Control, synchronization and encryption

Jeremy Parriaux

▶ **To cite this version:**

Jeremy Parriaux. Control, synchronization and encryption. Optimization and Control [math.OC]. Université de Lorraine, 2012. English. NNT : 2012LORR0129 . tel-01749323v2

HAL Id: tel-01749323
https://theses.hal.science/tel-01749323v2

Submitted on 6 Nov 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Département de formation doctorale en automatique **École doctorale IAEM Lorraine**

Contrôle, synchronisation et chiffrement

THÈSE

présentée et soutenue publiquement le 03/10/2012

pour l'obtention du

Doctorat de l'Université de Lorraine

(spécialité automatique)

par

Jérémy Parriaux

Composition du jury

Président :	Taha Boukhobza	Professeur Université de Lorraine
Rapporteurs :	Jean-Pierre Barbot	Professeur ENSEA, Cergy-Pontoise
	Thierry Berger	Professeur Université de Limoges
Examinateurs :	Joan Daemen	Principal Cryptographer ST microelectronics
	Antoine Girard	Maître de Conférences Université Joseph Fourier
	Marc Mouffron	Head of Cryptography and Wireless Security, Senior expert CASSIDIAN, Paris
Co-directeur :	Philippe Guillot	Maître de Conférences Université Paris 8
Directeur :	Gilles Millérioux	Professeur Université de Lorraine

Centre de Recherche en Automatique de Nancy
UMR 7039 Université de Lorraine – CNRS

2, rue Jean Lamour F-54519 Vandœuvre-lès-Nancy
Tél. +33 (0)383 685 000 — Fax +33 (0)383 685 001

Mis en page avec la classe thloria.

Remerciements

Avant d'aborder le contenu de cette thèse, je tiens à remercier tout ceux qui de près ou de loin m'ont permis de réaliser ces travaux.

J'adresse mes remerciements à Gilles Millérioux et à Philippe Guillot pour m'avoir encadré durant ces trois années. Le temps qu'ils m'ont accordé a largement contribué au bon déroulement de cette thése.

Je remercie également Éric Garrido et Hieu Phan de s'être intéressé à mon travail ainsi que de m'avoir suggéré des pistes de réflexion.

Je désire remercier le jury qui a accepté d'évaluer cette thèse.

Je remercie également Phuoc V., Matthieu W., Hugo M. et Florian S. pour leurs contributions au développement de la plateforme FPGA.

Je remercie les secrétaires Sylvie F., Carole C., Mélanie T., Sabrina M. et Christelle L. pour leur serviabilité.

Je souhaite également remercier Éric G., Jean-Luc N., Jean-Marie O., Jean-Michel R. et Marion G. pour leurs conseils pédagogiques.

Je remercie l'ensemble des membres du CRAN que j'ai cotoyé durant ces années Boyi N., Fengwei C., Floriane A., Gérard B., Hugues G., Julien S., Meriem H., Tatiana M. et Vincent L. et qui ont contribué à la bonne ambiance des pauses café.

Je remercie également ma famille pour m'avoir permis de faire mes études et soulager de quelques contraintes de la vie quotidienne.

Contents

Abbreviations

AES	advanced encryption standard
ANF	algebraic normal form
CBC	cipher block chaining
CCA	chosen ciphertext attack
CFB	cipher feedback
CPA	chosen plaintext attack
CTR	counter mode
DES	data encryption standard
ECB	electronic code book
FPGA	field programmable gate arrays
IV	initialization vector
LUT	lookup table
LPV	linear parameter varying
MIMO	multiple inputs multiple outputs
MISO	multiple inputs single output
NIST	national institute of standards and technology
NNF	numerical normal form
OFB	output feedback
SISO	single input single output
SIMO	single input multiple outputs
SSSC	self-synchronizing stream cipher

Notations

This thesis discusses a topic related to different communities. Each community has its own customs when it comes to the notations and they were not always compatible. The notations were tried to be kept as natural as possible for the different communities. The author hopes that they will not confuse the reader.

$\mathbf{1}_n$	$n \times n$ dimensional identity matrix
$\mathbf{0}_{n,m}$	$n \times m$ dimensional zero matrix
$\mathbf{0}_n$	$\mathbf{0}_{n,n}$
$\mathbb{N}$	the set of non-negative integer numbers
$\mathbb{Z}$	the ring of integer numbers
$\mathbb{R}$	the field of real numbers
$\mathbb{C}$	the field of complex numbers
$\mathbb{F}$	a field
$\mathbb{F}_2$	the two-element field
$\mathbb{F}^n$	the n dimensional vector space over the field $\mathbb{F}$
$\mathbb{F}^{n*}$	the set $\mathbb{F}_2$ deprived from the null element
$\overline{\mathbb{F}}$	algebraic closure of the field $\mathbb{F}$
$\mathbb{A}^\star$	free monoid over the set $\mathbb{A}$
$\oplus$	sum modulo 2
k	discrete time variable, $k \in \mathbb{Z}$
m_k	plaintext at time k
$\hat{m}_k$	decrypted message at time k
c_k	ciphertext at time k
x^i	component i of vector x
$(x)^i$	element $x \in \mathbb{F}$ raised to the power i
$u \cdot x$	for $u, x \in \mathbb{F}_2^n, u \cdot x = u^0 x^0 \oplus \cdots \oplus u^{n-1} x^{n-1}$ is the dot product of u and x
$u^{\hat{}\, x}$	for $u, x \in \mathbb{F}_2^n$, $u^{\hat{}\, x} = (u^0)^{x^0} (u^1)^{x^1} \cdots (u^{n-1})^{x^{n-1}}$
$\mathrm{supp}(x)$	support of the binary vector x
$\mathrm{hw}(x)$	Hamming weight of the vector x
$\mathfrak{f}$	next-state function of a dynamical system
$\mathfrak{g}$	output function of a dynamical system
$f \otimes g$	correlation of the function f with the function g
$\mathfrak{s}_n$	shift function with n cells
$\mathfrak{s}_{n,n_s}$	shift function with n cells of symbols of $\mathbb{F}_2^{n_s}$
$\mathfrak{l}(a, b)$	length of the shortest path from the vertex a to the vertex b when it exists
$\log_2$	logarithm to the base 2
$\Pr[X = x]$	probability that the random variable X is equal to the value x
$\mathrm{Card}\,(I)$	cardinal of the set I
$f \circ g$	composition of the function g with the function f
n_i	number of inputs of a dynamical system
n_o	number of outputs of a dynamical system
$(a)_{k_1}^{k_2}$	sequence of elements a_k from time $k = k_1$ and $k = k_2$

$M_{i,j}$	coefficient at row i and column j of the matrix M
$M^a \vert M^b$	horizontal concatenation of the two matrices (or vectors) M^a and M^b
${}^tM$	transpose of the matrix M
$\mathrm{rank}(M)$	rank of the matrix M
$\mathrm{tr}(M)$	trace of the matrix M
$\det(M)$	determinant of the matrix M
$M^\dagger$	Moore-Penrose generalized inverse of the matrix M
$M^\bullet$	matrix M in which the first row and column have been removed
$\mathcal{W}^f$	Walsh matrix of the vectorial Boolean function f
$\mathcal{R}^f$	correlation matrix of the vectorial Boolean function f
$\mathcal{A}^f$	algebraic matrix of the vectorial Boolean function f
$\mathcal{N}^f$	numerical matrix of the vectorial Boolean function f
$\mathcal{F}^f$	adjacency matrix of the vectorial Boolean function f
$\mathscr{C}^f$	adjacency matrix of the connection graph of the vectorial Boolean function f
H	Hadamard matrix
A	dynamical matrix
B	input matrix
C	output matrix
D	direct transfer matrix
$\mathcal{J}$	the set of possible modes of a switched linear system
r	inherent delay
R	relative degree
k_f	synchronization delay
σ	commutation rule
n_s	number of bits of a symbol in a stream cipher
n_b	number of bits in a block of a block cipher
$u \preceq x$	$\mathrm{supp}(u) \subseteq \mathrm{supp}(x)$
f_χ	sign function of the Boolean function f
$\widehat{f}$	Fourier transform of the Boolean function f
$\widehat{f_\chi}$	Walsh transform of the Boolean function f
$\widetilde{f}$	algebraic (or numerical depending on the context) transform of the Boolean function f
$f^{\circ a}$	composition a times of the function f is some sense which is context dependent
$\mathcal{E}$	dynamical system that plays the role of the encryptor
$\mathcal{D}$	dynamical system that plays the role of the decryptor
$\mathcal{B}$	block cipher
$I_f(S)$	influence of the set of variables $\{x^i,\, i \in S\}$ over the function $x \longmapsto f(x)$
$\mathrm{Im}(f)$	range of the function f
d	distance
$f^{\to c}$	$(n+n_s, n)$–function f iterated with the sequence (c) of symbols of $\mathbb{F}_2^{n_s}$

Chapter 1

Introduction

Synchronization is a process through which a set of interconnected entities reach a similar behaviour. This phenomenon is ubiquitous in nature, whether it concerns living things or not. A striking example is the synchronization of the flash of fireflies. These small animals emit flashes of light and they tend to do it altogether at the same time as the night falls. Another example is the synchronization of pacemaker cells in the heart. Each of these cells emits current spikes responsible for the heart contraction. One isolated cell cannot do anything. The heart beats only because of the synchronized action of all the cells. In the nonliving world, synchronization is important for instance in lasers. The light emitted by these devices is obtained by synchronizing all the photons together. A common point to all these objects is that they are all dynamical entities evolving according to the time. The evolution of these entities may also be influenced by the environment. We refer to such entities by the name "dynamical systems". An interesting popular book that discusses the synchronization issue is [1]. It is shown that this phenomenon is everywhere: scientists from many fields like biology, physics, and even engineering are concerned with this phenomenon. Secure communications, more specifically cryptography, is an engineering field that requires, in special contexts, synchronization between the emitter and the receiver.

Cryptography is the discipline that covers all the techniques to communicate in a secure way. It is of first importance as the modern world involves thousands of communicating devices on which many aspects of our daily life rely on. Among the concerns of cryptography, an important one, called encryption, consists in transforming a meaningful message into a meaningless one during the time it is conveyed from the emitter to the receiver. Encryption has to be achieved in such a way that, only the receiver can transform the meaningless message into a meaningful message. The algorithm that defines the transformation to be applied is called a cipher. There are several classes of ciphers defined in the literature. One of them is the class of self-synchronizing stream ciphers. Such ciphers will be a central purpose of this thesis. According to the terminology, the accomplishment of the synchronization mechanism is an intrinsic property of self-synchronizing stream ciphers. Such a property makes their use easier in some specific situations. In this thesis, the emitter and the receiver are modelled by dynamical systems that process the messages to be transmitted. In order to properly work, the receiver needs to be in a consistent state with respect to the emitter and hence, requires synchronization. With a non self-synchronizing system, a specific supervising setup should be implemented to ensure the consistency of the states. In particular, it should be required that a protocol between the receiver and the emitter ensures synchronization. Self-synchronizing systems get intrinsically this ability. This might be interesting when the receiver is a low power device and is not powerful enough to communicate with the emitter to ensure synchronization. Another interest of this kind of cipher is that it can be used to render safe and secure a system simply by adding the cipher on top of the system, hence avoiding to redesigning it. Besides, for some low-resource devices, the overhead inherent to the use of error correcting codes and to any synchronization protocol cannot be afforded. Self-synchronizing can handle such a problem.

For the control community, it is worth putting the present work in the context of the so-called "chaotic cryptography". The idea of chaotic cryptography was publicly proposed in 1993. It takes advantage

of the complex behaviour of chaotic dynamical systems to "hide" or "mask" information. An overview of the different methods devised over the years can be found in many books [2, 3, 4, 5, 6]. Indeed, chaotic behaviour can be distinguished by its extreme sensitivity to initial conditions, leading to long-term unpredictability. Moreover, signals resulting from chaotic dynamics are broadband and present random-like statistical properties, albeit they are generated by deterministic systems. All this explains why there is likely a connection between the random-looking behaviour exhibited by chaotic systems and the important cryptographic properties of confusion and diffusion established by Shannon [7]. Consequently, it motivates the use of chaotic systems for secure communications, even though the terminology "secure" is sometimes questionable. It turns out that very few works have really established the connection between standard and chaos-based encryption algorithms, but see [8, 9, 10] for some interesting exceptions.
There are basically two classes of chaotic cryptosystems. The first one amounts to numerically computing a great number of iterations of a discrete chaotic system, using the message as initial data (see [11, 12] and references therein). This is basically also the strategy in [13, 14], where periodic approximations of chaotic automorphisms are used to define substitutions (so-called S-boxes) resistant to linear and differential cryptanalysis. The second class, amounts to scrambling a message with a chaotic dynamics. Various cryptosystems, corresponding to distinct ways of hiding messages, have drawn the attention of researchers over the years. The most important schemes obeying such a principle are additive masking, chaotic switching, discrete or continuous parameter modulation, two-channel transmission, and message-embedding. Additive masking was first suggested in [15] and [16]. Chaotic switching is also referred to as chaotic modulation or chaos shift keying. Such a technique has been mostly proposed in the digital communications context. A description can be found in [17], even though the method was proposed a couple of years before, say, in 1993 [18]. Basically, two kinds of parameter modulations can be distinguished: the discrete [19, 18] and the continuous one [20, 21, 22, 23]. The two-channel transmission approach has been proposed, for example, in [24, 25]. The message-embedded technique is given different names in the literature: embedding [26, 27], non-autonomous modulation [28] or direct chaotic modulation [29]. A slightly different method derived from the message-embedding is the hybrid message-embedding. It was first proposed in [30] but the terminology "hybrid" has been really introduced in [31].

This thesis mixes, in an original way, two research fields: automatic control and cryptography. Regarding cryptography, a special emphasis will be placed on self-synchronizing stream ciphers. However, the synchronization mechanism studied here is general and can be applied to other contexts. The novelty of the approach compared to other automatic control attempts at designing self-synchronizing stream ciphers is that, we do not make any connection with chaotic systems. Indeed, although control theoretical results are established in the first part of the manuscript within the field of real numbers, it will be shown that they still hold over finite fields. And yet, they will be precisely used to come up with self-synchronizing systems involving finite state dynamical systems. As a result, chaos is not appropriate in our context. The goal of this thesis is to understand the synchronizing mechanism in a unidirectional point-to-point setup. We focus more on this mechanism than on the security even though elements on this topic are given at the end of the manuscript. The entities we will consider are called either finite state automata or discrete dynamical systems depending on the context. The book [32] proposes a general framework for the study of discrete dynamical systems, their interactions and the resulting effect over their dynamics. However, even though the proposed framework is very convenient, it is very conservative and requires to make too many assumptions to properly characterize synchronization. The contribution of this thesis should be considered as complementary.

The layout is the following: Chapter 2 brings out the connection between automatic control and cryptography. Recalls about cryptography are done. The central notion of flatness is introduced to cope with synchronization in a unified framework both from control theory and cryptography points of view. Chapter 3 discusses a special class of systems encountered in automatic control called switched linear systems. The study of the flatness property for this class of systems is performed and the concept of nilpotent semigroups is emphasized for its characterization. An extension of the results to Linear Parameter-Varying (LPV) systems is done as well. Next, we focus on the Boolean context. Chapter 4 gives the required background on Boolean functions. It is recalled existing matrix representations of functions and new ones are proposed. The eigenanalysis of these matrix representations is performed. Chapter 5 addresses differ-

ent self-synchronization characterizations. An important one rests on a nilpotent semigroups approach. Chapter 6 is devoted to the construction of suitable functions for the design of self-synchronizing stream ciphers. The security aspect inherent to any cryptographic system is discussed along with implementations issues.

The published papers related to this work are listed below:

1. J. Parriaux and G. Millérioux, *Synchronization of hybrid systems for secure multimedia streaming*. 7th IEEE IET International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2010, Newcastle, United Kingdom

2. J. Parriaux, P. Guillot and G. Millérioux, *Synchronization of Boolean Dynamical Systems: a Spectral Characterization*. SEquence and Their Applications, SETA 2010, Paris, France

3. J. Parriaux, P. Guillot and G. Millérioux, *Towards a spectral approach for the design of self-synchronizing stream ciphers*. Yet Another Conference in Cryptography, YACC 2010, Porquerolles, France

4. J. Parriaux, P. Guillot and G. Millérioux, *A Spectral Approach for characterizing the self-synchronization of stream ciphers*. Symmetric Key Encryption Workshop, SKEW 2011, Copenhagen, Denmark

5. J. Parriaux, P. Guillot and G. Millérioux, *Towards a spectral approach for the design of self-synchronizing stream ciphers*. Cryptography and Communications, (3), 3:p259-274, 2011

6. J. Parriaux and G. Millérioux, *A constructive approach for the design of self-synchronizing dynamical systems: an application to communications*. IFAC World Congress, IFAC 2011, Milano, Italy

7. J. Parriaux and G. Millérioux, *Designing self-synchronizing switched linear systems: an application to communications*. Nonlinear Analysis: Hybrid Systems, (7)1:68-79, 2013

8. J. Parriaux and G. Millérioux, *Nilpotent semigroups for the characterization of flat outputs of discrete-time switched linear and LPV systems*. Conference on Decision and Control, CDC 2012, Maui, Hawaii

Chapter 2

Dynamical systems and stream ciphers

The aim of this chapter is to bridge the gap between automatic control and cryptography. In Section 2.1, the definition and properties of dynamical systems are recalled. Section 2.2 presents the master-slave setup which involves two or more dynamical systems connected together in a unidirectional way. This configuration is central in this thesis. A brief overview of cryptography is given in Section 2.3, including the description of the different classes of ciphers. Then, in Section 2.4, the state of the art on self-synchronizing stream ciphers is carried out. Finally, Section 2.5 shows how the property called *flatness*, borrowed from control theory, is related to self-synchronizing stream ciphers.

2.1 Dynamical systems

Throughout this thesis, *dynamical systems* are discussed. They are often used in control theory to model the evolution of systems. When the time is a continuous quantity, they are referred to as continuous-time dynamical systems. Alternatively, when the time is a discrete quantity, they are referred to as discrete-time dynamical systems. A formal definition is provided below.

Definition 2.1.1 (Discrete-time dynamical system)**.** A n–dimensional discrete-time dynamical system is a five–tuple $(\mathbb{A}, \mathbb{B}, \mathbb{X}^n, \mathfrak{f}, \mathfrak{g})$ where

- $\mathbb{A}$ is the input set that is, the set of input values u_k;
- $\mathbb{B}$ is the output set that is, the set of output values y_k;
- $\mathbb{X}^n$ is the set of state vectors x_k, also called state space;
- $\mathfrak{f} : \mathbb{X}^n \times \mathbb{A} \longrightarrow \mathbb{X}^n$ is the (next) state transition function;
- $\mathfrak{g} : \mathbb{X}^n \times \mathbb{A} \longrightarrow \mathbb{B}$ is the output function.

The subscript k denotes the discrete-time. Let us notice that, according to the context, the subscript will be specified or not. The sets $\mathbb{A}, \mathbb{B}$ and $\mathbb{X}$ being given, the dynamical system is commonly written

$$\begin{cases} x_{k+1} &= \mathfrak{f}(x_k, u_k) \\ y_k &= \mathfrak{g}(x_k, u_k) \end{cases} \tag{2.1}$$

When the set $\mathbb{X}$ is finite, the dynamical system is said to be finite. The functions $\mathfrak{f}$ and $\mathfrak{g}$ may sometimes be parametrized by some quantity κ. In such a case, they are respectively written $\mathfrak{f}[\kappa]$ and $\mathfrak{g}[\kappa]$. Typically, in cryptography, it corresponds to the secret key.

The ith coordinate of the vector x_k is denoted by x_k^i. Similarly, the coordinate function of $\mathfrak{f}$ that updates the component i of x_k is denoted by $\mathfrak{f}^i$.

A dynamical system is usually intended to model a physical system whose instantaneous state is represented by its state vector x_k and its time evolution is described by the function $\mathfrak{f}$. The components of the state vector may correspond to some physical quantities, for instance, the velocity or the position. It may also represent a current flowing through the wires of an electrical circuit. The state x_k evolves in time, depending on the past state (explaining the terminology) and possibly on some exogenous quantities corresponding to the input u_k. It may happen that the components of the state vector are not directly accessible. The available quantities are the measurements y_k which are a function of the internal state x_k and possibly of the input u_k. This explains the introduction of the function $\mathfrak{g}$. When the system is intended to represent a physical phenomenon, quantities are usually real numbers and then: $\mathbb{A} = \mathbb{R}^{n_i}$, $\mathbb{B} = \mathbb{R}^{n_o}$, $\mathbb{X} = \mathbb{R}$ where n_i denotes the number of inputs and n_o the number of outputs. When $n_i = n_o = 1$, the system is said to be *Single Input Single Output* (SISO for short). When both the number of inputs and the number of outputs are greater than one, the system is said to be *Multiple Inputs Multiple Outputs* (MIMO for short). SIMO and MISO systems can also be defined. Finally, when the number of inputs equals the number of outputs, the system is said to be square. The behaviour of the dynamics that is, the evolution of the vector x, can be investigated regardless of the nature of the system: physical, biological, economical, etc. The vector x evolves in time in the space $\mathbb{X}^n$ and describes a trajectory (also called an *orbit*). Dynamical systems can be divided in two categories: autonomous and non autonomous. Their specificities are detailed in the discrete-time case in the following sections. Let us notice that in the remaining part of this manuscript, we will exclusively focus on discrete-time dynamical systems.

2.1.1 Autonomous dynamical system

In this section, we consider a dynamical system with no input, its dynamics is described by

$$x_{k+1} = \mathfrak{f}(x_k) \tag{2.2}$$

It is called an autonomous system. More formally, an *autonomous system* is a dynamical system for which $\mathfrak{f}$ does not explicitly depend on any exogenous quantity that is, any quantity external to the system, including the time. The trajectory of x in the state space $\mathbb{X}^n$ is completely determined by the initial condition x_0. In general, the analytic solution of (2.2) in terms of known elementary and transcendental functions cannot be determined but, we are often only interested in the steady-state behaviours. The *steady-state* behaviour of a system is the behaviour obtained after a possible transient time. The trajectory of the state vector may reach more or less complex steady-states which can coexist, usual ones are listed below.

A stationary point, also called equilibrium point or fixed point is a steady-state that satisfy $x_{k+1} = x_k = x^*$.

A periodic orbit is a cycle of finite order K. Every points belonging to the cycle obeys $x_{k+K} = x_k$ and $x_{k+K'} \neq x_k$ for $K' < K$. Points of periodic orbits are called periodic points.

Another trajectory is the chaotic orbit. First, we give a brief history of *chaos*. An interesting popular book on that topic is [33]. The discovery of chaos is originally attributed to Poincaré while he was studying the three-body problem [34] in 1892. He was interested in the solar system and tried to predict the motion of three bodies submitted to their own gravitational force. He found out that, the predictions were very sensitive to the initial conditions. This is one of the features of chaos. However, the notion of sensitivity to initial conditions did not catch the attention of the scientific community until 1963 when Lorenz experienced this effect in weather simulations [35]. Since then, scientists have started to pay a special attention to this phenomenon. The word *chaos* has been introduced for the first time in 1975 in the paper [36].

Now, let us give a formal characterization of chaos. Let $(I \subseteq \mathbb{X}^n, \mathrm{d})$ denote a metric space (d is a distance) and consider the nonlinear continuous function defining the map:

$$\begin{array}{rccc} \mathfrak{f}: & I & \longrightarrow & I \\ & x_k & \longmapsto & x_{k+1} = \mathfrak{f}(x_k) \end{array} \tag{2.3}$$

Before providing a strict definition of chaos, some preliminary definitions are required. The kth order iterated function of $\mathfrak{f}$ is defined as follows.

Definition 2.1.2 (kth order iterated next-state function)**.** Let $\mathfrak{f}$ be a function from $\mathbb{X}^n$ into itself, then, its kth order iterated function is

$$\mathfrak{f}^{\circ k}(x) = \underbrace{\mathfrak{f} \circ \cdots \circ \mathfrak{f}}_{k \text{ times}}(x) \tag{2.4}$$

Definition 2.1.3 (Sensitive dependence on initial condition)**.** The function $\mathfrak{f} : I \longrightarrow I$ is said to have the property of sensitive dependence on initial conditions or to be sensitive to initial conditions if there exists $\varepsilon > 0$ such that, for any $x_0 \in I$ and any $\varepsilon_0 > 0$, there exists a point $x'_0 \in I$ where $d(x_0, x'_0) < \varepsilon_0$ and $k \in \mathbb{N}$ so that $\mathrm{d}(\mathfrak{f}^{\circ k}(x_0), \mathfrak{f}^{\circ k}(x'_0)) > \varepsilon$.

Definition 2.1.4 (Topologically transitive)**.** The function $\mathfrak{f} : I \longrightarrow I$ is said to be topologically transitive if, U and V being non empty open sets in I, there exists a state $x_0 \in U$ and an index $j \in \mathbb{N}$ such that $\mathfrak{f}^{\circ j}(x_0) \in V$ or equivalently, there exists an index $j \in \mathbb{N}$ such that $\mathfrak{f}^{\circ j}(U) \cap V \neq \emptyset$.

Definition 2.1.5 (Dense set)**.** Let X be a set and let Y be a subset of X. The set Y is dense in X if for any point $x \in X$ and any $\varepsilon > 0$ there is a point $y \in Y$ such that $d(x, y) < \varepsilon$.

We are now in position to recall the definition of chaos stated by Devaney in [37].

Definition 2.1.6 (Chaotic map)**.** A continuous function $\mathfrak{f} : I \longrightarrow I$ is said to be a chaotic map or to define a chaotic dynamical system if:

1. The function $\mathfrak{f}$ is sensitive to initial conditions;
2. The function $\mathfrak{f}$ is topologically transitive;
3. The set of periodic points of $\mathfrak{f}$ is dense in I.

When the set $\mathbb{X}$ is finite, the only possible steady-states are either stationary points or periodic orbits.

2.1.2 Non autonomous dynamical system

For non autonomous dynamical systems, we can consider an external input u. It may correspond to an external perturbation or to a control that aims at driving the system to reach a prescribed dynamics. We denote a sequence of inputs $u_{k_1}, \ldots, u_{k_2}$ by $(u)_{k_1}^{k_2}$. If k_1 and k_2 are unspecified, we merely write (u) or u if there is no possible confusion. Hence, a sequence of state vectors $x_{k_1}, \ldots, x_{k_2}$ is denoted by $(x)_{k_1}^{k_2}$. Note that the sequence $(u)_{k_1}^{k_2}$ is an element of $\mathbb{A}^{k_2-k_1+1}$. The set of all possible input sequences is denoted by $\mathbb{A}^\star$, the free monoid over the set $\mathbb{A}$. Considering that the initial state x_0 of (2.1) is known as well as the input sequence $(u)_0^{k-1}$ then, the sequence of states $(x)_0^k$ is completely defined and the state x_k is given by the kth order iterated function defined as follows.

Definition 2.1.7 (kth order iterated next-state function)**.** Let $\mathfrak{f}$ be a function from $\mathbb{X}^n \times \mathbb{A}$ to $\mathbb{X}^n$. Its kth order iterated function is defined by

$$\mathfrak{f}^{\circ k}(x_0, (u)_0^{k-1}) = \begin{cases} x_0 & \text{if } k = 0 \\ \mathfrak{f}(\cdots \mathfrak{f}(\mathfrak{f}(x_0, u_0), u_1) \cdots, u_{k-1}) & \text{if } k > 0 \end{cases} \tag{2.5}$$

Note that, despite identical notations in Definition 2.1.2 and Definition 2.1.7, there is no possible confusion. The first one involves functions with an argument in the set $\mathbb{X}^n$ while the second one involves functions with an argument in the set $\mathbb{X}^n \times \mathbb{A}^k$. We define a similar iterated function for the output function as follows.

Definition 2.1.8 (kth order iterated output function)**.** Let $\mathfrak{g}$ be a an output function, that is, a function from $\mathbb{X}^n \times \mathbb{A}$ to $\mathbb{B}$. Its kth order iterated function is denoted by $\mathfrak{g}^{\circ k}$ and is defined by

$$\mathfrak{g}^{\circ k}(x_0, (u)_0^k) = \mathfrak{g}(\mathfrak{f}^{\circ k}(x_0, (u)_0^{k-1}), u_k) \tag{2.6}$$

A special quantity characterizing non autonomous dynamical systems and useful for the purposes studied later on is the *relative degree*.

Definition 2.1.9 (Relative degree)**.** The relative degree of a SISO dynamical system is the minimum number of iterations so that the output explicitly depends on the input. The relative degree will be denoted by R.

Then, we have that

- if $R = 0$, there exists $x_k \in \mathbb{X}^n$ and two distinct input symbols u_k and u'_k at time k that lead to different values of the output y_k;
- if $R > 0$, then for $i < R$, the iterated output function $\mathfrak{g}^{\circ i}$ depends only on x_k for $i \geq R$, it depends on both x_k and on the sequence of input symbols $(u)_k^{k+i-R}$. In particular, for $i = R$, the iterated output function depends on both x_k and u_k.

The function $\mathfrak{g}^{\circ R}$ may be considered as a function on $\mathbb{X}^n \times \mathbb{A}$ and thereby, for $R \geq 0$,

$$y_{k+R} = \mathfrak{g}^{\circ R}(x_k, u_k) \tag{2.7}$$

We do not consider here the case when R depends on k.
There are several ways to connect dynamical systems together. One which is central in this thesis is called *master-slave setup*.

2.2 Self-synchronization in a master-slave setup

2.2.1 Master-slave configuration

The master-slave setup involves two dynamical systems. The first one, called the *master*, delivers a time varying signal which is used to feed the input of the second one, called the *slave*. The setup is illustrated in Figure 2.1 and the corresponding equations are

$$\begin{cases} x_{k+1} &= \mathfrak{f}(x_k, u_k) \\ y_k &= \mathfrak{g}(x_k, u_k) \end{cases} \quad \text{(Master equation)} \tag{2.8a}$$

$$\begin{cases} \hat{x}_{k+1} &= \hat{\mathfrak{f}}(\hat{x}_k, y_k) \\ \hat{u}_k &= \hat{\mathfrak{g}}(\hat{x}_k, y_k) \end{cases} \quad \text{(Slave equation)} \tag{2.8b}$$

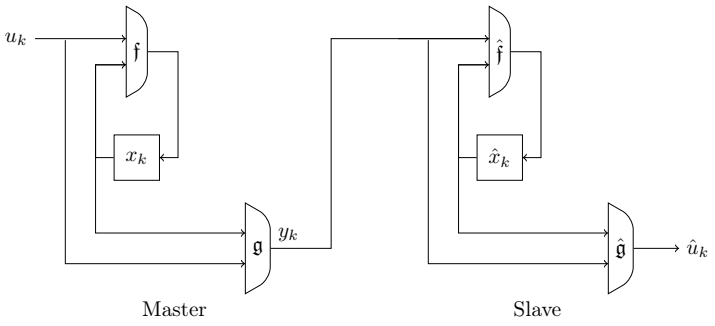


Figure 2.1: Master-slave setup (unidirectional point-to-point)

This configuration is also known as *unidirectional point-to-point*. In a communication system, the emitter is represented by the master and the receiver is represented by the slave. An extension of the one-to-one setup is the *unidirectional point-to-multipoints* one. In this setup, there is still one master but, there are several slaves corresponding to several receivers. The situation with two slaves is illustrated in Figure 2.2. The number of receivers does not matter for our purpose. Therefore, all along the thesis, we always consider the simplest situation, that is, with only one receiver. Section 2.2.2 deals with the synchronization issue between the slave and the master.

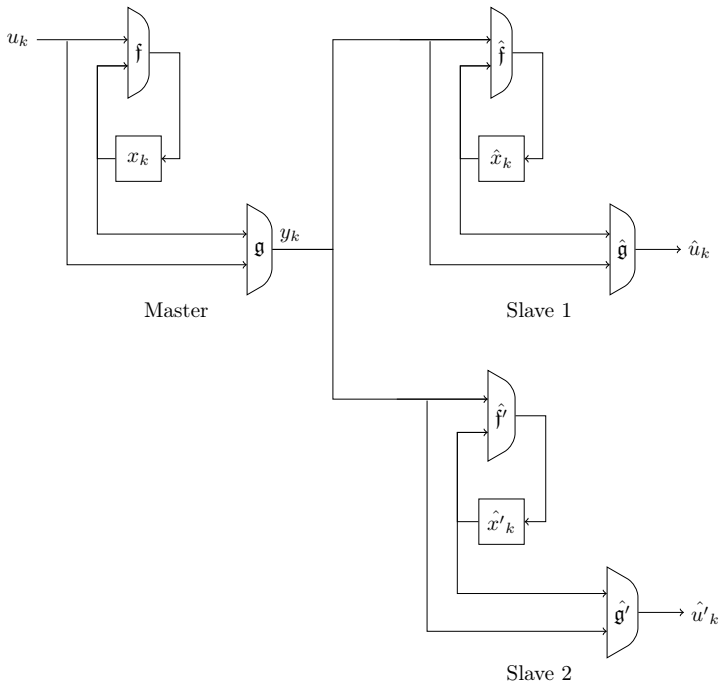


Figure 2.2: unidirectional point-to-multipoints setup with two slaves

2.2.2 Self-synchronization

In some applications, the slave needs to be synchronized with the master to guarantee that it processes the information properly. The notions related to synchronization are introduced in this section. There are many ways to define synchronization of dynamical systems, an important reference is [38]. However, for our purpose we consider the following restricted definition where the operator $||\cdot||$ stands for the norm of a vector.

Definition 2.2.1 (Finite-time convergence)**.** The state $\hat{x}$ of the slave (2.8b) is said to converge in finite-time toward the state x of the master (2.8a) if it satisfies

$$\forall x_0,\, \hat{x}_0 \in \mathbb{X}^n,\, \exists k_f \in \mathbb{N},\, \forall k \geq k_f,\, ||x_k - \hat{x}_k|| = 0 \tag{2.9}$$

A less restrictive version of Definition 2.2.1 is the so-called *asymptotic convergence*.

Definition 2.2.2 (Asymptotic convergence)**.** The state $\hat{x}$ of the slave (2.8b) is said to converge asymptotically toward the state x of the master (2.8a) if, for any input sequence (u), it satisfies

$$\forall x_0,\, \hat{x}_0 \in \mathbb{X}^n,\, \lim_{k\to\infty} ||x_k - \hat{x}_k|| = 0 \tag{2.10}$$

Definition 2.2.2 of asymptotic convergence coincides with definition 2.2.1 when the set $\mathbb{X}$ is finite.

When a specific sequence (y) delivered by the master is considered, the following definition can be stated.

Definition 2.2.3 (Synchronizing sequence)**.** A sequence (y) generated by (2.8a) is synchronizing for (2.8b) if there exists an integer k_y so that for all initial states $x_0, \hat{x}_0 \in \mathbb{X}^n$:

$$\forall k \geq k_y,\, x_k = \hat{x}_k \tag{2.11}$$

The value k_y is called the synchronization delay of the sequence (y).

Remark 2.2.1. This definition can be generalized by adding a constant delay R so that (2.11) turns into

$$\forall k \geq k_y,\ x_k = \hat{x}_{k+R} \tag{2.12}$$

Since the synchronization is achieved without any external control, we shall refer to self-synchronization. This being the case, we introduce below two definitions which will be used all along this thesis.

Definition 2.2.4 (Finite-time self-synchronization)**.** The slave system (2.8b) is finite-time self-synchronizing with the master (2.8a) if the minimum value k_y of Definition 2.2.3 is upper bounded when (y) lies in the set of admissible sequences. When it exists, the upper bound is called the *synchronization delay* and is denoted by k_f.

If (y) is a random sequence then, it is a random variable, and we shall denote it by (Y). The same consideration applies to k_y which shall be denoted K_Y since it is also a random variable in this case. A corresponding definition of self-synchronization can be introduced.

Definition 2.2.5 (Statistical self-synchronization)**.** The slave system (2.8b) is statistically self-synchronizing with the master (2.8a) if the probability Pr that $\hat{x}_k$ converges toward x_k satisfies

$$\lim_{k\to\infty} \Pr[K_Y \leq k] = 1. \tag{2.13}$$

Remark 2.2.2. The probability of occurrence of one sequence in a sequence whose length reaches infinity is one (provided that no value has a probability null to appear). Therefore, a system is statistically self-synchronizing as long as there is at least one synchronizing sequence. Hence, finite-time self-synchronization is a special case of statistical self-synchronization. Obviously, in practice, statistical self-synchronization must be achieved for a large enough number of sequences with a reasonable length.

2.3 Brief overview on cryptography

2.3.1 General concepts

Cryptography aims at ensuring privacy in communications. The most ancient discovered encrypted document can be traced back to the fifteen century B.C. Until recently, the scope was limited to hide the meaning of messages. However, with the needs of the last decades, it now covers more issues, which are in order:

Confidentiality provides the ability to communicate without being understood by any unauthorized party;

Authentication ensures that the emitter of a message is who he claims to be;

Non repudiation is a way to prevent someone to deny he has sent a message;

Integrity provides a way to ensure that the data have not been corrupted during the transmission.

In this thesis we are only concerned with confidentiality. The problem of private communications is described in Figure 2.3. An entity, P_A (a human, a computer, etc.), wants to send messages, called *plaintexts*, to another entity, P_B. To this end, it has to send the messages though a public channel. The channel is said to be public because anyone can intercept the messages conveyed on it. Let P_E be the entity that intercepts the messages. It is commonly called the adversary, the enemy or the eavesdropper. Since the use of a public channel cannot, most of the times, be avoided, if P_A wants to privately send messages to P_B, he must render the messages meaningless before sending them through the public channel. Hence, P_A has to apply a transformation, denoted by $\mathcal{E}$, to the messages. The transformed messages are called *cryptograms*. Of course, this transformation has to be invertible so that there exists a unique way to recover the original message. The inverse transformation of $\mathcal{E}$ is denoted by $\mathcal{D}$ and must be known only by P_B. The process of applying the transformation $\mathcal{E}$ is called encryption and the process

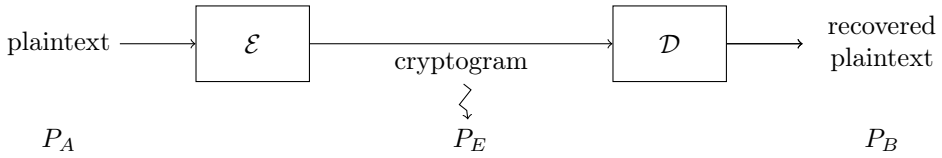


Figure 2.3: General privacy setup

of applying the reverse transformation $\mathcal{D}$ is called decryption.

The art of designing the system $\mathcal{E}$–$\mathcal{D}$ is called *cryptography*. Its counterpart, that is, the art of determining the weaknesses of the encryption process is called *cryptanalysis*. It allows to recover the information of the captured messages that travels over the public channel. Cryptography together with cryptanalysis constitute *cryptology*. Through the ages, different stratagems to hide information have been invented but, it quickly appeared that it was not possible to design a whole new system for each communication. A much better idea is to design a system parameterized by some secret value called the *key*. We denote it by $\kappa_{\mathcal{E}}$ at the emitter side and $\kappa_{\mathcal{D}}$ at the receiver side. This parameter has to be involved in the algorith $\mathcal{E}$ such that its knowledge is absolutely vital to determine $\mathcal{D}$ and then, to retrieve the original message. In this situation, both algorithms are denoted by $\mathcal{E}[\kappa_{\mathcal{E}}]$ and $\mathcal{D}[\kappa_{\mathcal{D}}]$. The importance of this key-based approach is that, once an algorithm is believed to be reliable, it can be used by a large number of parties. It is only required that P_B uses the key $\kappa_{\mathcal{D}}$ consistent with $\kappa_{\mathcal{E}}$ assigned to P_A. Another advantage is that, it does not assume that the specification of the algorithm is secret. It is in accordance with a principle stated by Kerckhoffs [39] in 1883. It states that, all the security of a cryptographic algorithm has to rely on the secret parameter. That is, although the complete specification of the algorithm is supposed to be public, it still has to fulfil its privacy guarantees. The following is a quotation of [39] which states this famous principle:

> Quant à la question du secret, qui, à mes yeux constitue le principal défaut de tous nos systèmes de cryptographie, je ferai observer qu'elle restreint en quelque sorte l'emploi de la correspondance chiffrée aux seuls commandants en chef. Et ici j'entends par secret, non la clef proprement dite, mais ce qui constitue la partie matérielle du système : tableaux, dictionnaires ou appareils mécaniques quelconques qui doivent en permettre l'application. En effet, il n'est pas nécessaire de se créer des fantômes imaginaires et de mettre en suspicion l'incorruptibilité des employés ou agents subalternes, pour comprendre que, si un système exigeant le secret se trouvait entre les mains d'un trop grand nombre d'individus, il pourrait être compromis à chaque engagement auquel l'un ou l'autre d'entre eux prendrait part.

In the general case, a message can be considered as a string of symbols in a set $\mathbb{A}$ of finite cardinality called an alphabet. For instance, if English messages are considered, the alphabet $\mathbb{A}$ is the set of the twenty six letters from the Latin alphabet. A message m of length ℓ is an element of the set $\mathbb{A}^\ell$. Concealing the string $m \in \mathbb{A}^\ell$, called the *plaintext*, means transforming m into another string c, called the *ciphertext* of length $\ell' \geq \ell$ of symbols of an alphabet $\mathbb{B}$. We usually assume that $\mathbb{B} = \mathbb{A}$ even though it is not required. The operation that transforms m into c has to be invertible and the recovered plaintext is denoted by $\hat{m}$. When the messages are processed through the time, we need to consider the plaintext, the ciphertext and the recovered plaintext processed at every time $k \in \mathbb{Z}$. We respectively denote them by m_k, c_k and $\hat{m}_k$.

Remark 2.3.1. The encryption process does not require to be a deterministic process. The encryption process may depend on some random elements. Obviously, this is not the case for the decryption process which has to properly recover the plaintext.

For many years, cryptology was essentially performed by people skilled at letters such as writers. However, a major change has occurred with the development of computer science. In 1948, Shannon published the seminal paper [40] on information theory. He formalized the notion of communications and defined a way

to quantify the amount of information contained in a message. Then, in 1949 he published the paper [7] which is a formalization of the concept of secret communications. The paper gives precise notions of what security is and how to measure it. It transforms the cryptography from the state of an art, to the state of a science. The paper introduces the notions of diffusion and confusion which are recalled below.

Diffusion is the principle through which the statistical structure of the plaintext m is dissipated into the long range statistics of the cryptogram c.
For instance, if m contains many times the letter "e", it should not be possible to find a pattern that reveals this structure in c.

Confusion is the principle through which the relation between the cryptogram c and the key $\kappa_{\mathcal{E}}$ has to be very complex and involved.

Nowadays, the ability to guaranty privacy is of first importance since our environment is made of many communicating electronic devices and that, many central aspects of our daily life depend on them. Encryption is prominent to ensure the privacy of the data whether they are intended to be stored on media or to be transmitted through a public communication channel. Moreover, they are not necessarily restricted to text messages but can be applied to pictures, videos, sounds, binary executables, etc. Hence, privacy techniques based on some alphabet composed of letters is of no use and other alphabets have to be considered.
Whatever the data are, when processed by electronic devices, they can always be represented by strings of symbols 0 or 1. In this case, a symbol is called a *bit*. We denote by $\mathbb{F}_2$ the field whose only elements are 0 and 1 (the definition of "field" is recalled in the appendix, Definition B.0.1). With this notation, the alphabet $\mathbb{A}$ can be chosen equal to the underlying set of $\mathbb{F}_2$. It is also possible to define symbols of several bits and hence to increase the size of the alphabet. The number of bits of a symbol is denoted by n_s and the corresponding alphabet has cardinality 2^{n_s}. For instance, a symbol can be a sequence of $n_s = 8$ bits and in this case, it is called a *byte* and the alphabet is composed of $2^8 = 256$ symbols.

Two categories of algorithms that ensure confidentiality can be identified depending on the way $\kappa_{\mathcal{E}}$ and $\kappa_{\mathcal{D}}$ are related. When the key $\kappa_{\mathcal{E}}$ can be easily deduced from $\kappa_{\mathcal{D}}$ (or conversely), algorithms are qualified of *symmetric*. In such a case, we usually consider both keys to be identical $\kappa_{\mathcal{E}} = \kappa_{\mathcal{D}}$ and we merely denote them by κ. In 1976, the existence of another kind of algorithms, called *asymmetric*, for which this property is not satisfied was conjectured [41]. The terminology asymmetric means that for such algorithms $\kappa_{\mathcal{E}}$ cannot be easily deduced from $\kappa_{\mathcal{D}}$ and conversely. Two years later, in 1978, the paper [42] proposed the first cryptographic algorithm with this property. This algorithm is named RSA and still has many applications today. Contrary to symmetric algorithms, asymmetric algorithms are based on mathematical problems known to be hard. For instance, in RSA, the security relies on the fact that we do not know how to efficiently decompose an integer into prime factors. This second type of algorithms offers new possibilities for dealing with cryptographic issues [43]. In this thesis, we are only concerned with symmetric algorithms. The different kinds of symmetric algorithms are detailed in the following section.

2.3.2 Classes of ciphers

Symmetric algorithms can be divided in two classes: *block ciphers* and *stream ciphers*. Usually, block ciphers operate on large blocks of n_b bits while stream ciphers operate on a single bit. However, stream ciphers that process several bits per time step are also relevant. This is typically the case when a symbol is encoded with more than one bit. We recall that the number of bits of a symbol is denoted by n_s. Hence, from this point of view, stream ciphers also operate on blocks of bits. Even though the number of bits n_b is much larger than n_s, the difference is not clear. A better criterion is provided in [44]. Block ciphers are ciphers such that the decryption transformation is unique but complex. Stream ciphers are such that the decryption transformation is simple but changes for every symbol and the security relies on the fact that the adversary does not know which transformation has to be used. Block ciphers are discussed in Section 2.3.2.1 and stream ciphers are discussed in Section 2.3.2.2.

2.3.2.1 Block ciphers

Block ciphers are algorithms that conceal large blocks of bits. The size of a block is denoted by n_b, a typical value for n_b is 64 or 128. The encryption function parameterized by a key κ is denoted by $\mathcal{B}[\kappa] : \mathbb{F}_2^{n_b} \longrightarrow \mathbb{F}_2^{n_b}$. The ciphertext c is obtained by applying the function $\mathcal{B}[\kappa]$ to the message m that is, $c = \mathcal{B}[\kappa](m)$. In accordance with Shannon's diffusion and confusion principles mentioned in Section 2.3.1, a security requirement for the transformation $\mathcal{B}[\kappa]$ is that $\mathcal{B}[\kappa]$ looks like a random function for any κ so that m, c and κ look unrelated. The recovered message $\hat{m}$ is obtained by applying the inverse transformation to c, that is $\hat{m} = \mathcal{B}[\kappa]^{-1}(c)$. If κ is the same at the encryption and decryption sides, the original message is properly recovered and $\hat{m} = m$.

Structure of block ciphers Directly implementing a transformation in accordance with the diffusion and confusion principles formalized by Shannon is not very convenient. For this reason, block ciphers are often composed of a relatively simple transformation, which does not properly satisfy the diffusion and confusion principles. However, a finite number of compositions of the simple transformation must satisfy the desired properties. Due to the iterated nature of these algorithms, they are called iterated block ciphers. The notion of iteration is important because, as discussed in Section 6.1, it allows to relate some flaws of block ciphers to some possible weaknesses of self-synchronizing stream ciphers. A more extensive discussion on the structure of block ciphers is out of the scope of this thesis. However, let us mention two important block ciphers.
One of the most well known block ciphers is the 56 key bits algorithm called Data Encryption Standard (DES for short) defined in [45]. It was selected in 1977 by the National Security Agency as a standard for the encryption of unclassified data. With the advances of technology, it has turned out that the size of the key was too short because computers could, in a reasonable time, find its value by an exhaustive search. In 1997, a call for a new algorithm was launched by the NIST (National Institute of Standard and Technology). The winner of the contest was an algorithm called *Rinjdael* [46] with a variable key size of 128, 192 or 256 bits. The algorithm is the Advanced Encryption Standard (AES for short) which is now the standard for block ciphers.

