UNIVERSITÉ DE LORRAINE

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

Synchronisation et systèmes dynamiques : application à la cryptographie

THÈSE

présentée et soutenue publiquement le 06 Juillet 2017

pour l'obtention du

Doctorat de l'Université de Lorraine

(Spécialité Automatique)

par

Brandon DRAVIE

Composition du jury

Rapporteurs :	Jean-Pierre BARBOT	Professeur QUARTZ EA 7393 / ENSEA, Cergy-Pontoise
	Anne CANTEAUT	Directrice de recherche SECRET, INRIA (Paris)
Examinateurs :	Joos VANDEWALLE	Professeur KU Leuven
	Florent BERNARD	Maître de Conférences Université Jean Monnet, Saint-Etienne
	Marine MINIER	Professeur LORIA, Université de Lorraine
Co-directeur :	Philippe GUILLOT	Maître de Conférences LAGA, Université Paris 8
Directeur :	Gilles MILLERIOUX	Professeur CRAN, Université de Lorraine

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

CRAN

Résumé détaillé de la thèse

La synchronisation a été un sujet important en automatique depuis des années. Rigoureusement parlant, le terme synchronisation désigne des comportements similaires ou corrélés de deux ou plusieurs entités qui sont connectées par des réseaux virtuel ou physique. Au cours de ces derniers siècles, des scientifiques ont tenté d'expliquer l'existence de l'ordre à travers le concept de synchronisation. C Huygens est considéré comme un des pionniers à s'intéresser à ce sujet en 1665. On peut citer un nombre remarquable de phénomènes synchronisés que l'on peut observer que ce soit dans la nature, la biologie, les neurosciences, la physiologie et plus récemment dans les réseaux sociaux.

La synchronisation peut être un moyen très efficace pour aborder un problème en ingénierie. Par exemple, il existe un intérêt croissant aux problèmes de contrôle collaboratif. De tels problèmes font intervenir plusieurs entités autonomes (aussi appelé agents) qui, à travers un accouplement ou une connectivité approprié, tentent d'atteindre collectivement un objectif global. On peut citer dans ce cas comme applications, des robots mobiles, des véhicules autonomes et sans pilotes, les satellites, les contrôleurs de pollution d'air. Par ailleurs, la synchronisation joue un rôle important dans la communication: vidéo, radiodiffusion, des équipements utilisant une boucle à verrouillage de phase. Lorsque la synchronisation a lieu dan le cadre d'une intercommunication qui ne fait intervenir aucun protocole externe, on parle d'**auto-synchronisation**. L'auto-synchronisation joue un rôle important en cryptographie, plus précisément en cryptographie symétrique avec l'utilisation des chiffreurs dits auto-synchronisants ou Self-Synchronizing Stream Ciphers (SSSC) en anglais [76]. De tels chiffreurs peuvent être conçus à partir de générateurs obtenus en utilisant des systèmes dynamiques. Ces systèmes dynamiques, dans le contexte de la cryptographie, opèrent sur des corps finis tout en générant des séquences qui sont assez complexes pour brouiller des informations qui doivent être transmises dans le secret. Ces séquences générées doivent toutefois être auto-synchronisantes afin de garantir correctement le déchiffrement de l'information transmise.

Dans cette thèse on s'intéresse à l'utilisation des SSSCs en cryptographie d'un point de vue de la théorie du contrôle en automatique. Le principal résultat réside dans le concept de la *platitude* qui combiné à des notions utilisées en théorie du graphe offre une approche constructive et systématique de classes plus générales de SSSCs.

Le Chapître 1 de la thèse aborde donc les notions et concepts de la théorie du contrôle qui sont utiles pour la construction des SSSCs. Il illustre l'intérêt potentiel des systèmes dynamiques LPV plats pour répondre à des problématiques en cryptographie. En effet ces systèmes LPV sont des systèmes linéaires dont la représentation d'état dépend d'un vecteur de paramètres qui varient dans le temps. Ils sont utilisés depuis bien des années en contrôle [2, 40, 69, 70, 83, 99] et dans le domaine de l'observation ou filtrage [6, 52, 98, 106]. Les modèles LPV peuvent être par exemple utilisés pour construire des contrôleurs programmables utilisés par exemple dans le contrôle de l'espace aérien [95] ou le contrôle de véhicules [101]. En outres, ces modèles LPV peuvent représenter des systèmes non linéaires sous certaines conditions bien définies. Ceci a

fait l'objet d'études dans [19] ou dans [79].

Un système linéaire à paramètre variant à une seule entrée et une seule sortie ou Single Input Single Output (SISO) en anglais noté Σ_ρ et défini sur un corps fini $\mathbb{F}$ est décrit par la représentation d'espace d'état suivant:

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

où $k \in \mathbb{N}$ correspond au temps discret, $x_k \in \mathbb{F}^n$ correspond au vecteur d'état, $u_k \in \mathbb{F}$ correspond à l'entrée, $y_k \in \mathbb{F}$ correspond à la sortie. Les matrices $A \in \mathbb{F}, B \in \mathbb{F}^{n\times 1}$, $C \in \mathbb{F}^{1\times n}$ et $D \in \mathbb{F}^{1\times 1}$ correspondent respectivement à la matrice dynamique, la matrice d'entrée, la matrice de sortie et la matrice de transfert direct. Comme on peut donc le voir dans l'Equation (1) caractéristique d'un système LPV, le vecteur d'état dépend linéairement de l'entrée et de la sortie du système. Cependant, les matrices A, B, C et D dépendent d'un vecteur de paramètres $\rho(k) = \left[\ \rho^1(k), \rho^2(k), ..., \rho^{L_\rho}(k)\ \right] \in \mathbb{F}^{L_\rho}$ qui confèrent une dynamique non linéaire au système. Ici L_ρ désigne le nombre total de coefficients (éventuellement variables) non nuls des matrices. En effet les paramètres variants $\rho^i(k)$ sont définis comme des fonctions non linéaires φ_i en la sortie y_k: $\rho^i(k) = \varphi_i(y_k, y_{k-1}, \cdots)$.

Une fois la définition des systèmes dynamiques LPV établie, on va établir ensuite une correspondance avec les SSSCs à travers la notion de platitude que l'on définit ci-après. On définit avant tout, pour tout entier positif k_0, une réalisation que l'on notera par ρ, toute séquence $\{\rho(k_0), \rho(k_0+1), \ldots\}$ de paramètres à temps variant. La définition d'un système LPV est donc donnée d'un point de vue générique i.e pour presque toute réalisation ρ.

Définition 1 *On dit que le système (1) est génériquement plat, si pour presque toute réalisation ρ, il existe une variable y_k, appelée sortie plate, telle qu'on puisse exprimer toutes variables du système comme fonction des itérés antérieurs et ultérieurs de cette sortie plate. En d'autres termes, il existe deux fonctions $\mathcal{F}_\rho$ et $\mathcal{G}_\rho$ paramétrisées par ρ telles que:*

$$\begin{cases} x_k &= \mathcal{F}_\rho\big(y_{k+k_\mathcal{F}}, \ldots, y_{k+k'_\mathcal{F}}\big) \\ u_k &= \mathcal{G}_\rho\big(y_{k+k_\mathcal{G}}, \ldots, y_{k+k'_\mathcal{G}}\big) \end{cases} \tag{2}$$

avec $k_\mathcal{F}$, $k'_\mathcal{F}$, $k_\mathcal{G}$ et $k'_\mathcal{G}$ des valeurs entières de $\mathbb{Z}$.

La platitude est cependant une notion importante qui admet de divers et variés domaines d'application comme dans la planification de trajectoire [25, 78], en contrôle prédictif ou encore dans la gestion de contraintes [32, 43, 60]. Dans cette thèse, on va s'intéresser, au-delà d'une simple vérification des sorties plates d'un système LPV, à la construction des systèmes LPV qui admettent des sorties plates. Ceci revient donc d'après la définition 1, à construire des systèmes LPV dont les variables s'expriment comme fonction de la sortie.

Il existe plusieurs approches qui permettent de caractériser les sorties plates d'un système LPV. On peut citer:

l'approche directe: elle consiste à exprimer par itérations successives les variables du système comme fonction de la sortie. Il en résulte cependant que cette approche est onéreuse même pour des systèmes de petite dimension et devient presque impossible à adopter pour des systèmes de grande dimension.

l'approche basée sur le système inverse: il s'agit d'une approche qui utilise, lorsqu'il existe, le *système inverse à gauche* du système LPV (1). Ce système inverse admet dans ce cas une matrice dynamique notée P et qui s'exprime en fonction des matrices A, B, C et D du système LPV.

Une caratérisation de la sortie plate basée sur le système inverse a été proposé pour les systèmes linéaires switchés dans [80]. L'extension de cette caractérisation aux systèmes LPV est immédiate, en considérant la fonction de commutation comme une fonction à valeurs dans un ensemble indéterminé au lieu d'un ensemble fini qui est appelé modes. Cette caractérisation est basée sur la notion de degré relatif d'un système dynamique [89] que l'on rappelle ci-après:

Définition 2 *Le degré relatif d'un système dynamique est le nombre minimal r d'itérations à partir duquel la sortie y_{k+r} à l'instant $k+r$ est influencée par l'entrée u_k.*

Dans le cas d'un système LPV (1), on montre que si le système admet un degré relatif r alors, pour toute réalisation ρ, la sortie s'exprime par:

$$y_{k+r} = C_{\rho(k+r)} \prod_{l=k+r-1}^{k} A_{\rho(l)} x_k + \mathcal{T}^{r,0}_{\rho(k)} u_k \tag{3}$$

où $\mathcal{T}^{r,0}_{\rho(k)}$ est une quantité définie suivant les valeurs de r par:

1. si $r = 0$: $\mathcal{T}^{0,0}_{\rho(k)} \neq 0$ pour tout entier relatif k
2. sinon si $r < \infty$, alors r est le plus petit entier s tel que:

$$\begin{array}{l} \mathcal{T}^{i,j}_{\rho(k)} = 0 \quad \text{pour } i = 0, \ldots, s-1 \text{ et } j = 0, \ldots, \ i, \\ \mathcal{T}^{s,0}_{\rho(k)} \neq 0 \end{array} \tag{4}$$

pour tout entier relatif $k \geq 0$ et pour toute réalisation ρ.

On a donc le théorème suivant qui permet de caractériser la sortie plate d'un système LPV (1).

Théorème 1 ([34]) *Si le système LPV (1) admet un degré relatif r, alors y_k est une sortie plate si et seulement si il existe un entier positif K tel que pour tout entier relatif*

$k \geq 0$ et pour toute séquence $\{\rho(k), \ldots, \rho(k+K-1+r)\} \in \Theta^{r+K}$, la condition suivante est vérifiée:

$$P_{\rho(k+K-1:k+K-1+r)} P_{\rho(k+K-2:k+K-2+r)} \cdots P_{\rho(k:k+r)} = 0 \tag{5}$$

avec

$$P_{\rho(k:k+r)} = A_{\rho(k)} - B_{\rho(k)} (\mathcal{T}^{r,0}_{\rho(k)})^{-1} C_{\rho(k+r)} \prod_{l=k+r-1}^{k} A_{\rho(l)} \tag{6}$$

La preuve de ce théorème est constructive et permet en outres de montrer que l'état interne x_k ainsi que l'entrée u_k du système LPV (1) s'expriment en fonction des itérés de la sortie y_k par:

$$x_k = \sum_{i=0}^{K-1} \left[\prod_{j=1}^{i} P_{\rho(k-j:k-j+r)} \right] (\mathcal{T}^{r,0}_{\rho(k-1-i)})^{-1} B_{\rho(k-1-i)} y_{k-1-i+r} \tag{7}$$

et

$$\begin{aligned} u_k \quad = \quad & (\mathcal{T}^{r,0}_{\rho(k)})^{-1} y_{k+r} - (\mathcal{T}^{r,0}_{\rho(k)})^{-1} C_{\rho(k+r)} \prod_{l=k+r-1}^{k} A_{\rho(l)} \\ & \cdot \sum_{i=0}^{K-1} \left[\prod_{j=0}^{i-1} P_{\rho(k-j-1:k-j-1+r)} \right] (\mathcal{T}^{r,0}_{\rho(k-1-i)})^{-1} B_{\rho(k-1-i)} y_{k-1-i+r} \end{aligned} \tag{8}$$

L'intérêt du Théorème 1 est qu'il offre une expression explicite des fonctions $\mathcal{F}_\rho$ et $\mathcal{G}_\rho$ lorsque y_k est une sortie plate. Cependant vérifier que y_k est une sortie plate exigerait de vérifier le produit 5 pour un nombre infini de réalisations ρ; ce qui ne peut être réalisé dans la pratique. Pour cela, il sera proposé une approche basée sur les graphes orientés que l'on décrira un peu plus bas.

Problème de la mortalité: il est à noter que l'égalité de l'Equation (5) n'est pas nécessairement vérifiée pour tout produit de matrice $P_{\rho(k:k+r)}$ mais seulement pour un nombre de séquences admissibles de telles matrices. Cette égalité est connue de façon plus générale sous le nom de *problème de mortalité* qui est un problème indécidable ([12, 17, 90]). Il a cependant été proposé dans [89] une approche basée sur les semi-groupes nilpotents de matrices vérifiant l'égalité (5).

Définition 3 (Semi-groupe nilpotent) *Un semi-groupe $\mathcal{S}$ est un ensemble avec une loi multiplicative associative interne. On dit qu'un semi-groupe $\mathcal{S}$ qui admet une élément absorbant 0 est nilpotent s'il existe un entier $t \in \mathbb{N}^*$ tel que tout produit de t éléments appartenant à $\mathcal{S}$ est nul. Le plus petit entier t est appelé dans ce cas la classe de nilpotence de $\mathcal{S}$.*

Cette notion de semi-groupe nilpotent est intrinsèquement liée à la notion de *triangularisation simultanée*. En effet le Théorème de Levitzky [92] dit qu'un ensemble de matrices forme un semi-groupe nilpotent si et seulement si cet ensemble est simultanément triangularisable autrement dit si et seulement si cet ensemble admet une base commune de triangularisation.

On a donc le théorème suivant qui caractérise une sortie plate d'un système LPV.

Théorème 2 ([89]) *Si la matrice $P_{\rho(k:k+r)}$ est triangularisable indépendemment de ρ, alors y_k est une sortie plate du système (1).*

Cette dernière approche offre une complexité polynômiale, lorsque l'ensemble de matrices est donné, pour vérifier si une sortie du système est plate ou non. L'inconvénient est qu'elle n'est pas appropriée pour répondre à notre problématique à savoir la construction de matrices qui doivent vérifier le Théorème 2.

Pour toutes les raisons énumérées ci-dessus, il sera proposé une approche complémentaire basée sur les graphes orientés qui fournit des conditions nécessaires et suffisantes afin de caractériser les sorties plates d'un système LPV et plus encore qui fournit un moyen de construction systématique de systèmes LPV qui admettent une sortie plate.

