Reprint - Proceedings
INTERNATIONAL
ZURICH SEMINAR
March 6-8, 1984

A SELF-SYNCHRONIZING DIGITAL SCRAMBLER FOR CRYPTOGRAPHIC PROTECTION OF DATA

James L. Massey
Andrea Gubser
Alfred Fischer
Peter Hochstrasser
Bruno Huber
Roland Sutter

Telecommunications Institute
Swiss Federal Institute of Technology
8092 Zurich, Switzerland

ABSTRACT

A generalization of binary stream cipher systems is proposed that retains the desirable features of such systems (no ciphering delay, limited error-propagation, and analyzable security) but avoids their synchronization difficulty. Reception of a specified number of correct consecutive ciphertext digits always restores the decrypter of the proposed system to correct operation, regardless of the prior state of the decrypter. The general structure of such self-synchronizing inverse systems is identified, and a fundamental sufficient condition is obtained for such encrypters to be true scramblers in the sense of transducers of periodic sequences. This condition is shown also to be the condition for non-singularity of an associated nonlinear shift-register and for state irreducibility of the encrypter. Some principles for designing secure scramblers of the proposed type are given together with some lessons learned from an experimental study of such scramblers with a 16 bit key used to encrypt 32 Kbps delta-modulated speech.

1. INTRODUCTION

This paper introduces a type of cryptographic system that is intended to retain the main advantages of conventional stream cipher systems while avoiding their principal disadvantage. We begin our discussion, therefore, with a brief review of stream cipher principles.

In a binary stream cipher system, the plaintext $\underset{\sim}{x} = (x_1, x_2, x_3, \ldots)$ is a (semi-infinite) binary sequence that is converted into the ciphertext $\underset{\sim}{y} = (y_1, y_2, y_3, \ldots)$, another binary sequence, under the control of the key K in the manner shown in Fig. 1.1. The sequence $\underset{\sim}{z} = (z_1, z_2, z_3, \ldots)$ is called the

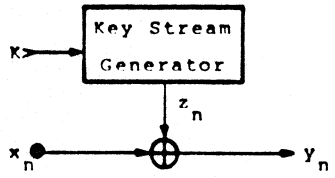

Fig. 1.1: A Binary Stream Cipher Encrypter

running key and is added bit-by-bit to the plaintext $\underset{\sim}{x}$ to produce the ciphertext $\underset{\sim}{y}$, i.e.,

$$y_n = x_n + z_n \qquad (1)$$

where here and hereafter the addition is modulo-two (i.e., addition in the finite field GF(2)) unless indicated otherwise. Adding z_n to both sides of (1), we obtain

$$y_n + z_n = x_n \qquad (2)$$

as follows from the fact that $0 + 0 = 1 + 1 = 0$ in GF(2), i.e., from the fact that addition and subtraction coincide in GF(2). It follows that the decrypter for a binary stream cipher coincides with the encrypter. A binary stream cipher system can thus be constructed as shown in Fig. 1.2. The transmission means

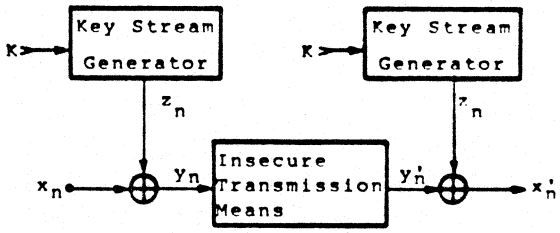

Fig. 1.2: A Complete Stream Cipher Cryptosystem

is designated "insecure" to imply that the enemy cryptanalyst can also observe the received binary ciphertext $\underset{\sim}{y}' = (y_1', y_2', y_3', \ldots)$. In the absence of transmission errors, $y_n' = y_n$ so that $x_n' = y_n' + z_n = x_n$ according to (2). Moreover, a single transmission error causes only a single decrypting error; this limited error-propagation is a very desirable feature of stream cipher systems. Another desirable feature is the delayless character of the encipherment; the first n bits of the plaintext can be obtained from the first n bits of the ciphertext for all n. The fact that the encipherment is delayless implies also that there is no expansion of the plaintext, i.e., that if the plaintext has finite length N then so also does the ciphertext.

When assessing the security of a stream cipher system, it is customary to assume a known-plaintext attack, i.e., to assume that the cryptanalyst knows the first N bits of the plaintext for some large N. From (2), it follows that this is equivalent to knowing the first N bits of the running key $\underset{\sim}{z}$. The system is said to be secure if, given $z_1, z_2, \ldots, z_N$, it is extremely difficult to predict reliably the future bits of the running key. In particular, if the system is secure, it must be extremely difficult to determine the secret key K from these N bits of the running key, as knowledge of K would permit perfect prediction of the entire running key. A desirable feature of key stream systems is that their structure lends itself to an analysis of their security, i.e., they possess analyzable security, at least to some practically significant degree if not to a completely satisfactory one. In general, one can describe a key stream generator by the relation

$$z_n = f_K(n), \qquad (3)$$

which emphasizes that, for each choice of the key K, the key stream generator is an autonomous (i.e., no input) sequential machine. By selecting this sequential machine always to be of a type whose security can be quantified, say a maximal-length linear feedback shift-register with nonlinear output logic [1]-[2], one can ensure a certain degree of security for the cryptosystem.

