Modes of Operation of Stream Ciphers

Jovan Dj. Golić[*]

School of Electrical Engineering, University of Belgrade,
Bulevar Revolucije 73, 11001 Beograd, Yugoslavia
golic@galeb.etf.bg.ac.yu

Abstract. A general stream cipher with memory in which each ciphertext symbol depends on both the current and previous plaintext symbols, as well as each plaintext symbol depends on both the current and previous ciphertext symbols, is pointed out. It is shown how to convert any keystream generator into a stream cipher with memory and their security is discussed. It is proposed how to construct secure self-synchronizing stream ciphers, keyed hash functions, hash functions, and block ciphers from any secure stream cipher with memory. Rather new and unusual designs can thus be obtained, such as the designs of block ciphers and (keyed) hash functions based on clock-controlled shift registers only.

Key words: Stream ciphers, block ciphers, keyed hash functions, hash functions, conversions, security.

1 Introduction

The electronic codebook (ECB) mode of block ciphers is mostly confined to encryption of relatively short messages in cryptographic protocols for confidentiality and/or authentication purposes. To increase the resistance to cryptanalysis, various modes with memory have been suggested, such as the output feedback (OFB) mode, the cipher block chaining (CBC) mode, the cipher feedback (CFB) mode, and the counter mode (e.g., see [22]). If a block cipher is used for encryption in one of these modes with memory, it then essentially becomes a stream cipher whose next-state and/or output functions are determined by the secret-key-dependent encryption and/or decryption functions of the block cipher.

The CBC and CFB as well as some other modes with memory [20] can be used to produce a message authentication code (MAC), usually also called a keyed hash function (KHF), which can be combined with encryption as well. It is proved in [2] that the CBC-MAC (used without CBC encryption) is at least as secure as the underlying block cipher with respect to the well-known pseudorandom function probabilistic model. A more general theoretical framework for the relative security analysis of symmetric encryption modes is proposed in [3]. Block ciphers can also be used in a number of various modes with memory to build hash functions (HF's) (e.g., see [20]).

[*] Part of this work was done while the author was with the Information Security Research Centre, Queensland University of Technology, Brisbane, Australia

However, stream ciphers need not be based on block ciphers, that is, on one-to-one functions that are difficult to compute in both directions without knowledge of the secret key. Instead, there exist many proposals of practically secure stream ciphers (keystream generators) whose next-state and/or output functions are very simple and whose initial state is controlled by the secret key (e.g., see [21], [22], and [19]).

A general way to produce a KHF from a keystream generator and an unconditionally secure MAC (authentication code, which is typically based on some linear operations modulo an integer or modulo a polynomial) is to use the keystream sequence to define the time-variant MAC secret key (see [23], [12], and [14]). A different construction directly based on a keystream generator is given in [15], but is shown to be insecure in [23]. The KHF can be combined with the keystream generator encryption, but then an additional portion of the keystream sequence is required.

In [1] it is suggested how to build block ciphers of unbalanced Luby-Rackoff type, with just a few rounds, from a keystream generator and a KHF or a HF. Note that a KHF itself can be produced from a HF by incorporating the secret key in the message. Similar constructions are also proposed in [16], along with a more elaborate security analysis.

The main objective of this paper is to show how to construct secure self-synchronizing stream ciphers, keyed hash functions, hash functions, and block ciphers from secure stream ciphers in a general, simple, and direct way. The crucial point is that instead of the keystream generator mode, which is almost exclusively treated in the open literature, we make use of a more general stream cipher mode in which each ciphertext symbol depends not only on the current plaintext symbol, but also on all the previous plaintext symbols. We discuss the security of all the modes proposed.

A formal proof that a derived mode of operation is at least as secure as the original mode with respect to certain attacks means that any efficient attack on the derived mode gives rise to an efficient attack on the original mode. However, if the original mode is not proved to be secure itself, as is the case with all known practical stream and block ciphers and (keyed) hash functions in the open literature, then formal security reduction proofs only show alternative ways of developing efficient attacks on the original mode. If the original mode is considered heuristically secure with respect to known attacks, then formal security reduction proofs do not imply that the derived modes are such, because they may be vulnerable to previously unknown, specially adapted attacks. Moreover, in our case, the security of the underlying stream cipher mode with plaintext memory has not been considered in the open literature. Accordingly, apart from the formal security analysis, we also deal with the practical (heuristic) security analysis of the modes proposed.

As usual (e.g., see [18] and [1]), the formal security statements and proofs are presented in a general language, which can be made mathematically precise by assuming various mathematical models for cryptanalytic attacks, such as the polynomial-complexity algorithms.