Encryption scheme There are different options, called *modes of operation*, to use block ciphers for encryption. The assignment of a mode of operation to a cipher is called an encryption scheme. Encryption schemes often use a transformation called the exclusive-or and denoted by $\oplus$. Table 2.1 gives the definition of $c = m \oplus z$ when $n_b = 1$ and where z is an intermediate variable. When $n_b > 1$, the elements c, m and z are vectors and their component i are respectively denoted by c^i, m^i and z^i. The exclusive-or of two vectors is defined componentwise by $c^i = m^i \oplus z^i$, $i = 0, \ldots, n_b - 1$.

m	z	c
0	0	0
0	1	1
1	0	1
1	1	0

Table 2.1: $c = m \oplus z$ when $n_b = 1$

Some encryption schemes require that the emitter and the receiver agree on a public parameter called an initialization vector (IV for short). The NIST has standardized the main modes in [47]. They are recalled below.

- Electronic code book (ECB for short) processes each plaintext independently of each other. Hence, two identical plaintexts produce the same ciphertexts. As a result, the encryption scheme is not secure. The ECB mode is illustrated in Figure 2.4a and is described by the following equations

$$\mathcal{E}\left\{c_k = \mathcal{B}[\kappa](m_k) \qquad \mathcal{D}\left\{\hat{m}_k = \mathcal{B}[\kappa]^{-1}(c_k)\right.\right. \tag{2.14}$$

- Cipher block chaining (CBC for short) is such that each block depends on the previous one. The CBC mode is illustrated in Figure 2.4b and is described by the following equations

$$\mathcal{E}\begin{cases} c_{-1} &= IV \\ c_k &= \mathcal{B}[\kappa](c_{k-1} \oplus m_k) \end{cases} \qquad \mathcal{D}\begin{cases} c_{-1} &= IV \\ \hat{m}_k &= \mathcal{B}[\kappa]^{-1}(c_k) \oplus c_{k-1} \end{cases} \tag{2.15}$$

- Output feedback (OFB for short) turns a block cipher into a synchronous stream cipher (see Section 2.3.2.2). The OFB mode is illustrated in Figure 2.4c and is described by the following equations

$$\mathcal{E}\begin{cases} z_{-1} &= IV \\ z_k &= \mathcal{B}[\kappa](z_{k-1}) \\ c_k &= m_k \oplus z_k \end{cases} \qquad \mathcal{D}\begin{cases} \hat{z}_{-1} &= IV \\ \hat{z}_k &= \mathcal{B}[\kappa](\hat{z}_{k-1}) \\ \hat{m}_k &= c_k \oplus \hat{z}_k \end{cases} \tag{2.16}$$

- Counter mode (CTR for short) encrypts a counter and performs the exclusive-or of the encrypted counter with the plaintext. The CTR mode is illustrated in Figure 2.4d and is described by the following equations

$$\mathcal{E}\begin{cases} z_{-1} &= IV \\ z_k &= z_{k-1} + 1 \\ c_k &= m_k \oplus \mathcal{B}[\kappa](z_{k-1}) \end{cases} \qquad \mathcal{D}\begin{cases} \hat{z}_{-1} &= IV \\ \hat{z}_k &= \hat{z}_{k-1} + 1 \\ \hat{m}_k &= c_k \oplus \mathcal{B}[\kappa](\hat{z}_{k-1}) \end{cases} \tag{2.17}$$

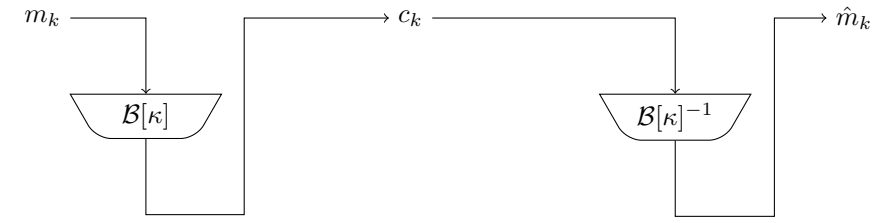
- Cipher feedback (CFB for short) turns a block cipher in a self-synchronizing encryption scheme. This mode is detailed in Section 2.4.4.

2.3.2.2 Stream ciphers

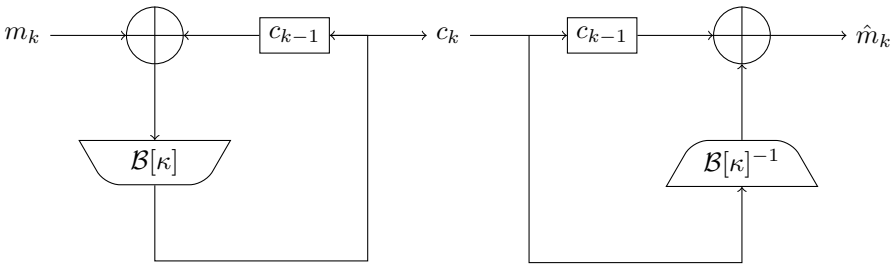
This section introduces the general principle of stream ciphers. Two categories are highlighted: synchronous ones and self-synchronizing ones. Unlike block ciphers, no attempt at defining any standardized stream cipher has succeeded, even though algorithms exist. For that reason, in 2001, the European Union has initiated a four year project called NESSIE to handle the problem. However, it did not succeed in producing any satisfactory result for this purpose. In 2004, a new four years European project called eSTREAM was initiated by the ECRYPT network. Several algorithms were proposed. However, very few self-synchronizing stream ciphers were presented. Indeed, among the sixteen proposals, only two belonged to the subclass of self-synchronizing stream ciphers. The algorithms were submitted to the cryptographic community and most of them have been rejected. Only a few synchronous stream ciphers have been kept in the contest and no self-synchronizing stream cipher was retained.

Stream ciphers are based on the following principle. At the emitter side, the cryptogram $c_k \in \mathbb{F}_2^{n_s}$ is obtained by mixing a symbol $m_k \in \mathbb{F}_2^{n_s}$ with a random symbol $z_k \in \mathbb{F}_2^{n_s'}$, called *keystream symbol* or *running key symbol*, through an invertible transformation $e(z_k, m_k) : \mathbb{F}_2^{n_s'} \times \mathbb{F}_2^{n_s} \longrightarrow \mathbb{F}_2^{n_s}$. At the receiver side, the information is recovered by using the function $e^{-1}(z_k, c_k) : \mathbb{F}_2^{n_s'} \times \mathbb{F}_2^{n_s} \longrightarrow \mathbb{F}_2^{n_s}$, the inverse transformation of e. By "inverse transformation of e", it is meant a transformation which has the ability to uniquely recover m_k from the knowledge of c_k for any prescribed z_k. This being the case, the knowledge of the random symbol z_k is required at the receiver side but, only $\hat{z}_k$, the receiver keystream, is accessible. Therefore, the decryption can be achieved if the same random source of information is available at the receiver side. It holds whenever $\hat{z}_k = z_k$ or in other words, if the keystreams are synchronized. This principle is illustrated in Figure 2.5.

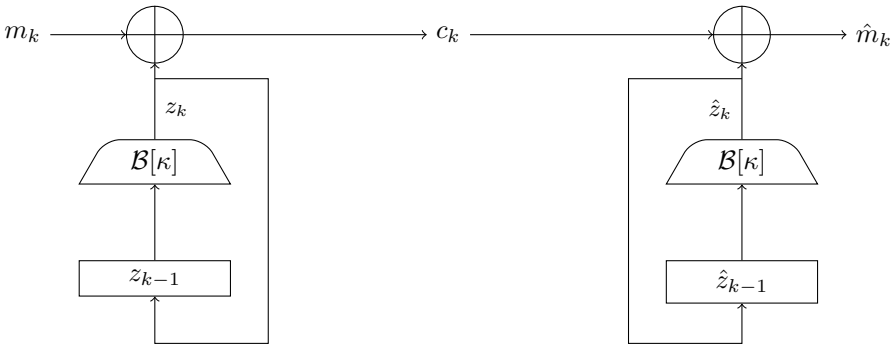
However, truly random signals cannot be reproduced. As a consequence, the strict application of this scenario requires to securely provide the emitter and the receiver with the random signal (through a diplomatic channel for instance). A typical cipher that obeys this principle is known as Vernam cipher and was patented in 1919 [48]. It is illustrated in Figure 2.6. In this context, the encryption function e is chosen to be the exclusive-or operation of m_k with z_k. The corresponding inverse function is also



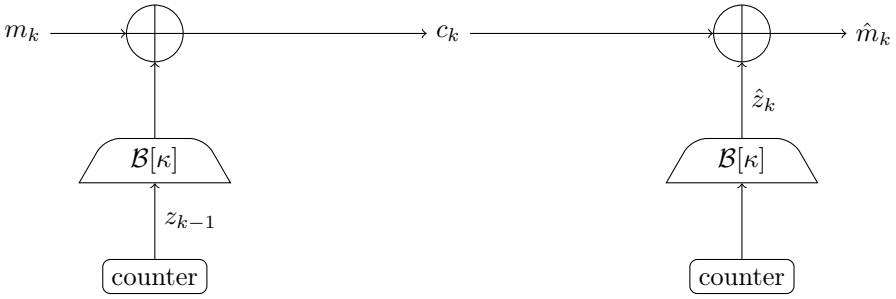
(a) ECB



(b) CBC



(c) OFB



(d) CTR

Figure 2.4: Four modes of operations for block ciphers

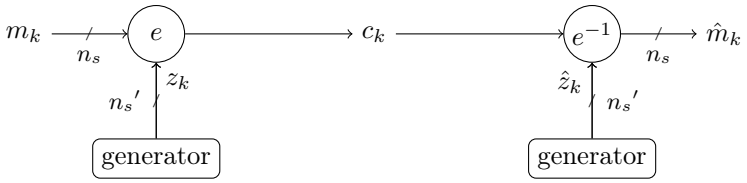


Figure 2.5: Stream cipher

the exclusive-or. With this cipher, there is no other strategy to recover the plaintext message than guessing the meaning of the cryptogram at random, a notion called unconditional security and discussed in Section 6.1. Vernam cipher has been used in some diplomatic correspondences, for instance during Cold War to protect the Red Phone communications between Washington and Moscow.

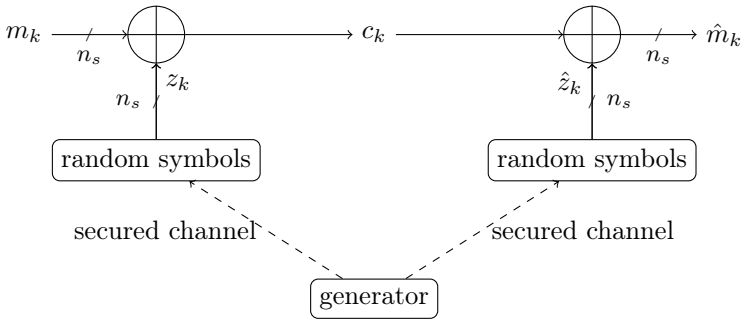


Figure 2.6: Vernam cipher

The direct use of Vernam cipher has two drawbacks. First, it requires the transmission of the keystream through a secured channel. Furthermore, the high level of security is only achieved if the symbols of the random stream are only used once. Thus, the keystream has to be as long as the data to be concealed. For these reasons, we usually resort to a degraded version of this cipher based on a pseudo-random generator that mimics Vernam cipher. A pseudo random generator is a symbol generator that produces a stream of symbols whose statistical properties look like random although it is clearly not. Indeed, these generators are based on finite dynamical systems. As a consequence, the produced sequences are periodic. However, it is only needed that the sequences look random and it can be the case if the period is large enough. Obviously, additional statistical properties are also required. A set of statistical tests to decide whether or not a sequence is random is given by the NIST in [49] along with software routines. One central feature of pseudo random generators is that they often rely on a deterministic process which allows the sequence to be reproducible. Hence, transmitting the random symbols through a secure channel is no longer required. Synchronous and self-synchronizing stream ciphers obey this principle.

Synchronous stream ciphers Synchronous stream ciphers are based on keystream generators in the form of dynamical systems at the emitter and receiver sides. At each time k, the states $x_k \in \mathbb{F}_2^n$ and $\hat{x}_k \in \mathbb{F}_2^n$ are updated by a next-state function $\mathfrak{f} : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$. A function $h : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^{n_s'}$ filters the state and produces a pseudo random symbol. The generated sequence completely depends on the initial state of the generator. For this reason the secret key of the generator is often chosen to be a part of the initial state and it is required to agree on this parameter before starting the process. The encryptor $\mathcal{E}$ and the decryptor $\mathcal{D}$ are governed by the equations

$$\mathcal{E} \begin{cases} x_{k+1} &= \mathfrak{f}(x_k) \\ c_k &= e(h(x_k), m_k) \end{cases} \tag{2.18a}$$

$$\mathcal{D}\begin{cases} \hat{x}_{k+1} & = \mathfrak{f}(\hat{x}_k) \\ \hat{m}_k & = e^{-1}(h(\hat{x}_k), c_k) \end{cases} \tag{2.18b}$$

Whenever $\hat{x}_k = x_k$, one has $\hat{m}_k = m_k$. In other words, the information is properly recovered if the dynamical systems are synchronized. In real life situations, communication channels are not error-free. Even though error correction mechanisms may exist, it is relevant to investigate the consequences of:

1. A symbol flip (the value of a symbol changes);
2. A symbol slip (a symbol is inserted or deleted).

Assume that the ciphertext c_τ is corrupted. The only recovered plaintext that depends on c_τ is $\hat{m}_\tau$. Therefore, the only symbol which is not properly recovered is $\hat{m}_\tau$. When a symbol slip occurs at time $k = \tau$, the receiver decodes the symbol $\hat{m}_\tau$ with the symbol $c_{\tau+1}$ or $c_{\tau-1}$ instead of c_τ. As a result, all the symbols for $k \geq \tau$ are not properly decoded. Therefore, it is clear that a synchronization mechanism must be added to recover from symbols slips. It can consist of an overhead in the implementation of the system which is not always affordable due to performance constraints such as the throughput or the device resources. Typically, it can be a reset of the system on a regular basis. As a conclusion, this method is not very sensitive to bit flips but bits slips dramatically degrade the performances.

Self-synchronizing stream ciphers Another kind of stream ciphers which does not suffer from such a problem is called self-synchronizing stream ciphers. They are extensively presented in Section 2.4.

2.4 Self-synchronizing stream ciphers

Self-synchronizing stream ciphers (SSSC for short) are a special class of stream ciphers to which very few attention has been paid despite the following advantages:

1. If a ciphertext is deleted, inserted or flipped during the transmission, the receiver automatically resumes proper decryption after a short, finite and predictable transient time. Hence, SSSC do not require any additional synchronization flag or interactive protocols for recovering from lost synchronization. Besides, since a SSSC does not require any additional bandwidth, the encrypted information or stream having exactly the same size as the plain information, an SSSC allows for flexibility from an implementation point of view when we aim at enhancing the security of a communication setup. In other words, SSSC can simplify the evolution of an existing encryption scheme if the former synchronization channel was too small or inadequate for security enhancement;
2. The self-synchronizing mechanism enables any receiver to switch at any time into an ongoing enciphered transmission. This ability of very fast synchronization is especially pertaining to group communications that are on increased usage today;
3. Any modification of ciphertext symbols by an active eavesdropper causes incorrect decryption for a fixed number of next symbols. The practical asset derived from the property is that an SSSC prevents active eavesdroppers from undetectable tampering with the plaintext: message integrity may also be guaranteed in complement to the confidentiality;
4. Each plaintext symbol influences the following ciphertexts, the statistical properties of the plaintext are thereby diffused through the ciphertext. Hence, SSSC are interesting against attacks based on plaintext redundancy and the property of diffusion is structurally fulfilled;
5. The information which is conveyed through the channel is encrypted whether there is traffic or not. Hence, the SSSC mechanism prevents from traffic analysis.

The literature about this topic is very limited. Most of the cryptographic books barely mention them and the number of publications on this topic is very poor.

The general principle is the following. Like other stream ciphers, SSSC are also based on keystream generators. However, unlike synchronous stream ciphers, the dynamical systems that correspond to the generators are not autonomous. In other words, the evolution of the state depends on a time dependent quantity namely, the last ciphertexts. In case of a correct transmission for a sufficiently long time, the same ciphertexts are available at the encryptor side and at the decryptor side. Therefore, the keystream generators share the same arguments and generate thereby the same symbols. A direct consequence of this principle is that the sequence generated at the receiver side does not depend on the initial state of the decryptor after a transient time. Hence, the secret parameter of the system can no longer be the initial state. One option is to parametrize the function of the generator by the key. Nevertheless, the initialization of the state, corresponding to the IV, is important. It should be chosen at random as explained in Section 6.1.1. When the transient time (also called delay) is finite, the system admits canonical equations.

Section 2.4.1 presents the canonical equations of an SSSC. Section 2.4.2 provides an extension called *generalized form*. Section 2.4.3 is a state of the art of the existing constructions. Section 2.4.4 explains how to turn a block cipher into a self-synchronizing encryption scheme. Finally, Section 2.4.5 is devoted to non standardized statistical self-synchronizing encryption schemes for block ciphers.

2.4.1 Canonical form

The canonical form can be described as follows. The keystream z_k (also called running key) is produced by a function $h[\kappa]$ which depends on a finite number k_f of past ciphertexts $c_{k-1}, \ldots, c_{k-k_f}$ and on the current plaintext m_k. The purpose of the function $h[\kappa]$ is to generate a random symbol from the past ciphertext. The running key is then combined in a very simple way through an invertible operation e depending on the plaintext m_k and on the running key z_k as explained in Section 2.3.2.2. Regarding this setup, if the ciphertexts $c_k, \ldots, c_{k-k_f}$ are properly transmitted, the plaintext m_k is properly recovered. The equations of the corresponding encryptor $\mathcal{E}$ and decryptor $\mathcal{D}$ obey the most general form:

$$\mathcal{E}\begin{cases} z_k &= h[\kappa](c_{k-1}, \ldots, c_{k-k_f}) \\ c_k &= e(z_k, m_k) \end{cases} \tag{2.19a}$$

$$\mathcal{D}\begin{cases} \hat{z}_k &= h[\kappa](c_{k-1}, \ldots, c_{k-k_f}) \\ \hat{m}_k &= e^{-1}(\hat{z}_k, c_k) \end{cases} \tag{2.19b}$$

From the equations above, it is now clear that SSSC are characterized by the fact that, at the decryptor side, the system loses any information about its initial state after some delay.

The sensitivity to transmission errors is now discussed. If an error occurs on the symbol c_τ, the recovered plaintexts are impacted as long as c_τ is an argument of $h[\kappa]$ or of e, that is, from time $k = \tau$ to $\tau + k_f$. Contrary to synchronous ciphers that do not suffer a lot from this kind of error, SSSC do. In case of symbol slip at time τ, only the ciphertexts from time τ to $\tau + k_f$ are affected and cannot be properly decrypted. Hence, the situation concerning symbols slips is much better than in the case of synchronous ciphers. Equations (2.19) are appropriate to describe SSSC. However, for our purpose it is better to write it in a different form.
We introduce $\mathfrak{s}_{k_f} : \mathbb{B} \times \mathbb{X}^{k_f} \longrightarrow \mathbb{X}^{k_f}$ with $\mathbb{X} = \mathbb{B}$, the shift-function composed of k_f cells of elements of $\mathbb{B}$ componentwise defined by

$$\mathfrak{s}_{k_f}(c_k, x_k) = \begin{pmatrix} c_k \\ x_k^0 \\ \vdots \\ x_k^{k_f-2} \end{pmatrix} \tag{2.20}$$

where $x_k^i \in \mathbb{B}$ is the component i of the vector x_k.
When $\mathbb{B} = \mathbb{F}_2^{n_s}$, we may specify that $\mathfrak{s}_{k_f}$ depends on n_s hence, we denote it by $\mathfrak{s}_{k_f,n_s}$. It is a $(k_f n_s + n_s, k_f n_s)$–function.

The system (2.19) can be equivalently written in the so-called canonical recursive form which is

$$\mathcal{E}\begin{cases} x_{k+1} &= \mathfrak{s}_{k_f}(c_k, x_k) \\ c_k &= e(h[\kappa](x_k), m_k) \end{cases} \tag{2.21a}$$

$$\mathcal{D}\begin{cases} \hat{x}_{k+1} &= \mathfrak{s}_{k_f}(c_k, \hat{x}_k) \\ \hat{m}_k &= e^{-1}(h[\kappa](\hat{x}_k), c_k) \end{cases} \tag{2.21b}$$

The corresponding system is depicted in Figure 2.7.

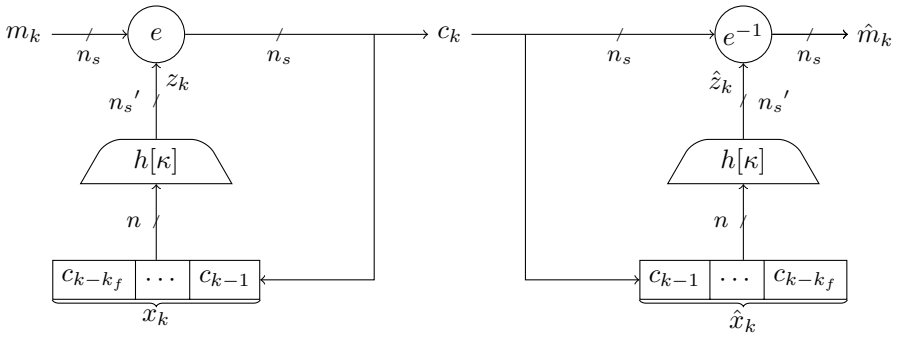


Figure 2.7: Canonical recursive form of an SSSC

After a sufficiently large amount of time which is equal to k_f, the state of the receiver $\hat{x}_k$ is equal to the state of the emitter x_k. Hence, the keystreams $(z)_{k_f}^{\infty}$ and $(\hat{z})_{k_f}^{\infty}$ are equal. This property holds regardless of the initial values x_0 and $\hat{x}_0$.

2.4.2 Generalized form

The specificity of the canonical recursive form is that the next-state function is a shift function. An interesting extension is to replace the shift by another next-state function. To this end, we define the generalized form. The corresponding scheme is depicted in Figure 2.8 and its equations read

$$\mathcal{E}\begin{cases} x_{k+1} &= f[\kappa](c_k, x_k) \\ c_k &= e(h[\kappa](x_k), m_k) \end{cases} \tag{2.22a}$$

$$\mathcal{D}\begin{cases} \hat{x}_{k+1} &= f[\kappa](c_k, \hat{x}_k) \\ \hat{m}_k &= e^{-1}(h[\kappa](\hat{x}_k), c_k) \end{cases} \tag{2.22b}$$

For short, we merely write f and h instead of $f[\kappa]$ and $h[\kappa]$. Obviously, not any function f can be used. It must have the property that, after a sufficiently large number of iterations, the state $\hat{x}_k$ of the decryptor does no longer depend on the initial state $\hat{x}_0$. This issue will be the main purpose of Chapter 5. Besides, we emphasize that system (2.22a) differs from the classical equation of dynamical systems (2.8a) in that the function f of (2.22a) depends on the state of the system and on the output while the function $\mathfrak{f}$ of (2.8a) depends on the state of the system and on the input.
Considering (2.22a)–(2.22b) and Figure 2.8, the following definitions can be stated.

Definition 2.4.1 (Finite-time self-synchronizing function)**.** The function $f : \mathbb{F}_2^{n_s} \times \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ is finite-time self-synchronizing if, when (c) stands in the set of all input sequences, the minimum value of the synchronization delay of each sequence k_c is upper bounded. The upper bound is called the synchronization delay of f and is denoted by k_f.

Remark 2.4.1. Finite-time self-synchronization means that, there exists an integer k_c such that any sequence of length at least k_c is a self-synchronizing sequence. The synchronization delay depends on the pair of initial states x_0 and $\hat{x}_0$. The delay k_c is defined as the maximum delay over all initial state pairs.

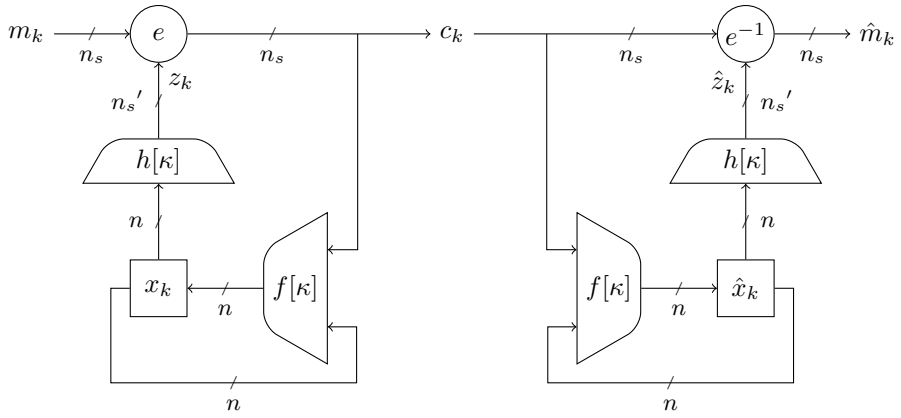


Figure 2.8: Generalized form of an SSSC

In the same way, Definition 2.2.5 can be restated.

Definition 2.4.2 (Statistical self-synchronizing function)**.** A function $f : \mathbb{F}_2^{n_s} \times \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ is statistically self-synchronizing if the probability Pr that $\hat{x}_k$ synchronizes with x_k satisfies $\lim_{k\to\infty} \Pr(K_C \leq k) = 1$, where K_C is the random synchronization delay for the random sequence (C).

Next, we recall the constructions provided in the literature. New constructions will be proposed in Section 6.2.

2.4.3 Self-synchronizing primitives

In this section, we give the state of the art on self-synchronizing primitives which are the core of SSSC setups. As explained earlier, the literature is very poor. The most extensive discussion on the topic is given in Chapter 9 of [44]. In 1992, a SSSC called KNOT was proposed in [50]. The algorithm has been broken in 2003 in the paper [51]. During that time an improved version of the algorithm called $\Upsilon\Gamma$ was proposed in [44]. Another algorithm called Hiji-Bij-Bij (HBB for short) was proposed in 2001 in [52]. It has been broken in [53, 54].

In 2005, the eSTREAM project has led to the specifications of two self-synchronizing stream ciphers: SSS [55] and Mosquito [56]. The latter one is an improved version of $\Upsilon\Gamma$. They where respectively broken in [57] and in [58]. Mosquito has then been fixed and a new version called Moustique [59] (the French vocable for Mosquito) has been specified in 2008. The latter has also been broken in [60] the same year. As a consequence, today, there is no publicly known SSSC considered as safe. A more recent work [61] partially discusses the design issue and the paper [62] provides a cryptanalysis framework to this design. An original construction is [63]. To the knowledge of the author, the only public proposition of statistical self-synchronizing stream cipher. However, it has been broken in [64].

2.4.3.1 Primary constructions

Primary constructions are elementary self-synchronizing units. They are all based on a type of functions that we call *generalized strict T–function*. The definition is the following.

Definition 2.4.3 (Generalized strict T–function)**.** A mapping $f(c, x) : \mathbb{F}_2^{n_s} \times \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ is called a generalized strict T–function if the coordinate function f^j depends only on the variables c and x^i for $i = 0, \ldots, j-1$.

Such a function admits the general form

$$f(c,x) = \begin{pmatrix} f^0(c) \\ f^1(c,x^0) \\ \vdots \\ f^{n-2}(c,x^0,\ldots,x^{n-4},x^{n-3}) \\ f^{n-1}(c,x^0,\ldots,\ldots,x^{n-3},x^{n-2}) \end{pmatrix} \tag{2.23}$$

Such a terminology has been proposed here to comply with the name of the functions used in [65] and called *T–function*. The definition is the following.

Definition 2.4.4 (*T*–function)**.** A mapping $f(x) : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ is called a *T*–function if the coordinate function f^j depends only on the variables x^i with $i = 0,\ldots,j$.

Some special *T*–functions of interest in our study are called strict *T*–functions. We define them as follows.

Definition 2.4.5 (Strict *T*–function)**.** A mapping from $f(x) : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ is called a strict *T*–function if the coordinate function f^j depends only on the variables x^i with $i = 0,\ldots,j-1$.

Remark 2.4.2. What we call *strict T–function* is called *parameter* in [66]. However, we think that the name *parameter* is misleading in our context and we prefer to use the name *strict T–function* in order not to confuse the reader.

If the function f is a generalized strict *T*–function, it can be seen that the effect of the initial state is progressively shifted out towards the part of the state with the highest index and finally disappears after at most n iterations. That agrees with the property of self-synchronization. This observation was already stressed in [61].

2.4.3.2 Secondary constructions

In order for a system to achieve confidentiality, it is required to have a large enough state as it will be further discussed in Section 6.1. Secondary constructions provide a solution to combine self-synchronizing constructions to extend the size of the state. The paper [67] addresses this issue. It is not so recent but no other constructions have been proposed so far. The serial and parallel compositions are two options for that purpose.

Serial composition Serial composition merely consists in connecting the output of a dynamical system to the input of another one. The setup is illustrated in Figure 2.9. The synchronizing delay of the resulting

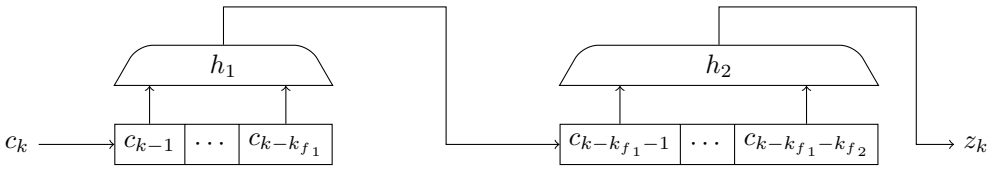


Figure 2.9: Serial composition

self-synchronizing system is equal to the sum of the synchronization delays of each of the systems involved in the setup.

Parallel composition A set of systems can also be connected in a parallel fashion. In this setup, all the systems are fed with the same input. The output of the whole system is a function of the outputs of the dynamical systems. Parallel composition is depicted in Figure 2.10. The corresponding synchronization delay is equal to the maximum synchronizing delay among all the systems.

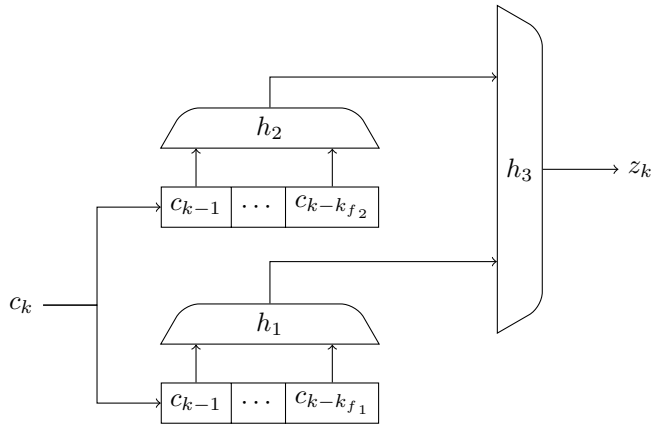


Figure 2.10: Parallel composition

Serial-parallel The serial-parallel composition is a combination of the serial and parallel constructions. It is illustrated in Figure 2.11. The corresponding synchronization delay is equal to the maximum synchronization delay of the serial components. Obviously, other compositions can be proposed on the same spirit.

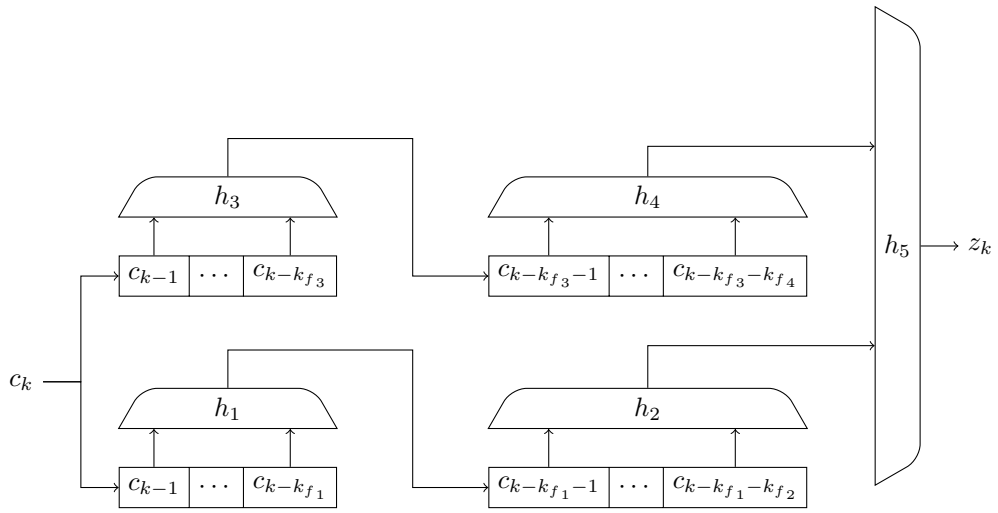


Figure 2.11: Serial-parallel composition

Next section explains how to turn a block cipher into a self-synchronizing encryption scheme.

2.4.4 Cipher feedback mode

We turn back and detail the approach introduced in Section 2.3.2.1 which consists in using a block cipher in cipher feedback mode (CFB for short) to turn it into a self-synchronizing encryption scheme [47]. The principle is the following, at each time k, a shift register is fed with the past ciphertext symbols. The register is then used as an input for a block cipher $\mathcal{B}[\kappa]$. Therefore, the size of the register has to be equal to n_b, the block size of $\mathcal{B}[\kappa]$. The running key z_k is obtained by extracting a subset of bits of the output of the block cipher. The mechanism is illustrated in Figure 2.12 and the corresponding equations are the

following.

$$\mathcal{E}\begin{cases} x_{-1} & = IV \\ x_k & = \mathfrak{s}_{k_f,n_s}(c_{k-1}, x_{k-1}) \\ z_k & = \mathcal{B}[\kappa](x_k) \\ c_k & = m_k \oplus (z_k^0, \ldots, z_k^{n_s-1}) \end{cases} \qquad \mathcal{D}\begin{cases} \hat{x}_{-1} & = IV \\ \hat{x}_k & = \mathfrak{s}_{k_f,n_s}(c_{k-1}, \hat{x}_{k-1}) \\ \hat{z}_k & = \mathcal{B}[\kappa](\hat{x}_k) \\ \hat{m}_k & = c_k \oplus (\hat{z}_k^0, \ldots, \hat{z}_k^{n_s-1}) \end{cases}$$

where IV is an initialization vector as explained in Section 2.3.2.1.

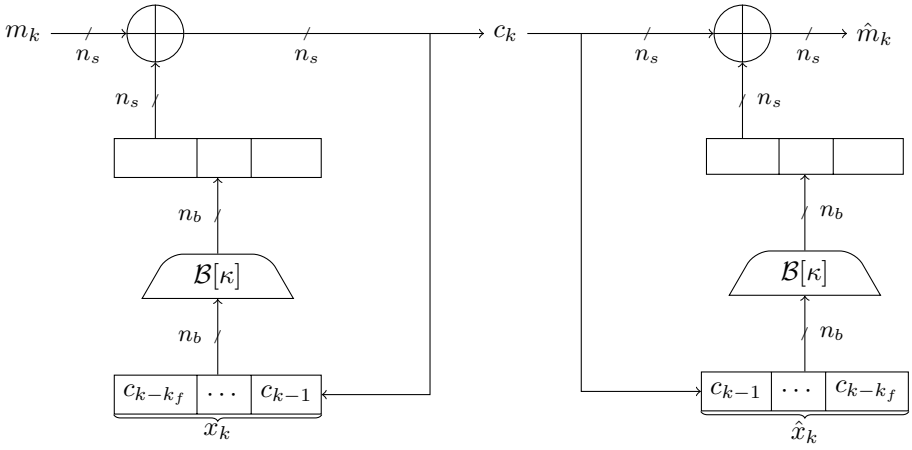


Figure 2.12: Block cipher in CFB mode

The resulting throughput is n_s bits per time step. However, due to the structure of block ciphers, all the n_b bits of its output must be computed before encrypting the message. Thus, this approach is questionable from a computational point of view. Despite this drawback, it is the most well established approach for the design of SSSC. The reason is that block ciphers are the only ones that have been standardized and that are trustworthy. Indeed, in cryptography, it is always better to rely on well-tested structures, if the application can afford it, than to invent a whole new design that may suffer from unexpected weaknesses.

2.4.5 Self-synchronizing statistical modes

Other modes that turn a block cipher into a self-synchronizing encryption scheme are reported in the literature. However, contrary to the CFB mode, they are not standardized. Two modes of operations are discussed. One of their specificities is that the self-synchronization is statistical.

2.4.5.1 Statistical cipher feedback mode

Statistical cipher feedback (SCFB for short) is a mode which was proposed in [68]. Its statistical self-synchronization effect is studied in [69]. The idea is to combine the high throughput provided by the OFB mode with the self-synchronization capability of the CFB mode. The system switches between these two modes depending on some sequences in the ciphertext symbols. A *sync module* scans the ciphertext at the encryptor and decryptor sides and loads the input register of the block ciphers with it when a specific pattern is detected. Thus, it operates as in the CFB mode. The rest of the time, it operates as in the OFB mode. The symbol size is the same as the block size: $n_s = n_b$. Figure 2.13 provides an illustration of this mode.
In case of synchronization loss, the decryptor works erroneously until the encryptor delivers a cryptogram that involves a synchronizing sequence. At each time step, n_b bits of data can be encrypted.

2.4.5.2 Optimized cipher feedback mode

Optimized cipher feedback mode (OCFB for short) is another statistical self-synchronizing mode introduced in [70]. It is depicted in Figure 2.14 and works as follows. The input registers are fed with the

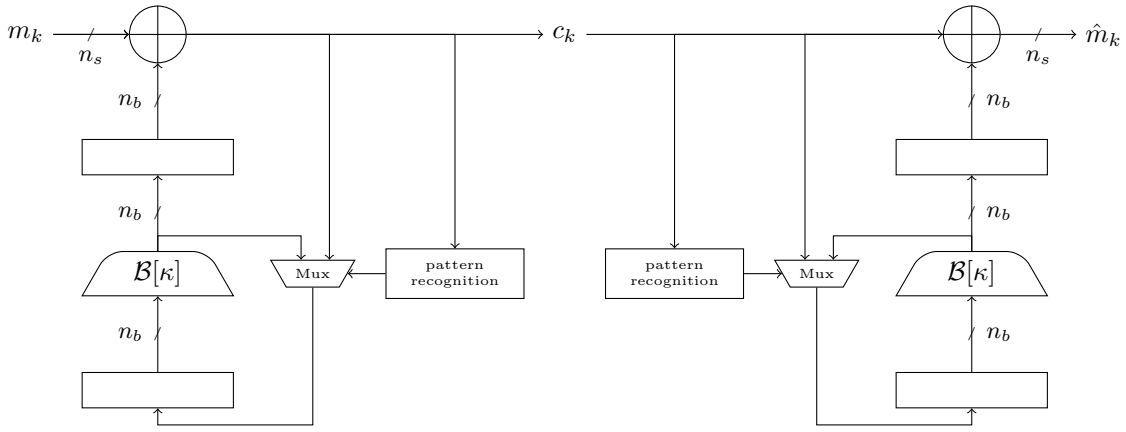


Figure 2.13: Statistical cipher feedback

past ciphertexts and the output registers are fed with the cryptograms corresponding to the encryption of the input registers. Unlike the previous principle, the input register is not encrypted at each time step. Besides, the size of the blocks has to be congruent modulo the size of the symbols $n_b = \ell n_s, \ell \in \mathbb{N}^*$. At each time step, the control unit (denoted by ctrl in the figure) shifts the output register by n_s bits as well. The input register is shifted by n_s bits and the cryptogram is introduced in the register. After ℓ times, it triggers the encryption of the input register. The control unit also triggers the encryption of the input register when a synchronization pattern is detected in the input register.

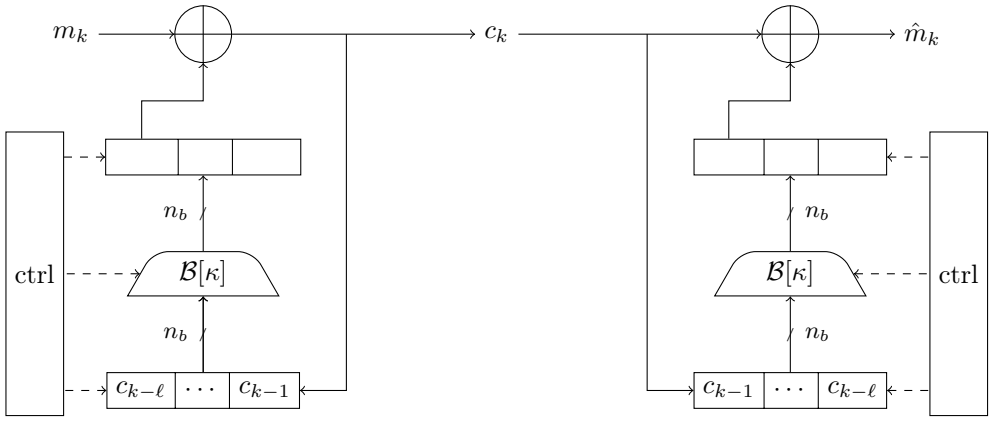


Figure 2.14: Optimized cipher feedback mode

The differences between synchronization probability law of the SCFB mode and the OCFB modes are discussed in [71].

2.5 Flatness for self-synchronization characterization

The concept of flat dynamical system has been introduced in [72]. A recent treatment can be found in [73]. From the engineering point of view, applications dealing in particular with motor control can be found for example in [74, 75]. Interestingly, it can also be used in an original way in the context of self-synchronization [10]. Essentially, we show in Section 2.5.1 the connection between flat dynamical systems and the canonical form of self-synchronizing stream ciphers. Section 2.5.2 motivates this approach.

Definition 2.5.1 (Flat dynamical system [76])**.** A square dynamical system, that is when the number of inputs equals the number of outputs, is said to be *flat* if there exists a set of independent output

variables y_k, referred to as *flat outputs* such that, all the system variables can be expressed as a function of the flat outputs and a finite number of its backward and/or forward iterates. In particular, there exist two functions F and G which obey

$$\left\{ \begin{array}{lcl} x_k & = & F(y_{k+k_F}, \ldots, y_{k+k'_F}) \\ u_k & = & G(y_{k+k_G}, \ldots, y_{k+k'_G}) \end{array} \right. \tag{2.24}$$

where k_F, k'_F, k_G and k'_G are $\mathbb{Z}$-valued integers. We call the integer $k'_F - k_F + 1$ the flatness characteristic number.

2.5.1 Connection between the different forms

In this section, we establish the connection between the equations of the master (2.8a) of Section 2.2, the recursive canonical form (2.21a) of Section 2.4.1 and the generalized form (2.22a) of Section (2.4.2). The purpose is to show that, under the flatness assumption fulfilled by (2.8a), all these representations produce the same input/output sequences after a finite transient time.

Proposition 2.5.1. *The equations of the master* (2.8a) *produce the same input/output sequences after a finite transient time than the recursive canonical equations* (2.21a) *if and only if the master* (2.8a) *is flat.*

Proof. The specificity of the canonical recursive form (2.21a) is that, the state x_k is updated by a shift function $\mathfrak{s}_{k_f}$. The proposition is proved by showing that, under flatness assumption of (2.8a), it is always possible to find a system with a state updated shift function that produces the same input/output sequences than (2.8a).
Assume that the master (2.8a) is flat then, (2.24) holds. Let us define $x'_k = (y_{k+k_F}, \ldots, y_{k+k'_F})$ with $k_f = k'_F - k_F + 1$ the flatness characteristic number. As a result, the equalities $x_k = F(x'_k)$ and $y_k = \mathfrak{g}(F(x'_k), u_k)$ hold. The state trajectory $(x) = x_k, x_{k+1}, \ldots$ can be generated by the following dynamical system.

$$\mathcal{E} \left\{ \begin{array}{ll} x'_{k+1} & = \mathfrak{s}_{k_f}(y_k, x'_k) \\ y_k & = \mathfrak{g}(F(x'_k), u_k) \end{array} \right. \tag{2.25}$$

with $\mathfrak{s}_{k_f}$ the shift function. The quantities involved in (2.25), and so in (2.8a), can be identified to the ones of (2.21a) as follows.

(2.21a)	(2.25)
m_k	u_k
c_k	y_k
x_k	$F(x'_k)$
$e(h(x_k), m_k)$	$\mathfrak{g}(F(x'_k), u_k)$

If the system is not flat, F would require an infinite number of arguments and a state vector with an infinite dimension would be required.
Next, we show that the recursive canonical form (2.21a) can always be written in the form of the master (2.8a). From (2.21a), the following equality holds $\mathfrak{s}_{k_f}(c_k, x_k) = \mathfrak{s}_{k_f}(e(h(x_k), m_k), x_k)$. Hence, $\mathfrak{s}_{k_f}(c_k, x_k)$ can be expressed as a function of x_k and m_k. As a result, (2.21a) can be equivalently rewritten as

$$\mathcal{E} \left\{ \begin{array}{ll} x_{k+1} & = \mathfrak{s}_{k_f}(e(h(x_k), m_k), x_k) \\ c_k & = e(h(x_k), m_k) \end{array} \right. \tag{2.26}$$

The quantities involved in (2.26), and in of (2.21a), can be identified to the ones of (2.8a) as follows

(2.8a)	(2.26)
u_k	m_k
y_k	c_k
x_k	x_k
$\mathfrak{f}(x_k, u_k)$	$\mathfrak{s}_{k_f}(e(h(x_k), m_k), x_k)$
$\mathfrak{g}(x_k, u_k)$	$e(h(x_k), m_k)$

which completes the proof. □

It should be interesting to have an explicit expression of F. The k_fth order iterated function of $\mathfrak{s}_{k_f}$ does no longer depend on the initial condition x_k after a transient time of length k_f by definition of the shift function. Hence, it reads $\mathfrak{s}_{k_f}^{\circ k_f}(c_{k-1}, \ldots, c_{k-k_f})$ and then, $F(y_{k-1}, \ldots, y_{k-k_f})$ can be identified to $\mathfrak{s}_{k_f}^{\circ k_f}(c_{k-1}, \ldots, c_{k-k_f})$.

Proposition 2.5.2. *The generalized form* (2.22a) *can always be written in the form of the master* (2.8a).

Proof. The problem amounts to showing that there always exists a way to express the dynamical system (2.22a) such that x_{k+1} depends on a function that exclusively depends on the state x_k and the input m_k. Such a function exists and can be obtained by replacing the expression $c_k = e(h(x_k), m_k)$ of (2.22a) in the equation $x_{k+1} = f(e(h(x_k), m_k), x_k)$ of (2.22a). Hence, (2.22a) turns into

$$\mathcal{E} \begin{cases} x_{k+1} & = f(e(h(x_k), m_k), x_k) \\ c_k & = e(h(x_k), m_k) \end{cases} \tag{2.27}$$

The quantities involved in (2.27), and so in (2.22a), can be identified to the ones of (2.8a) as follows.

(2.22a)	(2.27)
u_k	m_k
y_k	c_k
x_k	x_k
$\mathfrak{f}(x_k, u_k)$	$f(e(h(x_k), m_k), x_k)$
$\mathfrak{g}(x_k, u_k)$	$e(h(x_k), m_k)$

□

As a result, the following proposition can be stated.

Proposition 2.5.3. *The generalized form* (2.22a) *has the structure of a finite-time self-synchronizing stream cipher if and only if the system* (2.27) *is flat.*

Proof. This is a direct consequence of Proposition 2.5.1 and Proposition 2.5.2. □

2.5.2 Motivation of the generalized form approach

Considering the generalized form approach instead of the canonical form approach has the following advantages:

- it leads to a more efficient implementation;
- it offers new possibilities in the choice of the next-state functions;
- it allows statistical self-synchronization.

2.5.2.1 Efficient implementation

One of the main advantages of the generalized form (2.22) is that it yields to an efficient implementation. Implementing the system is more efficient than considering the recursive canonical form (2.21). With the canonical form, most of the complexity is determined by the output function and very little is left in the next-state function. The latter one is kept very simple in order to ensure the self-synchronizing property. The purpose of the generalized form is to balance the complexity of the next-state function and of the output function.

2.5.2.2 New possibilities in the choice of the key generators

SSSC have not been extensively studied but, some algorithms have been designed such as SSS [55] or Moustique [59]. The synchronization is always achieved with the same technique: using generalized strict T–functions as explained in Section 2.4.3. The existence in automatic control of complex flat systems suggests that the use of these generalized strict T–functions is not mandatory to achieve self-synchronization and that it is possible to find functions such that any variable is updated by a function that depends on any other variables. This topic is developed in Chapter 3 and in Chapter 5.