Approche basée sur les graphes orientés: Cette approche considère le système linéaire structuré associé au système LPV (1) et qui est décrit par le système d'équations:

$$\Sigma_\Lambda : x_{k+1} = I_A x_k + I_B u_k \tag{9}$$

Ce système structuré exprime une forme simplifiée du système LPV. Ceci permet de traduire les variables de ce dernier en termes de sommets d'un graphe orienté noté par $\mathcal{G}(\Sigma_\Lambda)$. Les arcs de ce graphe décrivant la dynamique du système LPV. Ainsi aux matrices $A_{\rho(k)}$ et $B_{\rho(k)}$ du système LPV (1), on associe les matrices I_A et I_B du système structuré (9) dont les coefficients sont des $'0'$ et des $'1'$; les variables non nulles des matrices du système LPV étant remplacées par des coefficients $'1'$ dans les matrices du système structuré.

Si on considère l'exemple suivant de matrices d'un système LPV

$$A_{\rho(k)} = \begin{pmatrix} 1 & \rho^1(k) \\ 0 & \rho^2(k) \end{pmatrix}, B_{\rho(k)} = \begin{pmatrix} 0 \\ \rho^3(k) \end{pmatrix} C_{\rho(k)} = \begin{pmatrix} 1 & 0 \end{pmatrix} D_{\rho(k)} = 0. \tag{10}$$

alors les matrices du système structuré sont donnés par

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

Le graphe orienté associé au système est donné par la Figure 1.

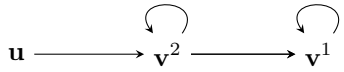

Figure 1: Graphe orienté du système linéaire structuré associé au système LPV (10).

On note par $\mathbf{v^F}$ et $\mathbf{u}$ les sommets du graphe orienté $\mathcal{G}(\Sigma_\Lambda)$ qui sont associés respectivement à la sortie (supposée plate) et l'entrée du système LPV (1); et par $\mathbf{X}$ l'ensemble des sommets de $\mathcal{G}(\Sigma_\Lambda)$ associés aux composantes de l'état interne x du système LPV. A partir de ce graphe orienté associé au système LPV (1), on établit trois conditions nécessaires et suffisantes données par le théorème suivant:

Théorème 3 *[16] Considérons le système linéaire structuré Σ_Λ décrit par (9). La sortie y^F associée aux sommets $\mathbf{v^F} \in \mathbf{X} \cup \{\mathbf{u}\}$ est génériquement une sortie plate si et seulement si les trois conditions suivantes sont vérifiées dans le graphe $\mathcal{G}(\Sigma_\Lambda)$ associé:*
C0. *$\mathbf{v^F}$ est un successeur de $\mathbf{u}$ autrement dit, il existe un chemin dans le graphe $\mathcal{G}(\Sigma_\Lambda)$ qui relie l'entrée à la sortie;*
C1. *tous les chemins simples[1] $\{\mathbf{u}\}$-$\{\mathbf{v^F}\}$ reliant les deux sommets, associés à l'entrée et à la sortie, sont de même longueur $\ell(\mathbf{u}, \mathbf{v^F})$;*
C2. *tous les cycles du graphe couvrent au moins un élément de l'ensemble des sommets essentiels[2] $V_{ess}(\{\mathbf{u}\}, \{\mathbf{v^F}\})$.*

Ainsi l'approche graphe-orienté, à travers ces trois conditions, permet de caractériser de façon plus aisée avec une complexité polynômiale les sorties plates du système LVP (1). Et en se basant sur ces conditions, on peut aisément construire des schémas ou structures de graphe qui donnent des systèmes LPV plats. Ce que ne permettait pas l'approche algébrique (Théorème 1). De plus l'approche graphe-orienté fournit explicitement les valeurs du degré relatif (Définition 2) et de l'entier K du Théorème 1. Ces valeurs sont données par les deux propositions suivantes:

Proposition 1 ([34]) *Le degré relatif r du système (1) est égal à $\ell(\mathbf{u}, \mathbf{v^F})$ dans le graphe orienté $\mathcal{G}(\Sigma_\Lambda)$.*

Proposition 2 ([34]) *Si y_k^F est une sortie plate, alors l'entier K du Théorème 1 est génériquement égal à la longueur maximale de tous les chemins simples $\{\mathbf{u}\}$-$\mathbf{X}$ du graphe orienté $\mathcal{G}(\Sigma_\Lambda)$.*

En combinant donc les deux approches algébrique et graphe-orienté, on obtient une caractérisation complète des sorties plates du système LPV (1) avec des expressions explicites des variables x_k et u_k comme données par les Equations (7) et (8).

[1] un chemin simple est un chemin qui passe une et une seule fois par les sommets qu'il couvre.
[2] un sommet essentiel de l'entrée et de la sortie est un chemin qui appartient à l'intersection de tous les chemins reliant l'entrée à la sortie.

Nous sommes donc en mesure, grâce aux idées qui ont été développées jusqu'ici, d'établir le lien entre les SSSCs et les systèmes dynamiques LPV plats tout en proposant une construction systématique des SSSCs. C'est ce qui fera l'objet du Chapitre 2 de la thèse.

Rappelons avant tout que les SSSCs sont utilisés en cryptographie symétrique comme chiffreurs à flot. De tels chiffreurs sont utilisés dans la transmission de données de façon sécurisée à un débit trés élevé et via des appareils électroniques qui sont limités en ressources matérielles. On distingue deux classes principales de chiffreurs à flots:

les chiffreurs dit synchrones (Synchronous Stream Ciphers ou SSC en anglais)

et les chiffreurs auto-synchronisants (SSSC) qui font donc l'objet d'étude de cette thèse.

L'avantage majeur de ces derniers par rapport aux premiers est qu'ils ne nécessitent aucun protocole supplémentaire de resynchronisation entre les entités qui échangent des données sécurisées. L'utilisation des SSSCs est donc d'un intérêt crucial pour l'échange de communication sécurisée en groupe et permet par exemple à toute entité autorisée de s'insérer dans une communication en cours sans qu'elle n'ait nullement besoin d'initialiser son dispositif de communication avec les autres membres du groupe.

Les SSSCs sont décrits d'une façon générale à travers une forme dite canonique illustrée par la Figure 2.

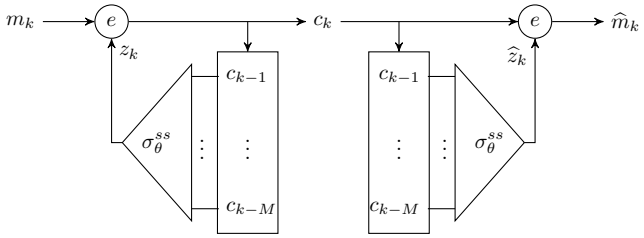

Figure 2: Forme canonique d'un SSSC pour $l = 1$.

Les équations de chiffrement et de déchiffrement sont données par les Equations (11) et (12) suivantes:

$$\begin{cases} z_k = & \sigma_\theta^{ss}(c_{k-l-M+1}, \ldots, c_{k-l}) \\ c_k = & e(z_k, m_k) \end{cases} \tag{11}$$

$$\begin{cases} \widehat{z}_k = & \sigma_\theta^{ss}(c_{k-l-M+1}, \ldots, c_{k-l}) \\ \widehat{m}_k = & e(z_k, c_k) \end{cases} \tag{12}$$

où θ désigne un paramètre secret partagé par l'émetteur et le récepteur des messages chiffrés échangés, σ_θ^{ss} est appelée *fonction de filtrage* et génère la suite chiffrante notée z_k côté chiffreur et $\widehat{z}_k$ côté déchiffreur. Ces suites chiffrantes dépendent des M précédents chiffrés c_k. On voit donc qu'il suffit au déchiffreur de recevoir les M symboles chiffrés précédant c_k et transmis sans erreur par le chiffreur pour pouvoir déchiffrer correctement c_k.

Les études sur les SSSCs ont été initiées au début des années 90 par les travaux de Maurer dans [74]. Dans la forme canonique des SSSCs, toute la complexité du schéma réside dans la

fonction de filtrage σ_θ^{ss} ce qui est n'est pas évident à concevoir. Pour remédier à cela, Maurer a proposé une construction de SSSCs basée sur des *automates à mémoire finie* (illustrée d'une façon généralisée par la Figure 3), qui de façon récursive produira la même séquence de suite chiffrante que la forme canonique.

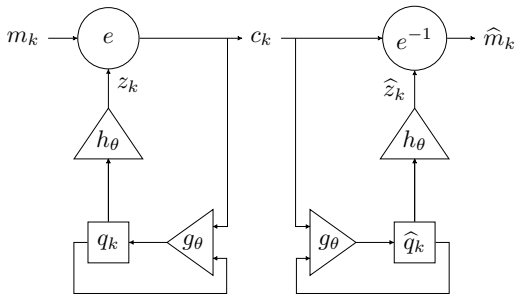

Figure 3: Architecture d'un SSSC basée sur un automate à mémoire finie.

L'automate à mémoire finie est caractérisé par le système d'équations suivant:

$$\begin{cases} q_{k+1} = g_\theta(q_k, c_{k+b}) \\ z_{k+b} = h_\theta(q_k) \end{cases} \tag{13}$$

où g_θ correspond à la fonction de transition d'état de l'automate, q_k désigne l'état interne et z_k la suite chiffrante générée par la fonction de filtrage h_θ.

Suite aux travaux de Maurer, quelques constructions ont été proposées [29, 31, 51, 96] mais ont été toutes cassées par des travaux de cryptanalyse dans [30, 56, 57, 58, 62]. Il y a cependant un point commun à toutes ces constructions, c'est que la fonction de transition g_θ dans (13) est une fonction dite *triangulaire* ou T-function [63] en anglais.

Un des objectifs de la thèse est donc de proposer une classe plus générale de fonctions de transition et qui ne sont pas nécessairement triangulaires. Les résultats sur la platitude des systèmes LPV, notamment à travers les Equations (7) et (8) nous permettent d'ores et déjà d'établir le lien entre les SSSCs et les systèmes LPV plats. On montre qu'on peut utiliser le système LPV direct (1) comme chiffreur d'un SSSC et le système inverse du système LPV comme déchiffreur. Ceci est traduit par la proposition suivante:

Proposition 3 ([36]) *Si le système LPV (1) admet un degré relatif r et est plat, alors l'automate à mémoire finie donnée par les systèmes d'équations suivants défini un SSSC:*

$$\begin{cases} q_{k+1} &= P_{\rho(k:k+r)} q_k + B_{\rho(k)} (\mathfrak{T}^{r,0}_{\rho(k)})^{-1} y_{k+r} \\ z_{k+r} &= C_{\rho(k+r)} \displaystyle\prod_{l=k+r-1}^{k} A_{\rho(l)} q_k \end{cases} \qquad \begin{cases} \widehat{q}_{k+1} &= P_{\rho(k:k+r)} \widehat{q}_k + B_{\rho(k)} (\mathfrak{T}^{r,0}_{\rho(k)})^{-1} y_{k+r} \\ \widehat{z}_{k+r} &= C_{\rho(k+r)} \displaystyle\prod_{l=k+r-1}^{k} A_{\rho(l)} \widehat{q}_k \end{cases}$$

En conclusion pour construire notre automate à mémoire finie qui peut être utilisé comme un SSSC, on génère un graphe-orienté à partir des Conditions **C0**–**C2** du Théorème 1. Ensuite

on extrait de ce graphe un système LPV plat et son système inverse. Ces systèmes correpondent à l'automate donné par la Proposition 3.

Dans la seconde partie de cette thèse, on s'intéresse à la sécurité d'algorithme auto-synchronisant proposé dans la thèse. Dans un premier temps on étudie dans le Chapitre 3, la sécurité physique de l'algorithme en réalisant une attaque par canaux cachés plus précisément une analyse par consommation de courant sur un dispositif qui contient l'algorithme. Cette analyse porte le nom de Correlation Power Analysis ou CPA en anglais et fut proposée dans [18]. L'originalité des travaux de ce Chapitre 3, c'est qu'on propose ici une approche spectrale pour réaliser une attaque CPA, approche qui est basée sur la transformée de Fourier. En effet l'objectif d'une attaque CPA est de retrouver des paramètres secrets de l'algorithme en établissant une corrélation entre d'une part la consommation réelle de courant du dispositif cryptographique attaqué et d'autre part une consommation hypothétique obtenue à partir d'un modèle d'attaque. Et plus la corrélation est élevée, plus les valeurs de paramètres secrets retrouvés sont fiables.

Les fonctions décrivant la consommation de courant sont ici des fonctions définies sur un ensemble de mots binaires à valeurs réelles et notée par φ:

$$\varphi : \{0,1\}^n \to \mathbb{R}$$

Définition 4 (Transformée de Fourier) *La transformée de Fourier de la fonction φ la fonction définie sur $\{0,1\}^n$ par:*

$$u \mapsto \widehat{\varphi}(u) = \langle \varphi, \chi_u \rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} \varphi(x)(-1)^{u \cdot x}.$$

On montre donc que cette transformée de Fourier conserve la norme de la fonction φ notée $\|\varphi\|$. Autrement dit $\|\varphi\| = \|\widehat{\varphi}\|$. Cette remarque sur la conservation de la norme est très utile et constitue l'idée centrale qui permet de proposer une approche spectrale de la CPA en appliquant la transformée de Fourier à la consommation réelle de courant φ et à la consommation hypothétique que l'on va notée par g.

On montre ensuite qu'en adoptant cette approche spectrale, on peut retrouver les paramètres secrets utilisés dans un algorithme cryptographique, notamment les sous clés qui sont utilisées comme entrée de fonctions non linéaires représentées par des boîtes S ou Subtitution Box (S-Box) en anglais. Ainsi on peut retrouver la valeur d'une sous clé utilisée comme entrée d'une boîte S pendant l'exécution d'un algorithme cryptographique en calculant la valeur qui maximise la fonction F suivante:

$$F(k) = \sum_{u \in \{0,1\}^n} \widehat{\varphi}(u)\widehat{g}(u)(-1)^{u \cdot k}. \tag{14}$$

Cette fonction n'est rien d'autre que la transformée de Fourier des transformées de Fourier des fonctions φ et g. Elle peut donc être calculée aisément à l'aide d'un algorithme de calcul de transformée de Fourier rapide ou Fast Fourier Transform algorithm en anglais (voir [22]). Ce qui offre une complexité de calcul quasi linéaire en la taille n des mots binaires traités, soit $n2^n$.

Les résultats expérimentaux ont été réalisés avec succès sur la boîte S de l'AES [1]. Cependant, les expériences réalisées sur l'algorithme SSSC basé sur un système LPV plat montre qu'il est difficile de retrouver les sous clés qui sont utilisées en entrée des boîtes S de l'algorithme. Cette difficulté étant liée à la façon dont sont effectuées les opérations qui font intervenir les boîtes S.