The stream cipher with memory (SCM) and other stream cipher modes are briefly described in Section 2. A general way of converting a stream cipher from the keystream generator mode into the SCM mode is proposed in Section 3 and practical security analysis of both the modes is addressed in Section 4. Conversions of a stream cipher with memory into a self-synchronizing stream cipher, a keyed hash function, a block cipher, and a hash function are presented in Sections 5, 6, 7, and 8, respectively, together with their security analyses. A proposal of a simple class of stream ciphers with memory to be used is described in Section 9. Conclusions are given in Section 10.

2 Stream Cipher with Memory (SCM) Mode

Let $x = (x_t)_{t=0}^{\infty}$ and $y = (y_t)_{t=0}^{\infty}$ denote the plaintext and ciphertext binary sequences, respectively. Let the binary strings k and r stand for the secret and randomizing keys, respectively, and let $s = (s_t)_{t=0}^{\infty}$ denote the internal state sequence where s_t is a binary string of length M and $s_0(k, r)$ is the initial internal state determined by k and r. Then, a general binary stream cipher decipherable without delay is an invertible nonautonomous finite-state machine with one input and one output that maps an input sequence x into the output sequence y by the encryption (sequential) transform recursively defined by

$$s_{t+1} \;=\; F_k(s_t, x_t), \quad y_t \;=\; x_t + f_k(s_t), \qquad t \geq 0 \tag{1}$$

where the addition is binary and $F_k : \{0,1\}^{M+1} \to \{0,1\}^M$ and $f_k : \{0,1\}^M \to \{0,1\}$ are the (secret-key-dependent) next-state and output functions, respectively. The inverse, decryption transform is recursively defined by

$$s_{t+1} \;=\; F_k(s_t, y_t + f_k(s_t)), \quad x_t \;=\; y_t + f_k(s_t), \qquad t \geq 0. \tag{2}$$

The encryption and decryption transforms are thus defined by the so-called keystream sequence $z = (z_t = f_k(s_t))_{t=0}^{\infty}$.

Let the input memories of the encryption and decryption transforms be referred to as the input and output memories of a stream cipher, respectively. All stream ciphers can be classified into the following three types with respect to the input and output memories: memoryless, with finite input or output memory, and with infinite input and output memory.

In the memoryless type, known as the keystream generator (KG) or the pseudorandom sequence generator type, the next-state function does not depend on the plaintext symbol, that is, $s_{t+1} = F_k(s_t)$, so that the keystream sequence z is plaintext independent. The KG type is sensitive to synchronization errors but has no substitution error propagation. To deal with a possible loss of synchronization when encrypting longer messages with the same secret key k, a long message is divided in shorter ones and a randomizing key r is used to reinitialize the keystream generator for every new message to be encrypted. It is typically sent in the clear and as such is public rather than secret. The randomizing key is generated in a random or deterministic way with the property of being with high probability different for every new message to be encrypted with the same secret key. This is in order to satisfy the so-called *one-time-pad assumption* that repetitions of long segments of keystream are highly unlikely. Namely, to this end, every new plaintext sequence should with high probability be encrypted by using a different initial state. More generally, repetitions of internal states should be highly unlikely.

In the finite input or output memory type, the encryption or decryption transform has finite input memory. In particular, in the self-synchronizing stream cipher (S^2SC) type (i.e., the cipher feedback type), the decryption transform has finite input memory, that is, $s_{t+1} = (y_{t-i})_{i=0}^{M-1}$, so that the keystream sequence depends on ciphertext only. As a consequence, the propagation of both synchronization and substitution errors in decryption is limited. Its security entirely depends on the output (feedback) function f_k which must be secret-key-dependent.

In the infinite input and output memory type, the next-state function effectively depends on the current plaintext symbol in such way that both the encryption and decryption transforms have infinite input memory. This type is typically either not mentioned or just overlooked as a practical possibility in the open literature on stream ciphers (e.g., see [21], [22], and [19]). Such a type is called here the stream cipher with memory (SCM) type, to emphasize the fact that in encryption each ciphertext bit does not depend on the current plaintext bit only, but also on the previous plaintext bits as well as that in decryption each plaintext bit depends on the current and previous ciphertext bits. Note that the PKZIP stream cipher is of this type (see [5]).

The SCM type is therefore sensitive to both synchronization and substitution errors, but, due to infinite input memory, has an inherent potential that can be used for message integrity purposes. The errors on real channels should be dealt with by separate error-correction and/or detection codes and, in addition, by the resynchronization method. However, instead of using and transmitting the randomizing key as described above, one may just prepend the randomizing key to each new message and keep the initial state secret-key-dependent only, as is done in the PKZIP stream cipher. In this case, the randomizing key is encrypted rather than transmitted in the clear and hence need not be public.

The basic types of stream ciphers will also be referred to as the modes of operation of stream ciphers, because it will be shown that they can be converted into each other by simple and general constructions. Similarly, other cryptographic primitives constructed from stream ciphers will also be referred to as the modes of operation of stream ciphers.

