Flatness and structural analysis as a constructive framework for private communication

Brandon Dravie, Philippe Guillot, Gilles Millérioux

Flatness and structural analysis as a constructive framework for private communication

B. Dravie[a], P. Guillot[b], G. Millérioux[a]

[a] *Université de Lorraine, CRAN, UMR 7039, France,*
CNRS, CRAN, UMR 7039, France
(e-mail: {brandon.dravie,gilles.millerioux}@univ-lorraine.fr).
[b] *Université Paris 8, LAGA, France,*
(e-mail: philippe.guillot@univ-paris8.fr)

Abstract

This paper deals with the use of control theoretical concepts, essentially flatness, along with structural analysis, in the context of private communication. A new and systematic methodology to design a cryptographic architecture belonging to the special class of ciphers called Self Synchronizing Stream Ciphers (SSSC) is proposed. Till now, the constructions of SSSC were based on automata with finite input memory involving shifts or triangular functions (T–functions) as state transition functions. Besides, only ad-hoc approaches were available in the literature. The contribution of this paper is to propose a general framework to design SSSC involving dynamical systems taking the form of switched linear systems over finite fields with arbitrary state transition functions. An example of cipher with complete specifications is given. Its implementation on a small scale breeding of an ICS-SCADA equipment is described.

Keywords: flatness, LPV and switched linear systems, structural analysis, ciphers

1. Introduction

Synchronization has been an important topic in automatic control for years. Roughly speaking, by synchronization, it is meant correlated (according to given criteria) behaviors of at least two or more interconnected entities in virtual or physical networks. Throughout the past centuries, scientists have attempted to explain the emergence of order through the concept of synchronization. C. Huygens in 1665 can be considered as a pioneer. A lot of examples of synchronized phenomena (see [26]) are borrowed from nature, biology, neuroscience, physiology and more recently social networks.

Synchronization can be a very efficient way of tackling engineering issues as well. For example, there is a growing interest in cooperative control problems. Such problems involve several autonomous entities (also called agents) which try to collectively reach a global objective by a suitable connectivity or by adequate couplings. The related applications are mobile robots, unmanned and autonomous vehicles, satellites, air traffic control. Synchronization is also central in communication: video broadcasting, Phase Lock Loop-based equipment. When synchronization must occur in a peer-to-peer communication setup without any external control, it is called self-synchronization. It turns out that self-synchronization is central in cryptography, more specifically, in symmetric cryptography involving the so-called Self-Synchronizing Stream Ciphers (SSSC) (see [20]). Such ciphers are based on generators which can take the form of dynamical systems operating on finite fields and must deliver sequences of high complexity. Those sequences are used to scramble the information to be safely transmitted. For proper decryption, those sequences must be self-synchronized.

Self-Synchronizing Stream Ciphers were patented in 1946. The self-synchronization property has many advantages and is especially relevant to group communications. Since 1960, specific SSSC have been designed and are still used to provide bulk encryption (for hertzian line, RNIS link, ...) in military applications or governmental radio mobile networks. In the early 90s, studies have been performed in [19, 12] to propose secure designs of SSSC. These works have been followed by effective constructions ([12, 24, 11]), but till now, all of these SSSC have been broken, which motivates the search of new constructions of SSSC. It is this issue that is investigated in this paper.

More precisely, we aim at addressing the design of SSSC in terms of control theoretical concepts. The ciphers being viewed as dynamical systems, the main result is that the concepts of flatness together with structural analysis and graph-theory allow for a convenient and systematic way to construct classes of SSSC which are more general than the ones proposed so far. Indeed, it will be shown that the resulting ciphers can incorporate dynamical systems with state transition functions more general than the so-called T-functions which are known to suffer from weakness [22]. Hence, we can expect that this generalization yields more secure SSSC. The special class of flat Linear Parameter Varying (LPV) systems is motivated. Let us stress that since the dynamics operate on finite fields, all the variables, including the varying parameters, take value in a finite set of elements. Hence, the LPV system can be considered as a switched linear system with a number of modes depending on the cardinality of this set.

The paper is organized as follows. Section 2 is devoted to the problem statement. The architecture of Self-Synchronizing Stream Ciphers is presented in terms of dynamical systems. In Section 3, the special class of LPV dynamical systems is presented, the definition of difference flatness is introduced and is particularized to this class. An algebraic characterization in terms of state space matrix representation is provided. The connection between flatness and SSSC is made. It is shown that an automaton with a finite input memory, as required for an SSSC, can be systematically obtained from the left inverse of a flat system. In Section 4, a construction of a family of SSSC involving LPV automata is provided. It is based on structural analysis and graphs. A simple example of construction is given to make a clear understanding of the method. Then, the design of a realistic example and its implementation on a small scale breeding of an ICS-SCADA equipment are described.

2. Self-Synchronizing Stream Ciphers and dynamical systems

2.1. Generalities on stream ciphers

For a stream cipher, it must be given an alphabet A, that is, a finite set of basic elements named symbols. Hereafter, the index k will stand for the discrete-time. On the transmitter part, the plaintext (also called information or message) $m \in \mathcal{M}$ ($\mathcal{M}$ is the message space) is a string of plaintext symbols $m_k \in A$. Each plaintext symbol is encrypted, by means of an encryption (or ciphering) function e, according to:

$$c_{k+b} = e(z_{k+b}, m_k), \tag{1}$$

where $z_k \in A$ is a so-called keystream (or running key) symbol delivered by a keystream generator. The function e is invertible for any prescribed z_k. The resulting quantity $c_k \in A$ is the ciphertext symbol. The integer $b \geq 0$ stands for a potential delay between the plaintext m_k and the corresponding ciphertext c_{k+b}. This is explained by computational or implementation reasons, for instance pipelining (see [8] for example). Consequently, for stream ciphers, the way how to encrypt each plaintext symbol changes on each iteration. The resulting ciphertext $c \in \mathcal{C}$ ($\mathcal{C}$ is called the ciphertext space), that is the string of symbols c_k, is conveyed to the receiver through a public channel.
At the receiver side, the ciphertext c_k is decrypted according to a decryption function d which depends on a running key $\widehat{z}_k \in A$ delivered, similarly to the cipher part, by a keystream generator. The decryption function d obeys the following rule. For any two keystream symbols $\widehat{z}_{k+b}, z_{k+b} \in A$, it holds that

$$\widehat{m}_{k+b} := d(c_{k+b}, \widehat{z}_{k+b}) = m_k \text{ whenever } \widehat{z}_{k+b} = z_{k+b}. \tag{2}$$

Equation (2) means that the running keys z_k and $\widehat{z}_k$ must be synchronized for a proper decryption. The generators delivering the keystreams are parametrized by a secret key denoted in the sequel by $\theta \in \mathcal{K}$ ($\mathcal{K}$ is the secret key space). The distinct classes of stream ciphers (synchronous or self-synchronizing) differ each other by the way on how the keystreams are generated and synchronized. Next, we detail the special class of stream ciphers called Self-Synchronizing Stream Ciphers.

2.2. Keystream generators for Self-Synchronizing Stream Ciphers

A well-admitted approach to generate keystreams has been first suggested in [19]. It is based on the use of so-called state automata with finite input memory as described below. This is typically the case in the cipher Moustique [18]. At the ciphering side, the automaton delivering the keystream takes the form:

$$\begin{cases} q_{k+1} = g_\theta(q_k, c_{k+b}), \\ z_{k+b} = h_\theta(q_k) \end{cases} \tag{3}$$

where $q_k \in A$ is the internal state, g is the next-state transition function parametrized by $\theta \in \mathcal{K}$. As previously stressed, the delay b is due to the fact that the output (also called filtering) function h is pipelined with an architecture involving b stages. If such an automaton has finite input memory, it means that, by iterating (3) a finite number of times, there exists a function ℓ_θ and a finite integer M such that

$$q_k = \ell_\theta(c_{k+b-1}, \ldots, c_{k+b-M}), \tag{4}$$

and thus,

$$z_{k+b} = h_\theta(\ell_\theta(c_{k+b-1}, \ldots, c_{k+b-M})). \tag{5}$$

Actually, the fact that the keystream symbol can be written in the general form

$$z_{k+b} \quad = \quad f_\theta(c_{k-\ell}, \ldots, c_{k-\ell'}), \tag{6}$$

with f_θ a function involving a finite number of past ciphertexts from time $k-\ell$ to $k-\ell'$ ($\ell, \ell' \in \mathbb{Z}$), is a common feature of the SSSC. Equation (6) is called the canonical equation.