Dans un deuxième temps, on traite de la sécurité théorique des SSSCs. En effet une question naturelle qui émerge, suite aux cryptanalyses réalisées sur les SSSCs qui ont été construits, c'est de savoir si l'on peut toutefois concevoir des SSSCs qui atteignent un certain niveau de sécurité souhaité. Le Chapitre 4 apporte une réponse à cette question en considérant la forme canonique des SSSCs. On traite plus précisément des notions d'indistingabilité introduites dans [47, 48]. Il s'agit donc d'étudier d'une part la résistance du schéma contre des attaques à clairs choisis ou indistinguishability against Chosen Plaintext Attack (IND-CPA) en anglais et d'autre part la résistance contre des attaques à chiffrés choisis ou indistinguishability against Chosen Cipher Attack (IND-CCA) en anglais.

Pour étudier les résistances contre de telles attaques, on établit des jeux de sécurité [8]. Ces jeux de sécurité font intervenir un *adversaire* ou un *challenger* noté $\mathcal{A}$ qui défi un *oracle* noté $\mathcal{O}$ qui a accès à l'algorithme cryptographique et qui peut chiffrer ou déchiffrer des symboles clairs ou symboles chiffrés que lui soumet l'adversaire $\mathcal{A}$. Le but de ce dernier est de savoir lequel de deux symboles clairs (respectivement chiffrés) soumis à l'oracle $\mathcal{O}$ a été chiffré (respectivement déchiffré) dans le cas d'une attaque IND-CPA (respectivement IND-CCA). Si l'adversaire parvient à deviner le bon symbole chiffré ou déchiffré par l'oracle alors il gagne le jeux, sinon il perd.

On montre donc à travers ces jeux de sécurité, que dans le cas des attaques IND-CCA le schéma n'est pas résistant à cause de sa malléabilité [38].

Par contre le schéma est résistant contre les attaques IND-CPA [38]. Pour montrer ce dernier résultat on se base sur les notions de fonction pseudo aléatoires ou Pseudo Random Function (PRF) en anglais (voir [46, 8].) On montre que si la fonction de filtrage σ_θ^{ss} de la forme canonique est une PRF alors le schéma est résistant conre les attaque IND-CCA. On va plus long en relaxant cette condition et en introduisant une notion plus faible que celle des PRF: il s'agit de la notion Weak Pseudo Random Function (WPRF) en anglais, qui permet de mieux caractériser la propriété d'auto-synchronisation et qui donne cette fois-ci une condition nécessaire et suffisante sur la fonction de filtrage pour que le schéma soit résistant contre les attaques IND-CPA.

Le dernier chapitre, Chapitre 5 de la thèse, porte sur la représentation matricielle de fonc-

tions vectorielles Booléennes [22] qui sont utilisées comme fonctions de transition d'un SSSC. Ces représentations matricielles décrivent en fait une même fonction mais dans différentes bases. Les matrices représentatives sont ici les matrices de corrélation, de Walsh, d'adjacence qui sont de façon usuelle utilisées en cryptographie pour analyser les propriétés des fonctions vectorielles Booléennes. On introduit une représentation matricielle supplémementaire qui est la matrice polynômiale. Un des résulats fort de ce chapitre établit des relations entre les valeurs propres, les espaces propres ainsi qu'une structure de graphe associée des matrices. Pour arriver à ces résultats on utilise les notions de fonctions booléennes: la forme algébrique normale ou Algebraic Normal Form (ANF) en anglais [71]) et la forme numérique normale ou Numerical Normal Form (NNF) en anglais (voir [24]).

L'application de ces représentations matricielles aux SSSCs, notamment la matrice de corrélation, permet de dire si la fonction de transition d'un SSSC est une fonction triangulaire ou non. Ce résultat a été déjà établi dans la thèse de Parriaux [84]. On donne dans ce chapitre un exemple qui confirme le résultat pour les SSSCs basés sur les systèmes LPV plat.

Remerciements

Je souhaiterais avant tout, adresser mes sincères remerciements et gratitude à mes deux directeurs de thèse Gilles MILLERIOUX et Philippe GUILLOT. Ils m'ont donné une grande opportunité de réaliser cette thèse dans le cadre d'un projet ANR qui a regroupé plusieurs partenaires. Je leur suis très reconnaissant de la confiance qu'ils m'ont accordée sur un tel projet. Leurs idées, leur conseils et parfois leurs exigences m'ont permis d'accomplir avec succès cette thèse. Ils m'ont transmis les idées clés qui sont à l'origine de l'élaboration fructueuse des travaux de la thèse. Je remercie également le directeur du laboratoire CRAN, M. Didier WOLF par qui mon acceptation en tant que doctorant au sein du laboratoire a été possible.

Je tiens également à remercier les membres du jury de thèse qui ont accepté d'examiner mes travaux. Un grand merci à mes deux rapporteurs de thèse pour le grand intérêt qu'ils ont porté à mes travaux ainsi que l'attention particulière qu'ils ont porté à la lecture de mon manuscrit. Leur conseil et correction me sont d'une grande utilité pour la suite de ma carrière.

Je ne saurai oublier tous les autres membres du laboratoire, et plus précisément du département CID, Hugues GARNIER avec qui j'ai découvert les vraies activités sportives. Ses conseils, sa bonne humeur et sa joie de vivre m'ont été d'une grande utilité au cours de ces années passées au sein du laboratoire CRAN. Je tiens également à remercier Marion GILSON-BAGREL pour toutes les activités sportives qu'on a eu à réaliser entre *midi et deux*. Je n'oublie pas également tous ceux et toutes celles que j'ai eu à côtoyer pendant ces années de ma thèse à POLYTECH Nancy et avec qui j'ai vraiment eu à partager pleines de nouvelles connaissances, Floriane COLLIN, Vincent LAURAIN, Gérard BLOCH, Jean-Luc NOISETTE, Eric GNAEDINGER.

Tous mes remerciements vont également à mes collègues de bureau avec qui j'ai passé des moments fantastiques, Arturo, Yusuf, Valentin, Paul, Sabra, Asma, Dawid, Quentin, Carlo, Tatiana.

Je tiens également à remercier Nadia EL MRABET qui m'a donné l'opportunité de passer un séjour de deux semaines au Centre Micro-électronique de Provence sur le Campus Georges Charpak Provence de Gardanne afin de m'inititier aux attaques par canaux cachés. Cette initiation ne saurait aboutir sans l'aide précieuse de Julien FRANCQ à qui je tiens à témoigner toute ma gratitude et mes remerciements pour sa disponibilité, ses conseils ainsi que la formation qu'il m'a apportée au cours de cette thèse en plus de celle de mes deux directeurs de thèse. Merci également à Marc MOUFFRON qui avec Julien FRANCQ m'ont permis de visiter le site de Airbus Defence and Space situé à Elancourt dans le cadre de réunions de travail.

J'aimerais également remercier les membres du personnel administratif qui m'ont permis de réaliser dans d'excellentes conditions tous les déplacements que j'ai effectués ainsi que les démarches administaratives au sein du laboratoire au cours de ma thèse, la très aimable et sympathique Christelle KONDRATOW-LHOSTE (merci pour tout), Carole COURRIER, Monique BILON, Séverine HEISSLER.

Un grand merci également à mes amis qui ont été présents et m'ont apporté leur soutien pendant la thèse en particulier, Eric, Ginette, Amen, Rico, Têtê, Hervé.

Finalement j'adresse mes profonds remerciements à ma famille qui a fait preuve d'un soutien sans faille durant ces années de thèse. Un sincère remerciement à ma mère et à mon père qui m'ont beaucoup encouragé et aidé.

Publications

The publications related to this work are listed below:

International Conference papers

- B. DRAVIE, P. GUILLOT and G. MILLERIOUX. **Security Proof of the Canonical Form of Self-Synchronizing Stream Cipher**. In *proceeding of the ninth Workshop on Coding and Cryptography (WCC'15)*, April 2015. Paris, France.
- P. GUILLOT, G. MILLERIOUX, B. DRAVIE, and N. EL MRABET. **Spectral Approach for Correlation Power Analysis**, In *proceeding of the Second International Conference of Codes, Cryptology and Information Security (C2SI2017)*, April 10 -12, 2017. Rabat, Morocco.
- B. DRAVIE, T. BOUKHOBZA and G. MILLERIOUX. **A mixed algebraic/graph-oriented approach for flatness of SISO LPV discrete-time systems**, In *proceeding of the 2017 American Control Conference (ACC 2017)*, May 24 – 26, 2017. Seattle, WA, USA.
- B. DRAVIE, P. GUILLOT and G. MILLERIOUX. **Design of Self-Synchronizing Stream Ciphers: a New Control-Theoretical Paradigm**. In *proceeding of the 20th IFAC World Congress (IFAC2017)*, July 9 –14 2017, Toulouse, France.

National Conference paper

- B. DRAVIE, P. GUILLOT and G. MILLERIOUX. **Construction d'un chiffreur auto-synchronisant basée sur la propriété de platitude**, *12e édition Journées Codage et Cryptographie (JC2'15)*, October 2015. La Londe-les-Maures, France.

Journal papers

- B. DRAVIE, J. PARRIAUX, P. GUILLOT and G. MILLERIOUX. **Matrix representations of vectorial Boolean functions and eigenanalysis**. In *Cryptography and Communications*(October 2016) Vol.8, Issue 4, p 555–577. Springer US. doi: 10.1007/s12095-015-0160-7.
- B. DRAVIE, P. GUILLOT and G. MILLERIOUX. **Security proof of the canonical form of self-synchronizing stream ciphers**. In *Designs, Codes and Cryptography*(2016) Springer US. doi:10.1007/s10623-016-0185-8.

Contents

List of Figures

LIST OF FIGURES

Acronyms

LPV	Linear Parameter Varying
SISO	Single Input Single Output
SSC	Synchronous Stream Cipher
SSSC	Self-Synchronizing Stream Cipher
PRF	Pseudo Random Function
WPRF	Weak Pseudo Random Function
LOR-IND	left-or-right indistinguishability
IND-CPA	Indistinguishability against Chosen Plaintext Attack
IND-CCA	Indistinguishability against Chosen Ciphertext Attack
DES	Data Encryption Standard
AES	Advanced Encryption Standard
CMOS	Complementary Metal Oxide Semiconductor
FET	Field Effect Transistors
CPA	Correlation Power Analysis

Notations

$\mathbb{F}$	finite field.
n	dimension of the system.
x	state-space vector of a dynamical system.
x_k	state vector at time k
x_k^i	state vector component number i at time k
u_k	input of a dynamical system.
y_k	output of a dynamical system.
$\widehat{x}$	state-space vector of the left inverse dynamical system.
$\widehat{x}_k$	state vector at time k of the left inverse dynamical system.
$\widehat{u}_k$	output of the left inverse dynamical system.
$\rho(k)$	time varying parameter vector.
$\rho^i(k)$	time varying parameter component number i.
ρ	realization $(\rho(k), \rho(k+1), \dots)$ in a finite or infinite time window of $\rho(k)$.
Σ_ρ	LPV discrete-time system.
r	relative degree of a system.
A_ρ	dynamical matrix of Σ_ρ.
B_ρ	input matrix of Σ_ρ.
C_ρ	output matrix of Σ_ρ.
D_ρ	direct transfer matrix of Σ_ρ.
P_ρ	inverse transition matrix of Σ_ρ.
ε_k	error $\widehat{x}_k - x_k$.
σ_θ^{ss}	keystream generator of an automaton
e	encryption function.
d	decryption function.
$\mathcal{M}$	message space.
$\mathcal{C}$	ciphertext space.
m_k	plaintext (input) of a cipher.
c_k	ciphertext (output) of a cipher.
q_k	internal state of a cipher.

z_k	keystream of a cipher.
$\widehat{m}_k$	recovered plaintext (output) of a decipher.
$\widehat{q}_k$	internal state of a decipher.
$\widehat{z}_k$	keystream of a decipher.
g_θ	next state transition function.
h_θ	filter function of the internal state.
M	synchronization delay.
$\mathrm{A}dv_{\mathrm{p}rf}(\mathcal{B})$	advantage of an adversary $\mathcal{B}$ in a PRF-game.
$\mathrm{A}dv_{\mathrm{w}prf}(\mathcal{B})$	advantage of an adversary $\mathcal{B}$ in a WPRF-game.
$\mathrm{Pr}_{\mathrm{prf}}(\mathcal{B} = \mathrm{prf})$	probability that $\mathcal{B}$ answers prf in a pseudo-random world.
$\mathrm{Pr}_{\mathrm{rf}}(\mathcal{B} = \mathrm{prf})$	probability that $\mathcal{B}$ answers prf in a random world.
$\mathrm{A}dv(\mathcal{A})$	advantage of an adversary $\mathcal{A}$ in a IND-game.
Σ_Λ	structured linear discrete-time system.
$\mathcal{G}$	digraph.
$\mathcal{G}(\Sigma_\Lambda)$	digraph associated to Σ_Λ.
$\mathcal{G}(\Sigma_\rho)$	digraph associated to Σ_ρ.
$\mathcal{V}$	set of state vertices in a digraph.
$\mathcal{E}$	set of edges in a digraph.
$\mathbf{v^i}$	the vertex number i of a digraph.
n_M	maximal number of edges of a digraph.

General Introduction

Synchronization has been an important topic in automatic control for years. Roughly speaking, by synchronization, it is meant correlated (according to given criteria) behaviors of at least two or more interconnected entities in virtual or physical networks. Throughout the past centuries, scientists have attempted to explain the emergence of order through the concept of synchronization. C. Huygens in 1665 can be considered as a pioneer. There is an outstanding number of examples of synchronized phenomena (see [104]) borrowed from nature, biology, neuroscience, physiology and more recently social networks.

Synchronization can be a very efficient way of tackling engineering issues as well. For example, there is a growing interest in cooperative control problems. Such problems involve several autonomous entities (also called agents) which try to collectively reach a global objective by a suitable connectivity or couplings. The related applications are mobile robots, unmanned and autonomous vehicles, satellites, air traffic control. Synchronization is also central in communication: video, broadcasting, Phase Lock Loop-based equipments. When synchronization must occur in a peer-to-peer communication setup without any external control, it is called self-synchronization. It turns out that self-synchronization is central in cryptography, more specifically, in symmetric cryptography involving the so-called Self-Synchronizing Stream Ciphers (SSSC) (see [76]). Such ciphers are based on generators which can take the form of dynamical systems operating on finite fields and must deliver complex sequences. Those sequences are used to scramble the information to be safely transmitted. For proper decryption, those sequences must be self-synchronized.

