Self-synchronizing stream ciphers and dynamical systems: state of the art and open issues

Gilles Millérioux, Philippe Guillot

SELF-SYNCHRONIZING STREAM CIPHERS AND DYNAMICAL SYSTEMS: STATE OF THE ART AND OPEN ISSUES

G. MILLERIOUX
Centre de Recherche en Automatique de Nancy (CRAN UMR 7039), Nancy University, CNRS
E-mail: gilles.millerioux@esstin.uhp-nancy.fr
P. GUILLOT
Laboratoire Analyse, Géométrie et Applications (LAGA UMR 7539) University of Paris 8,
philippe.guillot@univ-paris8.fr

November 29, 2010

Dynamical systems play a central role in the design of symmetric cryptosystems. Their use has been widely investigated both in "chaos-based" private communications and in stream ciphers over finite fields. In the former case, they get the form of automata named as Moore or Mealy machines. The main charateristic of stream ciphers lies in that they require synchronization of complex sequences generated by the dynamical systems involved at the transmitter and the receiver part. In this paper, we focus on a special class of symmetric ciphers, namely the Self-Synchronizing Stream Ciphers. Indeed, such ciphers have not been seriously explored so far although they get interesting properties of synchronization which could make them very appealing in practice. We review and compare different design approaches which have been proposed in the open literature and fully-specified algorithms are detailed for illustration purpose. Open issues related to the validation and the implementation of Self-Synchronizing Stream Ciphers are developped. We highlight the reason why some concepts borrowed from control theory appear to be useful to this end.

Keywords: dynamical systems, stream ciphers, control theory

1. Introduction

Nowadays, a huge amount of information exchange is carried out through public networks. This being the case, guaranteeing privacy in the communication is undoubtedly a great challenge. It turns out that cryptography is central in this context. Indeed, it is highly based on the notions of confusion and diffusion introduced by Shannon. Although these notions are intuitively understandable, they are actually explicited through a precise formalism (see for example [Massey, J.L., 1992]). Among a wide variety of cryptographic techniques, two major classes can be typically distinguished: *public-key* ciphers (or asymmetric-key ciphers) and *secret-key* ciphers (also called symmetric-key ciphers). Public-key ciphers are largely based upon computationally very demanding mathematical

problems, for instance, integer factorization into primes. Two milestones are 1976 with the seminal paper of Diffie and Hellmann [1976] that founded the public key cryptography and 1978, marked by the publication of RSA, the first full-fledged public-key algorithm. This discovery was important notably because it solved the key-exchange problem of symmetric cryptography. Modern symmetric cryptography originates in the works of Feistel at IBM during the late 1960s and early 1970s. One of the key dates is 1977, when the symmetric cipher Data Encryption Standard (DES) was adopted by the U.S. National Bureau of Standards (now the National Institute of Standards and Technology —NIST), for encrypting unclassified information. DES is now in the process of being replaced by the Advanced Encryption Standard (AES), a new standard adopted by NIST in 2001.

Among symmetric-key ciphers, stream ciphers are of special interest for high speed encryption in satellite communications, private TV channels broadcasting, RFID, networked embedded systems. They are mainly based on generators, precisely in the form of dynamical systems, delivering complex sequences which must be synchronized at the transmitter and receiver sides. Stream ciphers have received increasing attention quite recently. Two European projects have influenced on this evolution : the project NESSIE within the Information Society Technologies Programme of the European Commission which had started in 2000 and ended in 2004 followed by ECRYPT[1] launched on February 1st, 2004. Sponsored by ECRYPT, eSTREAM is a multi-year effort aiming at identifying promising both software and hardware oriented symmetric cryptosystems with proposals from industry to academia.

As it turns out, there is a connection between the properties of confusion and diffusion and the random-looking chaotic dynamics, or more generally complex dynamics. This is the main reason why in 1993 entered the scene "chaos-based privacy". Indeed, chaotic behavior is one of the most complex dynamics a nonlinear system can exhibit. A formal definition of chaos is due to R.L. Devaney [1989] and the sensitivity to initial conditions is one of the most central property characterizing chaos. It can roughly be described into the fact that a small change in the initial conditions can drastically change the long-term behavior of a system. Chaos has received considerable attention for some years. Actually, the terminology "chaos" has been really introduced for the first time in the seminal paper of Li and Yorke "Period Three Implies Chaos" [1975]. Complex dynamics had its beginnings in the work of the french mathematician Henri Poincaré (1854-1912). Sensitive dependent phenomena have been highlighted by Edward Lorenz in 1963 while simulating a simplified model of convection. Since the 90's, a huge number of applications have been proposed over the fields of circuits and systems, mechanics, physics, avionics, weather forecasting. Because the signals resulting from chaotic systems are broadband, noiselike and present random-like statistical properties, albeit they are generated by deterministic systems, they are difficult to predict. All this motivated the use of such dynamical systems for privacy issues. The year 1990 is a milestone with a pioneering work reported in [Carroll, T. L & Pecora, L. M., 1991]. Since then, many schemes have been proposed to scramble information with a complex sequence delivered by dynamical systems leading thereby to cryptosystems which mimic symmetric ciphers (see [Ogorzalek, M. J., 1993][Hasler, M., 1998] [Yang, T., 2004][Alvarez, G. & Li, S., 2006] [Millérioux G. *et al.*, 2008b] for some surveys). To highlight the fancy for this topic, let us stress that many special issues in international journals have been published, numerous invited sessions in conferences have been organized as well. But actually, it turns out that the chaos-based algorithms proposed so far belong more to the field of steganography than to pure cryptography.

After this brief recall on the important events which have marked the field of cryptography and the role played by the dynamical systems in this context, the following remark can be made. Less attention has been paid on a special class of symmetric ciphers, namely the self-synchronizing stream ciphers. Indeed, when looking at the open literature and some substantial courses on symmetric cryptography proposed by, to mention a few,

[1] website available at `http://www.ecrypt.eu.org/stream/`

M. Kiviharju[2], S. Paul[3], the web site PICSI[4] on Cryptology and Information security, these ciphers are just touched on but not really investigated (see [Daemen, J. *et al.*, 1992] for an exception). Even more is true, it is worth pointing out that within the eSTREAM project, a project of reference in the field of stream ciphers-based cryptography, over 34 stream cipher primitives which had been submitted for evaluation, only two of them belonged to the class of self-synchronizing stream ciphers. Nevertheless, they turn out to be very useful in secure communications. Indeed, as it will be detailed later in this paper, self-synchronizing stream ciphers offer serious advantages, the main one being the ability to automatically achieve synchronization between the two parts of a communication setup. As a result, there is no need to call for resynchronization protocols or synchronization flags. It is of first importance when one must be face for example drastic constraints concerning the throughput.