Remark 1. *The outcome of implementing the recursive form* (3) *instead of directly implementing the canonical form* (6) *is that we can obtain nonlinear functions f_θ of high complexity by implementing simpler nonlinear functions g_θ. The complexity results from the successive iterations which act as composition operations.*

At the deciphering side, the automaton takes the form

$$\begin{cases} \widehat{q}_{k+1} = g_\theta(\widehat{q}_k, c_{k+b}), \\ \widehat{z}_{k+b} = h_\theta(\widehat{q}_k) \end{cases} \tag{7}$$

where $\widehat{q}_k$ is the internal state. Following the same reasoning, since g_θ corresponds to the state transition function of an automaton with finite input memory, it is clear that after a transient time of maximal length equal to M, it holds that, for $k \geq M$,

$$\widehat{q}_k = q_k \text{ and } \widehat{z}_{k+b} = z_{k+b}. \tag{8}$$

In other words, the generators synchronize automatically after at most M iterations. Hence, the decryption is automatically and properly achieved after at most M iterations too. No specific synchronizing protocol between the cipher and the decipher is needed. This explains the terminology Self-Synchronizing Stream Ciphers. The quantity M is called the synchronization delay.

2.3. Motivation on the use of LPV systems

To obtain a finite input memory feature, the solutions proposed in the open literature call for state transition functions g_θ in the form of shifts or more generally T–functions (T for Triangle), which are functions that propagate dependencies in one direction only. Till now, none of the proposed SSSC have involved non triangular state transition functions although T–functions are known to suffer from weakness [22]. Indeed, T–functions get linear properties which make them easier to cryptanalyse. Furthermore, for the existing SSSC ([16, 9, 10]), the design relied more on ad-hoc approaches than on systematic rules of construction. It is explained by the fact that no systematic methodology for constructing automata with finite input memory and involving general non triangular state transition functions was proposed so far. As a clue to tackle those issues, we first restrict our investigation on the class of linear systems. Let us consider the finite state automaton given by

$$\left\{ \begin{array}{lcl} q_{k+1} & = & Pq_k + Qc_{k+b}, \\ z_{k+b} & = & Rq_k \end{array} \right. \tag{9}$$

$$c_{k+b} = z_{k+b} + Sm_k \tag{10}$$

and

$$\left\{ \begin{array}{lcl} \widehat{q}_{k+1} & = & P\widehat{q}_k + Qc_{k+b}, \\ \widehat{z}_{k+b} & = & R\widehat{q}_k \end{array} \right. \tag{11}$$

$$\widehat{m}_{k+b} = S^{-1}(c_{k+b} - \widehat{z}_{k+b}). \tag{12}$$

where P, Q, R and S are square matrices. Their entries belong to the set A. Assume that the following condition holds:

$$\exists K \in \mathbb{N}, \text{ such that } P^l = 0 \text{ for any } l \geq K. \tag{13}$$

In other words, if the matrix P is nilpotent (the integer K is the index of nilpotency of P), then after a finite number K of iterations of (9) (*resp.* (11)), q_k (*resp.* $\widehat{q}_k$) becomes independent of the initial condition q_0 (*resp.* $\hat{q}_0$) and so, the automaton (9) (*resp.* (11)) has finite input memory, as required for an SSSC. Equations (9) and (10) (*resp.* (11) and (12)) define respectively the cipher and the decipher part of an SSSC. The following correspondences apply:

- $z_{k+b} = Rq_k$ plays the role of the keystream symbol of the cipher;
- $\widehat{z}_{k+b} = R\widehat{q}_k$ plays the role of the keystream symbol of the decipher;
- the function $(z_{k+b}, m_k) \mapsto z_{k+b} + Sm_k$ plays the role of e (encryption function);
- the function $(\widehat{z}_{k+b}, c_{k+b}) \mapsto S^{-1}(c_{k+b} - \widehat{z}_{k+b})$ plays the role of d (decryption function);
- the integer K plays the role of the synchronization delay M.

Thereby, the design of a linear SSSC would be straightforward by choosing a nilpotent matrix P and arbitrary matrices Q, R and S. Unfortunately, linear dynamics are prohibited in cryptography because in such a case, it is well admitted that cryptanalysis is easy. To simply obtain nonlinear systems (with potentially complex nonlinearities as seen later), we propose here a structure inspired from the linear one. We replace a prescribed number of time-invariant entries of the matrices of Equations (9-10) and (11-12) (in particular matrix P) by time-varying parameters. If those parameters denoted by $\rho^i(k)$ ($i = 1, \ldots, L_\rho$ with L_ρ the number of time-varying parameters) get the form $\rho^i(k) = \varphi^i(c_k, c_{k-1}, \cdots, c_{k-s})$ ($s \in \mathbb{N}$) where every φ^i is a nonlinear function, the resulting system becomes nonlinear. It belongs to the class of Linear Parameter-Varying (LPV) systems. Let us note that the functions φ^i have the same form as the canonical equation of an SSSC (see (6)). It means that such an SSSC admits a recursive architecture. This architecture is new regarding the literature and differs from the serial and parallel ones proposed in the 90s by Maurer [19].

The point is that, the matrix raised to the power of l, involved in condition (13) and guaranteeing an automaton with finite input memory in the linear case, must be replaced by the ordered product of l time-varying matrices in the LPV case. Following the linear case, a direct design would consist in choosing parameter-dependent matrices such that the ordered product of l matrices is zero for $l \geq K$, for any initial condition and any values of $\rho^i(k)$. A non triangular state transition function would be simply obtained by non triangular matrices which do not share a common triangular basis. However, that raises a new problem. Indeed, it is closely related to the problem of *mortality*. It is said that a set of matrices is mortal if the zero matrix can be expressed as the product of a finite number of matrices. And yet, from the papers [23, 4, 6], except from some very special cases, the problem of checking mortality of a set of matrices is undecidable. The problem under consideration here is even more challenging since the matrices are not given but must be chosen. In other words, we are not concerned with analysis but with synthesis.

Summarizing our purpose, we are seeking for a systematic methodology to build a family of SSSC. This can be done by designing LPV automata in the form (9-10) and (11-12) where the linear matrices are substituted by parameter dependent matrices involving nonlinear functions φ^i of the shifted cryptograms c_k. A direct design cannot be achieved because it is related to the intricate problem of mortality. In the sequel, it is shown how to tackle this problem. The key idea is, first, to build a flat LPV system. Next, from this flat LPV system, an automaton with finite input memory can be systematically obtained by calculating its left inverse. It is recalled that in control theory, given a dynamical system and considering an input sequence and its corresponding output response, its left inverse is a system that reproduces the input sequence when it is forced by the output response. This connection between flatness and SSSC may appear as quite natural, but is new, regarding the literature. It is detailed in next section.

3. Connection between flat systems and SSSC

Difference flatness is the discrete-time counterpart of differential flatness, a property of some continuous-time controlled dynamical systems introduced in [15]. For a flat discrete-time system, the state variables as well as the input are written as a function of the flat output (including forward and backward shifts in the output). The reader can refer to the book [25] for an introduction on difference flatness and some applications. However, it deals with linear systems. Results on difference flatness for LPV systems have never been published before the paper [13] which is the source of the present paper (Subsections 3.1 and 3.2 are recalled from [13]). Since in this paper, only discrete-time dynamical systems are considered, the word flatness will be often used for short but without confusion.

3.1. Definition of difference flatness for LPV systems

Let us consider a Single Input Single Output (SISO) Linear Parameter-Varying (LPV) system denoted by Σ_ρ, defined over a field $\mathbb{F}$, described by the following state space representation:

$$\Sigma_\rho : \begin{cases} x_{k+1} = A_{\rho(k)}x_k + B_{\rho(k)}m_k \\ c_k = C_{\rho(k)}x_k + D_{\rho(k)}m_k \end{cases} \tag{14}$$

where $k \in \mathbb{N}$ stands for the discrete time, $x_k \in \mathbb{F}^n$ is the state vector, $m_k \in \mathbb{F}$ is the input, $c_k \in \mathbb{F}$ is the output. The matrices $A \in \mathbb{F}^{n\times n}$, $B \in \mathbb{F}^{n\times 1}$, $C \in \mathbb{F}^{1\times n}$ and $D \in \mathbb{F}^{1\times 1}$ are respectively the dynamical matrix, the input matrix, the output matrix and the direct transfer matrix. Such a system is called Linear Parameter-Varying because it is written with a linear dependency with respect to the state vector. The set of all varying parameters of A, B, C and D are collected on a vector denoted by $\rho(k) = \left[\ \rho^1(k), \rho^2(k), ..., \rho^{L_\rho}(k)\ \right] \in \mathbb{F}^{L_\rho}$ where L_ρ is the total number of non-zero (possibly varying) entries. The matrices A, B, C and D do not necessarily depend on all the parameters $\rho^i(k)$. Such systems can exhibit nonlinear dynamics. Indeed, the nonlinearity is obtained by defining the varying parameters $\rho^i(k)$ as nonlinear functions $\varphi^i : \mathbb{F}^{s+1} \to \mathbb{F}$ of the output c_k (or a finite number of shifts) $\rho^i(k) = \varphi^i(c_k, c_{k-1}, \cdots, c_{k-s})$ with s a natural number. Let us note that the notation $\rho^i(k)$ (usual in the literature for LPV systems) is somehow abusive because it does not reflect an explicit dependency with respect to the time k but on quantities, here c_k, indexed with k. Usually, in the context of cryptography, the systems operate on the q elements finite field $\mathbb{F}_q$ where q is a prime power. Hence, in this context, LPV systems can be considered as hybrid systems, and more precisely, switched linear systems with q^{L_ρ} modes since every $\rho^i(k)$ takes values in a finite set.