2.5.2.3 Statistical self-synchronization

The generalized form approach also allows to design statistical self-synchronizing ciphers which would not be possible otherwise. The reason why the canonical form approach does not permit the description of statistical designs is because the function h would require an infinity of arguments which is obviously not possible to implement.

Chapter 3

Flatness of hybrid dynamical systems

In Chapter 2, the relation between SSSC and flat dynamical systems have been highlighted. In this chapter, we focus on the flatness characterization of special classes of dynamical systems namely, switched linear one and linear parameter-varying one (LPV for short). The layout is the following. Section 3.1 recalls the necessary background on switched linear systems. Section 3.2 presents our flat output characterization approach. An extension to LPV systems is proposed in Section 3.3. Finally, a constructive approach to design master-slave setups based on flat switched linear systems, and so self-synchronizing, is proposed in Section 3.4. The results established for switched linear systems will be functional for the design of SSSC as detailed in Section 3.4 and in Chapter 5. They have been published in [77, 78, 79].

3.1 Switched linear systems

A switched linear system is a dynamical system described by the following equations

$$\left\{ \begin{array}{ll} x_{k+1} & = A_{\sigma(k)}x_k + B_{\sigma(k)}u_k \\ y_k & = C_{\sigma(k)}x_k + D_{\sigma(k)}u_k \end{array} \right. \tag{3.1}$$

with $\mathbb{F}$ a field and $u_k \in \mathbb{F}^{n_i}$, $y_k \in \mathbb{F}^{n_o}$, $x_k \in \mathbb{F}^n$. The switching rule σ obeys

$$\sigma : k \in \mathbb{N} \longmapsto j = \sigma(k) \in \{1, \ldots, J\} = \mathcal{J} \tag{3.2}$$

At a given time k, the index j is the mode of the system and J is the number of modes. All the matrices, namely $A_{\sigma(k)} \in \mathbb{F}^{n\times n}$, $B_{\sigma(k)} \in \mathbb{F}^{n\times n_i}$, $C_{\sigma(k)} \in \mathbb{F}^{n_o\times n}$ and $D_{\sigma(k)} \in \mathbb{F}^{n_o\times n_i}$ belong to the respective finite sets $\{A_j,\, j \in \mathcal{J}\}$, $\{B_j,\, j \in \mathcal{J}\}$, $\{C_j,\, j \in \mathcal{J}\}$ and $\{D_j,\, j \in \mathcal{J}\}$. The matrices A_j, $j \in \mathcal{J}$ are called the dynamical matrices and the matrices B_j, $j \in \mathcal{J}$ are called the input matrices. The matrices C_j, $j \in \mathcal{J}$ are the output matrices and D_j, $j \in \mathcal{J}$ are called the direct transfer matrices. When the number of modes is $J = 1$, the system (3.1) reduces to a linear system.

3.1.1 Left input invertibility

Flatness is closely related to the notion of left invertibility which actually stands for a necessary flatness condition. Roughly speaking, left invertibility of a dynamical system is the ability of uniquely determining the input sequence from the output sequence. The works dealing with left invertibility reported in [80] are considered throughout the literature as the pioneering ones. Left invertibility for switched linear systems has been addressed in [81] for continuous-time systems and in [82, 83, 84] for discrete-time systems.
The concept of left inverse system, related to left invertibility, will play a central role for our purpose. The following definition is in accordance with the papers [83, 84, 76].

Definition 3.1.1 (left r-delay inverse system)**.** A system is a left r-delay inverse for (3.1) if, under identical initial conditions x_0 and $\hat{x}_0$ and identical mode sequences (σ), there exists a nonnegative integer r such that, when driven by $(y)_k^{k+r}$, the equalities $\hat{x}_{k+r} = x_k$ and $\hat{u}_{k+r} = u_k$ for all $k \geq 0$ are ensured, $\hat{u}_k$ being its output at time k. The nonnegative integer r is called the inherent delay.

Remark 3.1.1. In Definition 3.1.1, the initial condition is considered at the particular discrete-time $k = 0$ but can be replaced by any other initial condition x_k taken at the discrete time k.

Let us notice that the terminology of r-delay inverse and inherent delay is borrowed from the work [85] which deals with linear systems. Besides, the consideration of the initial condition x_0 stands as a counterpart of the continuous case and the definition of *invertibility at point* x_0 introduced in [86]. Actually, the initial condition x_0 has been disregarded in [85] by assuming that it is zero or that "its effect can be substracted".

Definition 3.1.2 (Input left invertibility)**.** The system (3.1)–(3.2) is input left invertible if it admits a left r–delay inverse system.

The vocable *input* is introduced to stress that only the input must be recovered, the modes being known. The papers [83, 84, 76] give an explicit form of the left r-delay inverse system for (3.1). It is recalled below.
Let us define the following matrices.

$$\mathcal{O}_{\sigma(k:k+i)} = \begin{pmatrix} C_{\sigma(k)} \\ C_{\sigma(k+1)}A_{\sigma(k)} \\ \vdots \\ C_{\sigma(k+i)}A_{\sigma(k)}^{\sigma(k+i-1)} \end{pmatrix} \tag{3.3}$$

The matrix $\mathcal{O}_{\sigma(k:k+i)}$ involves the transition matrix defined by

$$\begin{array}{rcll} A_{\sigma(k_0)}^{\sigma(k_1)} & = & A_{\sigma(k_1)}A_{\sigma(k_1-1)}\cdots A_{\sigma(k_0)} & \text{if } k_1 \geq k_0 \\ & = & \mathbf{1}_n & \text{if } k_1 < k_0 \end{array}$$

Finally, we recursively define the matrix

$$M_{\sigma(k:k+i)} = \begin{pmatrix} D_{\sigma(k)} & \mathbf{0} \\ \mathcal{O}_{\sigma(k:k+i)}B_{\sigma(k)} & M_{\sigma(k+1:k+i)} \end{pmatrix} \tag{3.4}$$

with

$$M_{\sigma(k:k)} = D_{\sigma(k)}$$

By using the same notations than those devoted to sequences, we define the following vectors

$$(u)_k^{k+i} = \begin{pmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+i} \end{pmatrix}, (u')_k^{k+i} = \begin{pmatrix} u'_k \\ u'_{k+1} \\ \vdots \\ u'_{k+i} \end{pmatrix}, (y)_k^{k+i} = \begin{pmatrix} y_k \\ y_{k+1} \\ \vdots \\ y_{k+i} \end{pmatrix}, (y')_k^{k+i} = \begin{pmatrix} y'_k \\ y'_{k+1} \\ \vdots \\ y'_{k+i} \end{pmatrix} \tag{3.5}$$

When (3.1) is driven by an input sequence $(u) \in \mathbb{A}^\star$ and a corresponding mode sequence $(\sigma) \in \mathcal{J}^\star$, the following equality:

$$(y)_k^{k+i} = \mathcal{O}_{\sigma(k:k+i)}x_k + M_{\sigma(k:k+i)}\,(u)_k^{k+i} \tag{3.6}$$

We recall a theorem stated in [76]:

Theorem 3.1.1 ([76])**.** *The system* (3.1) *is input left invertible if there exists a nonnegative integer* $r < \infty$ *such that for all mode sequences in* $\mathcal{J}^r$*,*

$$\operatorname{rank}(M_{\sigma(k:k+r)}) - \operatorname{rank}(M_{\sigma(k+1:k+r-1)}) = m \tag{3.7}$$

The Moore-Penrose pseudo inverse generalizes the matrix inversion to non square or non full-rank matrices. More information can be found in [87]. The Moore-Penrose pseudo inverse of a matrix M is denoted by $M^\dagger$. We recall another theorem stated in [76].

Theorem 3.1.2 ([76]). *Assume that* (3.1) *is input left invertible with left inherent delay* r. *The system*

$$\left\{\begin{array}{lcl} \hat{x}_{k+r+1} & = & P_{\sigma(k:k+r)}\hat{x}_{k+r} + B_{\sigma(k)}(M_{\sigma(k:k+r)})^{\dagger}\,(y)_k^{k+r} \\ \hat{u}_{k+r} & = & -(M_{\sigma(k:k+r)})^{\dagger}\mathcal{O}_{\sigma(k:k+r)}\hat{x}_{k+r} + (M_{\sigma(k:k+r)})^{\dagger}\,(y)_k^{k+r} \end{array}\right. \tag{3.8}$$

with

$$P_{\sigma(k:k+r)} = A_{\sigma(k)} - B_{\sigma(k)}(M_{\sigma(k:k+r)})^{\dagger}\mathcal{O}_{\sigma(k:k+r)} \tag{3.9}$$

is a left r*-delay inverse system for (3.1).*

3.1.2 Flat switched linear systems

Section 2.5 recalled the definition of flatness. An interesting matter is to prove that a system is flat or not. In the linear case, the problem reduces to showing that a system is controllable. A flat output characterization has been performed in [88] for continuous-time linear dynamical systems. For nonlinear systems, the equivalence is no longer true. A less intricate problem is to decide whether a given output is a flat output or not. Flat output characterization for switched linear discrete-time systems is the purpose of this section.

Let us define the inverse transition matrix by

$$\begin{array}{rcl} P_{\sigma(k_0)}^{\sigma(k_1)} & = & P_{\sigma(k_1:k_1+r)}P_{\sigma(k_1-1:k_1-1+r)}\cdots P_{\sigma(k_0:k_0+r)} \text{ if } k_1 \geq k_0 \\ & = & \mathbf{1}_n, \text{ if } k_1 < k_0 \end{array} \tag{3.10}$$

Theorem 3.1.3 stated in [76] is recalled. It serves as a basis for our further developments.

Theorem 3.1.3 ([76]). *A componentwise independent output* y_k *of the system* (3.1) *assumed to be square and left input invertible with inherent delay* r*, is a* flat output *if there exists a positive integer* $K < \infty$ *such that, for all mode sequences in* $\mathcal{J}^{r+K}$*, the following equality applies for all* $k \geq 0$*:*

$$P_{\sigma(k+K-1)}^{\sigma(k)} = P_{\sigma(k+K-1:K-1+r)}P_{\sigma(k+K-2:K-2+r)}\cdots P_{\sigma(k:k+r)} = 0 \tag{3.11}$$

$\mathcal{J}^{r+K}$ *stands for the set of mode sequences over the interval of time* $[k, \ldots, k+r+K-1]$.

The point is that the computational cost of the test (3.11) grows exponentially with respect to the number K of matrices involved in (3.11). Besides, no upper bound for K is given. As the condition only involves the left-inverse dynamical matrices of the left r-delay inverse system, we define below an auxiliary system and rewrite Theorem 3.1.3.

3.2 Flatness criterion based on nilpotent semigroups

3.2.1 Auxiliary system

Let us define the auxiliary system of (3.1) as the switched linear system given by

$$q_{k+1} = Q_{\sigma'(k)}q_k \tag{3.12}$$

with $q_k \in \mathbb{R}^n$ and σ' a switching rule defined as follows:
Consider the mapping $\phi : \mathcal{J}^{r+1} \to \mathcal{H} = \{1, \ldots, J^{r+1}\}$ that assigns to each possible sequence $(\sigma(k), \ldots, \sigma(k+r))$ an integer h from the set $\mathcal{H}$ which uniquely identifies the sequence. Then, the switching rule σ' is defined as the function from $\mathbb{N}$ to $\mathcal{H}$ which associates to each integer $k \in \mathbb{N}$ the quantity $\sigma'(k) = \phi(\sigma(k), \ldots, \sigma(k+r)) \in \mathcal{H}$. The value $\sigma'(k)$ is the mode of the auxiliary switched linear system (3.12) and $Q_{\sigma'(k)} = P_{\sigma(k:k+r)}$. We denote by $\mathcal{Q}$ the set of all the matrices Q_h ($h \in \mathcal{H}$).

By considering the auxiliary system (3.12), we are now in position of reformulating Theorem 3.1.3 which turns into

Theorem 3.2.1. *An output y_k of the system* (3.1) *assumed to be square, with left inherent delay r, is a* flat output *if there exists a positive integer $K < \infty$ such that, for all admissible sequences $(\sigma'(k), \ldots, \sigma'(k+K-1)) \in \mathcal{H}^K$, the following equality, involving the product of the dynamical matrices of the auxiliary system* (3.12)*, applies for all $k \geq 0$:*

$$Q_{\sigma'(k+K-1)} Q_{\sigma'(k+K-2)} \cdots Q_{\sigma'(k)} = \mathbf{0} \tag{3.13}$$

Proof. The proof is a straightforward consequence of the definition of the auxiliary system. □

Obviously, this new formulation still suffers from the exponential complexity with respect to the number K of matrices involved in the product. But, it defines a new framework and allows to provide an alternative condition to (3.11) in Section 3.2.2. It is based on the notion of nilpotent semigroups.

3.2.2 Nilpotent semigroup approach

Let us recall the following definitions.

Definition 3.2.1 (Semigroup)**.** A semigroup $\mathcal{S}$ is a set together with an associative internal law. It is said to be finite if it has a finite number of elements.

If $\mathcal{S}$ is a set of matrices, the associative internal law is the matrix multiplication. We denote by 0 the absorbing element of a semigroup when it exists.

Definition 3.2.2 (Nilpotent semigroup [89])**.** A semigroup $\mathcal{S}$ with an absorbing element 0 is said to be nilpotent if there is an integer $t \in \mathbb{N}^*$ such that the internal law applied to any t elements of $\mathcal{S}$ is always equal to 0. The smallest integer t is called the nilpotency class of $\mathcal{S}$.

If $\mathcal{S}$ is a set of matrices, applying the internal law to any t elements of $\mathcal{S}$ amounts to performing the product of t matrices of $\mathcal{S}$. The absorbing element is in this case the null matrix.
A semigroup of matrices is said to be triangularizable if there is a change of basis in which all the elements of the semigroup can be written as upper triangular matrices. Since the eigenvalues of triangular matrices directly appear on the diagonal, the diagonal is null if and only if the matrices are nilpotent. We recall a theorem related to nilpotent semigroups that is useful for their characterization.

Theorem 3.2.2 (Levitsky's theorem [89])**.** *Any semigroup of nilpotent matrices is triangularizable.*

The interest of Theorem 3.2.2 lies in that it provides a tractable method to check whether or not a finite set of matrices generates a nilpotent is semigroup. The paper [90] gives an algorithm that allows to decide whether or not a set of matrices is simultaneously triangularizable. In the remaining, the elements of $\mathcal{S}$ are the matrices of $\mathcal{Q}$ and the law is the matrix multiplication.

Theorem 3.2.3. *If the matrices of $\mathcal{Q}$ of the auxiliary system* (3.12) *generate a nilpotent semigroup, then y_k is a flat output.*

Proof. If the matrices of $\mathcal{Q}$ of the auxiliary system (3.12) generate a nilpotent semigroup, by definition, for any $t-$uple $(h_1, \ldots, h_t) \in \mathcal{H}^t$, t being the class of nilpotency of $\mathcal{Q}$, one has

$$\prod_{i=1}^{t} Q_{h_i} = \mathbf{0} \tag{3.14}$$

Hence, (3.13) is fulfilled with $K = t$. As a result, Theorem 3.2.1 holds and means that y_k is a flat output. □

Corollary 3.2.1. *If the matrices of $\mathcal{Q}$ generate a nilpotent semigroup, the integer K is finite and is upper bounded by the dimension n of the system* (3.1).

Proof. If the matrices of $\mathcal{Q}$ generate a nilpotent semigroup, the integer K is equal to the class of nilpotency t of the semigroup. The class of nilpotency being actually bounded by the dimension of the matrices involved in the semigroup, K is bounded by the dimension of the matrices of $\mathcal{Q}$, which is precisely n, the dimension of the system (3.1). □

Remark 3.2.1. A necessary condition for the matrices of $\mathcal{Q}$ to generate a nilpotent semigroup is that all the matrices of $\mathcal{Q}$ are nilpotent, that is all their eigenvalues are zero. Indeed, (3.14) must hold in particular for h_i constant.

Remark 3.2.2. It is worth pointing out that different sequences $(\sigma(k), \ldots, \sigma(k+r))$ of (3.1) and so different modes $\sigma'(k) = \phi(\sigma(k), \ldots, \sigma(k+r))$ of (3.12) might lead to identical matrices $Q_{\sigma'(k)}$. As a result, $\mathcal{Q}$ contains several times the same element and is a multiset. We should consider only distinct matrices of $\mathcal{Q}$ to reduce the computational cost. We denote by $\mathcal{Z}$ the set of distinct matrices of $\mathcal{Q}$ of cardinality L. Its elements are denoted by Z_l, $l = 1, \ldots, L$. Clearly, $\mathcal{Z} \subseteq \mathcal{Q}$, $L \leq J^{r+1}$ and Theorem 3.2.3 still applies by considering $\mathcal{Z}$ instead of $\mathcal{Q}$.

3.2.3 Equivalence

Let us first point out that the switching rule σ' of the auxiliary system (3.12) is constrained. Indeed, since $\sigma'(k) = \phi(\sigma(k), \ldots, \sigma(k+r))$ and $\sigma'(k+1) = \phi(\sigma(k+1), \ldots, \sigma(k+r+1))$, $\sigma'(k)$ and $\sigma'(k+1)$ depend on the common subsequence $(\sigma(k+1), \ldots, \sigma(k+r))$ and thereby are related one another. Hence, even in the case when the switching rule σ of (3.1) is arbitrary, given a matrix $Q_{\sigma'(k)} = P_{\sigma(k:k+r)}$, the matrix $Q_{\sigma'(k+1)} = P_{\sigma(k+1:k+1+r)}$ is constrained. To formalize this constraint, it is convenient to introduce a so-called set of feasible transitions.

Definition 3.2.3 (Set of feasible transitions)**.** The set $\Gamma(\sigma'(k))$ of feasible transitions from mode $\sigma'(k)$ is the set defined by

$$\Gamma(\sigma'(k)) = \{h \in \mathcal{H},\, h = \phi(\sigma(k+1), \ldots, \sigma(k+r+1)), \forall \sigma(k+r+1) \in \mathcal{J}\} \tag{3.15}$$

In other words, $\Gamma(\sigma'(k))$ is the set $h \in \mathcal{H}$ which can be reached when $\sigma(k+r+1)$ varies over the whole range $\mathcal{J}$, $\sigma'(k)$ and so the sequence $(\sigma(k+1), \ldots, \sigma(k+r))$ being imposed. One has $\Gamma(\sigma'(k)) \subseteq \mathcal{H}$ which formalizes that σ' is constrained. It is clear that $\Gamma(\sigma'(k))$ can never be the empty set.

Definition 3.2.4 (Admissible sequence)**.** A sequence $(h_0, h_1, \ldots)$ is said admissible if for any $i \geq 0$

$$h_{i+1} \in \Gamma(h_i) \tag{3.16}$$

Let us introduce the map $\mu : \mathcal{H} \to \mathcal{Q}$ which assigns to each integer $h \in \mathcal{H}$ the matrix $Q_h \in \mathcal{Q}$. The restriction of μ to a particular subset $\Gamma(h)$ of $\mathcal{H}$ is denoted by $\mu_{\Gamma(h)}$. Let Im denote the image of a function.

Definition 3.2.5 (Admissible sequence of matrices)**.** A sequence of matrices $(Q_{h_0}, Q_{h_1}, \ldots)$ $(h_i \in \mathcal{H})$ is said admissible if, for any $h_i \in \mathcal{H}$,

$$Q_{h_{i+1}} \in \mathrm{Im}(\mu_{\Gamma(h_i)}) \tag{3.17}$$

The following proposition applies.

Proposition 3.2.1. *The conditions* (3.13) *and* (3.14) *are equivalent if and only if*

$$\forall h_i \in \mathcal{H},\, \mathrm{Im}(\mu_{\Gamma(h_i)}) = \mathcal{Q} \tag{3.18}$$

Proof. The statement (3.14) ⇒ (3.13) is always true regardless of the condition (3.18). Still, it must be shown that (3.13) implies (3.14) provided that (3.18) is fulfilled. The condition $\forall h_i,\ \mathrm{Im}(\mu_{\Gamma(h_i)}) = \mathcal{Q}$ means that, for any arbitrary mode $h_i \in \mathcal{H}$, $Q_{h_{i+1}}$ can be any matrix in $\mathcal{Q}$. Hence, for any $t-$uple $(h_1, \ldots, h_t)$, the sequence $(Q_{h_1}, \ldots, Q_{h_t})$ is an admissible sequence for (3.12). Finally, the set of products $Q_{h_1} \cdots Q_{h_t}$ for all $t-$uples $(h_1, \ldots, h_t)$ coincides with the set of products (3.13) for all $k \geq 0$. That completes the proof. □

Similarly to Remark 3.2.2, if different sequences $(\sigma(k), \ldots, \sigma(k+r))$ and so, different $\sigma'(k)$, lead to identical matrices $Q_{\sigma'(k)}$, Proposition 3.2.1 still applies with a lower computational cost if the multiset $\mathcal{Q}$ is replaced by the set $\mathcal{Z}$ of distinct elements of $\mathcal{Q}$. Hence, we consider hereafter the set $\mathcal{Z}$.

Remark 3.2.3. From (3.9), it can be seen that (3.18) is always satisfied for at least two particular cases: when the inherent delay r is equal to zero or if it is equal to one and that the output matrix C does not depend on σ. These two cases encompass a large class of systems and we illustrate in Section 3.2.6 that Proposition 3.2.1 also applies in other cases.

3.2.4 Computational issues

In this section, we propose an algorithm that checks whether or not a set of matrices generates a nilpotent semigroup, that is if Theorem 3.2.3 is fulfilled. It is shown that it has a polynomial complexity and is theoretically motivated by Theorem 3.2.2. In other words, all the matrices of a same nilpotent semigroup can be rewritten as upper triangular matrices with zeros on the diagonal up to a common change of basis. The consequence of this theorem is central for our purpose. Indeed, determining whether or not the matrices of $\mathcal{Z}$ generate a nilpotent semigroup amounts to checking whether or not $\mathcal{Z}$ can be simultaneously triangularized. It is a necessary and sufficient condition. The approach we propose to check Theorem 3.2.2 is inspired from the general triangularization method provided in [90] and corresponds to Algorithm 1. Some peculiarities that apply to our special case are addressed finally leading to a fully-specified algorithm for flat output characterization.

input : A set $\{Z_l\}$, $l \in \{1, \ldots, L\}$ of $n \times n$ matrices
output: A basis S that triangularizes the matrices if it exists

initialization;
for $l \leftarrow 1$ **to** L **do**
  $T_l \leftarrow Z_l$;
end
$S_1 \leftarrow \mathbf{0}_{n\times 0}$;
$S_2 \leftarrow \mathbf{1}_n$;
for $i \leftarrow 1$ **to** $n-1$ **do**
  $v_i \leftarrow$ an eigenvector common to the matrices T_l, $l \in \{1, \ldots, L\}$;
  if v_i *does not exist* **then**
    no simultaneous triangularization basis exists;
    return;
  end
  $w_i \leftarrow S_2 v_i$;
  $S_1 \leftarrow \begin{pmatrix} S_1 & w_i \end{pmatrix}$;
  $S_2 \leftarrow$*matrix whose column vectors are vectors that extend* S_1 *to a basis*;
  $S \leftarrow \begin{pmatrix} S_1 & S_2 \end{pmatrix}$;
  $I_1 \leftarrow \begin{pmatrix} \mathbf{0}_i & \mathbf{1}_{n-i} \end{pmatrix}$;
  $I_2 \leftarrow \begin{pmatrix} \mathbf{0}_i \\ \mathbf{1}_{n-i} \end{pmatrix}$;
  $S^{-1} \leftarrow$ inverse of S;
  for $l \leftarrow 1$ **to** L **do**
    $T_l \leftarrow I_1 S^{-1} Z_l S I_2$;
  end
end
return S;

Algorithm 1: Simultaneous triangularization algorithm

The following comments are in order:

- Line 2 to Line 6 correspond to the initialization. Matrices T_l ($l = 1, \ldots, L$) play the role of the auxiliary variables and are initialized at the beginning with the matrices Z_l of the set $\mathcal{Z}$;
- At most $n-1$ successive loops from Line 7 to Line 23 are performed as stressed by Corollary 3.2.1;
- Line 8 corresponds to the first step of a given loop i. It consists in finding out an eigenvector v_i which is common to the matrices T_l, $l = 1, \ldots, L$). Consequently, in the first loop, v_1 is an eigenvector common to the matrices Z_l of $\mathcal{Z}$. It is worth pointing out that if this step fails in the loop i, it means that no eigenvector v_i common to the matrices T_l, $l = 1, \ldots, L$ exists and the algorithm stops. Levitsky's theorem (Theorem 3.2.2) stating a necessary and sufficient condition, it can be concluded that $\mathcal{Z}$ does not generate a nilpotent semigroup and that the condition of Theorem 3.2.3 is not fulfilled;
- Line 14 describes the fact that the final change of basis S is built column after column. Each new vector v_i carried out in the loop v_i is added (actually after a change of basis notified at Line 13) resulting in a matrix $S_1 = (v_1 \cdots v_i)$. When $i = n$ then, $S = (S_1\ S_2)$, S_1 and S_2 resulting from the loop $n-1$;
- At Line 15, S_1 must be extended to a basis. By "extended", it is meant to find a set of $n-i$ vectors w_j so that $S = (S_1\ w_1 \cdots w_{n-i})$ is full rank. The matrix S_2 is precisely $S_2 = (w_1 \cdots w_{n-i})$;
- Line 21 performs the current change of basis S to the matrices Z_l. The multiplications by I_1 and I_2 merely correspond to the extraction of a square matrix of dimension $n-i$ from the matrix $S^{-1}Z_lS$. That is, the first i rows and columns of $S^{-1}Z_lS$ are removed. A new set of matrices T_l is thereby obtained. A new loop can restart from Line 7.

The algorithm is general and also considers non-nilpotent matrices. In our context, the algorithm is only used when the elements of $\mathcal{Z}$ are nilpotent. As a consequence, we can particularize it to our situation. The determination of a common eigenvector at Line 8 and the extension to a basis at Line 15 can be done as follows.

3.2.4.1 Determination of a common eigenvector

According to Remark 3.2.1, a necessary condition for the set $\mathcal{Z}$ to generate a nilpotent semigroup is that all the matrices Z_l of $\mathcal{Z}$ are nilpotent. That is, all their eigenvalues are zero. The set $\mathcal{T} = \{T_l,\ l = 1, \ldots, L\}$ corresponds exactly to $\mathcal{Z}$ at the initialization and is updated at each loop i through a linear change of basis S at Line 21. The eigenvalues are preserved. Hence, the eigenvalues of the T_l's are all zero whatever the loop $i \geq 0$ is. Consequently, for any v_i, $i = 1, \ldots, n$ and $\forall T_l \in \mathcal{T}$, $T_l v_i = 0$ holds. Hence, v_i is a non zero solution of

$$Tv_i = \mathbf{0} \text{ with } T = \begin{pmatrix} T_1 \\ \vdots \\ T_L \end{pmatrix} \tag{3.19}$$

As a result, v_i is a non zero vector of the null space of T denoted by $\ker(T)$.

3.2.4.2 Extension to a basis

At Line 15, S_1 must be extended to a basis. Let $\cdot|\cdot$ denote the concatenation operator. We must thereby find out a set of $n-i$ vectors w_j so that $S = (S_1|w_1|\cdots|w_{n-i})$ is full rank. It can be obtained by determining a basis of the kernel of the transpose of S_1. In other words, $S_2 = (w_1|\cdots|w_{n-i})$ can be any basis of $\ker({}^tS_1)$ where ${}^tS_1$ stands for the transpose of S_1.

3.2.5 Complexity

All the operations can easily be performed by software involving usual built-in functions. As an example, we give the corresponding Matlab source in Appendix C. The complexity of the condition (3.11) in Theorem 3.1.3 is $O(J^{r+K}Kn^3)$. The problem lies in that the complexity is exponential with respect to

the number of matrices K involved in the product, which can be large. Let us estimate the computational cost of the flat output characterization approach stated in Theorem 3.2.3 based on nilpotent semigroups. To this end, we examine Algorithm 1. Considering a given loop, the most complex operations are performed at Lines 8, 15, 19 and 21. Lines 8 and 15 consist in determining the kernel of a matrix. It is usually based on singular value decomposition of a $\mathbb{R}^{a\times b}$ matrix for which known algorithms with complexity $O(4ab^2+8b^3)$ exist. Line 19 has complexity $O(n^3)$ with the usual algorithms. Line 21 is a change of basis. The multiplications by I_1 and I_2 can be omitted since the effect is merely to extract a square matrix of dimension $n-i$ from the matrix $S^{-1}Z_lS$ so it can be done much more efficiently than by a matrix multiplication. Therefore, the two operations to be considered are the two matrix multiplications of Z_l by S and S^{-1}. Matrix multiplications have complexity at most $O(n^3)$. Therefore, the complexity of Lines 20 to 22 is $O(Ln^3)$ or $O(J^{r+1}n^3)$ due to the inequality $L\leq J^{r+1}$ (recall Remark 3.2.2). This part of the code is the one with the largest complexity. The operations are repeated over at most n loops. Therefore, the global complexity of Algorithm 1 is $O(J^{r+1}n^4)$. It is an improvement insofar as the complexity is no longer exponential with respect to the parameter K.

3.2.6 Example

Consider the SISO switched linear system of the form (3.1). The dimension is $n=4$, the switching rule σ, not detailed here, is assumed to deliver arbitrary sequences and the number of modes is $J=3$. According to the mode, the state space matrices numerically read

$$A_1=\begin{pmatrix}-1&-0.5&-0.5&0\\1&1.5&1.5&0\\1&0.5&0.5&1\\1&0&1&0\end{pmatrix},\ A_2=\begin{pmatrix}-1&-1&-1&-0.5\\1&2&2&0.5\\1&1&1&1.5\\1&0&1&0\end{pmatrix},\ A_3=\begin{pmatrix}-1&-2.5&-2.5&-2\\1&3.5&3.5&2\\1&2.5&2.5&3\\1&0&1&0\end{pmatrix}$$

$$B_1=B_2=B_3={}^t\begin{pmatrix}0&0&0&1\end{pmatrix}$$

$$C_1=C_2=C_3=\begin{pmatrix}2&1&1&0\end{pmatrix}$$

$$D_1=D_2=D_3=0$$

Since it is a SISO system, that for all $i\in\mathcal{J}$, $D_i=0$, for all $(i,j)\in\mathcal{J}^2$, $C_iB_j=0$ and that for all $(i,j,l)\in\mathcal{J}^3$, $C_iA_jB_l\neq 0$, the inherent delay is $r=2$. Let us derive the corresponding auxiliary system as defined in Section 3.2.1. To this end, we must define the mapping ϕ. The number of possible sequences over any interval of time $[k:k+r]$ is $J^{r+1}=3^{2+1}=27$ and

$$\begin{array}{ll}\phi((1,1,1))=1 & \cdots\\ \phi((1,1,2))=2 & \phi((3,3,1))=25\\ \phi((1,1,3))=3 & \phi((3,3,2))=26\\ \cdots & \phi((3,3,3))=27\end{array}$$

The sets Γ of feasible transitions are defined by

$$\begin{array}{ll}\Gamma(1)=(1,2,3) & \cdots\\ \Gamma(2)=(4,5,6) & \Gamma(26)=(23,24,25)\\ \cdots & \Gamma(27)=(25,26,27)\end{array}$$

It turns out that the multiset $\mathcal{Q}$ has $L=3$ distinct matrices and the matrices Z_l of the corresponding set $\mathcal{Z}$ numerically read

$$Z_1=\begin{pmatrix}-1&-0.5&-0.5&0\\1&1.5&1.5&0\\1&0.5&0.5&1\\-2&-2&-2&-1\end{pmatrix},\ Z_2=\begin{pmatrix}-1&-1&-1&-0.5\\1&2&2&0.5\\1&1&1&1.5\\-2&-3&-3&-2\end{pmatrix},\ Z_3=\begin{pmatrix}-1&-2.5&-2.5&-2\\1&3.5&3.5&2\\1&2.5&2.5&3\\-2&-6&-6&-5\end{pmatrix}$$

One has

$$\mu(\Gamma(1))=\cdots=\mu(\Gamma(27))=(Z_1,Z_2,Z_3)$$

and so, Proposition 3.2.1 is fulfilled.

Finally, it turns out that Algorithm 1 succeeds and returns the following change of basis S

$$S = \begin{pmatrix} 0 & 0.3780 & -0.9258 & 0 \\ 0.7071 & -0.3780 & -0.1543 & 0.5774 \\ -0.7071 & -0.3780 & -0.1543 & 0.5774 \\ 0 & 0.7559 & 0.3086 & 0.5774 \end{pmatrix}$$

As a consequence, based on Theorem 3.2.2 and Theorem 3.2.3, we conclude that y_k is a flat output.

3.2.7 Connection with common algebra notions

In this section, we recall the connection between nilpotent semigroups and other algebra notions commonly used in automatic control. Figure 3.1 illustrates the relations between algebraic structures whose generators have specific properties. The definitions of the different algebraic structures are provided in Appendix B. The following comments are in order

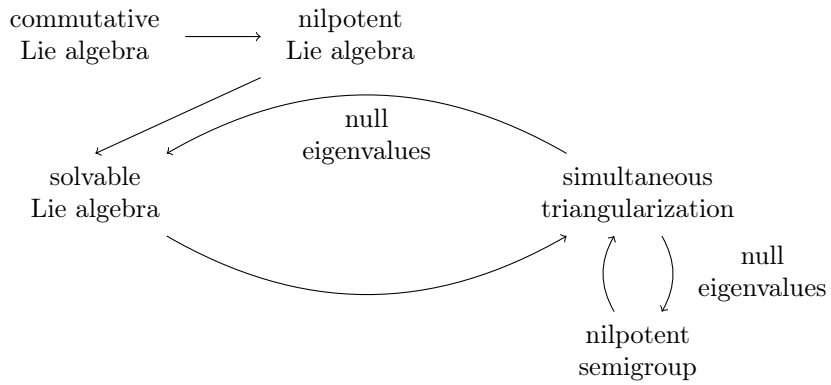


Figure 3.1: Connection between different algebra notions

1. A commutative Lie algebra is a nilpotent Lie algebra;
2. A nilpotent Lie algebra is a solvable Lie algebra;
3. A solvable Lie algebra is simultaneously triangularizable [91];
4. A set of matrices that are simultaneously triangularizable together with the Lie bracket and matrix addition and multiplication operations generate a solvable Lie algebra if there eigenvalues are null. The null eigenvalue condition is only sufficient;
5. A set of simultaneously triangularizable matrices generates a nilpotent semigroup if there eigenvalues are null;
6. A set of matrices that generates a nilpotent semigroup is simultaneously triangularizable. The justification is Theorem 3.2.2.

Remark 3.2.4. It is shown in [92] that, if the dynamical matrices of a switched linear continuous-time system are simultaneously triangularizable and that they are Hurwitz, they admit a quadratic Lyapunov function. Hence, the corresponding system is quadratically stable.

3.3 Flatness criterion extended to LPV systems

The results stated in the previous sections can be interestingly extended in a rather straightforward way to LPV systems.

3.3.1 Criterion

The system (3.1) can be viewed as an LPV system as soon as we consider that the switching rule σ is replaced by a function which takes values in a continuum. If so, the sets $\mathcal{J}$, $\mathcal{H}$, $\Gamma(\sigma'(k))$, $\mathcal{Q}$ and $\mathcal{Z}$ must be considered as uncountable sets. The $r-$delay inverse system (3.8), the auxiliary system (3.12) together with the mapping ϕ still make sense as soon as σ' is considered as a function, similarly to σ, taking values in a continuum. Besides and most importantly, it turns out that both, semigroups (Definition 3.2.1) and nilpotent semigroups (Definition 3.2.2), are still well defined for an uncountable set $\mathcal{S}$. As a result, Theorem 3.2.3 still applies.

On the other hand, Levitsky's theorem (Theorem 3.2.2), which allows for checking whether Theorem 3.2.3 is fulfilled, applies for any semigroup, including semigroups with infinite cardinality, which is precisely the case here. It is recalled that Levitsky's Theorem asserts that "$\mathcal{Z}$ generates a nilpotent semigroup if the matrices of $\mathcal{Z}$ can be simultaneously triangularized". The key point is that, considering $\mathcal{Z}$ as an uncountable set of matrices, it can be reformulated in a strictly equivalent way stating that "the matrices $Z_{\sigma'(k)}$ must be simultaneously triangularized with a change of basis that does not depend on $\sigma'(k)$". Taking into account the aforementioned considerations and combining Theorem 3.2.2 and Theorem 3.2.3, the following theorem holds for characterizing flat outputs of LPV systems

Theorem 3.3.1. *If the matrices of $\mathcal{Z}$ of the auxiliary system* (3.12) *can be simultaneously triangularized independently of $\sigma'(k)$ then, y_k is a flat output.*

Algorithm 1 still applies up to some minor modifications. The loops at Line 2 and Line 20 can be removed or equivalently, L can be set to $L = 1$. Besides, the determination of a common eigenvector at Line 8 that is, the search for a nonzero vector v_i of $\ker(T)$ as explained in Section 3.2.4.1 turns into the search for a nonzero vector v_i of $\ker(T_1)$ (since $L = 1$ and so $T = T_1$) independent of $\sigma'(k)$. Line 8 has to be replaced by

```
1 v_i ← one eigenvector of T_1 independent of σ'(k);
```

The flat outputs characterization based on nilpotent semigroups for LPV systems is valuable for two major reasons. First, flat outputs characterization of LPV systems has never been addressed so far in the literature. Secondly, the characterization through (3.11) or equivalently (3.13) cannot be done for LPV systems since it requires to check an infinite number of products. As a matter of fact, σ taking values in a continuum, the number of sequences (σ) in Theorem 3.1.3 or sequences (σ') in Theorem 3.2.3 would be infinite.

3.3.2 Example

We investigate an LPV system given by the following form

$$\left\{ \begin{array}{lcl} x_{k+1} & = & A_{\sigma(k)}x_k + B_{\sigma(k)}u_k \\ y_k & = & C_{\sigma(k)}x_k + D_{\sigma(k)}u_k \end{array} \right. \tag{3.20}$$

where $x_k \in \mathbb{R}^4$, $u_k \in \mathbb{R}$, $y_k \in \mathbb{R}$. In the framework of LPV systems, the notation $A(\rho_k)$ is often used and refers to matrices which depend on a time-varying parameter ρ_k. Hence here, by $A_{\sigma(k)}$, it must be understood a matrix A which depends on a time-varying parameter $\sigma(k)$ with $\sigma(k)$ taking values in a continuum. The notation $\sigma^i(k)$ refers to the ith component of $\sigma(k)$. The same consideration holds for $B_{\sigma(k)}$, $C_{\sigma(k)}$, $D_{\sigma(k)}$. The matrices numerically read

$$A_{\sigma(k)} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \sigma^{(1)}(k) & \sigma^{(2)}(k) & 1 & 0 \end{pmatrix}, B_{\sigma(k)} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, C_{\sigma(k)} = \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}, D_{\sigma(k)} = 0$$

Since it is a SISO system and that $C_{\sigma(k+1)}B_{\sigma(k)} = 1$ independently of σ, the relative degree is $R = 1$. The set $\mathcal{Q}$ of the dynamical matrices of the left-inverse system (3.8) are the matrices $Q_{\sigma'(k)} = P_{\sigma(k:k+1)} = A_{\sigma(k)} - B_{\sigma(k)}C_{\sigma(k)}A_{\sigma(k)}$. They read for all $\sigma'(k)$ (which can be identified to $(\sigma(k), \sigma(k+1))$)

$$Q_{\sigma'(k)} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ \sigma^1(k) & \sigma^2(k) & 1 & 0 \end{pmatrix}$$

It turns out that Theorem 3.3.1 is fulfilled with a triangularization basis S which numerically reads

$$S = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Finally, since $Q_{\sigma'(k)}$ only depends on σ' at time k, it is clear that condition (3.18) of Proposition 3.2.1 is fulfilled (and so applies beyond the particular cases mentioned in Remark 3.2.3) meaning that conditions (3.13) and (3.14) are equivalent. The interest of the result stated in this paper is that, (3.14) provides an alternative to (3.13), hence solving the intractability of (3.13) for LPV systems.

3.4 Design of self-synchronizing systems

It has been shown in Section 2.5 that, from the synchronization point of view, designing a self-synchronizing setup amounts to designing a flat dynamical system. However, the flatness criteria requires to determine the left-inverse system and to do an *a posteriori* analysis which is not very constructive. Therefore, we propose another strategy which consists in designing first the dynamical system that acts as the decryptor $\mathcal{D}$ and then, to derive the equation of the corresponding encryptor $\mathcal{E}$. Since we design the decryptor first, we can define it so that it matches the flatness criteria. As a result, when we derive the equations of the encryptor, they correspond to the ones of a flat system. Interestingly, from a control theory point of view, the encryptor is nothing but the right inverse system of the decryptor. In accordance with Section 2.3, we recall that in the cryptographic context, we denote by m the input of the master, by c its output and by $\hat{m}$ the output of the slave.

3.4.1 Constructive approaches

Consider a setup described at the transmitter part by

$$\mathcal{E}\begin{cases} x_{k+1} &= A_{\sigma(k)}x_k + B_{\sigma(k)}m_k \\ c_k &= C_{\sigma(k)}x_k + D_{\sigma(k)}m_k \end{cases} \tag{3.21}$$

and at the receiver part by

$$\mathcal{D}\begin{cases} \hat{x}_{k+1} &= \hat{A}_{\sigma(k)}\hat{x}_k + \hat{B}_{\sigma(k)}c_k \\ \hat{m}_k &= \hat{C}_{\sigma(k)}\hat{x}_k + \hat{D}_{\sigma(k)}c_k \end{cases} \tag{3.22}$$

3.4.1.1 Finite-time self-synchronizing construction

Theorem 3.4.1. *The setup* (3.21)–(3.22) *is finite-time self-synchronizing whenever the three following conditions are fulfilled:*

- $\forall j \in \mathcal{J},\ \hat{D}_j \neq 0$ (3.23)
- $\exists K \in \mathbb{N}, \forall x_0,\ \hat{x}_0,\ \forall\,(v) \in \mathcal{J}^K,\ \prod_{i=0}^{K-1} \hat{A}_{v_i} = 0$ (3.24)

- *Given the matrices* $(\hat{A}_j, \hat{B}_j, \hat{C}_j, \hat{D}_j)$ *of* $\mathcal{D}$*, the system* $\mathcal{E}$ *reads*

$$\begin{cases} x_{k+1} &= \left(\hat{A}_{\sigma(k)} - \hat{B}_{\sigma(k)}(\hat{D}_{\sigma(k)})^{-1}\hat{C}_{\sigma(k)}\right) x_k + \hat{B}_{\sigma(k)}(\hat{D}_{\sigma(k)})^{-1} m_k \\ c_k &= -(\hat{D}_{\sigma(k)})^{-1}\hat{C}_{\sigma(k)} x_k + (\hat{D}_{\sigma(k)})^{-1} m_k \end{cases} \tag{3.25}$$

Proof. Since $\hat{D}_j \neq 0$ for any $j \in \mathcal{J}$ (Condition (3.23)), the input m_k can be derived from the output equation of (3.25) and reads

$$m_k = \hat{D}_{\sigma(k)} c_k + \hat{C}_{\sigma(k)} x_k \tag{3.26}$$

Thus,

$$\begin{aligned} \hat{m}_k - m_k &= \hat{C}_{\sigma(k)}\hat{x}_k + \hat{D}_{\sigma(k)} c_k - \hat{D}_{\sigma(k)} c_k - \hat{C}_{\sigma(k)} x_k \\ &= \hat{C}_{\sigma(k)}(\hat{x}_k - x_k) \end{aligned}$$

Let the reconstruction error be $\varepsilon_k = \hat{x}_k - x_k$. Then, from (3.22) and (3.25)

$$\begin{aligned} \varepsilon_{k+1} &= \hat{A}_{\sigma(k)}\hat{x}_k + \hat{B}_{\sigma(k)} c_k - (\hat{A}_{\sigma(k)} - \hat{B}_{\sigma(k)}(\hat{D}_{\sigma(k)})^{-1}\hat{C}_{\sigma(k)}) x_k - \hat{B}_{\sigma(k)}(\hat{D}_{\sigma(k)})^{-1} m_k \\ &= \hat{A}_{\sigma(k)}\varepsilon_k - \hat{B}_{\sigma(k)}(c_k - (\hat{D}_{\sigma(k)})^{-1} m_k) - \hat{B}_{\sigma(k)}(\hat{D}_{\sigma(k)})^{-1} m_k + \hat{B}_{\sigma(k)} c_k \\ &= \hat{A}_{\sigma(k)}\varepsilon_k \end{aligned} \tag{3.27}$$

After iterating (3.27) K times and taking into account (3.24), $\varepsilon_k = 0$ or equivalently $x_k = \hat{x}_k$ for any $k \geq K$. Hence, according to Definition 2.2.4, the set-up (3.21)–(3.22) is finite-time self-synchronizing. □

No constraint is imposed on $\hat{B}_j$ and $\hat{C}_j$. Condition (3.24) means that, regardless of the order of multiplication of the matrices $\hat{A}_j$, and so for any mode sequence, the product is zero after a finite number K of iterations. The number K is the delay of synchronization.

Remark 3.4.1. The condition $\hat{D}_j \neq 0$ for any $j \in \mathcal{J}$ means that the relative degree of the systems $\mathcal{E}$ and $\mathcal{D}$ is zero.

Remark 3.4.2. The system (3.21) is a right inverse for the system (3.22). Indeed, according to the definition of a right inverse, for any identical conditions $x_0 = \hat{x}_0$ and for any identical mode sequence (v), the system (3.21) drives (3.22) such that $\forall k \geq 0$, $\hat{m}_k = m_k$.

Theorem 3.4.1 does not provide a constructive solution for the selection of appropriate matrices $\hat{A}_j$ which must fulfil the constraint (3.24). The purpose of the next paragraph is to obtain an equivalent constructive condition. It is based on the notion of nilpotent semigroups.

Proposition 3.4.1. *In order for* (3.24) *to be fulfilled, the set of dynamical matrices* $\{\hat{A}_j,\, j \in \mathcal{J}\}$ *must generate a nilpotent semigroup. The delay of synchronization* K *equals the class of nilpotency of this semigroup.*

Remark 3.4.3. The product of t nilpotent matrices which commute pairwise is 0 but the product of t nilpotent matrices is not, in general, nilpotent. Indeed, we observe that $\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$. Theorem 3.2.2 provides a generalization of this special case, should each matrix be nilpotent is only a necessary condition.

Hence, based on Levitsky's theorem, the construction of the family $\{\hat{A}_j,\, j \in \mathcal{J}\}$ which fulfils (3.24) follows three successive steps:

1. Choose an invertible matrix $T \in \mathbb{F}^{n\times n}$;
2. Choose a set of J upper triangular matrices $\hat{\tilde{A}}_j$ with zero on the diagonal;
3. For all $j \in \mathcal{J}$, compute $\hat{A}_j = T^{-1}\hat{\tilde{A}}_j T$.

The matrix T may possibly be the identity matrix.

Remark 3.4.4. Because of Levitzky's theorem, the consideration of a semigroup of n–dimensional matrices is equivalent to the consideration of the corresponding set of upper triangular matrices. And yet, for triangular matrices, it is clear that the nilpotency class is at most n. As a result, the delay of synchronization K is upper bounded by n.

3.4.1.2 Statistical self-synchronizing construction

So far, we have proposed a construction which guarantees self-synchronization with a finite delay K. This assumption limits the complexity of the encryption process which can 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 a random variable with a probability law that reaches one as time reaches infinity. It could be interesting to relax the finite-time synchronization constraint so that the synchronization probability follows a probability law that ensures synchronization for a large enough sequence (c). Let us stress that over the field of real numbers, we could have relaxed the finite-time synchronization constraint by allowing asymptotical synchronization with prescribed exponential decay rate. Over finite fields as it is the case here, asymptotical synchronization does no longer make sense.

General conditions We give an equivalent theorem to Theorem 3.4.1 that corresponds to this situation. Note that the only difference with Theorem 3.4.1 concerns (3.24) which turns into (3.29).