A binary stream cipher system, however, has one major disadvantage, namely its synchronization difficulty. This is the Achilles' heel of stream cipher systems that precludes their use in many practical situations. As can be seen from (2) and (3), it is in general necessary at the decrypter always to have the same clock time n as at the encrypter. The encrypter and decrypter must run in perfect synchronism. The decrypter can "jump into" an on-going ciphertext sequence only if it can determine the corresponding time instant n registered by the encrypter's clock. Bit losses or gains during transmission will result in garbles in the decrypted plaintext that can be remedied only be searching over the resulting possible offsets between the encrypter's and decrypter's clocks. These are serious practical problems whose solution generally constitutes an appreciable fraction of the cost and effort required to implement a stream cipher system.

This paper reports research whose goal was to design cryptosystems that retained insofar as possible the main advantages of binary stream ciphers but that were easy to synchronize. These design goals are made precise in the next section of this paper where it is proved that all such cryptosystems must have a particular system structure. Section 3 addresses the security of such cryptosystems and further specifies the system structure. Section 4 reports on an experimental system that was built and tested to confirm the practicality of such cryptosystems, and gives some of the lessons learned from this experimental study.

2. STRUCTURE OF SELF-SYNCHRONIZING GENERALIZED STREAM CIPHERS

A stream cipher encrypter is, for each choice of the key K, a finite-state machine (FSM) with a binary input and binary output alphabet. The general FSM of this type is described by the equations

$$y_n = \lambda_K(x_n, s_n) \tag{4}$$

$$s_{n+1} = \delta_K(x_n, s_n) \tag{5}$$

where s_n is the state at time instant n, λ_K is the output function and δ_K is the next-state function. Notice that both the next-state and the output functions may depend on the key K. A binary stream cipher encrypter is the special case of such a FSM where the next-state has no dependence on x_n (i.e., x_n is an idle variable in (5)) and where the dependence of the output function on the input is additive, i.e.,

$$\lambda_K(x_n, s_n) = x_n + f_K(s_n). \tag{6}$$

The former of these restrictions is rather arbitrary, but the latter is unavoidable if the encrypter is to be delayless, as we now prove.

Proposition 1: A binary-input binary-output FSM is invertible without delay (when the initial state s_1 is known) if and only if the output function is as specified by (6), i.e., if and only if

$$y_n = x_n + f_K(s_n). \tag{7}$$

Proof: Suppose first that (7) holds, then

$$x_n = y_n + f_K(s_n). \tag{8}$$

But (5) and (8) now show that the FSM of Fig. 2.1 is a delayless inverse of the original FSM when both FSM's have the same initial state.

Suppose conversely that (7), or equivalently (6), does not hold. Then there is some choice of s_n, say σ, such that $\lambda_K(0,\sigma) = \lambda_K(1,\sigma)$. Thus, when $s_1 = \sigma$, it follows from (4) that both $x_1 = 0$ and $x_1 = 1$ will yield the same value of y_1. Thus x_1 cannot always be recovered from s_1 and y_1 so that a delayless inverse does not exist.

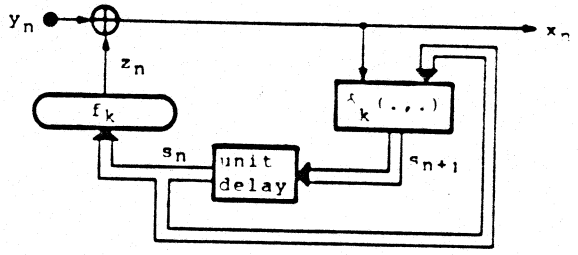

Fig. 2.1: A General Delayless Inverse for a FSM

We remark that Proposition 1 is the generalization of a result for FSM's with finite input memory due to Preparata [3]. We also remark that we have used the notation $z_n = f_K(s_n)$ in Fig. 2.1 to emphasize the correspondence to the decrypter for a stream cipher. In fact, if the next state function $\delta(x_n, s_n)$ in Fig. 2.1 has no dependence on the input variable x_n, then the general delayless inverse of Fig. 2.1 becomes precisely the decrypter for the general binary stream cipher of Fig. 1.2.

Our interest here is in cipher systems that are particularly simple to synchronize or, more precisely, that automatically synchronize themselves upon receipt of a given number M of correct consecutive ciphertext digits without any knowledge of the clock time at the encrypter. We shall say, therefore, that a delayless inverse of a binary-input binary-output FSM is self-synchronizing in M steps if the output digits of the inverse coincide with the input digits x_n of the FSM for all $n > M$, regardless of the initial states chosen for the FSM and for the inverse and regardless of the input sequence $\underset{\sim}{x}$. The smallest such positive integer M will be called the synchronization delay. The condition that a delayless inverse be self-synchronizing in M steps is equivalent to the condition that the "running key" digit z_n in Fig. 2.1 depends only on the "ciphertext" digits $y_{n-1}, y_{n-2}, \ldots, y_{n-M}$ for all $n > M$, i.e.,

$$z_n = g_K(y_{n-1}, y_{n-2}, \ldots, y_{n-M}) \tag{9}$$

for some boolean function g_K of M variables for all $n > M$.