3 Conversion of Keystream Generator (KG) Mode into SCM Mode

Any stream cipher in the KG mode can be converted into the SCM mode by letting the next-state function depend upon the current plaintext bit too. The main practical criterion to be respected in this regard is that a change of a single plaintext bit should give rise to a random looking change in the keystream (ciphertext) sequence to follow (forward propagation effect). Note that the KG mode should satisfy the same property, but with respect to changes of the initial state bits. So, the adaptation is easily achieved by adding the plaintext bit to one or more of the internal state bits, especially those with the significant forward propagation effect. For a nonbinary plaintext alphabet, the plaintext symbol should be incorporated by a function with the property that any change of this symbol necessarily gives rise to a change of one or more of the internal state variables. This can generally be achieved by using a quasigroup operation.

In shift-register-based keystream generators, the conversion is easily done by making the shift registers nonautonomous, that is, by adding the plaintext bit to the feedback bit for each of the shift registers, and especially those that are used for clock control. The output of such shift registers should then necessarily involve the first, input stage. In KGs based on the table-shuffling principle like RC4 (see [22]), the plaintext symbol should affect the internal state variables defining the table positions where the changes should be made.

4 Security Analysis of SCM and KG Modes

In view of (1), the change of one plaintext bit necessarily causes the change of the corresponding ciphertext bit in the KG and SCM modes as well as a pseudorandom change of only the subsequent ciphertext bits in the SCM mode. Therefore, without the one-time-pad assumption, cryptanalytic attacks generating new plaintext/ciphertext pairs by modifying some bits in the known pairs are feasible. This is relevant for achieving information authenticity. The one-time-pad assumption can be achieved either without using resynchronization or by using resynchronization by (with high probability) different randomizing keys.

We will discuss cryptanalytic attacks in a general, adaptive combined chosen plaintext and ciphertext scenario, possibly key related as well (e.g., see [18]). The situation is conceptually similar to one with block ciphers in the ECB mode, with a difference that we now deal with the plaintext/ciphertext strings rather than blocks and that each plaintext sequence effectively includes the randomizing key used. The attacks use a training set consisting of a number of known, arbitrarily, possibly adaptively, chosen plaintext/ciphertext string pairs obtained from the same secret key (and the same or different known randomizing keys), and possibly from a set of related keys as well. An attack is an algorithm that produces one or more new plaintext/ciphertext string pairs, not included in the training set, where either the plaintext or ciphertext string in each of these pairs is assumed to be given. As knowing a plaintext/ciphertext string pair in the KG mode is equivalent to knowing a keystream/ciphertext string pair, an attack on the KG mode is in fact an algorithm that reconstructs unknown portions of a keystream sequence from its known portions or, more generally, from known portions of a set of keystream sequences obtained from different randomizing keys and the same secret key.

Typically, it is assumed that the attacks work for any secret key. Ideally, the exhaustive search over the secret keys or over unknown plaintext or ciphertext strings should be the most efficient way to perform an attack. If an attack can recover only particular plaintexts from given ciphertexts, it is then said to be successful for such plaintexts only, e.g., corresponding to certain plaintext statistics. On the other hand, secret key reconstruction attacks are the best known examples of the attacks successful for all the plaintexts.

An attack on the SCM mode that produces new plaintext/ciphertext string pairs by modifying a number of end bits in a known plaintext/ciphertext string pair is said to be *trivial*. A trivial attack on the KG mode is defined similarly. As argued above, trivial attacks are always computationally feasible for any stream cipher in the SCM or KG mode if the one-time-pad assumption is not satisfied.

A stream cipher in the SCM or KG mode is said to be (practically) secure with respect to nontrivial cryptanalytic attacks if no suck attack is computationally feasible, relative to available computing power. This security essentially means that both the encryption and decryption transforms are infeasible to compute without knowing the secret key, under the one-time-pad assumption.

Since it does not appear possible to formally relate the security of the KG and SCM modes with respect to general attacks, we will concentrate on their practical security. It turns out that the security of the SCM mode is more related to the security of the KG mode with than without resynchronization, which, except for [6], is hardly analyzed in the open literature.

The fact that in the SCM mode the plaintext statistics is disguised by the plaintext dependent keystream sequence makes the cryptanalytic attacks successful for particular plaintexts only and the ciphertext-only cryptanalytic attacks both very unlikely. For the same reason, the attacks consisting in predicting the keystream without reconstructing the initial state (e.g., based on low linear [21] or 2-adic [13] complexities) and the attacks finding the statistical weaknesses of the keystream (e.g., [9]) are also unlikely to succeed. In addition, unlike the KG mode, assuming that the (prepended) randomizing key is known is not very realistic in the SCM mode. Also, the known plaintext/ciphertext scenario is generally less useful for the SCM than for the KG mode, as missing portions of the plaintext/ciphertext sequences present a difficulty which is harder to overcome in the SCM than in the KG mode.