Now, let us go into more detail. For a non negative integer k, a sequence $\{\rho(k), \rho(k+1), \ldots\}$ will be called a realization of varying parameters and denoted with ρ.

Definition 1. *The system* (14) *is flat, if for every realization ρ, there exists a variable c_k, called flat output, such that all system variables can be expressed as a function of the flat output and a finite number of its backward and forward shifts. In other words, there exist two functions $\mathcal{F}_\rho$ and $\mathcal{G}_\rho$ parametrized by ρ such that*

$$\left\{\begin{array}{lcl} x_k & = & \mathcal{F}_\rho\big(c_{k+k_\mathcal{F}}, \ldots, c_{k+k'_\mathcal{F}}\big) \\ m_k & = & \mathcal{G}_\rho\big(c_{k+k_\mathcal{G}}, \ldots, c_{k+k'_\mathcal{G}}\big) \end{array}\right. \tag{15}$$

where $k_\mathcal{F}$, $k'_\mathcal{F}$, $k_\mathcal{G}$ and $k'_\mathcal{G}$ are $\mathbb{Z}$-valued integers.

3.2. Conditions for flat outputs

First, it must be pointed out that for a given dynamics (first equation of (14)), whether c_k is a flat output or not necessarily depends on the matrices C and D. Besides, for some specific matrices A and B, it may happen that none of the matrices C and D yield to a flat output. As a result, the conditions which guarantee that an output of (14) is flat must be expressed in terms of properties verified by the 4-tuple of state matrices (A, B, C, D). Before proceeding further, it is necessary to deal with the relative degree of (14). We recall below a usual general definition.

Definition 2. *The relative degree of a SISO discrete-time dynamical system is the minimal number r of iterations such that the output c_{k+r} at time $k+r$ is sensitive to the input m_k.*

Now, let us restrict this general definition to the system defined by equation (14). To this end, for $k_2 \geq k_1$, let us denote by $\prod_{l=k_2}^{k_1} A_{\rho(l)}$ the product of matrices $A_{\rho(l)}$ from k_2 to k_1. For $k_2 < k_1$, $\prod_{l=k_2}^{k_1} A_{\rho(l)} = \mathbf{1}_n$ (the identity matrix of dimension n). Let us introduce the quantity $\mathcal{T}^{i,j}_{\rho(k)}$ defined for $j \leq i$ as

$$\begin{array}{lcl} \mathcal{T}^{i,j}_{\rho(k)} & = & C_{\rho(k+i)} \prod_{l=k+i-1}^{k+j+1} A_{\rho(l)} B_{\rho(k+j)} \text{ if } j \leq i-1 \text{ and} \\ \mathcal{T}^{i,i}_{\rho(k)} & = & D_{\rho(k+i)}. \end{array} \tag{16}$$

From (14), it holds that $c_{k+1} = C_{\rho(k+1)} A_{\rho(k)} x_k + \mathcal{T}^{1,0}_{\rho(k)} m_k + \mathcal{T}^{0,0}_{\rho(k+1)} m_{k+1}$. Then, by iterating the output c_k and noting that $\mathcal{T}^{i,j}_{\rho(k+1)} = \mathcal{T}^{i+1,j+1}_{\rho(k)}$ for $i \geq 0$, it follows that

$$c_{k+i} = C_{\rho(k+i)} \prod_{l=k+i-1}^{k} A_{\rho(l)} x_k + \sum_{j=0}^{i} \mathcal{T}^{i,j}_{\rho(k)} m_{k+j}. \tag{17}$$

Hence, if (14) admits a finite relative degree r, it follows from Definition 2 that it is:

1. $r = 0$ if $\mathcal{T}^{0,0}_{\rho(k)} \neq 0$ for any realization ρ;
2. the smallest non-zero integer r such that for any realization ρ;

$$\begin{array}{l} \mathcal{T}^{i,j}_{\rho(k)} = 0 \quad \text{ for } i = 0, \ldots, r-1 \text{ and } j = 0, \ldots,\ i, \\ \mathcal{T}^{r,0}_{\rho(k)} \neq 0. \end{array} \tag{18}$$

Hence, it holds that

$$c_{k+r} = C_{\rho(k+r)} \prod_{l=k+r-1}^{k} A_{\rho(l)} x_k + \mathcal{T}^{r,0}_{\rho(k)} m_k. \tag{19}$$

In the sequel, to make the notation less cluttered, we will merely write $\mathcal{T}$ instead of $\mathcal{T}^{r,0}_{\rho(k)}$.

Proposition 1. *If the LPV system* (14) *has a relative degree $r < \infty$, the following condition involving product of matrices, fulfilled for a positive integer K and any realization ρ*

$$P_{\rho(k+K-1:k+K-1+r)} P_{\rho(k+K-2:k+K-2+r)} \cdots P_{\rho(k:k+r)} = 0 \tag{20}$$

with

$$P_{\rho(k:k+r)} = A_{\rho(k)} - B_{\rho(k)} \mathcal{T}^{-1} C_{\rho(k+r)} \prod_{l=k+r-1}^{k} A_{\rho(l)} \tag{21}$$

is equivalent to c_k is a flat output.

Proof 1. *If the LPV system* (14) *has a relative degree $r < \infty$, the following dynamical system can always be defined since $\mathcal{T}$ is non-zero.*

$$\left\{ \begin{array}{lcl} q_{k+1} & = & P_{\rho(k:k+r)} q_k + B_{\rho(k)} \mathcal{T}^{-1} c_{k+r} \\ \widehat{m}_{k+r} & = & \mathcal{T}^{-1}(C_{\rho(k+r)} \prod_{l=k+r-1}^{k} A_{\rho(l)} q_k - c_{k+r}) \end{array} \right. \tag{22}$$

Iterating $K-1$ times the first equation of (22) *yields*

$$\begin{array}{l} x_k = P_{\rho(k-1:k-1+r)} \cdots P_{\rho(k-K:k-K+r)} x_{k-K} + \\ \sum_{i=1}^{K-1} \Big[\prod_{j=0}^{i-1} P_{\rho(k-j-1:k+r-j-1)} \Big] \mathcal{T}^{-1} B_{\rho(k-i-1)} c_{k-i-1+r}. \end{array} \tag{23}$$

Let us define $\varepsilon_k = x_k - q_k$. By simple manipulations, it can be shown that ε_k verifies $\varepsilon_{k+1} = P_{\rho(k:k+r)} \varepsilon_k$. Hence, after a finite transient time equal to K, since (20) *holds, one has $q_k = x_k$. Hence, since the parameters $\rho^i(k)$ depend on shifted outputs, the state vector x_k exclusively depends on shifted outputs if and only if* (20) *holds. Finally, x_k reads*

$$x_k = \sum_{i=0}^{K-1} \Big[\prod_{j=0}^{i-1} P_{\rho(k-j-1:k+r-j-1)} \Big] \mathcal{T}^{-1} B_{\rho(k-i-1)} c_{k-i-1+r}, \tag{24}$$

which gives the explicit function $\mathcal{F}_\rho$ involved in (15)*. Letting $\omega_k = m_k - \widehat{m}_{k+r}$ and following the same reasoning, it is shown that m_k exclusively depends on shifted outputs. The explicit form of $\mathcal{G}_\rho$ involved in* (15) *is obtained by substituting* (24) *into the second equality of* (22)*. It shows that the system defined by Equation* (14) *is flat with c_k as flat output if and only if* (20) *holds.*

Remark 1. *The system defined by Equation* (22) *is nothing but the left inverse system of the LPV system* (14).

3.3. Connection

In this subsection, we bring out the connection between flat LPV systems and SSSC.