The main objective of this paper is to review, compare and discuss the different design approaches devoted to Self-Synchronizing Stream Ciphers while showing how dynamical systems are used to confer them the interesting property of self-synchronization. The layout is the following.

After a recall of basic background on cryptography with a special emphasis on symmetric cryptography (Section 2), the different approaches which have been proposed in the open literature are detailed and illustrated through interesting fully-specified algorithms (Section 3). Thus, we suggest some open issues related to the design, the validation and the implementation of self-synchronizing stream ciphers (Section 4).

2. Background on symmetric cryptography

2.1. Generalities

A general encryption mechanism, also called cryptosystem or cipher, is illustrated in Fig. 1. We are given an alphabet A, that is, a finite set of basic elements named symbols. On the *transmitter* part,

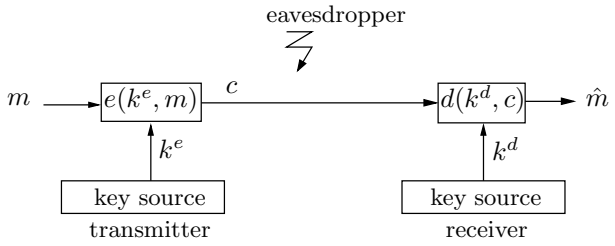

Fig. 1: General encryption mechanism

a plaintext (also called information or message) $m \in \mathcal{M}$ ($\mathcal{M}$ is called the message space) consisting of a string of symbols $m_k \in A$ is encrypted according to an encryption function e which depends on the key $k^e \in \mathcal{K}$ ($\mathcal{K}$ is called the key space). The resulting ciphertext $c \in \mathcal{C}$ ($\mathcal{C}$ is called the ciphertext space), a string of symbols c_k from an alphabet B usually (and assumed hereafter) identical to A, is conveyed through a public channel to the *receiver*. At the receiver side, the ciphertext c is decrypted according to a decryption function d which depends on the key $k^d \in \mathcal{K}$. For a prescribed k^e, the function e must be invertible. In symmetric encryption, the pair (e, d) is such that the key k^d can be easily recovered from k^e. Hence, not only k^d must be kept secret but the key k^e as well. It is customary that both keys are identical, that is $k^d = k^e$. Another property of a symmetric encryption scheme is that there must exist a unique pair (k^e, k^d) such that $d(k^d, c) = m$ where $c = e(k^e, m)$.
There are two classes of symmetric-key encryption schemes which are commonly distinguished: block ciphers and stream ciphers. A block cipher is an encryption scheme that breaks up the plaintext messages into strings (called blocks) of a fixed length over an alphabet and encrypts one block at a time. Block ciphers usually involve compositions of substitution and transposition operations. Next we describe stream ciphers in more detail.

2.2. Stream ciphers

In the case of stream ciphers, the encryption function e can change for each symbol because it depends on a time-varying key z_k also called *running key*. The sequence $\{z_k\}$ is called the *keystream*. This being the case, stream ciphers are generally well appropriate and their use can even be compulsory when buffering is limited or when only one symbol can be processed at a time: the field of

[2]available at `www.tcs.hut.fi/Studies/T-79.514/`
[3]available at `http://homes.esat.kuleuven.be/psourady/stream_cipher_course-I.htm`
[4]available at `www.picsi.org`

telecommunications often include such constraints. They benefit from smaller footprint (gates, power consumption, ...) in low-end hardware implementation, high encryption speed, small input/output delay and simple protocols for handling variable sized inputs. They are efficient and compact in constrained devices.
Stream ciphers require a keystream generator. It is usual that the plaintext m_k and the ciphertext c_k are binary words. If so, the most widely adopted function e is the bitwise XOR operation and if the generator delivers a truly random keystream $\{z_k\}$ which is never used again, the encryption scheme is called *one-time pad* —the only cipher known to be unconditionally secure so far. However, in order to decrypt the ciphertext, the recipient party of a one-time pad encryption setup would have to know the random keystream and, thus, would require again a secure transmission of the key. Besides, for the *one-time pad* cipher, the key should be as long as the plaintext and would drastically increase the difficulty of the key distribution. As an alternative to such an ideal encryption scheme, one can resort to pseudo-random generators. Indeed, for such generators, the keystream is produced by a deterministic function (often involving feedback shift registers along with nonlinearities [Knuth, D. E., 1998]) while its statistical properties look random. There are two classes of stream ciphers, the difference lying in the way the keystream is generated: the synchronous stream ciphers and the self-synchronizing stream ciphers.
Synchronous Stream Ciphers (written hereafter SSC for short) admit the equations:

$$\begin{cases} q_k = \sigma^s(q_{k-1}) \\ z_k = s(q_k) \\ c_k = e(z_k, m_k) \end{cases} \quad (1)$$

σ^s is the next-state transition function while s acts as a filter and generates the keystream $\{z_k\}$.

Self-Synchronizing Stream Ciphers (written hereafter SSSC for short) admit the equations:

$$\begin{cases} z_k = \sigma_\theta^{ss}(c_{k-l-M}, \ldots, c_{k-l}) \\ c_k = e(z_k, m_k) \end{cases} \quad (2)$$

σ_θ^{ss} is the function that generates the keystream $\{z_k\}$. l is a nonnegative integer standing for a possible delay. σ_θ^{ss} depends on past values of c_k. The number of past values is most often bounded and equals M, the *delay of memorization*.
Actually, many chaos-based encryption schemes proposed in the literature (see for example [Millérioux G. *et al.*, 2008b] and references therein for a survey) involve observers. When the state reconstruction is ensured asymptotically, such cryptosystems can be considered as belonging to the class of SSSC with $M \to \infty$.
It may be also considered SSSC for which M is a random variable with a probability law that decreases to zero as time grows to infinity. These SSSC are called statistical SSSC but have never been investigated so far. A little bit more will be told at the end of this paper in the Subsection 4.3 devoted to open issues.
Unless otherwise stated, only the case when M is bounded will be addressed in the sequel.