It follows from (9) that any inverse that is self-synchronizing in M steps can be realized as shown in Fig. 2.2, at least in the sense that the circuit of Fig. 2.2 will duplicate the input/output behavior of the true inverse at all time instants n such that $n > M$. The structure of the circuit in Fig. 2.2 is that of a simple M-stage shift-register without feedback whose state determines the running key digits according to (9). It should be pointed out that the circuit of Fig. 2.2 may not duplicate the input/output behavior of the true inverse for time instants n, $1 \leq n \leq M$, for the reason that the original FSM [and hence also its true inverse] may possess "transient" states that can occur only within this initial time span; it is possible that some of these transient states in the true inverse are not

equivalent to any of the 2^M states of the circuit in Fig. 2.2. Such transient phenomena are, however, of no interest in a (generalized) stream cipher system. Thus, we might as well assume that the original FSM has no such transient states so that the circuit of Fig. 2.2 is a true inverse of the original FSM. Equivalently, the original FSM is an inverse of the circuit in Fig. 2.2 so that we may apply our previous reasoning (with the role of x_n and y_n interchanged) to conclude that the original FSM can be realized as shown in Fig. 2.3, which is the specialization of the inverse in Fig.2.1 for the FSM of Fig. 2.2. Note that the original FSM as shown in Fig. 2.3 has the structure of a feedback shift-register in which the input is an additive component of the feedback digit. We summarize our observations in the following proposition.

Proposition 2: Any FSM (without transient states) that has a delayless inverse that is self-synchronizing in M steps can be realized as shown in Fig. 2.3, and its inverse can be realized as shown in Fig. 2.2.

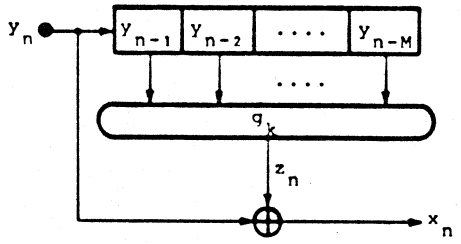

Fig. 2.2 : General Delayless Inverse that is Self-Synchronizing in M-Steps.

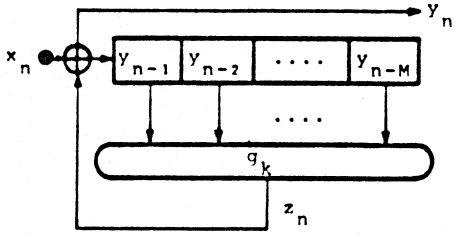

Fig. 2.3 : The FSM Inverted by the Circuit of Fig. 2.2 .

The FSM of Fig. 2.3 and its inverse in Fig. 2.2 are our desired generalizations of the delayless encrypter and decrypter in a stream cipher system with no synchronization difficulty. Even if the initial state of the encrypter is unknown, the decrypter (started in an arbitrary initial state) will begin decrypting correctly after M (possibly) incorrect output digits. Moreover, a transmission error (i.e., an incorrect value of y_n) will affect the output of the decrypter for only M + 1 time units as can be seen from Fig. 2.2. Thus, the error propagation is limited; a transmission error can cause at most M + 1 decryption errors. It remains to show that we can obtain cipher systems with "analyzable security" within this class of generalized stream ciphers.

3. SCRAMBLERS AND SECURITY CONSIDERATIONS

A scrambler is a FSM that, regardless of the initial state, converts a periodic input sequence into a periodic output sequence with generally higher, but at least the same, period. As remarked by Zegers [4, p. 62], scramblers "have a randomizing effect on the data patterns transmitted" and thus "can reduce the sensitivity of synchronization systems to specific periodic data patterns". As we shall soon argue, scramblers are also useful cryptographic devices. We first prove the following result, which will lead to a rather complete characterization of the connection between scramblers and FSM's with self-synchronizing delayless inverses.

Proposition 3: The following three statements are equivalent:

(i) The feedback function in Fig. 2.3 is of the form

$$g_K(y_{n-1}, y_{n-2}, \ldots, y_{n-M}) = y_{n-M} + f_K(y_{n-1}, \ldots, y_{n-M+1}). \quad (10)$$

(ii) With no input (i.e., with $x_n \equiv 0$), the FSM of Fig. 2.3 is a nonsingular nonlinear feedback shift-register (NFSR), i.e., an NFSR all of whose states lie on closed cycles.

(iii) The binary-input binary-output FSM of Fig.2.3 is state irreducible, i.e.,no two of its 2^M states are equivalent.

Proof: We first show the equivalence of statements (i) and (iii).

Suppose that (10) does not hold. Then there exist binary digits $b_1, b_2, \ldots, b_{M-1}$ such that $g_K(b_1, \ldots, b_{M-1}, 0)$ and $g_K(b_1, \ldots, b_{M-1}, 1)$ have the same value, say b_0. But then $s_1 = (b_1, \ldots, b_{M-1}, 0)$ and $s_1 = (b_1, \ldots, b_{M-1}, 1)$ in the FSM of Fig. 2.3 will both give $z_1 = b_0$ and hence will both give the same $y_1 = b_0 + x_1$ and the same $s_2 = (b_0 + x_1, b_1, \ldots, b_{M-1})$, for any x_1. Thus, $y_2, y_3, \ldots$, will also be the same so that the states $(b_1, \ldots, b_{M-1}, 0)$ and $(b_1, \ldots, b_{M-1}, 1)$ are equivalent.