In this thesis, we aim at addressing SSSC-based cryptography in terms of control theoretical concepts. The main result is that the concept of flatness together with graph-theory allows to provide a convenient and systematic way to construct general classes of SSSC, the ciphers being viewed as dynamical systems. The flatness (differential or difference) on one hand and the SSSC on the other hand are detailed a little bit more below.

Differential flatness is a property of some continuous-time controlled dynamical systems introduced in [42]. The counterpart for discrete-time systems is called difference flatness. For a flat discrete-time system, the state variables as well as the input are written as a function of the flat output (including forward and backward shifts in the output). Difference flatness has been

first reported in [43, 103] although we should mention that a closely related notion, namely the dynamic feedback linearization, was addressed earlier in [3]. Flatness-based control has been involved in many applications and has an outstanding interest in, for instance, trajectory planning as reported in [78, 25], predictive control and constraint handling as detailed in [32, 60]. The reader can also refer to the book [103] for further applications. Since in this thesis, only discrete-time finite state dynamical systems are considered, the word flatness will be often used for short without confusion. Chapter 1 will be devoted to characterization of flat output of LPV systems. It will show how graph approach can be used to achieve such a characterization.

Self-Synchronizing Stream Ciphers were patented in 1946. This self-synchronization property has many advantages and is especially relevant to group communications. Since 1960, specific SSSC have been designed and are still used to provide bulk encryption (for Hertzian line, RNIS link, ...) in military applications or governmental radio mobile networks. In the early $90'$s, studies have been performed in [74, 27] to propose secure designs of SSSC. These works have been followed by effective constructions ([27, 96, 31]), but till now, all of these SSSC have been broken ([57, 58, 56, 61, 62]) which motivates the search for new constructions of SSSC.

It turns out that, all the constructions of SSSC that have been proposed in the literature are based on T-function (for Triangular function, see [63]). In [81], it has been shown that a new architecture of SSSC not restricted to T-functions can be designed using concepts and results borrowed from Control Theory. In Chapter 2, it is shown that the consideration of the special class of flat Linear Parameter Varying (LPV) systems, with the help of graph theory, allows for a convenient and systematic design of SSSC admitting not exclusively T-functions.

The second part of the work is devoted to security. Regarding security, SSSC have intrinsic interesting properties due to the specific architecture. For example, as each plaintext symbol influences potentially all subsequent ciphertexts, they naturally have good diffusion properties and are efficient against attacks based on plaintext redundancy. Furthermore, they can prevent from traffic analysis as the information which is conveyed through the channel is encrypted whether there is traffic or not. In the middle of the $90's$, a new kind of attack on cryptosystems, called side-channel attack, has been developed with a pioneering work of P. Kocher [66]. Side-channel attacks are attacks that exploit leakages of devices running a cryptographic algorithm. These leakages can result from physical emanations such as power consumption, electromagnetic radiations, temperature or execution time when running the algorithm. This is due to Substitution Box (S-Box) used as nonlinear functions within a cryptographic algorithm and that are also implemented in SSSC algorithms. Indeed, S-Box are highly sensitive to power consumption and electromagnetic leakages. Although a cryptographic algorithm can be proved secure theoretically, it is not ensured that the algorithm does not reveal secret information when it is implemented in a device. Common side channel attacks are Simple Power Analysis (SPA) and Differential Power Analysis (DPA). The aim of Chapter 3 is to propose a new and simple approach to perform a Correlation Power Analysis (CPA) which is an advanced form of

DPA. The approach is based on spectral theory, especially on Fourier transform of real valued function over the set of binary words. We use hereafter this approach to perform a CPA attack on the S-Box used within the designed LPV-based SSSC.

Considering theoretical security, we show in Chapter 4 that the canonical form of the Self-Synchronizing Stream Cipher is not resistant against chosen ciphertext attack (IND-CCA security) but can reach the resistance against chosen plaintext attack (IND-CPA security) provided that the filtering function is pseudo random. We introduced a new family of functions in order to characterize the security of the canonical SSSC from its filtering function. Those functions are called Weak Pseudo Random Functions (WPRF) and provide a weaker property than pseudo-randomness. A connection with the *left-or-right indistinguishability* (LOR-IND) is made. This property provides a necessary and sufficient condition to characterize the indistinguishablity of SSSC. The technical developments used to establish the security proof follow similar lines than those used when dealing with symmetric encryption schemes based on block ciphers.

Finally, the approach used in this thesis to design Self-Synchronizing Stream Cipher is a mixed algebraic and graph theory approach. However, it is interesting to note that we can make a general characterization of functions that can be used in the design of Self-Synchronizing Stream Ciphers. The work of the paper [85] tackled this problem by giving a characterization that is based on Walsh transform and matrix representation of vectorial boolean functions. The aim of Chapter 5 is then devoted to results that are useful to establish such characterizations. It gives a general and unified overview on matrix representations of vectorial boolean functions. These results are not only useful to characterize self-synchronizing property of next-state function from spectral approach but are also really interesting in application of boolean function to cryptography.

Chapter 1

LPV Systems and Characterization of Flatness

The aim of this chapter is to provide concepts from control theory that will be useful in the design of Self-Synchronizing Stream Ciphers. It illustrates the potential interest of LPV flat dynamical systems for cryptographic issues. The special class of LPV dynamical systems is presented. Then the definition of difference flatness is introduced and is particularized for LPV systems. First, necessary and sufficient conditions for an output to be difference flat are established. They are expressed in terms of algebraic conditions that must be verified by the state space matrices realization of the LPV system. It is shown how the combination of both the algebraic conditions and additional graph-based conditions allows to define a complete and tractable framework not only to find the flat outputs but also to explicit the functions of the state and the input with respect to the flat output. The results in this chapter are published in [34].

Contents

1.1 Linear Parameter Varying Systems

Linear Parameter Varying (LPV) systems are linear models whose state representation depends on a parameter vector which may vary in time. Since several years, more and more attention has been paid to these systems both in control [2] [83] [70] [99] [69] [40] and in observation or filtering [6] [52] [106] [98]. The interest of LPV models is that they provide a systematic procedure to design gain-scheduled controllers like in the area of aerospace control [95] or vehicle control [101]. Besides, LPV models can represent nonlinear systems under appropriate conditions. Such a purpose has been investigated in the works reported in [19] or in [79].

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

$$\Sigma_\rho : \left\{ \begin{array}{l} x_{k+1} = A_{\rho(k)} x_k + B_{\rho(k)} u_k \\ y_k = C_{\rho(k)} x_k + D_{\rho(k)} u_k \end{array} \right. \tag{1.1}$$

where $k \in \mathbb{N}$ stands for the discrete time, $x_k \in \mathbb{F}^n$ is the state vector, $u_k \in \mathbb{F}$ is the input, $y_k \in \mathbb{F}$ is the output. The matrices $A \in \mathbb{F}^{n\times n}$, $B \in \mathbb{F}^{n\times 1}$, $C \in \mathbb{F}^{1\times n}$ and $D \in \mathbb{F}^{1\times 1}$ are respectively the dynamical matrix, the input matrix, the output matrix and the direct transfer matrix. Such a system is called Linear Parameter-Varying because it is written with a linear dependency with respect to the state vector. The set of all the varying parameters of A, B, C and D are collected on a vector denoted by $\rho(k) = \left[\begin{array}{c} \rho^1(k), \rho^2(k), ..., \rho^{L_\rho}(k) \end{array} \right] \in \mathbb{F}^{L_\rho}$ where L_ρ is the total number of non zero (possibly varying) entries. The matrices A, B, C and D do not necessarily depend on all the parameters $\rho^i(k)$. Such systems can exhibit nonlinear dynamics. Indeed, the nonlinearity is obtained by defining the varying parameters $\rho^i(k)$ as nonlinear functions φ_i of the output y_k (or a finite number of shifts) $\rho^i(k) = \varphi_i(y_k, y_{k-1}, \cdots)$. Let us notice that the notation $\rho^i(k)$ (usual in the literature for LPV systems) is somehow abusive because it does not reflect an explicit dependency with respect to the time k but on quantities indexed with k. In the next section, we focus on the property of flatness for LPV systems.

1.2 Difference Flatness for LPV Systems

For a non negative integer k_0, a sequence $\{\rho(k_0), \rho(k_0+1), \ldots\}$ will be called a realization and denoted with ρ. The definition of a flat LPV system is given below in a generic sense, that is for almost every realization ρ.

Definition 1.2.1 *The system (1.1) is said to be generically flat if, for almost every realization ρ, there exists a variable y_k, called a flat output, such that all system variables can be expressed as a function of the flat outputs and a finite number of its backward and forward shifts. In other*

words, there exist two functions $\mathcal{F}_\rho$ *and* $\mathcal{G}_\rho$ *parametrized by* ρ *such that*

$$\begin{cases} x_k &= \mathcal{F}_\rho\big(y_{k+k_\mathcal{F}}, \ldots, y_{k+k'_\mathcal{F}}\big) \\ u_k &= \mathcal{G}_\rho\big(y_{k+k_\mathcal{G}}, \ldots, y_{k+k'_\mathcal{G}}\big) \end{cases} \tag{1.2}$$

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

Flatness is an important concept insofar as the applications of flatness (differential or difference) has a clear interest for trajectory planning [78, 25], predictive control or constraints handling [43, 32, 60] for example. Difference flatness has been recently addressed in [59][97][80] for nonlinear discrete-time systems. However, those works suffer from either poorly tractable conditions or only sufficient conditions. Although flatness is closely related to controllability, only proving controllability of a system may not be satisfactory in practice. Indeed, beyond merely checking whether a system is flat or not, we should be not only interested in more constructive issues like finding the flat outputs and finding the functions that explicit the dependence of the state variables as well as the input with respect to the flat output. Those functions are central since they provide a convenient way to reconstruct the state of a system or to provide a feedforward control.

For LPV systems, several approaches can be used to characterize a flat output. They are recalled below and their shortcomings are pointed out to motivate the graph-oriented approach as a relevant complementary method. The discussion is illustrated through Example 1.2.1. It corresponds to the particular state space matrices setting of (1.1) given below.

Example 1.2.1

$$\begin{aligned} A_{\rho(k)} &= \begin{pmatrix} 1 & \rho^1(k) \\ 0 & \rho^2(k) \end{pmatrix}, \\ B_{\rho(k)} &= \begin{pmatrix} 0 \\ \rho^3(k) \end{pmatrix}, \\ C_{\rho(k)} &= \begin{pmatrix} 1 & 0 \end{pmatrix}, \\ D_{\rho(k)} &= 0. \end{aligned} \tag{1.3}$$

The state component x^1_k *is considered as the measured output, that is* $y_k = x^1_k$.

1.2.1 Direct approach

This approach consists in trying to directly agree with the definition, that is attempting to express the input and the state vector as a function involving exclusively shifts of the output.

For this example, y_k is a flat output. Indeed, Definition 1.2.1 is satisfied with (1.2) which reads

$$\left\{\begin{array}{rcl} x_k^1 &=& y_k \\ x_k^2 &=& (\rho^1(k))^{-1}(y_{k+1}-y_k) \\ u_k &=& (\rho^3(k))^{-1}\big((\rho^1(k+1))^{-1}y_{k+2} \\ & & -(\rho^1(k+1))^{-1}\rho^2(k+1)\rho^2(k)(\rho^1(k))^{-1}(y_{k+1}-y_k)\big) \end{array}\right.$$

The problem of such an approach lies in that even for a small system, the task is not trivial and becomes prohibitive when the dimension grows up. However, let us mention that efficient algorithms involving elimination theory and symbolic computation can be used ([20]).

The next subsection is devoted to the conditions which must be fulfilled by (1.1) to guarantee that a given output y_k is flat.

1.2.2 Inverse system approach

First, it must be pointed out that for a given dynamics (first equation of (1.1)), y_k is not necessarily a flat output. It depends on the matrices C and D. Besides, for some specific matrices A and B, it may happen that none of the matrices C and D yield to a flat output. As a result, the conditions which guarantee that an output of (1.1) is flat must be expressed in terms of properties verified by the 4-uple of state matrices A, B, C and D.

A flat output characterization based on the inverse system has been proposed in [80] for switched linear systems. However, the extension to LPV systems is straightforward by considering the switching function as a function that takes values in a continuum rather than a finite set (the modes). Before giving the theorem, we must recall the definition and the characterization of the relative degree of (1.1) (see [89]).

Definition 1.2.2 *The relative degree of a discrete-time dynamical system is the minimal number r of iterations such that its output y_{k+r} at time $k+r$ is sensitive to its input u_k.*

Now, let us particularize this general definition to the system defined by equation (1.1). To this end, let us denote, for $k_2 \geq k_1$, by $\prod_{l=k_2}^{k_1} A_{\rho(l)}$ the product of the matrices $A_{\rho(l)}$, from k_2 down to k_1, and $\prod_{l=k_2}^{k_1} A_{\rho(l)} = \mathbf{1}_n$ if $k_2 < k_1$, and introduce the quantity $\mathcal{T}^{i,j}_{\rho(k)}$ defined for $j \leq i$ as

$$\begin{array}{rcl} \mathcal{T}^{i,j}_{\rho(k)} &=& C_{\rho(k+i)} \prod_{l=k+i-1}^{k+j+1} A_{\rho(l)} B_{\rho(k+j)} \text{ if } j \leq i-1 \text{ and} \\ \mathcal{T}^{i,i}_{\rho(k)} &=& D_{\rho(k+i)} \end{array} \tag{1.4}$$

From (1.1), it holds that $y_{k+1} = C_{\rho(k+1)}A_{\rho(k)}x_k + \mathcal{T}^{1,0}_{\rho(k)}u_k + \mathcal{T}^{0,0}_{\rho(k+1)}u_{k+1}$. Then, by iterating

the output y_k for $i \geq 1$ and noticing that $\mathcal{T}^{i,j}_{\rho(k+1)} = \mathcal{T}^{i+1,j+1}_{\rho(k)}$, it follows that

$$y_{k+i} = C_{\rho(k+i)} \prod_{l=k+i-1}^{k} A_{\rho(l)} x_k + \sum_{j=0}^{i} \mathcal{T}^{i,j}_{\rho(k)} u_{k+j} \tag{1.5}$$

Hence, if (1.1) admits a finite relative degree r, it follows from Definition 1.2.2 that:

1. $r = 0$ if $\mathcal{T}^{0,0}_{\rho(k)} \neq 0$ for all k
2. $r < \infty$ is the least integer s such that for all $k \geq 0$ and for all realization ρ.

$$\begin{array}{l} \mathcal{T}^{i,j}_{\rho(k)} = 0 \quad \text{for } i = 0, \ldots, s-1 \text{ and } j = 0, \ldots, \ i, \\ \mathcal{T}^{s,0}_{\rho(k)} \neq 0 \end{array} \tag{1.6}$$

Hence, it holds that

$$y_{k+r} = C_{\rho(k+r)} \prod_{l=k+r-1}^{k} A_{\rho(l)} x_k + \mathcal{T}^{r,0}_{\rho(k)} u_k \tag{1.7}$$

Theorem 1.2.1 ([34]) *If the LPV system (1.1) has a finite relative degree r, the following condition that must hold for a positive integer K, for all $k \geq 0$ and for all sequences $\{\rho(k), \ldots, \rho(k+K-1+r)\} \in \Theta^{r+K}$*

$$P_{\rho(k+K-1:k+K-1+r)} P_{\rho(k+K-2:k+K-2+r)} \cdots P_{\rho(k:k+r)} = 0 \tag{1.8}$$

with

$$P_{\rho(k:k+r)} = A_{\rho(k)} - B_{\rho(k)} (\mathcal{T}^{r,0}_{\rho(k)})^{-1} C_{\rho(k+r)} \prod_{l=k+r-1}^{k} A_{\rho(l)} \tag{1.9}$$

is equivalent to that y_k is a flat output.