Regardless the class of ciphers, synchronous or self-synchronizing, the ciphertext c_k is worked out through an encryption function e which must be invertible for any prescribed z_k. In the binary case, one has $A = B = \{0, 1\}$ and $e(z_k, m_k) = z_k \oplus m_k$ where $\oplus$ denotes the modulo 2 addition on the 2-element field. The decryption is performed through a function d depending on the ciphertext c_k and the running key $\hat{z}_k$ of the receiver's generator. Such a function must obey the rule:

$$\hat{m}_k := d(c_k, \hat{z}_k) = m_k \text{ if } \hat{z}_k = z_k \quad (3)$$

In the binary case, one has $d(\hat{z}_k, c_k) = \hat{z}_k \oplus c_k$

Synchronization issues
For stream ciphers, the generators at both sides have same generator function and synchronization of keystreams $\{z_k\}$ and $\{\hat{z}_k\}$ generated respectively at the transmitter and receiver sides is a condition for proper decryption.
For SSC, the generators are not coupled each other. Consequently, the only way to guarantee synchronization of the keystreams is to share the seed (the initial running key z_0). This being the case, the secret key θ is nothing but the seed z_0.
For SSSC, since the generator function σ_θ^{ss} shares, at the transmitter and receiver sides, the same quantities, namely the past ciphertexts, it is clear that the generators synchronize automatically after a finite transient time of length M. The secret

key is some suitable (according to the security) parameters of the function σ_θ^{ss}.

Advantages of self-synchronizing stream ciphers

As far as SSSC are concerned, the ability to self-synchronizing provides many advantages. First, if a ciphertext is deleted, inserted or flipped, the SSSC will automatically resume proper decryption after a short, finite and predictable transient time. Hence, SSSC does not require any additional synchronization flags or interactive protocols for recovering lost synchronization. Secondly, the self-synchronizing mechanism also enables the receiver to switch at any time into an ongoing enciphered transmission. Third, any modification of ciphertext symbols by an active eavesdropper causes incorrect decryption for a fixed number of next symbols. As a result, an SSSC prevents active eavesdroppers from undetectable tampering with the plaintext: message authenticity is guaranteed. Finally, since each plaintext symbol influences a fixed number of following ciphertexts, the statistical properties of the plaintext are thereby diffused through the ciphertext. Hence, SSSC are very efficient against attacks based on plaintext redundancy and the property of diffusion is structurally fulfilled.

3. State of the art in the design of SSSC

Actually, the model (2) of an SSSC is a conceptual model, called canonical representation, that can correspond to different architectures and that result from different design approaches. In the open literature, few designs methods have been proposed. They are detailed below in a way which highlights the central role played by dynamical systems and the reason why some concepts borrowed from control theory appear to be useful.

3.1. Block ciphers in CFB mode

This SSSC design approach resorts to a length M shift register and a block cipher (DES for instance) both inserted in a closed-loop architecture. It is a very special mode of operation involving block ciphers naturally called Cipher FeedBack (CFB) mode. The block cipher's input is the shift register state. Usually a limited number of the block cipher output bits are retained, the selection being performed through a so-called filter function denoted h' on the Fig. 2. Such a configuration is often used in 1-bit CFB mode. In such a case, the encryption function e is a XOR (modulo 2 addition over $\{0,1\}$). The keystream generator σ_θ^{ss} of the corresponding canonical form (2) results from the composition of three functions: the state transition function of the shift register, the block cipher and the filter function h'.

This mode is quite inefficient in terms of encryption speed since one block cipher operation, and so multiple rounds, are required for enciphering a single plaintext m_k.

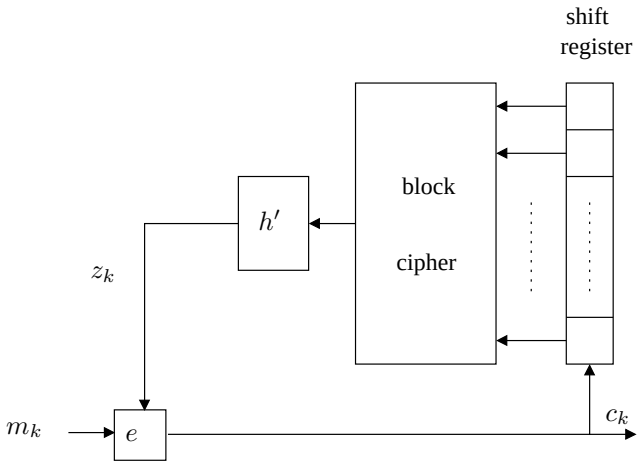

Fig. 2: Block cipher in CFB mode

3.2. Maurer's approach

In [Maurer, U. M., 1991], it is suggested an alternate design approach exclusively dedicated to SSSC. It includes two main ideas.

The first idea consists in replacing the shift register, the block cipher and the output bit filter function of the CFB mode architecture by an automaton. The automaton obeys the dynamics

$$\begin{cases} q_{k+1} &= g_\theta(q_k, c_k) \\ z_k &= h_\theta(q_k) \end{cases} \tag{4}$$

The function g_θ is called (next) state transition function while h_θ is called output function.
The automaton must have a finite input memory of size M meaning that the state q_k must be expressed

by mean of a function l_θ which depends on a finite number of past ciphertexts c_{k-i} :

$$q_k = l_\theta(c_{k-M}, \ldots, c_{k-1}) \tag{5}$$

Substituting the above expression of q_k into the second equation of (4) gives the function σ_θ^{ss} of the canonical form (2). One has the following composition : $\sigma_\theta^{ss} = h_\theta \circ l_\theta$. According to the discussion of Subsect. 2.2 on synchronization issues, self-synchronization is guaranteed.

Let us notice that the CFB mode can be rewritten into the form (4)-(5). The function l_θ is very simple since it merely reduces to a shift. The output function h_θ results from the composition of the block cipher (parametrized by its secret key θ) and the filter function h'.

In the Maurer's approach, the SSSC is based on a cryptographically secure state-transition function g_θ as well as on a cryptographically secure output function h_θ. Consequently, the resulting SSSC can be secure unless both functions are simultaneously unsecure. That differs from the CFB mode for which the security relies entirely on the security of the output function h_θ and so mostly on the block cipher function.

The second idea of the Maurer's principle consists in increasing the complexity by combining several finite automata in serial or in parallel or more generally by performing composition. As a result, many components that are relatively simple in terms of implementation complexity and memory size can be combined to form an SSSC realizing a very complicated function σ_θ^{ss} in the corresponding canonical representation (2). For a serial composition of multiple automata, the resulting memory size equals the sum of the memory size of each automaton. For a parallel composition of multiple automata, the resulting memory size equals the upper memory size. When implemented in hardware, parallelization leads to very high achievable encryption speed. An example of architecture involving four automata is depicted in Fig. 3.