Conversely, suppose that there are two distinct equivalent states, σ and σ', in the FSM of Fig. 2.3. Let i be the first component in which these equivalent states differ so that we may write $\sigma = (b_1, \ldots, b_{i-1}, 0, b_{i+1}, \ldots, b_M)$ and $\sigma = (b_1, \ldots, b_{i-1}, 1, b'_{i+1}, \ldots, b'_M)$. Consider applying the all-zero input sequence, $x_n \equiv 0$. Because the initial states are equivalent, the output sequences must be the same when $s_1 = \sigma$ and when $s_1 = \sigma'$, say $y = (a_1, a_2, a_3, \ldots)$. Thus, one obtains

$s_{1+M-i} = (a_{M-i}, \ldots, a_1, b_1, \ldots, b_{i-1}, 0)$ and

$s_{1+M-i} = (a_{M-i}, \ldots, a_1, b_1, \ldots, b_{i-1}, 1)$ according as

$s_1 = \sigma$ or $s_1 = \sigma'$, respectively. In either case,

$y_{1+M-i} = z_{1+m-i}$ must be the same, so it follows that

$z_{1+m-i} = g_K(a_{M-i}, \ldots, a_1, b_1, \ldots, b_{i-1}, 0) =$

$g_K(a_{M-i}, \ldots, a_1, b_1, \ldots, b_{i-1}, 1)$ and hence (10) does not hold. This completes the proof of the equivalence of statements (i) and (iii).

When $x_n \equiv 0$, the FSM of Fig. 2.3 becomes the autonomous nonlinear feedback shift-register (NFSR) shown in Fig. 3.1. Condition (10) is the well-known [5, pp. 115-116] necessary and sufficient condition that such an NFSR be nonsingular, i.e., that all its states lie on closed cycles. This follows from the fact that the NFSR will be nonsingular if and only if each state has a single predecessor. But the only possible predecessors of state $(b_1, b_2, .., b_M)$ in the NFSR of Fig. 3.1 are $(b_2, .., b_M, 0)$ and $(b_2, .., b_M, 1)$. Thus, there will be a state with two predecessors (and one with none) if and only if $g_K(b_2, .., b_M, 0) = g_K(b_2, .., b_M, 1)$ for some choice of $b_2, b_3, .., b_M$, but this is precisely the condition that (10) not be satisfied. It follows then that statements (i) and (ii)

are equivalent, and this completes the proof of the proposition.

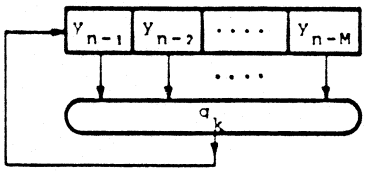

Fig. 3.1: The Nonlinear Feedback Shift-Register (NFSR) Obtained from the FSM of Fig. 2.3 when $x_n \equiv 0$.

When the circuit of Fig. 2.3 is employed as a generalized stream cipher, it appears desirable for security purposes that its state complexity be as large as possible. Thus, statement (iii) of Proposition 3 is a persuasive argument that the feedback function should be chosen, for each choice of the key K, such that (10) is satisfied. The following proposition provides a further argument that the feedback function should be so chosen.

Proposition 4: When the feedback function of the FSM in Fig. 2.3 satisfies (10), then this FSM is a scrambler. Moreover, when (10) is satisfied, the period of the output sequence is a multiple of the period of the input sequence; this multiple (which may depend on the initial state and on the input sequence) is at most 2^M.

Remarks: Scramblers of the type in Fig. 2.3 when f_K in (10) is a linear function were proposed by Savage [6] for randomizing "periodic" data patterns; equivalent linear scramblers were independently proposed in a different realization by Zegers [7]. These linear scramblers were all proposed precisely for their self-synchronizing properties. Proposition 4 generalizes a result proved by Savage for linear scramblers whose corresponding shift-register in Fig. 3.1 is a maximal-length linear feedback shift-register. It may well be that (10) is both a sufficient (as asserted in Proposition 4) and necessary condition for the FSM of Fig. 2.3 to be a scrambler when the synchronization delay is exactly M; however, we have not succeeded in proving this. Nonetheless, in light of Proposition 4, we shall use the terminology self-synchronizing digital scrambler (SSDS) to describe the FSM of Fig. 2.3 when the feedback function satisfies (10). The general SSDS is shown in Fig. 3.2 and the corresponding descrambler is shown in Fig. 3.3.

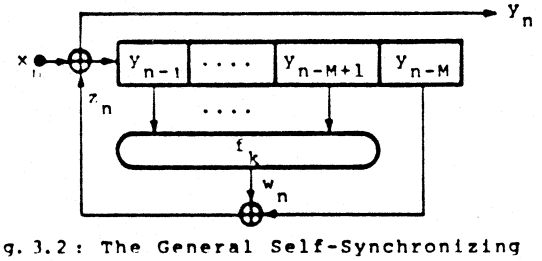

Fig. 3.2: The General Self-Synchronizing Digital Scrambler (SSDS).

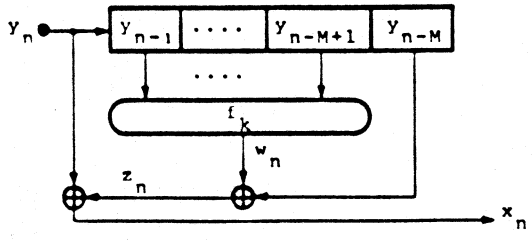

Fig. 3.3: The Descrambler for the General SSDS of Fig. 3.2.

