New Approaches to the Design of Self-Synchronizing Stream Ciphers[1]

Ueli M. Maurer

Department of Computer Science
Princeton University
Princeton, NJ 08544

Abstract. Self-synchronizing stream ciphers (SSSC) are a commonly used encryption technique for channels with low bit error rate but for which bit synchronization can present a problem. Most presently used such ciphers are based on a block cipher (e.g. DES) in 1-bit cipher feedback mode. In this paper, several alternative design approaches for SSSCs are proposed that are superior to the design based on a block cipher with respect to encryption speed and potentially also with respect to security. A method for combining several SSSCs is presented that allows to prove that the combined SSSC is at least as secure as any of the component ciphers. The problem of designing SSSCs is contrasted with the problem of designing conventional synchronous additive stream ciphers and it is shown that different security criteria must be applied.
Furthermore, an efficient algorithm is presented for finding a function of low degree that approximates a given Boolean function, if such an approximation exists. Its significance for the cryptographic security of SSSCs and its applications in coding theory are discussed.

1. Introduction

Cryptographic protocols and techniques are often designed under the assumption that the communication channels are error-free. Such protocols can therefore be extremely vulnerable to unexpected errors on the channel. Many existing communication channels introduce some errors because their complete removal by the use of error-correcting codes would be too costly in terms of transmission rate reduction or encoder/decoder complexity. When encryption has

[1]This work was supported by Omnisec AG, CH-8105 Regensdorf, Switzerland.

D.W. Davies (Ed.): Advances in Cryptology - EUROCRYPT '91, LNCS 547, pp. 458-471, 1991.

to be built into an existing application on such a channel then no reduction of the data rate can be tolerated and thus no synchronization bits nor redundancy can be introduced for the purpose of error-correction. Therefore the encryption algorithm itself must be designed to be as error-resistant as possible to ensure that errors on the channel result in as little distortion of the deciphered plaintext as possible. For instance, when the channel introduces single bit errors, the optimal solution is to use a (conventional) synchronous additive stream cipher because every bit error in the ciphertext results in only a single bit error in the corresponding deciphered plaintext. This is in contrast to a block cipher in electronic codebook mode where a single bit error destroys the entire corresponding plaintext block. If the channel introduces bit slips (deletion or insertion of bits), however, both synchronous stream ciphers and block ciphers behave catastrophically since loss of synchronization results in a completely erroneous decryption of the entire following ciphertext.

A well-known cryptographic technique (e.g., see [5]) that is resistant against bit slips on the transmission channel (without introducing additional synchronization bits and without using an interactive higher-level protocol for recovering lost synchronization) is the use of a self-synchronizing stream cipher (SSSC for short). Properties and advantages of SSSCs are discussed in Section 2, and in Section 3 a theoretical treatment of SSSCs is given and several new design strategies for SSSCs are presented. The cryptographic security of SSSCs is discussed in Section 4. In Section 5, an efficient algorithm is presented for finding a function of low degree that approximates a given Boolean function, if such an approximation exists, and its significance in cryptography and coding theory is discussed.

2. Self-Synchronizing Stream Ciphers (SSSC)

The basic idea behind an SSSC (see Figure 1 for a canonical representation of an SSSC) is to encipher every plaintext digit using an encryption transformation that depends only on a fixed number M of previous ciphertext digits and on the secret key, but that does not depend on past plaintext digits other than through its dependence on past ciphertext digits. Therefore every ciphertext digit can be deciphered correctly when the previous M ciphertext digits have been received correctly. This self-synchronizing mechanism not only allows to resynchronize after bit slips on the channel, but it also enables the receiver to switch at any time into an ongoing enciphered transmission without knowing the current bit position within the message.

SSSCs are resistant against bit slips but on the other hand are less resistant than synchronous stream ciphers against single bit errors because every bit error in the ciphertext results in an error burst of length M in the deciphered plaintext. Hence SSSCs are suitable only when the bit error rate on the channel is sufficiently small. The trade-off between security and error-propagation is discussed in Section 4.

An alternative synchronization technique for stream ciphers is described in [7]. The running key generator of an ordinary additive stream cipher is reset whenever a given (secret or public) synchronization pattern (of length for example 16 bits) occurs in the ciphertext. The reset state of the generator is different each time since it depends not only on the secret key but also on a certain number of ciphertext bits following the synchronization pattern. For every bit error

probability and bit slip probability on the channel, this technique, even when modified so that a set of synchronization patterns rather than a single one triggers resynchronization, can be shown to have no advantages over SSSCs with respect to error-tolerance. Note that an SSSC can be viewed as a special case of this synchronization technique where the length of the synchronization pattern is zero, i.e., the generator is synchronized after every bit.

In addition to the resynchronization feature, SSSCs also offer two advantages over synchronous additive stream and block ciphers from a security point of view. First, the fact that single bit errors in the ciphertext result in error bursts in the plaintext prevents active eavesdroppers from undetectable tampering with the plaintext, thus assuring message authenticity. Note that when a synchronous additive stream cipher is used, plaintext bits can selectively be flipped by flipping the corresponding ciphertext bits. This would for instance allow an enemy to selectively modify the account number of an encrypted financial transaction. The second advantage is that unlike in additive stream ciphers or block ciphers, every plaintext bit influences the entire following ciphertext. Compared to block ciphers which are insecure when the plaintext is strongly redundant, i.e., when some plaintext blocks are likely to occur repeatedly, SSSCs are more resistant against attacks based on plaintext redundancy.