On the other hand, the plaintext dependent keystream sequence may open new possibilities for secret key reconstruction attacks especially if resynchronization is used and if prepended resynchronization key is assumed to be known, although the cryptanalysis is more complicated. For a survey of various (initial state) secret key reconstruction attacks on the KG mode without resynchronization, see [21], [7], [10], and [19].

Since the randomizing key plays the role of known plaintext, the cryptanalytic methods for block ciphers in the ECB mode, such as the differential [4] and linear [17] cryptanalysis, in principle extend to the KG and SCM modes too, especially if the secret and randomizing keys are linearly combined together. They are less likely to succeed here because of the underlying iterative structure. On the other hand, if long all zero plaintext sequences are allowed to be encrypted, then, formally, any secret key reconstruction attack on the KG mode directly extends to the SCM mode. So, the basic criterion for the SCM mode is that the underlying KG mode is secure (e.g., this is not true for the PKZIP stream cipher cryptanalyzed in [5]).

If the prepended randomizing key is assumed to be known, then the SCM mode is required to be secure with respect to nontrivial cryptanalytic attacks (in particular, secret key reconstruction attacks), for any training set of known plaintext/ciphertext sequence pairs produced from the same initial state. To this end, when the plaintext symbol is introduced into a part of the next-state function of the underlying KG mode by a linear function, it appears reasonable to recommend that this part of the next-state function should not be linear, as a whole, with respect to the same type of linearity. Accordingly, in binary shift-register-based keystream generators at least one of the shift registers affected by the plaintext bit should be clock controlled or have nonlinear feedback.

5 Conversion of SCM Mode into Self-Synchronizing Stream Cipher (S^{2}SC) Mode

We should define the secret-key-controlled feedback function f_k of m binary variables, where m is the output memory size, on the basis of the SCM mode of a given stream cipher so as to prevent trivial attacks on the SCM mode. A general and simple way to achieve this is to use an SCM mode with the secret key k, with the initial state determined only by k, and with the plaintext sequence whose first m plaintext bits are defined by a given m-bit input to f_k and the remaining plaintext bits are fixed, possibly to zero. The output bit of f_k is then defined as the last ciphertext bit obtained after clocking the SCM mode m times and a specified additional number of times, typically, on the order of several (e.g., three) internal memory sizes M of the SCM mode. Similarly, a faster conversion is obtained by producing a block of n, $n \leq M$, ciphertext bits at a time, thus constructing an n-bit output feedback function f_k. In the S^{2}SC mode, its output is then combined with an n-bit block of plaintext at a time, e.g., by using the bitwise binary addition.

Any attack on the S^{2}SC mode is essentially an algorithm for producing the unknown outputs of the feedback function for one or more given inputs, where a training set of a number of input/output pairs of the feedback function is assumed to be known. The S^{2}SC mode of a stream cipher is said to be secure if no such attack is computationally feasible. In particular, the security can be defined with respect to secret key reconstruction attacks only.

Proposition 1. *If the underlying* SCM *mode is secure with respect to nontrivial cryptanalytic attacks, then the derived* S^{2}SC *mode is secure. If the underlying* SCM *mode is secure with respect to secret key reconstruction cryptanalytic attacks, so is the derived* S^{2}SC *mode.*

Proof. It should be proved that any computationally feasible attack on the S^{2}SC mode can be converted into a nontrivial attack on the SCM mode at an additional computational cost that is not comparatively significant. This is a direct consequence of the proposed construction, as any input/output pair for the feedback function can be extended into a plaintext/ciphertext string pair for the SCM mode by appending a required number of fixed plaintext bits to the input bits, where only the last bit (or the last n bits) of ciphertext is known. Accordingly, one can modify the training set for any attack on the S^2SC mode into the corresponding training set for an attack on the SCM mode and vice versa. As well, producing an unknown input/output pair of the feedback function directly extends into producing an unknown bit (or n bits) of ciphertext for a known plaintext string of the SCM mode. This is by definition a nontrivial attack on the SCM mode, since sufficiently many fixed plaintext bits are appended.

A similar proof holds for secret key reconstruction attacks. □

In other words, the derived S^2SC mode is at least as secure as the underlying SCM mode with respect to nontrivial attacks, as trivial attacks on the SCM mode are prevented by additional clocking.

6 Conversion of SCM Mode into Keyed Hash Function (KHF) Mode

A binary keyed hash function (KHF) or a message authentication code (MAC) is a secret-key-dependent function $\{0,1\}^l \to \{0,1\}^n$ that maps binary strings of variable length l (messages) into binary strings of a fixed length n, where $l \geq n$. This function should be easy to compute when the secret key is known. The main computational security property required from a KHF is that it should be computationally infeasible, without knowledge of k, to perform existential forgery in the (adaptive) chosen message scenario, that is, to find another, distinct message and its hash value under k, provided a set of (adaptively) chosen messages and their hash values under k is given. In particular, a KHF should be secure against any secret key reconstruction attack. Ideally, the exhaustive search over k should be the most efficient way to find k.