Proposition 2. *If the LPV system* (14) *has relative degree $r < \infty$ and is flat, then the finite state automata given by*

$$\left\{\begin{array}{lcl} q_{k+1} & = & P_{\rho(k:k+r)}q_k + B_{\rho(k)}\mathcal{T}^{-1}c_{k+r}, \\ z_{k+r} & = & C_{\rho(k+r)}\prod_{l=k+r-1}^{k} A_{\rho(l)}q_k \end{array}\right. \tag{25}$$

$$c_{k+r} = z_{k+r} + \mathcal{T}m_k \tag{26}$$

and

$$\left\{\begin{array}{lcl} \widehat{q}_{k+1} & = & P_{\rho(k:k+r)}\widehat{q}_k + B_{\rho(k)}\mathcal{T}^{-1}c_{k+r}, \\ \widehat{z}_{k+r} & = & C_{\rho(k+r)}\prod_{l=k+r-1}^{k} A_{\rho(l)}\widehat{q}_k \end{array}\right. \tag{27}$$

$$\widehat{m}_{k+r} = \mathcal{T}^{-1}(c_{k+r} - \widehat{z}_{k+r}) \tag{28}$$

define an SSSC.

Proof 2. *If the system* (14) *has relative degree $r < \infty$ and is flat, then $\mathcal{T}$ is non-zero and thus, systems (25-26) and (27-28) are well defined. Systems (25) and (27) are the left inverse system of* (14) *as emphasized in Remark 1. The property* (20)*, that is flatness, ensures that systems* (25) *and* (27) *are automata with finite input memory. They can respectively be identified with the system* (3) *and with the system* (7)*. Identifying* (26) *with* (1) *and* (28) *with* (2) *gives respectively the encryption and the decryption function.*

The following correspondences hold:

- $z_{k+r} = C_{\rho(k+r)}\prod_{l=k+r-1}^{k} A_{\rho(l)}q_k$ plays the role of the keystream symbol of the cipher;
- $\widehat{z}_{k+r} = C_{\rho(k+r)}\prod_{l=k+r-1}^{k} A_{\rho(l)}\widehat{q}_k$ plays the role of the keystream symbol of the decipher;
- r (relative degree) plays the role of b (delay);
- the function $(z_{k+r}, m_k) \mapsto z_{k+r} + \mathcal{T}m_k$ plays the role of e (encryption function);
- the function $(\widehat{z}_{k+r}, c_{k+r}) \mapsto \mathcal{T}^{-1}(c_{k+r} - \widehat{z}_{k+r})$ plays the role of d (decryption function);
- the integer K plays the role of the synchronization delay M.

It is recalled that the nonlinearity is obtained by defining the values of every varying parameters $\rho^i(k)$ $(i = 1, \ldots, L_\rho)$ involved in the matrices of (25-28) as nonlinear functions φ^i of a finite number of past cryptograms ($\rho^i(k) = \varphi^i(c_k, c_{k-1}, \cdots, c_{k-s})$). In cryptography, those functions are called S-boxes.

Remark 2. *The encryption function e and the decryption functions d are quite simple. This is a common feature for SSSC. For example in the Boolean case, those functions are often nothing but the exclusive or. Actually, the security is essentially based on the properties of the keystream sequences delivered by* (25) *and* (27).

To sum up, as an alternative to a direct design of automata with finite input memory required for SSSC, which is in general impossible because related to the problem of mortality, an indirect approach has been proposed. It consists, first, on the design of a flat LPV system (14). Next, we deduce the automata (25) and (27) by calculating the left inverse system of the LPV system. It is central to emphasize that since the LPV system is flat, the automata are necessarily with finite input memory. The outcome of such an approach is that the matrix $P_{\rho(k:k+r)}$ involved in (25) and (27) define a state transition function which is not necessarily triangular and thus, may represent a larger class than T–functions. This approach is new in cryptography.
In our context, the flatness property must be guaranteed for any non-zero entries of the matrices involved in (25) and (27). Indeed, some entries correspond to time-varying coefficients, the other ones are constant entries. But all of these entries are parametrized by the secret key θ which ranges in the key space $\mathcal{K}$, a huge set of values. All in all, flatness must be ensured from a structural point of view. Hence, structural analysis can be a natural and relevant solution to construct structural flat systems. The construction is detailed in next section.

4. Construction of structural flat systems and SSSC

4.1. Structured systems and digraphs

The core idea is that the LPV system (14) can be considered as an admissible realization of a corresponding structured linear system. A structured linear system is a system only defined by the sparsity pattern of the state space realization matrices. In other words, for a structured linear system, we distinguish between the entries that are fixed to zero and the other ones that can take any value in $\mathbb{F}$, including the ones which are time-varying. Hence, a structured linear discrete-time system is a system that admits the form:

$$\Sigma_\Lambda : \quad x_{k+1} \quad = \quad I_A x_k + I_B m_k \tag{29}$$

The entries of the matrices of Σ_Λ are '0' or '1'. In particular, the entries $A(i,j)$ of I_A (*resp.* $B(i)$ of I_B) that are '0' means that there are no relation (dynamical interaction) between the state x^i_{k+1} at time $k+1$ and the state x^j_k at time k (*resp.* the state x^i_{k+1} at time $k+1$ and the input m_k at time k). The entries that are '1' means that there is a relation. As a simple example, let us consider an LPV system with the setting

$$A_{\rho(k)} = \begin{pmatrix} a & 0 \\ 1 & \rho^1(k) \end{pmatrix} \text{ and } B_{\rho(k)} = \begin{pmatrix} \rho^2(k) \\ 0 \end{pmatrix}$$

where a is a constant element in $\mathbb{F}$, $\rho^1(k)$ and $\rho^2(k)$ are varying parameters in $\mathbb{F}$. The dynamical matrix and the input matrix I_A and I_B of the corresponding structured linear system read:

$$I_A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix},\ I_B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

As a consequence, if the structural linear system (29) derived from (14) is flat, the flatness will hold for any realization ρ of (14). Hence, the challenge is to define a methodology to construct flat linear structural systems. A graph-based approach is suggested below to that purpose. It is based on the result provided in [21]. Let us recall some basic requirements on digraphs. A digraph $\mathcal{G}(\Sigma_\Lambda)$ describing the structured linear system associated to the state equations (14), is the combination of a vertex set $\mathcal{V}$ and an edge set $\mathcal{E}$. The vertices represent the states and the input components of Σ_Λ while the edges describe the dynamic relations between these variables. One has $\mathcal{V} = \mathbf{X} \cup \{\mathbf{m}\}$ where $\mathbf{X}$ is the set of state vertices defined as $\mathbf{X} = \left\{\mathbf{x^1}, \ldots, \mathbf{x^n}\right\}$ and $\mathbf{m}$ is the input vertex. The edge set is $\mathcal{E} = \mathcal{E}_A \cup \mathcal{E}_B$, with $\mathcal{E}_A = \left\{(\mathbf{x^i}, \mathbf{x^j})\,|A(i,j) \neq 0\right\}$ and $\mathcal{E}_B = \left\{(\mathbf{m}, \mathbf{x^i})\,|B(i) \neq 0\right\}$. They also correspond to the weights of the edges of the digraph. In the sequel, we will denote by $\mathbf{v^j}$, $(j = 0, \ldots, n)$ a vertex of the digraph $\mathcal{G}(\Sigma_\Lambda)$ regardless whether it is the input or a state vertex.

- A directed path $\mathbf{P}$ is a sequence of successive edges directed in the same direction which connect a sequence of vertices. It is said that the path $\mathbf{P}$ covers a vertex if this vertex is the begin or the end vertex of one of the edges of $\mathbf{P}$;
- In a directed path from a vertex $\mathbf{v^i}$ to a vertex $\mathbf{v^j}$, it is said that $\mathbf{v^j}$ is a successor of $\mathbf{v^i}$ and conversely, $\mathbf{v^i}$ is a predecessor of $\mathbf{v^j}$;
- A simple path is a path which contains no repeated vertices;
- The length of a directed path $\mathbf{P}$ is equal to the number of edges involved in $\mathbf{P}$. We let $\ell(\mathbf{v^i}, \mathbf{v^j})$ denote the minimal length of a path connecting $\mathbf{v^i}$ to $\mathbf{v^j}$;
- $V_{ess}(\mathbf{v^i}, \mathbf{v^j})$ is the set of vertices, called essential vertices from $\mathbf{v^i}$ to $\mathbf{v^j}$, which are common to all the paths connecting $\mathbf{v^i}$ to $\mathbf{v^j}$.

Remark 2. *In graph theory, the terminology may be different according to authors. A directed path as defined here may be called a walk. A simple path as defined here may be merely called a path.*

We recall from [21] the necessary and sufficient conditions which must be satisfied for any vertex $\mathbf{v^i}$ $(i \in \{1, \ldots, n\})$ of the digraph $\mathcal{G}(\Sigma_\Lambda)$ to be associated to a flat output $c_k^F = x_k^i$ $(i \in \{1, \ldots, n\})$ of (14). In such a case, the output state space matrix $C_{\rho(k)} = C$ is constant and has zero entries except the entry located at the column number i which is equal to one. The direct transfer matrix $D_{\rho(k)} = D$ is the zero matrix.