Most presently used SSSCs are based on a block cipher (e.g. DES) in 1-bit cipher feedback mode [13]. This mode is quite inefficient in terms of encryption speed since one block cipher operation is required for enciphering a single plaintext bit. Moreover, the published design criteria, security analysis and cryptanalytic attacks [4] for most block ciphers is restricted to the electronic codebook mode. While some design and security criteria are known for synchronous stream ciphers and block ciphers, only little is known about the design of SSSCs. The major goals of this paper are to narrow this gap and to contrast the problem of designing SSSCs with that of designing conventional synchronous additive stream ciphers.

3. Theoretical Foudations and Design Strategies

Without essential loss of generality, only binary SSSCs are considered in this paper. Let $B = \{0,1\}$. Let $\underline{X} = X_1, X_2, \ldots$, $\underline{Y} = Y_1, Y_2, \ldots$ and $\underline{W} = W_1, W_2, \ldots$ denote the binary plaintext, ciphertext and keystream sequences, respectively, and let Z denote the secret key that is chosen randomly (and uniformly) from the set $\mathcal{Z}$ of possible keys. The encryption transformation can be described by

$$Y_i = X_i \oplus W_i \tag{1}$$

for $i = 1, 2, \ldots$, where every keystream digit W_i depends on the previous M ciphertext digits $Y_{i-M}, \ldots, Y_{i-1}$ and the secret key Z, and where $Y_{-M+1}, \ldots, Y_0$ are predefined constants. Figure 1 shows a canonical representation of an SSSC.

The feedback part of an SSSC is a finite automaton with input the ciphertext sequence $\underline{Y} = Y_1, Y_2, \ldots$, with output the keystream sequence $\underline{W} = W_1, W_2, \ldots$ and with state sequence $\sigma_0, \sigma_1, \sigma_2, \ldots$. It has the special property that the input memory is finite and equal to M, i.e., the state depends only on the previous M input digits and the secret key, but is independent of all other input digits. This automaton is characterized by the state space Σ and the (possibly) key-dependent state-transition and output functions captured in the following definition.

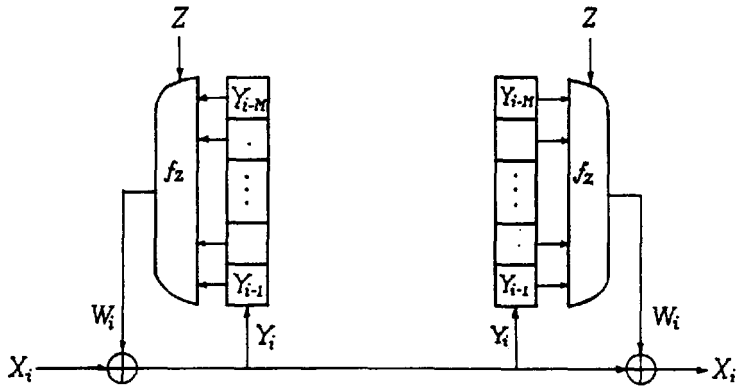

Figure 1: Canonical representation of a self-synchronizing stream cipher as a length M shift-register with a memoryless feedback function.

Definition: A *keyed state-transition function* for state space Σ, with memory M and key space $\mathcal{Z}$ is a family $G_{\mathcal{Z}} = \{g_z : z \in \mathcal{Z}\}$ of functions $g_z : \Sigma \times B \to \Sigma$ such that for all $z \in \mathcal{Z}$, when z is given, $g_z(\sigma, y)$ is easy to compute for all $\sigma \in \Sigma$ and $y \in B$, and such that the corresponding automaton has input memory M. A *keyed output function* for state space Σ and with key space $\mathcal{Z}$ is a family $H_{\mathcal{Z}} = \{h_z : z \in \mathcal{Z}\}$ of functions $h_z : \Sigma \to B$ such that for all $z \in \mathcal{Z}$, when z is given, $h_z(\sigma)$ is easy to compute for all $\sigma \in \Sigma$.

The encryption transformation of an SSSC with state-transition function $G_{\mathcal{Z}} = \{g_z : z \in \mathcal{Z}\}$ and output function $H_{\mathcal{Z}} = \{h_z : z \in \mathcal{Z}\}$, secret key Z and initial state σ_0 is specified by equation (1) and the two equations

$$W_i = h_Z(\sigma_{i-1}) \tag{2}$$

$$\text{and} \quad \sigma_i = g_Z(\sigma_{i-1}, Y_i) \quad \text{for } i = 1, 2, \ldots. \tag{3}$$

Every finite automaton with finite input memory M can be represented in a *canonical form* that consists of an input shift-register of length M with an attached memoryless function whose input is the shift-register state. In other words, the encryption transformation specified by equations (1)-(3) can equivalently be described by (1) and

$$W_i = f_Z(Y_{i-1}, \ldots, Y_{i-M}),$$

for $i = 1, 2, \ldots$, where the feedback function f_Z is completely specified by the secret key Z (capital letters denote random variables whereas small letter denote particular values that a random variable can take on), the state-transition function g_Z and the output function h_Z and where $Y_{-M+1}, \ldots, Y_0$ are specified by the secret key and the (for the key Z reachable) initial state σ_0 (see Figure 1). The concept of a memoryless key-dependent function is captured in the following definition.