Theorem 3.4.2. *The set-up* (3.21)–(3.22) *is statistically self-synchronizing whenever the three following conditions are fulfilled:*

- $\forall j \in \mathcal{J},\ \hat{D}_j \neq 0$ (3.28)
- $\exists K \in \mathbb{N},\ \forall x_0,\ \widehat{x}_0,\ \exists\,(v) \in \mathcal{J}^K,\ \prod_{i=0}^{K-1} \hat{A}_{v_i} = 0$ (3.29)
- *Given the pairs* $\{\hat{A}_j, \hat{D}_j\}$ *fulfilling* (3.28) *and* (3.29) *and arbitrary pairs* $\{\hat{C}_j, \hat{D}_j\}$ *of* $\mathcal{D}$, *the system* $\mathcal{E}$ *reads*

$$\begin{cases} x_{k+1} &= \left(\hat{A}_{\sigma(k)} - \hat{B}_{\sigma(k)}(\hat{D}_{\sigma(k)})^{-1}\hat{C}_{\sigma(k)}\right) x_k + \hat{B}_{\sigma(k)}(\hat{D}_{\sigma(k)})^{-1} m_k \\ c_k &= -(\hat{D}_{\sigma(k)})^{-1}\hat{C}_{\sigma(k)} x_k + (\hat{D}_{\sigma(k)})^{-1} m_k \end{cases} \tag{3.30}$$

Proof. The proof follows the same development than the one of Theorem 3.4.1 till Equation (3.27). Equation (3.29) means that there exists a finite sequence of length K so that the product of K matrices $\hat{A}_j$ is zero. Considering that any finite sequence has the probability one to appear in an infinite sequence (provided that any symbol has a non null probability of occurrence), satisfying (3.29) implies satisfying Equation (2.13) of Definition 2.2.5. □

Likewise the finite-time self-synchronization case, it should be interesting to check for constructive conditions, equivalent to (3.29), but with additionally the ability of controlling the probability of the synchronization delay while designing the system. Again, it turns out that we can resort to nilpotent semigroups.

Nilpotent semigroups for statistical self-synchronization The proposed construction considers ℓ distinct nilpotent semigroups $\mathcal{S}_i$, $i \in \{1, \ldots, \ell\}$ of square n dimensional matrices each generated by the corresponding set of matrices $\mathcal{N}_i$. The cardinality of the set $\mathcal{N}_i$, $i \in \{1, \ldots, \ell\}$ is denoted by J_i. The construction of the family $\{\hat{A}_j,\ j \in \mathcal{J}\}$ which fulfils (3.29) obeys the three following steps:

1. For each nilpotent semigroup $\mathcal{S}_i$, $i \in \{1, \ldots, \ell\}$ to be built, choose a distinct invertible matrix $T_i \in \mathbb{F}^{n \times n}$;
2. For $i \in \{1, \ldots, \ell\}$, choose a set of J_i upper triangular matrices $\hat{\tilde{A}}_{j'}$ $j' \in \{1, \ldots, J_i\}$ with zeros on the diagonal;

3. For $i \in \{1, \ldots, \ell\}$, for $j' \in \{1, \ldots, J_i\}$, compute $\hat{A}_j = T_i^{-1} \hat{\tilde{A}}_{j'} T_i$ with $j = (i-1)\ell + j'$. These matrices are the elements of the set $\mathcal{N}_i$.

Let t_i be the class of nilpotency of $\mathcal{S}_i$. The synchronization of $\mathcal{E}$–$\mathcal{D}$ is ensured if the switching rule σ selects t_i successive modes in the same nilpotent semigroup $\mathcal{S}_i$.

Remark 3.4.5. It is worth emphasizing that finite-time synchronization is a special case of statistical self-synchronization which corresponds to $\ell = 1$, $J_1 = J$.

Synchronization probability When considering statistical self-synchronization, a question of interest is the shape of the synchronization probability function. Such a system is viable only if, for sequences of reasonable length, the synchronization probability is close to one. As explained in Section 6.1, for security purposes, it is also important that the synchronization does not occur too quickly.
The parameters that can be used to control the synchronization delay while designing the system are essentially, the dimension n of the system, the number ℓ of nilpotent semigroups $\mathcal{S}_i$, the number of generators J_i and the class of nilpotency t_i of $\mathcal{S}_i$. Section 3.4.5.2 illustrates such a purpose.

3.4.2 Connection with flatness

Proposition 3.4.2. *The system $\mathcal{E}$ resulting from the conditions* (3.23)–(3.24)–(3.25) *is flat with flat output c_k.*

Proof. At time $k+K$, the state of the switched system (3.22) is

$$\hat{x}_{k+K} = \prod_{i=0}^{K-1} \hat{A}_{\sigma(k+K-1-i)} \hat{x}_k \sum_{i=0}^{K-1} \left[\prod_{j=i+1}^{K-1} \hat{A}_{\sigma(k+K-j)} \right] \hat{B}_{\sigma(k+i)} c_{k+i}$$

Therefore, if (3.24) holds, any state at time $k \geq 0$ reads:

$$\hat{x}_{k+K} = \sum_{i=0}^{K-1} \left[\prod_{j=i+1}^{K-1} \hat{A}_{\sigma(k+K-j)} \right] \hat{B}_{\sigma(k+i)} c_{k+i} \tag{3.31}$$

And yet, $\epsilon_k = 0$ or equivalently $x_k = \hat{x}_k$ for any $k \geq K$. Hence, after a shift of K, the following equality holds

$$\hat{x}_k = x_k = \sum_{i=0}^{K-1} \left[\prod_{j=i+1}^{K-1} \hat{A}_{\sigma(k-j)} \right] \hat{B}_{\sigma(k-K+i)} c_{k-K+i} \tag{3.32}$$

which gives the function F of (2.24).
On the other hand, since $\hat{D}_j \neq 0$ for any $j \in \mathcal{J}$, the input m_k reads like (3.26). Substituting the expression (3.32) of x_k into (3.26) gives the function G of (2.24). That completes the proof. □

As a result, and as pointed out in Section 2.5, there is an equivalence between the recursive part of both equations (3.21)–(3.22) and the function F of (2.24). The equivalence applies under flatness condition. In the special case of switched linear systems, (3.21)–(3.22) can be equivalently rewritten into the respective canonical forms (2.19a)–(2.19b)

$$\begin{cases} x_k &= \sum_{i=0}^{K-1} \left[\prod_{j=i+1}^{K-1} \hat{A}_{\sigma(k-j)} \right] \hat{B}_{\sigma(k-K+i)} c_{k-K+i} \\ c_k &= C_{\sigma(k)} x_k + D_{\sigma(k)} m_k \end{cases} \tag{3.33}$$

$$\begin{cases} \hat{x}_k &= \sum_{i=0}^{K-1} \left[\prod_{j=i+1}^{K-1} \hat{A}_{\sigma(k-j)} \right] \hat{B}_{\sigma(k-K+i)} c_{k-K+i} \\ \hat{m}_k &= \hat{C}_{\sigma(k)} \hat{x}_k + \hat{D}_{\sigma(k)} c_k \end{cases} \tag{3.34}$$

It is worth pointing out that, from a computational point of view, the recursive form (3.21)–(3.22) is more relevant than (3.33)–(3.34).

3.4.3 Transmission zeros and surjectivity

An important requirement is that all the states are reachable. Indeed, for cryptographic perspectives, the state of the dynamical system must not stay confined in some subspace of the state space. For this reason, it is relevant that the maps $x_k \longmapsto A_j x_k$, $j \in \mathcal{J}$ are surjective. In other words, we want to guarantee that

$$\forall j \in \mathcal{J},\ \operatorname{rank}(A_j) = n \tag{3.35}$$

The problem lies in that, according to Theorem 3.4.1, the matrices A_j, B_j, C_j, D_j of the system $\mathcal{E}$ are not designed directly but are derived from $\hat{A}_j$, $\hat{B}_j$, $\hat{C}_j$, $\hat{D}_j$ of $\mathcal{D}$. Hence, we must find out a condition on the matrices $\hat{A}'_j$, $\hat{B}_j$, $\hat{C}_j$, $\hat{D}_j$ so that (3.35) is ensured. It turns out that the notion of *transmission zeros* is relevant to this end. A definition of transmission zeros can be found for example in [93]. It is recalled below and particularized for a SISO system.

Definition 3.4.1. Let us consider a SISO linear system with state space realization A, B, C, D. The *transmission zeros* are the complex numbers s_i which satisfy

$$\operatorname{rank}\begin{pmatrix} A - s_i \mathbf{1}_n & B \\ C & D \end{pmatrix} < n + 1 \tag{3.36}$$

Before proceeding further, we must introduce some notations. Consider the matrix T and the corresponding matrices $\hat{A}_j = T^{-1}\bar{\hat{A}}_j T$ derived from $\bar{\hat{A}}_j$, $j \in \mathcal{J}$ as explained in Section 3.4.3 devoted to the constructive approach. Let us write $\hat{\bar{A}}_j$ as

$$\hat{\bar{A}}_j = \begin{pmatrix} 0 & a_j^1 & & & \\ & 0 & a_j^2 & A_j^* & \\ \vdots & & \ddots & \ddots & \\ & \mathbf{0} & & 0 & a_j^{n-1} \\ & & & \cdots & 0 \end{pmatrix} \tag{3.37}$$

where A_j^* denotes the coefficients above the $n-1$ diagonal entries a_j^m, $m = 1, \ldots, n-1$ located above the zero diagonal. Let have

$$T\hat{B}_j = {}^t\big(b_j^1 | \cdots | b_j^n\big) \tag{3.38}$$

where b_j^m stands for the mth component of the column vector $T\hat{B}_j$ and

$$\hat{C}_j T^{-1} = \big(c_j^1 | \cdots | c_j^n\big) \tag{3.39}$$

where c_j^m stands for the mth component of the row vector $\hat{C}_j T^{-1}$.

Proposition 3.4.3. *The surjectivity of each map $x_k \longmapsto A_j x_k$ ($j \in \mathcal{J}$) of $\mathcal{E}$ is guaranteed whenever*

$$c_j^1 b_j^n \prod_{m=1}^{n-1} a_j^m \neq 0 \tag{3.40}$$

Proof. According to Remark 3.4.2, $\mathcal{E}$ is a right inverse for $\mathcal{D}$. Furthermore, let us recall that (see Remark 3.4.1) the relative degree of $\mathcal{E}$ and $\mathcal{D}$ is zero. We conclude that each realization $\hat{A}_j$, $\hat{B}_j$, $\hat{C}_j$, $\hat{D}_j$, $j \in \mathcal{J}$ of $\mathcal{D}$ has n transmission zeros s_i and the s_i's are nothing but the n eigenvalues λ_i of A_j of $\mathcal{E}$. They are the roots of

$$\Psi_j(s) = \det(U) = 0 \text{ with } U = \begin{pmatrix} \hat{A}_j - s\mathbf{1}_n & \hat{B}_j \\ \hat{C}_j & \hat{D}_j \end{pmatrix} \tag{3.41}$$

U is often called the Rosenbrock's matrix. $\Psi_j(s)$ is a polynomial, its constant monomial is $\Psi_j(0)$ and corresponds to the product $\prod_{i=1}^n$ of the roots of $\Psi_j(s)$ and so corresponds to the product $\prod_{i=1}^n \lambda_i$ of the

eigenvalues of A_j of $\mathcal{E}$. Hence, surjectivity of $x_k \longmapsto A_j x_k$ $j \in \mathcal{J}$ is guaranteed whenever $\Psi_j(0) \neq 0$. The following equalities apply

$$\Psi_j(0) = \det \begin{pmatrix} \hat{A}_j & \hat{B}_j \\ \hat{C}_j & \hat{D}_j \end{pmatrix} \tag{3.42}$$

$$= \det \begin{pmatrix} T^{-1}\hat{\hat{A}}_j T & \hat{B}_j \\ \hat{C}_j & \hat{D}_j \end{pmatrix} \tag{3.43}$$

$$= \det \begin{pmatrix} T^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \begin{pmatrix} \hat{\hat{A}}_j & T\hat{B}_j \\ \hat{C}_j T^{-1} & \hat{D}_j \end{pmatrix} \begin{pmatrix} T & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \tag{3.44}$$

$$= \det \begin{pmatrix} \hat{\hat{A}}_j & T\hat{B}_j \\ \hat{C}_j T^{-1} & \hat{D}_j \end{pmatrix} \tag{3.45}$$

Consider a partitioned matrix with four sub-blocks E, F, G, H of compatible dimensions and such that H is invertible. We recall the following result.

$$\det \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \det(H) \cdot \det(E - FH^{-1}G) \tag{3.46}$$

Taking into account the special structure (3.37) of $\hat{\hat{A}}_j$, (3.38) and (3.39), it turns out that basic manipulations yield

$$\Psi_j(0) = \det \begin{pmatrix} \hat{\hat{A}}_j & T\hat{B}_j \\ \hat{C}_j T^{-1} & \hat{D}_j \end{pmatrix} = c_j^1 b_j^n \prod_{m=1}^{n-1} a_j^m \tag{3.47}$$

□

3.4.4 Boolean specificities

When considering the case of Boolean dynamical systems, a few simplifications can be done for the design. Indeed, one can take into account that the only elements are 0 and 1 and that 1 is the only invertible element. Moreover the subtraction is the same operation as the addition and corresponds to an exclusive-or $\oplus$. Hence,

- the first condition of Theorem 3.4.1 or Theorem 3.4.2 reduces to $\hat{D}_j = 1$, $\forall j \in \mathcal{J}$
- the third condition of Theorem 3.4.1 or Theorem 3.4.2 reduces to

$$\begin{cases} x_{k+1} &= \left(\hat{A}_{\sigma(k)} \oplus \hat{B}_{\sigma(k)}\hat{C}_{\sigma(k)}\right) x_k \oplus \hat{B}_{\sigma(k)} m_k \\ c_k &= \hat{C}_{\sigma(k)} x_k \oplus m_k \end{cases}$$

- the condition (3.40) on surjectivity implies that the coefficients c_j^1, b_j^n and a_j^m to be 1 for any $j \in \mathcal{J}$ and so

$$\hat{\hat{A}}_j = \begin{pmatrix} 0 & 1 & & & \\ \vdots & 0 & 1 & A_j^* & \\ & & \ddots & \ddots & \\ & \mathbf{0} & & 0 & 1 \\ & & & \cdots & 0 \end{pmatrix}$$

$$T_i \hat{B}_j = {}^t\begin{pmatrix} b_j^1 & \cdots & b_j^{n-1} & 1 \end{pmatrix}$$

$$\hat{C}_j T_i^{-1} = \begin{pmatrix} 1 & c_j^2 & \cdots & c_j^n \end{pmatrix}$$

It is worth pointing out that, in this situation, the complexities of the cipher and of the decipher are almost the same which is important when a practical implementations is sought.

3.4.5 Examples

3.4.5.1 Finite-time self-synchronization

This section gives an example that illustrates the construction of a finite-time self-synchronizing setup. We propose to design a finite-time self-synchronizing system of dimension $n = 3$ and with $J = 3$ modes. We consider matrices defined over the finite field $\mathbb{F} = \mathbb{Z}/7\mathbb{Z}$. This means that the only entries allowed for the matrices are elements in the set $\{0, \dots, 6\}$ and that the operations of additions and multiplications are performed modulo 7.

The design starts with the setting of the matrices $\hat{A}_j$, $\hat{B}_j$, $\hat{C}_j$, $\hat{D}_j$ which must fulfil the three conditions of Theorem 3.4.1, the condition (3.24) being replaced by the constructive approach provided in Section 3.4.1. We add the condition (3.40) on surjectivity.
First, for simplicity, we choose $\hat{D}_j = 1$ for any $j \in \{1, 2, 3\}$.
Secondly, we choose a set of three 3–dimensional matrices $\hat{\hat{A}}_j$ in the form of strict upper triangular matrices and with non zero entries located above the diagonal in order to guarantee the surjectivity.

$$\hat{\hat{A}}_1 = \begin{pmatrix} 0 & 3 & 2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \hat{\hat{A}}_2 = \begin{pmatrix} 0 & 2 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix} \hat{\hat{A}}_3 = \begin{pmatrix} 0 & 1 & 3 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix}$$

We then choose an invertible matrix T

$$T = \begin{pmatrix} 4 & 0 & 5 \\ 1 & 5 & 2 \\ 5 & 5 & 5 \end{pmatrix}$$

Its inverse over $\mathbb{F} = \mathbb{Z}/7\mathbb{Z}$ reads

$$T^{-1} = \begin{pmatrix} 4 & 2 & 5 \\ 6 & 1 & 2 \\ 4 & 4 & 3 \end{pmatrix}$$

Applying the change of basis $\hat{A}_j = T^{-1}\hat{\hat{A}}_j T$, we get that

$$\hat{A}_1 = \begin{pmatrix} 5 & 5 & 4 \\ 6 & 1 & 3 \\ 2 & 1 & 0 \end{pmatrix} \hat{A}_2 = \begin{pmatrix} 6 & 3 & 0 \\ 3 & 2 & 1 \\ 5 & 2 & 6 \end{pmatrix} \hat{A}_3 = \begin{pmatrix} 0 & 2 & 4 \\ 1 & 4 & 0 \\ 6 & 1 & 3 \end{pmatrix}$$

Finally, we choose arbitrary matrices $\hat{B}_j$ and $\hat{C}_j$ except the fact that the first entry c_j^1 of $\hat{C}_j T^{-1}$ and the last entry b_j^n of $T\hat{B}_j$ are not zero to fulfil the surjectivity condition (3.40).

$$\hat{B}_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \hat{B}_2 = \begin{pmatrix} 1 \\ 2 \\ 5 \end{pmatrix}, \hat{B}_3 = \begin{pmatrix} 3 \\ 6 \\ 1 \end{pmatrix}, \quad \begin{array}{l} \hat{C}_1 = \begin{pmatrix} 2 & 1 & 3 \end{pmatrix} \\ \hat{C}_2 = \begin{pmatrix} 6 & 2 & 1 \end{pmatrix} \\ \hat{C}_3 = \begin{pmatrix} 3 & 1 & 1 \end{pmatrix} \end{array}$$

$$D'_1 = D'_2 = D'_3 = 1$$

Finally, we derive the equations (3.25) of $\mathcal{E}$. The matrices read

$$A_1 = \begin{pmatrix} 6 & 5 & 4 \\ 6 & 1 & 3 \\ 0 & 0 & 4 \end{pmatrix}, A_2 = \begin{pmatrix} 0 & 1 & 6 \\ 5 & 5 & 6 \\ 3 & 6 & 1 \end{pmatrix}, A_3 = \begin{pmatrix} 5 & 6 & 1 \\ 4 & 5 & 1 \\ 3 & 0 & 2 \end{pmatrix}$$

$$B_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, B_2 = \begin{pmatrix} 1 \\ 2 \\ 5 \end{pmatrix}, B_3 = \begin{pmatrix} 3 \\ 6 \\ 1 \end{pmatrix}, \quad \begin{array}{l} C_1 = \begin{pmatrix} 5 & 6 & 4 \end{pmatrix} \\ C_2 = \begin{pmatrix} 1 & 5 & 6 \end{pmatrix} \\ C_3 = \begin{pmatrix} 4 & 6 & 6 \end{pmatrix} \end{array}$$

$$D_1 = D_2 = D_3 = 1$$

After the setting is completed, a sequence (m) is applied to $\mathcal{E}$. As expected, the self-synchronization is achieved after a finite transient time, so does the recovery of the input sequence as illustrated in Figure 3.2. The transient time before synchronization is of length $K = 3$ since the class of nilpotency t of the semigroup generated by $\{\hat{A}_1, \hat{A}_2, \hat{A}_3\}$ equals 3.

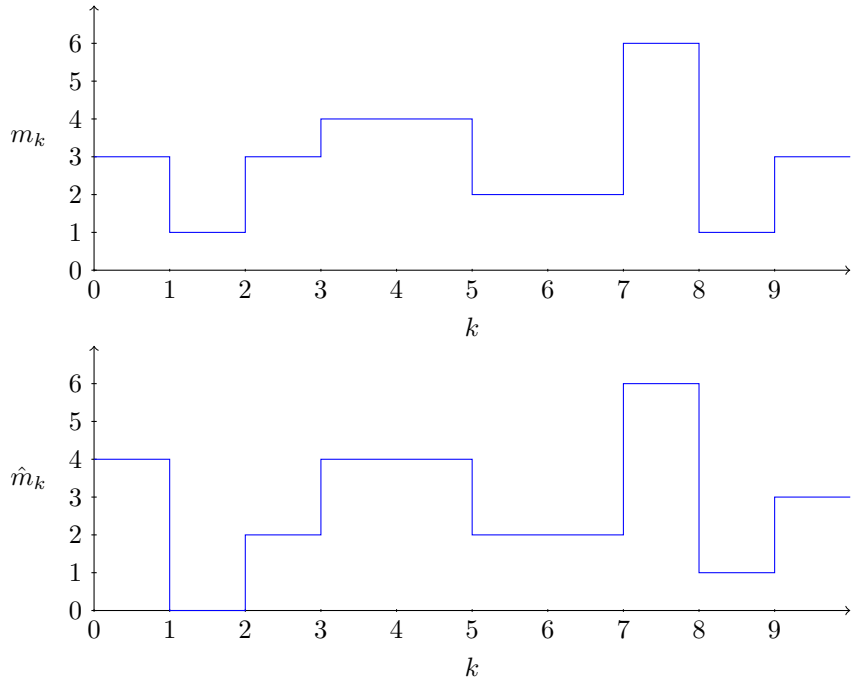


Figure 3.2: Time evolution of m and $\hat{m}$ of the setup $\mathcal{E}$–$\mathcal{D}$

3.4.5.2 Statistical self-synchronization

In this example, we aim at designing a setup having the statistical self-synchronization property. Besides, we assess the impact of the variation of the number of nilpotent semigroups ℓ on the synchronization delay. The dimension of the dynamical system is $n = 10$. The number of nilpotent semigroups varies from $\ell = 1$ to 6. They are built according to the constructive approach given in Section 3.4.1.2. The nilpotent semigroups have the same number of generators $J_i = 5$ and class of nilpotency $t_i = 10$, $\forall i \in \{1, \ldots, \ell\}$. The experiment is conducted by generating random mode sequences (σ) and determining, over 2000 runs, the percentage of sequences for which self-synchronization occurs. The result is depicted in Figure 3.3. The experiment shows that the more semigroups the higher the synchronization delay on average. In any case, the percentage gets close to 100% as K increases and is in accordance with Definition 2.2.5. The case when there is only one nilpotent semigroup deserves a special comment. The curve reaches 100% after $K = 10$. Indeed it corresponds to a finite self-synchronization according to Remark 3.4.5. The delay $K = 10$ corresponds to the class of nilpotency $t = 10$ of the set of matrices A'_j.
The probability law of synchronization seems to have an exponential-like shape. It is not trivial to figure out the exact expression of the law. Indeed, the problem amounts to determining the probability of occurrence of the mode sequences (σ) that induce self-synchronization. And yet, it is shown in [94] that a general treatment of this issue can be very intricate.

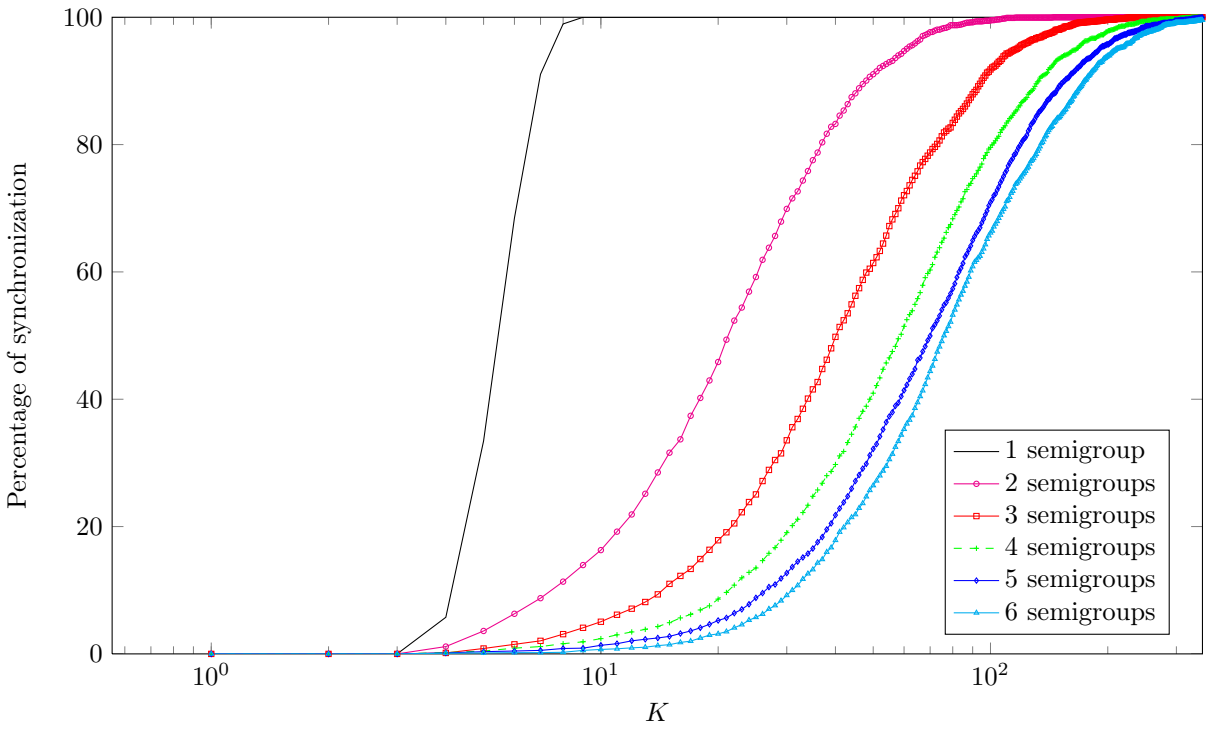


Figure 3.3: Percentage of the number of times the system $\mathcal{E}$–$\mathcal{D}$ has synchronized with respect to the delay of synchronization for different number of nilpotent semigroups.

Chapter 4

Boolean functions representation and properties

This chapter provides the necessary background on Boolean functions to discuss self-synchronization in the Boolean context later in Chapter 5. Boolean functions admit several representations, each one having its own interest to reveal specific properties. These representations are presented in Section 4.1. For our purpose, we also give some recalls on vectorial Boolean functions which are the multioutputs version of Boolean functions. Likewise Boolean functions, they admit several representations presented in Section 4.2. Section 4.3 is devoted to the study of the eigenstructures of the matrices discussed in Section 4.2. Section 4.4 defines the so-called reduced matrices that we shall use in next chapter.

We denote by $\mathbb{F}_2$ the two-element field. Its elements are 0 and 1, the addition is the *exclusive-or* denoted by $\oplus$ and the multiplication is the logical *and*. We denote it by $\cdot$ and omit it when not confusing. The $\oplus$ operation is nothing but the modulo two addition and the multiplication $\cdot$ is the usual operation as recalled in Table 4.1.

$\oplus$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

Table 4.1: Operations over $\mathbb{F}_2$

We denote by $\mathbb{F}_2^n$ the n dimensional vector space over $\mathbb{F}_2$.

Remark 4.0.6. A binary vector $x \in \mathbb{F}_2^n$ can be considered as the binary expansion of the integer

$$\sum_{i=0}^{n-1} x^i 2^i \tag{4.1}$$

To shorten the expressions, we may consider x to be the integer corresponding to the binary expansion of x.

Definition 4.0.2 (Hamming weight)**.** The Hamming weight of a vector $x \in \mathbb{F}_2^n$ is the number of nonzero coordinates. It is denoted by $\mathrm{hw}(x)$.

Definition 4.0.3 (Support)**.** The support of a vector $x \in \mathbb{F}_2^n$ is the set of indices i such that $x^i \neq 0$. It is denoted by $\mathrm{supp}(x)$.

4.1 Boolean functions

A Boolean function is a function from the vector space $\mathbb{F}_2^n$ to $\mathbb{F}_2$. We also call such a function a (n)–function. There are 2^{2^n} of them. Hence, their study and the search of functions with specific properties

cannot be performed by an exhaustive approach. The convenience of the representation depends on the property we want to characterize. In any case, the functions can be represented by a 2^n–dimensional vector with coefficients in $\mathbb{F}_2$ (or in $\mathbb{C}$). Each representation corresponds to the decomposition of this vector in a different basis. The main representations are recalled hereafter. For more information about these functions and their properties, the reader can refer to [95].

4.1.1 Truthtable

One straightforward way to represent a (n)–function is the truthtable. It is a 2^n–dimensional vector whose coefficients are either 0 or 1.
The basis used to express the truthable of f is composed of the 2^n Boolean functions

$$\delta_u(x) = \begin{cases} 1 & \text{if } x = u \\ 0 & \text{else} \end{cases}$$

defined for $x \in \mathbb{F}_2^n$ and parameterized by $u \in \mathbb{F}_2^n$.

Definition 4.1.1 (Distance)**.** The distance between two (n)–functions f and g is denoted by $\mathrm{d}(f, g)$. It is equal to the cardinality of the set $\{x \in \mathbb{F}_2^n,\ f(x) \neq g(x)\}$. It is also equal to $\mathrm{hw}(f \oplus g)$ where f and g are considered as vectors.

The balancedness property is important in cryptography.

Definition 4.1.2 (Balanced Boolean function)**.** An (n)–function is said to be balanced if half the possible values $f(x)$ are equal to 0 and the remaining values equal to 1.

From the truthtable, it suffices to check that there are exactly 2^{n-1} components equal to 0 (or 1).
All along this section we provide an illustrative example of the representations of the (3)–function f_{e1} defined in Table 4.2.

x	$f_{e1}(x)$
000	0
001	0
010	1
011	1
100	0
101	1
110	0
111	0

Table 4.2: Truthable of the function f_{e1}

For short, we may use the following vectorial notation

$$f_{e1} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

4.1.2 Algebraic normal form

A (n)–function can be represented by a multivariate polynomial over $\mathbb{F}_2$. However, $\forall a \in \mathbb{F}_2$, $a = a \cdot a$ that is, each element is equal to its own square. Therefore, several polynomials represent the same function. In order to obtain the uniqueness of the representation, we only consider the polynomials in the ring of multivariate polynomials whose coefficients have power at most one. Let a and b be two elements of $\mathbb{F}_2$. The element a raised to the power b is denoted by a^b. The power table in $\mathbb{F}_2$ is given in Table 4.3.

a^b	0	1
0	1	0
1	1	1

Table 4.3: Power table in $\mathbb{F}_2$

When x and u are elements of $\mathbb{F}_2^n$, the notation is extended to $x^{\hat{}u}$ and is defined by

$$x^{\hat{}u} = (x^0)^{u^0} \cdots (x^{n-1})^{u^{n-1}} \tag{4.2}$$

The element $x^{\hat{}u}$ is called a *monomial*.
Any (n)–function f can be written as a polynomial called *Algebraic Normal Form* (ANF for short).

$$f(x) = \sum_{u \in \mathbb{F}_2^n} a^u x^{\hat{}u} \tag{4.3}$$

where $a \in \mathbb{F}_2^n$. We recall that a^u denotes the component indexed by u of the vector a. Any such function represents a Boolean function. The function f is completely defined by the coefficients of a. The basis used to express the ANF is composed of the 2^n Boolean functions $x \longmapsto x^{\hat{}u}$ defined for $x \in \mathbb{F}_2^n$ and parameterized by $u \in \mathbb{F}_2^n$. This representation is often used in cryptography.

The ANF of f_{e1} is

$$f_{e1}(x) = x^1 \oplus x^0x^2 \oplus x^1x^2 \oplus x^0x^1x^2$$

Let $\mathrm{Card}\,(u)$ be the cardinal of the set u, the algebraic degree of a function is defined as follows:

Definition 4.1.3 (Algebraic degree)**.** The algebraic degree of a Boolean function f is equal to

$$\max_{u \in \mathbb{F}_2^n} \{\mathrm{hw}(u),\, a^u \neq 0\}.$$

In other words, it is equal to the maximum number of variables involved in a monomial x^u with nonzero coefficient a^u. The algebraic degree of f_{e1} is 3.

4.1.3 Fourier/Walsh transform

If f is a (n)–function, we denote by $\widehat{f}$ its Fourier transform, which is by definition the real-valued mapping $\mathbb{F}_2^n \longrightarrow \mathbb{R}$ defined for any $u \in \mathbb{F}_2^n$ by

$$\widehat{f}(u) = \sum_{x \in \mathbb{F}_2^n} f(x)(-1)^{x \cdot u} \tag{4.4}$$

where $x \cdot u = x^0u^0 \oplus \cdots \oplus x^{n-1}u^{n-1}$ is the dot product of x and u.
When f is considered as a vector, as explained in Section 4.1.1, the expression of the Fourier transform (4.4) also admits a matrix oriented representation

$$\widehat{f} = Hf \tag{4.5}$$

where H is the so-called *Hadamard matrix* whose coefficient at row u and column v is $H_{u,v} = (-1)^{u \cdot v}$. The Fourier transform of f_{e1} is

$$\widehat{f_{e1}} = \begin{pmatrix} 4 \\ 0 \\ -2 \\ -2 \\ -2 \\ 2 \\ 0 \\ 0 \end{pmatrix}$$

The Hadamard matrix H is invertible and

$$H^{-1} = 2^{-n} H \tag{4.6}$$

The basis used to express the Fourier transform of f is composed of the 2^n Boolean functions $x \longmapsto (-1)^{x \cdot u}$ defined for $x \in \mathbb{F}_2^n$ and parameterized by $u \in \mathbb{F}_2^n$.
This transform is invertible and the inverse is given by:

$$\widehat{\widehat{f}} = 2^n f \tag{4.7}$$

Let us recall Parseval's theorem ([95]):

Theorem 4.1.1 (Parseval's theorem)**.** *For any (n)–function f, the following statement holds:*

$$\sum_{u \in \mathbb{F}_2^n} \left[\widehat{f}(u)\right]^2 = 2^n \sum_{x \in \mathbb{F}_2^n} [f(x)]^2 \tag{4.8}$$

When dealing with Boolean functions, we rather resort to the Walsh transform which gets nicer properties than the Fourier transform in most cases. The Walsh transform of a Boolean function f is the Fourier transform of its sign function f_χ where $f_\chi(x) = (-1)^{f(x)} = 1 - 2f(x)$ for $x \in \mathbb{F}_2^n$. That is,

$$\widehat{f_\chi}(u) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus x \cdot u} \tag{4.9}$$

The Walsh transform of f_{e1} is

$$\widehat{f_{e1}}_\chi = \begin{pmatrix} 2 \\ 2 \\ 2 \\ 2 \\ -2 \\ -2 \\ 6 \\ -2 \end{pmatrix}$$

Proposition 4.1.1 ([95])**.** *A (n)–function f is balanced if and only if $\widehat{f_\chi}(0) = 0$.*

The correlation function of two functions is defined as follows.

Definition 4.1.4 (Correlation function [95])**.** The correlation of two (n)–functions f and g is defined for any $u \in \mathbb{F}_2^n$ by

$$(f \otimes g)(u) = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus g(x \oplus u)} \tag{4.10}$$

The Walsh coefficient $\widehat{f_\chi}(a), a \in \mathbb{F}_2^n$ can be regarded as a measurement of the correlation of a function f with a linear function $x \longmapsto a \cdot x$. It introduces the notion of nonlinearity defined as follows.

Definition 4.1.5 (Nonlinearity)**.** The nonlinearity of a (n)–function f is defined as the distance between f and the set of all affine functions. It is obtained by the expression

$$2^{n-1} - 2^{-1} \max_{a \in \mathbb{F}_2^n} |\widehat{f_\chi}(a)|. \tag{4.11}$$

Regarding security, the nonlinearity is an important property of Boolean functions involved in cryptographic application. It denotes the difficulty for the cryptanalysts to approximate a function by an affine one. This is discussed in Section 6.1.
Boolean functions whose nonlinearity is maximum are called *bent functions*.

Definition 4.1.6 (Bent function)**.** A (n)–function is said to be bent if the absolute value of all the coefficients of its Walsh transform are equal to $2^{\frac{n}{2}}$.

Hence, a balanced function cannot be bent. Bent functions only exist for even values of n. They are the functions which are at equal distance from any linear functions.

4.1.4 Numerical normal form

This section introduces another multivariate polynomial representation of vectorial Boolean functions. We call it *Numerical Normal Form* (or NNF for short). Unlike the ANF, the coefficients of the polynomial do not lie in $\mathbb{F}_2$ but in $\mathbb{Z}$ (or more generally in $\mathbb{C}$). This representation is studied in [96]. Of course, not any such polynomial corresponds to a Boolean function. To define the NNF, some notations are required. If the support of $x \in \mathbb{F}_2^n$ is included in the support of $u \in \mathbb{F}_2^n$, we write $x \preceq u$. The following equivalence holds: $x \preceq u \iff u \hat{} x = 1$. The NNF of a (n)–function f is defined, for any $u \in \mathbb{F}_2^n$, by

$$\widetilde{f}(u) = \sum_{x \in \mathbb{F}_2^n} (-1)^{\mathrm{hw}(x)-\mathrm{hw}(u)} f(x) u \hat{} x = \sum_{x \in \mathbb{F}_2^n | x \preceq u} (-1)^{\mathrm{hw}(x)-\mathrm{hw}(u)} f(x) \tag{4.12}$$

Likewise for the Walsh transform, a matrix relation exists between the NNF and the function f:

$$\widetilde{f} = Nf \tag{4.13}$$

with N a 2^n dimensional square matrix whose coefficient at row $u \in \mathbb{F}_2^n$ and column $v \in \mathbb{F}_2^m$ is given by $N_{u,v} = (-1)^{\mathrm{hw}(v)-\mathrm{hw}(u)} u \hat{} v$.
The numerical normal form of f_{e1} is

$$\widetilde{f_{e1}} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ -1 \\ -1 \end{pmatrix}$$

The basis used to express the NNF of f is composed of the 2^n Boolean functions $x \longmapsto (-1)^{\mathrm{hw}(x)-\mathrm{hw}(u)} u \hat{} x$ defined for $x \in \mathbb{F}_2^n$ and parameterized by $u \in \mathbb{F}_2^n$.

4.2 Vectorial Boolean functions

A vectorial Boolean function is a function from the vector space $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. It can be considered as a vector of m (n)–functions, which justifies the name. Such a function is also called a (n, m)–function. Vectorial Boolean functions are extensively discussed in [97]. We recall some of their representations. For any vectorial Boolean function f, it is possible to define different matrix representations denoted by $\mathcal{W}^f$, $\mathcal{R}^f$, $\mathcal{A}^f$, $\mathcal{N}^f$ and $\mathcal{F}^f$. They are respectively named Walsh, correlation, algebraic, numerical and adjacency matrices. The superscript of the matrices is omitted when the corresponding function is obvious.

These matrices are discussed in the following.

A first representation consists in giving the truthable of each of the coordinate functions and is called truthtable. However, it is not useful for our purpose and we only use it to quickly specify a function all along this section. For each representation, an example is provided. As a reference function, let us consider the function $f_{e2} : \mathbb{F}_2^3 \longrightarrow \mathbb{F}_2^3$ defined in Table 4.4.

x	$f_{e2}(x)$
000	010
001	100
010	001
011	101
100	010
101	101
110	010
111	110

(a) Truthable

$$\begin{pmatrix} x^1 \oplus x^0x^2 \oplus x^1x^2 \oplus x^0x^1x^2 \\ 1 \oplus x^0 \oplus x^1 \oplus x^0x^1 \oplus x^1x^2 \\ x^0 \end{pmatrix}$$

(b) ANF vector

Table 4.4: Some representations of the example function f_{e2}

The extension of Definition 4.1.2 to vectorial Boolean function is

Definition 4.2.1 (Balanced vectorial Boolean function)**.** Let f be a (n, m)–function. It is said to be balanced if, when $x \in \mathbb{F}_2^n$ spans all the space, the values $f(x) \in \mathbb{F}_2^m$ are taken an equal number of times.

4.2.1 Walsh/correlation matrix

The Walsh matrix of any (n, m)–function is the $2^m \times 2^n$ dimensional matrix $\mathcal{W}$ whose coefficients are defined by

$$\mathcal{W}_{u,v} = \sum_{x \in \mathbb{F}_2^n} (-1)^{u \cdot f(x) \oplus v \cdot x} \tag{4.14}$$

The row $u \in \mathbb{F}_2^m$ of the matrix $\mathcal{W}$ is the Walsh transform of the linear combinations of the coordinates of f defined by $x \longmapsto u \cdot f(x)$. The coefficients of the Walsh matrix of a function are called the *spectrum* of the function. The Walsh matrix of f_{e2} is

$$\mathcal{W}^{f_{e2}} = \begin{pmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & -2 & -2 & 6 & -2 \\ 0 & -4 & 0 & -4 & 4 & 0 & -4 & 0 \\ -6 & -2 & 2 & -2 & 2 & -2 & 2 & -2 \\ 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & -2 & -2 & -2 & 6 \\ -4 & 0 & -4 & 0 & 0 & 4 & 0 & -4 \\ -2 & -6 & -2 & 2 & -2 & 2 & -2 & 2 \end{pmatrix}$$

Correlation matrices have been defined in [98]. They are related to Walsh matrices by a mere normalization coefficient. If $\mathcal{R}$ is the correlation matrix of f then,

$$\mathcal{R} = 2^{-n}\mathcal{W} \tag{4.15}$$

The coefficient $\mathcal{R}_{u,v}$ is the correlation coefficient of the function $x \longmapsto u \cdot f(x)$ with the linear function $x \longmapsto v \cdot x$. When u spans the set $\mathbb{F}_2^m$, all the linear combinations of the coordinate functions of f are considered. When v spans the set $\mathbb{F}_2^n$, all the linear functions with n variables are spanned. Hence, the matrix $\mathcal{R}$ contains all the correlation coefficients of any linear combination of the coordinate functions of f with all the linear functions.

Proposition 4.2.1 ([98]). *If f is a (n,m)–function and g is a (p,n)–function then, the correlation matrix of the composed function $f \circ g$ is given by*

$$\mathcal{R}^{f\circ g} = \mathcal{R}^f \mathcal{R}^g \tag{4.16}$$

Proposition 4.2.2 ([97]). *A (n,m)–function f is balanced if and only if the first column of its Walsh matrix is equal to 0 except the first coefficient. The same holds for the correlation matrix.*

The remaining part of Section 4.2.1 is devoted to important results used later on and has been published in [99]. We are given a (n,m)–function g and a random variable $X \in \mathbb{F}_2^n$ whose value is described by the probability law $p : \mathbb{F}_2^n \longrightarrow \mathbb{R}$ defined by $p(x) = \Pr[X = x]$. We want to infer the probability law $q : \mathbb{F}_2^m \longrightarrow \mathbb{R}$ that describes the random variable $Y \in \mathbb{F}_2^m$ defined by $Y = g(X)$, q being defined by $q(y) = \Pr[y = g(X)]$. Without ambiguity, the notation p (respectively q) refers either to the function or to the 2^n (respectively 2^m) column vectors whose coordinate index $x \in \mathbb{F}_2^n$ (respectively $y \in \mathbb{F}_2^m$) has the value $p(x)$ (respectively $q(y)$). The same holds for $\widehat{p}$ and $\widehat{q}$ which are the respective Fourier transforms of p and q.

Proposition 4.2.3 ([99]). *Let $\mathcal{R}^g$ be the correlation matrix of g. Applying the function g to a variable whose value is chosen according to the probability law described by p gives a vector whose value is described by the probability law q. They are related by*

$$\widehat{q} = \mathcal{R}^g \widehat{p} \tag{4.17}$$

Proof. Let us first relate q and p:

$$\begin{aligned} q(y) &= \textstyle\sum_{x\in\mathbb{F}_2^n | g(x)=y} p(x) \\ &= 2^{-m} \textstyle\sum_{x\in\mathbb{F}_2^n} p(x) \sum_{u\in\mathbb{F}_2^m} (-1)^{u(g(x)\oplus y)} \\ &= 2^{-m} \textstyle\sum_{u\in\mathbb{F}_2^m} \sum_{x\in\mathbb{F}_2^n} (-1)^{u\cdot y} p(x) (-1)^{u\cdot g(x)} \end{aligned}$$

We now express the Fourier transform of q.

$$\begin{aligned} \widehat{q}(s) &= \textstyle\sum_{y\in\mathbb{F}_2^m} q(y)(-1)^{s\cdot y} \\ &= 2^{-m} \textstyle\sum_{u\in\mathbb{F}_2^m, x\in\mathbb{F}_2^n} \underbrace{\sum_{y\in\mathbb{F}_2^m} (-1)^{u\cdot y \oplus s\cdot y}}_{\begin{cases} 2^m & \text{if } u = s \\ 0 & \text{else} \end{cases}} p(x)(-1)^{u\cdot g(x)} \\ &= \textstyle\sum_{x\in\mathbb{F}_2^n} p(x)(-1)^{s\cdot g(x)} \\ &= 2^{-n} \textstyle\sum_{x\in\mathbb{F}_2^n} p(x) \sum_{z\in\mathbb{F}_2^n} (-1)^{s\cdot g(z)} \sum_{v\in\mathbb{F}_2^n} (-1)^{v\cdot(x\oplus z)} \\ &= \textstyle\sum_{v\in\mathbb{F}_2^n} \underbrace{\sum_{x\in\mathbb{F}_2^n} p(x)(-1)^{v\cdot x}}_{\widehat{p}(v)} \underbrace{2^{-n} \sum_{z\in\mathbb{F}_2^n} (-1)^{s\cdot g(z)\oplus v\cdot z}}_{\mathcal{R}^g_{s,v} = 2^{-n}\mathcal{W}^g_{s,v}} \end{aligned}$$

□

The following corollary can be stated.

Corollary 4.2.1 ([99]).

$$q = H^{-1} \mathcal{R}^g H p \tag{4.18}$$

Proof. Equation (4.17) also reads $Hq = \mathcal{R}^g H p$. □

If we restrict the vector p of Proposition 4.2.3 to the uniform probability vector, the following corollary, which corresponds to Lemma 1 in [100], is straightforwardly obtained.

Corollary 4.2.2 ([99]). *Let X follow the uniform distribution and g be a (n,n)–function. The probability distribution, after applying the function g to X reads $\forall x \in \mathbb{F}_2^n$,*

$$\Pr[g(X) = x] = 2^{-n} \sum_{s \in \mathbb{F}_2^n} (-1)^{s \cdot x} \mathcal{R}^g_{s,0} \tag{4.19}$$

Proof. We recall that $\widehat{p} = Hp$. If p is the uniform probability vector then, $\widehat{p}(u) = 1$ if $u = 0$ and $\widehat{p}(u) = 0$ if $u \neq 0$. The expression $\mathcal{R}^g Hp$ of Corollary 4.2.1 is equal to the first column of $\mathcal{R}$. Then, taking into account (4.6), the expression holds. □

4.2.2 Numerical and algebraic matrices

The extension of the NNF to (n,m)–functions gives rise to a $2^n \times 2^m$ dimensional matrix $\mathcal{N}$. It was suggested by Éric Garrido. We call it *numerical matrix* and the entry at row $u \in \mathbb{F}_2^n$ and column $v \in \mathbb{F}_2^m$ is defined by

$$\mathcal{N}_{u,v} = \sum_{x \in \mathbb{F}_2^n} (-1)^{\mathrm{hw}(x) - \mathrm{hw}(u)} f(x)^{\hat{}\, v} u^{\hat{}\, x} \tag{4.20}$$

Note that the columns $v \in \mathbb{F}_2^m$ for which $\mathrm{hw}(v) = 1$ correspond to the NNF of a component function of f. The matrix $\mathcal{N}$ is the representation of f in the basis of the polynomials $x \longmapsto (-1)^{\mathrm{hw}(x) - \mathrm{hw}(u)} u^{\hat{}\, x}$. In the same way that the ANF can be obtained by performing a modulo two reduction of the NNF, we define the matrix $\mathcal{A}$ as the modulo two version of the matrix $\mathcal{N}$. We call it the *algebraic matrix*. In this case, the function is represented in the basis of the polynomials $x \longmapsto u^{\hat{}\, x}$. The numerical matrix and the algebraic matrix of f_{e2} are

$$\mathcal{N}^{f_{e2}} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & -2 & 1 & 0 \end{pmatrix} \qquad \mathcal{A}^{f_{e2}} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

4.2.3 Adjacency matrix

The adjacency matrix representation of f is denoted by $\mathcal{F}$. It is the representation of f in the basis of the indicator functions δ_u, parametrized by $u \in \mathbb{F}_2^n$ and defined for any $x \in \mathbb{F}_2^n$ by

$$\delta_u(x) = \begin{cases} 1 & \text{if } x = u \\ 0 & \text{else} \end{cases} \tag{4.21}$$

Definition 4.2.2 (Adjacency matrix). Let f be a (n,m)–function. Its adjacency matrix $\mathcal{F}$ is a $2^n \times 2^m$ dimensional matrix for which each row $x \in \mathbb{F}_2^n$ is null except the coefficient on the column $y \in \mathbb{F}_2^m$ where $y = f(x)$.