Our objective now is to show how to construct a KHF from a stream cipher in the SCM mode. The designs proposed in the literature are either dedicated or are based on block ciphers, hash functions, or unconditionally secure MACs combined with KGs. They typically require that the message length be a multiple of a given positive integer which is achieved by padding. Our construction allows an arbitrary message length and is solely based on a stream cipher in the SCM mode, which can be easily obtained from any KG mode (as shown in Section 3).

The construction is similar to the one proposed for the S^2SC mode. Let k be the secret key for the SCM mode of a given stream cipher with the initial state determined only by k and with the plaintext sequence whose first l plaintext bits are defined by the given message and the remaining bits are fixed, possibly to zero. Let $M \geq n$, where M is the internal memory size of the SCM mode. The hash value is then defined as the last n successive ciphertext bits obtained after clocking the SCM mode l times and a specified additional number of times, several (e.g., three) times bigger than M.

In a similar way as for the S^2SC mode, producing a message and the corresponding hash value under k for the defined KHF mode directly extends into producing n unknown ciphertext bits for a known plaintext string of the underlying SCM mode. Consequently, the following proposition is proved in essentially the same way as Proposition 1.

Proposition 2. *If the underlying* SCM *mode is secure with respect to nontrivial cryptanalytic attacks, then the derived* KHF *mode is secure. If the underlying* SCM *mode is secure with respect to secret key reconstruction cryptanalytic attacks, so is the derived* KHF *mode.*

In other words, the derived KHF mode is at least as secure as the underlying SCM mode with respect to nontrivial attacks, as trivial attacks on the SCM mode are prevented by additional clocking. The same proposition holds even if the hash value of a given message is used together with the corresponding ciphertext obtained by the SCM with the same k. Accordingly, the KHF mode can be combined with the SCM encryption with the same secret key. Additional protection measures or constraints typically required for the designs proposed in the literature (e.g., due to the finite input memory of the CBC or CFB decryption) are not here needed.

7 Conversion of SCM Mode into Block Cipher (BC) Mode

A binary block cipher is a secret-key-dependent one-to-one function $\{0,1\}^n \rightarrow \{0,1\}^n$ that maps binary strings of length n (plaintext blocks) into binary strings of the same length (ciphertext blocks), where the block length n is usually fixed and relatively short. The function is called the encryption function, and its inverse is called the decryption function. Both the functions should be easy to compute when the secret key k is known and infeasible to compute when k is not known, in the (adaptive) chosen plaintext/ciphertext scenario, possibly key related as well. More precisely, an attack on a block cipher is an algorithm that, on the basis of a training set consisting of a number of known, arbitrarily chosen plaintext/ciphertext block pairs, produces one or more new plaintext/ciphertext block pairs, where either plaintext or ciphertext blocks are assumed to be given. A block cipher is said to be secure with respect to cryptanalytic attacks if no suck attack is computationally feasible.

Typically, the objective of cryptanalytic attacks on block ciphers is to reconstruct the secret key, in which case the attacks are successful for all the plaintexts. The differential cryptanalysis [4] in the chosen plaintext scenario and the linear cryptanalysis [17] in the known plaintext scenario are the well-known examples. The vast majority of existing proposals for block ciphers use the product structure composed of a number of rounds each involving a relatively simple one-round function, such as the Feistel type ciphers like DES.

We will now describe a simple and general way to construct a secure block cipher (BC) mode starting from any secure stream cipher in the SCM mode.

The construction essentially requires only three rounds and works for variable block sizes that are practically limited only by the memory space available. We are interested in the SCM mode whose initial state depends on the secret key only, without using any randomizing key to satisfy the one-time-pad assumption. Such a mode is not secure with respect to the so-called trivial attacks, as it is possible to generate, with high probability of success, new plaintext/ciphertext string pairs by modifying a number of end bits in the plaintext/ciphertext string pairs already known. So, what is essentially needed is a construction that would prevent such attacks.

We propose a product connection of three stream ciphers in the same SCM mode whose secret keys are the same, independent, or different (but related). Product ciphers with independent keys are usually called cascade ciphers (see [18]). In fact, we suggest different keys related in a very simple way, which removes the need for a special key schedule. For example, the keys for the second and the third cipher can be obtained as cyclic shifts of the key for the first stream cipher (represented as a binary string). The main point is to use the output bits from the first/second stream cipher in the reverse order to define the input bits for the second/third stream cipher, respectively. This creates the required forward propagation effect for the second half of the plaintext bits, as noted in [11] in a different context of block ciphers based on specific finite automata. It is required to memorize the outputs of the first and the second stream cipher. One can also use a product connection of only two stream ciphers, but in this case the change of the last plaintext bit necessarily gives rise to the change of the first ciphertext bit, which may be considered as a weakness for some applications.