Theorem 1. *The output $c_k^F = x_k^i$ ($i \in \{1,\ldots,n\}$) of the structured linear system* (29)*, associated to the vertex $\mathbf{v^F} \in \mathbf{X}$ in the associated digraph $\mathcal{G}(\Sigma_\Lambda)$, is generically a flat output iff, the three following conditions hold:*
C0. *$\mathbf{v^F}$ is a successor of $\mathbf{m}$;*
C1. *Any simple paths from $\mathbf{m}$ to $\mathbf{v^F}$ have the same length equal to $\ell(\mathbf{m}, \mathbf{v^F})$;*
C2. *Any cycles cover at least an element of $V_{ess}(\mathbf{m}, \mathbf{v^F})$.*

Actually, as shown in [21], Conditions **C0-C2** ensure that the state and the input of the structured linear system (29) do no longer depend on the initial state after a finite transient time regardless of the weights of the edges. Consequently, it is shown that they can be exclusively expressed as some functions of shifted outputs parametrized by the weights of the edges. Since a weight is a constant or a function φ^i of shifted outputs (in our context, the shifted cryptograms), both the state and the input depend exclusively on shifted outputs (here, the cryptograms). Hence, Conditions **C0-C2** are instrumental for the construction of structural flat dynamical systems. A systematic construction of digraphs fulfilling **C0-C2** is proposed below.

4.2. Construction of flat LPV systems

The digraph $\mathcal{G}(\Sigma_\Lambda)$ related to the system Σ_Λ of dimension n involves $n+1$ vertices. The input is assigned to the vertex denoted by $\mathbf{v^0}$. The other n vertices are denoted by $\mathbf{v^1}, \ldots, \mathbf{v^n}$. Let $\mathbf{v^r}$ be the vertex that corresponds to the flat output $\mathbf{v^r}$, that is $\mathbf{v^F} = \mathbf{v^r}$.

Step 1: For, $i = 0, \ldots, n-1$, add the edges $(\mathbf{v^i}, \mathbf{v^{i+1}})$. There are r edges which connect $\mathbf{v^0}$ to $\mathbf{v^r}$. Hence, the relative degree of the corresponding structured dynamical system is r.

After Step 1, Conditions **C0-C2** are fulfilled for $\mathbf{v^F} = \mathbf{v^r}$. However, this line topology corresponds to quite trivial dynamical systems since it corresponds to state transition functions in the form of simple shifts. Let us recall that we aim at designing an automaton possibly involving state transition functions more general than T–functions. A shift is a special and trivial T–function. To this end, the following steps provide a way of adding edges $(\mathbf{v^i}, \mathbf{v^j})$ while guaranteeing that Conditions **C0-C2** are still fulfilled.

Step 2: Add the edges $(\mathbf{v^{r+i}}, \mathbf{v^{r+i+1}})$ for $i = 1, \ldots, n-r-1$. Step 2 allows vertex $\mathbf{v^j}$, $j = r+1, \ldots, n$ to have a predecessor. Indeed, if not so, the dynamics of the corresponding vertex $\mathbf{v^j}$ would reduce to $x_{k+1}^j = 0$ and would be clearly useless. The resulting path will be called main directed path and is depicted in Figure 1.

$$\begin{tikzcd}
\mathbf{v^0} \arrow[r] & \mathbf{v^1} \arrow[r] & \mathbf{v^2} \arrow[r, dashed] & \mathbf{v^{r-1}} \arrow[r] & \mathbf{v^r} \arrow[r, dashed] & \mathbf{v^n}
\end{tikzcd}$$

Figure 1: Digraph obtained after completion of Step 1-2. The vertex $\mathbf{v^r}$ corresponds to the flat output

Step 3: Add the edges $(\mathbf{v^r}, \mathbf{v^i})$, $i = 1, \ldots, n$ that connect the vertex $\mathbf{v^r}$ to any other vertices of the graph (except the vertex $\mathbf{v^0}$ related to the input).

Step 4: For every vertex $\mathbf{v^i}$, $i = 1, \ldots, r-1$, add the directed edge $(\mathbf{v^i}, \mathbf{v^j})$ for $j = 1, \ldots, i$. Note that those edges introduce cycles, including cycles of order 1, but those cycles satisfy Condition **C2** since $\mathbf{v^i}$, for $i = 0, \ldots, r-1$, belong to $V_{ess}(\mathbf{m}, \mathbf{v^r})$.

The graph obtained after Step 1-4 is depicted in Figure 2.

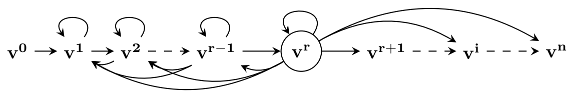


Figure 2: Graph obtained after Step 1-4.

Step 5: For every vertex $\mathbf{v_i}$, $i = r+1, \ldots, n$, add the directed edge $(\mathbf{v_i}, \mathbf{v_j})$ for $j = 1, \ldots, r$ and $j = i+2, \ldots, n$. Let us note that this step generates again cycles but Condition **C2** is still satisfied.

The resulting digraph after completion of Step 1-5 is depicted in Figure 3.

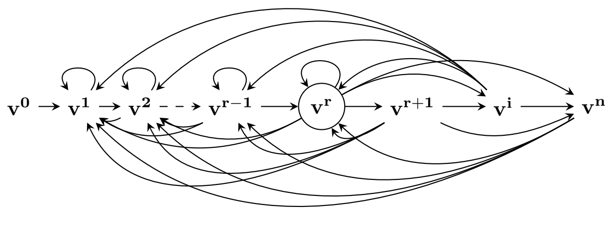


Figure 3: Graph obtained after completion of Step 1-5.

As a result, a digraph is parametrized by the triplet (n, r, n_a). The number of vertices of the digraph is equal to $n+1$ (being n the dimension of the system (29) and (14)). Indeed, there are n vertices assigned to the state components and one assigned to the input. The relative degree r fulfilling (18) gives the number of edges in the main directed path. The integer n_a defines the desired number of edges in the digraph $\mathcal{G}(\Sigma_\Lambda)$. It must satisfy $n_a \leq n_M$, where n_M is the maximal number of edges resulting from the construction Step 1-5. A simple counting leads to:

$$n_M = \frac{n(n+1)}{2} + r. \tag{30}$$

During the construction, at each step, we can decide whether we actually add the edges or not. That introduces flexibility in the perspective of providing distinct graphs and thus, distinct SSSC as detailed in Subsection 4.3.

The adjacency matrix, denoted by $\mathcal{I}$, associated to the digraph $\mathcal{G}(\Sigma_\Lambda)$ and obtained after completion of Step 1-5, can be defined as follows. It is the $(n+1) \times (n+1)$ matrix

$$\mathcal{I} = \left(\begin{array}{c:c} 0 & I_B^t \\ \hdashline 0 & \\ \vdots & I_A^t \\ 0 & \end{array} \right) \tag{31}$$

where I_A^t and I_B^t stands respectively for the transpose of the structured matrices I_A and I_B. The entries $\mathcal{I}_{ij}$ are defined as follows for $1 \leq i, j \leq n$

$$\mathcal{I}_{ij} = \begin{cases} 1 \text{ if there exists an edge from } \mathbf{v^j} \text{ to } \mathbf{v^i} \\ 0 \text{ otherwise.} \end{cases} \tag{32}$$

The adjacency matrix associated to $\mathcal{G}(\Sigma_\Lambda)$, obtained after completion of Step 1-5, is given by

$$\begin{array}{c} \\ \mathbf{v^0} \\ \mathbf{v^1} \\ \mathbf{v^2} \\ \mathbf{v^3} \\ \vdots \\ \mathbf{v^r} \\ \mathbf{v^{r+1}} \\ \vdots \\ \mathbf{v^{n-1}} \\ \mathbf{v^n} \end{array} \begin{array}{c} \begin{array}{cccccccccc} \mathbf{v^0} & \mathbf{v^1} & \mathbf{v^2} & \mathbf{v^3} & \cdots & \mathbf{v^r} & \mathbf{v^{r+1}} & \cdots & \mathbf{v^{n-1}} & \mathbf{v^n} \end{array} \\ \left(\begin{array}{cccccccccc} 0 & 1 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & \cdots & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & \cdots & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & \cdots & 1 & 0 & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 1 & 1 & 1 & \cdots & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & \cdots & 1 & 0 & 0 & 0 & 0 \end{array} \right) \end{array} \tag{33}$$

This being the case, we could reformulate Step 1-5 in terms of adjacency matrix. However, the design has been given in terms of a digraph construction because it appears as more natural since Conditions **C0-C2** involve a digraph consideration.