Definition: A *keyed Boolean function (KBF)* of size M and with key space $\mathcal{Z}$ is a family

$F_Z = \{f_z : z \in \mathcal{Z}\}$ of functions $f_z : B^M \to B$ such that for all $z \in \mathcal{Z}$, when z is given, $f_z(x)$ is easy to compute for all $x \in B^M$.

The above defined canonical representation of a finite-memory automaton assigns a keyed Boolean function F_Z to every pair (G_Z, H_Z) of state-transition and output function. The security of an SSSC is determined by the properties of the corresponding KBF F_Z.

The design of most presently used SSSCs was motivated by the canonical representation of a finite-memory automaton. These SSSCs consist of a ciphertext shift-register of length M and a memoryless key-dependent function with input the state of the register (see Figure 1). Most often, a block cipher algorithm (e.g. DES) is used for implementing the key-dependent memoryless function, where the block cipher's input is the shift-register state and where all but one of the block cipher output bits are discarded. This mode of using a block cipher is usually referred to as 1-bit cipher feedback mode [13].

Note that in the canonical form, the state-transition function is extremely simple and independent of the secret key. Therefore the security of ciphers based on a block cipher in 1-bit cipher feedback mode relies entirely on the security of the output function. In this paper we suggest to design SSSCs that are based on a cryptographically secure state-transition function as well as on a cryptographically secure output function such that the SSSC is secure unless both functions are simultaneously insecure. In particular, we suggest to choose a state space with cardinality much greater than 2^M where clearly for a given key, at most 2^M states can be reached. The idea is that unlike for the conventional 1-bit cipher feedback mode, it should be infeasible for an enemy knowing the input sequence to even determine the state of the automaton. In addition, it should also be infeasible for an enemy to determine the output of the automaton, even if an oracle provided the (actually hidden) state sequence for free.

A general finite automaton has infinite input memory. In the following we present a method for designing state-transition functions that are guaranteed to correspond to an automaton with finite input memory M. Without essential loss of generality we assume in the following that the state can be represented by $T \geq M$ binary digits, i.e. $\sigma_i = (\sigma_i[1], \ldots, \sigma_i[T])$. Hence $\Sigma = B^T$. For $1 \leq k \leq T$, $\sigma_i[k]$ is a (memoryless) generally key-dependent function of the ciphertext digit Y_i, some of the variables $\sigma_{i-1}[1], \ldots, \sigma_{i-1}[T]$ and the secret key Z, but it is independent of the time i. One can show that there always exists a relabeling of the memory cells such that for $1 \leq k \leq T$, $\sigma_i[k]$ only depends on $\sigma_{i-1}[1], \ldots, \sigma_{i-1}[k-1]$, Y_i and Z and is independent of $\sigma_{i-1}[k], \ldots, \sigma_{i-1}[T]$. The dependence structure of such an automaton A is characterized by a loop-free directed graph G_A with vertex set $V = \{I, \sigma[1], \sigma[2], \ldots, \sigma[T], O\}$ and edge set E, where I and O denote the input and output, respectively. A directed edge from $\sigma[j]$ to $\sigma[k]$ indicates that $\sigma_i[k]$ functionally depends on $\sigma_{i-1}[j]$. Similarly, an edge from I to $\sigma[k]$ indicates that $\sigma_i[k]$ functionally depends on Y_i, and an edge from $\sigma[k]$ to O indicates that W_i depends on $\sigma_{i-1}[k]$. I has no incoming edges and O has no outgoing edges, and there exists no edge from I to O. The memory of A is given by the length of the longest path from I to O and can be determined as $\mu(O)$ where the function μ maps the vertices of G_A to the integers and is defined by $\mu(I) = 0$ and

$$\mu(k) \;=\; 1 + \max_{j:(j,k)\in E} \mu(j).$$

For a given dependence structure (i.e., a given matrix G_A), M can efficiently be determined by

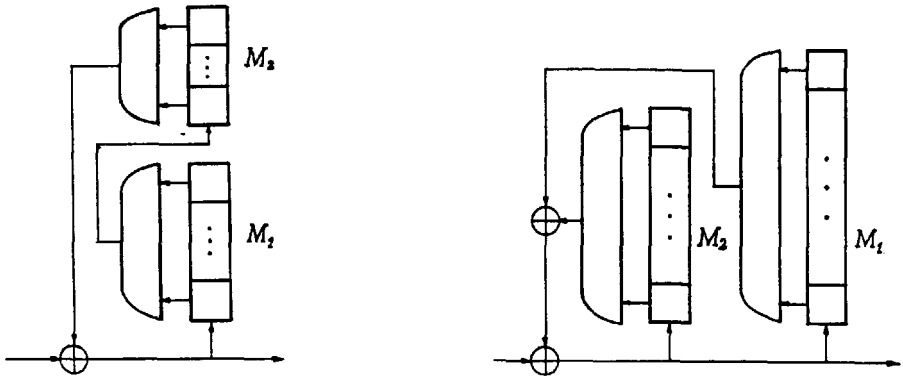

Figure 2: Serial composition (left) and parallel composition (right) of two SSSCs in the canonical form with memories M_1 and M_2, respectively. The serial and parallel compositions are equivalent to SSSCs with memory $M = M_1 + M_2$ and $M = \max(M_1, M_2)$, respectively.

a dynamic program.