Throughout the eSTREAM project, two fully specified algorithms have retained attention: SSS and Moustique. They are shortly described to illustrate how the general principle of Maurer is taken

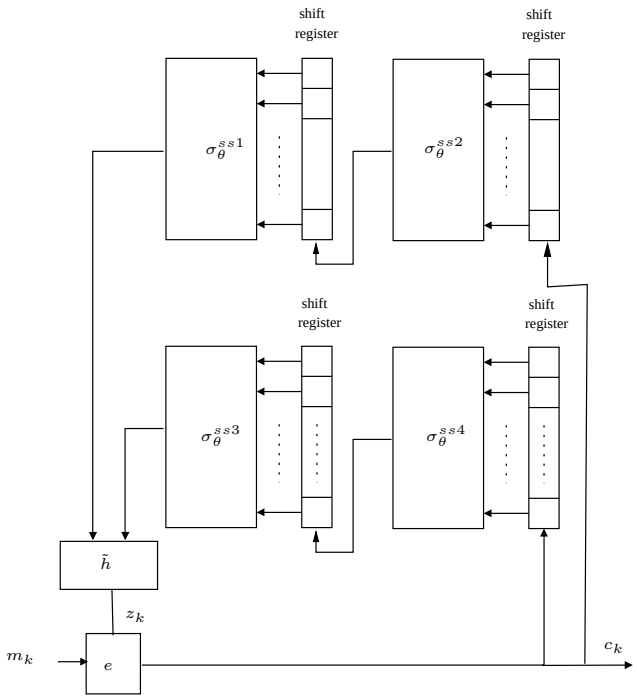

Fig. 3: Example of serial/parallel connection of four automata. The function $\tilde{h}$ combines the accessible automata outputs to deliver the keystream z_k

into account. As a matter of fact, only the first idea of Maurer consisting in resorting to an automaton with finite input memory has been adopted throughout these two examples. Indeed, as it turns out, the second idea is too general as is. These examples are also interesting in that they give us a better understanding in the way how the dynamical systems are "shaped" to guarantee the self-synchronization property.

3.2(A). SSS

SSS is a software bit oriented cryptosystem which has been proposed in [Hawkes P *et al.*, 2004]. The corresponding block diagram is depicted on Fig. 4.

The following notations are necessary to describe SSS.

- $x >>> n$ denoted the rotation of n bits to the right of the word x
- $S_\theta(x) = SBOX_\theta(x_H) \oplus x$ with x_H the most significant byte of the word x is the XOR operation between x and the result of $SBOX_\theta$ which is a combination of two S-boxes imple-

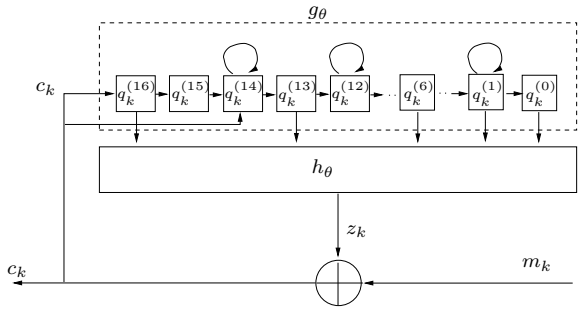

Fig. 4: Block diagram of SSS

menting nonlinear substitutions called Skipjack S-box and Q-box and parametrized by the secret key θ

The keystream generator obeys (4). The dimension of the state vector q_k equals $n = 17$ that is the number of shift registers. Each component $q_k^{(j)}$ assigned to a shift register obeys an independent dynamics g_θ^j :

$$\begin{array}{lcl} q_{k+1}^{(16)} & = & c_k \\ q_{k+1}^{(j)} & = & q_k^{(j+1)} \ (j = 0, 2, \ldots, 11, 13, 15) \\ q_{k+1}^{(14)} & = & q_k^{(15)} + S_\theta(c_k >>> 8) \\ q_{k+1}^{(12)} & = & S_\theta(q_k^{(13)}) \\ q_{k+1}^{(1)} & = & q_k^{(2)} >>> 8 \end{array} \tag{6}$$

The initial state of the shift register number 16 fulfills $q_1^{(16)} = c_0$. Furthermore, insofar as the state $q_{k+1}^{(j)}$ $(j = 1, \ldots, 16)$, at time $k+1$, depends on the state $q_k^{(j+1)}$ at time k, thus after 16 iterations, the internal state q_k will depend exclusively on the 16 past ciphertexts c_{k-i}. Hence for all $k \geq 16$ there exists a function l_θ fulfilling

$$q_k = l_\theta(c_{k-16}, \ldots, c_{k-1}) \tag{7}$$

The output function h_θ delivering the keystream z_k is defined as:

$$z_k = h_\theta(q_k) = A_\theta >>> 8 \oplus q_k^{(0)} \tag{8}$$

with $A_\theta = S_\theta\big(S_\theta(q_k^{(0)} + q_k^{(16)}) + q_k^{(1)} + q_k^{(6)} + q_k^{(13)}\big)$ Finally, combining the equations (8) and (7), the keystream generator can be equivalently rewritten in the SSSC canonical form (2):

$$\begin{array}{rl} z_k & = h_\theta(l_\theta(c_{k-16}, .., c_{k-1})) \\ & = \sigma_\theta^{ss}(c_{k-16}, .., c_{k-1}) \end{array} \tag{9}$$

and guarantees the self-synchronization property. The encryption function e and decryption function d follow the classical rules described in Subsection 2.2 where $\oplus$ is viewed in this case as a componentwise addition over the 2-element field.

3.2(B). Moustique

Another interesting SSSC, called Moustique, which follows the second idea in the Maurer's approach, has been proposed in [Daemen, J. & Kitsos, P., 2005]. It is a revisited version of two former algorithms called Mosquito and Knoth. Unlike SSS, it is an hardware bit oriented algorithm. Furthermore, although the structure still relies on the automaton (4) which must have a finite memory, a different "shape" for the state transition function g_θ is provided to guarantee self-synchronizing property. Moreover, the output function is designed through the concept of *pipelining*. Those two facts are explicited below.

For Moustique, the dimension of the state vector q_k in (4) equals $n = 96$.