The computational complexity of the algorithms for checking Condition **C0-C2** is of the same order than the algorithms used for finding successors and predecessors of vertex subsets or for computing essential vertices in a digraph [5]. Thus, the complexity of the construction is polynomial and is $O(n^3)$. The open source software Sagemath (see `http://www.sagemath.org/`) is appropriate to construct the digraph $\mathcal{G}(\Sigma_\Lambda)$.

We have summed up after Remark 2 the main line of the design of an SSSC and have raised the problem of constructing a structural flat system. Now, being this issue fixed, next subsection aims at giving the steps for a complete design of SSSC leading to Equations (25)-(28). It is illustrated with a simple example for a clear understanding. A realistic example will be given in Section **??**.

4.3. Construction of SSSC

Step S1: Choose a triplet (n, r, n_a). Since the flat output c_k^F is one component x_k^i $(i = 1, \dots, n)$, it follows that $C_{\rho(k)} = \ (0 \cdots 0\ 1\ 0\ \dots 0)$ (the only entry 1 is located at the r^{th} column) and $D_{\rho(k)} = (0\ \cdots 0)$. It is recalled that c_k^F will play the role of the ciphertext.
Example: Consider the triplet $(n = 2, r = 2, n_a = 5)$. As the relative degree r is equal to 2, it means that the component x_k^2 of the state vector of the corresponding LPV system (14) will be the flat output c_k^F. Hence, $C_{\rho(k)} = C = [0\ 1]$ and $D_{\rho(k)} = D = 0$.

Step S2: Construct the corresponding digraph $\mathcal{G}(\Sigma_\Lambda)$ according to Step 1-5.
Example: The graph corresponding to the triplet $(n = 2, r = 2, n_a = 5)$ is depicted in Figure 4.

$$\begin{tikzcd}
\mathbf{v}^0 \arrow[r] & \mathbf{v}^1 \arrow[loop above] \arrow[r] & \mathbf{v}^2 \arrow[loop above] \arrow[l, bend left]
\end{tikzcd}$$

Figure 4: Digraph obtained for $n = 2$, $r = 2$, $n_a = n_M = 5$. The flat output is $c_k^F = x_k^2$ and corresponds to the vertex $\mathbf{v^2}$

Step S3: Given the digraph $\mathcal{G}(\Sigma_\Lambda)$ characterized by the triplet (n, r, n_a), extract, from the adjacency matrix $\mathcal{I}$, the matrices I_A and I_B of the structured linear system.
Example: For the digraph $\mathcal{G}(\Sigma_\Lambda)$ depicted in Figure 4, the adjacency matrix $\mathcal{I}$ and the structured matrices I_A and I_B as defined by (31) are

$$\mathcal{I} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \ I_A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \ I_B = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{34}$$

Step S4: Replace some of the non-zero entries of I_A and I_B by a nonlinear function $\rho^i(k) = \varphi^i(c_k, c_{k-1}, \cdots, c_{k-s})$ (S-box) to construct the matrices $A_{\rho(k)}$ and $B_{\rho(k)}$ of (14). Not all '1' entries of I_A and I_B must be assigned to a nonlinear function. Some of them can be merely constant. The choice must obey a trade-off between complexity of the architecture and security (a matter only discussed

in a shallow way in Subsection 4.5). Since the construction ensures structural flatness, any choice will preserve the self-synchronization property except for some particular cases which deserve a special treatment. Indeed, it may happen that cryptograms cause $\rho^i(k)$ to vanish at time k and thus, the corresponding edge to be cancelled. To preserve a finite and constant relative degree for any k, it is sufficient that the weights assigned to the main path connecting the input vertex $\mathbf{v^0}$ to the vertex $\mathbf{v^r}$ are constant.
Example: Let us keep constant and equal to 1 the first three entries of A and let the fourth entry be time-varying. It is denoted by $\rho^1(k)$. This finally leads to the following matrices $A_{\rho(k)}$ and $B_{\rho(k)}$:

$$A_{\rho(k)} = \begin{pmatrix} 1 & 1 \\ 1 & \rho^1(k) \end{pmatrix} \text{ and } B_{\rho(k)} = B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Step S5: Complete the design by deriving the equations of Proposition 2. In particular, calculate the matrix (21) governing the state transition function of the automata (25) and (27).

Example: For the example, one has $P_{\rho(k:k+2)} = A_{\rho(k)} - B \cdot C \cdot A_{\rho(k+1)} \cdot A_{\rho(k)}$. One obtains

$$P_{\rho(k:k+2)} = \begin{pmatrix} -\rho^1(k+1) & -\rho^1(k) \cdot \rho^1(k+1) \\ 1 & \rho^1(k) \end{pmatrix}.$$

Let us point out that (20) holds for $K = 2$. Indeed, it's a simple matter to see that $P_{\rho(k+1:k+3)}P_{\rho(k:k+2)} = 0$. It corroborates that c_k is a flat output (but it is by construction since the digraph $\mathcal{G}(\Sigma_\Lambda)$ fulfills **C0-C2**).

The approach proposed here gives thereby a systematic procedure to construct a family of SSSC according to the triplet (n, r, n_a), the type of functions φ^i and their position in the adjacency matrix. It remains to show that the state transition functions are likely to be non triangular. Next subsection precisely illustrates that, even for this simple example, the state transition matrix $P_{\rho(k:k+2)}$ is non triangular.

4.4. Non triangular state transition function

We aim at proving that the matrix $P_{\rho(k:k+2)}$ is not conjugate to a shift or to a T–function. To this end, it must be shown that the matrices $P_{\rho(k:k+2)}$, for any realization ρ, cannot be simultaneously triangularizable. The characteristic polynomial of $P_{\rho(k:k+2)}$ is given by $N(X) = X\big(X + (\rho^1(k+1) - \rho^1(k))\big)$. Hence, the eigenvalues of $P_{\rho(k:k+2)}$ are 0 and $\rho^1(k) - \rho^1(k+1)$. It follows that the eigenspace related to the eigenvalues of $P_{\rho(k:k+2)}$ is spanned by:

$$V_1 = \begin{pmatrix} -\rho^1(k) \\ 1 \end{pmatrix} \quad \text{and} \quad V_2 = \begin{pmatrix} -\rho^1(k+1) \\ 1 \end{pmatrix}$$

And yet, a necessary condition for simultaneous triangularization (see [14]) is that the matrices $P_{\rho(k:k+2)}$ share a common eigenvector for any realization ρ.

As a result, since the parameter $\rho^1(k)$ varies, the matrices $P_{\rho(k:k+2)}$ do not fulfill such a requirement. Clearly, even in this trivial example, we have shown that an SSSC with state transition function different from a T–function can be obtained.

4.5. Realistic example and platform

A realistic algorithm compatible with some central cryptographic considerations has been designed and experimented. Clearly, detailed considerations on security are out of the scope of the paper. However, we give below the main specifications of the SSSC followed by some arguments that motivate the choices.

- Field on which the cryptosystem operates: Galois field $GF(16)$
- Dimension n: 40
- S-boxes φ^i: the function $c_k \longmapsto \frac{1}{c_k} + \alpha^2$ where α is a primitive element of $GF(16)$
- Relative degree r: 3
- Number n_a of edges in the digraph $\mathcal{G}(\Sigma_\Lambda)$: 120
- Number of S-boxes: 80

The specifications are based on criteria which are, to mention a few, resistance to time-memory trade-off attacks [17], resistance against algebraic attacks [7], good diffusion delay [2] and depth [3].

The open source software Sagemath (see `http://www.sagemath.org/`) has been used to elaborate the digraph $\mathcal{G}(\Sigma_\Lambda)$ corresponding to the triplet $(n = 40, r = 3, n_a = 120)$. The code is provided in the Appendix (see Algorithm 1). The construction has been performed on an Intel CORE i7 CPU 2.26 GHz running Linux Ubuntu 14.04. All experiments ran single-threaded on the processors. It takes 21 ms on the computer to obtain the digraph $\mathcal{G}(\Sigma_\Lambda)$.

The platform is a small scale breeding of an ICS-SCADA (Industrial Control Systems - Supervisory Control and Data Acquisition) equipment. It involves several sources of information (temperature sensors, touchscreen, ...). Those information are securely conveyed through a local network, the encryption and decryption being achieved by the way of Arduino MEGA cards involving an ATMEGA 2560 processor. It has a 16 MHz Clock Speed and a 256 KB Flash Memory. The execution of the encryption and decryption algorithms are performed with a throughput of 8.263 ms. A monitoring system (Figure 7) allows to select the source, the destination and to analyze the self-synchronization performances.