An important concept in cryptography is that of composition. In the following we consider two different ways of combining two or more finite automata: parallel and serial composition. By the parallel composition of several automata we mean the automaton that is realized by connecting its input to every of the component automata's inputs and taking as its output the modulo-2 sum of the components' outputs. The memory of the parallel composition is the maximum memory of the component automata. By the serial composition of two automata A_1 and A_2 we mean the automaton whose input is the input of A_1, whose output is the output of A_2 and where the output of A_1 is connected to the input of A_2. The memory of the serial composition of two or more automata is equal to the sum of the component memories. Parallel composition allows to increase the number T of memory cells, i.e., the size of the state space, without increasing the memory. A theorem about the security provided by parallel feedback is stated in the next section. Figure 2 shows the serial and the parallel composition of two SSSCs in the canonical form.

By iterative application of parallel and serial composition, many component SSSCs that are relatively simple in terms of implementation complexity and memory length can be combined to form an SSSC realizing a very complicated keyed Boolean function with desired memory size. One possible architecture for realizing the state-transition function of an SSSC with memory $M = 8L$ and state space of size 2^{2M} using 16 SSSCs of memory length L is shown in Figure 3. The component SSSCs can all be different and can either be realized by a shift-register with a memoryless key-dependent output function or they can themselves be designed as the combination of several smaller component SSSCs. The design of Figure 3 consists of the serial composition of four component SSSCs, each component SSSC consisting of the parallel composi-

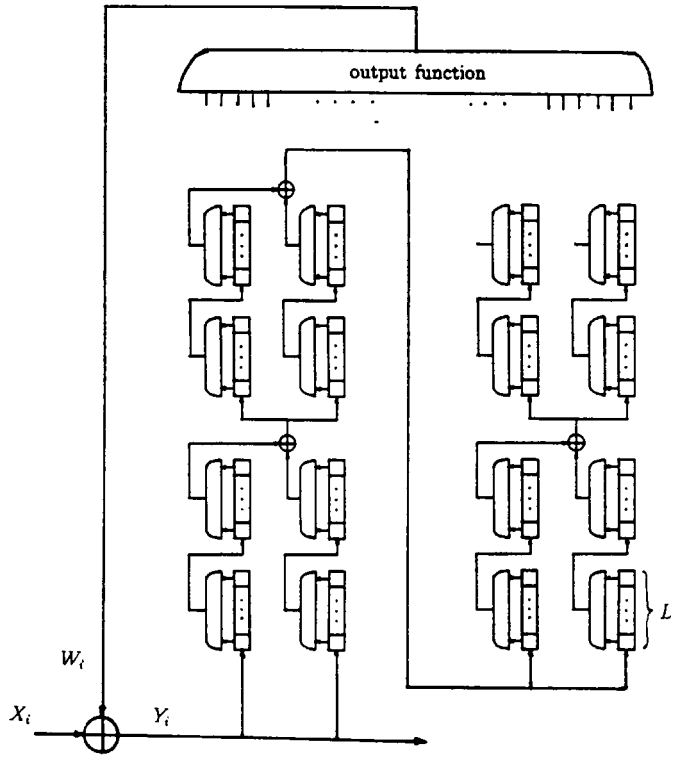

Figure 3: A possible design for the state-transition function of an SSSC by iterative application of serial and parallel composition of component SSSCs.

tion of two further component SSSCs, each of which is the serial composition of two elementary SSSCs with memory L. For instance, L could be 16 such that the resulting SSSC has memory $M = 128$, or L could be as small as 3 such that the resulting SSSC would have a memory of $M = 24$. Such an SSSC would not be secure by itself, but could be used as one component SSSC in the design of Figure 3 such that the finally resulting SSSC has memory $M = 192$. Clearly, the design of Figure 3 is only one example of a vast number of ways several component SSSCs can be combined. It should also be pointed out that only the state transition function is specified by the suggested design and that the output function would have to be some cryptographically secure function of all or part of the state.

The goal of a design strategy similar to the one described above, is to realize a cryptographically secure key-dependent state-transition function having the property that it should be infeasible to determine the state corresponding to a given input sequence. In addition to the potential improvement in cryptographic security compared the conventional design method of using a block cipher in 1-bit cipher feedback mode, another major advantage of the suggested architecture is the very high achievable encryption speed. When such a cipher is implemented

in hardware, all the component functions can be evaluated in parallel. The time required for enciphering one plaintext bit is the maximum of the execution times of all the memoryless component functions, including state transition and output functions. These memoryless functions can be realized by tables which could for instance be compiled by a key-dependent precomputation. The most time-consuming component function will usually be the overall output function that is applied to all or at least a substantial part of the state, which can for instance be realized by combining the results of several table lookups.