For the encryption of long messages, the proposed BC mode can be used in its basic, ECB form with a large block size. Also, it can be used in the usual CBC and CFB modes, with a finite input memory of the decryption transform. This may be suitable for some applications, e.g., for the encryption of random access files. In addition, one may also use the existing internal memory of the underlying stream cipher. For example, one can use the last generated internal state for the current plaintext block as the initial state for the next one. The obtained cipher can then be considered as an enhanced version of the original stream cipher in the SCM mode, because of the triple encryption.

By using the standard argument (e.g., see [18]), we obtain the following two propositions, which essentially show that the BC mode is at least as secure as the underlying SCM mode with respect to nontrivial attacks, as trivial attacks on the SCM mode are prevented by reversing the intermediate ciphertexts. It is assumed that the attacks work for any secret key.

Proposition 3. *If the underlying* SCM *mode is secure with respect to nontrivial cryptanalytic attacks, then the derived* BC *mode with the cascade connection is secure. If the underlying* SCM *mode is secure with respect to secret key reconstruction cryptanalytic attacks, so is the derived* BC *mode with the cascade connection.*

Proof. It should be proved that any computationally feasible attack on the BC mode with the cascade connection can be converted into a nontrivial attack on the SCM mode at an additional computational cost that is not comparatively significant. Consider the BC mode where the secret key for the first stream cipher is unknown and arbitrary and where the keys for the second and the third stream cipher are known and fixed. Then any plaintext/ciphertext block pair for the BC mode can be converted into the corresponding plaintext/ciphertext string pair for the SCM mode of the first stream cipher and vice versa, at a computational cost of two SCM encryptions/decryptions (with known keys) per pair.

Accordingly, one can modify the training set for any attack on the BC mode into the corresponding training set for an attack on the SCM mode and vice versa. Also, producing an unknown plaintext/ciphertext block pair for the BC mode directly extends into producing an unknown ciphertext string for a known plaintext string for the SCM mode. This is a nontrivial attack on the SCM mode, since the reversion of intermediate ciphertexts renders the produced plaintext/ciphertext string pair(s) different from those obtained by trivial attacks (i.e., by modifying pairs from the training set).

A similar proof holds for secret key reconstruction attacks. □

Proposition 4. *If the underlying* SCM *mode is secure with respect to secret key reconstruction cryptanalytic attacks in the related key scenario, then the derived* BC *mode with the product connection is secure with respect to secret key reconstruction cryptanalytic attacks.*

Proof. It should be proved that any computationally feasible secret key reconstruction attack on the BC mode with the product connection can be converted into a secret key reconstruction attack on the underlying SCM mode at an additional computational cost that is not comparatively significant. As the keys of the second and the third SCM mode are derived from the secret key of the first SCM mode, any given plaintext/ciphertext block pair for the BC mode can be converted into the corresponding plaintext/ciphertext string pair for the first SCM mode and vice versa if two more plaintext/ciphertext string pairs are known: one for the second SCM mode and one for the third SCM mode in the product connection, both obtained from the (related) keys derived from the secret key of the first SCM mode.

Accordingly, the training set for any secret key reconstruction attack on the BC mode can be obtained from the corresponding training set for the SCM mode with the same secret key and from two additional training sets for the SCM modes with related keys. This is achieved by combining the plaintext/ciphertext string pairs from the three training sets into the corresponding plaintext/ciphertext block pairs for the BC mode, respectively. The secret key for the first SCM mode can then be produced by the secret key reconstruction attack on the corresponding BC mode. □

Note that in a special case, Proposition 3 is true for cryptanalytic attacks successful for particular plaintexts only, as the plaintexts for the first SCM mode in the cascade and for the whole cascade are the same (see [18]). If very simple stream ciphers are used so that the security of the SCM mode may be questionable, then the number of rounds can be increased.

8 Conversion of SCM Mode into Hash Function (HF) Mode

A binary hash function (HF) is defined in the same way as a KHF except that the secret key parameter is not used. One-wayness and collision-resistance (or collision-freedom) are the two main computational security properties required from a HF. A HF is called one-way if it is computationally infeasible to find any input that hashes to a given output, for almost all outputs. Ideally, the random guessing, with the computational complexity $O(2^n)$, should be the most efficient way of inverting a HF. A HF is called collision-resistant if it is computationally infeasible to find any two distinct inputs that hash to the same output. Ideally, the birthday attack, with the computational complexity $O(2^{n/2})$, should be the most efficient way of producing collisions. Note that the existing proposals for HF's are either dedicated or are based on block ciphers like DES, and are typically defined in terms of a so-called compression function which is applied iteratively (e.g., see [20] and [1]).