The adjacency matrix of f_{e2} is

$$\mathcal{F}^{f_{e2}} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Definition 4.2.2 matches the one of graph theory [101]. When the number of inputs is equal to the number of outputs, the function can also be represented by a labeled directed graph $\mathcal{G}$. In this thesis, we only consider directed graphs. Hence, for short, we call them graph which obeys the following definition.

Definition 4.2.3 (Directed graph [101])**.** A directed graph is a pair $\mathcal{G} = (V, E)$ where V is called the set of vertices and E the set of arcs. An arc is an ordered pair of vertices.

When the arc is an unordered pair, it is called an *edge* and the corresponding graph is an *undirected graph*.
In our context, the set V has cardinality 2^n and we assign to each vertex in V a distinct element of $\mathbb{F}_2^n$. Hence, we can identify V to $\mathbb{F}_2^n$. The element (x, y) is an element of E if and only if $y = f(x)$.
Next, we recall some graph theoretic vocabulary that we will need in the further development. For each definition, the corresponding structure is identified in Figure 4.1.

Definition 4.2.4 (Graph related vocabulary)**.**

- a vertex a is said to be incident to a vertex b (or to be a preimage of a vertex b) if there is an arc from a to b. The vertex 111 is incident to 110;
- the *in-degree of* of a vertex is the number of vertices incident to that vertex. The in-degree of vertex 000 is 0. The in-degree of vertex 010 is 3;
- the *out-degree* of a vertex is the number of vertices incident to that vertex. The out-degree of each vertex in Figure 4.1 is 1, including vertex 101;
- a *path* is a sequence of vertices $(v_0, \ldots, v_k)$ such that, for each vertex, there is an arc from v_i to v_{i+1}. The number of arcs involved in the sequence is the length of the path. The sequence $(110, 010, 001)$ is a path of length 2;
- a *cycle* is a path such that the start vertex and the end vertex are the same, $(010, 001, 100, 010)$ is a cycle of length 3;
- a *junction* is a vertex such that the in-degree is at least two. The multiplicity of the junction is equal to the in-degree minus one. The vertex 101 is a junction of multiplicity one. The vertex 010 is a junction of multiplicity two;
- a *preimage set* of a vertex is the set of vertices incident to that vertex. The preimage set of the junction 010 is $\{000, 110, 100\}$ and the preimage set of the junction 101 is $\{011, 101\}$;
- a *sink* is a vertex with at least one incident vertex and such that it is not incident to another vertex than itself. Any sink can be identified to a cycle of length one. The vertex 101 is a sink;
- a *leaf* is a vertex with no incident vertex. The vertices $000, 011$ and 111 are the leaves of the graph;
- a *connected component* of an undirected graph is a maximal subgraph in which there is a path between any two vertices.
- a *weakly connected component* is a set of vertices that would be connected by ignoring the direction of arcs. The vertices $\{111, 110, 000, 010, 100, 001\}$ correspond to one weakly connected component and the vertices $\{011, 101\}$ correspond to another weakly connected component.

Note that, for an (n, n)–function, the existence of junctions, leaves and sinks is possible if and only if the function is not a permutation. It is shown in Section 4.3 that these elements are related to the presence of null eigenvalues in the adjacency matrix.

Remark 4.2.1. When the graph corresponds to a function, there is exactly one cycle per weakly connected component.

Definition 4.2.5 (Isomorphic graphs)**.** Two graphs $\mathcal{G}$ and $\mathcal{G}'$ are isomorphic if they are the same graph up to a relabelling of the vertices.

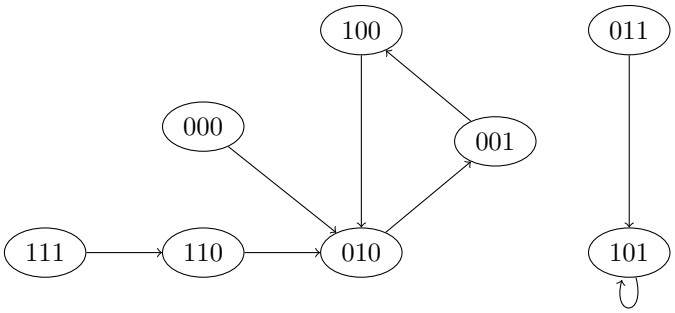


Figure 4.1: Example of the graph of f_{e2}

4.2.4 Relation between the matrix representations

An original contribution of this thesis, not yet published, is the connection between the different matrix representations. When $m = n$, the numerical matrix $\mathcal{N}$, the correlation matrix $\mathcal{R} = 2^{-n}\mathcal{W}$ and the adjacency $\mathcal{F}$ (with coefficients in $\mathbb{R}$) are related by similarity transforms. We also show that, when the coefficients of the adjacency matrix $\mathcal{F}$ and of the algebraic matrix $\mathcal{A}$ are considered in $\mathbb{F}_2$, there exists a similarity transform that relates them. This relation permits to simplify the analysis of the eigenstructures of these matrices. Before proceeding further, the following lemma is required:

Lemma 4.2.1 (Orthogonality lemma [102])**.** *Let $x, y \in \mathbb{F}_2^n$ then,*

$$\begin{aligned}\sum_{s\in\mathbb{F}_2^n}(-1)^{\mathrm{hw}(s)}s\hat{}x\,y\hat{}s &= \sum_{s\in\mathbb{F}_2^n|x\preceq s\preceq y}(-1)^{\mathrm{hw}(s)}\\ &= \begin{cases}(-1)^{\mathrm{hw}(y)} & \text{if } x=y\\ 0 & \text{else}\end{cases}\end{aligned}$$

Proposition 4.2.4. *Let f be a (n,n)–function then, its numerical matrix $\mathcal{N}$ and its adjacency matrix $\mathcal{F}$ are related by $\mathcal{N} = N\mathcal{F}N^{-1}$.*

Proof. We want to show that $\mathcal{N}N = N\mathcal{F}$. To this end, we compute the coefficients of each member of the equality and show that they are the same. The coefficient at row $u \in \mathbb{F}_2^n$ and column $v \in \mathbb{F}_2^n$ of $N\mathcal{F}$ is

$$\sum_{w\in\mathbb{F}_2^n}N_{u,w}\mathcal{F}_{w,v} = \sum_{w\in\mathbb{F}_2^n|v=f(w)}N_{u,w} \tag{4.22}$$

The coefficient at row $u \in \mathbb{F}_2^n$ and column $v \in \mathbb{F}_2^n$ of $\mathcal{N}N$ is

$$\sum_{w\in\mathbb{F}_2^n}\mathcal{N}_{u,w}N_{w,v} = \sum_{w\in\mathbb{F}_2^n}\sum_{x\in\mathbb{F}_2^n}N_{u,x}f(x)\hat{}wN_{w,v} \tag{4.23}$$

$$= \sum_{x\in\mathbb{F}_2^n}N_{u,x}(-1)^{\mathrm{hw}(v)}\sum_{w\in\mathbb{F}_2^n}f(x)\hat{}w(-1)^{-\mathrm{hw}(w)}w\hat{}v \tag{4.24}$$

In view of Lemma 4.2.1

$$\sum_{w\in\mathbb{F}_2^n}\mathcal{N}_{u,w}N_{w,v} = \sum_{x\in\mathbb{F}_2^n}N_{u,x}(-1)^{\mathrm{hw}(v)}\underbrace{\sum_{w\in[v,f(x)]}(-1)^{-\mathrm{hw}(w)}}_{\begin{cases}(-1)^{-\mathrm{hw}(v)} & \text{if } v=f(x)\\ 0 & \text{else}\end{cases}} \tag{4.25}$$

□

The coefficients of the matrix $\mathcal{F}$ can also be considered as elements of $\mathbb{F}_2$. In this situation, we show that $\mathcal{F}$ and $\mathcal{A}$ are similar and are related by the expression:

$$\mathcal{A} = \mathcal{N} \mod 2 = N\mathcal{F}N^{-1} \mod 2 = (N \mod 2)\mathcal{F}(N^{-1} \mod 2) \tag{4.26}$$

and $(N \mod 2) = (N^{-1} \mod 2) = N^{-1}$.
We are now in position to relate the correlation matrix (or the Walsh matrix) of a function to the adjacency matrix of its graph.

Proposition 4.2.5. *Let f be a (n, n)–function, then its adjacency matrix $\mathcal{F}$ and its correlation matrix $\mathcal{R}$ are related as follows*

$$\mathcal{F} = H^t\mathcal{R}H^{-1} \tag{4.27}$$

Proof. Let p be a probability vector such that its xth coordinate p^x is one. By definition of a probability vector, all the other components are null. Due to Corollary 4.2.2, the vector $q = H\mathcal{R}H^{-1}p$ is a probability vector and all its components equal zero except the component $y = f(x)$. This implies that the coefficients of the column x of $H\mathcal{R}H^{-1}$ are all zero except the one at row y. Therefore, the coefficient at row y and column x of $H\mathcal{R}H^{-1}$ is one if $x = f(y)$ and zero elsewhere. This holds for any $x, y \in \mathbb{F}_2^n$. By definition, this is the transpose of the adjacency matrix. We recall that H is a symmetric matrix. Therefore, $H^t\mathcal{R}H^{-1}$ is the adjacency matrix of f. □

Considering Propositions 4.2.4 and 4.2.5 and taking into account the fact that a matrix and its transpose are similar, we conclude that $\mathcal{F}, {}^t\mathcal{F}, \mathcal{N}$ and $\mathcal{R}$ are similar matrices. Since the Walsh matrix $\mathcal{W}$ is related to $\mathcal{R}$ by a mere scaling with factor 2^n as shown by (4.15), its eigenvalues are simply the eigenvalues of $\mathcal{R}$ scaled by 2^n.

Due to the similarity transform and to Proposition 4.2.1, the following important relations also hold

$$\mathcal{F}^{f\circ g} = \mathcal{F}^g\mathcal{F}^f \tag{4.28}$$

$$\mathcal{N}^{f\circ g} = \mathcal{N}^g\mathcal{N}^f \tag{4.29}$$

$$\mathcal{A}^{f\circ g} = \mathcal{A}^g\mathcal{A}^f \tag{4.30}$$

Note that, the order in which the matrices are multiplied is reversed compared to (4.16). This is due to the transpose operator in (4.27).

Constant functions

The following expressions, directly derived from the definition, give the form of the matrix representations of a constant (n, m)–function.

A (n, n)–function is constant if and only if its correlation matrix has the form

$$\mathcal{R} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \pm 1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ \pm 1 & 0 & \cdots & 0 \end{pmatrix} \tag{4.31}$$

In the same way, a (n, n)–function is constant if and only if its algebraic matrix has the form

$$\mathcal{A} = \begin{pmatrix} 1 & \mathcal{A}_{0,1} & \cdots & \mathcal{A}_{0,2^n-1} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \tag{4.32}$$

with $\mathcal{A}_{0,i} \in \{0,1\}$. In the same way, a (n,n)–function is constant if and only if its numerical matrix has the form

$$\mathcal{N} = \begin{pmatrix} 1 & \mathcal{N}_{0,1} & \cdots & \mathcal{N}_{0,2^n-1} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \tag{4.33}$$

with $\mathcal{N}_{0,i} \in \{0,1\}$.

4.2.5 Univariate polynomial

When the number of inputs is equal to the number of outputs, vectorial Boolean functions also admit a univariate polynomial representation. A (n,n)–function f can always be uniquely represented by the following univariate polynomial.

$$f(x) = \sum_{i=0}^{2^n-1} a^i (x)^i,\ a^i \in \mathbb{F}_{2^n} \tag{4.34}$$

Note that here, a is a 2^n-dimensional vector of elements of $\mathbb{F}_{2^n}$ and that a^i denote the ith component of a. The element x belongs to $\mathbb{F}_{2^n}$ and the notation $(x)^i$ denotes the vector x multiplied i times by itself according to the multiplication of the field $\mathbb{F}_{2^n}$.

4.3 Eigenanalysis of the matrix representations

This section is a contribution, not yet published, to the study of vectorial Boolean functions. It is important as it establishes connections between the eigenanalysis of the matrix representations of a function and its graph representation. In Chapter 5, we will be interested in the matrix representations with null eigenvalues. Let f be a (n,n)–function, the corresponding matrices, $\mathcal{F}$, ${}^t\mathcal{F}$, $\mathcal{N}, \mathcal{A}$ and $\mathcal{R}$ are square and their eigenanalysis can be performed. Due to the similarity transforms that relate them, the eigenvalue analysis can be done on any of them. However, the fact that the adjacency matrix has exactly one nonzero component equal to one per row makes its study easier. As explained in Section 4.2.3, it is possible to associate a graph $\mathcal{G}$ to the function f. We show that the eigenvalues of these matrices are directly related to the number of cycles, to their length and to the number of leaves in the graph $\mathcal{G}$. The study of the eigenvectors depends on the matrix under consideration. It has been mentioned in Section 4.2.4 that $\mathcal{F}$ can be considered as an $\mathbb{R}$-valued matrix or an $\mathbb{F}_2$-valued matrix. The same holds for $\mathcal{A}$ which is the modulo two version of $\mathcal{N}$. However, the matrix $\mathcal{R}$ can only be considered as an $\mathbb{R}$-valued matrix. We perform the eigenanalysis regardless of every field $\mathbb{F}$ considered. We denote by $\overline{\mathbb{F}}$ its algebraic closure. The results are particularized to a specific field when relevant.

Section 4.3.1 is devoted to eigenvalues. Section 4.3.2 shows how to determine the corresponding eigenvectors from the graph of the function. Section 4.3.3 provides an illustrative example of the results presented in Section 4.3.1 and Section 4.3.2.

4.3.1 Eigenvalues

We use the specificities of the adjacency matrix $\mathcal{F}$ to show that the eigenvalues of the similar matrices $\mathcal{F}$, ${}^t\mathcal{F}$, $\mathcal{N}, \mathcal{A}$, and $\mathcal{R}$ are either 0 or roots of the unity.

Proposition 4.3.1. *If an eigenvalue of the adjacency matrix $\mathcal{F}$ is not null, it is a root of the unity.*

Proof. Let $\alpha \in \overline{\mathbb{F}}$ be a nonzero eigenvalue and $x \in \overline{\mathbb{F}}^{2^n}$ be an associated eigenvector then, $\mathcal{F}x = \alpha x$. Since $\mathcal{F}$ is the adjacency matrix of an application, each row has all its coefficients equal to zero except one which is equal to one. As a consequence, the components of $\mathcal{F}x$ are nothing but some components of x. We denote by $(\mathcal{F})^k$ the matrix $\mathcal{F}$ raised to the power k. Consider the sequence of vectors $(\mathcal{F})^k x$, $k \in \mathbb{N}$. This sequence has a finite number of distinct elements. Therefore, there exists two indexes $i < j$ such

that $(\mathcal{F})^i x = (\mathcal{F})^j x$. That is, $\alpha^i x - \alpha^j x = 0$. It follows that $\alpha^i(1 - \alpha^{j-i})x = 0$ which implies that $1 - \alpha^{j-i} = 0$ and α is a root of the unity. □

According to Proposition 4.3.1, if an eigenvalue is not a root of the unity, it is equal to zero. Next proposition relates the eigenvalue zero to the leaves of the graph $\mathcal{G}$.

Proposition 4.3.2. *Let $x \in \overline{\mathbb{F}}^{2^n}$ be a leaf of the graph of the (n, n)–function f. It implies the existence of a null eigenvalue of the adjacency matrix $\mathcal{F}$. Conversely, the existence of a null eigenvalue implies the existence of a leaf in the graph of f. We sometimes refer to this eigenvalue as the* eigenvalue associated to a leaf.

Proof. By definition, if $x \in \overline{\mathbb{F}}^{2^n}$ is a leaf, the column x of the adjacency matrix $\mathcal{F}$ is null. Let $v \in \overline{\mathbb{F}}^{2^n}$ be a vector whose components are all null except the one at row x which is one. Then, $\mathcal{F}v = 0$ and v is an eigenvector of the matrix $\mathcal{F}$ associated to the eigenvalue zero. Conversely, assume that there exists an eigenvector $y \in \overline{\mathbb{F}}^{2^n}$ of $\mathcal{F}$ which is associated to the eigenvalue zero. Each row of the adjacency matrix has exactly one coefficient equal to one and the others are null. The only possibility for the equality $\mathcal{F}y = 0$ to hold is that the columns of $\mathcal{F}$ corresponding to non null values of y are null. The index of each null column indicates a leaf of the graph. □

Proposition 4.3.3. *Let α be a primitive root of the unity of order ℓ and multiplicity β in the adjacency matrix of f. Then, there are β cycles in the graph of f whose lengths are a multiple of ℓ.*

Proof. Let v be an eigenvector associated to α, by definition $\mathcal{F}v = \alpha v$. Let v^i be a nonzero component of v. All the components of v that correspond to preimages of v^i have the value $\alpha^{-1}v^i$. In the same way, all the preimages of the preimages of v^i have the value $\alpha^{-2}v^i$. By continuing the process it can be seen that all the values of the vector v corresponding to the a vertex of the weakly connected component of v^i are defined.
Each weakly connected component has exactly one cycle. Let γ be the length of the cycle of the previous weakly connected component. We consider only the vertices that belong to this cycle. Let v^i be a component of that cycle, it has exactly one preimage that belongs to that cycle. As explained at the beginning of the proof, the preimage has the value $\alpha^{-1}v^i$. By continuing the process, for each vertex of the cycle we get $v^i = \alpha^{-\gamma}v^i$. Hence, $\alpha^{-\gamma} = 1$ and $\gamma = \nu\ell$ with ν an integer. The length of the cycle is a multiple of ℓ.
We now show that there are exactly β such cycles in the graph. Let w be a second eigenvector. Assume that the component w^i is different from zero. Then, there exists $\omega \in \overline{\mathbb{F}}$, $v^i = \omega w^i$. As a result, the restriction of v and w to the weakly connected component to which v^i (or w^i) belong are proportional. Hence, in order to have β independent eigenvectors, it is required to have at least β weakly connected components. With the same rational as before, we deduce that the length of the associated cycle is a multiple of ℓ. If there are more than β such components corresponding to a cycle whose length is a multiple of ℓ then there are more than β eigenvectors associated to α. That completes the proof. □

Remark 4.3.1. A path of length ℓ which does not include a leaf and is not a part of a cycle nor a part of a junction decreases the geometric multiplicity of the eigenspaces of the matrix by ℓ.

Corollary 4.3.1. *Let f be a (n, n)–function. When considering $\mathbb{F} = \mathbb{R}$, the following statements hold:*

1. *The function f is an involution if and only if the only eigenvalues of its adjacency matrix are -1 and 1;*
2. *The trace of the adjacency matrix of f is equal to the number of cycles of length one;*
3. *The multiplicity of the eigenvalue zero is equal to the number of leaves.*

Proof.

1. By definition, an involution is an invertible function, hence there are no leaf. Besides, all the vertices belong to a cycle of length one or two. According to Proposition 4.3.2 and Proposition 4.3.3, the only possible eigenvalues are -1 and 1;

2. According to Proposition 4.3.7, each cycle of length ℓ introduces exactly on time each ℓth root of unity as eigenvalues. The other eigenvalues are 0. The sum of all the ℓth roots of the unity is equal to zero except when $\ell = 1$. In such a case, it is equal to one;

3. This is a direct consequence of Proposition 4.3.2.

□

The statements of Corollary 4.3.1 concerns the eigenvalues of the adjacency matrix $\mathcal{F}$. The eigenvalues are the same for the other matrices $\mathcal{R}$, $\mathcal{A}$, $\mathcal{N}$. Therefore, the corollary also applies to them.

4.3.2 Eigenspaces

In this section we are interested in identifying the eigenspaces associated to the eigenvalues for the different matrices. Unlike eigenvalues, eigenvectors depend on the considered basis. Due to Proposition 4.2.4, once the eigenvectors of $\mathcal{F}$ are found, it is easy to deduce the eigenvectors of $\mathcal{N}$. Things get more complicated when we want to determine the eigenvectors of $\mathcal{R}$. As shown by Proposition 4.2.5, $\mathcal{F}$ and $\mathcal{R}$ are related by a change of basis and a transpose operation. Even though a matrix and its transpose are similar, there is no general change of basis suitable for any matrix to replace the transpose operation. As a consequence, in general, we cannot give the expression of the eigenvectors of $\mathcal{R}$ assuming the eigenvectors of $\mathcal{F}$. However, as shown by Proposition 4.2.5, we can do it with the expression of the eigenvectors of ${}^t\mathcal{F}$ that we can deduce from the graph. For each matrix, there is a natural way to derive a basis of the eigenspaces from the graph of the function. The rest of this section details the approach.

4.3.2.1 Adjacency matrix $\mathcal{F}$

The proof of Proposition 4.3.2 gives a basis of the eigenspace associated to the eigenvalue zero. We now deal with the case when the eigenvalue is a root of the unity. They are a direct consequence of the existence of cycles in the graph of the function. For this reason we talk about eigenvalues and eigenvectors associated to a cycle.

When considering a graph, we denote by $\mathfrak{l}(a, b)$ the length of the smallest path from the vertex a to the vertex b when it exists.

Proposition 4.3.4. *Let $\mathcal{L} = (x_0, \dots, x_{\ell-1}, x_0)$ be a cycle of length ℓ and $\overline{\mathcal{L}}$ be the set of all vertices of the weakly connected components to which $\mathcal{L}$ belongs. Let α be a primitive root of the unity of order ℓ. Then, α^i for $i = 0, \dots, \ell - 1$ is an eigenvalue of $\mathcal{F}$ for the vectors v defined as follows*

$$v^x = \begin{cases} \alpha^{-i\mathfrak{l}(x,x_0)} & \text{if } x \in \overline{\mathcal{L}} \\ 0 & \text{else} \end{cases} \tag{4.35}$$

Note that, since x_0 belongs to the cycle $\mathcal{L}$, the quantity $\mathfrak{l}(f(x), x_0)$ is defined for any $x \in \overline{\mathcal{L}}$.

Proof. Let w be the vector defined by $w = \mathcal{F}v$ then, $w^x = \sum_{y \in \mathbb{F}_2^n} \mathcal{F}_{x,y} v^y = v^{f(x)}$. Hence, the problem amounts to showing that $v^{f(x)} = \alpha^i v^x$. We stress that for $x \in \overline{\mathcal{L}}$, the following equality holds.

$$\mathfrak{l}(f(x), x_0) = \begin{cases} \ell - 1 & \text{if } x = x_0 \\ \mathfrak{l}(x, x_0) - 1 & \text{if } x \in \overline{\mathcal{L}},\ x \neq x_0 \end{cases}$$

Then,

- if $x = x_0$
  - $\alpha^i v^{x_0} = \alpha^i \alpha^{-i\mathfrak{l}(x_0,x_0)} = \alpha^i$
  - $v^{f(x_0)} = \alpha^{-i\mathfrak{l}(f(x_0),x_0)} = \alpha^i$
- if $x \in \overline{\mathcal{L}}$, $x \neq x_0$

  - $\alpha^i v^x = \alpha^i \alpha^{-i\mathfrak{l}(x,x_0)}$
  - $v^{f(x)} = \alpha^{-i\mathfrak{l}(f(x),x_0)} = \alpha^i \alpha^{-i\mathfrak{l}(x,x_0)}$
- if $x \notin \overline{\mathcal{L}}$ it follows that $f(x) \notin \overline{\mathcal{L}}$ and
  - $\alpha^i v^x = 0$
  - $v^{f(x)} = 0$

Eventually, the equality $\mathcal{F}v = \alpha^i v$ holds. □

The eigenvectors defined in the proof of Proposition 4.3.2 can be given an interpretation in terms of function:

Proposition 4.3.5. *Let f be a (n,n)–function and g a (n)–function. If $g \circ f = 0$ then the truthtable of g is an eigenvector of $\mathcal{F}^f$. Conversely, assume that g is an eigenvector associated to the eigenvalue 0 such that all its components are either zero or one then, g can be seen as a (n)–function and $g \circ f = 0$.*

Proof. The vector $\mathcal{F}^f g = 0$ is equal to the second column of $\mathcal{F}^{g\circ f}$ hence, the matrix $\mathcal{F}^{g\circ f}$ has its first column equal to one and the second equal to zero. This is the adjacency matrix of the zero function. □

Note that once the eigenvectors associated to 0 are obtained as explained in the proof of Proposition 4.3.2, it is easy to determine all the functions v for which Proposition 4.3.5 applies. They are the set of all linear combinations with coefficients in $\{0,1\}$ of the eigenvectors associated to 0. There are no other. Hence, if there are ℓ leaves in the graph there are exactly 2^ℓ (n)–functions g such that $g \circ f = 0$.

4.3.2.2 Numerical matrix $\mathcal{N}$

The eigenvectors of $\mathcal{F}$ being determined, the change of basis of Proposition 4.2.4 can be used to determine the eigenspaces of $\mathcal{N}$. If v is an eigenvector of $\mathcal{N}$ then $\widetilde{v} = Nv$ is an eigenvector for $\mathcal{N}$.
If v is an eigenvector of $\mathcal{F}$ associated to the eigenvalue 0 then, v has all its components equal to zero except the one with index s that corresponds to a leaf. Therefore, $\widetilde{v}(u) = \sum_{x\in\mathbb{F}_2^n} (-1)^{\mathrm{hw}(x)-\mathrm{hw}(u)} u\hat{}\,x v^x = (-1)^{\mathrm{hw}(x_s)-\mathrm{hw}(u)} u\hat{}\,x_s$.

4.3.2.3 Transpose of the adjacency matrix ${}^t\mathcal{F}$

Next proposition shows that, when considering the transpose of the adjacency matrix ${}^t\mathcal{F}$, eigenvectors corresponding to the zero eigenvalue can easily be deduced from the junctions.

Proposition 4.3.6. *Let the vertex y be a junction of the (n,n)–function f and x_1 and x_2 two incident vertices of this junction. Let e_{x_1} (respectively e_{x_2}) be the 2^n-dimensional vector with all its components null except the one at coordinate x_1 (respectively x_2) which is equal to one. Then, the vector $e = e_{x_1} - e_{x_2}$ is an eigenvector of the matrix ${}^t\mathcal{F}$ corresponding to the eigenvalue 0.*

Proof. By assumption, the equality $f(x_1) = f(x_2)$ holds. Let $y = f(x_1)$ and e_y be the 2^n-dimensional vector with all its components equal to 0 except the coordinate y which is equal to one. According to Propositions 4.2.3 and 4.2.5, $e_y = {}^t\mathcal{F} e_{x_1}$ and $e_y = {}^t\mathcal{F} e_{x_2}$. By subtracting the two equations we get ${}^t\mathcal{F}(e_{x_1} - e_{x_2}) = 0$ which completes the proof. □

Proposition 4.3.7. *Let $\mathcal{L} = (x_0, \ldots, x_{\ell-1}, x_0)$ be a cycle of length ℓ and α be a primitive root of the unity of order ℓ. Then, for $i = 0, \ldots, \ell-1$, α^i is an eigenvalue of ${}^t\mathcal{F}$. For each value of i, the corresponding eigenvector can be constructed as follows.*

$$v^x = \begin{cases} \alpha^{i\mathfrak{l}(x,x_0)} & \text{if } x \in \mathcal{L} \\ 0 & \text{else} \end{cases} \tag{4.36}$$

Proof. Let w be the vector defined by $w = {}^t\mathcal{F} v$ then, $w^x = \sum_{y\in\mathbb{F}_2^n} \mathcal{F}_{x,y} v^y = \sum_{y\in\mathbb{F}_2^n,\, x=f(y)} v^y$. We want to show that $w = \alpha^k v$. The two following cases can be distinguished.

- if $x \notin \mathcal{L}$, there is no element $y \in \mathcal{L}$ such that $x = f(y)$. As a result, all the vertices v^y involved in the sum $\sum_{y \in \mathbb{F}_2^n,\, x=f(y)} v^y$ are null and $w^x = 0$, by assumption, $\alpha^i v^x = 0$;

- if $x \in \mathcal{L}$, there is a single y such that $x = f(y)$ and $v^y \neq 0$. The element y belongs to $\mathcal{L}$. As a consequence, $w^x = v^y$ that is, $w^{f(y)} = v^y$. By assumption, the following equalities hold $v^y = \alpha^{i\mathfrak{l}(y,x_0)} = \alpha^i \alpha^{i(\mathfrak{l}(y,x_0)-1)} = \alpha^i \alpha^{i\mathfrak{l}(f(y),x_0)} = \alpha^i v^{f(y)}$ and finally $w^{f(y)} = v^y = \alpha^i v^{f(y)}$.

Eventually, the equality $w = {}^t\mathcal{F} v = \alpha^i v$ holds. □

Note that if there is a cycle, the eigenvalue 1 always exists no matter what is its length.

Remark 4.3.2. The multiplicity of the eigenvalue 1 is equal to the number of weakly connected components. The rational is that there is exactly one cycle per weakly connected component. Each cycle introduces exactly one time the eigenvalue one as justified by Proposition 4.3.7.

Remark 4.3.3. The number of junctions (taking into account the multiplicities) is equal to the number of leaves. Hence, for each junction of multiplicity β, it is possible to get β independent eigenvectors associated to the eigenvalue 0.

4.3.2.4 Correlation matrix $\mathcal{R}$

The result of the last section can be used to deduce the eigenspaces of $\mathcal{R}$. If v is an eigenvector for ${}^t\mathcal{F}$ then, due to (4.5), the Fourier transform of v, denoted by $\widehat{v}$ is an eigenvector of $\mathcal{W}$ and so of $\mathcal{R}$.
Let v be an eigenvector of ${}^t\mathcal{F}$ associated to the eigenvalue 0. From Proposition 4.3.6, it has only two nonzero components x_1 and x_2 corresponding to preimage vertices of a junction. Therefore,

$$\widehat{v}(u) = \sum_x v(x)(-1)^{x \cdot u} = (-1)^{x_2 \cdot u} - (-1)^{x_1 \cdot u} \tag{4.37}$$

Due to (4.15), the eigenvalues of the Walsh matrix are the eigenvalues of $\mathcal{R}$ times 2^n. The eigenvectors associated to the corresponding eigenvalues are the same.

4.3.3 Example

From the graph in Figure 4.1, the following can be inferred. The three leaves 000, 111 and 011 are responsible for the eigenvalue 0 with multiplicity three. The cycle $(101, 101)$ of length one is responsible for the eigenvalue 1 with multiplicity one. The cycle $(010, 001, 100, 010)$ of length three is responsible for the eigenvalues α, α^2, 1, the three 3rd roots of the unity, with multiplicity one.

eigenvalue	multiplicity
0	3
1	2
α	1
α^2	1

Table 4.5: Eigenvalues of the matrix representations

4.3.3.1 Adjacency matrix

We determine the corresponding eigenvectors for the adjacency matrix $\mathcal{F}^{f_{e2}}$. According to the proof of Proposition 4.3.2, the eigenvectors associated to 0 are

$$a_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \; a_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \; a_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

According to Proposition 4.3.4, the following eigenvector associated to the eigenvalue 1 exists and is

$$a_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

According to Proposition 4.3.4, the following eigenvectors exist and are respectively associated to the eigenvalues 1, α, α^2.

$$a_4 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}, \; a_5 = \begin{pmatrix} \alpha \\ 1 \\ \alpha^2 \\ 0 \\ \alpha \\ 0 \\ \alpha \\ 1 \end{pmatrix}, \; a_6 = \begin{pmatrix} \alpha^2 \\ 1 \\ \alpha \\ 0 \\ \alpha^2 \\ 0 \\ \alpha^2 \\ 1 \end{pmatrix}$$

The eigenvectors of the matrix $\mathcal{N}^{f_{e2}}$ (respectively $\mathcal{A}^{f_{e2}}$) can be derived by multiplying N (respectively $N \mod 2$) by the vectors a_0, a_1, a_2, a_3, a_4, a_5, a_6 as explained in Section 4.3.2.

4.3.3.2 Transpose of the adjacency matrix $\mathcal{F}^{f_{e2}}$

We determine the eigenvectors of the transpose of the adjacency matrix ${}^t\mathcal{F}^{f_{e2}}$. According to Proposition 4.3.6, the three eigenvectors associated to 0 are related to

- the junction 010 with preimage set $\{000, 100, 110\}$ and hence multiplicity two;
- the junction 101 with preimage set $\{011, 101\}$ and hence multiplicity one.

They are denoted by a_0, a_1, a_2 and can be derived as follows.

$$\underbrace{\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{a_0} = \underbrace{\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{v_{000}} - \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{v_{100}}, \quad \underbrace{\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \end{pmatrix}}_{a_1} = \underbrace{\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{v_{000}} - \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}}_{v_{110}},$$

$$\underbrace{\begin{pmatrix}0\\0\\0\\1\\0\\-1\\0\\0\end{pmatrix}}_{a_2} = \underbrace{\begin{pmatrix}0\\0\\0\\1\\0\\0\\0\\0\end{pmatrix}}_{v_{011}} - \underbrace{\begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{pmatrix}}_{v_{101}}$$

According to Proposition 4.3.7, due to the cycle $(101, 101)$, the following eigenvector exists for the eigenvalue 1.

$$a_3 = \begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{pmatrix}$$

According to Proposition 4.3.7, the following eigenvectors exist and are respectively associated to the eigenvalues 1, α, α^2.

$$a_4 = \begin{pmatrix}0\\1\\1\\0\\1\\0\\0\\0\end{pmatrix}, a_5 = \begin{pmatrix}0\\\alpha\\\alpha^2\\0\\1\\0\\0\\0\end{pmatrix}, a_6 = \begin{pmatrix}0\\\alpha^2\\\alpha\\0\\1\\0\\0\\0\end{pmatrix},$$

The eigenvectors of the correlation matrix can be obtained by multiplying the Hadamard matrix H by the vectors a_0, a_1, a_2, a_3, a_4, a_5, a_6 as explained in Section 4.3.2.

4.4 Reduced matrix

We define a reduction transformation that applies to the matrix representations of vectorial Boolean functions. This reduction transformation is the base of the self-synchronizing characterization presented in Section 5.3 of Chapter 5.

Definition 4.4.1 (Reduced matrix)**.** For any square correlation matrix $\mathcal{R}$ of dimension 2^n, let us define its reduced matrix $\mathcal{R}^\bullet$ of dimension $(2^n-1)\times(2^n-1)$ which is the matrix $\mathcal{R}$ in which the first row and column have been removed.

$$\mathcal{R}^\bullet = \begin{pmatrix}\mathcal{R}_{1,1} & \cdots & \mathcal{R}_{1,q-1}\\ \vdots & & \vdots\\ \mathcal{R}_{q-1,1} & \cdots & \mathcal{R}_{q-1,q-1}\end{pmatrix}$$

The reduced Walsh matrix $\mathcal{W}^\bullet$, reduced algebraic matrix $\mathcal{A}^\bullet$ and reduced numerical $\mathcal{N}^\bullet$ are defined in the same way.
The question on whether it is possible to uniquely reconstruct the original function f from this reduced correlation matrix, or equivalently from the reduced Walsh matrix is interesting. To answer this question, let us state the following proposition:

Proposition 4.4.1. *A (n)–function f can be uniquely recovered from its last 2^n-1 Walsh coefficients provided that it is not a constant function.*

Proof. The inverse transform formula (4.7) requires the knowledge of the 2^n coefficients. The absolute value of the missing coefficient can be found using Parseval theorem (Theorem 4.1.1). The ambiguity is on the sign of the coefficient. Next, we show that, if the function is not a constant one, the value of the correct sign can always be found and the original function can systematically be reconstructed. The inverse Fourier transform formula (4.7) reads:

$$f_\chi(x) = \frac{1}{2^n}\widehat{\widehat{f_\chi}}(x) = \frac{1}{2^n}\left[\sum_{u\in\mathbb{F}_2^n}(-1)^{x\cdot u}\widehat{f_\chi}(u)\right] = \frac{1}{2^n}\left[\widehat{f_\chi}(0) + \sum_{u\in\mathbb{F}_2^{n*}}(-1)^{x\cdot u}\widehat{f_\chi}(u)\right] \tag{4.38}$$

For simplicity, let $a = \widehat{f_\chi}(0)$. The expression of f_χ reads

$$f_\chi(x) = \frac{1}{2^n}\left[a + \sum_{u\in\mathbb{F}_2^{n*}}(-1)^{x\cdot u}\widehat{f_\chi}(u)\right] \tag{4.39}$$

The function f can be recovered if there is no function $g : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ different from f whose sign function reads

$$g_\chi(x) = \frac{1}{2^n}\left[-a + \sum_{u\in\mathbb{F}_2^{n*}}(-1)^{x\cdot u}\widehat{f_\chi}(u)\right] \tag{4.40}$$

From (4.39), it can be deduced that

$$\sum_{u\in\mathbb{F}_2^{n*}}(-1)^{x\cdot u}\widehat{f_\chi}(u) = 2^n f_\chi(x) - a \tag{4.41}$$

Then, after replacing (4.41) into (4.40), g_χ reads

$$g_\chi(x) = \frac{1}{2^n}\left[-2a + 2^n f_\chi(x)\right] = -\frac{2a}{2^n} + f_\chi(x) \tag{4.42}$$

Since f is a Boolean function, $f_\chi(x)$ is either 1 or -1 and so are the values of $g_\chi(x)$. The following cases determine the value of a for all the possible situations:

- if $f_\chi(0) = 1$ then, $g_\chi(0) = -2^{1-n}a + 1$ and
  - if $g_\chi(0) = 1$ then, $a = 0$
  - if $g_\chi(0) = -1$ then, $a = 2^n$
- if $f_\chi(0) = -1$ then, $g_\chi(0) = -2^{1-n}a - 1$ and
  - if $g_\chi(0) = 1$ then, $a = -2^n$
  - if $g_\chi(0) = -1$ then, $a = 0$

It follows that

- if $a = 0$, the functions f and g are the same and can be uniquely recovered;
- if $a = 2^n$, from Parseval theorem (Theorem 4.1.1), all the other coefficients are null: $\forall u \in \mathbb{F}_2^{n*}$, $\widehat{f_\chi}(u) = 0$. It can be deduced that the function f is $x \longmapsto 0$;
- if $a = -2^n$, from Parseval theorem (Theorem 4.1.1), all the other coefficients are null: $\forall u \in \mathbb{F}_2^{n*}$, $\widehat{f_\chi}(u) = 0$. It can be deduced that the function f is $x \longmapsto 1$.

□

We give below a vectorial counterpart of Proposition 4.4.1.

Proposition 4.4.2. *A (n, m)–function can be uniquely recovered from its reduced Walsh coefficients provided that it is not a constant function.*

Proof. Let f be a (n, m)–function. The rows of its Walsh matrix $\mathcal{W}$ are the Walsh transforms of all the linear combinations of the coordinate functions of f. It suffices to reconstruct these coordinates to retrieve the function. From Proposition 4.4.1, this is possible if and only if these coordinates are not constant. Now, assume that the coordinate function f^i is constant and that there is a non constant coordinate function f^j. Then, there is a row in $\mathcal{W}$ which is the Walsh transform of $f^i \oplus f^j$. The function $f^i \oplus f^j$ is not constant and so, Proposition 4.4.1 applies. Let $g = f^i \oplus f^j$. If f^i is $x \longmapsto 0$ then, $\widehat{g_\chi} = \widehat{f^j_\chi}$ and if f^i is $x \longmapsto 1$ then, $\widehat{g_\chi} = -\widehat{f^j_\chi}$. Therefore, it is always possible to decide whether or not f^i is the constant function $x \longmapsto 0$ or the constant function $x \longmapsto 1$ provided that there is at least one non constant coordinate function. □

As a conclusion, the only case when a (n, m)–function f cannot be built from its reduced Walsh matrix is when it is one of the 2^m constant functions.

Remark 4.4.1. In other words, the reduction transformation of Definition 4.4.1 defines an equivalence relation in which all the constant functions are in the same class and where each non constant function is in its own class. This is important as our approach for characterizing self-synchronization needs to distinguish non constant functions of constant functions. Besides, when we are concerned with constant functions, we do not care about distinguishing them. Due to relation (4.15), the result also holds for the correlation matrix.

The important properties of Boolean functions being presented, we consider again systems and their dynamics in order to decide whether or not they are self-synchronizing.

Chapter 5

Characterization of self-synchronization for Boolean dynamical systems

The purpose of this chapter is to characterize self-synchronization in the context of Boolean dynamical systems. As explained in Section 2.4.2, the self-synchronization property is directly related to the choice of the next-state function. We aim at giving a characterization of these functions. Section 5.1 gives preliminary results. Section 5.2 is devoted to a first characterization based on the notion of influence of the initial state. Section 5.3 provides a second characterization based on matrix representations and nilpotent matrices. The consequence on the graph representation of the function is discussed.

We address the issue of self-synchronization of the mater-slave setup recalled below. The plaintext symbol m_k and ciphertext symbol c_k lie in the set $\mathbb{F}_2^{n_s}$. The state x belongs to the vector space $\mathbb{F}_2^n$. In this chapter, we do not consider any cryptographic issue. These concerns will be addressed in Chapter 6. The next state function f is a $(n_s + n, n)$–function and h is a (n, n_s)–function. The system that we consider is described by the following equations:

$$\mathcal{E} \begin{cases} x_{k+1} &= f[\kappa](c_k, x_k) \\ c_k &= e(h[\kappa](x_k), m_k) \end{cases} \tag{5.1}$$

$$\mathcal{D} \begin{cases} \hat{x}_{k+1} &= f[\kappa](c_k, \hat{x}_k) \\ \hat{m}_k &= e(h[\kappa](\hat{x}_k), c_k) \end{cases} \tag{5.2}$$

It corresponds to the generalized form of an SSSC whenever f is self-synchronizing (see Section 2.4.2). The corresponding scheme is depicted in Figure 5.1. The parameter κ indicates that the functions depend on the key κ. Since this parameter is only relevant for security purposes, it will be omitted hereafter to make the equations more readable.

The following decomposition of a $(n + n_s, n)$–function was introduced in [103] and is called Shannon decomposition. Any $(n + n_s, n)$–function can be decomposed as follows

$$f(c, x) = \begin{cases} f_{0\cdots0}(x) & \text{if } c = (0, \ldots, 0) \\ \vdots & \\ f_{1\cdots1}(x) & \text{if } c = (1, \ldots, 1) \end{cases}$$

Shannon decomposition is illustrated in Figure 5.2 where MUX is a multiplexor.
When $n_s = 1$, the next state function reduces to $f(c, x) = (1 \oplus c)f_0(x) \oplus cf_1(x)$.

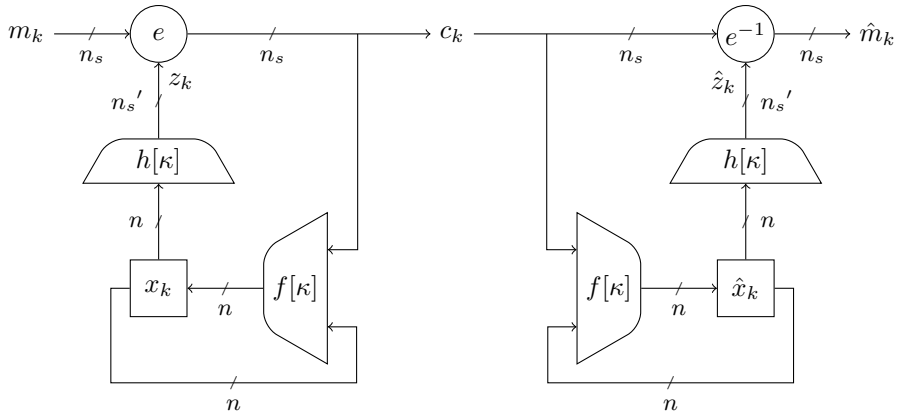


Figure 5.1: Generalized form

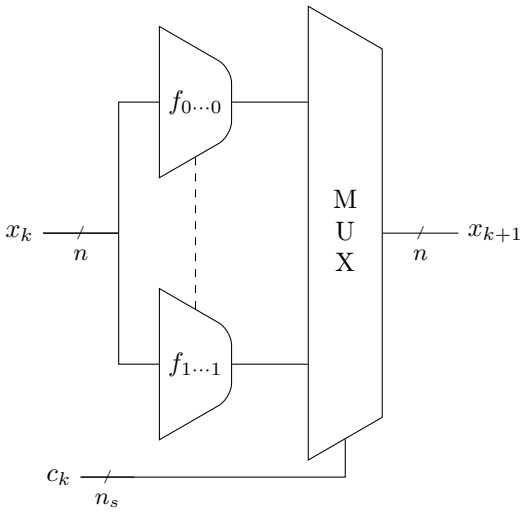


Figure 5.2: Shannon decomposition

5.1 Preliminary results

This section gives preliminary results on correlation matrices (a spectral representation of vectorial Boolean functions) that we need throughout this chapter. The kth-order iterated function $f^{\circ k}$ (such as specified in Definition 2.1.7 in Chapter 2) restricted to a specific sequence $(c)_0^{k-1}$ is denoted by $f^{\to(c)_0^{k-1}}$ (or $f^{\to c}$ if not misleading). It is a (n,n)–function.

Proposition 5.1.1. *For a given sequence $(c)_0^{k-1}$ of elements of $\mathbb{F}_2^{n_s}$, the correlation matrix of $f^{\to c}$ is*

$$\mathcal{R}^{f^{\to c}} = \mathcal{R}^{f_{c_{k-1}}} \times \cdots \times \mathcal{R}^{f_{c_0}} \tag{5.3}$$

Proof. The proof is a direct consequence of Proposition 4.2.1. □

Roughly speaking, the composition in the time domain corresponds to the multiplication in the spectral domain.

For two vectors $u \in \mathbb{F}_2^\ell$ and $v \in \mathbb{F}_2^n$, their concatenation, denoted by $u|v$, is by definition the $(n+\ell)$ dimensional vector $u|v = (u^0, \ldots, u^{\ell-1}, v^0, \ldots, v^{n-1})$. We express the correlation matrix of $f^{\circ\ell}$, the ℓth order iterated function of the $(n+n_s, n)$–function f.

Proposition 5.1.2. *Let $v, t \in \mathbb{F}_2^n$, $u \in \mathbb{F}_2^\ell$, $z = u|v$. The entries of the correlation matrix of the iterated function $f^{\circ\ell}$ are defined by*

$$\mathcal{R}_{t,z}^{f^{\circ\ell}} = \mathcal{R}_{t,u|v}^{f^{\circ\ell}} = \frac{1}{2^\ell} \sum_{c \in \mathbb{F}_2^\ell} (-1)^{u \cdot c} \mathcal{R}_{t,v}^{f^{\to c}} \tag{5.4}$$

Proof. By definition, the coefficient at row $t \in \mathbb{F}_2^n$ and column $z \in \mathbb{F}_2^{n+\ell}$ of the matrix $\mathcal{R}^{f^{\circ\ell}}$ is

$$\mathcal{R}_{t,z}^{f^{\circ\ell}} = \mathcal{R}_{t,u|v}^{f^{\circ\ell}} = \frac{1}{2^{n+\ell}} \sum_{x \in \mathbb{F}_2^n, c \in \mathbb{F}_2^\ell} (-1)^{t \cdot f^{\circ\ell}(c,x) \oplus (u|v) \cdot (c|x)} \tag{5.5}$$

$$= \frac{1}{2^{n+\ell}} \sum_{x \in \mathbb{F}_2^n, c \in \mathbb{F}_2^\ell} (-1)^{t \cdot f^{\circ\ell}(c,x) \oplus u \cdot c \oplus v \cdot x} = \frac{1}{2^\ell} \sum_{c \in \mathbb{F}_2^\ell} (-1)^{u \cdot c} \mathcal{R}_{t,v}^{f^{\to c}} \tag{5.6}$$

□

According to Proposition (5.1.2), the correlation matrix of $f^{\circ\ell}$ can be expressed as sums and differences of the Walsh matrices $\mathcal{R}^{f^{\to c}}$ obtained for all sequences $(c)_0^{\ell-1}$.