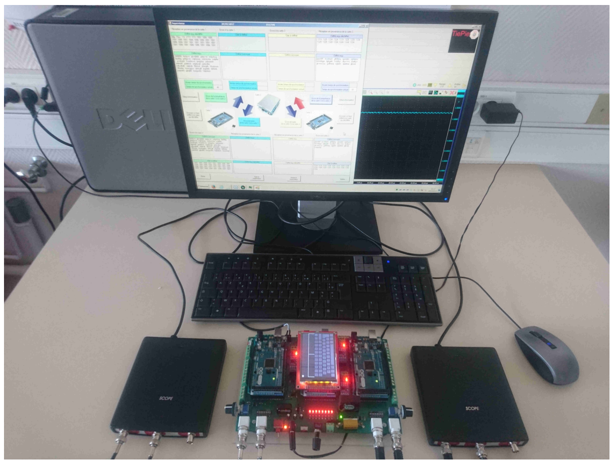


Figure 5: The platform involving the sensors, the touch screen, the Arduino MEGA cards and the oscilloscopes to visualize the analog signals.

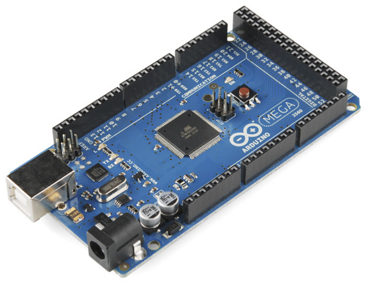
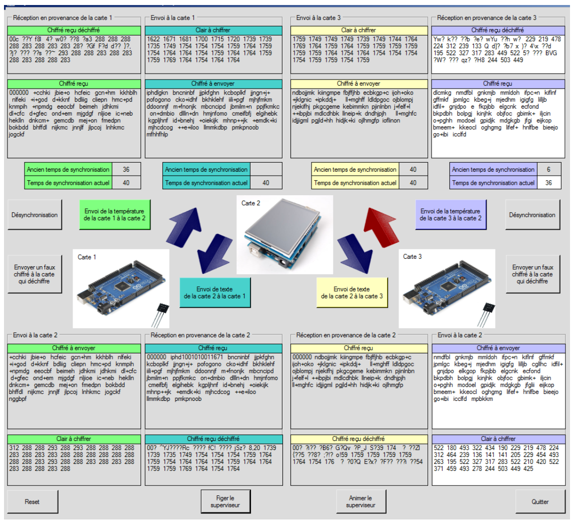


Figure 6: Arduino mega 2560 card

5. Conclusion

Cryptography has been investigated as a new field of application of control theory. A systematic and general construction of Self Synchronizing Stream Ciphers based on flat Linear Parameter Varying (LPV) dynamical systems has been proposed. It is based on constructive algebraic and graph-based conditions guaranteeing structural flatness of an LPV system and so self-synchronizing property. It holds for any type of nonlinearities (S-boxes) and values of the secret key. More precisely, a systematic methodology has been proposed in order to obtain SSSC involving non triangular state transition functions. It has been shown that such an approach allows to enlarge the existing classes of SSSC.


Réception en provenance de la carte 1
Envoi à la carte 1
Envoi à la carte 3
Réception en provenance de la carte 3
Désynchronisation
Envoi de la température de la carte 1 à la carte 2
Carte 2
Envoi de la température de la carte 3 à la carte 2
Désynchronisation
Envoyer un faux chiffré à la carte qui déchiffre
Carte 1
Envoi de texte de la carte 2 à la carte 1
Envoi de texte de la carte 2 à la carte 3
Carte 3
Envoyer un faux chiffré à la carte qui déchiffre
Envoi à la carte 2
Réception en provenance de la carte 2
Réception en provenance de la carte 2
Envoi à la carte 2
Reset
Figer le superviseur
Animer le superviseur
Quitter

Réception en provenance de la carte 1 — Chiffré reçu déchiffré

```
00c ??Y f8l  4? w0? ??8 ?a3 288 288 288
288 283 288 283 283 28? ?Gf F?d d?? }?.
?j? ??? ??s ??~ 293 288 288 283 288 283
288 283 288 288 283
```

Réception en provenance de la carte 1 — Chiffré reçu

```
000000  +cchki  jbie+o  hcfeic  gcn+hm  kkhblh
 nlfeki  +i+god  d+kknf  bdliig  cliepn  hmc+pd
knmpih  +npmdg  eeocbf  beimeh  jdhkmi
dl+cfc  d+gfec  ond+em  mjgdgf  nljioe  ic+neb
heklln  dnkcm+  gemcdb  mej+on  fmedpn
bokbdd  bhffdl  nijkmc  jnnjlf  jlpcoj  lnhkmc
jogckf
```

Ancien temps de synchronisation	36
Temps de synchronisation actuel	40

Envoi à la carte 1 — Clair à chiffrer

```
1622 1671 1681 1700 1715 1720 1739 1739
1735 1749 1754 1754 1754 1759 1764 1759
1754 1764 1764 1754 1759 1764 1764 1759
1759 1769 1764 1754 1764 1764
```

Envoi à la carte 1 — Chiffré à envoyer

```
iphdlgkn  bncninbf  jjpkfghn  kcboplkf  jjngn+j+
pofogono  cko+idhf  bkhklehf  iili+pgf  mjhjfmkm
ddoonnjf  m+fnonjk  mbcncipd  jbmilm+n  ppjfkmkc
 on+dmbio  dllln+dn  hmjnfomo  cmeifbfj  elgihebk
 kgpljhnf  id+bnehj  +oiekijk  mhnp++jk  +emdk+ki
 mjhcdcog  ++e+iloo  llmmkdbp  pmkpnoob
mfhhfhlp
```

Ancien temps de synchronisation	
Temps de synchronisation actuel	40

Envoi à la carte 3 — Clair à chiffrer

```
1759 1749 1749 1749 1739 1749 1744 1764
1769 1764 1759 1764 1759 1759 1759 1759
1759 1759 1764 1754 1764 1754 1754 1754
1754 1759 1759 1754 1759
```

Envoi à la carte 3 — Chiffré à envoyer

```
ndbojjmk  kiingmpe  fbjffjhb  ecbkgp+c  ijoh+oko
+jklgnic  +ipkddj+    ll+mghff  ldldpgoc  ojblompj
njekifhj  pkgcgeme  kebimmkn  pjinlnbn  j+felf+l
++ibpjbi  mdlcdhbk  llneip+k  dndhjpjh     ll+mghfc
idjjigml  pgjld+hh  hidjk+ki  ojlhmgfp  ioflinon
```

Ancien temps de synchronisation	40
Temps de synchronisation actuel	40

Réception en provenance de la carte 3 — Chiffré reçu déchiffré

```
Yw? k?? ??b ?e? wYu ??h w?  229 219 478
224 312 239 133 Q d]? ?b7 x )? 4\x ??d
195 522 327 317 283 449 522 5? ??? BVG
?W? ??? qz? ?H8 244 503 449
```

Réception en provenance de la carte 3 — Chiffré reçu

```
dlcmkg  nmdfbl  gnkmjb  mmldoh  ifpc+n  kiflnf
gffmkf  jpmlgc  kbeg+j  mjedhm  igigfg  lilljb
idfil+  gnjdpo  e  fkpjbb  elgcnk  ecfond
bkpdbh  bolpgj  kinjhk  objfoc  gbimk+  iljcin
o+pghh  modoel  gpidjk  mdgkgb  jfgi  eijkop
bmeem+  kkeocl  oghgmg  lifef+  hnlfbe  bieejo
go+ibi  icclfd
```

Ancien temps de synchronisation	6
Temps de synchronisation actuel	36

Envoi à la carte 2 — Chiffré à envoyer

```
+cchki  jbie+o  hcfeic  gcn+hm  kkhblh  nlfeki
+i+god  d+kknf  bdliig  cliepn  hmc+pd  knmpih
+npmdg  eeocbf  beimeh  jdhkmi  jdhkmi  dl+cfc
 d+gfec  ond+em  mjgdgf  nljioe  ic+neb  heklln
dnkcm+  gemcdb  mej+on  fmedpn  bokbdd
bhffdl  nijkmc  jnnjlf  jlpcoj  lnhkmc  jogckf
nggbpf
```

Envoi à la carte 2 — Clair à chiffrer

```
312 288 288 293 288 293 288 288 288 288
283 288 283 283 288 288 288 288 288 288
288 288 283 293 288 288 283 288 283 288
283 288 288 283 288
```

Réception en provenance de la carte 2 — Chiffré reçu

```
000000  iphd1001010011671  bncninbf  jjpkfghn
kcboplkf  jjngn+j+  pofogono  cko+idhf  bkhklehf
iili+pgf  mjhjfmkm  ddoonnjf  m+fnonjk  mbcncipd
jbmilm+n  ppjfkmkc  on+dmbio  dllln+dn  hmjnfomo
 cmeifbfj  elgihebk  kgpljhnf  id+bnehj  +oiekijk
mhnp++jk  +emdk+ki  mjhcdcog  ++e+iloo
llmmkdbp  pmkpnoob
```