As far as the state transition function g_θ is concerned, each component $q_k^{(j)}$ obeys a dynamics g_θ^j in the form:

$$q_{k+1}^{(j)} = g_\theta^j(q_k^{(j-1)}, q_k^{(j-2)}, ..., q_k^{(1)}, c_k) \ j = 1, \ldots, n \tag{10}$$

The j^{th} component of q_{k+1} does no longer depend exclusively on one component of q_k (as it is for SSS), but it depends actually on several components of q_k, especially $q_k^{(l)}$ with $l < j$. The function g_θ has a "triangular" form and ensures q_k to be independent of the initial condition q_0 after n iterations. Similarly to SSS, there exists thereby a function l_θ which enables (10) to be rewritten in a strictly equivalent way for $k \geq n$ and depends exclusively on a finite number of past ciphertexts c_{k-i}

$$q_k = l_\theta(c_{k-n}, ..., c_{k-1}) \tag{11}$$

Besides, unlike SSS, the output function is pipelined (see Fig. 5). The keystream is computed in a sequential way and the computation involves $b_s = 9$ successive stages. Each stage corresponds to a specific function s_i $(i = 0, \ldots, b_s - 1)$ depending on the result of the previous stage. For the function s_0 one has $s_0(q_k) = q_k$. The output function is made up

of a composition of b_s functions. The keystream is computed from the state functions q_k but is delivered at time $k + b_s$:

$$z_{k+b_s} = s_8(s_7(...(s_0(q_k)))) = h(q_k) \qquad (12)$$

Combining (11) and (12) gives σ_θ^{ss}

$$\begin{array}{rl} z_{k+b_s} & = h(l_\theta(c_{k-n}, ..., c_{k-1})) \\ & = \sigma_\theta^{ss}(c_{k-n}, ..., c_{k-1}) \end{array} \qquad (13)$$

It's a simple matter to see that the keystream generator can be again equivalently rewritten in the SSSC canonical form (2) and self-synchronization is guaranteed.
The pipeline is interesting in that it enables to increase the complexity of the output function while a single clock cycle is still needed to deliver the running key. Indeed the computation of each function s_i is parallelized. That induces a delay b_s between the plaintext and the corresponding ciphertext. Let us notice that none of the function s_i depend on the secret key θ. Actually, the output function h_θ in (4) should be rewritten as a non parametrized function h.
Similarly to SSS, the encryption function e and decryption function d follow the classical rules described in Subsection 2.2.

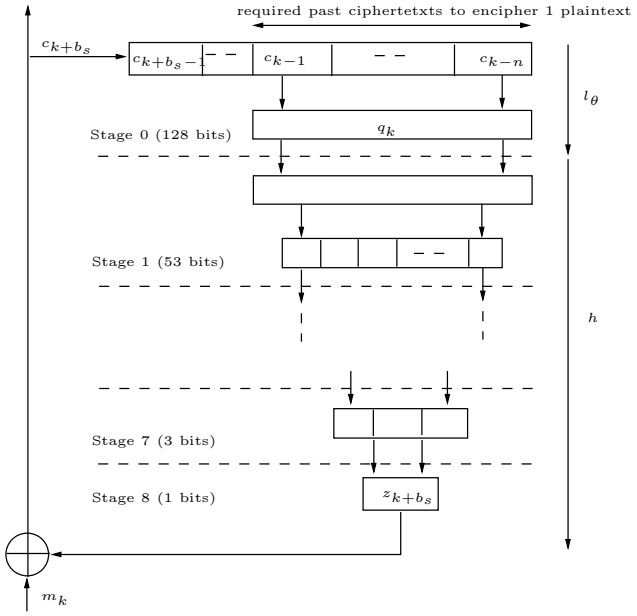

Fig. 5: Block diagram of Moustique. The functions s_i deliver a quantity of decreasing size: from 128 bits for the stage 0 to a single bit for the last stage 8

3.3. Message embedding

The message-embedded technique is derived from cryptosystems which have been proposed first for "chaos-based" private communications. It is given different names in the literature: direct chaotic modulation [Hasler, M., 1998], embedding [Lian K-Y. & Liu P., 2000] [Millérioux, G. & Daafouz, J., 2004], non autonomous modulation [Yang, T., 2004]. Different structures very similar to the message-embedding was also been proposed in [Yang, T. *et al.*, 1997] or [Parker, A. T. & Short, K. M., 2001]. Very recently, it has been provided in [Millérioux G. *et al.*, 2008b] a general framework, based on concepts borrowed from control theory, which allows to derive a self-synchronizing cryptosystem from the message-embedded structure. This Subsection aims at recalling the approach.

The equations governing a message-embedded cryptosystem are given by the dynamical system:

$$\left\{ \begin{array}{l} x_{k+1} = f_\theta(x_k, m_k) \\ y_k = h_\theta(x_k, m_k) \end{array} \right. \qquad (14)$$

Such a dynamical system is described by the 5-tuple $(A, B, X, f_\theta, h_\theta)$ where

- A is the input alphabet, which is the finite set of input symbols m_k
- $B = A$ is the output alphabet, which is the finite set of output symbols y_k
- X is the finite set of internal states x_k also called state vectors
- $f_\theta : X \times A \longrightarrow X$ is the (next) state transition function
- $h_\theta : X \times A \longrightarrow B$ is the output function.

The ciphering consists of injecting (or, as it is also usually said, embedding) the plaintext $m_k \in A$ (the input of the dynamical system) into a dynamics f_θ. The resulting system turns into a non autonomous one since the information to be encrypted acts as an exogenous input. The ciphertext $y_k \in A$ of the dynamical system is worked out through an output function h_θ of the plaintext m_k and the

internal state $x_k \in X$. θ parametrizes the dynamical and output functions and acts as the secret key.

Under special conditions, (14) can be rewritten into the form (2) and is thereby structurally equivalent to a self-synchronizing stream cipher. The correspondence is based on usual concepts of control theory. The basic background is recalled below.

We first define the so-called iterated functions associated respectively to f_θ and h_θ.

Definition 3.1. The i-order iterated next-state function, $f_\theta^{(i)} : X \times A^i \longrightarrow X$ describes the way how the internal state $x_{k+i} \in X$ of (14) at time $k+i$ depends on the state $x_k \in X$ and on the sequence of i input symbols $m_k \cdots m_{k+i-1} \in A^i$. It is defined for $i \geq 1$ and recursively obeys for $k \geq 0$,

$$\begin{cases} f_\theta^{(1)}(x_k, m_k) = f_\theta(x_k, m_k), \\ f_\theta^{(i+1)}(x_k, m_k \cdots m_{k+i}) = \\ \quad f_\theta\big(f_\theta^{(i)}(x_k, m_k \cdots m_{k+i-1}), m_{k+i}\big) \quad \text{for } i \geq 1 \end{cases}$$

Definition 3.2. The i-order iterated output function $h_\theta^{(i)} : X \times A^{i+1} \longrightarrow A$ describes the way how the output y_{k+i} of (14) at time $k+i$ depends on the state $x_k \in X$ and on the sequence of $i+1$ input symbols $m_k \cdots m_{k+i} \in A^{i+1}$. It is defined for $i \geq 0$ and recursively obeys for $k \geq 0$,

$$\begin{cases} h_\theta^{(0)}(x_k, m_k) = h_\theta(x_k, m_k), \\ h_\theta^{(i)}(x_k, m_k \ldots m_{k+i}) = \\ \quad h_\theta\big(f_\theta^{(i)}(x_k, m_k \cdots m_{k+i-1}), m_{k+i}\big) \quad \text{for } i \geq 1 \end{cases}$$

Then we recall, throughout the three following definitions, properties of dynamical systems which are central to our purpose.