By fixing the value of the secret key, any KHF obtained from the SCM mode of a given stream cipher becomes a candidate HF. However, its security is no longer guaranteed by the security of the SCM mode. Namely, the one-wayness and collision-resistance properties impose stronger requirements for the SCM mode which do not involve the output function of the SCM mode at all and are hence more difficult to satisfy. For example, collision-resistance implies that it should be computationally infeasible to find any two different plaintext sequences that will, starting from the same initial state, produce the same internal state at a given time in future.

We now define a more complicated, but still simple construction of a HF from the SCM mode whose security relies on the output function as well. The basic construction consists of two stages. The first stage is similar to one for a KHF, except that the plaintext sequence for the SCM mode consists of the l message bits only and that the (secret) key is fixed and known. The SCM is clocked l times and the corresponding l bits of the ciphertext are memorized. In the second stage, the l ciphertext bits in the reverse order are used as the plaintext sequence for the same SCM mode, but now starting from the last internal state produced in the first stage. The SCM is clocked l times and an additional number of times several times bigger than M (as before), and the last n successive ciphertext bits produced are the hash value.

As in the BC mode, the ciphertext bits are used in the reverse order to increase the forward propagation effect for the second half of the message bits. Since finding the collisions necessarily involves the output function of the SCM mode, the constructed HF seems to be, at least heuristically, at least as secure as the underlying SCM mode with respect to nontrivial cryptanalytic attacks. Clearly, the number of stages can be made bigger than two by proceeding in a similar way. This would increase the security.

If the underlying SCM mode is secure, then the resulting KHF and HF modes also satisfy a stronger security property that any change of the message bits gives rise to a random looking change of the corresponding hash value.

As the basic construction described above requires memorizing intermediate ciphertext(s) of the same length as the message itself, a derived construction with lesser memory requirements would be to use the basic construction to define the compression function which is applied iteratively in the usual way. Namely, let the message be divided into blocks of a given, relatively large length l with the last block of variable length as required (without any padding). Let, for simplicity, the memory size of the SCM mode utilized be equal to the hash value length n. Then each round of the iterative construction is the basic construction applied to a new message block and with the hash value from the previous iteration as the initial internal state.

9 Proposal

The underlying keystream generator to be used in the proposed constructions can be as simple as a self-clock-controlled nonlinear filter generator, where irregular clocking is needed to ensure that the next-state function is nonlinear. The nonlinear filter generator, when regularly clocked, should be designed so as to resist known initial state reconstruction attacks including the fast correlation attack, the conditional correlation attack, and the inversion attack as well as to achieve (with a high probability) a long period, a high linear complexity, and good statistical properties of the output sequence (see [8]). We propose that the LFSR length be at least 256 and that the LFSR initial state be defined by the secret key at least 128 bits long. As the next-state function is not one-to-one due to irregular clocking, one may expect a reduced period of the keystream sequence, but not less than about 2^{128}, which is long enough even for stream cipher applications.

The binary clock-control output is produced by an additional boolean function with a few inputs (e.g., three) taken from the LFSR taps chosen according to a full positive difference set. The difference sets used for clock control and for the filter function should be disjoint, as in a nonlinear filter generator with two binary outputs (see [8]). According to the binary clock-control output, the LFSR is clocked once or twice per each output bit. The SCM mode is formed by adding the plaintext bit to the feedback bit at each time, with the plaintext bit repeated if the LFSR is clocked twice.

10 Conclusions

A general stream cipher with memory (SCM) mode, which is typically overlooked in the open literature, is pointed out. Its main characteristic is that each ciphertext symbol depends on both the current and previous plaintext symbols. Similarly, in decryption, each plaintext symbol depends on the current and previous ciphertext symbols. It is shown how to convert any keystream generator (KG) mode into the SCM mode and their practical security is discussed. Investigating the practical security of the SCM mode of stream ciphers is a new interesting research area in public cryptology. Developing attacks on the SCM mode would reveal weaknesses of the underlying KG mode, especially when used with resynchronization.

It is proposed how to obtain a secure self-synchronizing stream cipher from any secure stream cipher in the SCM mode. It is then proposed how to construct secure keyed hash functions, block ciphers, and hash functions from any secure stream cipher in the SCM mode. In all the modes, the message length can be made large and variable in a simple and natural way.

The resulting designs are rather new and unusual and are based on the iterative structure of stream ciphers which is symbol rather than block based. The way the secret key is incorporated is new too. For example, there is no need for specially designed S-boxes or a special key schedule algorithm. In particular, the underlying stream cipher can be as simple as a single self-clock-controlled nonlinear filter generator, with the secret key controlling its initial state only.

All the constructions directly extend from the binary to an arbitrary plaintext/ciphertext alphabet, e.g., to stream ciphers based on multiple rather than individual shift registers which are suitable for software realizations.

