Z;X)$	mutual information between Z and X
$H(Z)$	uncertainty about Z
$H(X\|Z)$	conditional uncertainty about X given Z
$S_F(\omega)$	Walsh transform of F
$\underline{\omega}\cdot\underline{X}$	linear combination $\omega_0X_0 + \omega_1X_1 + \ldots + \omega_{N-1}X_{N-1}$
d	speed factor, decimation factor
$\tilde{s}[d]$	d-th decimation of $\tilde{s}$
$(\tilde{s}^t[d])_j$	the j-th digit of the d-th decimation of the t-th phase of $\tilde{s}$
S	knapsack sum
$w_1,\ldots,w_N$	ordered sequence of knapsack weights
$\underline{x}=(x_1,\ldots,x_N)$	selection vector of the knapsack weights
$F_w(\underline{x})$	knapsack mapping
$A(n_1,n_2,..,n_i)$	set of all nonzero partial sums of i integers n_1, $\ldots,n_i$
$\pi^k(b_1,..,b_N)$	GF(2)-sum of all distinct kth-order products of N binary variables

c_{jk}	direct carry from position k to position j in an addition
s_j	jth bit of the knapsack sum S
$f_{j,K}(\underline{x})$	boolean function which produces s_j
$Nr(f_{j,K})$	number of different functions
FSM	finite-state machine
f_0	output function of an FSM
f_s	next-state function of an FSM
S_j	State of an FSM, as random variable
qSS	q-ary symmetric source
$Z^j=Z_1,..,Z_j$	finite sequence of j random variables
$ord_q(a)$	multiplicative order of a modulo q

Index