Definition 3.3. The relative degree of the dynamical system (14) is the quantity denoted r with

- $r = 0$ if $\exists x_k \in X,\ \exists m_k, m'_k \in A$ with $h_\theta(x_k, m_k) \neq h_\theta(x_k, m'_k)$.

  In other words, there exists a state $x_k \in X$ and two distinct input symbols $m_k, m'_k \in A$ that lead to different values of the output,

- $r > 0$ if

  for $0 < i < r, \forall x_k \in X,\ \forall m_k \cdots m_{k+i}, m'_k \cdots m'_{k+i} \in A^{i+1}$ one has

  $$h_\theta^{(i)}(x_k, m_k \cdots m_{k+i}) = h_\theta^{(i)}(x_k, m'_k \cdots m'_{k+i})$$

  and

  for $i = r, \exists x_k \in X,\ \exists m_k \cdots m_{k+i}, m'_k \cdots m'_{k+i} \in A^{i+1}$ with

  $$h_\theta^{(r)}(x_k, m_k \cdots m_{k+r}) \neq h_\theta^{(r)}(x_k, m'_k \cdots m'_{k+r})$$

In others words, for $i < r$, the iterated output function $h_\theta^{(i)}$ only depends on x_k while for $i \geq r$, it depends both on x_k and on the sequence of $i-r+1$ input symbols $m_k \cdots m_{k+i-r}$. In particular, for $i = r$, the iterated output function depends both on m_k and on x_k, that is, there exists a state $x_k \in X$ and two distinct input symbols $m_k \in A$ and $m'_k \in A$ that lead to different values of the output, for any sequence $m_{k+1} \cdots m_{k+r}$ of input symbols.

Roughly speaking the relative degree of the dynamical system (14) is the minimum number of iterations such that the output at time $k+r$ is influenced by the input at time k.

Consequently, for $r > 0$, the r-order output function $h_\theta^{(r)}$ may be considered as a function on $X \times A$, and thus one has for $r \geq 0$:

$$y_{k+r} = h_\theta^{(r)}(x_k, m_k) \tag{15}$$

Definition 3.4. The dynamical system (14) is *left invertible* if for any internal state $x_k \in X$, the map

$$h_{x_k} : \begin{array}{ccc} A & \longrightarrow & A \\ m_k & \longmapsto & h_\theta^{(r)}(x_k, m_k) \end{array}$$

is a permutation, where $r \geq 0$ is the relative degree of (14).

The left invertibility property means that the input m_k is uniquely determined by the knowledge of the state x_k and the output y_{k+r}. The output function $h_\theta^{(r)}$ may be considered as a family of permutations on A, indexed by the set X of the internal states, or at least by a subset.

Definition 3.5. An output for (14) is said to be *flat* if all system variables of (14) can be expressed as a function of y_k and a finite number of its forward/backward iterates. In particular, there exists two functions F and G and integers $t_1 < t_2$ and $t'_1 < t'_2$ such that

$$\begin{array}{rcl} x_k & = & F(y_{k+t_1}, \cdots, y_{k+t_2}) \\ m_k & = & G(y_{k+t'_1}, \cdots, y_{k+t'_2}) \end{array} \quad (16)$$

Then, the dynamical system (14) is said to be *flat* if it admits a flat output and the *flatness characteristic number* is defined as the quantity $t_2 - t_1 + 1$.

The following Proposition is a major result.

Proposition 3.6. *If the system (14) fulfills the following assumptions $H1$ to $H3$*

- *it has a finite relative degree r ($H1$)*
- *it is left invertible ($H2$)*
- *it is flat with flat output y_k and a flatness characteristic number $t_2 - t_1 + 1$ ($H3$)*

then it is equivalent, from a structural point of view, to the transmitter part of a self-synchronizing stream cipher of the form (2) with the correspondences (explicited below for short by the symbol $\leftrightarrow$)

- *a keystream generator (also named ciphering function) $\sigma_\theta^{ss} \leftrightarrow F$*
- *a running key $z_k \leftrightarrow x_k$*
- *a ciphertext c_{k+r} corresponding to the plaintext m_k $\leftrightarrow y_{k+r}$*
- *a ciphering function $e \leftrightarrow h_\theta^{(r)}$*

This is the direct consequence of the assumptions $H1 - H3$ since, if they are fulfilled, Equations (15) and (16) hold and identification of (15) and the first equation of (16) with the respective equations (2) gives the correspondence.

Let us point out that H2 is a necessary condition for H3

Based on the previous result, the state (and so the running key) reads:

$$x_k = F(y_{k+t_1}, \cdots, y_{k+t_2}) \quad (17)$$

This expression of x_k guarantees the self-synchronizing property. A key point of the message-embedded approach lies in that the implementation of (14) is much more computationally efficient than its canonical representation (17).

Furthermore, when the relative degree r is strictly greater than zero, there is a delay r between the plaintext m_k and the corresponding ciphertext y_{k+r} and it similar to what typically happens when the output function is pipelined (see the algorithm Moustique described in Subsection 3.2(B)).

The equivalent representation of the message-embedded cryptosystem, called self-synchronizing message embedded stream cipher (14) is depicted on Fig. 6.

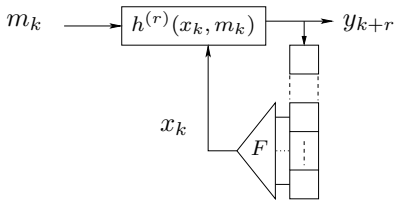

Fig. 6: Self-synchronizing Message Embedded Stream Cipher

To conclude on this Section, the properties of invertibility and flatness, borrowed from control theory, define a general and flexible framework for the design of SSSC.

4. Issues

In this Section, it is discussed some perspectives regarding the design, the validation and the implementation of SSSC.

4.1. Hybrid dynamical systems as a relevant class of ciphers

For obvious reasons, it is always aimed at providing ciphers having high speed and low hardware or software complexity. To this end, it must be thought about suitable class of dynamical systems which

benefit from an ease of design without degrading their complexity regarding the security. A general idea has been proposed in the literature from this perspective: mixing algebraic domains. For example, in [Lai, X. & Massey, J. M., 1991] the authors suggest a software/hardware oriented block cipher (called IDEA) which mixes three operations with distinct algebraic domains: xor, modulo 2^{16} addition and modulo $2^{16}+1$ multiplication. Shamir suggests in [Klimov A. & Shamir A., 2004] to use primitives built from combinations of boolean and arithmetic operations. He defines the class of so-called T-functions which contains arbitrary compositions of plus, minus, times, or, and, xor operations on n-bit words. That confers to ciphers resistance against pure algebraic or bit-oriented attacks as linear and differential attacks. In others words, it sounds relevant to introduce heterogeneity in the ciphers.