The result can be proved by using the inverse dynamical system of (1.1).

Proof. If the LPV system (1.1) has a finite relative degree r, the following dynamical system can always be defined since $\mathcal{T}^{r,0}_{\rho(k)}$ is invertible.

$$\left\{ \begin{array}{lcl} \widehat{x}_{k+r+1} & = & P_{\rho(k:k+r)} \widehat{x}_{k+r} + B_{\rho(k)} (\mathcal{T}^{r,0}_{\rho(k)})^{-1} y_{k+r} \\ \widehat{u}_{k+r} & = & (\mathcal{T}^{r,0}_{\rho(k)})^{-1} (C_{\rho(k+r)} \prod_{l=k+r-1}^{k} A_{\rho(l)} \widehat{x}_{k+r} - y_{k+r}) \end{array} \right. \tag{1.10}$$

Iterating $K-1$ times the first equation of (1.10) yields

$$\begin{array}{rcl} \widehat{x}_{K+k+r} = & & P_{\rho(K+k-1:K+k-1+r)} \cdots P_{\rho(k:k+r)} \widehat{x}_{k+r} + \\ & & \sum_{i=0}^{K-1} \left[\prod_{j=1}^{i} P_{\rho(K+k-j:K+k-j+r)} \right] (\mathcal{T}^{r,0}_{\rho(k-1-i+k)})^{-1} B_{\rho(k-1-i+k)} y_{k-1-i+k+r} \end{array} \tag{1.11}$$

Let us define $\varepsilon_k = \widehat{x}_{k+r} - x_k$. Then ε_k verifies

$$\begin{aligned}\varepsilon_{k+1} &= \widehat{x}_{k+r+1} - x_{k+1} \\ &= P_{\rho(k:k+r)}\widehat{x}_{k+r} - P_{\rho(k:k+r)}x_k \\ &= P_{\rho(k:k+r)}\varepsilon_k\end{aligned}$$

and then

$$\varepsilon_{K+k} = P_{\rho(K+k-1:K+k-1+r)} \cdots P_{\rho(k:k+r)}\varepsilon_k$$

Hence, after a finite transient time equal to K, since (1.8) holds, one has $\widehat{x}_{k+r} = x_k$ for all $k \geq 0$. Hence, the state vector x_k exclusively depends on shifted outputs if and only if (1.8) holds and x_k reads for all $k \geq 0$ and for all sequences $\{\rho(k), \ldots, \rho(k+K-1+r)\} \in \Theta^{r+K}$

$$x_k = \sum_{i=0}^{K-1} \left[\prod_{j=1}^{i} P_{\rho(k-j:k-j+r)} \right] (\mathfrak{T}^{r,0}_{\rho(k-1-i)})^{-1} B_{\rho(k-1-i)} y_{k-1-i+r} \tag{1.12}$$

That is the LPV system (1.1) admits y_k as flat output if and only if $P_{\rho(K+k-1:K+k-1+r)} \cdots P_{\rho(k:k+r)} = 0$. □

Another way to prove Theorem 1.2.1 is to unfold (1.1).

Remark 1.2.1 *From (1.1), $x_k = A_{\rho(k-1)}x_{k-1} + B_{\rho(k-1)}u_{k-1}$. Then, if the system admits a finite relative degree r, one can infer from (1.7) that:*

$$u_k = (\mathfrak{T}^{r,0}_{\rho(k)})^{-1} y_{k+r} - (\mathfrak{T}^{r,0}_{\rho(k)})^{-1} C_{\rho(k+r)} \prod_{l=k+r-1}^{k} A_{\rho(l)} x_k \tag{1.13}$$

Then, substituting (1.13) into the expression of x_k above gives:

$$\begin{aligned}x_{k+1} &= A_{\rho(k)}x_k + (\mathfrak{T}^{r,0}_{\rho(k)})^{-1} B_{\rho(k)} y_{k+r} \\ &\quad -(\mathfrak{T}^{r,0}_{\rho(k)})^{-1} B_{\rho(k)} C_{\rho(k+r)} \prod_{l=k+r-1}^{k} A_{\rho(l)} x_k \\ &= P_{\rho(k:k+r)}x_k + (\mathfrak{T}^{r,0}_{\rho(k)})^{-1} B_{\rho(k)} y_{k+r}\end{aligned} \tag{1.14}$$

Iterating (1.14) $K-1$ times and performing the change of variable $k+K \rightarrow k$ yields

$$\begin{aligned}x_k =& P_{\rho(k-1:k-1+r)} \cdots P_{\rho(k-K:k-K+r)} x_{k-K} \\ &+ \sum_{i=0}^{K-1} \left[\prod_{j=0}^{i-1} P_{\rho(k-j-1:k-j-1+r)} \right] (\mathfrak{T}^{r,0}_{\rho(k-1-i)})^{-1} B_{\rho(k-1-i)} y_{k-1-i+r}\end{aligned} \tag{1.15}$$

The right hand side of (1.15) giving the expression of x^i_k ($i = 1, \ldots, n$) is a multivariate polynomial with indeterminates x^i_{k-K} and $y_{k-1-i+r}$. It follows that (1.2) holds for x_k if and

only if, for all sequences $\{\rho(k-K),\ldots,\rho(k-1+r)\} \in \Theta^{K+r}$ *and for all* $k \geq 0$, (1.8) *is fulfilled. In this case, the explicit form of* $\mathcal{F}_\rho$ *reads*

$$x_k = \sum_{i=0}^{K-1} \left[\prod_{j=0}^{i-1} P_{\rho(k-j-1:k-j-1+r)}\right] (\mathcal{T}^{r,0}_{\rho(k-1-i)})^{-1} B_{\rho(k-1-i)} y_{k-1-i+r} \tag{1.16}$$

The explicit form of $\mathcal{G}_\rho$ *involved in* (1.2) *is obtained by substituting* (1.12) *into* (1.13)*:*

$$\begin{aligned} u_k \;=\; & (\mathcal{T}^{r,0}_{\rho(k)})^{-1} y_{k+r} - (\mathcal{T}^{r,0}_{\rho(k)})^{-1} C_{\rho(k+r)} \prod_{l=k+r-1}^{k} A_{\rho(l)} \\ & \cdot \sum_{i=0}^{K-1} \left[\prod_{j=0}^{i-1} P_{\rho(k-j-1:k-j-1+r)}\right] (\mathcal{T}^{r,0}_{\rho(k-1-i)})^{-1} B_{\rho(k-1-i)} y_{k-1-i+r} \end{aligned} \tag{1.17}$$

The relevance of Theorem 1.2.1 is that it provides an explicit form for $\mathcal{F}_\rho$ and $\mathcal{G}_\rho$, assuming that y_k is a flat output. On the other hand, testing whether an output y_k is flat or not requires the knowledge of r and K and the test to calculate r and K would need the computation of the product (1.8) for an infinite number of realizations ρ. Clearly, it is infeasible in practice. It will be shown in Section 1.3 that with additional conditions borrowed from structural analysis and expressed in terms of a digraph associated to the LPV system, a complete framework handling those problems can be defined.

Considering Example 1.2.1 with the setting (1.3), one has $D_{\rho(k)} = 0$, $C_{\rho(k+1)}B_{\rho(k)} = 0$ and $C_{\rho(k+2)}A_{\rho(k+1)}B_{\rho(k)} = \rho^1(k+1)\cdot\rho^3(k)$. If $\rho^1(k)$ and $\rho^3(k)$ never vanish, the relative degree is $r = 2$. Hence $\mathcal{T}^{2,0}_{\rho(k)} = \rho^1(k+1)\cdot\rho^3(k)$ for all $k \geq 0$. The matrix $P_{\rho(k:k+2)} = A_{\rho(k)} - B_{\rho(k)}C_{\rho(k+1)}A_{\rho(k+1)}A_{\rho(k)}$ reads

$$P_{\rho(k:k+2)} = \begin{pmatrix} 1 & \rho^1(k) \\ -(\rho^1(k+1))^{-1} & -(\rho^1(k+1))^{-1}\rho^1(k) \end{pmatrix} \tag{1.18}$$

It turns out that $P_{\rho(k+2:k+4)}P_{\rho(k+1:k+3)} = 0$ for any realization ρ and thus, Equation (1.8) is fulfilled and $K = 2$.

The explicit expressions of $\mathcal{F}$ and $\mathcal{G}$ are given by Equations (1.16) and (1.17) respectively:

$$
\begin{aligned}
x_k &= \sum_{i=0}^{1}\left[\prod_{j=0}^{i-1} P_{\rho(k-j-1:k-j+1)}\right](\mathcal{T}^{2,0}_{\rho(k-1-i)})^{-1}B_{\rho(k-1-i)}y_{k-i+1}\\
&= (\mathcal{T}^{2,0}_{\rho(k-1)})^{-1}B_{\rho(k-1)}y_{k+1} + P_{\rho(k-1:k+1)}(\mathcal{T}^{2,0}_{\rho(k-2)})^{-1}B_{\rho(k-2)}y_k\\
&= \frac{1}{\rho^1(k)\rho^3(k-1)}\begin{pmatrix}0\\ \rho^3(k-1)y_{k+1}\end{pmatrix}\\
&\quad + \frac{1}{\rho^1(k-1)\rho^3(k-2)}\begin{pmatrix}1 & \rho^1(k-1)\\ -\frac{1}{\rho^1(k)} & -\frac{1}{\rho^1(k)}\rho^1(k-1)\end{pmatrix}\begin{pmatrix}0\\ \rho^3(k-2)y_k\end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
x_k &= \begin{pmatrix} y_k\\ \frac{1}{\rho^1(k)}(y_{k+1}-y_k)\end{pmatrix}
\end{aligned}
$$

and

$$
\begin{aligned}
u_k &= (\mathcal{T}^{2,0}_{\rho(k)})^{-1}y_{k+2} - (\mathcal{T}^{2,0}_{\rho(k)})^{-1}C_{\rho(k+r)}\prod_{l=k+r-1}^{k}A_{\rho(l)}\\
&\quad \cdot\sum_{i=0}^{K-1}\left[\prod_{j=0}^{i-1}P_{\rho(k-j-1:k-j+1)}\right](\mathcal{T}^{2,0}_{\rho(k-1-i)})^{-1}B_{\rho(k-1-i)}y_{k+1-i}\\
&= \frac{1}{\rho^1(k+1)\rho^3(k)}y_{k+2} - \frac{1}{\rho^1(k+1)\rho^3(k)}A_{\rho(k+1)}[2]A_{\rho(k)}x_k\\
&= \frac{1}{\rho^1(k+1)\rho^3(k)}\left[y_{k+2} - \begin{pmatrix}0 & \rho^2(k+1)\end{pmatrix}\begin{pmatrix}1 & \rho^1(k)\\ 0 & \rho^2(k)\end{pmatrix}\begin{pmatrix} y_k\\ \frac{1}{\rho^1(k)}(y_{k+1}-y_k)\end{pmatrix}\right]\\
u_k &= \frac{1}{\rho^1(k+1)\rho^3(k)}\left[y_{k+2} - \frac{\rho^2(k+1)\rho^2(k)}{\rho^1(k)}(y_{k+1}-y_k)\right]
\end{aligned}
$$

1.2.3 Mortality problem

It is worth pointing out that $P_{\rho(k:k+r)}$ is not imposed to satisfy (1.8) for every K products but for admissible sequences of products checking (1.8). Such problem is closely related to the notion of *mortality*. Indeed, it is said that a set of matrices is mortal if the zero matrix can be expressed as the product of a finite number of matrices. And yet, from the papers [90, 12, 17], except from some very special cases, the problem of checking mortality of a set of matrices is unsolvable because it is an undecidable problem. This verification can be done in some special cases by resorting to nilpotent semigroup as proposed in [89]. From a computational point of view, the complexity of the nilpotent semigroup approach to generate matrices that fulfil (1.8) is polynomial. The definition of nilpotent semigroup is recalled below.

Definition 1.2.3 (Nilpotent semigroup) *A semigroup $\mathcal{S}$ is a set together with an associative internal law. A semigroup $\mathcal{S}$ with an absorbing element 0 is said to be nilpotent if there exists 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 class of nilpotency of $\mathcal{S}$.*

If $\mathcal{S}$ is a set of matrices, the associative internal law is the matrix multiplication. Hence, 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.

Nilpotent semigroup is also related to *simultaneous triangularization*. Indeed, from Levitsky's Theorem [92], a set of matrices generate a nilpotent semigroup if and only if they are simultaneously triangularizable (see Appendix B) and all of them admit exclusively zero as eigenvalues.

Thus, we have the following theorem recalled from [89].

Theorem 1.2.2 *If the matrix $P_{\rho(k:k+r)}$ can be triangularized independently of ρ, then y_k is a flat output for (1.1).*

Actually, it is shown that if Theorem 1.2.2 holds, the uncountable set $\mathcal{S}$ of matrices defined as $\mathcal{S} = \{P_{\rho(k:k+r)} | \rho(k:k+r) \in \Theta^{r+1}\}$ generates a nilpotent semigroup.

The problem of such an approach lies in that the structure of a semigroup assumes that the matrices involved in the product commute. Hence, Theorem 1.2.2 is more conservative than Theorem 1.2.1. This is precisely the issue regarding Example 1.2.1 with the setting (1.3). Indeed, the set of matrices $\mathcal{S} = \{P_{\rho(k:k+1)} = A_{\rho(k)} - B_{\rho(k)}C_{\rho(k+1)}A_{\rho(k+1)}A_{\rho(k)} | \rho(k:k+1) \in \Theta^2\}$ defined in (1.18) does not generate a nilpotent semigroup, although y_k is actually a flat output.

As a conclusion, the existing methods that apply for LPV systems must consider any realization ρ. Symbolic computation can be a solution. However, the complexity of the algorithms for checking the conditions is either exponential or polynomial but in the latter case, the conditions are more conservative. The purpose of next Section 1.3 is to propose a complementary solution with a graph-oriented approach. Necessary and sufficient conditions will be derived to check whether a given output is flat or not. The corresponding algorithms to check the conditions have the polynomial complexity $O(n^3)$.