4. Cryptographic Security of SSSCs

A necessary condition for an SSSC to be secure is that M is sufficiently large such that the probability is negligible that a length M ciphertext pattern is repeated before the secret key is changed. When an enemy can observe two occurrences of the same length M ciphertext pattern he is able to compute the modulo-2 sum of the corresponding two plaintext bits.

For the purpose of determining the minimum memory M required to achieve a certain level of security, assume that the ciphertext can be modeled as a random sequence and moreover, that consecutive (overlapping) ciphertext patterns are independent. The probability $q(N, M)$ that in a sequence of N random binary length M patterns all patterns are distinct is given by

$$q(N, M) = \prod_{i=0}^{N-1} (1 - i2^{-M}).$$

Using the fact that $\ln(1 - x) \approx -x$ for small x we obtain

$$\ln q(N, M) = \sum_{i=0}^{N-1} \ln(1 - i2^{-M}) \approx -2^{-M} \sum_{i=0}^{N-1} i = -N(N-1)2^{-M-1}.$$

Thus when a probability p of a ciphertext pattern repeating can be tolerated and the expected plaintext length for a given secret key is N, M must be chosen such that $p < 1 - q(N, M) \approx \ln q(N, M)$, hence

$$M > \log_2 \left(\frac{N^2}{p} \right).$$

For example, for $p = 10^{-9}$ and $N = 10^{12}$, one should choose $M \geq 110$. Because bit errors on the channel propagate over the following M deciphered plaintext bits, M should on the other hand be as small as possible. To choose M between 80 and 128 seems to be a reasonable compromise for the tradeoff between security and error performance.

An important observation for SSSCs is that essentially no security can be gained by letting the initial state σ_0 of the finite automaton depend on the secret key because the states $\sigma_M, \sigma_{M+1}, \ldots$ are independent of the initial state and therefore an enemy can simply discard the first M ciphertext bits and attack the cipher based on the remaining ciphertext.

Although no presently-used cipher can rigorously be proved computationally secure, some necessary security criteria are known for synchronous stream ciphers [1, 15]. A synchronous stream cipher is insecure unless the period and the linear complexity of the keystream sequence are sufficiently large. These criteria cannot be applied to SSSCs because there exists no sequence

in the encipherment process that is independent of the plaintext and has the property that the SSSC is insecure unless the sequence's period and the linear complexity are large. However, it is shown in Section 4 that a necessary condition for an SSSC to be secure is that the corresponding KBF is (for all but a negligible fraction of the keys) not approximable by a Boolean function of low degree. Considering the limitations of the cryptographic significance of linear complexity for synchronous stream ciphers, this result is somewhat surprising for two reasons. First, the concept of linear complexity looses its original significance when generalized to quadratic or higher order complexity while such a generalization is possible in our case. Second, there exists no known efficient method for cryptanalyzing a synchronous stream cipher whose keystream sequence can only be *approximated* by a sequence with low linear complexity. For SSSCs, only the approximability by a low degree function is required for breaking the cipher. On the other hand, it should be pointed out that while lower bounds on the linear complexity of practical keystream generators can be proved, it seems to be extremely difficult to design SSSCs whose KBF can be proved to have no low degree approximation.

For the purpose of analyzing the security of a cipher it is generally assumed that the enemy cryptanalyst knows the cipher system precisely, but that he has no *a priori* information about the secret key. Moreover, one generally assumes that the enemy is capable of intercepting the ciphertext completely. Regardless of the particular implementation of an SSSC, an enemy can always consider its canonical form, i.e., he can analyze the corresponding KBF. Cryptanalytic attacks against a cipher are usually classified according to the type and amount of information about the plaintext the enemy cryptanalyst is assumed to have available. Commonly considered attacks are ciphertext-only, known-plaintext, chosen-plaintext and chosen-ciphertext attacks. Unlike for block ciphers where a chosen-plaintext attack is in general much more powerful than a known-plaintext attack (see [4]), these two attacks seem for SSSCs to be both equivalent to the enemy seeing certain randomly selected input/output pairs of the SSSCs keyed Boolean function.

The by far most powerful type of attack against an SSSC is the *chosen-ciphertext attack*. It allows an enemy to choose arguments of the KBF as he wishes, except that he is assumed to be unable to choose arguments occurring in the actual ciphertext. The task of cryptanalyzing an SSSC with KBF F_Z and key space $\mathcal{Z}$ in a chosen-ciphertext attack is equivalent to the problem of predicting $f_Z(\xi_1), f_Z(\xi_2), \ldots$, where $\xi_1, \xi_2, \ldots \in B^M$ are given and where Z is a randomly and uniformly (from $\mathcal{Z}$) selected secret key, when one can obtain for free the values of $f_Z(\tilde{\xi}_1), f_Z(\tilde{\xi}_2), \ldots$ for arbitrarily chosen $\tilde{\xi}_1, \tilde{\xi}_2, \ldots \in B^M$ such that the two sets $\{\xi_1, \xi_2, \ldots\}$ and $\{\tilde{\xi}_1, \tilde{\xi}_2, \ldots\}$ are disjoint.