Issues: In automatic control, hybrid systems is a typical class of dynamical systems involving heterogeneity. Indeed, they involve several algebraic models called modes which are switched in time according to a logical rule. Issues regarding hybrid systems in conjunction with the special context of cryptography, and more specifically self-synchronizing stream ciphering, has never been explored yet. That constitutes a very interesting and challenging problem. In particular, a major specificity must be taken into account. In usual control theory, the variables are assumed to take values in a continuum (often $\mathbb{R}^n$ or a subset of $\mathbb{R}^n$) since they are related to physical quantities. In the cryptographic context, variables take values in finite cardinality sets (e.g. finite fields like $\mathbb{Z}/p\mathbb{Z}$ of $\mathbb{F}_{2^n}$). As a result, many control-theoretical concepts involved especially in the message embedding approach must be definitively revisited. A first study has been addressed in [Millérioux, G. *et al.*, 2008a] with a special class of hybrid systems namely the piecewise linear systems.

Others dynamical systems, in particular chaotic ones, with the interesting properties of confusion and diffusion, are defined with polynomial or rational next-state functions in (14). It would be interesting to investigate properties of the resulting dynamics after having transposed the equations over finite fields. Previous substantial works ([Fridrich, J., 1998][Schmitz, R., 2001] or [Szczepanski, J. *et al.*, 2005] [Kocarev, L. *et al.*, 2006]) have been already been conducted in the same spirit to design permutations through discretization of chaotic maps. These studies could be considered as a good guideline in the context under consideration here.

4.2. Cryptanalysis

An essential issue for the validation of cryptosystems is the cryptanalysis, that is the study of attacks against cryptographic schemes in order to reveal their possible weakness. The consideration in the design of the possible attacks and their complexity dictates the way how the secret key involved in (14) must be defined. Let us review some of cryptanalysis approaches which appear to be relevant in the context of the message-embedded approach and let us highlight the corresponding issues to be addressed.

1. Algebraic attacks

It is worthwhile pointing out that the design of a cryptographic scheme must take into account that the sets A, B, $\mathcal{K}$ and the pair (e, d) are known. Only the pair (k^e, k^d) can be assumed to be secret in symmetric-key cryptography. This is a fundamental premise in cryptanalysis, first stated by A. Kerkhoff in 1883. Based on this principle, the algebraic attack has been suggested by Shannon and has recently been widely studied with some success on certain classes of synchronous stream ciphers. Its principle relies on the algebraic model of the cipher. The objective of an algebraic attack is to find out a set of algebraic equations which can be solved efficiently. An efficient algebraic attack is a one for which the complexity is below the complexity of an exhaustive search. One of the main tool for that purpose is the elimination technique in particular based on the use of Grobner basis. In general, the eavesdropper is assumed to control the input of the cipher or the decipher and is able to collect and to analyze plaintext/ciphertext pairs to generate the equations and perform a so-called chosen plaintext or ciphertext algebraic attack. The security with respect to algebraic attack is directly related to the complexity of the parameters (secret key) recovering task.

Issues: The parameters recovering task in automatic control is nothing but identification. This being the case, the security is related to the complexity of the identification procedure required for retrieving the secret parameter θ of the dynamical system (14) which are expected to act as the secret key. An identification procedure has been provided for switched linear self-synchronizing primitives in [Vo Tan, P. *et al.*, 2010]. As it turns out, neither bit-oriented algebraic attacks, nor classical identification procedure apply if heterogeneity in the form of more general hybrid systems is thought. This issue deserves thereby new approaches and tools.

2. Others attacks

Ciphering function reconstruction
The core of an SSSC is the ciphering function F. Its complexity can be assessed through the "distance" from a given function having low algebraic degree. If the "distance" is not large enough, then there exists decoding algorithms that are able to reconstruct the whole low degree approximation of F and provide thereby an estimation of the plaintext.

Another way to reconstruct the ciphering function is to call for statistical learning with artificial neural networks as example of efficient tools.

These approaches deserve deeper investigation for heterogeneous ciphering functions.

Distinguisher
It can be proved that an SSSC is secure as long as the ciphering function F behaves like a random function. Indeed, in this case, the cryptanalyst has no information at all on the keystream $\{z_k\}$. As a result, a sufficient condition for an SSSC to be secure is that the adversary cannot distinguish the ciphering function from a random one. The existence of a distinguisher is a weakness in the ciphering function.

Checking for efficient distinguishers of heterogeneous ciphering functions remains a challenging issue.

Linear attack
The linear attack is a known plaintext attack that belongs to the family of statistical attacks. It has been first published by Matsui [1993] for cryptanalyzing the DES. A variant of this attack may be applied to SSSC. This attack recovers the secret key θ. It is also based on a linear approximations of the ciphering function F. For a prescribed linear approximation, several pairs of input/output data of F are lumped together. They are accessible when a known plaintext attack is performed. The number of required known plaintext depends on the quality of the linear approximation. This process is repeated with several other linear approximations. Then a simple linear algebra algorithm, eventually together with a remaining exhaustive search, retrieves the key θ.

This attack may be extended to non linear approximations by dint of increasing the complexity of the key recovery. Assessing the complexity and the efficiency of such an attack for hybrid systems would be of great interest.

Side channel attacks
If the secret key is embedded in a device such as a smart card or an electronic component, an adversary who has temporarily access to the device may try to recover the secret key through physical measures such as time, power consumption, glitch and so on. These attacks is a modern topic of great interest at the moment. Cryptographic algorithms must be implemented with great care, either on hardware or software target, to resist these attacks.

4.3. Statistical Self-Synchronizing Stream Ciphers

The actual synchronization delay of self-synchronizing stream ciphers is the number M of symbols required for the receiver to recover the same internal state as the transmitter (See Eq. (2)). The canonical representation of SSSC assumes that the synchronization delay is bounded. This assumption limits the complexity of the ciphering function, as in this case, it may be represented as a memoryless function. This requirement is not mandatory in practice, and it is acceptable that the synchronization delay is not a constant value, but may be a random variable with a probability law that decreases

to zero as time grows to infinity. Regarding cryptographic applications, it may be expected to bring in more complex dynamic. The way how to introduce randomness in the synchronization is a challenging issue. A solution has been suggested in [Burda, K., 2007] but deeper investigation and new alternative methods are really lacking. The tool to control the probability law of the synchronization delay remains to be developed. It may be based on spectral analysis, through discrete Fourier transform of the next-state iterated function.