Proposition 5.1.3. *The sequence (c) is synchronizing if and only if $\mathcal{R}^{f^{\to c}}$, the correlation matrix of $f^{\to c}$, is the matrix of a constant function. That is, according to* (4.31)*, a $2^n \times 2^n$ matrix of the form*

$$\mathcal{R}^{f^{\to c}} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \pm 1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ \pm 1 & 0 & \cdots & 0 \end{pmatrix}$$

Proof. By definition, if (c) is a synchronizing sequence, $f^{\to c}(x)$ does not depend on x. Thus, $f^{\to c}$ is a constant function. And yet, any linear combination of the coordinate functions of $f^{\to c}$ is also a constant function. □

Remark 5.1.1. Let f be a $(n + n_s, n)$–function and $\{f_c, c \in \mathbb{F}_2^{n_s}\}$ be the 2^{n_s} (n, n)–functions corresponding to Shannon decomposition. These functions are square and admit a univariate polynomial representation as explained in Section 4.2.5. When considering the univariate polynomial representation, Proposition 5.1.3 amounts to ensuring that for some k, whatever is the sequence $(c)_0^{k-1}$ of elements of $\mathbb{F}_2^{n_s}$, the function $f_{c_k} \circ \cdots \circ f_{c_0}$ is constant. With the univariate polynomial representation, constant functions are those whose coefficients of degree different from zero are null. It has to be related to Remark 4.4.1.

5.2 Self-synchronization based on the influence

This section provides a characterization of self-synchronization through the notion of influence. Roughly speaking, the influence of a variable reveals the ability of this variable to change the output of a function. And yet, the general principle of SSSC is that, for sufficiently long sequences, all the iterated functions of the initial state should no longer depend on the initial state. In other words, the initial state should not be influential. For Boolean functions, the notion of influence has been addressed in several papers (e.g. [104, 105] to mention a few). However, it was essentially in the context of game theory. For reasons explained later, we must revisit the existing formal definitions because they are not suited for our purpose. The results of this section have been published in [106].

5.2.1 Influence of a single variable

Let f be a Boolean function of the variable $x \in \mathbb{F}_2^n$. The influence of one coordinate x^i over a Boolean function f is defined as the probability that the value of $f(x)$ varies if the value of the component x^i varies too, the other coordinates being constant and set at random.

Definition 5.2.1 (Influence of a single variable [104])**.** Let $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ be a Boolean function and $i \in \{1, \ldots, n\}$ a set of integers. Let e_i be the n–dimensional vector whose components are zero except the ith one which is equal to 1. The influence of x^i on f is denoted by $I_f(i)$ and is defined by

$$I_f(i) = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} [f(x) + f(x \oplus e_i)]$$

Remark 5.2.1. This is related to the auto-correlation function of f as defined in Definition 4.1.4 of Section 4.1.3 as follows

$$I_f(i) = 2^{2n-1} - 2^{n-1}(f \otimes f)(e_i) \tag{5.7}$$

5.2.2 Influence of a set of variables

In this section, we consider the influence of a set S of variables. A variable may be identified to its index. Thus, for short, the set S may be also considered as a set of indexes in $\{1, \ldots, n\}$. There exists several ways to extend Definition 5.2.1. None is more natural than the others. The relevance of the definition depends on the context.

The influence of a subset S of components of x can be defined as the probability that the value $f(x)$ changes if one of the variables in S changes too. It does not take into account how often the variables of the set S are able to change the value $f(x)$. A more suitable definition for our purpose should involve the balancedness of the restricted function obtained by fixing the variables not in the set S. This is the reason why we should rather introduce a new definition of the influence that takes these points into account.

Definition 5.2.2 (Influence of a set of variables [106])**.** Let $f(x)$ be a Boolean function of n variables, S be a set of ℓ components of x. The influence $I_f(S)$ is:

$$I_f(S) = \frac{1}{2^n(2^\ell - 1)} \sum_{x \in \mathbb{F}_2^n} \sum_{u \in \mathbb{F}_2^n,\, u \neq 0,\ \mathrm{supp}(u) \subseteq S} [f(x) \oplus f(x \oplus u)] \tag{5.8}$$

In other words, the influence of a set of variables is the mean of the probabilities that $f(x)$ changes when x is uniformly randomly chosen, the mean being computed for all possible combinations of the values of the variables in S.

Remark 5.2.2. When the set S reduces to one element, Definition 5.2.1 and Definition 5.2.2 coincide.

5.2.3 Spectral expression of the influence

The influence of a set of variables over a Boolean function f can simply be expressed by means of its spectral representation.

Proposition 5.2.1 ([106])**.** *Let $f(x)$ be a Boolean function of n variables, S be a set of ℓ components of x. The influence $I_f(S)$ is:*

$$I_f(S) = \frac{2^{\ell-1}}{2^{2n}(2^\ell - 1)} \sum_{v \in \mathbb{F}_2^n,\, \mathrm{supp}(v) \cap S \neq \emptyset} \left[\widehat{f_\chi}(v)\right]^2 \tag{5.9}$$

Proof. For any vector u, let $f_u : x \longmapsto f(x) - f(x \oplus u)$. It is easy to see that $f_u(x) = 0$ if $f(x) = f(x \oplus u)$ and $f_u(x) = \pm 1$ if $f(x) \neq f(x \oplus u)$. This implies that $[f_u(x)]^2 = f(x) \oplus f(x \oplus u)$. Therefore, by using (4.8),

$$I_f(S) = \frac{1}{2^{2n}(2^\ell - 1)} \sum_{v \in \mathbb{F}_2^n} \sum_{u \in \mathbb{F}_2^n,\, u \neq 0,\ \mathrm{supp}(u) \subseteq S} \left[\widehat{f_u}(v)\right]^2 . \tag{5.10}$$

By expressing $\widehat{f_u}(v)$ by means of $\widehat{f}(v)$, we get

$$\widehat{f_u}(v) = \sum_{x \in \mathbb{F}_2^n} f(x)(-1)^{v \cdot x}(1 - (-1)^{v \cdot u}) = (1 - (-1)^{v \cdot u})\widehat{f}(v), \tag{5.11}$$

By substituting (5.11) into (5.10), we get

$$I_f(S) = \frac{1}{2^{2n}(2^\ell - 1)} \sum_{v \in \mathbb{F}_2^n} \left[\widehat{f}(v)\right]^2 \sum_{u \in \mathbb{F}_2^n,\, u \neq 0,\ \mathrm{supp}(u) \subseteq S} \left[1 - (-1)^{v \cdot u}\right]^2, \tag{5.12}$$

and thus,

$$I_f(S) = \frac{1}{2^{2n-2}(2^\ell - 1)} \sum_{v \in \mathbb{F}_2^n,\, \mathrm{supp}(v) \cap S \neq \emptyset} \left[\widehat{f}(v)\right]^2. \tag{5.13}$$

□

Remark 5.2.3. Definition 5.2.2 of the influence of variables is very close, up to a factor that depends on the cardinality of S, to the definition of the so-called *variable variation* given in [105].

Proposition 5.2.2 ([106])**.**

1. *The function f is bent if and only if, for all non-empty subset S of variable indexes, one has $I_f(S) = \frac{1}{2}$;*
2. *The function f does not depend on the variables in the subset S if and only if $I_f(S) = 0$.*

Proof.

1. If f is bent, then $\forall u \in \mathbb{F}_2^n$, $\widehat{f_\chi}(u) = \pm 2^{\frac{n}{2}}$. Then replacing this expression in (5.9), we get

$$I_f(S) = \frac{2^{\ell-1} \times 2^n}{2^{2n}(2^\ell - 1)} \sum_{v \in \mathbb{F}_2^n,\, \mathrm{supp}(v) \cap S \neq \emptyset} 1 = \frac{2^{\ell-1} \times 2^n}{2^{2n}(2^\ell - 1)}(2^n - 2^{n-\ell}) = \frac{1}{2}$$

   Conversely, if for all non-empty subset of variable indexes S of k elements, one has $I_f(S) = \frac{1}{2}$, then, by replacing in relation (5.9), one gets

$$\frac{2^{\ell-1}}{2^{2n}(2^\ell - 1)} \sum_{v \in \mathbb{F}_2^n,\, \mathrm{supp}(v) \cap S \neq \emptyset} \left[\widehat{f_\chi}(v)\right]^2 = \frac{1}{2}.$$

   From Parseval Theorem (Theorem 3.1.3), and as $\mathrm{supp}(v) \cap S \neq \emptyset \iff \mathrm{supp}(v) \subseteq \overline{S}$, where $\overline{S}$ denotes the complementary set of S, we get that

$$2^{2n} - \sum_{v \in \mathbb{F}_2^n,\, \mathrm{supp}(v) \subseteq \overline{S}} \left[\widehat{f_\chi}(v)\right]^2 = 2^{2n-\ell}(2^\ell - 1),$$

   Thus,

$$\sum_{v \in \mathbb{F}_2^n,\, \mathrm{supp}(v) \subseteq \overline{S}} \left[\widehat{f_\chi}(v)\right]^2 = 2^{2n} - 2^{2n-\ell}(2^\ell - 1) = 2^{2n-\ell} \tag{5.14}$$

   When applying this relation with $S = \{1, \ldots, n\}$, the sum (5.14) has only one term which is $[\widehat{f_\chi}(0)]^2 = 2^{2n-n} = 2^n$. The other values are obtained by induction on the weight of vector u. Let v be a nonzero vector. Let us choose S such that $\overline{S} = \mathrm{supp}(u)$. One can split the sum of relation (5.14) into the term $[\widehat{f_\chi}(v)]^2$ and the $2^{n-\ell} - 1$ others terms which are all equal to 2^n by the induction hypothesis as they all have weight strictly lower than the weight of v. Thus, $[\widehat{f_\chi}(u)]^2 + (2^{n-\ell} - 1) \cdot 2^n = 2^{2n-\ell}$ which completes the proof.

2. If f is constant with respect to the variables in S, $\forall x, \forall v \in \mathrm{supp}(S)$, $f(x) \oplus f(x \oplus v) = 0$ thus, $I_f(S) = 0$. Conversely, $I_f(S) = 0$ implies that for all v the terms $f(x) \oplus f(x \oplus v)$ equal 0 since $I_f(S)$ is a sum of positive terms.

□

5.2.4 Extension of the influence to vectorial Boolean functions

In this section, we extend the definition of the influence to vectorial Boolean functions having in mind the characterization of the self-synchronization property of f as a next-state function.

Definition 5.2.3 (Influence of a set of variable on a vectorial Boolean function [106])**.** The influence of a set of variables S over a vectorial Boolean function f is the mean of the influence of S over each coordinate function f^j.

$$I_f(S) = \frac{1}{q}\sum_{j=0}^{q-1} I_{f^j}(S) \tag{5.15}$$

Proposition 5.2.3 ([106])**.** *If f does not depend on the variables in S then, the influence is $I_f(S) = 0$.*

Proof. This is a simple consequence of Proposition 5.2.2. □

5.2.5 Self-synchronization and influence

We aim at relating the self-synchronization property of the function f stated in Definitions 2.4.1 and 2.4.2 to the influence of the initial state on the corresponding iterated function $f^{\circ k}$ of the dynamical systems (5.1) and (5.2). Let S_x denote the subset of variables that corresponds to the initial state x.

Proposition 5.2.4 ([106])**.** *The function f is finite-time self-synchronizing if and only if, there exists an integer k large enough so that for any sequence $(c)_0^{k-1}$, the iterated function $f^{\circ k}((c)_0^{k-1}, x)$ does not depend on the internal state x. In other words, the variable x of $f^{\circ k}((c)_0^{k-1}, x)$ has no longer any influence after a finite transient time that is, $I_{f^{\circ k}}(S_x) = 0$*

It can be inferred that this implies for $\mathcal{R}^{f^{\circ k}}$ to be sparse. The only possible nonzero coefficients are located on the column $(v_c|v_x) \in \mathbb{F}_2^{n+kn_s}$ such that $v_x = 0$.

5.3 Self-synchronization based on nilpotent semigroups

Section 5.3.1 aims at providing another characterization of self-synchronizing vectorial Boolean functions and at classifying them. It is based on the reduced matrices defined in Section 4.4. A graph condition required for self-synchronization on the function resulting from Shannon decomposition is provided in Section 5.3.2.1.

5.3.1 Matrix-based approach

5.3.1.1 Characterization

Let us first notice some important features of correlation matrices. In the sequel, we consider $\mathcal{R}$ to be a square correlation matrix of dimension $q \times q$.

$$\mathcal{R} = \begin{pmatrix} \mathcal{R}_{0,0} & 0 & \cdots & 0 \\ \mathcal{R}_{1,0} & \mathcal{R}_{1,1} & \cdots & \mathcal{R}_{1,q-1} \\ \vdots & \vdots & & \vdots \\ \mathcal{R}_{q-1,0} & \mathcal{R}_{q-1,1} & \cdots & \mathcal{R}_{q-1,q-1} \end{pmatrix} \tag{5.16}$$

The matrix $\mathcal{R}$ can be decomposed as $\mathcal{R} = \mathcal{R}_A + \mathcal{R}_N$ with

$$\mathcal{R}_A = \begin{pmatrix} \mathcal{R}_{0,0} & 0 & \cdots & 0 \\ \mathcal{R}_{1,0} & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ \mathcal{R}_{q-1,0} & 0 & \cdots & 0 \end{pmatrix} \qquad \mathcal{R}_N = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & \mathcal{R}_{1,1} & \cdots & \mathcal{R}_{1,q-1} \\ \vdots & \vdots & & \vdots \\ 0 & \mathcal{R}_{q-1,1} & \cdots & \mathcal{R}_{q-1,q-1} \end{pmatrix}$$

A matrix is said to be of type A if the only nonzero coefficients are located on the first column. A matrix is said to be of type N if all the coefficients of the first row and first column are zero.
We recall that $\mathcal{R}^\bullet$ denotes the reduced matrix of $\mathcal{R}$ as given by Definition 4.4.1.

Remark 5.3.1. The reduced matrix of $\mathcal{R}_N$ is $\mathcal{R}^\bullet$ as well.

Remark 5.3.2. It is straightforward to verify the following results:

- the product of any two matrices of type A is a matrix of type A;
- the product of any matrix of type A with any matrix of type N is a zero matrix;
- the product of any matrix of type N with any matrix of type A is a matrix of type A;
- the product of any two matrices of type N is a matrix of type N.

Proposition 5.3.1 ([99]). *Consider a sequence $(c)_0^k$ of symbols of $\mathbb{F}_2^{n_s}$. The correlation matrices of the 2^{n_s} (n,n)–functions $\{f_c,\, c \in \mathbb{F}_2^{n_s}\}$ corresponding to Shannon decomposition of a $(n+n_s,n)$–function f are $\mathcal{R}^{f_c} = \mathcal{R}_A^{f_c} + \mathcal{R}_N^{f_c}$. The product $\mathcal{R} = \mathcal{R}^{f_{c_k}} \times \cdots \times \mathcal{R}^{f_{c_0}}$ is of type A if and only if the matrix $\mathcal{R}_N^{f_{c_k}} \times \cdots \times \mathcal{R}_N^{f_{c_0}}$ is null.*

Proof.

$$\mathcal{R} = \mathcal{R}_{f_{c_k}} \times \cdots \times \mathcal{R}_{f_{c_0}} = (\mathcal{R}_A^{f_{c_k}} + \mathcal{R}_N^{f_{c_k}}) \times \cdots \times (\mathcal{R}_A^{f_{c_0}} + \mathcal{R}_N^{f_{c_0}}) \tag{5.17}$$

By expanding the expression and using Remark 5.3.2, $\mathcal{R}$ can be rewritten $\mathcal{R} = \mathcal{R}_A + \mathcal{R}_N$ with a type-A matrix $\mathcal{R}_A$ and a type-N matrix $\mathcal{R}_N = \mathcal{R}_N^{f_{c_k}} \times \cdots \times \mathcal{R}_N^{f_{c_0}}$. Because of its structure, there is no coefficient in $\mathcal{R}_A$ that can cancel the nonzero coefficients of $\mathcal{R}_N$ in the addition $\mathcal{R}_A + \mathcal{R}_N$. Therefore, $\mathcal{R}$ is a type-A matrix if and only if $\mathcal{R}_N$ is null. □

The self-synchronization property in the spectral domain has an algebraic interpretation. It is based on the concept of semigroup. Notice first that the set of the $2^n \times 2^n$ dimensional correlation matrices together with the matrix multiplication is a semigroup.

Proposition 5.3.2 ([99]). *The system (5.1)–(5.2) is finite-time self-synchronizing if and only if the set of matrices $\{\mathcal{R}_N^{f_c},\, c \in \mathbb{F}_2^{n_s}\}$ generates a nilpotent semigroup.*

Proof. According to Remark 2.4.1, a system is finite-time self-synchronizing if and only if there is a positive integer k_c such that any sequence of length greater than k_c is synchronizing. That is, in view of Proposition 5.1.3, for $k \geq k_c$, any correlation matrix $\mathcal{R}^{f^{\to(c)_0^k}}$ is of type A. The expression of $\mathcal{R}^{f^{\to(c)_0^k}}$ given by (5.4) is, up to a constant factor, the product of k elements of the set $\{\mathcal{R}_N^{f_c}, c \in \mathbb{F}_2^{n_s}\}$. According to Proposition 5.3.1, this product is of type A if and only if whatever the sequence $(c)_0^k$ of symbols of $\mathbb{F}_2^{n_s}$ is, the product $\mathcal{R}_N^{f_{c_k}} \times \cdots \times \mathcal{R}_N^{f_{c_0}}$ is null. This is the case if and only if the set $\{\mathcal{R}_N^{f_c},\, c \in \mathbb{F}_2^{n_s}\}$ generate a nilpotent semigroup of nilpotency class at most k_c. □

Next proposition makes a classification of the possible situations that allow the system (5.1)–(5.2) to be finite-time self-synchronizing. It clearly gives a characterization of the functions that can be used in the design of finite-time SSSC.

Proposition 5.3.3 ([99]). *The system (5.1)–(5.2) with the next-state function f (and the associated Shannon decomposition f_c, $c \in \mathbb{F}_2^{n_s}$) is finite-time self-synchronizing if and only if the reduced correlation matrices $\mathcal{R}^{f_c{}^\bullet}, c \in \mathbb{F}_2^{n_s}$ are nilpotent and fulfill one of the following cases:*

Case 1 *All the matrices $\mathcal{R}^{f_c{}^\bullet}$, $c \in \mathbb{F}_2^{n_s}$ are lower triangular.*

Case 2 *All the matrices* $\mathcal{R}^{f_c{}^\bullet}$, $c \in \mathbb{F}_2^{n_s}$ *are not lower triangular but can be simultaneously triangularized by a change of basis whose matrix is the reduced correlation matrix* $\mathcal{R}^{p^\bullet}$ *of some* (n,n)*–function* p*. This matrix has to be invertible. In this situation, the following equalities hold* $\mathcal{R}^{p^\bullet}\mathcal{R}^{f_c{}^\bullet}(\mathcal{R}^{p^\bullet})^{-1} = \overline{\mathcal{R}}^{f_c{}^\bullet}$ *for,* $c \in \mathbb{F}_2^{n_s}$ *with* $\overline{\mathcal{R}}^{f_c{}^\bullet}$ *lower triangular matrices with zeros on the diagonal.*

Case 3 *All the matrices* $\mathcal{R}^{f_c{}^\bullet}$, $c \in \mathbb{F}_2^{n_s}$ *are not lower triangular. However, likewise in Case 2, they can be simultaneously triangularized. But, unlike Case 2,* $\mathcal{R}^{p^\bullet}$ *does not correspond to a correlation matrix.*

Proof. Proposition 5.3.2 states that the system (5.1)–(5.2) is finite-time self-synchronizing if and only if the set $\{\mathcal{R}_N^{f_c}, c \in \mathbb{F}_2^{n_s}\}$ generates a nilpotent semigroup. In view of Remark 5.3.1, the same holds for the set $\{\mathcal{R}^{f_c{}^\bullet}, c \in \mathbb{F}_2^{n_s}\}$. Then, in view of Theorem 3.2.2, the set $\{\mathcal{R}^{f_c{}^\bullet}, c \in \mathbb{F}_2^{n_s}\}$ can be triangularized. Cases 1, 2 and 3 are exclusive and describe all the possible situations. □

The following lemmas are required in order to give an interpretation of Proposition 5.3.3.

Lemma 5.3.1. *The reduction operation is compatible with the inversion: For any bijection* p *of* $\mathbb{F}_2^n$*, the relation* $(\mathcal{R}^{p^\bullet})^{-1} = \mathcal{R}^{p^{-1}{}^\bullet}$ *holds.*

Proof. It can be seen from (4.16) that $\mathcal{R}^{p^{-1}} = (\mathcal{R}^p)^{-1}$. Since p is an invertible transformation, according to the result given in [98], its correlation matrix is orthogonal. Finally,

$$(\mathcal{R}^{p^\bullet})^{-1} = {}^t(\mathcal{R}^{\bullet p}) = ({}^t\mathcal{R}^p)^\bullet = (\mathcal{R}^{-1p})^\bullet = \mathcal{R}^{p^{-1}{}^\bullet}$$

□

Lemma 5.3.2 ([106])**.** *Let* f *be a* (n)*–function. The function* f *depends only on the first* i*th variables* $(i \leq n)$ *if and only if*

$$\forall u,\ \mathrm{supp}(u) \notin \{1,\ldots,i\},\ \widehat{f_\chi}(u) = 0 \tag{5.18}$$

Proof. Let us express the Walsh transform of a (n)–function f that really depends only on the first ith variables. It can be expressed, for $u \in \mathbb{F}_2^i$ and $v \in \mathbb{F}_2^{n-i}$, as

$$\widehat{f_\chi}(u|v) = \sum_{y \in \mathbb{F}_2^{n-i}} (-1)^{v\cdot y} \sum_{x \in \mathbb{F}_2^i} (-1)^{f(x|0)\oplus u\cdot x}$$

This implies that $\widehat{f_\chi}(u|v) = 0$ if $v \neq 0$, which proves that if $\mathrm{supp}(v) \notin \{1,\ldots,i\}$, then $\widehat{f_\chi}(v) = 0$.
Conversely, for $x \in \mathbb{F}_2^i$ and $y \in \mathbb{F}_2^{n-i}$, one has $f_\chi(x|y) = \frac{1}{2^n}\widehat{\widehat{f_\chi}}(x|y) = \frac{1}{2^n}\sum_{u,v}\widehat{f_\chi}(u|v)(-1)^{x\cdot u\oplus v\cdot y}$. As it is assumed that, for $v \neq 0$, one has $\widehat{f_\chi}(u|v) = 0$, we deduce $f(x|y) = \frac{1}{2^n}\sum_u \widehat{f_\chi}(u|0)(-1)^{u\cdot x}$. It is clear that this expression does not depend on y which completes the proof. □

Case 1 corresponds to the case when f_c, $c \in \mathbb{F}_2^{n_s}$ are strict T–functions. This observation directly follows from Lemma 5.3.2. Therefore, Case 1 refers to functions which have been already proposed through the open literature.

Case 2 corresponds to the situation when f_c, $c \in \mathbb{F}_2^{n_s}$ are not strict T–functions but functions of the form $f_c = p \circ \overline{f}_c \circ p^{-1}$ where $\overline{f}_c$, $c \in \mathbb{F}_2^{n_s}$ are strict T–functions and p a bijection over $\mathbb{F}_2^n$. A consequence of Lemma 5.3.1 is that this case is nothing but Case 1 in which the functions f_c, $c \in \mathbb{F}_2^{n_s}$ have been conjugated by p, a bijection of $\mathbb{F}_2^n$. Thus, this case is equivalent to Case 1 up to an invertible transformation of the internal state.

Case 3 corresponds to self-synchronizing functions that are not based on strict T–functions. This case is the most interesting one insofar as it allows to identify new classes of self-synchronizing functions.

Remark 5.3.3. It is interesting to note that the synchronization delay k_f discussed in Definition 2.2.4 of Section 2.2.2 precisely corresponds to the nilpotency class of the semigroup generated by the set $\{\mathcal{R}^{f_c\bullet}, c \in \mathbb{F}_2^{n_s}\}$. Moreover, since Cases 1 and 2 are based on strict T–functions, the maximum nilpotency class is bounded by n. In Case 3, the maximum nilpotency class is the dimension of the matrices which is $2^n - 1$. Therefore, if a set of reduced correlation matrices $\{\mathcal{R}^{f_c\bullet}, c \in \mathbb{F}_2^{n_s}\}$ generates a nilpotent semigroup of nilpotency class greater than n, it necessarily corresponds to Case 3.

The problem of determining if a set of (n, n)–functions can be used to design finite-time self-synchronizing systems (5.1)–(5.2) amounts to checking whether their reduced correlation matrices generate a nilpotent semigroup. From Proposition 5.3.2, if this is the case, they can be simultaneously triangularized. A similar problem has already been addressed by Algorithm 1 in Section 3.2.4 of Chapter 3.

The eigenanalysis of the matrices $\mathcal{F}$, $\mathcal{R}$, $\mathcal{N}$ is relevant for the comprehension of the self-synchronization mechanism. A necessary and sufficient self-synchronizing condition for the 2^{n_s} (n, n)–functions f_c, $c \in \mathbb{F}_2^{n_s}$ to be used to design a self-synchronizing system is that their reduced correlation matrices (the first row and column are removed) are nilpotent and simultaneously triangularizable. Equivalently, due to Proposition 5.3.2, they have to generate a nilpotent semigroup. It applies to any matrix representation such that it:

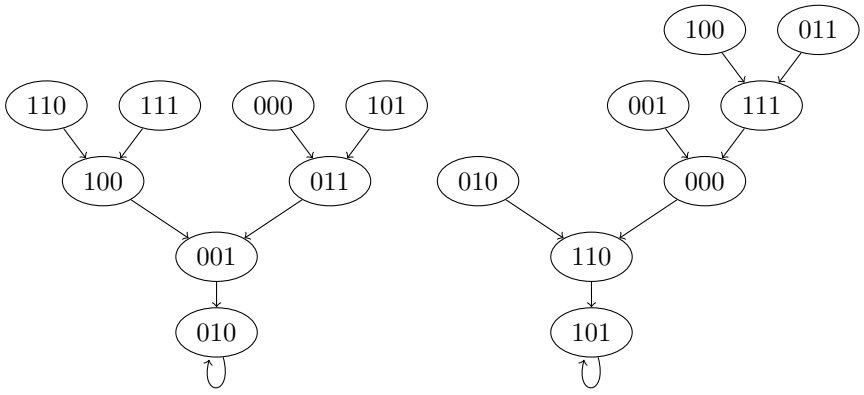
- has a similar composition property as in Proposition 4.2.1;
- represents constant elements (and only them) by matrices with null matrices except on the first row (or column).

Hence, the nilpotent semigroup criterion also applies to numerical matrices $\mathcal{N}$ and to algebraic matrices $\mathcal{A}$.

5.3.1.2 Example

Next, we give a next-state function that illustrates Case 3. Only one representation is provided here, the others are given in Appendix A.2. Consider the following next-state function with $n = 3$ and $n_s = 1$ whose Shannon decomposition is as follows.

$$f_0(x) = \begin{pmatrix} 1 \oplus x^0 \oplus x^1 \oplus x^0x^2 \\ 1 \oplus x^0x^1 \oplus x^2 \oplus x^0x^2 \\ x^1x^2 \end{pmatrix} \qquad f_1(x) = \begin{pmatrix} x^0x^1 \oplus x^2 \\ 1 \oplus x^0 \oplus x^0x^1 \oplus x^1x^2 \\ 1 \oplus x^0 \oplus x^0x^1 \oplus x^0x^2 \end{pmatrix}$$



(a) f_0 (b) f_1

Figure 5.3: Graph of the functions f_0 and f_1

The algebraic normal form of f is

$$f(c,x) = \begin{pmatrix} 1 \oplus x^0 \oplus x^1 \oplus x^0x^2 \oplus c \oplus x^0c \oplus x^1c \oplus x^0x^1c \oplus x^2c \oplus x^0x^2c \\ 1 \oplus x^0x^1 \oplus x^2 \oplus x^0x^2 \oplus x^0c \oplus x^2c \oplus x^0x^2c \oplus x^1x^2c \\ x^1x^2 \oplus c \oplus x^0c \oplus x^0x^1c \oplus x^0x^2c \oplus x^1x^2c \end{pmatrix}$$

The corresponding Walsh matrices are

$$\mathcal{W}^{f_0} = \begin{pmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4 & -4 & 0 & 0 & 4 & -4 \\ 0 & 0 & -4 & 4 & -4 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & -4 & 4 & 4 & 4 \\ 0 & 0 & 4 & 0 & 4 & 0 & -4 & 0 \\ -4 & 0 & 0 & -4 & 4 & 0 & 0 & -4 \\ -4 & 0 & 0 & 4 & 0 & -4 & -4 & 0 \\ -4 & 0 & 4 & 0 & 0 & 4 & 0 & 4 \end{pmatrix} \quad \mathcal{W}^{f_1} = \begin{pmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 4 & 4 & -4 \\ 0 & -4 & 0 & -4 & -4 & 0 & 4 & 0 \\ 0 & -4 & 0 & 4 & 0 & -4 & 0 & -4 \\ -4 & -4 & 0 & 0 & 0 & 0 & 4 & -4 \\ 4 & -4 & 0 & 0 & -4 & -4 & 0 & 0 \\ 4 & 0 & 0 & 4 & 4 & 0 & 0 & -4 \\ 4 & 0 & 0 & -4 & 0 & 4 & 4 & 0 \end{pmatrix}$$

According to Theorem 3.2.2, the matrices $\mathcal{W}^{f_0\bullet}$ and $\mathcal{W}^{f_1\bullet}$ generate a nilpotent semigroup. Indeed, they can be simultaneously triangularized and Algorithm 1 allows to find out one possible change of basis. The matrix

$$\mathcal{W}^{\bullet} = \begin{pmatrix} 1 & -1 & -2 & -2 & 4 & 0 & 0 \\ 1 & -1 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4 & -2 & 2 & 0 \\ 0 & 2 & 1 & -3 & 1 & -1 & -1 \\ -1 & -1 & 1 & -3 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -3 & -1 & 1 \\ 0 & 2 & 1 & 1 & 3 & 1 & 1 \end{pmatrix}$$

triangularizes both $\mathcal{W}^{f_0\bullet}$ and $\mathcal{W}^{f_1\bullet}$.

It can be checked that the class of nilpotency of this semigroup is six. From Remark 5.3.3, there is no bijection from $\mathbb{F}_2^3$ that triangularizes the semigroup generated by $\mathcal{W}^{f_0\bullet}$ and $\mathcal{W}^{f_1\bullet}$. Therefore, the function f corresponds to Case 3. Besides, this function is balanced. Hence, in view of Remark 6.1.4, each state is reached with the same probability.

It is interesting to observe that, according to Algorithm 1, a necessary condition for simultaneous triangularization to hold is that the matrices $\mathcal{W}^{f_0\bullet}$ and $\mathcal{W}^{f_1\bullet}$ have to share an eigenvector. It has to be related to the eigenanalysis of the correlation matrices discussed in Section 4.3 of Chapter 4. In particular, it is required that, one junction of the graph of f_0 and one junction of the graph f_1 share two identical vertices in their preimage set. The graph of these functions can be checked in Appendix A.2.

5.3.1.3 An algorithm to compute the synchronization delay

Besides the issue of determining whether a next-state function is self-synchronizing, it can be interesting to find out the value of the synchronization delay k_f. A first remark worth to be mentioned is the following.

Remark 5.3.4. The nilpotency class of a semigroup can be larger than the nilpotency class of its generators. For instance, consider the semigroup generated by A function that illustrates this remark can be found in Appendix A.4.

A methodology has to be set up to determine the exact value of k_f. The straightforward approach is to test all the sequences of length k for any state. The smallest k such that all the sequences are synchronizing is the synchronization delay. A more efficient approach due to Éric Garrido is proposed in Algorithm 2.

The idea of the algorithm is the following. Two identical systems initialized in two different states are considered. The evolution of the two systems is tested for each possible pair of different initial states that

input : the $(n_s + n, n)$–function f to test
output: the synchronization delay if finite

delay $\longleftarrow 0$;
$P(0) \longleftarrow \{\{a, b\},\ a, b \in \mathbb{F}_2^n,\ a \neq b\}$;
while $P(\text{delay}) \neq \emptyset$ **do**
  delay $\longleftarrow$ delay $+ 1$;
  $P(\text{delay}) \longleftarrow \emptyset$;
  foreach $c \in \mathbb{F}_2^{n_s}$ **do**
    $P(\text{delay}) \longleftarrow P(\text{delay}) \cup \{\{f_c(a), f_c(b)\}, \{a, b\} \in P(\text{delay} - 1),\ f_c(a) \neq f_c(b)\}$;
  end
end
return delay;

Algorithm 2: Algorithm that determines the synchronization delay of a self-synchronizing system

is, for all the elements of the set $P(0)$. At each step k, the set $P(k)$ contains all the pairs of states to be tested. The set $P(1)$ is obtained by applying all the functions corresponding to Shannon decomposition f_c, $c \in \mathbb{F}_2^{n_s}$ to each pair $\{a, b\} \in P(0)$. If for all $c \in \mathbb{F}_2^{n_s}$, the function f_c is applied to a and b and does not produce the same value then, the pair $\{a, b\}$ is added to $P(1)$. The pair is added since it means that if the state of two systems are set respectively to a and b, there exists at least one symbol $c \in \mathbb{F}_2^{n_s}$ such that the next-function f leads to different states after one iteration. As a result, the two systems are not synchronized. The set $P(k+1)$ is obtained similarly from the set $P(k)$. When the set $P(k)$ is empty, it means that, after k iterations, there is no pair of states that leads to different values when applying the next-state function f. The synchronization delay is the smallest k such that the set $P(k)$ is empty. Note that, if the function is not self-synchronizing, the algorithm does not stop. However, due Remark 5.3.3, the algorithm can be stopped if the variable *delay* is larger than $2^n - 1$ and it can be concluded that the function is not self-synchronizing.

5.3.2 Graph-based approach

5.3.2.1 Tree functions based approach

Next, we show that the functions f_c, $c \in \mathbb{F}_2^{n_s}$, corresponding to Shannon decomposition of a self-synchronizing function f, have a specific graph structure that we call a *tree structure*.

Definition 5.3.1 (Directed tree)**.** A directed tree is a directed graph with a single weakly connected component and whose unique cycle has a length equal to one. The vertex associated to the cycle of length one is called the *root* of the graph.

Definition 5.3.2 (Tree-function)**.** A tree function is a (n, n)–function whose graph is a directed tree.

Examples of such graphs are given in Figure 5.3.

The nilpotent requirement of the reduced correlation matrices implies that, all the eigenvalues of the corresponding correlation matrices $\mathcal{R}^{f_c}$, $c \in \mathbb{F}_2^{n_s}$ are null (except one which is always equal to one). From Proposition 4.3.3 of Section 4.3, there is exactly one cycle of length one in the graph. We recall that, the graph of an (n, n)–function is composed of as many weakly connected components as cycles. As a consequence, the (n, n)-functions with nilpotent reduced correlation matrices $\mathcal{R}^\bullet$ precisely correspond to Definition 5.3.2.
The number of (n, n)–function is $(2^{2^n})^n$. The number of tree-functions is given by the following proposition.

Proposition 5.3.4. *There are* $(2^{2^n-1})^n$ (n, n)*–tree-functions.*

Proof. Cayley's formula [107] states that, there are a^{a-2} ways to connect a set of a labelled vertices in a tree structure. However, the trees considered in this formula do not match Definition 5.3.1. The trees considered in [107] are undirected graphs whereas we consider directed graphs. Besides [107] considers $a-1$ connection between the vertices while we have a of them. These two differences have to be taken into account to extend Cayley's formula to the number of trees that matters to us. Any labelled tree considered in [107] can be modified by choosing one of its vertices to be the origin of this extra arc. Let v be this vertex, the end of the extra arc is necessarily the vertex v otherwise it introduces a cycle of length larger than one and the graph does not match Definition 5.3.1 any more. Let us note that, for the modified graph to correspond to a function, once the root v is selected, there is no choice in the orientation of the arcs of the graph. Therefore, adding a single arc to the trees of [107] suffices to match Definition 5.3.1. Hence, there are a possibilities to choose the vertex v and for each of the a^{a-2} graphs considered in [107], it can be derived a distinct tree that match Definition 5.3.1. As a result, there are a^{a-1} trees such as defined by Definition 5.3.1. When trees corresponding to (n,n)–function are considered, the number of vertices is $a=2^n$ hence, there are $(2^{2^n-1})^n$ of them. □

As a consequence, as n increases, the ratio of the number of tree-functions over all the possible functions is 2^{-n} and thus rapidly decreases which prevents any attempts at finding them by a blind search.

Remark 5.3.5. Tree-functions cannot be balanced. The only balanced (n,n)–functions are the invertible functions. By definition, a tree function is not invertible.

5.3.2.2 Connection graph based-approach

The book [32] describes an interesting framework to deal with discrete dynamical systems. It does not consider any synchronization issue but this framework can be used for that purpose. However, it does not allow to deal with other functions than T–functions as shown below. The interest of the approach described hereafter is that it considers square matrices of dimension n. Therefore, the problem is much more tractable than when dealing with matrices of dimension 2^n. In [32], functions are characterized by their connection graph, or equivalently by the adjacency matrix of the connection graph $\mathscr{C}$.

Definition 5.3.3 (Connection graph)**.** Let f be a (n,n)–function of the variable x. Its connection graph is the graph with n vertices labelled according to the indices of x. The element (i,j) is an arc of the graph if and only if the ith coordinate function depends on the variable x^j.

For example, the connection graph of f_{e2} (defined in Section 4.2) is given in Figure 5.4 and the corresponding adjacency matrix is

$$\mathscr{C}^{f_{e2}} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

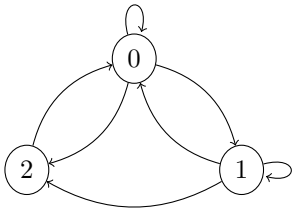


Figure 5.4: Connection graph of f_{e2}

Proposition 5.3.5 ([32])**.** *Let f,g be two (n,n)–functions then*

$$\mathscr{C}^{f\circ g} \leq \mathscr{C}^{f}\mathscr{C}^{g} \tag{5.19}$$

The matrix multiplication involved in the inequality is not the usual one. It is still based on multiplications and additions of the entries of the matrices. However, the addition operation is replaced by the inclusive-or. Here, the symbol $\leq$ means that the left matrix is componentwise lower or equal to the right matrix.

The following proposition can now be stated.

Proposition 5.3.6. *Let f be a $(n+n_s,n)$–function and f_c, $c \in \mathbb{F}_2^{n_s}$ the corresponding Shannon decomposition. If there exists an integer $k_f \in \mathbb{N}$ such that $\forall\,(c)_0^{k_f-1}$ of element of $\mathbb{F}_2^{n_s}$,*

$$\mathscr{C}^{f_{c_{k_f-1}}} \times \cdots \times \mathscr{C}^{f_{c_0}} = \mathbf{0}$$

then, the function f is self-synchronizing.

Proof. A function $f(x)$ is constant if each coordinate function does not depend on any coordinate of x hence, if its connection matrix is null. Considering Proposition 5.3.5, the result holds. □

Note that, this is only a sufficient condition, only the functions of Case 1 and some functions of Case 2 in Proposition 5.3.3 can be identified.

Corollary 5.3.1. *If the set of matrices $\{\mathscr{C}^{f_c},\, c \in \mathbb{F}_2^{n_s}\}$ generates a nilpotent semigroup, the function f is self-synchronizing.*

Proof. This is a direct consequence of the nilpotent semigroup definition. □

Chapter 6

Security issues and construction elements of SSSC

In this chapter, we do not intend to deliver a fully specified SSSC but rather to discuss security concepts and implementations aspects. Security is the purpose of Section 6.1. Section 6.2 presents some construction elements. Section 6.3 is devoted to the test platform that we have developed.

6.1 Security issues

6.1.1 Basics on security

As explained in the introduction, the purpose of confidentiality is to conceal the meaning of a message. Regarding this aspect, the notion of information must be formalized. In 1948, the paper [40] by Shannon provided a complete theoretical framework to study communications. One year later, in 1949, he published a report [7] devoted to secrecy systems. The basic idea associated to the notion of information such as defined by Shannon is the following. The less likely a message, the more information it provides. Also, the information provided by two independent messages has to be the sum of each information. Furthermore, if the additional constraint that the information is a positive quantity is added then, information should be defined as follows.

Definition 6.1.1 (Information of an event)**.** Let Ω be a probability space. The information provided by the event $\omega \in \Omega$ is the random variable $\mathcal{I} : \Omega \longrightarrow \mathbb{R}$ defined by

$$\mathcal{I}(\omega) = -\log_2(\Pr(\omega)) \tag{6.1}$$

Another important notion in information theory is the *entropy*. It quantifies the uncertainty of a message.

Definition 6.1.2 (Shannon entropy of a random variable)**.** Let $X : \Omega \longrightarrow \mathcal{X}$ be a random variable, its Shannon entropy is defined by

$$\mathcal{H}(X) = \sum_{x \in \mathcal{X}} -\Pr(x)\log_2(\Pr(x)) \tag{6.2}$$

Note that, other definitions of entropy exist such as Renyi entropy. However, for our purpose, we only need Shannon entropy.

Conditional probabilities naturally lead to the notion of conditional entropies. It corresponds to the uncertainty unveiled on a random variable when the occurrence of a specific event is known. Let $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ be two events, we denote by $\Pr(x|y)$ the probability of the event x based on the knowledge that the event y has occurred.

Definition 6.1.3 (Conditional entropy)**.** Let $X : \Omega \longrightarrow \mathcal{X}$, $Y : \Omega \longrightarrow \mathcal{Y}$ be two random variables. The conditional entropy of X knowing that the event $y \in \mathcal{Y}$ has occurred is

$$\mathcal{H}(X|y) = -\sum_{x \in \mathcal{X}} \Pr(x|y) \log_2 \Pr(x|y) \tag{6.3}$$

When the event y is replaced by a random variable, the following definition has to be considered.

Definition 6.1.4 (Conditional entropy)**.** Let $X : \Omega \longrightarrow \mathcal{X}$, $Y : \Omega \longrightarrow \mathcal{Y}$ be two random variables. The conditional entropy of X knowing Y is

$$\mathcal{H}(X|Y) = -\sum_{y \in \mathcal{Y}} \Pr(y) \sum_{x \in \mathcal{X}} \Pr(x|y) \log_2 \Pr(x|y) \tag{6.4}$$

The information theoretic background being recalled, security notions can be presented.

Definition 6.1.5 (Unconditional security)**.** A system is called unconditionally secure if the adversary has no better strategy than choosing the plaintext at random to guess the meaning of the cryptogram.

In other words, the ciphertext does not provide any information about the plaintext. Let M be a random variable corresponding to a plaintext and C the random variable corresponding to its ciphertext. In terms of entropy, unconditional security means that $\mathcal{H}(M|C) = \mathcal{H}(M)$. A system that achieves unconditional security is called a *perfect system*. The most well known perfect system is the Vernam cipher discussed in Section 2.3.2.2. Next remark explains why the choice of the exclusive-or operation $\oplus$ for combining the plaintext with a random symbol is relevant.

Remark 6.1.1. Let $m \in \mathbb{F}_2^{n_s}$ be a message to be concealed, $z \in \mathbb{F}_2^{n_s}$ be a symbol drawn according to the uniform probability. We show that, the cryptogram $c \in \mathbb{F}_2^{n_s}$ obtained by the transformation $c = m \oplus z$, does not carry any information about m. Let have $n_s = 1$. Let M be a random variable corresponding to a message and C the random variable of the corresponding cryptogram. We want to show that $\mathcal{H}(M|C) = \mathcal{H}(M)$. Having in mind that whatever m is and whatever z is, the probability of occurrence of a specific symbol for c is $\frac{1}{2}$. Let us derive $\mathcal{H}(M)$

$$\mathcal{H}(M) = -\sum_{m \in \mathbb{F}_2} \Pr(m) \log_2 \Pr(m) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$$

And then,

$$\mathcal{H}(M|C) = -\sum_{m \in \mathbb{F}_2} \sum_{c \in \mathbb{F}_2} \Pr(c, m) \log_2 \left(\frac{\Pr(c)}{\Pr(m, c)}\right) \tag{6.5}$$

$$= -4 \times \frac{1}{2} \times \frac{1}{2} \log_2 \left(\frac{\frac{1}{2}}{\frac{1}{2} \times \frac{1}{2}}\right) \tag{6.6}$$

$$= 1 \tag{6.7}$$

Hence, the knowledge of c does not unveil any uncertainty about m if z is random. When $n_s > 1$, and that the symbol z is drawn according to the uniform distribution, the bits of z are independent and the same reasoning can be applied independently on each bit of m.

Hence, if the information of the key is smaller than the information of the message, unconditional security cannot be achieved. In particular, it is the case when the number of bits of the key is smaller than the number of bits of the message. As a matter of fact, the adversary can always try to guess the key at random which is a better strategy than guessing the message at random since there are less possibilities. As explained in Section 2.3.2.2, having a key whose length is the same than the length of the message is not convenient. Since the key is, in practice, shorter than the message to be concealed, unconditional security cannot be achieved. However, security notions need to be defined to evaluate systems. Since the security cannot be unconditional, it must be computational. That is, it relies on the fact that an adversary has only a limited amount of resources and cannot perform any computation he desires. Hence,

computational security depends on the technology. The security is assessed with respect to a *security parameter* denoted by n_κ. It usually corresponds to the number of bits of the key. The probability to properly recover a message is a decreasing function of n_κ.

Definition 6.1.6 (Negligible quantity)**.** A quantity is said to be negligible if it decreases faster than any polynomial in $\frac{1}{n_\kappa}$.

Definition 6.1.7 (Advantage)**.** Let p be the probability to recover some information. The advantage associated to this probability is defined by $2p - 1$.

This definition allows to define semantic security as follows.

Definition 6.1.8 (Semantic security)**.** A cipher is called semantically secure if the adversary has a negligible advantage to recover any bit of plaintext based on the knowledge of the ciphertext.

In other words, a cipher is semantically secure if an adversary has not a probability significantly larger than one half of recovering any bit of plaintext. An important property for a cipher is the following.

Definition 6.1.9 (Indistinguishability property)**.** A cipher is said to have the indistinguishability property if, when an adversary chooses two plaintexts m, m' and gets the ciphertext c corresponding to one of these messages, he is not able to figure out with a probability significantly larger than one half to which message the ciphertext corresponds.

The scenario associated to this property is usually presented in the form of a game. An adversary (the player) chooses two plaintexts m, m' and is given the ciphertext c of one of the plaintexts. He has to decide whether c is the ciphertext corresponding to m or to m'. The triplet (m, m', c) is called the *challenge*. It was shown in [108] that a cipher is semantically secure if and only if it has the indistinguishably property.

Different models of adversaries can be considered. For instance, the adversary can be helped by an *oracle*. The oracle can encrypt any plaintext (except m and m') that the adversary wishes. The oracle provides an additional source of information to the adversary and allows to define different security levels. If the scenario does not allow the adversary to win the game with a significant advantage, the algorithm is said to be resistant against chosen plaintext attacks (CPA for short). The adversary can also be given the access to an oracle that is able to decrypt any cryptogram (except c) of his choice. If the scenario does not allow the adversary to win with a significant advantage, the algorithm is said to be resistant against chosen ciphertext attacks (CCA for short). Two kinds of CCA can be distinguished depending on when the oracle is accessible. When the adversary can only access the oracle before being aware of the challenge, the scenario is called CCA1. When the adversary can access the oracle both before and after being aware of the challenge, the scenario is called CCA2.

When considering a cipher, it is important to have in mind the possible relevant attack scenarios. In [55], it is claimed that chosen ciphertext attacks were not to be considered in their algorithm. However, the authors of [57] have pointed out the problem of such a statement. Besides, history has shown that not considering CPA is devastating because most of the SSSC, if not all, have been broken with a CPA.

When a SSSC is to be used, three steps are required to properly transmit the information. First, the encryptor needs to be initialized with a random initial state. Then, it needs to make sure that the receiver is synchronized by encrypting a random sequence whose length is equal to the synchronization delay. Finally, it can process the plaintext. With this procedure in mind, the following result proposed by Philippe Guillot and Hieu Phan can be stated.

Proposition 6.1.1. *A self-synchronizing stream cipher is not CCA2-indistinguishable.*

Proof. Consider the following scenario.

1. The adversary chooses two messages m and m';

2. The oracle chooses a random initial state;
3. The oracle chooses a random plaintext sequence whose length k_f is the synchronization delay. He then prepends this sequence to the plaintext. The purpose of the prepended sequence is to synchronize the decryptor with the encryptor;
4. The oracle transmits the encrypted sequence (c).

The adversary can ask the oracle to decrypt the cryptogram for which the last symbol is modified. The resulting cryptogram is almost equal to one of the messages m or m' and the adversary can distinguish to which plaintext c corresponds. □

In order to prove the security of a system, it is important to define an upper bound on what can be done. This leads to the definition of an ideal SSSC which is as follows.