The design goal for a cryptographic system is usually to make it secure against the most powerful conceivable attack for the strongest possible definition of security. Not only should it be infeasible for the enemy to determine any useful information about the plaintext or the secret key, but it should even be infeasible to determine any information whatsoever about the system that cannot be computed in an obvious way. It should for instance be infeasible to obtain any information about not obviously accessible intermediate results of the encryption process, even if this seems to be of no help in deriving information about the plaintext. In particular, it should be infeasible to distinguish the ciphertext from a truly random sequence for an appropriate definition of distinguishing.

For SSSCs, one of the strongest conceivable definitions of security is that the corresponding KBF be indistinguishable from a random function. We define a cryptographically secure KBF as follows. Let H_M be the set of 2^{2^M} Boolean functions with M inputs. Consider the following random experiment. Let S be a binary random variable that takes on the values 0 and 1 equally likely. S is assumed to be secret. Let g be a Boolean function with M inputs that is selected randomly from the set H_M when $S = 0$ and that is selected randomly from the set F_Z when $S = 1$.

Definition: A keyed Boolean function F_Z of size M and with key space $\mathcal{Z}$ is *cryptographically secure* if it is computationally infeasible to distinguish F_Z from H_M, i.e., to predict S defined above with probability of being correct significantly greater than 1/2, when an oracle for the function g is available.

This paper is focussed mainly on the practical aspects of designing SSSCs and therefore the definitions are by intention only stated informally for the sake of simplicity. The above definition (as well as other definitions in this paper) could however be further formalized similar to the definition of a pseudorandom function generator (PRFG) introduced by Goldreich, Goldwasser and Micali [6] and Luby and Rackoff [9]. For instance, one could consider a family $\mathcal{F} = \{F^k_{B^{t(k)}} : k = 1, 2, \ldots\}$ of KBFs where for every $k \geq 1, 2, \ldots$, $F^k_{B^{t(k)}}$ is a KBF of size k with key space $B^{t(k)}$, and where $t(k)$ is an integer-valued key-length function. Such a family $\mathcal{F}$ of KBFs could be defined to be cryptographically secure if there exists no polynomial Q and polynomial-time (in k) algorithm T such that for all sufficiently large k, when given k as an input and when given access to an oracle for g corresponding to the KBF $F^k_{B^{t(k)}}$ as defined above, T can guess S with success probability at least $1/Q(k)$. A KBF could then be shown to be equivalent to a PRFG: A KBF can trivially be obtained from a PRFG by discarding all but one of the output bits and a KBF can be transformed into a PRFG by letting the function argument be the initial state of a feedback shift-register whose feedback function is the KBF, and defining the state of the shift-register after $k - 1$ shifts as the output of the pseudo-random function generator.

The following proposition demonstrates that a cryptocraphically secure SSSC can be transformed into a cryptographically secure synchronous additive stream cipher. This demonstrates that the design strategies presented in this paper have also applications to the design of conventional stream ciphers.

Proposition: *The autonomous finite automaton resulting by feeding the output of a finite automaton realizing a cryptographically secure KBF back to its input, is a pseudo-random number generator.*

The following theorem about the cryptographic security of the parallel composition of SSSCs or, equivalently, the sum of KBFs, can be proved using a similar argument as the one used in [11] for proving that the cascade of several conventional synchronous stream ciphers is at least as secure as the most secure component cipher. The theorem holds for virtually every reasonable definition of distinguishing and under virtually every assumption about the enemy's computational resources and knowledge.

Theorem: *The modulo-2 sum of several keyed Boolean functions with statistically independent keys is at least as difficult to distinguish from a random function as any of the component*

KBFs. Hence the parallel composition of several SSSCs with independent keys is at least as cryptographically secure as any of the component SSSCs.

This theorem suggests to design an SSSC as the parallel combination of two or more SSSCs with independent keys. Each component SSSC can be based on an entirely different design principle. As long as at least one of the component SSSCs is secure, the combined SSSC is provably at least as secure, but intuitively even much more secure. The risk that all design strategies fail simultaneously is much smaller than the risk that each design strategy fails individually.

The goal of designing an SSSC must be to make the corresponding KBF cryptographically secure, i.e., indistinguishable from a random function. The design approaches discussed in this paper try to achieve this goal by designing the state-transition and the output function of the finite automaton *both* to be cryptographically secure.

A state-transition function can be defined to be cryptographically secure if it is infeasible to determine the state for a given length M input sequence, even when an oracle would provide the state for free for any other length M input sequence. Note that for the state-transition function of Figure 3 to be secure against such a hypothetical attack it would have to be improved by introducing additional dependencies between the component SSSCs. The reason is that for every memoryless function with only few inputs, a table realizing the function could be determined by exhaustive search using the above mentioned oracle.

5. Low Degree Approximations for Boolean Functions

Assume that the algebraic normal form of the feedback function $f_z(x_1, \ldots, x_M)$ in an SSSC is of the special form that almost all coefficients are known to be zero, and only few coefficients can possibly be non-zero. Then an enemy who is able to perform a chosen-ciphertext attack can break the cipher since he can determine the coefficients of f by solving a system of linear equations of size the number of potential non-zero coefficients.