5. Conclusion

This paper aimed at surveying a special application of dynamical systems in the context of cryptography. We hope that this paper has highlighted the interest of self-synchronizing stream ciphers, a class not really addressed so far, and has opened a new field of investigation. The list of questions to be addressed is undoubtedly not exhaustive but we hope that it will help any designer who would intend to provide new SSSC really competitive.

References

Alvarez, G. & Li, S. [2006] "Some basic cryptographic requirements for chaos-based cryptosystems," *Int. J. of Bifurcations and Chaos* **16**, 2129–2151.

Burda, K. [2007] "Resynchronization interval of self-synchronizing modes of block ciphers," *Int. J. of Computer and Network Security* **7**, 8–13.

Carroll, T. L. & Pecora, L. M. [1991] "Synchronizing chaotic circuits," *IEEE Trans. Circuits and Systems* **38**, 453–456.

Daemen, J., Govaerts, R. & Vandewalle, J. [1992] "On the design of high speed self-synchronizing stream ciphers," *Proc. of the ICCS/ISITA'92 conference* **1**, 279–283.

Daemen, J. & Kitsos, P. [2005] "The self-synchronizing stream cipher moustique," *eSTREAM, ECRYPT Stream Cipher Project* , Available online at http://www.ecrypt.eu.org/stream.

Devaney, R. L. [1989] *An introduction to Chaotic Dynamical Systems* (Addison-Wesley, Redwood City, CA).

Diffie, W. & Hellman, M. [1976] "New directions in cryptography," *IEEE Trans. on Information Theory* **22**, 644–654.

Fliess, M., Levine, J., Martin, P. & Rouchon P. [1995] "Flatness and defect of non-linear systems: introductory theory and examples," *Int. Jour. of Control* **61**, 1327–1361.

Fridrich, J. [1998] "Symmetric ciphers based on two-dimensional chaotic maps," *International Journal of Bifurcation and Chaos* **8**, 1259–1284.

Guillot P. & Mesnager S. [2005] "Nonlinearity and security of self-synchronizing stream ciphers," Proc. of the 2005 International Symposium on Nonlinear Theory and its Applications (NOLTA 2005), Bruges, Belgium, October.

Hasler M. [1998] "Synchronization of chaotic systems and transmission of information," *International Journal of Bifurcation and Chaos* **8**, 647–659.

Hawkes P. , Paddon, M., Rose G. G. & Miriam W. V [2004] "Primitive specification for sss, Technical report," *e-Stream Project* , Available at: http://www.ecrypt.eu.org/stream/ciphers/sss/sss.pdf.

Isidori, A. [1995] *Nonlinear control systems* (Communications and control engineering series, Springer).

Klimov A. & Shamir A. [2004] *Fast Software Encryption, Chapter 1, New cryptographic primitives based on multiword T-functions* (Springer Berlin / Heidelberg).

Knuth, D. E. [1998] *The Art of Computer Programming, Vol. 2* (Addison-Wesley, Reading, MA).

Kocarev, L., Szczepanski, J., Amigo, J. M & Tomosvski, I. [2006] "Discrete chaos: part i," *IEEE Trans. on Circuits and Systems I* **53**, 1300–1309.

Lai, X. & Massey, J. M. [1991] "A proposal for a new block encryption standard," Lectures Notes

in Computer Science 473, Advances in Cryptology (EUROCRYPT'90), Aarhus, Denmark, Springer-Verlag, May.

Li T-Y. & Yorke J. A. [1975] "Period three implies chaos," *Amer. Math. Monthly* **82**, 985–992.

Lian K-Y. & Liu P. [2000] "Synchronization with message embedded for generalized lorenz chaotic circuits and its error analysis," *IEEE Trans. Circuits. Syst. I: Fundamental Theo. Appl* **47**, 1418–1424.

Massey, J.L. [1992] *Contemporary cryptology: an introduction* (G.J. Simmons, New York, ieee press edition).

Matsui, M. [1993] "Linear cryptanalysis method for des cipher," Advances in Cryptology - EUROCRYPT'93, Lofthus, Norway, May.

Maurer, U. M. [1991] "New approaches to the design of self-synchronizing stream cipher," Lecture Notes in Computer Science, Advances in Cryptography (EUROCRYPT'91), Brighton, UK, April.

Menezes, A. J., Oorschot P. C. & Vanstone, S. A. [1996] *Handbook of Applied Cryptography* (CRC Press).

Millérioux, G., Guillot P., Amigó, J. M. & Daafouz, J. [2008] "Flat dynamical systems and self-synchronizing stream ciphers," In Proc. of the Fourth Workshop on Boolean Functions : Cryptography and Applications (BFCA'08), Copenhagen, Denmark, May.

Millérioux G., Amigo J. M. & Daafouz J. [2008] "A connection between chaotic and conventional cryptography," *IEEE Trans. on Circuits and Systems I: Regular Papers* **55**, 1695–1703.

Millérioux, G. and Daafouz, J. [2004] "Unknown input observers for message-embedded chaos synchronization of discrete-time systems," *International Journal of Bifurcation and Chaos* **14**, 1357–1368.

National Bureau of Standards [1980] *Des mode of operations, Technical report, Fed. Inform. Proc. Standards Publication, 81, Nat. Inform. Service* (Springfield, VA).

Ogorzalek, M. J. [1993] "Taming chaos - part I: synchronization," *IEEE Trans. Circuits. Syst. I: Fundamental Theo. Appl* **40**, 693–699.

Parker, A. T. & Short, K. M. [2001] "Reconstructing the keystream from a chaotic encryption scheme," *IEEE Trans. on Circ. and Syst.* **48**, 624–630.

Schmitz, R. [2001] "Use of chaotic dynamical systems in cryptography," *Journal of the Franklin Institute* **338**, 429–441.

Sira-Ramirez, H. & Agrawal, S. K. [2004] *Differentially Flat Systems* (Marcel Dekker, New York).

Szczepanski, J., Amigó, J.M., Michalek, T. & Kocarev L. [2005] "Crytographically secure substitutions based on the approximation of mixing maps," *IEEE Trans. Circuits and Systems I : Regular Papers* **52**, 443–453.

Vo Tan, P. Millérioux, G. and Daafouz, J. [2010] "Left invertibility, flatness and identifiability of switched linear dynamical systems: a framework for cryptographic applications," *International Journal of Control* **1**, 145–153.

Yang, T. [2004] "A survey of chaotic secure communication systems," *Int. J. of Computational Cognition* , (available at `http://www.YangSky.com/yangijcc.htm`).

Yang, T., Wu, C. W. & Chua, L. O. [1997] "Cryptography based on chaotic systems," *IEEE Trans. Circuits. Syst. I: Fundamental Theo. Appl* **44**, 469–472.