Réception en provenance de la carte 2 — Chiffré reçu déchiffré

```
00? ^YJ????Rc ???? fC! ???? jSz? 8,20 1739
1739 1735 1749 1754 1754 1754 1759 1764
1759 1754 1764 1764 1754 1759 1764 1764
1759 1759 1769 1764 1754 1764
```

Réception en provenance de la carte 2 — Chiffré reçu

```
000000 ndbojjmk  kiingmpe  fbjffjhb  ecbkgp+c
ijoh+oko  +jklgnic  +ipkddj+    ll+mghff  ldldpgoc
ojblompj  njekifhj  pkgcgeme  kebimmkn  pjinlnbn
j+felf+l  ++ibpjbi  mdlcdhbk  llneip+k  dndhjpjh
ll+mghfc  idjjigml  pgjld+hh  hidjk+ki  ojlhmgfp
```

Réception en provenance de la carte 2 — Chiffré reçu déchiffré

```
00? ?i?? ?B6? G?Qv ?P_J S?39 174   ? ??Zl
{??5 ??8? ;?!? o!59 1759 1759 1759 1759
1764 1754 176   ? ?0?Q E?x? ?F?? ???i ??54
```

Envoi à la carte 2 — Chiffré à envoyer

```
nmdfbl  gnkmjb  mmldoh  ifpc+n  kiflnf  gffmkf
jpmlgc  kbeg+j  mjedhm  igigfg  lilljb  cgllhc  idfil+
 gnjdpo  elkgop  fkpjbb  elgcnk  ecfond
bkpdbh  bolpgj  kinjhk  objfoc  gbimk+  iljcin
o+pghh  modoel  gpidjk  mdgkgb  jfglii  eijkop
bmeem+  kkeocl  oghgmg  lifef+  hnlfbe  bieejo
go+ibi  icclfd  mpbkkm
```

Envoi à la carte 2 — Clair à chiffrer

```
522 180 493 322 434 190 229 219 478 224
312 464 239 136 141 141 205 229 454 493
263 195 522 327 317 283 522 210 420 522
371 459 493 278 244 503 449 425
```

Figure 7: Supervisor

This new structure allows to circumvent the problem raised by the weakness of T–functions. Then, a realistic example with detailed specifications has been proposed. As usual in cryptography, we can expect that eavesdroppers will try to break this cryptosystem. Time will tell.

Acknowledgement

This work was supported by Research Grants ANR-13-INSE-0005-01 from the Agence Nationale de la Recherche in France

Appendix

Algorithm 1 Digraph Generation Algorithm

```
Require: $n, na, r$
Ensure: $G$
  indexTab = range($n+1$)
  $\beta$ = binomial($n+1, 2$) + $r$
  if $r > n$ then
    print 'KO! impossible to construct the graph: $r > n$
    return False
  end if
  if ($na < n$) or ($na > \beta$) then
    print 'KO! impossible to construct the graph: number of arc not valid
    return False
  end if
  $\alpha = 0$
  $G$ = DiGraph(loops=True)
  prevertex = G.add_vertex()
  for $i$ in range($r$) do
    vertex = indexTab[$i$]
    G.add_edge(vertex,indexTab[$i+1$])
    G.add_edge(vertex,vertex,$\alpha$)
    $\alpha + = 1$
    for  $j$ in range($i$) do
      G.add_edge(vertex,indexTab[$j$],$\alpha$)
      $\alpha + = 1$
    end for
  end for
  for $i$ in range($r-2, r$) do
    vertex = indexTab[$i$]
    for $j$ in range($i+2, n$) do
      G.add_edge(vertex,indexTab[$j$],$\alpha$)
      $\alpha$+=1
    end for
  end for
  for $i$ in range($r, n$) do
    vertex = indexTab[$i$]
    if $i < (n-1)$ then
      G.add_edge(vertex,indexTab[$i+1$])
    end if
    for $j$ in range($r-1$)+range($i+2, n$) do
      G.add_edge(vertex, indexTab[j],$\alpha$)
      $\alpha$+=1
    end for
  end for
  rmvArc_lab = random.sample(range($\beta - n$),$\beta - na$)
  rmvArc_list = []
  for ($u, v, l$) in G.edges() do
    if $l$ in rmvArc_lab then
      rmvArc_list.append(($u, v$))
    end if
    G.delete_edges(rmvArc_list)
  end for
  print 'Graph generated succesfully'
  return G
```

[1] AES, N. (November 26, 2001). FIPS publication 197 - advanced encryption standard.

[2] Arnault, F., Berger, T. P., Minier, M., and Pousse, B. (2011). Revisiting lfsrs for cryptographic applications. *IEEE Transactions on Information Theory*, **57**(12), 8095–8113.

[3] Berger, T. P. and Minier, M. (2015). Some results using the matrix methods on impossible, integral and zero-correlation distinguishers for feistel-like ciphers. In A. Biryukov and V. Goyal, editors, *Progress in Cryptology - INDOCRYPT 2015 - 16th International Conference on Cryptology in India, Bangalore, India, December 6-9, 2015, Proceedings*, volume 9462 of *Lecture Notes in Computer Science*, pages 180–197. Springer.

[4] Blondel, V. D. and Tsitsiklis, J. N. (1997). When is a pair of matrices mortal? *Information Processing Letters*, **63**, 283–286.

[5] Boukhobza, T. (2010). Partial state and input observability recovering by additional sensor implementation: a graph-theoretic approach. *International Journal of Systems Science*, **41**(11), 1281 – 1291.

[6] Bournez, O. and Branicky, M. (2002). The mortality problem for matrices of low dimensions. *Theory of Computing Systems*, **35**(4), 433–448.

[7] Courtois, N., Klimov, A., Patarin, J., and Shamir, A. (2000). Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, pages 392–407.

[8] Daemen, J. and K., P. (2005). The self-synchronizing stream cipher mosquito: estream documentation, version 2. Technical report, e-Stream Project. Available at: www.ecrypt.eu.org/stream/p3ciphers/mosquito/mosquito.pdf.

[9] Daemen, J. and Kitsos, P. (2005a). The self-synchronizing stream cipher mosquito: estream documentation, version 2. *eSTREAM, ECRYPT Stream Cipher Project, Report 2005/018*. Available online at http://www.ecrypt.eu.org/stream.

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

[11] Daemen, J. and Kitsos, P. (2008). The self-synchronizing stream cipher moustique. In *New Stream Cipher Designs - The eSTREAM Finalists*, pages 210–223. Springer-Verlag Berlin Heidelberg.

[12] Daemen, J., Govaerts, R., and Vandewalle, J. (1992). On the design of high speed self-synchronizing stream ciphers. In *Proc. of the ICCS/ISITA'92 conference*, volume 1, pages 279–283, Singapore.

[13] Dravie, B., Guillot, P., and Millérioux, G. (2017). Design of self-synchronizing stream ciphers: A new control-theoretical paradigm. In *IFAC World Congress, (IFAC 2017)*, Toulouse, France.

[14] Dubi, C. (2009). An algorithmic approach to simultaneous triangularization. *Linear Algebra and its Applications*, **430**(11-12), 2975 – 2981.

[15] Fliess, M., Levine, J., Martin, P., and Rouchon, P. (1995). Flatness and defect of non-linear systems: introductory theory and examples. *Int. Jour. of Control*, **61**(6), 1327–1361.

[16] Hawkes, P., Paddon, M., Rose, G. G., and 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.

[17] Hellman, M. E. (1980). A cryptanalytic time-memory trade-off. *IEEE Trans. Information Theory*, **26**(4), 401–406.

[18] Kasper, E., Rijmen, V., E.Bjorstad, Rechberger, C., Robshaw, M., and Sekar, G. (2004). Correlated keystreams in moustique. Technical report, ESTREAM Project.

[19] Maurer, U. M. (1991). New approaches to the design of self-synchronizing stream cipher. *Advance in Cryptography, In Proc. Eurocrypt '91, Lecture Notes in Computer Science*, pages 548–471.

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

[21] Millérioux, G. and Boukhobza, T. (2015). Characterization of flat outputs for LPV discrete-time systems: a graph-oriented approach. In *54th Conference on Decision and Control (CDC 2015)*, Osaka, Japan.

[22] Molland, H. and Helleseth, T. (2006). Linear properties in t-functions. *IEEE Trans. Information Theory*, **52**(11), 5151–5157.

[23] Paterson, M. S. (1970). Unsolvability in 3×3 matrices. *Studies in Applied Mathematics*, **49**(1), 105–107.

[24] Sarkar, P. (2003). Hiji-bij-bij: A new stream cipher with a self-synchronizing mode of operation. *IACR Cryptology ePrint Archive*, **2003**, 14.

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

[26] Strogatz, S. H. (2003). *SYNC: The Emerging Science of Spontaneous Order*. Penguin Group.