Definition 6.1.10 (Ideal SSSC)**.** An SSSC is ideal if, for any secret parameter κ, when the canonical form (2.19) is considered, the output function cannot be distinguished from a uniformly distributed random Boolean function.

Remark 6.1.2. If the function e is chosen to be the $\oplus$ operation then, according to Remark 6.1.1, an ideal SSSC is obtained if the function $h[\kappa]$ of the canonical form cannot be distinguished from a uniformly random Boolean function.

6.1.2 Cryptanalysis elements

Even though SSSC belong to the class of stream ciphers, from the cryptanalysis point of view, it is interesting to consider them as block ciphers. As a matter of fact, every finite-time SSSC admits a canonical representation. The canonical representation can be considered as a block cipher $B[\kappa] : \mathbb{F}_2^{k_f n_s} \longrightarrow \mathbb{F}_2^{n_s}$ where k_f is the synchronization delay. One of the main differences with block ciphers is they usually have as many output bits as input bits. Besides, one iteration of the generalized form can be considered as one round transformation of a block cipher. Hence, some of the cryptanalysis techniques that apply to block ciphers can be extended to SSSC.

A major difference between synchronous stream ciphers and SSSC is that with the former, the dynamics is autonomous while for the latter it is not. Hence, for SSSC, chosen messages can be injected in the dynamics in order to reveal information. This is the reason of the failure of many SSSC.
Differential attacks were initially developed in [109] to attack block ciphers. However, they can be extended to SSSC. They are applicable when in average, a specific input difference leads to a specific output difference more often than expected. Differential attacks were theoretically discussed in the case of SSSC in [50, 110] and successfully applied to break KNOT [51] or HBB [53].
Linear attacks were introduced in [111] and were successfully applied to block ciphers. The general idea is to approximate the canonical form by a linear relation involving input and output bits and some key bits. In order for the system to be resistant against such an attack, the output function of the canonical form must not be too close from a function with a low algebraic degree. The paper [50] outlines the possibility of applying linear attacks to SSSC.

6.1.3 State probability and security

The purpose of this section is to derive probability laws regarding the state of Boolean dynamical systems. The characterization of the probability law is interesting for security purposes. To this end, let us analyze the evolution of the probability law when the next-state function f is iterated.

Proposition 6.1.2. *Let $(C)_0^k$ be a uniform random sequence and assume a uniform random distribution of the initial state X_0. Then, the probability that the iterated function $f^{\to(C)_0^k}$ returns the state $x \in \mathbb{F}_2^n$ is*

$$P[f^{\to(C)_0^k}(X_0) = x] = \frac{1}{2^{n+k+1}} \sum_{u \in \mathbb{F}_2^n} (-1)^{u \cdot x} \left(\left(\sum_{c \in \mathbb{F}_2^{n_s}} \mathcal{R}^{f_c} \right)^{k+1} \right)_{u,0} \tag{6.8}$$

Proof. The probability of a specific sequence of length $k+1$ to be drawn is 2^{-k-1}.

$$P[f^{\to (C)_0^k}(X_0) = x] = \frac{1}{2^{k+1}} \sum_{c \in \mathbb{F}_2^{k+1}} P[f_{c_k} \circ \cdots \circ f_{c_0}(X_0) = x]$$

Then, in view of Proposition 5.3.5 and Corollary 4.2.2

$$\begin{aligned} P[f^{\to (C)_0^k}(X_0) = x] &= \frac{1}{2^{k+1}} \sum_{(c)_0^{k+1} \in (\mathbb{F}_2^{n_s})^{k+1}} \frac{1}{2^n} \sum_{u \in \mathbb{F}_2^n} (-1)^{u \cdot x} \left(\mathcal{R}^{f_{c_k}} \times \cdots \times \mathcal{R}^{f_{c_0}} \right)_{u,0} \\ &= \frac{1}{2^{n+k+1}} \sum_{u \in \mathbb{F}_2^n} (-1)^{u \cdot x} \left(\left(\sum_{c \in \mathbb{F}_2^{n_s}} \mathcal{R}^{f_c} \right)^{k+1} \right)_{u,0} \end{aligned}$$

□

Corollary 6.1.1. *Assuming a random sequence $(C)_0^k$ and a random initial state X_0, the system* (2.22a)–(2.22b) *has an equal probability to be in each state at time $k+1$ if and only if the first column of the matrix*

$$\left(\sum_{c \in \mathbb{F}_2^{n_s}} \mathcal{R}^{f_c} \right)^{k+1}$$

denoted by a is given by

$$a = {}^t\begin{pmatrix} 2^{n_s+k+1} & 0 & \cdots & 0 \end{pmatrix} \tag{6.9}$$

Proof. Proving this result amounts to solving a linear algebra problem. Let ν be the 2^n–dimensional column vector whose coefficients at row x is the probability of being in the state x. In our case, we set the value of each coefficient to 2^{-n}. Considering Proposition 6.1.2, denoting by H the 2^n–dimensional Hadamard matrix defined by $H = (h_{u,x}) = (-1)^{u \cdot x}$ for $u, x \in \mathbb{F}_2^n$ and the constant γ by 2^{-n-k-1}, the problem reads

$$\nu = \gamma H a$$

where a is the unknown. Since both k and H are invertible, the system can be solved and has a unique solution. □

Remark 6.1.3. The fact that each state is reached with an equal probability for a random sequence of length k does not mean that each state is reached with an equal probability with a uniform random sequence of length $k+1$.

Remark 6.1.4. Corollary 6.1.1 states that, assuming a uniform distribution of the initial state X_0 and a uniform random sequence (C), the uniform distribution of the internal state at time k is achieved if and only if the first column vector of

$$\left(\sum_{c \in \mathbb{F}_2^{n_s}} \mathcal{R}^{f_c} \right)^{k+1}$$

is given by the relation (6.9). Under this condition, a uniform distribution is achieved at any time if and only if f is balanced.

Note that the first column of $\sum_{c \in \mathbb{F}_2^{n_s}} \mathcal{R}^{f_c}$ is the same as the first column of $\mathcal{R}^f$. Hence, according to Proposition 4.2.2, f is balanced. In particular, when $n_s = 1$, the following equality holds

$$\forall u \in \mathbb{F}_2^{n*},\ \mathcal{R}^{f_0}_{u,0} = -\mathcal{R}^{f_1}_{u,0} \qquad (\text{or } \mathcal{W}^{f_0}_{u,0} = -\mathcal{W}^{f_1}_{u,0})$$

An important security property is that, all the states are reached with the same probability. Indeed, the strict application of this principle is not mandatory. But, in practice the system should have almost the same probability to be in any state in order to avoid biases and to introduce thereby any weakness.

Proposition 6.1.3. *The $(n+n_s, n)$–function f is balanced if and only if all the nodes of the graph of the function have the same in-degree.*

Proof. This statement is a direct consequence of Definition 4.2.1. □

6.2 New self-synchronizing constructions

This section proposes, both primary and secondary, new self-synchronizing constructions. We recall that primary constructions are atomic self-synchronizing constructions while secondary constructions are those based on primary constructions assembly. The constructions presented in this section are either completely new or inspired of the ones of the literature and recalled in Section 2.4.3 of Chapter 2.

6.2.1 Permutation-based constructions

In this section, we define constructions based on the notion of equivalence between several functions. The idea is to identify all the functions that behave similarly in some sense defined later and the transformation that preserve the self-synchronization property. Let us formalize the self-synchronization concept in terms of graph. First of all, we need to define the graph of a $(n+n_s, n)$–function. They are different from the graph introduced earlier in Section 4.2.3 in Chapter 4 as they were related to (n, n)–functions.

Definition 6.2.1 (Graph of a $(n+n_s, n)$–function)**.** Let $f(c, x)$ be a $(n+n_s, n)$–function and f_c, $c \in \mathbb{F}_2^{n_s}$ be the corresponding Shannon decomposition. Its graph is the pair $\mathcal{G} = (V, E)$ where V is the set of labelled vertices and E is the set of labelled arcs. An arc is an ordered pair of vertices denoted by (x, y). The pair (x, y) is an element of E if and only if there exists $c \in \mathbb{F}_2^{n_s}$ such that $y = f_c(x)$. The pair (x, y) is labelled by c. Hence, the out-degree of each vertex is n_s.

For example, the graph of the shift function $\mathfrak{s}_{3,1}$ such as defined by (2.20) in Chapter 2 is given in Figure 6.1. This graph is called the De Bruijn graph.

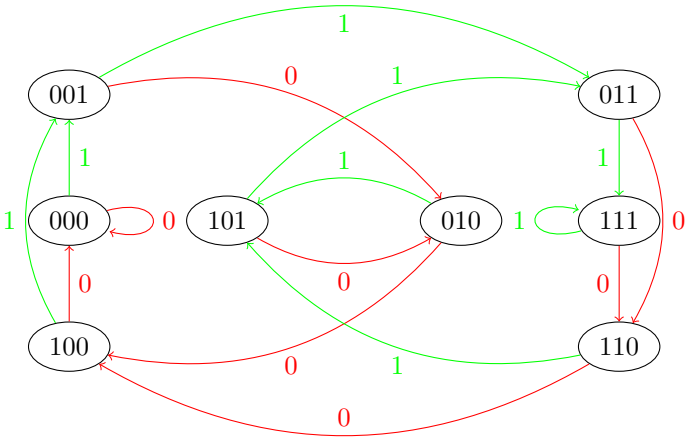


Figure 6.1: Graph of the shift function $\mathfrak{s}_{3,1}$.

Proposition 6.2.1. *Permuting the labels of the vertices of the graph of an $(n+n_s, n)$–function does not change the self-synchronizing property.*

Proof. The graph corresponds to a self-synchronizing function if there is an integer $k_f \in \mathbb{N}^*$ such that for any vertex $x_0 \in \mathbb{F}_2^n$, any path of length k_f yields to a vertex that does not depend on x_0. Since it holds for any vertex, relabelling the vertex does not change the self-synchronizing property. □

Definition 6.2.2 (Graph isomorphism)**.** The graphs $\mathcal{G}$ and $\mathcal{G}'$ are said to be isomorphic if they are identical up to a relabeling of the vertices.

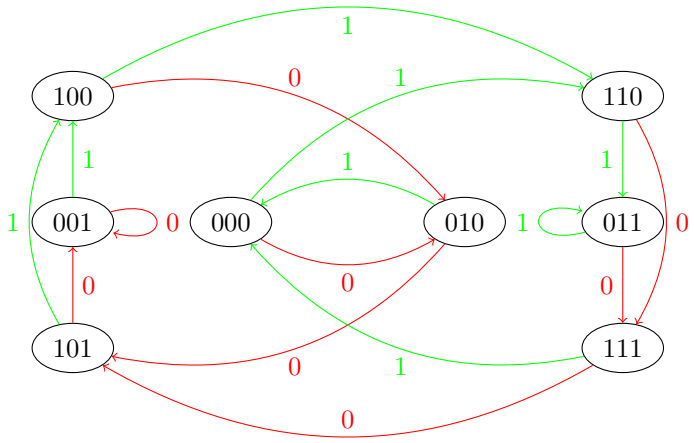


Figure 6.2: Graph isomorphic to the graph of Figure 6.1

Definition 6.2.3 (Isomorphic next-state functions)**.** Let f be a $(n+n_s, n)$–function (respectively f') with graph $\mathcal{G}$ (respectively $\mathcal{G}'$). The functions f and f' are said to be isomorphic if their graphs $\mathcal{G}$ and $\mathcal{G}'$ are isomorphic.

An equivalent definition is the following.

Definition 6.2.4 (Isomorphic next-state functions)**.** Let f (respectively f') be a $(n+n_s, n)$–function whose Shannon decomposition is $\{f_c,\, c \in \mathbb{F}_2^{n_s}\}$ (respectively $\{f'_c,\, c \in \mathbb{F}_2^{n_s}\}$). The functions f and f' are said to be isomorphic if there exits a permutation $p : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ such that $\forall x \in \mathbb{F}_2^n$, $f'_c(x) = p^{-1} \circ f_c(x) \circ p$. The function p defines the relabelling of the vertices.

Hence, the graphs of the $(n+n_s, n)$ functions in Figure 6.1 and in Figure 6.2 are isomorphic.

Remark 6.2.1. Note that, by definition, each function corresponding to Case 2 in Proposition 5.3.3 is isomorphic to a function of Case 1.

Let f and f' be two isomorphic $(n+n_s, n)$–functions and p a permutation of $\mathbb{F}_2^n$ that transforms one function into the other. The self-synchronizing system based on f' can be written as follows.

$$\mathcal{E} \begin{cases} x'_{k+1} & = p^{-1} \circ f(c_k, p(x'_k)) \\ z'_k & = h(x'_k) \\ c_k & = m_k \oplus z'_k \end{cases} \tag{6.10}$$

$$\mathcal{D} \begin{cases} \hat{x}'_{k+1} & = p^{-1} \circ f(c_k, p(\hat{x}'_k)) \\ \hat{z}'_k & = h(\hat{x}'_k) \\ \hat{m}_k & = c_k \oplus \hat{z}'_k \end{cases} \tag{6.11}$$

An equivalent form, in the sense that the same input sequence leads to the same output sequence is the following.

$$\mathcal{E} \begin{cases} x_{k+1} & = f(c_k, x_k) \\ z_k & = h \circ p(x_k) \\ c_k & = m_k \oplus z_k \end{cases} \tag{6.12}$$

$$\mathcal{D}\begin{cases} \hat{x}_{k+1} & = f(c_k, \hat{x}_k) \\ \hat{z}_k & = h \circ p(\hat{x}_k) \\ \hat{m}_k & = c_k \oplus \hat{z}_k \end{cases} \tag{6.13}$$

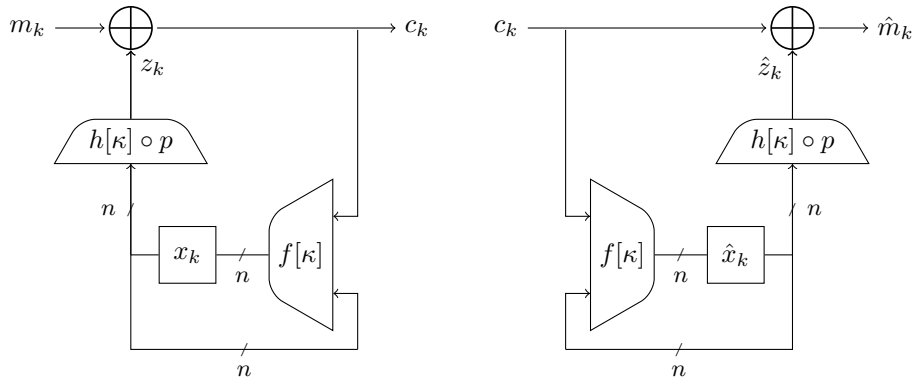


Figure 6.3: Equivalent composition construction

In the same way, another self-synchronizing construction can be derived from f with the help of q, permutation of $\mathbb{F}_2^{n_s}$. Consider the $(n+n_s, n)$–function f'' defined by $f''(c,x) = f(q(c),x)$. It is self-synchronizing according to Proposition 5.3.2. The corresponding system can be written as follows.

$$\mathcal{E}\begin{cases} x''_{k+1} & = f(q(c_k), x''_k) \\ z''_k & = h(x''_k) \\ c_k & = m_k \oplus z''_k \end{cases} \tag{6.14}$$

$$\mathcal{D}\begin{cases} \hat{x}''_{k+1} & = f(q(c_k), \hat{x}''_k) \\ \hat{z}''_k & = h(\hat{x}''_k) \\ \hat{m}_k & = c_k \oplus \hat{z}_k \end{cases} \tag{6.15}$$

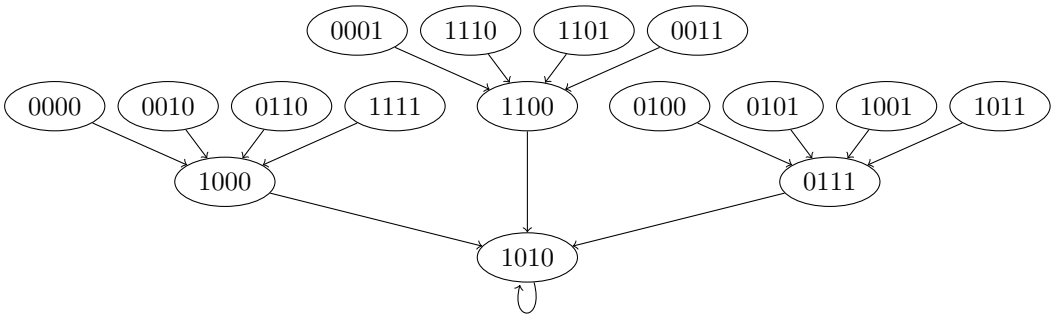
Note that the function f is not necessarily isomorphic to f''.

6.2.2 Shift isomorphism-based constructions

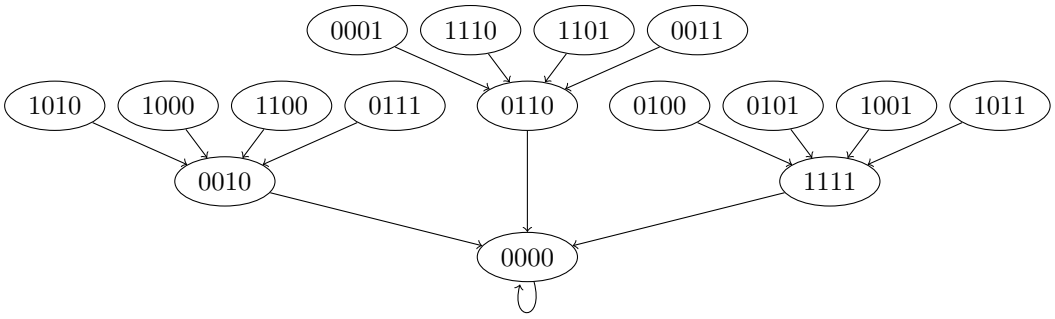
The construction presented here can be considered both as a primary and secondary construction. It is primary in the sense that it is explained how to obtain the construction from scratch without referring to any self-synchronizing construction. However, it can also be considered as secondary because it can be derived from the shift function.

It can be checked that the following construction is isomorphic to the shift function $\mathfrak{s}_{k_f,n_s}$:

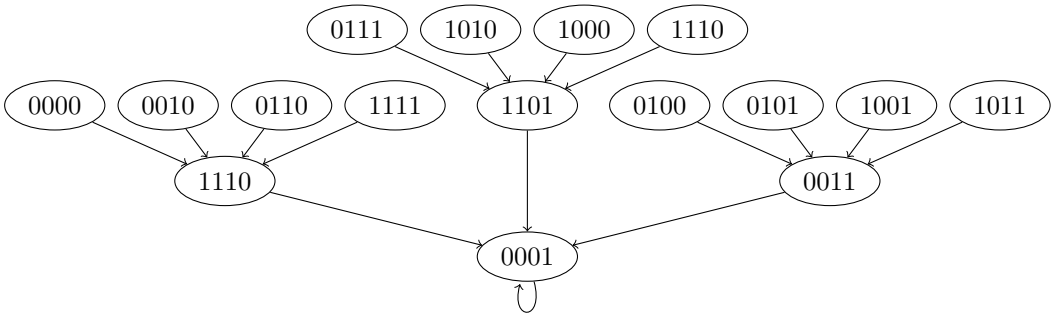
1. Select the number of bits per symbol n_s and the dimension of the system $n = k_f n_s$, $k_f \in \mathbb{N}^*$;
2. Create 2^{n_s} unlabelled balanced trees with 2^n vertices and depth k_f;
3. Label the vertices of each tree with the elements of $\mathbb{F}_2^n$ so that the preimage sets of the junctions are identical for all trees and so that the junctions of one tree are leaves in the others. Also, ensure that the preimage sets of the junctions of each graph is the same. Then, define 2^{n_s} (n,n)–functions f_c, $c \in \mathbb{F}_2^{n_s}$ so that each of them corresponds to one of these 2^{n_s} graphs;
4. Build the $(n+n_s, n)$–function f such that its Shannon decomposition is f_c, $c \in \mathbb{F}_2^{n_s}$.



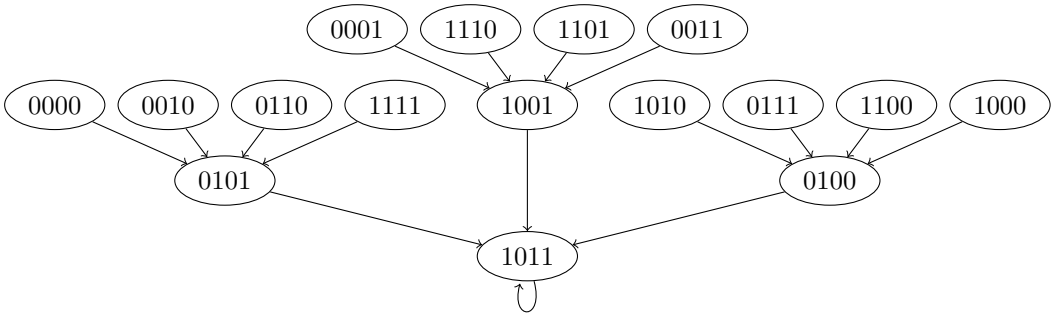
(a) f_{00}



(b) f_{01}



(c) f_{10}



(d) f_{11}

Figure 6.4: Balanced tree construction

The function f is self-synchronizing with synchronization delay k_f. An example of this construction is given in Figure 6.4 for $n_s = 2$ and $k_f = 2$. Of course, the dimension of the system is $n = n_s k_f = 4$.

Based on Section 6.2.1, we can see that such a shift isomorphic construction is an ideal cipher if the permutation p cannot be distinguished from a random one.

We stress that, not any self-synchronizing function is isomorphic to a shift. The example given at the end of Section 5.3 and developed in Appendix A.2 illustrates this statement. Clearly the graph of this function has not the same aspect that the graph of the shift.
Also, not any self-synchronizing function based on tree-like graphs is self-synchronizing. The additional constraint concerning the preimage sets of junctions added in Step 3 is essential. The $(4, 3)$–function provided in Appendix A.4 is an example. Shannon decomposition provides two functions with a tree-like structure. However, it can be seen that the preimage sets of the junctions of one graph are not the same as the preimage sets of the junctions of the other graph. The function is not shift-isomorphic. Besides, the self-synchronization delay is larger than the dimension of the system which is another proof that this function is not shift isomorphic.

6.2.3 Switched constructions

The following construction, directly based on Proposition 5.3.2 of Chapter 5, uses nilpotent semigroups to construct a self-synchronization function. The basic idea is to find a set of 2^{n_s} (n, n)–functions f_c, $c \in \mathbb{F}_2^{n_s}$ such that Proposition 5.3.2 is satisfied, that is, such that the reduced correlation matrices $\mathcal{R}^{f_c \bullet}$, $c \in \mathbb{F}_2^{n_s}$ generate a nilpotent semigroup. The problem of this approach is the huge number of candidate functions and an efficient strategy needs to be established even for very small values of n. Taking into account the result of Section 5.3.2.1 which states that the graph of f_c necessarily has a tree structure, we search suitable tree-functions at random. This approach works for small values of n typically, $n = 3$ or $n = 4$. The simultaneous triangularization can be checked by ensuring that the matrices $\mathcal{R}^{f_c \bullet}$, $c \in \mathbb{F}_2^{n_s}$ are simultaneously triangularizable by using Algorithm 1 of Chapter 3. We believe that, a better understanding of the simultaneous triangularization process will lead to additional constraints on the relation between the graphs of the functions f_c, $c \in \mathbb{F}_2^{n_s}$ and would improve the approach. According to the vocable of automatic control, it is a switched nonlinear system and thus, can be seen as an extension of switched linear systems discussed in Chapter 3.

6.3 FPGA implementation

So far, we have only discussed theoretical aspects of SSSC. However, it is important to have in mind that some Boolean functions are more relevant than others regarding hardware and software implementations. The only relevant functions for software implementations are those that can be easily expressed by means of the built-in instructions of processors. The design of SSSC based on these instructions is discussed in [66]. However, to our opinion, SSSC are best suited for hardware applications and more specifically when the size of the transmitted symbols is n_s bits. The advantage of this approach is that the encryptor can directly be plugged between the emitter and the channel and the decryptor can be plugged between the channel and the receiver. The setup is illustrated in Figure 6.5. It is particularly well suited to secure transmission systems that use modulation techniques with n_s symbols such as Phase Shift Keying (PSK for short) or Quadrature Amplitude Modulation (QAM for short).
We have designed our own test platform by following this principle. A description is provided below.

Test platform

The core of our test platform is a field-programmable gate array named Cyclone IV manufactured by Altera. A field-programmable gate array (FPGA for short) is an hardware programmable logic device. The platform is based on the *tPad development board* provided by Terasic and depicted in Figure 6.6. Besides the FPGA, the platform incorporates many input/output devices. A detailed description of its

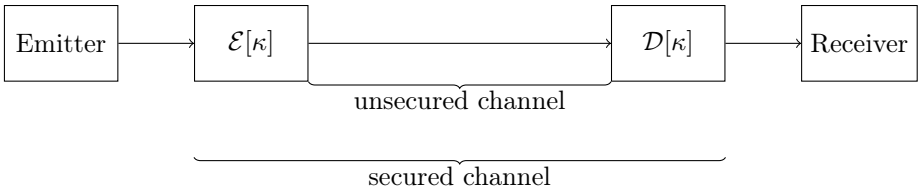


Figure 6.5

components can be found in [112]. The platform aims at processing a video stream delivered by the webcam embedded on the tPad board. The processed signal is displayed on the $8''$ LCD screen of the tPad board and sent to the VGA output which allows to display it on a video projector.

Figure 6.6: Tpad overview

Depending on the manufacturer, the architecture of an FPGA and its components may vary. However, the general scheme is always the same [113, 114]. It is composed of a grid of components which essentially contains programmable functions and memory. The connection between these components as well as their settings can be defined at the programming stage. One of the most important components for our purpose is called logical element. A simplified representation of this component is depicted in Figure 6.7. The name "logical element" may vary depending on the manufacturer. A logical element is essentially composed of a lookup table (LUT for short) which corresponds to the truthtable of a function and a one bit register. A multiplexer allows to bypass the register whenever needed.
The complexity of the logical elements keeps on increases over the years. Nowadays, the number of inputs of the LUT usually varies between six and eight [115, 116, 117] for the most advanced FPGA. It is worth pointing out that the LUT implementation of complex Boolean functions is not more expensive to implement than to implement a very simple function. Only the number of inputs matters (and the distance between the elements in the FPGA). This remark is interesting since, in many cases, the methodology that we suggest in this thesis may yield to vectorial Boolean functions with high complexity.

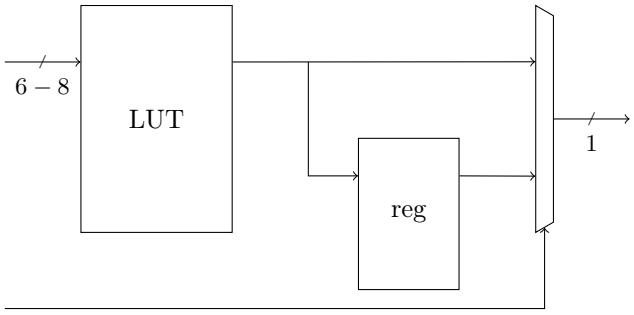
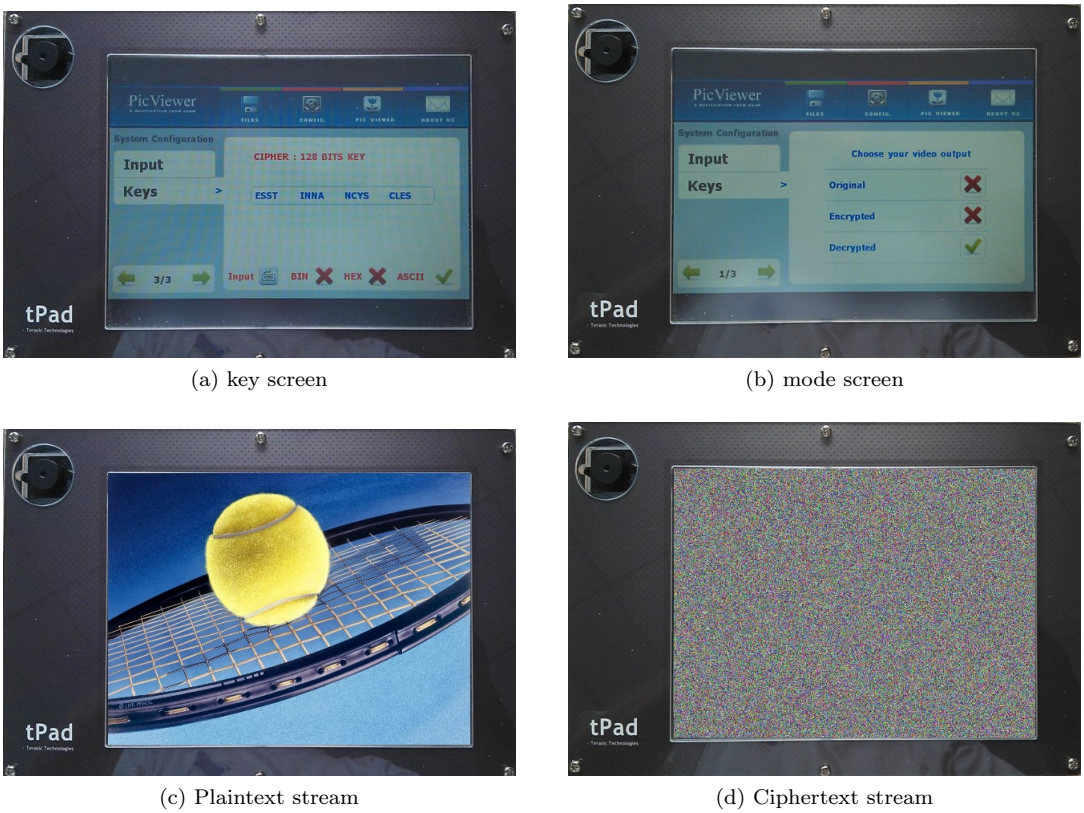


Figure 6.7: logical element

These functions would be very inefficient to implement in software. Therefore, for efficiency purposes, the current architecture of FPGA suggests to design primary constructions whose next-state function has six or eight inputs. Then, to use secondary constructions in order to reach a state with an appropriate size.

The platform involves the controller of the input/output devices, a video processing chain that includes a connection with a generic interface to adapt a self-synchronizing stream cipher and a processor. All these elements are implemented in the FPGA. As a toy example, we have implemented a self-synchronizing function with a state of size $n = 128$ bits and $n_\kappa = 128$ key bits. It is composed of a primary self-synchronizing function extended with a serial secondary construction. The primary self-synchronizing function has been chosen randomly according to the procedure described in Section 6.2.3. It is composed of a set of 16 $(5, 4)$–functions. Depending on the value of a 4-bits parameter, one of these functions is used.

A software component runs on the processor and is devoted to the device settings. A user interface has been developed for the sake of ergonomics. It allows to select and change the key of the system as shown in Figure 6.8a. It also provides a menu that let the user select what kind of signal (the plaintext, the ciphertext or the recovered plaintext) the device should display as shown in Figure 6.8b. The resources used by the test platform takes about 20% of the overall logical elements of the FPGA which let enough resources to implement cryptographic functions. Among the possible evolution we think of using the two gigabit Ethernet ports as data source and data sink.



(a) key screen

(b) mode screen

(c) Plaintext stream

(d) Ciphertext stream

Figure 6.8: tPad screenshots

Conclusion

In Chapter 2, the relation between flat dynamical systems and self-synchronizing stream ciphers has been investigated. It has been shown that, flat dynamical systems offer a novel approach for the design of self-synchronizing systems. Flat dynamical systems completely define the set of self-synchronizing systems and thus extend the existing structure of self-synchronizing ciphers.

In Chapter 3, a criterion for the characterization of flat outputs of switched linear dynamical systems has been provided. It is based on the notion of nilpotent semigroups. A connection with simultaneous triangularization is outlined and an algorithm is provided. An extension of the flat output characterization to LPV systems has also been developed and a practical test has been given. Based on the flat output characterization and on the notion of right invertibility, a constructive approach for the design of self-synchronizing master-slave setups involving switched linear systems has been suggested.

In Chapter 4, new matrix representations of vectorial Boolean functions, called numerical and algebraic normal forms, have been defined. They extend the algebraic normal form commonly used for Boolean functions. It has been shown that the matrix representations of vectorial Boolean functions are related by similarity transforms and their eigenanalysis is performed. A connection between the eigenanalysis of the matrix representations and a graph representation of the function has been carried out.

In Chapter 5, a framework to characterize self-synchronization in the Boolean context has been proposed. A first characterization of this property has been done through the notion of influence of variables. However, this characterization does not allow to give a constructive approach in view of synthesizing an SSSC. Then, a new approach using the matrix representations of the vectorial Boolean functions resulting from Shannon decomposition of the next-state function has been proposed. The nilpotent semigroups approach shows that, self-synchronization is always achieved using strict T-functions, thus restricting the possibilities. The nilpotent semigroup approach permits to use other functions than strict T-functions. Self-synchronization is related to the ability to simultaneously triangularize a set of matrices. A connection with graph theory is finally provided and a necessary graph condition is mentioned.

In Chapter 6, background on security analysis is recalled and elements on potential existing attacks are given. Then, we propose new constructions of SSSC with security elements. Finally, we describe an FPGA-based test platform which has been developed.

Overall, this work is a contribution to the characterization of the self-synchronizing mechanism for discrete-time dynamical systems both for systems described over the field of real numbers and for systems defined over finite field, in particular, involving Boolean functions. Different characterizations of the self-synchronization property have been derived through:

1. Nilpotent semigroups for switched linear systems;
2. The notion of influence;
3. The correlation or algebraic matrices and nilpotent semigroups, in the spectral domain;
4. The notion of graphs.

Even so the self-synchronization property of switched linear systems is characterized, their interest in cryptography needs to be assessed. A connection with the Maiorana McFarland construction which produces interesting functions from the cryptographic point of view might be drawn. The characterization based on the influence has a theoretical interest but not practical approach for the design of functions has been found. The most promising approach for the design of self-synchronizing functions seems to be based on correlation and algebraic matrices together with graph construction. As a matter of fact, the nilpotent semigroup criterion, or equivalently the simultaneous triangularization requirement, implies a specific structure to these matrices. Besides, the relation with the eigenanalysis allows to establish a constraint on the structure of the graph of the function. Even so the connection is not fully specified, elements have been given and were used to propose new constructions.

Future works

Flatness

Chapter 3 addressed the characterization of flat outputs. More specifically, given a specific output of the system, an approach has been proposed to check whether an output is flat for the special class of switched linear systems. A more complicated issue is the determination of the set of all admissible flat outputs of a dynamical system, given its dynamics. It has been solved in [88] for continuous-time linear systems. Further work on this topic is required.

Influence

The concept of influence of variables has been used to derive a self-synchronization characterization. It turns out that, our definition of influence was not very convenient to relate this characterization to a constructive design approach. By using the same approach and slightly changing the definition of the influence, a better connection could be found. Another approach is to use the concept of neutral bit. Neutrality of key bits on the output of a function expresses the fact that, for certain subsets of inputs corresponding to key bits, the variables of these subsets do not change the output of the function. The problem of self-synchronization can be though as a neutrality problem regarding the bits of the initial state. It could be interesting to draw a connection between influence and neutrality.

Triangularization

As explained in Section 5.3.1, triangularizable sets of reduced correlation matrices, nilpotent semigroups and self-synchronization are closely related. The graph representation together with the eigenanalysis and the simultaneous triangularization criteria suggests a well defined structure of self-synchronizing functions. A more complete connection between simultaneous triangularization and structure of the function is required.

Constructions

From the synthesis point of view, it would be interesting to find next-state functions easy to implement and such that the resulting complexity in the canonical form is high. It would also be interesting to derive constructions that could be scaled to any state of size n. More generally, an interesting result would be to determine conditions on transformations that preserve the self-synchronization property.

Key introduction

The introduction of the key in the system has not been addressed. Several approaches have been identified but deserve to be improved.

Output function

One of the main advantages of the generalized form (2.22) is that, it yields to an efficient implementation. Implementing the system is more efficient than considering the recursive canonical form (2.21). With the canonical form, most of the complexity is determined by the output function and very little is left in the next-state function. The latter one is kept very simple in order to ensure the self-synchronizing property. The purpose of the generalized form is to balance the complexity of the next-state function and of the output function. A special treatment on the choice of the output function should be interesting.

Self-synchronization

Other disciplines than cryptography are also concerned with the design of self-synchronizing systems. The paper [61] points out the use of self-synchronizing functions in the algorithm *rsync* described in [118] and in some string matching algorithms. Another relevant connection concerns self-synchronizing compression codes such as discussed in [119]. An interesting matter would be to draw the connections between these topics and the present work.

Dynamical systems over finite fields

It turns out that, finite dynamical systems are extensively used to model physical phenomena. Discrete dynamical systems are often encountered in life sciences as well [120]. Indeed, they have been used for modelling biological systems since the invention of cellular automata by Von Neumann when attempting to model a self-replicating organism in the 1950s. Since then, they have been used increasingly in biological systems to model a variety of biochemical networks. Let us mention for instance molecular networks inside human cells which process external signals and drive cellular metabolism, gene regulatory networks, large-scale epidemiological networks. Discrete dynamical systems are relevant insofar as most often, the experimental data are available with parsimony, causing continuous models such as ordinary differential equations not very suitable and difficult to obtain with accuracy. Control theory in connection with dynamical systems has not been deeply investigated, see [121, 122, 123, 124, 125] for exceptions. In [125], control for state space models over a finite field is considered. In particular, a method for synthesizing linear state feedback which, under the assumption of controllability, is proposed. Further investigations on control theory over finite fields should be interesting, in particular for dealing with discrete event systems [126].

Appendix A

Examples of functions

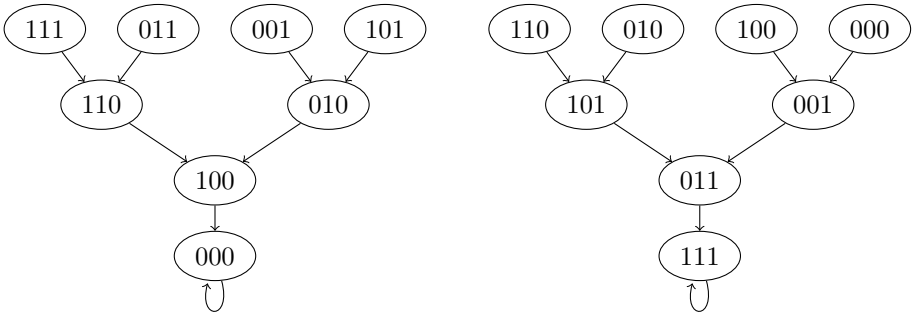
In the section we give some examples of self-synchronizing functions. Their different representations are given and the eigenanalysis of their matrix representations is performed. The next-state functions are split according to Shannon decomposition recalled in Chapter 5.

A.1 Example 1: Shift

The shift function is the next-state function corresponding to the canonical recursive form. When $n_s = 1$ and $n = 3$, it is the function $f : \mathbb{F}_2 \times \mathbb{F}_2^3 \longrightarrow \mathbb{F}_2^3$ whose Shannon decomposition is given by the two subfunctions $f_0, f_1 : \mathbb{F}_2^3 \longrightarrow \mathbb{F}_2^3$ defined by the following ANF vectors.

$$f_0(x) = \begin{pmatrix} 0 \\ x^0 \\ x^1 \end{pmatrix} \qquad f_1(x) = \begin{pmatrix} 1 \\ x^0 \\ x^1 \end{pmatrix} \tag{A.1}$$

The corresponding graphs are given in Figure A.1. The graph of the next-state function is depicted in Figure A.2. It can be checked that the preimage sets of junctions are the same in the two graphs as



(a) f_0 (b) f_1

Figure A.1: Graphs of the functions corresponding to Shannon decomposition of the shift function for $n_s = 1$ and $n = 3$

illustrated by Table A.1. One of the specificities of the shift function $\mathfrak{s}_{k_f}$ and hence, of any function isomorphic to the shift function is that, there is no synchronizing sequence for $k < k_f$. This has to be compared to the functions provided in Appendix A.2 and Appendix A.4 where the ratio of self-synchronizing sequences gradually increases until it reaches 100%.

junction	preimage set
000	$\{000, 100\}$
100	$\{110, 010\}$
110	$\{111, 011\}$
010	$\{001, 101\}$

(a) f_0

junction	preimage set
101	$\{110, 010\}$
001	$\{100, 000\}$
011	$\{101, 001\}$
111	$\{111, 011\}$

(b) f_1

Table A.1: Preimage sets of the junctions of f_0 and f_1

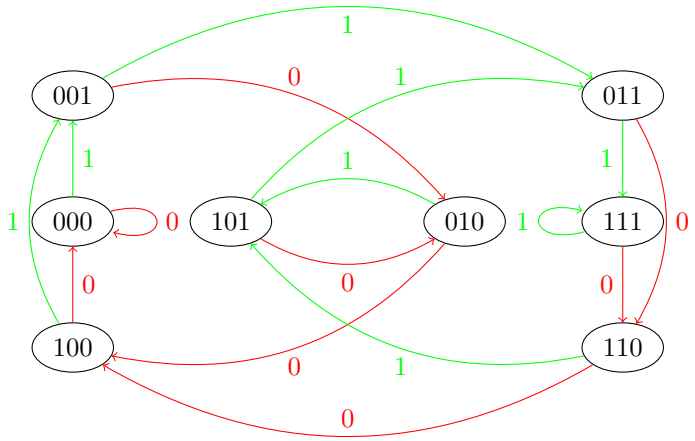


Figure A.2: Graph of the shift function for $n_s = 1$ and $n = 3$

A.1.1 Matrices

$$\mathcal{W}^{f_0} = \begin{pmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \mathcal{W}^{f_1} = \begin{pmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -8 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathcal{F}^{f_0} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \qquad \mathcal{F}^{f_1} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathcal{A}^{f_0} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \mathcal{A}^{f_1} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

A.1.2 Eigenanaysis

The eigenanalysis of these matrices is performed below according to the discussion of Section 4.3. Each of these matrices has five eigenvalues: 1 with multiplicity 1 and 0 with multiplicity 4.

$$\begin{array}{lcccccc} \text{eigenvalues} & 1 & & 0 & 0 & 0 & 0 \\ \mathcal{F}^{f_0} & \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} & & \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \end{array}$$

$$\begin{array}{lcccccc} \text{eigenvalues} & 1 & & 0 & 0 & 0 & 0 \\ \mathcal{F}^{f_1} & \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} & & \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \end{array}$$

$$\begin{array}{lcccccc} \text{eigenvalues} & 1 & & 0 & 0 & 0 & 0 \\ {}^t\mathcal{F}^{f_0} & \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} & & \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \end{pmatrix} \end{array}$$

$$\begin{array}{lcccccc} \text{eigenvalues} & 1 & & 0 & 0 & 0 & 0 \\ {}^t\mathcal{F}^{f_1} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} & & \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \end{pmatrix} \end{array}$$

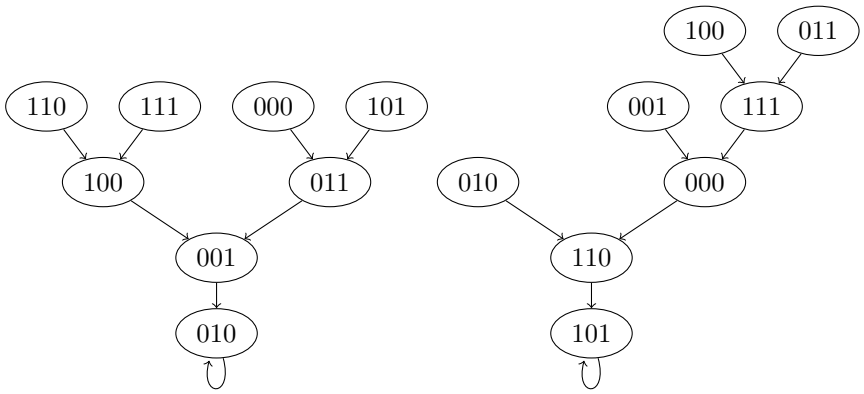
$$
\begin{array}{lccccc}
\text{eigenvalues} & 1 & 0 & 0 & 0 & 0 \\
\mathcal{W}^{f_0} & \begin{pmatrix} 1\\1\\1\\1\\1\\1\\1\\1 \end{pmatrix} & \begin{pmatrix} 0\\0\\0\\0\\1\\1\\1\\1 \end{pmatrix} & \begin{pmatrix} 0\\0\\0\\0\\1\\-1\\1\\-1 \end{pmatrix} & \begin{pmatrix} 0\\0\\0\\0\\1\\1\\-1\\-1 \end{pmatrix} & \begin{pmatrix} 0\\0\\0\\0\\1\\-1\\-1\\1 \end{pmatrix} \\
\text{eigenvalues} & 1 & 0 & 0 & 0 & 0 \\
\mathcal{W}^{f_1} & \begin{pmatrix} 1\\1\\1\\1\\1\\1\\1\\1 \end{pmatrix} & \begin{pmatrix} 0\\0\\0\\0\\1\\1\\1\\1 \end{pmatrix} & \begin{pmatrix} 0\\0\\0\\0\\1\\-1\\1\\-1 \end{pmatrix} & \begin{pmatrix} 0\\0\\0\\0\\1\\1\\-1\\-1 \end{pmatrix} & \begin{pmatrix} 0\\0\\0\\0\\1\\-1\\-1\\1 \end{pmatrix}
\end{array}
$$

A.2 Example 2

This example presents the additional representations of the example function used in Section 5.3.1 of Chapter 5.

$$f_0(x) = \begin{pmatrix} 1 \oplus x^0 \oplus x^1 \oplus x^0x^2 \\ 1 \oplus x^0x^1 \oplus x^2 \oplus x^0x^2 \\ x^1x^2 \end{pmatrix} \qquad f_1(x) = \begin{pmatrix} x^0x^1 \oplus x^2 \\ 1 \oplus x^0 \oplus x^0x^1 \oplus x^1x^2 \\ 1 \oplus x^0 \oplus x^0x^1 \oplus x^0x^2 \end{pmatrix}$$

The corresponding graphs are given in Figure A.3. The graph of the next-state function is depicted in Figure A.4.