Proof of Proposition 4: Let $(a_1, a_2, .., a_T)^\infty$ denote the semi-infinite periodic sequence consisting of repetitions of the sequence $a_1, a_2, .., a_T$; the period is either T or a divisor of T, and every sequence of period T can be so represented. Thus, when the input sequence $\tilde{x}$ in Fig. 2.3 is periodic with period T, the output sequence $\tilde{y}$ will be the same as the output sequence y of the autonomous FSM in Fig. 3.4 consisting of the FSM of Fig. 2.3, whose input is now the output sequence of a pure cyclicing register (PCR) of T stages whose initial loading is the first period of $\tilde{x}$. The output sequence $\tilde{y}$ will surely be periodic if the state sequence of this autonomous FSM is periodic. Moreover, the state sequence will be periodic if and only if every state $\sigma = (a_1, a_2, .., a_T, b_1, b_2, .., b_M)$ has a unique predecessor. But a predecessor of this state σ must be of the form $\sigma' = (a_T, a_1, .., a_{T-1}, b_2, .., b_M, c)$ where the binary digit c must satisfy

$$b_1 = a_T + g_K(b_2, .., b_M, c). \qquad (11)$$

But (10) is precisely the necessary and sufficient condition that (11) have a unique solution c for every choice of $b_1, b_2, .., b_M$ and a_T. It follows that (10) is a sufficient condition for $\tilde{y}$ to be periodic whenever the input $\tilde{x}$ to the FSM of Fig. 2.3 is periodic. To show that this FSM is a scrambler, it remains to show that the period of $\tilde{y}$ is a multiple of the period of $\tilde{x}$.

The period of the state sequence s of the FSM in Fig. 2.3 coincide with the period T' of the output sequence $\tilde{y}$ because $s_n = (y_{n-1}, y_{n-2}, .., y_{n-M})$. Now the running key $\tilde{z}$ is determined by (9) so that surely $z_{n+T'} = z_n$. But $\tilde{x} = \tilde{y} + \tilde{z}$ so it follows also that $x_{n+T'} = x_n$ and hence that the period T of $\tilde{x}$ is either T' or a divisor of T'. This proves that the FSM of Fig. 2.3 is indeed a scrambler whenever (10) is satisfied.

When $\tilde{x} = (a_1, .., a_T)^\infty$ has period T and $s_1 = (b_1, .., b_M)$, the period T' of the output sequence cannot exceed the period T" of the state sequence of the autonomous FSM in Fig. 3.4 when its initial state is $s_1' = (a_1, a_2, .., a_T, b_1, b_2, .., b_M)$. The first T components of s_n' will coincide with those of s_1' if and only if $n = 1 + i\,T$ for some positive integer i. Now consider the state subsequence $s_1', s_{1+T}', s_{1+2T}', ...$ All of these states have the same first T components. Because there are only 2^M possible values for the last M components of the state, this subsequence can have period at most 2^M. Thus,

$$s_1' = s_{1+NT}$$

for some integer N satisfying $1 \leq N \leq 2^M$. It follows that the state sequence has period at most $2^M T$, and thus the output sequence $\tilde{y}$ also has period at most $2^M T$. This completes the proof of Proposition 4.

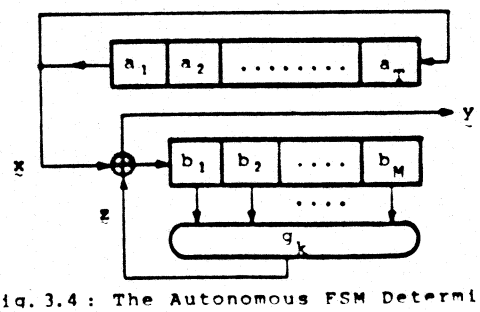

Fig. 3.4: The Autonomous FSM Determined by the FSM of Fig. 2.3 with an Input Sequence of Period T.

Remarks: It should be pointed out that the output sequence $\underset{\sim}{y}$ of the autonomous FSM in Fig. 3.4 can be periodic even when the state sequence is not. For example, it can readily be checked that the FSM in Fig. 3.5 (a) always gives a periodic output sequence, but the state sequence is not periodic when the initial state $s_1' = (a_1; b_1, b_2)$ is any of the following: (0; 1,0), (0; 0,1) or (1; 0,0). For instance, for $s_1' = (0; 0,1)$, the output sequence is $y = (0)^\infty$ of period 1, but the state sequence (0; 0,1), (0;0,0), (0,0,0), ... is not periodic. Note that the invertible subcircuit portion of Fig. 3.5(a) is an FSM of the type in Fig. 2.3 in which

$$g_K(y_{n-1}, y_{n-2}) = y_{n-1} \cdot y_{n-2} \tag{12}$$

so that (10) is not satisfied. The autonomous FSM of Fig. 3.5(b) consists of this same FSM, but now with a period T = 2 PCR providing the input. It can be checked that now, with the initial state $s_1' = (1,0; 1,0)$, neither the state sequence nor the output sequence is periodic. Thus, the invertible subcircuit described by (12) is not a scrambler. It is an open question whether (10) must be satisfied in order for the FSM of Fig.2.3 to be a scrambler.

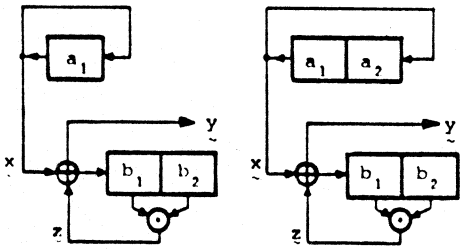

Fig. 3.5 : Two Autonomous FSM's Determined by the Same FSM of the Type in Fig. 2.3.