1.3 Graph Approach to Characterize Flat Outputs

As a clue to tackle the problem, we consider the LPV system (1.1) as a particular realization of a linear structured system. A structured linear discrete-time system is a system that admits the form:

$$\Sigma_\Lambda : x_{k+1} = I_A x_k + I_B u_k \tag{1.19}$$

where $x_k \in \mathbb{R}^n$ is the state vector and $u_k \in \mathbb{R}$ the input. The matrices $I_A \in \mathbb{R}^{n\times n}$ and $I_B \in \mathbb{R}^{n\times 1}$ are respectively the dynamical and the input matrices. By structured system, it is meant a system where the entries of the matrices I_A and I_B are '0' or '1'. If the entries a_{ij} of I_A (resp. b_i of I_B) are '0', it means that there are no relation between the state x^i_{k+1} at time $k+1$ and the state x^j_k at time k (resp. the state x^i_{k+1} at time $k+1$ and the input u_k). If the

entries are '1', it means that there exists a relation. Hence I_A and I_B act as incident matrices. In particular, in our context, to the entries '1' of the linear structured system Σ_Λ are assigned the non-zero (possibly time-varying parameters) of the LPV system (1.1).
Thereby, generic flatness for the LPV system (1.1) is equivalent to flatness of the structured system (1.19). Thus, a graph-oriented approach, well dedicated to linear structured systems, can be used to define generic flatness for the LPV system (1.1). It is shown below how the combination of both the algebraic conditions of Theorem 1.2.1 and additional graph-based conditions allows to define a complete and tractable framework to find the flat outputs and to explicit the functions of the state and the input with respect to the flat output in a convenient way. Having in mind this line, the technical developments are now detailed.

1.3.1 Necessary and sufficient conditions for flatness based on a digraph

A digraph denoted by $\mathcal{G}(\Sigma_\Lambda)$, associated to the state equations (1.19) of the system Σ_Λ describes the dependence relation between the state components and the input. It involves a vertex set $\mathcal{V}$ and an edge set $\mathcal{E}$. The vertices represent the state and the input components of Σ_Λ whereas the edges model the static or the dynamic relations between these variables. More precisely, $\mathcal{V} = \mathbf{X} \cup \mathbf{U}$ where $\mathbf{X}$ is the set of state vertices defined as $\mathbf{X} = \left\{\mathbf{x^1}, \ldots, \mathbf{x^n}\right\}$ and $\mathbf{U}$ is the set of input vertices. In the case under consideration here, that is SISO system, $\mathbf{U}$ is a singleton, that is $\mathbf{U} = \{\mathbf{u}\}$. The edge set is $\mathcal{E} = \mathcal{E}_A \cup \mathcal{E}_B$, with $\mathcal{E}_A = \left\{(\mathbf{x^j}, \mathbf{x^i}) \,|I_A(i,j) \neq 0\right\}$ and $\mathcal{E}_B = \left\{(\mathbf{u}, \mathbf{x^i}) \,|I_B(i) \neq 0\right\}$ where $I_A(i,j)$ and $I_B(i)$ denote respectively the $(i,j)th$ element of the matrix I_A and the ith element of the vector I_B. The pair $(\mathbf{v^i}, \mathbf{v^j})$ denotes a directed edge from the vertex $\mathbf{v^i} \in \mathcal{V}$ to the vertex $\mathbf{v^j} \in \mathcal{V}$. The vertex $\mathbf{v^F}$ will be associated to the flat output.

- A directed path P is a sequence of successive edges directed in the same direction which connect a sequence of vertices. It is said that the path P covers a vertex if this vertex is the begin or the end vertex of one of the edges of P;
- In a directed path from a vertex $\mathbf{v^i}$ to a vertex $\mathbf{v^j}$, it is said that $\mathbf{v^j}$ is a successor of $\mathbf{v^i}$ and conversely, $\mathbf{v^i}$ is a predecessor of $\mathbf{v^j}$;
- A directed path is simple when every vertex occurs only once in this path;
- The length of a directed path P is equal to the number of the edges involved in P. We denote by $\ell(\mathbf{v^i}, \mathbf{v^j})$ the minimal length of a path linking $\mathbf{v^i}$ to $\mathbf{v^j}$;
- $V_{ess}(\mathbf{v^i}, \mathbf{v^j})$ is the set of vertices, called essential vertices from $\mathbf{v^i}$ to $\mathbf{v^j}$, which are common to all the paths linking $\mathbf{v^i}$ to $\mathbf{v^j}$ whenever at least one path exists;
- $Pred(\mathbf{v^F})$ (resp. $Succ(\mathbf{u})$) is the set of all the predecessors (resp. successors) of $\mathbf{v^F}$ (resp. $\mathbf{u}$).

Let us recall the necessary and sufficient conditions which must be satisfied by a vertex $\mathbf{x^i}$ ($\mathbf{i} \in \{1,\dots,n\}$) or $\mathbf{u}$ of the digraph $\mathcal{G}(\Sigma_\Lambda)$ to be associated to a flat output $y_k^F = x_k^i$ ($i \in \{1,\dots,n\}$) or $y_k^F = u_k$ of (1.19). The extension to any linear combination of x_k^i and possibly u_k is then discussed.

Theorem 1.3.1 *[16] Consider the structured linear discrete-time system Σ_Λ described by (1.19). The output denoted y^F associated to a specific vertex $\mathbf{v^F} \in \mathbf{X} \cup \mathbf{U}$ is generically a flat output if and only if, in the associated digraph $\mathcal{G}(\Sigma_\Lambda)$, the following three conditions hold:*
C0. *$\mathbf{v^F}$ is a successor of $\mathbf{u}$;*
C1. *The length of all the $\{\mathbf{u}\}$-$\{\mathbf{v^F}\}$ simple paths is equal to $\ell(\mathbf{u},\mathbf{v^F})$;*
C2. *All the cycles cover at least an element of $V_{ess}(\{\mathbf{u}\},\{\mathbf{v^F}\})$.*

Proof.

The proof is based on the structural and graphical subdivision of the system Σ_Λ into 5 subsets Σ_i, $i = 0,\dots,4$. The subsets are defined according to their position with regard to the flat output and the input as depicted in Figure 1.1 and defined as follows:

- Σ_0 merges the input vertex $\mathbf{u}$, vertex $\mathbf{v^F}$ and all the state vertices which are predecessors of $\mathbf{v^F}$ and which are reachable from $\mathbf{u}$ without necessarily covering $\mathbf{v^F}$ *i.e.* the components of Σ_0 are $\mathbf{U} \cup \{\mathbf{v^F}\} \cup \big(Pred(\mathbf{v^F}) \cap Succ(\mathbf{u}) \cap \{\mathbf{v^i} \in \mathbf{X},\ \mathbf{v^F} \notin V_{ess}(\{\mathbf{u}\},\mathbf{v^i})\}\big)$;
- Σ_1 merges all the state vertices which are predecessors of $\mathbf{v^F}$ and which are reachable from $\mathbf{u}$ only by covering $\mathbf{v^F}$, *i.e.* the components of Σ_1 are $Pred(\mathbf{v^F}) \cap Succ(\mathbf{u}) \cap \{\mathbf{v^i} \in \mathbf{X},\ \mathbf{v^F} \in V_{ess}(\{\mathbf{u}\},\mathbf{v^i})\}$;
- Σ_2 merges all the state vertices which are predecessors of $\mathbf{v^F}$ but not successors of $\mathbf{u}$ *i.e.* the components of Σ_2 are $Pred(\mathbf{v^F}) \setminus Succ(\mathbf{u})$;
- Σ_3 merges all the state vertices which are successors of $\mathbf{u}$ but neither predecessors nor successors of $\mathbf{v^F}$, *i.e.* the components of Σ_3 are $Succ(\mathbf{u}) \setminus \big(Succ(\mathbf{v^F}) \cup Pred(\mathbf{v^F})\big)$;
- Σ_4 merges all the state vertices which are not successors of $\mathbf{u}$, not successors of $\mathbf{v^F}$ and not predecessors of $\mathbf{v^F}$ *i.e.* the components of Σ_4 are $\mathbf{X} \setminus \big(Succ(\mathbf{u}) \cup Succ(\mathbf{v^F}) \cup Pred(\mathbf{v^F})\big)$. These vertices can be linked to Σ_2 and Σ_3.

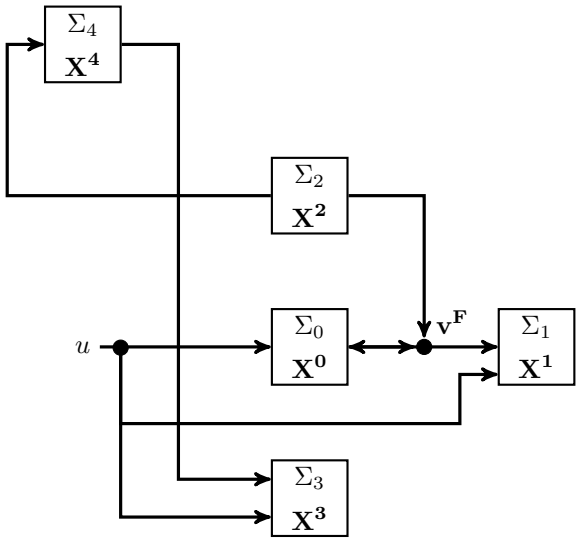

Figure 1.1: *Structural system subdivision of* Σ_Λ.

Note that this subdivision is unique and is motivated by the fact that it allows to characterize y^F as a flat output using the controllability and observability characteristics of each subsystem. Indeed, we can make two important remarks on subsystems Σ_2 and Σ_4:

Remark 1.3.1 *Subsystems* Σ_2 *and* Σ_4 *are not forced by the input* u*. So, it is clear that they don't involve any cycle (equivalently to a memorization operator), their state components will go to zero after a finite transient time. Conversely, if they involve at least one cycle, some of their state components will depend, at time* k*, on its own past value and so on the initial state value. And yet, the initial state of these subsystems cannot be expressed only as a function of the component associated to* $\mathbf{v^F}$ i.e. y^F*. Indeed,* y^F *is not sensitive to the state of* Σ_4 *and since* u *is an input for* Σ_0*, there isn't any equation linking the present, past and future values of* y^F *to the state components of* Σ_2 *only.*

Remark 1.3.2 *According to Definition 1.2.1, it is clear that* y^F *associated to the vertex* $\mathbf{v^F}$ *is a flat output if and only if:*

- *The input can be expressed as a function of the past, present, future values of* y^F *by considering simultaneously subsystems* Σ_0 *and* Σ_2 *since* y^F *is an output of only these two subsystems;*
- *There exists an integer* k_0 *such that for all* $k \geq k_0$*, all the state components of all the subsystems are either equal to zero or can be expressed as a function of the past, present, future values of* y^F*.*

We are now in position of proving the sufficiency and the necessity.
Sufficiency: When Condition **C2** is satisfied, there is no cycle in subsystems Σ_2 and Σ_4. In

this case and since these two subsystems are not forced by the inputs, after a finite time which is equal to the length of the longest path in these two subsystems, all the state components of these subsystems are equal to zero according to Remark 1.3.1.
Regarding subsystem Σ_0, the generic dimension of the observability subspace of the structured system Σ_Λ, considering that y^F associated to $\mathbf{v^F}$ is the output and that u is unknown, is equal to the length of the shortest $\{\mathbf{u}\}$-$\{\mathbf{v^F}\}$ path plus one *i.e.* $\ell(\mathbf{u}, \mathbf{v^F}) + 1$ [15]. Moreover, considering now that y^F associated to $\mathbf{v^F}$ is the output and that u is known, the generic dimension of the observability subspace of the structured system Σ_Λ is equal to the longest $\{\mathbf{u}\}$-$\{\mathbf{v^F}\}$ path plus the length of all the disjoint cycles which are not covered by this path. Nevertheless, when Conditions **C0**, **C1** and **C2** are valid, the longest $\{\mathbf{u}\}$-$\{\mathbf{v^F}\}$ path is also the shortest $\{\mathbf{u}\}$-$\{\mathbf{v^F}\}$ and there are no cycles that are disjoint with these paths in Σ_0. Therefore, if Conditions **C0**, **C1** and **C2** are satisfied, u can be expressed generically using the present and the past values of y^F associated to $\mathbf{v^F}$. Furthermore, all the state components of Σ_0 belonging to $V_{ess}(\mathbf{U}, \{\mathbf{v^F}\})$ can also generically be expressed using the present and the past values of y^F since they are generically observable considering that y^F is the output [15]. Therefore, substituting the input u and all the state vertices of all $V_{ess}(\mathbf{U}, \{\mathbf{v^F}\})$ by their expression in function of y^F, there exists a positive constant ν such that the linear structured system Σ_Λ can be written as:

$$\Sigma_\Lambda : x_{k+1} = I_{\tilde{A}} x_k + \phi(y^F_k, y^F_{k-1}, \dots, y^F_{k-\nu}) \tag{1.20}$$

where, as the cycles involve only elements of $V_{ess}(\mathbf{U}, \{\mathbf{v^F}\})$, the matrix $I_{\tilde{A}}$ is an adjacency matrix for a digraph of a structured system having no cycles. Thus, $I_{\tilde{A}}$ is nilpotent and verifies $I^n_{\tilde{A}} = 0$. As a result, every state can be expressed as a function of the past, present and future values of y^F which is thereby a flat output according to Remark 1.3.2.