(a) f_0 (b) f_1

Figure A.3

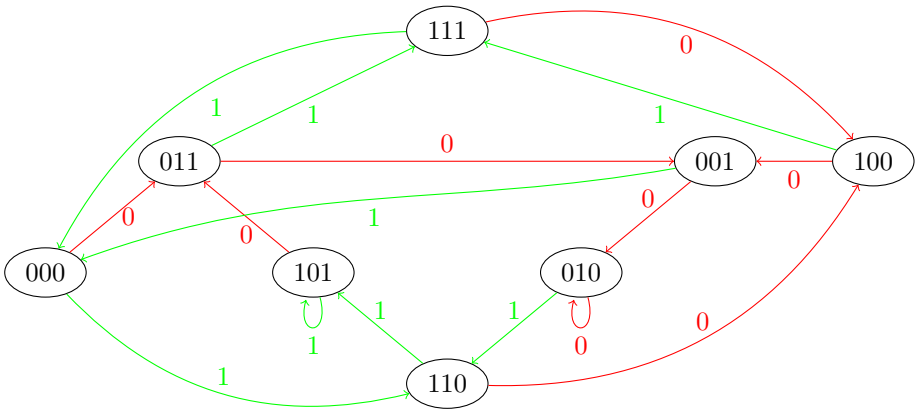


Figure A.4

It can be checked that the preimage sets of junctions are the same in the two graphs as illustrated by Table A.2.

junction	preimage set
010	$\{010, 001\}$
001	$\{100, 011\}$
100	$\{110, 111\}$
011	$\{000, 101\}$

(a) f_0

junction	preimage set
101	$\{101, 110\}$
110	$\{010, 000\}$
000	$\{001, 111\}$
111	$\{100, 011\}$

(b) f_1

Table A.2: Preimage sets of the junctions of f_0 and f_1

A.2.1 Matrices

$$\mathcal{W}^{f_0} = \begin{pmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4 & -4 & 0 & 0 & 4 & -4 \\ 0 & 0 & -4 & 4 & -4 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & -4 & 4 & 4 & 4 \\ 0 & 0 & 4 & 0 & 4 & 0 & -4 & 0 \\ -4 & 0 & 0 & -4 & 4 & 0 & 0 & -4 \\ -4 & 0 & 0 & 4 & 0 & -4 & -4 & 0 \\ -4 & 0 & 4 & 0 & 0 & 4 & 0 & 4 \end{pmatrix} \quad \mathcal{W}^{f_1} = \begin{pmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 4 & 4 & -4 \\ 0 & -4 & 0 & -4 & -4 & 0 & 4 & 0 \\ 0 & -4 & 0 & 4 & 0 & -4 & 0 & -4 \\ -4 & -4 & 0 & 0 & 0 & 0 & 4 & -4 \\ 4 & -4 & 0 & 0 & -4 & -4 & 0 & 0 \\ 4 & 0 & 0 & 4 & 4 & 0 & 0 & -4 \\ 4 & 0 & 0 & -4 & 0 & 4 & 4 & 0 \end{pmatrix}$$

$$\text{eigenvalues} \quad 1 \quad 0 \quad 0 \quad 0 \quad 0$$

$$\mathcal{A}^{f_0} \quad \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \\ 0 \\ -1 \\ 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\text{eigenvalues} \quad 1 \quad 0 \quad 0 \quad 0 \quad 0$$

$$\mathcal{A}^{f_1} \quad \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ 1 \\ -1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ -1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \end{pmatrix}$$

A.2.2 Eigenanalysis

The eigenanalysis of these matrices is performed below according to the discussion of Section 4.3. Each of these matrices has five eigenvalues: 1 with multiplicity 1 and 0 with multiplicity 4.

$$\text{eigenvalues} \quad 1 \quad 0 \quad 0 \quad 0 \quad 0$$

$$\mathcal{F}^{f_0} \quad \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\begin{array}{lccccc} \text{eigenvalues} & 1 & 0 & 0 & 0 & 0 \\ \mathcal{F}^{f_1} & \begin{pmatrix} 1\\1\\1\\1\\1\\1\\1\\1 \end{pmatrix} & \begin{pmatrix} 0\\1\\0\\0\\0\\0\\0\\0 \end{pmatrix} & \begin{pmatrix} 0\\0\\1\\0\\0\\0\\0\\0 \end{pmatrix} & \begin{pmatrix} 0\\0\\0\\1\\0\\0\\0\\0 \end{pmatrix} & \begin{pmatrix} 0\\0\\0\\0\\1\\0\\0\\0 \end{pmatrix} \end{array}$$

$$\begin{array}{lccccc} \text{eigenvalues} & 1 & 0 & 0 & 0 & 0 \\ {}^t\mathcal{F}^{f_0} & \begin{pmatrix} 0\\0\\1\\0\\0\\0\\0\\0 \end{pmatrix} & \begin{pmatrix} 0\\1\\-1\\0\\0\\0\\0\\0 \end{pmatrix} & \begin{pmatrix} 0\\0\\0\\1\\-1\\0\\0\\0 \end{pmatrix} & \begin{pmatrix} 1\\0\\0\\0\\0\\-1\\0\\0 \end{pmatrix} & \begin{pmatrix} 0\\0\\0\\0\\0\\0\\1\\-1 \end{pmatrix} \end{array}$$

$$\begin{array}{lccccc} \text{eigenvalues} & 1 & 0 & 0 & 0 & 0 \\ {}^t\mathcal{F}^{f_1} & \begin{pmatrix} 1\\1\\1\\1\\1\\1\\1\\1 \end{pmatrix} & \begin{pmatrix} 1\\0\\-1\\0\\0\\0\\0\\0 \end{pmatrix} & \begin{pmatrix} 0\\0\\0\\1\\-1\\0\\0\\0 \end{pmatrix} & \begin{pmatrix} 0\\0\\0\\0\\0\\1\\-1\\0 \end{pmatrix} & \begin{pmatrix} 0\\1\\0\\0\\0\\0\\0\\-1 \end{pmatrix} \end{array}$$

$$\begin{array}{lccccc} \text{eigenvalues} & 1 & 0 & 0 & 0 & 0 \\ \mathcal{W}^{f_0} & \begin{pmatrix} 1\\1\\-1\\-1\\1\\1\\-1\\-1 \end{pmatrix} & \begin{pmatrix} 0\\-1\\1\\0\\0\\-1\\1\\0 \end{pmatrix} & \begin{pmatrix} 0\\-1\\-1\\0\\1\\-1\\1\\0 \end{pmatrix} & \begin{pmatrix} 0\\1\\0\\1\\1\\0\\0\\1 \end{pmatrix} & \begin{pmatrix} 0\\1\\0\\-1\\0\\-1\\0\\1 \end{pmatrix} \end{array}$$

$$\begin{array}{lccccc} \text{eigenvalues} & 1 & 0 & 0 & 0 & 0 \\ \mathcal{W}^{f_1} & \begin{pmatrix} 1\\-1\\1\\-1\\-1\\1\\-1\\-1 \end{pmatrix} & \begin{pmatrix} 0\\0\\1\\1\\0\\0\\1\\1 \end{pmatrix} & \begin{pmatrix} 0\\-1\\-1\\0\\1\\0\\0\\1 \end{pmatrix} & \begin{pmatrix} 0\\-1\\1\\0\\0\\1\\-1\\0 \end{pmatrix} & \begin{pmatrix} 0\\0\\1\\-1\\1\\-1\\0\\0 \end{pmatrix} \end{array}$$

$$
\begin{array}{lccccc}
\text{eigenvalues} & 1 & 0 & 0 & 0 & 0 \\
\mathcal{A}^{f_0} & \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ 1 \\ -1 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \\
\text{eigenvalues} & 1 & 0 & 0 & 0 & 0 \\
\mathcal{A}^{f_1} & \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \\ 0 \\ -1 \\ 0 \\ 1 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ -1 \\ 1 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ -1 \\ 1 \end{pmatrix}
\end{array}
$$

A.3 Example 3

The purpose of this example is to complete the illustration of the relation between graph and eigenvalues of the matrix representations provided in Section 4.3.3 of Chapter 4. In particular, this graph has a long path $(111, 110, 101, 100, 010)$ which reduces the number of eigenvalues.

A.3.1 Matrices

Consider the $(3,3)$–function whose truthtable is

x	$f(x)$
000	010
001	000
010	001
011	001
100	010
101	100
110	101
111	110

The corresponding algebraic normal form is

$$\begin{pmatrix} x_1 \oplus x_0x_1x_2 \\ 1 \oplus x_0 \oplus x_1 \oplus x_0x_1 \oplus x_0x_1x_2 \\ x_0x_2 \oplus x_1x_2 \oplus x_0x_1x_2 \end{pmatrix}$$

and its graph is given by Figure A.5.

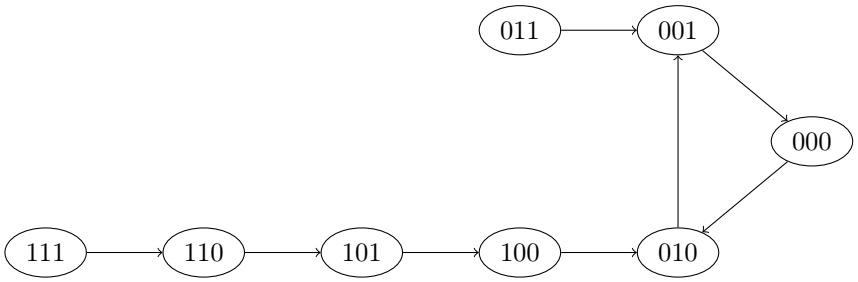


Figure A.5

The corresponding adjacency and algebraic matrices are

$$\mathcal{F} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad \mathcal{N} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & -1 & -1 & 1 & 0 \end{pmatrix}$$

$$ {}^t\mathcal{F} = \begin{pmatrix} 0&1&0&0&0&0&0&0\\ 0&0&1&1&0&0&0&0\\ 1&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0 \end{pmatrix} \qquad \mathcal{W} = \begin{pmatrix} 8&0&0&0&0&0&0&0\\ 2&-2&6&2&-2&2&2&-2\\ 2&-2&-2&-6&2&-2&-2&2\\ -4&-4&4&-4&0&0&0&0\\ 2&2&2&2&6&-2&-2&-2\\ 0&4&4&0&0&-4&4&0\\ 0&-4&-4&0&4&0&0&-4\\ -2&-2&-2&-2&-2&-2&6&-2 \end{pmatrix} $$

A.3.2 Eigenanalysis

In view of Proposition 4.3.2, the leaves 011 and 111 correspond to the eigenvalue 0 with multiplicity two. The existence of a single cycle $(000, 010, 001, 000)$ implies, due to Proposition 4.3.7, that the three 3rd roots of the unity $\alpha, \alpha^2, 1$ are also eigenvalues with multiplicity one. The following corresponding eigenvectors can be deduced.

eigenvalues	0	0	α	α^2	$\alpha^3=1$
$\mathcal{F}$	$\begin{pmatrix}0\\0\\0\\1\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix}$	$\begin{pmatrix}1\\\alpha^2\\\alpha\\\alpha\\1\\\alpha^2\\\alpha\\1\end{pmatrix}$	$\begin{pmatrix}1\\\alpha\\\alpha^2\\\alpha^2\\1\\\alpha\\\alpha^2\\1\end{pmatrix}$	$\begin{pmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{pmatrix}$

eigenvalues	0	0	α	α^2	$\alpha^3=1$
$\mathcal{N}$	$\begin{pmatrix}0\\0\\0\\1\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix}$	$\begin{pmatrix}1\\\alpha^2\\\alpha\\\alpha\\1\\\alpha^2\\\alpha\\1\end{pmatrix}$	$\begin{pmatrix}1\\\alpha\\\alpha^2\\\alpha^2\\1\\\alpha\\\alpha^2\\1\end{pmatrix}$	$\begin{pmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{pmatrix}$

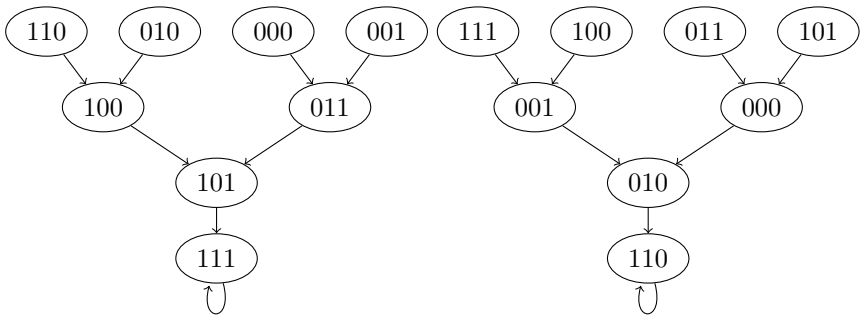
eigenvalues	0	0	α	α^2	$\alpha^3=1$
${}^t\mathcal{F}$	$\begin{pmatrix}0\\0\\1\\-1\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}1\\0\\0\\0\\-1\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}1\\\alpha\\\alpha^2\\0\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}1\\\alpha^2\\\alpha\\0\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}1\\1\\1\\0\\0\\0\\0\\0\end{pmatrix}$

eigenvalues	0	0	α	α^2	$\alpha^3=1$
$\mathcal{R}$	$\begin{pmatrix}0\\0\\0\\1\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix}$	$\begin{pmatrix}1\\\alpha^2\\\alpha\\\alpha\\1\\\alpha^2\\\alpha\\1\end{pmatrix}$	$\begin{pmatrix}1\\\alpha\\\alpha^2\\\alpha^2\\1\\\alpha\\\alpha^2\\1\end{pmatrix}$	$\begin{pmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{pmatrix}$

A.4 Example 4

This example provides the equation of a self-synchronizing function based on a balanced tree structure. However, the preimage sets of the junctions of the graph of f_0 is not the same as the preimage set of the junction of f_1 as shown by Table A.3. As a result, from Section 6.2.2, the function is not isomorphic to a shift function. Besides, a specificity of this function is that 0% of the sequence of length 2 are synchronizing, 75% of the Boolean sequence of length 3 are synchronizing and 100% of the Boolean sequence of length 4 are synchronizing.

$$f_0(x) = \begin{pmatrix} 1 \oplus x^1 \oplus x^0x^1 \\ 1 \oplus x^1 \oplus x^2 \oplus x^0x^2 \oplus x^1x^2 \\ x^1 \oplus x^2 \oplus x^1x^2 \end{pmatrix} \qquad f_1(x) = \begin{pmatrix} x^2 \oplus x^0x^2 \oplus x^1x^2 \\ 1 \oplus x^0x^1 \oplus x^2 \oplus x^1x^2 \\ x^1 \oplus x^0x^1 \end{pmatrix} \tag{A.2}$$



(a) f_0 (b) f_1

Figure A.6

junction	preimage set
100	$\{110, 010\}$
011	$\{000, 001\}$
101	$\{100, 011\}$
111	$\{101, 111\}$

(a) f_0

junction	preimage set
001	$\{111, 100\}$
000	$\{011, 101\}$
010	$\{001, 000\}$
110	$\{010, 110\}$

(b) f_1

Table A.3: Preimage sets of the junctions of f_0 and f_1

A.4.1 Matrices

$$\mathcal{W}^{f_0} = \begin{pmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -4 & 4 & -4 & -4 & 0 & 0 & 0 & 0 \\ 0 & 4 & -4 & 0 & 0 & -4 & -4 & 0 \\ 4 & 0 & 0 & -4 & 0 & 4 & 4 & 0 \\ -4 & 0 & 4 & 0 & 4 & 0 & 4 & 0 \\ 0 & -4 & 0 & 4 & -4 & 0 & -4 & 0 \\ -4 & -4 & 0 & 0 & -4 & 4 & 0 & 0 \\ 0 & 0 & 4 & 4 & 4 & -4 & 0 & 0 \end{pmatrix} \quad \mathcal{W}^{f_1} = \begin{pmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & -4 & 4 & 0 & 0 & 4 \\ 0 & -4 & 0 & 4 & -4 & 0 & -4 & 0 \\ -4 & -4 & 0 & 0 & 0 & 0 & -4 & 4 \\ 4 & -4 & 4 & 4 & 0 & 0 & 0 & 0 \\ 0 & -4 & 4 & 0 & 4 & 0 & 0 & 4 \\ 4 & 0 & -4 & 0 & -4 & 0 & -4 & 0 \\ 0 & 0 & -4 & -4 & 0 & 0 & -4 & 4 \end{pmatrix}$$

$$\mathcal{F}^{f_0} = \begin{pmatrix} 0&0&0&1&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix} \quad \mathcal{F}^{f_1} = \begin{pmatrix} 0&0&1&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&1&0&0&0&0&0&0 \end{pmatrix}$$

$$\mathcal{A}^{f_0} = \begin{pmatrix} 1&1&1&1&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&-1&-1&-1&1&0&0&0\\ 0&1&0&0&0&1&0&0\\ 0&0&-1&-1&1&1&0&0\\ 0&0&1&1&0&0&1&1\\ 0&0&1&1&-1&-1&0&0\\ 0&0&0&0&0&0&0&0 \end{pmatrix} \quad \mathcal{A}^{f_1} = \begin{pmatrix} 1&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&1&0\\ 0&0&-1&0&-1&0&-1&0\\ 0&1&-1&0&0&0&0&0\\ 0&-1&0&0&0&0&0&0\\ 0&-1&1&0&0&0&0&0\\ 0&2&0&0&0&0&0&0 \end{pmatrix}$$

A.4.2 Eigenanalysis

$$\begin{array}{ll} \text{eigenvalues} & \begin{matrix}1\end{matrix} \quad \begin{matrix}0\end{matrix} \quad \begin{matrix}0\end{matrix} \quad \begin{matrix}0\end{matrix} \quad \begin{matrix}0\end{matrix} \\ \mathcal{F}^{f_0} & \begin{pmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{pmatrix} \quad \begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{pmatrix} \quad \begin{pmatrix}0\\1\\0\\0\\0\\0\\0\\0\end{pmatrix} \quad \begin{pmatrix}0\\0\\1\\0\\0\\0\\0\\0\end{pmatrix} \quad \begin{pmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{pmatrix} \end{array}$$

$$\begin{array}{ll} \text{eigenvalues} & \begin{matrix}1\end{matrix} \quad \begin{matrix}0\end{matrix} \quad \begin{matrix}0\end{matrix} \quad \begin{matrix}0\end{matrix} \quad \begin{matrix}0\end{matrix} \\ \mathcal{F}^{f_1} & \begin{pmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{pmatrix} \quad \begin{pmatrix}0\\0\\0\\1\\0\\0\\0\\0\end{pmatrix} \quad \begin{pmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{pmatrix} \quad \begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{pmatrix} \quad \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix} \end{array}$$

$$\begin{array}{ll} \text{eigenvalues} & \begin{matrix}1\end{matrix} \quad \begin{matrix}0\end{matrix} \quad \begin{matrix}0\end{matrix} \quad \begin{matrix}0\end{matrix} \quad \begin{matrix}0\end{matrix} \\ {}^t\mathcal{F}^{f_0} & \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix} \quad \begin{pmatrix}1\\-1\\0\\0\\0\\0\\0\\0\end{pmatrix} \quad \begin{pmatrix}0\\0\\0\\1\\-1\\0\\0\\0\end{pmatrix} \quad \begin{pmatrix}0\\0\\1\\0\\0\\0\\-1\\0\end{pmatrix} \quad \begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\-1\end{pmatrix} \end{array}$$

eigenvalues	1	0	0	0	0
${}^t\mathcal{F}^{f_1}$	$\begin{pmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{pmatrix}$	$\begin{pmatrix}1\\-1\\0\\0\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\1\\0\\0\\-1\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\1\\0\\0\\0\\0\\-1\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{pmatrix}$
eigenvalues	1	0	0	0	0
$\mathcal{W}^{f_0}$	$\begin{pmatrix}1\\-1\\-1\\1\\-1\\1\\1\\-1\end{pmatrix}$	$\begin{pmatrix}0\\1\\0\\1\\0\\1\\0\\1\end{pmatrix}$	$\begin{pmatrix}0\\-1\\-1\\0\\1\\0\\0\\1\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\0\\1\\1\\-1\\-1\end{pmatrix}$	$\begin{pmatrix}0\\0\\1\\-1\\0\\0\\-1\\1\end{pmatrix}$
eigenvalues	1	0	0	0	0
$\mathcal{W}^{f_1}$	$\begin{pmatrix}1\\1\\-1\\-1\\-1\\-1\\1\\1\end{pmatrix}$	$\begin{pmatrix}0\\1\\0\\1\\0\\1\\0\\1\end{pmatrix}$	$\begin{pmatrix}0\\0\\-1\\1\\1\\-1\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\0\\1\\1\\-1\\-1\end{pmatrix}$	$\begin{pmatrix}0\\1\\1\\0\\0\\-1\\-1\\0\end{pmatrix}$
eigenvalues	1	0	0	0	0
$\mathcal{A}^{f_0}$	$\begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}1\\-1\\-1\\1\\-1\\1\\1\\-1\end{pmatrix}$	$\begin{pmatrix}0\\1\\0\\-1\\0\\-1\\0\\1\end{pmatrix}$	$\begin{pmatrix}0\\0\\1\\-1\\0\\0\\-1\\1\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\0\\0\\0\\1\\-1\end{pmatrix}$
eigenvalues	1	0	0	0	0
$\mathcal{A}^{f_1}$	$\begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\1\\0\\0\\0\\-1\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\0\\1\\-1\\-1\\1\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\-1\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix}$

Appendix B

Basics on algebra

Definition B.0.1 (Field)**.** A field is a 3-uplet $(\mathbb{F},+,\cdot)$ where $\mathbb{F}$ is a set, $+$ an operation usually called *addition* and $\cdot$ an operation usually called *multiplication*. The following properties apply:

1. For all a and b in $\mathbb{F}$ both $a+b$ and $a\cdot b$ are $\mathbb{F}$;
2. For all a, b and c in $\mathbb{F}$, associativity holds: $a+(b+c)=(a+b)+c$ and $a\cdot(b\cdot c)=(a\cdot b)\cdot c$;
3. For all a and b in $\mathbb{F}$, commutativity holds: $a+b=b+a$ and $a\cdot b=b\cdot a$;
4. There exists an element of $\mathbb{F}$, called the additive identity element and denoted by 0 such that for all a in $\mathbb{F}$, $a+0=a$. Likewise, there is an element called the multiplicative identity element and denoted by 1, such that for all a in $\mathbb{F}$, $a\cdot 1=a$. The identity elements 0 and 1 have to be different;
5. For every a in $\mathbb{F}$, there exists an element $-a$ in $\mathbb{F}$ such that $a+(-a)=0$. Similarly, for any a in $\mathbb{F}$ other than 0, there exists an element a^{-1} in $\mathbb{F}$ such that $a\cdot a^{-1}=1$;
6. For all a, b and c in $\mathbb{F}$ distributivity of the multiplication over the addition holds: $a\cdot(b+c)=(a\cdot b)+(a\cdot c)$.

A field is said to be finite if it has a finite number of elements.

Definition B.0.2 (Lie algebra)**.** A lie algebra $\mathfrak{g}$ is a vector space $\mathbb{F}^n$ over the field $\mathbb{F}$ together with a binary operation $[\cdot,\cdot]:\mathfrak{g}\times\mathfrak{g}\longrightarrow\mathfrak{g}$ called Lie bracket which:

- is bilinear, $\forall a,b\in\mathbb{F}$, $x,y,z\in\mathfrak{g}$, $[ax+by,z]=a[x,z]+b[y,z]+b[y,z]$ and $[z,ax+by]=a[z,x]+b[z,y]$;
- is alternating on $\mathfrak{g}$: $\forall x\in\mathfrak{g}$, $[x,x]=0$;
- satisfies Jacoby identity, $\forall x,y,z\in\mathfrak{g}$, $[x,[y,z]]+[y,[z,x]]+[z,[x,y]]=0$.

Definition B.0.3 (Solvable Lie algebra)**.** Let $\mathfrak{g}$ be a Lie algebra. We recursively define $\mathbb{U}_k$ for $k\in\mathbb{N}$ by

$$\mathcal{U}_k=\begin{cases}\mathfrak{g} & \text{if } k=0\\ [\mathcal{U}_{k-1},\mathcal{U}_{k-1}] & \text{else}\end{cases}$$

A Lie algebra is said to be solvable if $\exists k\in\mathbb{N}$, $\mathcal{U}_k=0$

Definition B.0.4 (Nilpotent Lie algebra)**.** A Lie algebra $\mathfrak{g}$ is nilpotent if

$$\exists k\in\mathbb{N},\ \forall x_1,\dots,x_k\in\mathfrak{g},\ [[\cdots[[x_1,x_2],x_3],\dots],x_k]=0$$

Appendix C

Simultaneous triangularization

This appendix provides the Matlab code corresponding to Algorithm 1 which return a simultaneous triangularization basis of a set of matrices whenever it exists, as required in Chapter 3 and Chapter 5.

```
function S=GetSimultaneousTrig(P)
    [m,n,p]=size(P);
    S2=eye(n);
    k=0;
    T=P;
    S1=zeros(m,0);
    for k=0:(n-2)
        %Determines a common eigenvector named v
        clear Q;
        Q=T(:,:,1);
        for j=2:p
            Q=vertcat(Q,T(:,:,j));
        end
        commoneigenvectors=null(Q);
        if(length(commoneigenvectors)==0)
            display('No basis exists.');
            break;
        end
        %Construct the change of basis
        v=commoneigenvectors(:,1);
        vkp1=S2*v;
        S1=[S1,vkp1];
        %Extends the set of column vectors of S1 into a basis
        S2=null(S1');
        %Prepares the matrices T for the next iteration
        S=[S1,S2];
        Sinv=inv(S);
        clear T
        for i=1:p
            tmp =Sinv*P(:,:,i)*S;
            T(:,:,i)=tmp(k+2:n,k+2:n);
        end
    end
end
```

Index

Bibliography

[1] S. Strogatz. *Sync*. Hyperion, 2004.

[2] *Controlling Chaos and Bifurcations in Engineering Systems*. CRC Press, 1999.

[3] J.H. Holland. *Emergence: From Chaos to Order*. Oxford Univ Pr (Sd), 2000.

[4] W. Perruquetti and J-P Barbot. *Chaos in Automatic Control (Automation and Control Engineering)*. CRC Press, 2005.

[5] S. Banerjee. *Chaos Synchronization and Cryptography for Secure Communications: Applications for Encryption*. IGI Global, 2010.

[6] C. Cruz-Hernandez and A.A. Martynyuk. *Advances in Chaotic Dynamics and Applications*. 2010.

[7] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949.

[8] F. Dachselt, K. Kelber, J. Vandewalle, and W. Schwarz. Chaotic versus classical stream ciphers – a comparative study. In *Proc. of Int. Symp. on Circuits and Systems ISCAS'98*, volume IV, pages 518–521, Monterey, June 1998.

[9] L. Kocarev. Chaos-based cryptography :a brief overview. *IEEE Circuits and Systems Magazine*, 1(3):6–21, 2001.

[10] G. Millérioux, J.M. Amigó, and J. Daafouz. A connection between chaotic and conventional cryptography. *IEEE Trans. On Circuits and Systems I: Regular Papers*, 2008.

[11] J. Fridrich. Symmetric ciphers based on two-dimensional chaotic maps. *International Journal of Bifurcation and Chaos*, 8(6):1259 – 1284, June 1998.

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

[13] J. Szczepanski, J.M. Amigó, T. Michalek, and L. Kocarev. Crytographically secure substitutions based on the approximation of mixing maps. *IEEE Trans. Circuits and Systems I : Regular Papers*, 52(2):443–453, February 2005.

[14] J.M. Amigó, J. Szczepanski, and L. Kocarev. A chaos-based approach to the design of crytographically secure substitutions. *Phys. Lett. A*, 343:55–60, February 2005.

[15] K.M. Cuomo, A.V. Oppenheim, and S.H. Strogatz. Synchronization of lorenz-based chaotic circuits with applications to communications. *IEEE Trans. Circuits. Syst. II: Anal. Digit. Sign. Process*, 40(10):626–633, 1993.

[16] C.W. Wu and L.O. Chua. A simple way to synchronize chaotic systems with applications to secure communications systems. *International Journal of Bifurcation and Chaos*, 3(6):1619–1627, 1993.

[17] G. Kolumban, M.P. Kennedy, and Chua L.O. The role of synchronization in digital communications using chaos - part II: Chaotic modulation and chaotic synchronization. *IEEE Trans. Circuits. Syst. I: Fundamental Theo. Appl*, 45:1129–1140, November 1998.

[18] H. Dedieu, M.P. Kennedy, and M. Hasler. Chaos shift keying: modulation and demodulation of a chaotic carrier using self-synchronizing chua's circuits. *IEEE Trans. Circuits. Syst. II: Anal. Digit. Sign. Process*, 40:634–642, 1993.

[19] U. Parlitz, L.O. Chua, L. Kocarev, K.S. Halle, and A. Shang. Transmission of digital signals by chaotic synchronization. *International Journal of Bifurcation and Chaos*, 3(2):973–977, 1993.

[20] A.L. Fradkov and A.Y. Markov. Adaptive synchronization of chaotic systems based on speed-gradient method and passification. *IEEE Trans. Circuits. Syst. I: Fundamental Theo. Appl*, 44(10):905–912, Oct. 1997.

[21] H.J.C. Huijberts, H. Nijmeijer, and R. Willems. System identification in communication with chaotic systems. *IEEE Trans. Circuits. Syst. I: Fundamental Theo. Appl*, 47(6):800–808, 2000.

[22] H. Dedieu and M. Ogorzalek. Identification of chaotic systems based on adaptive synchronization. In *Proc. ECCTD'97*, pages 290–295, Budapest, Sept. 1997.

[23] F. Anstett, G. Millérioux, and G. Bloch. Global adaptive synchronization based upon polytopic observers. In *Proc. of IEEE International symposium on circuit and systems, ISCAS'04*, pages 728 – 731, Vancouver, Canada, May 2004.

[24] G. Millérioux and C. Mira. Coding scheme based on chaos synchronization from noninvertible maps. *International Journal of Bifurcation and Chaos*, 8(10):2019–2029, 1998.

[25] Z.P. Jiang. A note on chaotic secure communication systems. *IEEE Trans. Circuits. Syst. I: Fundamental Theo. Appl*, 49(1):92–96, January 2002.

[26] K-Y. Lian and P. Liu. Synchronization with message embedded for generalized lorenz chaotic circuits and its error analysis. *IEEE Trans. Circuits. Syst. I: Fundamental Theo. Appl*, 47(9):1418–1424, 2000.

[27] G. Millérioux and J. Daafouz. Unknown input observers for message-embedded chaos synchronization of discrete-time systems. *International Journal of Bifurcation and Chaos*, 14(4):1357–1368, April 2004.

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

[29] M. Hasler. Synchronization of chaotic systems and transmission of information. *International Journal of Bifurcation and Chaos*, 8(4):647–659, April 1998.

[30] T. Yang, C.W. Wu, and L.O. Chua. Cryptography based on chaotic systems. *IEEE Trans. Circuits. Syst. I: Fundamental Theo. Appl*, 44(5):469–472, May 1997.

[31] A.T. Parker and K.M. Short. Reconstructing the keystream from a chaotic encryption scheme. *IEEE Trans. on Circ. and Syst.*, 48(5):624–630, May 2001.

[32] F. Robert. *Les Systèmes Dynamiques Discrets*. Springer, 1995.

[33] J. Gleick. *Chaos: Making a New Science*. Vintage, 1997.

[34] H. Poincaré. *Les nouvelles méthodes de la mécanique céleste*. Gauthier-Villars., 1982. In English: NASA Translation TTF-450/452, U.S. Federal CLearinhouse, Springfield, VA, 1967.

[35] E.N. Lorenz. Deterministic nonperiodic flow. *American Meteorological Society*, 20:130–141, 1963.

[36] T.Y. Li and J.A. Yorke. Period three implies chaos. *Amer. Math. Monthly*, 82:985–992, 1975.

[37] R.L. Devaney. *An Introduction to Chaotic Dynamical Systems*. Westview Press, 2003.

[38] I.I. Blekhman, A.L. Fradkov, Nijmeijer H., and A.Y Pogromsky. On self-synchronization and controlled synchronization. *Systems and Control letters*, 31(5):299–305, 1997.

[39] A. Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–83, January 1883.

[40] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[41] D. Whitfield and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644 – 654, 1976.

[42] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[43] B. Schneier. *Applied Cryptography*. John Wiley & Sons, 1996.

[44] J. Daemen. *Cipher and Hash function design, strategies based on linear and differential cryptanalysis*. PhD Thesis, Katholieke Universiteit Leuven, 1995.

[45] Data encryption standard (des). Technical report, NIST, 1999. http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

[46] J. Daemen and V. Rijmen. *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag, 2002.

[47] M. Dworkin. Recommendation for block cipher modes of operations. Technical report, NIST, 2001. NIST Special Publication 800-38A.

[48] G.S. Vernam. Secret signaling system. patent 1,310,719A.

[49] NIST. A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications. Technical report, NIST, April 2010. NIST Special Publication 800-22rev1a.

[50] J. Daemen, R. Govaerts, and J. Vandewalle. A practical approach to the design of high speed self-synchronizing stream-ciphers. In *Singapore ICCS/ISITA, 1992*, pages 279–283. Springer-Verlag, 1992.

[51] A. Joux and F. Muller. Loosening the knot. *Fast Software Encryption, Lecture Note in Computer Science*, pages 87–99, Springer 2003.

[52] P. Sarkar. Hiji-bij-bij: A new stream cipher with a self-synchronizing mode of operation. In *INDOCRYPT 2003*, 2003.

[53] A. Joux and F. Muller. Two attacks against the hbb stream cipher. Technical report, DGA, DCSSI, 2004.

[54] V. Klíma. Cryptanalysis of hiji-bij-bij (hbb). Technical report, 2005. http://eprint.iacr.org/2005/003.pdf.

[55] P. Hawkes, M. Paddon, G.R. Gregory, and W.V. Miriam. Primitive specification for sss. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/028, 2005. http://www.ecrypt.eu.org/stream/ciphers/sss/sss.pdf.

[56] J. Daemen and P. Kitsos. The self-synchronizing stream cipher mosquito: estream documentation, version 2. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/018, 2005. http://www.ecrypt.eu.org/stream/p3ciphers/mosquito/mosquito.pdf.

[57] J. Daemen, J. Lano, and B. Preneel. Chosen ciphertext attack on sss. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/044, 2005. http://www.ecrypt.eu.org/stream/papersdir/044.pdf.

[58] A. Joux and F. Muller. Chosen-ciphertext attacks against mosquito. *Fast Software Encryption, Lecture Note in Computer Science*, 4047, Springer 2006.

[59] J. Daemen and P. Kitsos. The self-synchronizing stream cipher moustique. In Matthew Robshaw and Olivier Billet, editors, *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 210–223. Springer Berlin / Heidelberg, 2008. http://dx.doi.org/10.1007/978-3-540-68351-3_16.

[60] E. Käsper, V. Rijmen, T. Bjørstad, C. Rechberger, M. Robshaw, and G. Sekar. Correlated keystreams in moustique. In *Proceedings of the Cryptology in Africa 1st international conference on Progress in cryptology*, AFRICACRYPT'08, pages 246–257, Berlin, Heidelberg, 2008. Springer-Verlag.

[61] A. Klimov and A. Shamir. New applications of t-functions in block ciphers and hash functions. *Fast Software Encryption*, pages 18–31, 2005.

[62] S. Khazaei and W. Meier. New directions in cryptanalysis of self-synchronizing stream ciphers. *INDOCRYPT*, 2008.

[63] F. Arnault, T. Berger, and A. Necer. A new class of stream ciphers combining lfsr and fcsr architectures. In Alfred Menezes and Palash Sarkar, editors, *Progress in Cryptology – INDOCRYPT 2002*, volume 2551 of *Lecture Notes in Computer Science*, pages 22–33. Springer Berlin / Heidelberg, 2002.

[64] B. Zhang, H. Wu, D. Feng, and F. Bao. Chosen ciphertext attack on a new class of self-synchronizing stream ciphers. In Anne Canteaut and Kapaleeswaran Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004*, volume 3348 of *Lecture Notes in Computer Science*, pages 1219–1221. Springer Berlin / Heidelberg, 2005.

[65] A. Klimov and A. Shamir. A new class of invertible mappings. In Burton Kaliski, çetin Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 470–483. Springer Berlin / Heidelberg, 2003. 10.1007/3-540-36400-5_34.

[66] A. Klimov and A. Shamir. Cryptographic applications of t-functions. In Mitsuru Matsui and Robert Zuccherato, editors, *Selected Areas in Cryptography*, volume 3006 of *Lecture Notes in Computer Science*, pages 248–261. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-24654-1_18.

[67] U. Maurer. New approaches to the design of self-synchronizing stream ciphers. pages 458–471. Springer Verlag, 1991.

[68] O. Jung and C. Ruland. Encryption with statistical self-synchronization in synchronous broadband networks. In Çetin Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems*, volume 1717 of *Lecture Notes in Computer Science*, pages 74–74. Springer Berlin / Heidelberg, 1999. 10.1007/3-540-48059-5_29.

[69] H.M. Heys. An analysis of the statistical self-synchronization of stream ciphers. *IEEE Computer and Communications Societies. Proceedings*, 2:897–904, 2001.

[70] A. Alkassar, A. Geraldy, B. Pfitzmann, and A. Sadeghi. Optimized self-synchronizing mode of operation. In *IN PROCEEDINGS OF THE 8TH INTERNATIONAL WORKSHOP ON FAST SOFTWARE ENCRYPTION*, page 87. Springer-Verlag, 2001.

[71] F. Yang and H.M. Heys. Comparison of two self-synchronizing cipher modes. *Queen's 22nd Biennial Symposium on Communications*, 2004.

[72] M. Fliess, J. Lévine, and P. Rouchon. Flatness and defect of nonlinear systems: Introductory theory and examples. *International Journal of Control*, 61:1327–1361, 1995.

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

[74] R. Nitsche, M. Bitzer, M. El Khaldi, and G. Bloch. Fault tolerant oxygen control of a diesel engine air system. 2010.

[75] F. Cazaurang. *Commande robuste des systèmes plats application à la commande d'une machine synchrone*. PhD thesis, 1997.

[76] G. Millérioux and J. Daafouz. Flatness of Switched Linear Discrete-time Systems. *IEEE Transactions on Automatic Control*, 54(3):615–619, 03 2009.

[77] J. Parriaux and G. Millérioux. A constructive approach for the design of self-synchronizing dynamical systems: an application to communications. *International Federation of Automatic Control, IFAC World Congress*, 2011.

[78] Jérémy Parriaux and Gilles Millérioux. Designing self-synchronizing switched linear systems: An application to communications. *Nonlinear Analysis: Hybrid Systems*, 7(1):68 – 79, 2013. <ce:title>IFAC World Congress 2011</ce:title>.

[79] J. Parriaux and G. Millérioux. Nilpotent semigroups for the characterization of flat outputs of discrete-time switched linear and lpv systems. *Conference on Decision and Control, CDC*, 2012.

[80] R.W. Brockett and M.D. Mesarovic. The reproducibility of multivariable systems. *J. Math. Anal. Appl.*, 11:548–563, July 1965.

[81] L. Vu and D. Liberzon. Invertibility of switched linear systems. *Automatica*, 44(4):949–958, 2008.

[82] T. Floquet and J-P. Barbot. State and unknown input estimation for linear discrete-time systems. *Automatica*, 42(11):1883 – 1889, 2006.

[83] S. Sundaram and C. Hadjicostis. Designing stable inverters and state observers for switched linear systems with unknown inputs. In *Proc. of the 45th IEEE Conference on Decision and Control*, San Diego, CA, USA, December 2006.

[84] G. Millérioux and J. Daafouz. *Proc. of the 10th International Conference on Hybrid Systems: Computation and Control (HSCC'07)*, volume 44, chapter Invertibility and Flatness of Switched Linear Discrete-time Systems, pages 714–717. Springer Verlag, Pisa, Italy, April 2007.

[85] M.K. Sain and J.L. Massey. Invertibility of linear time-invariant dynamical systems. *IEEE Trans. Automat Control*, 14(2):141–149, 1969.

[86] R.M. Hirschorn. Invertibility of nonlinear control systems. *SIAM J. Control and Optimization*, 17(2), March 1979.

[87] A. Ben-Israel and T. N. E. Greville. *Generalized inverses: Theory and applications*. Krieger, 1980.

[88] J. Lévine and D.V. Nguyen. Flat output characterization for linear systems using polynomial matrices. *IEEE Trans. Circuits. Syst. I: Fundamental Theo. Appl*, 48:69–75, 2003.

[89] H. Radjavi and P. Rosenthal. *Simultaneous Triangularization*. Springer, 2000.

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

[91] D. Liberzon, J.P. Hespanha, and A.S. Morse. Stability of switched systems : a lie-algebraic condition. *Systems and Control Letters*, 37:117–122, 1999.

[92] Y. Mori. A solution to the common lyapunov function problem for continuous-time systems. *Decision and Control*, 4:3530–3531, 1997.

[93] C.B. Schrader and M.K. Sain. Research on system zeros: a survey. *Int. Jour. of Control*, 50(4):1407–1433, 1989.

[94] D. Bajić and C. Stefanović. Statistical analysis of search for set of sequences in random and framed data. In Claude Carlet and Alexander Pott, editors, *Sequences and Their Applications – SETA 2010*, volume 6338 of *Lecture Notes in Computer Science*, pages 320–332. Springer Berlin / Heidelberg, 2010.

[95] C. Carlet. *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, chapter Boolean Functions for Cryptography and Error-Correcting Codes. In [127], 2010.

[96] C. Carlet and P. Guillot. A new representation of boolean functions. In Marc Fossorier, Hideki Imai, Shu Lin, and Alain Poli, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 1719 of *Lecture Notes in Computer Science*, pages 731–731. Springer Berlin / Heidelberg, 1999. 10.1007/3-540-46796-3_10.

[97] C. Carlet. *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, chapter Vectorial Boolean Functions for Cryptography. In [127], 2010.

[98] J. Daemen, R. Govaerts, and J. Vandewalle. Correlation matrices. In *Fast Software Encryption : Second International Workshop, LNCS 1008*, pages 275–285. Springer-Verlag, 1994.

[99] J. Parriaux, P. Guillot, and G. Millérioux. Towards a spectral approach for the design of self-synchronizing stream ciphers. *Cryptography and Communications*, 3:259–274, 2011. 10.1007/s12095-011-0046-2.

[100] K. Nyberg and M. Hermelin. Multidimensional walsh transform and a characterization of bent functions. *Proceedings of the 2007 IEEE Information Theory Workshop on Information Theory for Wireless Networks*, 2007.

[101] C. Godsil and G. Royle. *Algebraic Graph Theory*. Springer, 2001.

[102] P. Guillot. *Fonctions courbes binaires et transformation de Möbius*. PhD thesis, 1999.

[103] C. Shannon. The synthesis of two-terminal switching circuits. *Bell System Technical Journal*, 28:59–98, 1949.

[104] J. Kahn, G. Kalai, and N. Linial. The influence of variables on boolean functions. In *SFCS '88: Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 68–80, Washington, DC, USA, 1988. IEEE Computer Society.

[105] G. Kindler and S. Safra. Noise-resistant boolean-functions are juntas, 2003.

[106] J. Parriaux, P. Guillot, and G. Millérioux. Synchronization of boolean dynamical systems: A spectral characterization. In Claude Carlet and Alexander Pott, editors, *Sequences and Their Applications, SETA 2010*, volume 6338 of *Lecture Notes in Computer Science*, pages 373–386. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-15874-2_32.

[107] A. Cayley. A theorem on trees. *Quaterly Journal of Pure and Applied Mathematics*, pages 376–378, 1889.

[108] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270 – 299, 1984.

[109] E. Biham and A. Shamir. Differential cryptanalysis of des-like cryptosystems. In Alfred Menezes and Scott Vanstone, editors, *Advances in Cryptology-CRYPTO 90*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer Berlin / Heidelberg, 1991. 10.1007/3-540-38424-3_1.

[110] F. Muller. Differential attacks and stream ciphers. Technical report, ECRYPT Network of Excellence in Cryptology, 2004.

[111] M. Matsui. Linear cryptanalysis method for des cipher. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT 93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer Berlin / Heidelberg, 1994. 10.1007/3-540-48285-7_33.

[112] Terasic. tpad user manual, 2011. http://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=550&FID=bb7ae3410a29302fc7eb4e60cf63fa1e.

[113] Altera. Fpga architecture. Technical report, 2006. http://www.altera.com/literature/wp/wp-01003.pdf.

[114] Xilinx. Xilinx 7 series fpgas embedded memory advantages. Technical report, 2012.

[115] Altera. Stratix v device handbook. Technical report, 2012.

[116] Xilinx. Spartan-6 family overview. Technical report, 2011.

[117] Xilinx. 7 series fpgas overview. Technical report, 2012.

[118] A. Tridgell and P. Mackerras. The rsync algorithm. Technical report, Australian National University, 1996.

[119] M.R. Titchener. Generalized t-codes: An extended construction algorithm for self-synchronizing codes. *Communications, IEE Proceedings*, 143(3):122–128, 1996.

[120] B. Stigler. Polynomial dynamical systems in biology. In *Modeling and Simulation of Biological Networks*, pages 59–84, 2006.

[121] M. Le Borgne, A. Benveniste, and P. Le Guernic. Polynomial dynamical systems over finite fields. In Gérard Jacob and Françoise Lamnabhi-Lagarrigue, editors, *Algebraic Computing in Control*, volume 165 of *Lecture Notes in Control and Information Sciences*, pages 212–222. Springer Berlin / Heidelberg, 1991. 10.1007/BFb0006940.

[122] O. Colón-Reyes, A. Jarrah, R. Laubenbacher, and B. Sturmfels. Monomial dynamical systems over finite fields. 2006.

[123] L. Misiurewicz, G. Stevens John, and keywords = "Ducci sequences" keywords = "Cellular automata" keywords = "Dynamics of linear maps" Diana, M.T.". Iterations of linear maps over finite fields. *Linear Algebra and its Applications*, 413(1):218 – 234, 2006.

[124] M. Le Borgne. Dynamical systems over galois fields: Applications to des and to the SIGNAL language. 1993.

[125] J. Reger. *Linear Systems over Finite Fields – Modeling, Analysis and Synthesis*. PhD thesis, Det Technischen Fakultät der Universität Erlangen-Nürnberg, 2004.

[126] J. Gunnarsson. Algebraic methods for discrete event systems - a tutorial. Technical report, Division of Electrical Engineering Linkoping University, 1996.

[127] Y. Crama. *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*. Cambridge Press, 2010.

Résumé

Cette thèse traite de la synchronisation des systèmes dynamiques. La synchronisation est étudiée pour une configuration de type maître-esclave, c'est-à-dire pour des systèmes couplés de façon unidirectionnelle. Ce type de configuration s'avère d'un intérêt tout particulier car elle correspond à des architectures de communications chiffrées un-vers-un ou un-vers-plusieurs. Une attention spécifique est portée sur l'autosynchronisation, comportement qui caractérise la synchronisation par le simple couplage maître-esclave et donc en l'absence de tout contrôle extérieur. Elle joue un rôle majeur dans les communications impliquant des chiffreurs par flot autosynchronisants. L'étude de l'autosynchronisation dans le contexte cryptographique s'appuie sur la théorie du contrôle. Un lien original entre l'autosynchronisation et le principe de chiffrement/déchiffrement en cryptographie est mis en évidence. Il fait appel à la propriété de platitude des systèmes dynamiques, un concept emprunté à l'automatique. On montre que les systèmes dynamiques plats définissent complètement l'ensemble des systèmes autosynchronisants et permettent d'élargir les structures existantes des chiffreurs autosynchronisants. La platitude est tout d'abord étudiée pour deux types de systèmes non linéaires : les systèmes linéaires commutés et à paramètres variants (LPV). La caractérisation des sorties plates s'appuie sur le concept de semigroupes nilpotents et un algorithme performant est proposé. Une approche constructive pour réaliser des structures maître-esclave autosynchronisantes est proposée sur la base de systèmes plats et les notions d'inversibilité à gauche et à droite empruntées à la théorie du contrôle. Par la suite, l'autosynchronisation est étudiée dans le contexte booléen privilégié en cryptographie. Elle est caractérisée en premier lieu au travers la notion d'influence. Ensuite, différentes représentations matricielles associées aux fonctions booléennes sont proposées. Ces représentations s'avèrent particulièrement intéressantes pour l'analyse des propriétés liées à la sécurité. Un lien entre l'autosynchronisation et les structures propres des représentations matricielles est établi. Une approche orientée graphes est finalement élaborée pour la caractérisation. De nouvelles constructions de structures autosynchronisantes en sont déduites et des éléments de sécurité sont discutés. Enfin, une plateforme de test à base de FPGA qui a été réalisée est décrite.

Mots-clés: systèmes à commutation ; platitude ; chiffreur par flot autosynchronisant ; fonctions booléennes ; corps finis

Abstract

This thesis deals with the synchronization of dynamical systems. The synchronization considered is called master-slave, that is, the dynamical systems are connected in a unidirectional way. This configuration is of interest because it corresponds to an architecture encountered in secured communications of type one-to-one or one-to-many. A special attention is paid to self-synchronization, the behaviour that characterizes synchronization achieved with a simple master-slave coupling and so, without any external control. It is a central feature of self-synchronizing stream ciphers. The study of self-synchronization in the cryptographic context relies on control theory. An original connection between self-synchronization and encryption/decryption is provided. It is based on the flatness property of dynamical systems, a property borrowed from automatic control. It is shown that flat dynamical systems completely define the set of all self-synchronizing systems and thus, enlarge the existing structures of self-synchronizing stream ciphers. Flatness is first of all studied for the case of two nonlinear systems: switched linear systems and linear parameter-varying (LPV) systems. Flatness characterization is based on the concept of nilpotent semigroups and an efficient algorithm is provided. A constructive approach for self-synchronizing master-slave structures is proposed. It relies on the construction of flat systems as well as on left and right invertibility also borrowed from control theory. Then, self-synchronization is studied in the Boolean context which is preferred in cryptography. Self-synchronization is characterized through the notion of influence. Several matrix representations of Boolean functions are proposed. These representations are especially interesting for security analysis. A connection between self-synchronization and the eigenstructures of these matrices is established. Then, a graph oriented approach is provided. New self-synchronizing constructions are deduced and security elements are discussed. Eventually, the description of the FPGA based test platform that we have designed is provided.

Keywords: switched systems; flatness; self-synchronizing stream cipher; Boolean functions; dynamical systems over finite fields