We turn our attention now more directly to security considerations for the SSDS shown in Fig. 3.2, for which the feedback function g_K is given by (10). We shall find it convenient to represent g_K in so-called algebraic normal form as a (modulo-two) sum of products of its factors; by way of convention, the binary constant 1 is considered a zero-order product. The first-order terms are also called linear terms. Note that (10) specifies that g_K must have at least one linear term. The order of g_K is the maximum of the order of its product terms; by way of convention, the zero function has order 0. For example,

$$g_K(b_1, b_2, \ldots, b_4) = b_1 b_2 b_3 + b_4$$

is a third-order function.

The linear complexity of a binary sequence, defined as the length of the shortest linear feedback shift-register (LFSR) that can produce this sequence as its output, has proved to be a very useful measure in the study of stream cipher systems. Roughly, the linear complexity is a measure of how difficult the sequence is to predict. If the linear complexity if L, there is a well-known and efficient algorithm [8] that finds the minimal generating LFSR after processing 2L digits and predicts the sequence exactly thereafter. Linear complexity has its limitations, however. Note that a periodic sequence has linear complexity at most equal to its period T since it can be produced by the PCR of length L, which is an LFSR. The sequence $(0,0,..,0,1)^\infty$ has linear complexity exactly equal to its period T, but is "easy to predict" if one is satisfied with good, but imperfect, prediction. Nonetheless, for the bulk of periodic sequences, linear complexity is a reasonable measure of the work that will have to be done by an observer of the sequence before he can reliably predict its future values.

In an important paper [2], Key has shown a result that can be stated equivalently for the FSM of Fig. 2.2 as follows: when the input sequence $\underset{\sim}{y}$ in Fig. 2.2 is a maximal-length sequence of period $2^M - 1$ and the function g_K has $g_K(0,0,..,0) = 0$ and has order N, then the sequence $\underset{\sim}{z}$ has linear complexity L satisfying

$$L \leq \sum_{i=1}^{N} \binom{M}{i}. \tag{13}$$

Moreover, equality, or near equality, will hold in (13) for most functions g_K of order N. It is important to note that Key's result speaks to the difficulty of predicting the entire future of $\underset{\sim}{z}$ from a small number of its leading digits when neither the particular maximal-length sequence $\underset{\sim}{y}$ nor the particular function g_K is known. In this case, we must examine the first 2L digits of $\underset{\sim}{z}$, for instance with the algorithm of [8], before the rest of $\underset{\sim}{z}$ can be predicted exactly. We next observe that, even when $\underset{\sim}{y}$ is known, there appears to be no way to simplify substantially the prediction of $\underset{\sim}{z}$ -- the simplest way still apears to be to ignore $\underset{\sim}{y}$ and to use the algorithm of [8] only on $\underset{\sim}{z}$. Moreover, even if one could find a practical way to use $\underset{\sim}{y}$ in the prediction of $\underset{\sim}{z}$, one would still need to process at least half as many digits of $\underset{\sim}{z}$ as when $\underset{\sim}{y}$ was unknown, as we shall now prove.

Proposition 5: If the sequence $\underset{\sim}{y}$ in Fig. 2.2 is such that knowledge of $\underset{\sim}{y}$ and the digits in B specified positions of the sequence $\underset{\sim}{z}$ uniquely determines g_K when g_K is known to have order N or less and to have $g_K(0,0,..,0) = 0$, then

$$B \geq \sum_{i=1}^{N} \binom{M}{i} \triangleq B_N. \tag{14}$$

Moreover, g_K is only very infrequently uniquely determined from $\underset{\sim}{y}$ and substantially fewer than B_N digits of the sequence $\underset{\sim}{z}$.

Remark: If the condition $g_K(0,0,..,0)$ is removed, the bound (14) on B can be improved by one by including the i = 0 term in the summation. We have not removed this condition only to make our result directly comparable to that of Key.

Proof: The function $g_K(b_1, b_2, .., b_M)$ has $\binom{M}{i}$ possible product terms of order i. Thus, B_N as defined in (14) is the number of distinct product terms with orders between 1 and N, inclusive. Thus, every function g_K of order N with $g_K(0,0,..,0) = 0$ can be identified with a binary vector of length B_N that has a 1 at each position where the corresponding product term is present in g_K , and every such vector corresponds to some such function. It follows that finding g_K is equivalent to solving for the B_N independent binary variables that form this vector, given $\underset{\sim}{y}$ and the known digits of $\underset{\sim}{z}$. But, according to (9) each known digit of $\underset{\sim}{z}$ provides one (not necessarily independent and in general nonlinear) binary equation satisfied by these B_N variables. There are at most 2^n possible values for the known n digits in $\underset{\sim}{z}$ and exactly 2^B possible values for the unknown B variables. Thus, the variables cannot all be determined uniquely unless $n \geq B_N$. Moreover, the maximum fraction $(2^n-1)/2^{BN}$ of functions that can be determined when $n < B_N$ is small when n is significantly less than B -- here we have used the fact that at least 1 of the at most 2^n possible values of the n known digits of $\underset{\sim}{z}$ must correspond to those functions that have not yet been determined uniquely from the n known digits of $\underset{\sim}{z}$.