Assume now it is only known that there exists a simple function $g(x_1, \ldots, x_M)$ that ϵ-approximates f, i.e., that agrees with f for at least a fraction $1 - \epsilon$ of the 2^M arguments. Knowing g would allow the enemy to determine the plaintext bits with error rate ϵ. When the plaintext is sufficiently redundant, even an error rate of $25 - 30\%$ would allow to determine the plaintext precisely. However, the problem of finding g corresponds to the problem of *approximately* solving a system of linear equations over $GF(2)$, i.e., of finding the solution that satisfies the most equations. This problem is equivalent to the problem of decoding a linear code to the nearest codeword, which is for general linear codes believed to be a very difficult problem. In fact, this problem is NP-complete [2].

However, for certain special types of codes there do exist efficient decoding algorithms. Moreover, a significant step towards decoding general linear codes has recently been announced [8]. Because the codewords in our application have length 2^M and are thus too long to be even only read in feasible time, general decoding algorithms are of no use however. In this section we present a *local decoding algorithm* for certain classes of codes that not only provides necessary security criteria for SSSCs but also has applications in coding theory.

Consider as a first example the problem of determining the best affine approximation

$$g(x_1,\ldots,x_M) = a_0 + a_1x_1 + a_2x_2 + \cdots + a_Mx_M$$

to a given function $f(x_1,\ldots,x_M)$. Every coefficient a_i for $1 \leq i \leq M$ can be expressed in many different ways as the sum of two function values. More precisely, a_1 can for every choice of $x_2,\ldots,x_M$ be expressed as $a_1 = g(0,x_2,\ldots,x_M) + g(1,x_2,\ldots,x_M)$ and therefore 2^{M-1} independent estimates of a_1 of the form $f(0,x_2,\ldots,x_M) + f(1,x_2,\ldots,x_M)$ can be obtained. When f agrees with some affine function g for more than 3/4 of the arguments, a majority decision for the above 2^{M-1} values yields the coefficient a_1. The coefficients $a_2,\ldots,a_M$ can be determined analogously, and a_0 can be determined by a majority decision over all 2^M bits $f(x_1,\ldots,x_M) - a_1x_1 - \cdots - a_Mx_M$.

When only linear rather than affine functions are considered (i.e., $a_0 = 0$), the described procedure can be interpreted as a decoding algorithm for the dual code of a $(2^M-1, 2^M-M-1)$ Hamming code [12]. The minimum distance of this code is 2^{M-1} so that only 25% errors are guaranteed to be corrected, but when errors occur randomly and independently, close to 50% errors can be corrected with high probability using the above procedure. Observe that for large M, even when a majority decision is made only from a small subset of the 2^{M-1} terms, this procedure still allows to correct close to 50% errors with high probability. In other words, this code can be decoded with a local decoding procedure that examines only a small fraction of the word to be decoded.

Consider now the more difficult problem of finding the best approximation $g(x_1,\ldots,x_M)$ of degree at most r to a given Boolean function $f(x_1,\ldots,x_M)$. g can be expressed in the algebraic normal form as

$$g(x_1,\ldots,x_M) = \sum_{S\subseteq\{1,\ldots,M\}:|S|\leq r} a_S \prod_{i\in S} x_i.$$

The sum is over all index sets $S \subseteq \{1,\ldots,M\}$ of cardinality at most r.

One can show that any two Boolean functions of degree at most r differ for at least a fraction 2^{-r} of the arguments. This is equivalent to saying that the minimum distance of an r-th order Reed-Muller code of length 2^M is 2^{M-r} [3]. Hence it is theoretically possible to uniquely determine the best r-th degree approximation g to a given function f provided that it differs in less than a fraction 2^{-r-1} of the function values. For instance, an error rate of up to 1/8 can be tolerated for finding the best second order approximation.

However, it is completely infeasible to examine all 2^M values of f, and therefore it seems to be infeasible to find an r-th degree approximation when the error rate is not considerably less than 2^{-r-1}. Surprisingly, a solution to this problem exists that is based on a novel local decoding procedure for Reed-Muller codes. Instead of finding several orthogonal expressions for every coefficient individually, we find systems of linear expressions for small subsets of the coefficients that can be solved by the (complete) decoding algorithm for much shorter Reed-Muller codes. We choose sets of $L > r$ variables $\{x_{i_1}, x_{i_2}, \ldots, x_{i_L}\}$ and consider the 2^L values of f for those arguments where the remaining variables take on the value 0. These function values depend solely on the $\sum_{i=0}^{r}\binom{L}{i}$ coefficients of the form $a_{S'}$ for $S' \subseteq \{i_1,\ldots,i_L\}$ and thus a system of 2^L linear expressions for $\sum_{i=0}^{r}\binom{L}{i} < 2^L$ unknowns is obtained or, equivalently, the codeword of an r-th order Reed-Muller code of length 2^L evaluated for the information bits equal to the coefficients with indices in the

set $\{i_1, , \ldots, i_L\}$, i.e., the coefficients $a_0, a_{i_1}, \ldots, a_{i_L}, a_{i_1 i_2}, \ldots, a_{i_{L-1} i_L}, a_{i_1 i_2 i_3}, \ldots, a_{i_1, i_2, \ldots, i_L}$. The decoding procedure for Reed-Muller codes that is fast for short codes can thus be used to determine these coefficients correctly when no more than $2^{L-r-1} - 1$ errors are among the 2^L function values.