Necessity: If Condition **C0** is not satisfied, then y^F is not sensitive to u. As a result, the input cannot generically be expressed using the past, present and future values of y^F and y^F cannot be a flat output. Condition **C0** is then necessary.
Condition **C1** is not applicable to subsystems Σ_1, Σ_3 or Σ_4. For subsystem Σ_0, if Conditions **C1** or **C2** [*i.e.* if there exist cycles which cover the state components which are not in $V_{ess}(\mathbf{U}, \{\mathbf{v^F}\})$] are not satisfied, the generic dimension of the observability subspaces with and without the input knowledge are different. In this case, the input cannot generically be expressed using y^F as output. Indeed, if there exist paths $\mathbf{u}$-$\mathbf{v}^F$ with different lengths $\ell_1 \neq \ell_2$, then the expression of y^F at time k, that is y^F_k, will involve at least $u_{k-\ell_1}$ and $u_{k-\ell_2}$. It is the same when Σ_0 involves a cycle. Therefore, Conditions **C1** and **C2** are necessary for Σ_0.
Only Condition **C2** is applicable to Σ_2. If it is not satisfied *i.e.* when there is a cycle in Σ_2 involving a vertex $\mathbf{x^i}$, the expression of y^F_k will always contain at least a term $x^i_{k-\ell_3}$ in addition to the $u_{k-\ell}$. So, it is impossible to express the input using only the past, present and future values of y^F, and thus y^F is not a flat output. Therefore, Condition **C2** is necessary for Σ_2.
Finally, assume that Condition **C2** is not satisfied for Σ_1, Σ_3 or Σ_4. In such a case, the

state components of Σ_1, Σ_3 and Σ_4 respectively depend on their initial state X_0^1, X_0^3 and X_0^4. However, neither X_0^1, X_0^3 nor X_0^4 are function of y^F. Therefore, these state components cannot be expressed as a function of exclusively the past, present and future values of y^F. As a result, y^F cannot be a flat output. □

Remark 1.3.3 *The conditions of Theorem 1.3.1 are generic. This means that they hold for almost every realization ρ of the corresponding LPV system (1.1). However, for some specific realizations ρ, they do no longer hold. This situation happens for example (and mainly) for the realizations ρ which cancel the weight associated to an edge linking the input to the flat output.*

Remark 1.3.4 *Conditions **C0-C2** of Theorem 1.3.1 can also be used as necessary and sufficient conditions to check if an output y in the form of linear combinations of some x_k^i ($i \in \{1, \ldots, n\}$) (and possibly u_k) is flat. Indeed, let us define $\mathbf{X_L} \subseteq \mathbf{X}$ be the set of variables involved in the linear combination under consideration. It suffices to add in the digraph $\mathcal{G}(\Sigma_\Lambda)$ a fictitious state $\mathbf{v^{n+1}}$ without outgoing edges and with ingoing edges from the vertices $\mathbf{v^i}$ corresponding to the variables of $\mathbf{X_L}$ (or $\mathbf{X_L} \cup \mathbf{U}$ if u_k is considered in the linear combination). considered in the linear combination (and $y = u$ if u is considered in the linear combination) is a flat output.*

Remark 1.3.5 *The computational complexity of the algorithms for checking the conditions is of the same order as the algorithms used for finding successors and predecessors of vertex subsets or for computing maximal linkings and essential vertices in a digraph. Thus, the complexity is polynomial and is $O(n^3)$ [15].*

1.3.2 Illustrative examples

Example 1.3.1 *Consider the linear discrete-time system of the example discussed in Section 1.1 with the setting (1.3). Its corresponding digraph is depicted in Figure 1.2. The component x^2 is not a flat output. Indeed, Condition **C2** is not fulfilled since there is a cycle on $\mathbf{v^1}$ and $\mathbf{v^1}$ does not belong to $V_{ess}(\{\mathbf{u}\}, \{\mathbf{v^2}\})$. On the other hand, x^1 is a flat output since all the Conditions **C0**, **C1** and **C2** are satisfied. That corroborates the result obtained from the direct approach detailed in Section 1.1.*

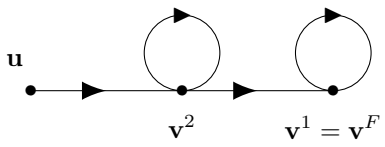

Figure 1.2: *Digraph associated to Example 1.3.1.*

Example 1.3.2 *Consider the linear discrete-time system represented by the digraph of Figure 1.3. The aim is to know if x^1 is a flat output. Condition **C0** is satisfied since there exists a*

path linking $\mathbf{u}$ *to* $\mathbf{v^1}$. *There are two simple paths from* $\mathbf{u}$ *to* $\mathbf{v^1}$ *of length respectively equal to 2 and 3. Therefore,* x^1 *cannot be a flat output since Condition* ***C1*** *is not satisfied.*

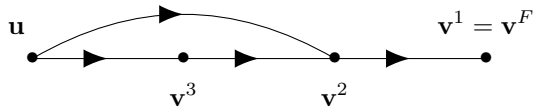

Figure 1.3: *Digraph associated to Example 1.3.2*

Example 1.3.3 *Consider the linear discrete-time system represented by the digraph of Figure 1.4. The aim is to know if* x^1 *is a flat output. Condition* ***C0*** *is satisfied since there exists a path linking* $\mathbf{u}$ *to* $\mathbf{v^1}$. *Moreover, Condition* ***C1*** *is also satisfied because the two simple paths linking* $\mathbf{u}$ *to* $\mathbf{v^1}$ *have the same length* $\ell(\mathbf{u}, \mathbf{v^1}) = 2$. *On the other hand, Condition* ***C2*** *is not satisfied because there exists a cycle including* $\mathbf{v^3}$ *which does not belong to* $V_{ess}(\{\mathbf{u}\}, \{\mathbf{v^1}\})$. *Thus,* x^1 *is not a flat output.*

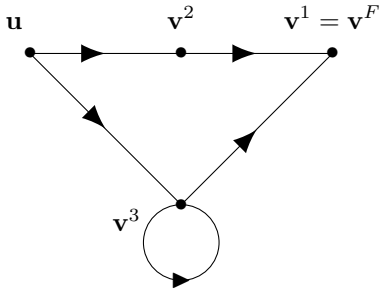

Figure 1.4: *Digraph associated to Example 1.3.3*

Example 1.3.4 *Consider the linear discrete-time system represented by the digraph of Figure 1.5. The aim is to know if* x^1 *is a flat output. Condition* ***C0*** *is satisfied since there exists a path linking* $\mathbf{u}$ *to* $\mathbf{v^1}$. *Moreover, Condition* ***C1*** *is also satisfied since there is only one simple path linking* $\mathbf{u}$ *to* $\mathbf{v^1}$. *Furthermore, Condition* ***C2*** *is satisfied since the cycles involve* $\mathbf{v^1}$, $\mathbf{v^2}$ *and* $\mathbf{v^3}$ *which are all elements of* $V_{ess}(\{\mathbf{u}\}, \{\mathbf{v^1}\})$. *Thus,* x^1 *is a flat output.*

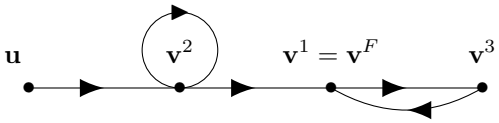

Figure 1.5: *Digraph associated to Example 1.3.4*

1.3.3 Connection with the algebraic conditions

The aim of this section is to make the connection between the quantities r (relative degree) and K involved in Theorem 1.2.1 and graph properties of $\mathcal{G}(\Sigma_\Lambda)$. Regarding r, it will be shown that

the connection is straightforward. On the other hand, regarding K, making the connection is not trivial. The proofs given below will be based on the same subdivision of the whole set of states into 5 subsets Σ_i, $i = 0, \ldots, 4$, as depicted in Figure 1.1.

Proposition 1.3.1 ([34]) *The relative degree r of the system (1.1) is equal to $\ell(\mathbf{u}, \mathbf{v^F})$ of $\mathcal{G}(\Sigma_\Lambda)$.*

Proof. From Definition 1.2.2, let us merely consider the subsystem Σ_0 of the subdivision of $\mathcal{G}(\Sigma_\Lambda)$ depicted in Figure 1.1. Indeed, subsystem Σ_0 directly connects the input $\mathbf{u}$ and the vertex $\mathbf{v^F}$ assigned to the flat output. Then, the result holds as a straightforward consequence of Condition **C1** of Theorem 1.3.1. □

Proposition 1.3.2 ([34]) *When y_k^F is a flat output, the integer K involved in Theorem 1.2.1 is generically equal to the maximal path length over all the simple $\{\mathbf{u}\}$-$\mathbf{X}$ paths of $\mathcal{G}(\Sigma_\Lambda)$.*

Proof. First, let us notice that K is equal to the minimal integer such that, for some sufficiently large integer k, the state vector x_k can be written as a function of $y_{k-K+r}, \ldots, y_{k+r-1}$ only. Let us denote with d_M the maximal path length over all the simple $\{\mathbf{u}\}$-$\mathbf{X}$ paths of $\mathcal{G}(\Sigma_\Lambda)$. Since the elements of Σ_2 and Σ_4 are not submitted to the input $\mathbf{u}$, they are equal to zero after a finite transient time and thus can always be expressed using y_{k-1+r}, y_{k+r}, $y_{k+1+r}, \ldots, y_{k+d_M-1+r}$.
Now, consider the subsystem Σ_0. Since y^F is a flat output, Condition **C1** of Theorem 1.3.1 is fulfilled. Thus, the paths between $\mathbf{u}$ and $\mathbf{v^F}$ have the same length which is equal to r, according to Proposition 1.3.1. Hence, u_k can be written as a function of $y_{k+1}, \ldots, y_{k+r}$. Thus, every element of Σ_0 can be written as a function of $y_k, \ldots, y_{k+r-1}$.
Regarding the subsystem Σ_3, after a finite transient time, its elements can be considered as having only $\mathbf{u}$ as input because the state components of Σ_4 go to zero. There is no cycle in Σ_3 because the system being flat, the vertices are not essential. All its state vertices are linked to $\mathbf{u}$ through simple paths of lengths less than d_M. Thus, the corresponding state components can be written as a function of $u_{k-1}, \ldots, u_{k-d_M}$ and so, after substituting the expressions of $u_{k-1}, \ldots, u_{k-d_M}$, as a function $y_{k+1-d_M}, \ldots, y_{k+r-1}$.
Finally, let us consider Σ_1. By the flatness assumption, it does not involve any cycle. Thus, the state components can be expressed as a function of u_{k-1}, $u_{k-2}, \ldots u_{k-d_M}$ and so as a function of $y_{k+1-d_M}, \ldots, y_{k-1+r}$.
Therefore, $K \leq d_M$. On the other hand, since a vertex linked by a path of length d_M to $\mathbf{u}$ depends on u_{k+d_M-1}, then at least y_{k+d_M-1+r} is necessary to express it. Thus, $K \geq d_M$ and finally $K = d_M$. □

Remark 1.3.6 *If the system (1.1) is of dimension n, the number of vertices of $\mathcal{G}(\Sigma_\Lambda)$ equals n. It follows, from Proposition 1.3.2, that if y_k^F is a flat output of (1.1) then the integer K involved in Theorem 1.2.1 is bounded by n.*

1.3.4 Search procedure for flat outputs

Considering the system (1.1), we can sum up the steps to find its flat outputs until the expression of functions $\mathcal{F}$ and $\mathcal{G}$ of (1.2) as follow:

1. search flat outputs by considering the related digraph of (1.1) and by checking Conditions **C0**–**C2** of Theorem 1.3.1
2. find the relative degree r and the quantity K for the flat output by applying Propositions 1.3.1 and 1.3.2
3. finally find $\mathcal{F}$ and $\mathcal{G}$ by applying Equations (1.16) and (1.17).

Example 1.3.5 *Let us consider the system of Example 1.2.1.*

$$A_{\rho(k)} = \begin{pmatrix} 1 & \rho^1(k) \\ 0 & \rho^2(k) \end{pmatrix}, B_{\rho(k)} = \begin{pmatrix} 0 \\ \rho^3(k) \end{pmatrix} C_{\rho(k)} = \begin{pmatrix} 1 & 0 \end{pmatrix} D_{\rho(k)} = 0.$$

1. Search for the flat outputs with the graph-based condition

The graph of the linear structured system associated to the LPV system (1.3) is given in Figure 1.6.

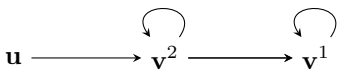

Figure 1.6: Digraph of the linear structured system associated to the LPV system (1.3). .

*Let us find (if any) the flat outputs of (1.3) by checking the Conditions **C0-C2**. It follows that :*

- x_k^2 *is not a flat output. Indeed, when considering* $\mathbf{v^2}$*, Condition **C2** is not satisfied because* $\mathbf{v^1}$ *does not belong to* $V_{ess}(\{\mathbf{u}\}, \{\mathbf{v^2}\})$ *and there is a cycle that covers* $\mathbf{v^1}$*.*
- x_k^1 *is a flat output because, when considering* $\mathbf{v^1}$ *as output, Conditions **C0-C2** are all satisfied.*

2. Find the relative degree r ***and the quantity*** K ***for the flat output***

From Proposition 1.3.1, x_k^1 *being a flat output, the relative degree* r *of the system (1.21) is given by* $\ell(\mathbf{u}, \mathbf{v^1}) = 2$*. From Proposition 1.3.2,* K *is given by the maximal simple path length between the input and any component of the internal state. Hence, the quantity* K *equals* 2 *when considering the path that links* $\mathbf{u}$ *to* $\mathbf{v^1}$*.*

3. Find the functions $\mathcal{F}_\rho$ ***and*** $\mathcal{G}_\rho$

Considering x_k^1 *as a flat output, it follows that* $C_{\rho(k)} = \begin{pmatrix} 1 & 0 \end{pmatrix}$.

From the values $K = 2$ and $r = 2$, one can compute $P_{\rho(k:k+2)}$ from (1.9). We have that

$$\mathcal{T}^{2,0}_{\rho(k)} = C_{\rho(k+2)} \cdot A_{\rho(k+1)} \cdot B_{\rho(k)} = \rho^1(k+1) \cdot \rho^3(k).$$

and thus

$$\begin{aligned} P_{\rho(k:k+2)} &= A_{\rho(k)} - \left(\mathcal{T}^{2,0}_{\rho(k)}\right)^{-1} B_{\rho(k)} C_{\rho(k+2)} A_{\rho(k+1)} A_{\rho(k)} \\ P_{\rho(k:k+3)} &= \begin{pmatrix} 1 & \rho^1(k) \\ -(\rho^1(k+1))^{-1} & -(\rho^1(k+1))^{-1}\rho^1(k) \end{pmatrix} \end{aligned}$$

Hence, by applying (1.16) one has:

$$x_k = \begin{pmatrix} y_k \\ \dfrac{1}{\rho^1(k)}(y_{k+1} - y_k) \end{pmatrix}$$

and from (1.17) one has :

$$u_k = \frac{1}{\rho^1(k+1)\rho^3(k)}\left[y_{k+2} - \frac{\rho^2(k+1)\rho^2(k)}{\rho^1(k)}(y_{k+1} - y_k)\right]$$

what gives the expressions of $\mathcal{F}_\rho$ and $\mathcal{G}_\rho$.