Remark: Because a maximal-length sequence $\underset{\sim}{y}$ of period 2^M-1 has the property that $(y_{n-1}, y_{n-2}, \ldots, y_{n-M})$ takes on all 2^M-1 possible non-zero values for any 2^M-1 consecutive values of n, it follows that one can always determine g_K uniquely from $\underset{\sim}{y}$ and the first $B_M = 2^M-1$ values of $\underset{\sim}{z}$ when it is known only that $g_K(0,0,\ldots,0)=0$. This gives insight into Key's bound (13); but it is indeed surprising that (as Key has shown) the first $2B_N$ digits of $\underset{\sim}{z}$ always suffice to determine g_K when it is also known that g_K has order at most N. Our bound (14) shows that fewer than the first B_N digits would never suffice in this latter case no matter what was the nature of y. It seems inescapable to conclude that the identity of g_K is revealed most easily and rapidly (or at least nearly so) from observation of the digits of $\underset{\sim}{z}$ when the known sequence $\underset{\sim}{y}$ is a maximal-length sequence of period 2^M-1, in case the a priori knowledge of g_K is limited to the fact that g_K has order at most N and $g_K(0,0,\ldots,0) = 0$.

Consider now a known plaintext attack by an enemy cryptographer on the SSDS of Fig. 3.2. Suppose that the cryptoanalyst knows the first n digits of the plaintext $\underset{\sim}{x}$ (or, equivalently, any n consecutive digits of $\underset{\sim}{x}$). Suppose also that the cryptographer observes the ciphertext $\underset{\sim}{y}$ with no channel errors. Because $z_i = x_i + y_i$, the cryptographer equivalently knows the first n digits of $\underset{\sim}{z}$ as well as observes $\underset{\sim}{y}$. His task is to predict the future of $\underset{\sim}{z}$ or, equivalently, to determine the function f_K (since this determines g_K according to (10).) But this is precisely the problem considered above, and the argument above suggests how to choose f_K as a function of the key K so as to ensure that the cryptographer's task will be of at least a certain difficulty. For each choice of K, the function f_K should have some specified order N so that the cryptographer will generally be forced to know at least some large number B_N of plaintext digits for his attack to succeed, and the difficulty of his task will then generally be at least as great as that of predicting a sequence of linear complexity $2B_N$. This conclusion presupposes that the number of keys K is large enough that the cryptographer cannot sidestep the general problem of solving for f_K simply by trying all possible keys or by taking significant advantage of special properties of that subset of functions of order N that correspond to valid keys. If the function f_K is chosen according to these principles, the SSDS of Fig. 3.2 should be a very effective alternative to conventional stream cipher systems.

4. CASE STUDIES

To gain experience with SSDS's as generalized stream ciphers, hardware scramblers and descramblers were built as a semester project in the winter semester 1982/83 by two of the authors (Huber and Suter) and used to scramble 32 Kbps Δ-modulated speech signals. It had been decided to use at most a 16 bit key and, for convenience, memory M = 16. Two different scramblers were tested. For the first,

$$f_K(b_1,b_2,\ldots,b_{15}) = b_\alpha + b_1b_\beta + b_2b_\gamma b_\delta, \qquad (15)$$

where $1 \leq \alpha \leq 15$, $2 \leq \beta \leq 15$ and $3 \leq \gamma < \delta \leq 15$. All 16,380 valid choices of α,β,γ and δ were selectable by the key. For the second system,

$$f_K(b_1,b_2,\ldots,b_{15}) = b_\alpha + b_\beta + b_1b_\gamma + b_2b_\delta \qquad (16)$$

where $1 \leq \alpha < \beta \leq 15$, $2 \leq \gamma \leq 15$ and $3 \leq \delta \leq 15$. Again, all 19,110 valid choices of α,β,γ and δ were selectable by the key. Note that the first function f_K of (15) has order N=3 whereas the second function f_K of (16) has order N=2. Delta-modulated speech was chosen as the plaintext because of its ability to retain intelligibility in the face of a heavy error rate or, equivalently, a poor stream cipher.

For both systems, it was found that the scrambled signal could not be distinguished from random noise by a listener who heard the signal through a delta demodulator. In particular, pauses in the speech could not be detected. The same conclusion was found to obtain when the scrambled signal was played through a descrambler with the wrong key in whose corresponding function the linear terms were not all correct. However, when the only higher order terms of the key function were incorrect, it was possible to distinguish pauses from active speech although the speech itself was totally intelligible. This means that the effective size of the key is really only as great as that needed to specify the linear terms in f_K [about 4 bits for the function in (15) and about 8 bits for that in (16)], since one would need only to search the linear terms exhaustively until the "descrambled" signal allowed pauses and active speech to be distinguished. One could then proceed to determine the remainder of the key.