There exist two somewhat different strategies for repeatedly using this local decoding procedure to determine all coefficients of the approximating function g. A first strategy is to choose sufficiently many sets of L variables such that all coefficients of g appear in at least one of the systems. The problem of choosing the subsets of variables of size L is related to the graph-theoretic problem of covering the edges of a complete hypergraph on M vertices with complete subgraphs on L vertices. The second strategy, in which L must be chosen smaller than in the first strategy in order for the algorithm to be feasible, is to consider all $\binom{M}{L}$ size L sets of variables and to make a majority decision over all obtained solutions for every coefficient. For a coefficient of order s $(0 \leq s \leq r)$ the number of solutions is $\binom{M-s}{L-s}$.

The second strategy suggests a new class of linear error-correcting codes with $\sum_{i=0}^{r} \binom{M}{r}$ information bits and codeword length $\sum_{i=0}^{L} \binom{M}{L}$ for some choices $r < L < M$. The encoding procedure is simply to evaluate the r-degree function of size M defined by the information bits taken as coefficients at all arguments of Hamming weight at most L. The information rate of these codes is much higher than that of the corresponding Reed-Muller codes while their error-resistance is nevertheless comparable to that of Reed-Muller codes. This is true although the minimum distance of the new codes is only

$$d_{\min} = \sum_{i=0}^{L-r} \binom{M-r}{i}$$

compared to 2^{M-L} for the corresponding Reed-Muller code. In other words we propose a decoding procedure for strongly truncated Reed-Mullers codes that is efficient even when decoding (and even encoding) the full length 2^M code is completely infeasible.

6. Conclusions

New approaches to designing self-synchronizing stream ciphers have been presented whose security is based both on a cryptographically-secure state-transition function of the corresponding finite automaton as well as on a cryptographically-secure output function and is argued to be potentially much higher than for the conventional design based or a block cipher in 1-bit cipher feedback mode. Another advantage of the presented design strategy is its suitability for high-speed applications. A necessary condition for an SSSC to be secure is that there exists no function of sufficiently small degree r (e.g. $r \leq 10$) that agrees with the feedback function in at least a fraction $1 - 2^{-r-1}$ of the function values.

Acknowledgements

I am grateful to Dr. P. Schmid and Martin Benninger of Omnisec AG for many helpful discussions and for their generous support, and to Laszlo Lovász and Jim Massey for their comments.

References

[1] H. Beker and F. Piper, *Cipher systems: the protection of communications*, New York, NY: Van Nostrand, 1982.

[2] E.R. Berlekamp, R.J. McEliece and H.C.A. van Tilborg, On the inherent intractability of certain coding problems, *IEEE Transactions on Information Theory*, vol. 24, pp. 384-386, 1978.

[3] R.E. Blahut, *Theory and practice of error control codes*, Reading, MA: Addison-Wesley, 1984.

[4] E. Biham and A. Shamir, Differential analysis of DES-like cryptosystems, presented at CRYPTO'90 (to appear in the proceedings).

[5] D.E.R. Denning, *Cryptography and data security*, Reading, MA: Addison-Wesley, 1982.

[6] O. Goldreich, S. Goldwasser and S. Micali, How to construct random functions, *Journal of the ACM*, vol. 33, no. 4, pp. 792-807, Oct. 1986.

[7] H.-J. Klemenz and W.R. Widmer, Swiss Patent Nr. CH 658 759 A5.

[8] V.I. Korzhik and A.I. Turkin, Cryptanalysis of McEliece's public-key cryptosystem, these proceedings.

[9] M. Luby and C. Rackoff, How to construct pseudorandom permutations from pseudorandom functions, *SIAM Journal on Computing*, vol. 17, no. 2, pp. 373-386, 1988.

[10] J.L. Massey, Shift-register synthesis and BCH decoding, *IEEE Transactions on Information Theory*, vol IT-15, no. 1, pp. 122-127, Jan. 1969.

[11] U.M. Maurer and J.L. Massey, Cascade ciphers: the importance of being first, presented at the 1990 IEEE Int. Symp. on Information Theory, San Diego, CA, Jan. 14-19, 1990 (also submitted to J. of Cryptology).

[12] F.J. MacWilliams and N.J. Sloane, *The theory of error-correcting codes*, Amsterdam: North-Holland, sixth printing, 1988.

[13] National Bureau of Standards, DES modes of operation, Fed. Inform. Proc. Standards Publication 81, Nat. Inform. Service, Springfield, VA, Dec. 1980.

[14] N. Proctor, A self-synchronizing cascaded cipher system with dynamic control of error propagation, *Advances in Cryptology - CRYPTO '84*, Lecture Notes in Computer Science, vol. 196, Berlin: Springer Verlag, pp. 174-190, 1985.

[15] R.A. Rueppel, *Analysis and design of stream ciphers*, Berlin: Springer Verlag, 1986.