References

1. R. J. Anderson and E. Biham, "Two practical and provably secure block ciphers: BEAR and LION," Fast Software Encryption – Cambridge '96, *Lecture Notes in Computer Science*, vol. 1039, D. Gollmann ed., Springer-Verlag, pp. 113-120, 1996.
2. M. Bellare, J. Kilian, and P. Rogaway, "The security of cipher block chaining," Advances in Cryptology – CRYPTO '94, *Lecture Notes in Computer Science*, vol. 839, Y. G. Desmedt ed., Springer-Verlag, pp. 341-358, 1994.
3. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption: analysis of the DES modes of operation," *Proc. of the 38. Annual Symposium on the Foundations of Computer Science – FOCS* '97, IEEE Press, 1997.
4. E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, vol. 4(1), pp. 3-72, 1991.
5. E. Biham and P. C. Kocher, "A known plaintext attack on the PKZIP stream cipher," Fast Software Encryption – Leuven '94, *Lecture Notes in Computer Science*, vol. 1008, B. Preneel ed., Springer-Verlag, pp. 144-153, 1995.
6. J. Daemen, R. Govaerts, and J. Vandewalle, "Resynchronization weakness in synchronous stream ciphers," Advances in Cryptology – EUROCRYPT '93, *Lecture Notes in Computer Science*, vol. 765, T. Helleseth ed., Springer-Verlag, pp. 159-167, 1994.
7. J. Dj. Golić, "On the security of shift register based keystream generators," Fast Software Encryption – Cambridge '93, *Lecture Notes of Computer Science*, vol. 809, R. J. Anderson ed., Springer-Verlag, pp. 90-100, 1994.
8. J. Dj. Golić, "On the security of nonlinear filter generators," Fast Software Encryption – Cambridge '96, *Lecture Notes in Computer Science*, vol. 1039, D. Gollmann ed., Springer-Verlag, pp. 173-188, 1996.
9. J. Dj. Golić, "Linear models for keystream generators," *IEEE Trans. Computers*, vol. C-45, pp. 41-49, Jan. 1996.
10. J. Dj. Golić, "Recent advances in stream cipher cryptanalysis," *Publications de l'Institut Mathematique*, vol. 64/78, pp. 183-204, 1998.

11. M. Gysin, "A one-key cryptosystem based on a finite nonlinear automaton," Cryptography: Policy and Algorithms – Brisbane '95, *Lecture Notes in Computer Science*, vol. 1029, E. Dawson and J. Golić eds., Springer-Verlag, pp. 165-173, 1996.
12. T. Johansson, "A shift register construction of unconditionally secure authentication codes," *Designs, Codes and Cryptography*, vol. 4, pp. 69-81, 1994.
13. A. Klapper and M. Goresky, "Cryptanalysis based on 2-adic rational approximation," Advances in Cryptology – CRYPTO '95, *Lecture Notes in Computer Science*, vol. 963, D. Coppersmith ed., Springer-Verlag, pp. 262-273, 1995.
14. H. Krawczyk, "LFSR-based hashing and authentication," Advances in Cryptology – CRYPTO '94, *Lecture Notes in Computer Science*, vol. 839, Y. G. Desmedt ed., Springer-Verlag, pp. 129-139, 1994.
15. X. Lai and R. A. Rueppel, "A fast cryptographic checksum algorithm based on stream ciphers," Advances in Cryptology – AUSCRYPT '92, *Lecture Notes in Computer Science*, vol. 718, J. Seberry and Y. Zheng eds., Springer-Verlag, pp. 339-348, 1993.
16. S. Lucks, "Faster Luby-Rackoff ciphers," Fast Software Encryption – Cambridge '96, *Lecture Notes in Computer Science*, vol. 1039, D. Gollmann ed., Springer-Verlag, pp. 189-203, 1996.
17. M. Matsui, "Linear cryptanalysis method for DES cipher," Advances in Cryptology – EUROCRYPT '93, *Lecture Notes in Computer Science*, vol. 765, T. Helleseth ed., Springer-Verlag, pp. 386-397, 1994.
18. U. M. Maurer and J. L. Massey, "Cascade ciphers: the importance of being first," *Journal of Cryptology*, vol. 6(1), pp. 55-61, 1993.
19. A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.
20. B. Preneel, R. Govaerts, and J. Vandewalle, "Hash functions based on block ciphers: a synthetic approach," Advances in Cryptology – CRYPTO '93, *Lecture Notes in Computer Science*, vol. 773, D. G. Stinson ed., Springer-Verlag, pp. 368-378, 1994.
21. R. A. Rueppel, "Stream ciphers," *Contemporary Cryptology: The Science of Information Integrity*, G. Simmons ed., pp. 65-134. New York: IEEE Press, 1991.
22. B. Schneier, *Applied Cryptography*. New York: Wiley, 1996.
23. R. Taylor, "An integrity check value algorithm for stream ciphers," Advances in Cryptology – CRYPTO '93, *Lecture Notes in Computer Science*, vol. 773, D. G. Stinson ed., Springer-Verlag, pp. 40-48, 1994.