The importance of the linear terms can be explained as follows. Suppose that one used the test function $f_{K'}$ in the descrambler of Fig. 3.3 when the scrambler of Fig. 3.2 had the function f_K. The descrambler output x'_n would then be

$$x'_n = f_{K'}(y_{n-1},\ldots,y_{n-M+1}) + y_{n-M} + y_n \qquad (17)$$

$$= x_n + f_K(y_{n-1},\ldots,y_{n-M+1}) + f_{K'}(y_{n-1},\ldots,y_{n-M+1})$$

where we have made use of (10) and (9) to rewrite y_{n-M} and then have used (2). It follows from (17) that, when f_K and $f_{K'}$ have matching linear terms, then the output $\underset{\sim}{x'}$ will differ from the plaintext $\underset{\sim}{x}$ only by an additive sequence of higher order product terms of $\underset{\sim}{y}$. A product of order m will have probability only 2^{-m} of being 1 when $\underset{\sim}{y}$ is quite random as it generally will be. Thus $\underset{\sim}{x'}$ will be a not-too-noisy version of $\underset{\sim}{x}$ unless there are many non-matching higher-order product terms. For example, when $f_{K'}(b_1,\ldots,b_{15})=b_\alpha$ and f_K is given by (15), we have from (17)

$$x'_n = x_n + y_{n-1}y_{n-\beta} + y_{n-2}y_{n-\gamma}y_{n-\delta}$$

so that $\underset{\sim}{x'}$ differs from $\underset{\sim}{x}$ by an error rate of only about $1/4 + 1/8 - (1/4)(1/8) = 11/32$. At this error rate, Δ-modulated speech signals $\underset{\sim}{x}$ are completely unintelligible, but it is rather easy to distinguish pauses from active speech. When there are any mismatches between the linear terms in $f_{K'}$ and f_K, however, we see from (17) that the error sequence will have the desired error rate of 1/2.

A real-time attack on the SSDS systems of (15) and (16) was developed by two of the authors (Fischer and Hochstrasser) as a semester project in summer semester 1983. Using a small computer (PDP 11/24), their attacking system would typically succeed in finding the correct key in about one minute. The attack could be described as a "probabilistic known plaintext attack" that exploited special features of the functions f_K in (15) and (16). The attack will be explained only for the function f_K in (15), as the two attacks were quite similar.

From (15) and Fig. 3.2, we see that

$$x_n = y_n + y_{n-\alpha} + y_{n-1}y_{n-\beta} + y_{n-2}y_{n-\gamma}y_{n-\delta} + y_{n-16} \qquad (18)$$

Suppose now that x_n is known. Since $\underset{\sim}{y}$ is observed, we see that, for each n such that $y_{n-1} = y_{n-2} = 0$, we can obtain from (18) the linear equation

$$x_n = y_n + y_{n-\alpha} + y_{n-16}$$

and only a small number of such n (about 4 tries on the average since about half of the incorrect values of α will be eliminated on each try) need to be tried before α can be found. Once α is known, we see that, for each n such that $y_{n-1} = 1$ and $y_{n-2} = 0$, we obtain from (18) the linear equation

$$x_n = y_n + y_{n-\alpha} + y_{n-\beta} + y_{n-16}$$

and only about 4 such further values of n will be needed before β is determined. Finally, one determines γ and δ from choices of n where $y_{n-2} = 1$. Note that the possibility of such an attack depends on the fact that some factors in higher-order products are known in advance. The conclusion is that all factors in all product terms should depend on the key in a well-designed system.

In Δ-modulated speech, the sequence $\underset{\sim}{x}$ will consist of repetitions of the pattern (0,1) during pauses of the speech. The attack used by Fischer and Hochstrasser assumed that this was always the nature of $\underset{\sim}{x}$, solved for $(\alpha,\beta,\gamma,\delta)$ by the above described attack, and took the majority value of the solution for many such attacks. The correct key was produced in most cases from only a one second sample of $\underset{\sim}{y}$, i.e. from 32K bits of ciphertext.

ACKNOWLEDGEMENTS

We are grateful Dipl. Ing. F. Meier of Autophon AG in Solothurn for his encouragement of this research and for the furnishing of the Δ-modulation system used in our experiments. We are also grateful to Dipl. Ing. I. Wigdorowits of Brown, Boveri and Company in Baden for calling our attention to what was probably the first use of the self-synchronizing principle that underlies our scramblers as well as those of Zegers [7] and Savage [6]. This principle of incorporating feedback at the sending side to eliminate the need for feedback at the receiving side was used by the Swiss telecommunications pioneer, Gustav Guanella, to obtain self-synchronization of a frequency-scrambled speech system in a patent filed 40 years ago [9]!

REFERENCES

[1] E.J. Groth, "Generation of Binary Sequences with Controllable Complexity", IEEE Trans. Info. Th., vol. IT-17, pp. 288-296, May 1971.

[2] E.L. Key, "An Analysis of the Structure and Complexity of Nonlinear Binary Sequence Generators", IEEE Trans. Info. Th., vol. IT-22, pp. 732-736, Sept. 1976.

[3] F.P. Preparata, "Convolutional Transformation and Recovery of Binary Sequences", IEEE Trans. Computers, vol. C-17, pp. 649-655, July 1968.

[4] L.E. Zegers, "Common Bandwidth Transmission of Data Signals and Wide-Band Pseudonoise Synchronization Waveforms, Philips Research Reports Suppl., No. 4, 1972.

[5] S.W. Golomb, "Shift Register Sequences". San Francisco: Holden-Day, 1967.

[6] J.E. Savage, "Some Simple Self-Synchronizing Digital Data Scramblers", Bell System Tech.J., vol. 46, No. 2, pp. 449-487, February 1967.

[7] L.E. Zegers and J. Kuilman, "Transmission Systems for the Transmission of Pulses", U.S. Patent No. 3,421,146, January 7, 1969.

[8] J.L. Massey, "Shift-Register Synthesis and BCH Decoding", IEEE Trans. Info. Th., vol. IT-15, pp. 122-127, January 1969.

[9] G. Guanella, "Means for and Methods of Secret Signaling", U.S. Patent No. 2,405,500, August 6,1946.