Example 1.3.6 *Let us consider the discrete-time dynamics of an unbalanced nonlinear dc motor studied in [68].*

$$\begin{aligned} \begin{pmatrix} \theta(k+1) \\ \omega(k+1) \\ I(k+1) \end{pmatrix} &= \begin{pmatrix} 1 & T_s & 0 \\ p(k) \cdot T_s Mgl/J & 1 - T_s b/J & T_s K/J \\ 0 & -T_s K/L & 1 - T_s R/L \end{pmatrix} \begin{pmatrix} \theta(k) \\ \omega(k) \\ I(k) \end{pmatrix} \\ &+ \begin{pmatrix} 0 \\ 0 \\ T_s/L \end{pmatrix} u_k \end{aligned} \tag{1.21}$$

where T_s, M, g, l, b, J, K are the parameters of the physical system and $p(k) \in [0.1, 1]$ is a scheduling time-varying parameter. The system can be written as an LPV system like (1.1) with state variables x^1_k, x^2_k and x^3_k respectively corresponding to $\theta(k), \omega(k)$ and $I(k)$. Letting $\rho^0(k) = 1$, $\rho^1(k) = T_s$, $\rho^2(k) = p(k)T_s Mgl/J$, $\rho^3(k) = 1 - T_s b/J$, $\rho^4(k) = T_s K/J$, $\rho^5(k) = -T_s K/L$, $\rho^6(k) = 1 - T_s R/L$ and $\rho^7(k) = T_s/L$, one has

$$A_{\rho(k)} = \begin{pmatrix} 1 & \rho^1(k) & 0 \\ \rho^2(k) & \rho^3(k) & \rho^4(k) \\ 0 & \rho^5(k) & \rho^6(k) \end{pmatrix}, \; B_{\rho(k)} = \begin{pmatrix} 0 \\ 0 \\ \rho^7(k) \end{pmatrix}.$$

Here, all the parameters $\rho^i(k)$ have constant values except $\rho^2(k)$. In the sequel, to avoid too

heavy notation, since the first entry 1 of A does not correspond to a physical paramater, we will keep 1 instead of $\rho^0(k)$ in the subsequent equations.
In the following, we describe the successive steps which define the complete framework, to find the flat outputs and then to explicit the functions $\mathcal{F}_\rho$ and $\mathcal{G}_\rho$ characterizing those outputs. The characterization will hold for any parameters, in particular for any values of $p(k)$. This will highlight the relevance of incorporating structural analysis in the complete framework.

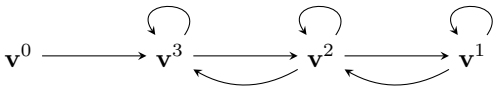

Figure 1.7: Digraph of the linear structured system associated to the LPV system (1.21). The vertex $\mathbf{v^i}$, $i = 1, 2, 3$ is associated to the component x_k^i of the state x of (1.21) and $\mathbf{v^0}$ is associated to the input u_k.

1. Search for the flat outputs with the graph-based condition

The graph of the linear structured system associated to the LPV system (1.21) is given in Figure 1.7.

*Let us find (if any) the flat outputs of (1.21) by checking the Conditions **C0-C2**. It follows that :*

- *x_k^2 (resp. x_k^3) is not a flat output. Indeed, when considering $\mathbf{v^2}$ (resp. $\mathbf{v^3}$), Condition **C2** is not satisfied since $\mathbf{v^1}$ does not belong to $V_{ess}(\{\mathbf{v^0}\}, \{\mathbf{v^2}\}) = \{\{\mathbf{v^0}\}, \{\mathbf{v^3}\}, \{\mathbf{v^2}\}\}$ (resp. $V_{ess}(\{\mathbf{v^0}\}, \{\mathbf{v^3}\}) = \{\{\mathbf{v^0}\}, \{\mathbf{v^3}\}\}$) whereas there is a cycle on $\mathbf{v^1}$.*
- *x_k^1 is as flat output because, when considered $\mathbf{v^1}$ as output, Conditions **C0-C2** are satisfied.*

2. Find the relative degree r and the quantity K for the flat output

From Proposition 1.3.1, x_k^1 being a flat output, the relative degree r of the system (1.21) is given by $\ell(\mathbf{v^0}, \mathbf{v^1}) = 3$. From Proposition 1.3.2, K is given by the maximal simple path length between the input and any component of the internal state. Hence, the quantity K equals 3 by considering the path linking $\mathbf{v^0}$ to $\mathbf{v^1}$.

3. Find the functions $\mathcal{F}_\rho$ and $\mathcal{G}_\rho$

Considering x_k^1 as a flat output, it follows that $C_{\rho(k)} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$.
From the values $K = 3$ and $r = 3$, one can compute $P_{\rho(k:k+3)}$ from (1.9). We have that

$$\mathcal{T}^{r,0}_{\rho(k)} = C_{\rho(k+3)} \cdot A^{\rho(k+2)}_{\rho(k+1)} \cdot B_{\rho(k)} = \rho^1(k+2)\rho^4(k+1)\rho^7(k).$$

and thus

$$P_{\rho(k:k+3)} = A_{\rho(k)} - \left(\mathfrak{T}^{r,0}_{\rho(k)}\right)^{-1} B_{\rho(k)} C_{\rho(k+3)} A^{\rho(k+2)}_{\rho(k)}$$

$$P_{\rho(k:k+3)} = \begin{pmatrix} 1 & \rho^1(k) & 0 \\ \rho^2(k) & \rho^3(k) & \rho^4(k) \\ -P_1(k) & -P_2(k) & -P_3(k) \end{pmatrix}$$

with

$$\begin{aligned} P_1(k) &= 1 + \rho^1(k+2)\rho^2(k+1) + \big(\rho^1(k+1) + \rho^1(k+2)\rho^3(k+1)\big)\rho^2(k) \\ P_2(k) &= \big(1 + \rho^1(k+2)\rho^2(k+1)\big)\rho^1(k) + (\rho^1(k+1) + \rho^1(k+2)\rho^3(k+1))\rho^7(k) \\ P_3(k) &= \big(\rho^1(k+1) + \rho^1(k+2)\rho^3(k+1)\big)\rho^4(k) \end{aligned}$$

Hence, by applying (1.16) one has:

$$\begin{aligned} x_k = {} & (\mathfrak{T}^{r,0}_{\rho(k-1)})^{-1} B_{\rho(k-1)} y_{k+2} + P_{\rho(k-1:k+2)} (\mathfrak{T}^{r,0}_{\rho(k-2)})^{-1} B_{\rho(k-2)} y_{k+1} \\ & + P^{\rho(k-1:k+2)}_{\rho(k-2:k+1)} (\mathfrak{T}^{r,0}_{\rho(k-3)})^{-1} B_{\rho(k-3)} y_k \end{aligned}$$

$$\begin{aligned} x_k = {} & \frac{1}{\rho^1(k+1)\rho^4(k)} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} y_{k+2} + \frac{1}{\rho^1(k)\rho^4(k-1)} \begin{pmatrix} 0 \\ \rho^4(k-1) \\ -P_3(k-1) \end{pmatrix} y_{k+1} \\ & + \frac{1}{\rho^1(k-1)\rho^4(k-2)} \begin{pmatrix} \rho^1(k-1)\rho^4(k-2) \\ \rho^3(k-1)\rho^4(k-2) - \rho^4(k-1)P_3(k-2) \\ -P_2(k-1)\rho^4(k-2) + P_3(k-1)P_3(k-2) \end{pmatrix} y_k \end{aligned}$$

$$x_k = \begin{pmatrix} y_k \\ \dfrac{y_{k+1}}{\rho^1(k)} + \dfrac{\rho^3(k-1)\rho^4(k-2) - \rho^4(k-1)P_3(k-2)}{\rho^1(k-1)\rho^4(k-2)} y_k \\ \dfrac{y_{k+2}}{\rho^1(k+1)\rho^4(k)} - \dfrac{P_3(k-1)}{\rho^1(k)\rho^4(k-1)} y_{k+1} + \\ \dfrac{-P_2(k-1)\rho^4(k-2) + P_3(k-1)P_3(k-2)}{\rho^1(k-1)\rho^4(k-2)} y_k \end{pmatrix}$$

and from (1.17) *one has :*

$$\begin{aligned} u_k &= \left(\mathfrak{T}^{r,0}_{\rho(k)}\right)^{-1}\left(y_{k+3} - C_{\rho(k+3)}A^{\rho(k+2)}_{\rho(k)}x_k\right) \\ &= \frac{1}{\rho^1(k+2)\rho^4(k+1)\rho^7(k)}\left[y_{k+3} - \begin{pmatrix} P_1(k) \\ P_2(k)+\rho^1(k+2)\rho^4(k+1)\rho^5(k) \\ P_3(k)+\rho^1(k+2)\rho^4(k+1)\rho^6(k)\end{pmatrix}^T x_k\right] \end{aligned}$$

where 'T' stands for the transpose of the matrix.
Finally,

$$\begin{aligned} x^1_k =& y_k \\ x^2_k =& \frac{y_{k+1}}{T_s} + \left(\frac{1-T_sb/J}{T_s} - (2-T_sb/J)T_s(K/J)\right)y_k \\ x^3_k =& \frac{J}{T_s^2K}y_{k+2} - 2\frac{T_sb}{J}y_{k+1} \\ & - \left[\left(1+T_s^2p(k+1)Mgl/J\right) + \left(2-T_sb/J\right)T_s/L + T_s^2\left(2-T_sb/J\right)^2K/J\right]y_k \end{aligned} \tag{1.22}$$

(1.23)

$$u_k = \frac{JL}{T_s^3K}\left(y_{k+3} - P_1(k)\,x^1_k - \left(P_2(k)+\tfrac{T_s^3K^2}{LJ}\right)x^2_k + \left(P_3(k)+\tfrac{T_s^2K}{J}\tfrac{1-T_sR}{L}\right)x^3_k\right) \tag{1.24}$$

what gives the expressions of $\mathcal{F}_\rho$ *and* $\mathcal{G}_\rho$.

1.4 Conclusion

In this chapter, a mixed algebraic/graph-oriented approach has been proposed to characterize flatness for discrete-time LPV systems. We have derived necessary and sufficient conditions in terms of state space matrices realization of the LPV system. Those conditions allow to give explicitly the functions expressing the state and the input in terms of the flat output. On the other hand, we have considered LPV systems as linear structured systems. Resorting to additional graph-based conditions borrowed from structural analysis, by a (non trivial) combination of both conditions, a complete and tractable framework to find the flat outputs and characterize the flat outputs has been proposed. The use of the algebraic approach only would require an infinite number of numerical tests. Another interesting point of view of this mixed approach is that, it is not only useful to check if a given set of matrices satisfy (1.8), in other words for analysis, but can also be applied to generate matrices that fulfil (1.8), in other words for synthesis. This latter outcome will be central to the design of SSSC for cryptographic purposes as shown in Chapter 2.

Chapter 2

Flat LPV Systems and Self-Synchronizing Stream Ciphers

This chapter aims at showing the correspondence between flat LPV systems and Self-Synchronizing Stream Ciphers which belong to one of the class of stream ciphers. Then, we propose a new construction of Self-Synchronizing Stream Ciphers with a more general class than the ones involving T-functions. The construction is based on the mixed algebraic/graph approaches described in Chapter 1. The results in this chapter are published in [36].

Contents

2.1 Introduction

Born from the imagination and intuition of historical strategists, cryptology - etymologically the science of secrecy - owes its longevity to all those who, over the centuries, have discovered new applications and revealed its potential. First reserved for military and diplomatic uses, cryptology has been an integral part of our everyday life since the advent of the Internet. Nowadays, this science of secrecy offers unrivaled opportunities to date. This science can be seen as a fusion of two fields: cryptography and cryptanalysis.

Cryptography is the study and the design of cryptographic algorithms to protect data. The original data to protect, called *plaintext*, are scrambled by a *transmitter* that uses cryptographic algorithms and send it through a public channel to a *receiver*. This latter, when receiving the scrambled data also called *encrypted data*, needs to recover the plaintext by resorting to reverse operations using also cryptographic algorithms. One says that the emitter encrypts the plaintext by using a *cipher* that is the encryption algorithm and conversely, one says that the receiver decrypts the encrypted data by using a *decipher* that is the decryption algorithm.

On the other hand, cryptanalysis is the study of weakness within algorithms that are designed by cryptographers. It involves attackers also called adversaries that try by using among others statistical or algebraic-based theory to break encryption schemes by revealing secret parameters that are used within these schemes.

These two fields can be seen as opposite and at the same time as complementary. Opposite because cryptanalysis breaks what cryptography designs and complementary because they allow each other to perform studies that are performed within these two fields.

Cryptography is used everywhere. People carry on them, without necessarily knowing, many cryptographic devices: mobile phone, smart cards (credit cards, biometric passport, social assurance card, pay-per-view cards), vehicle key start. Download applications on smartphones, software updates are marked with digital signature. Communications between people using Voice over IP or via social networks are encrypted with strong cryptographic algorithms.

Until the middle of the 70's, the only way that allows two entities to communicate securely, by using cryptography means, is to resort to symmetric cryptography. The *transmitter* uses a secret key that is shared with the *receiver*. This requires that the two entities need to communicate each other the secret key before initiating the exchange of encrypted data. In the middle of the 70's, Whitfield Diffie and Martin Hellman published a new concept that allows two entities, that never met before, to be able to communicate without sharing a same secret: it was the birth of asymmetric cryptography. Each entity owns a non-secret key called *public key* from which anybody can send him an encrypted data and a secret key called *private key* that allows him to decrypt encrypted data he received.

Any protection of information using cryptography requires either symmetric cryptography or asymmetric cryptography. And for a cryptographic system, it is a priority that one of the following four services should be provided:

Confidentiality: guarantees that the content of the message is kept secret and protected against unauthorized release

Authentication: guarantees the identity of the emitter of the message

Integrity: guarantees that the content of the message to protect has not be altered during the transmission

Non-Repudiation: guarantees that the emitter of the message cannot deny sending the message.

In this thesis, we tackle a problem that is related to symmetric cryptographic and that guarantees confidentiality of data: that is the design of so-called Self-Synchronizing Stream Ciphers, that will be developed in Section 2.3. The next section gives a general description on symmetric cryptography.

2.2 Generalities on Symmetric Cryptography

When using symmetric cryptography to encrypt or protect data, the quantity of the data, the device to process the cryptographic algorithms and the transmission speed of the encrypted data are a concern. The emergence of asymmetric cryptography in the the 70's with the public key concept may let believe to a regression of the use of symmetric cryptography. But this is certainly not the case. Indeed, architectures for symmetric key cryptographic algorithms are more suitable for hardware implementation, and are used more and more because of the number of devices needing lightweight cryptography increases. We recall the two families of ciphers used in symmetric cryptography.

2.2.1 Block ciphers

Block ciphers are widespread in symmetric cryptography. They operate on blocks of data i.e they can be seen as functions that map a n-bit data to a m-bit data. A block cipher encryption function can be used in a combination way called *mode of operation* to provide message authentication techniques, data integrity mechanisms, entity authentication protocols. As block cipher, we can mention FEAL [102], DES [33], AES [1]. We recall some example of mode of operation of block ciphers in Appendix A.1.

2.2.2 Stream ciphers

Stream ciphers are based on the so-called Vernam cipher where the plaintext (a binary string of some length) is bitwise added to a (binary) secret key of the same length, in order to produce

the ciphertext. The Vernam cipher is also called the one-time-pad because a new random secret key must be used for every encryption.

For a stream cipher, it must be given an alphabet A, that is, a finite set of basic elements named symbols. Hereafter, the index k will stand for the discrete-time. On the *transmitter* side, a plaintext (also called information or message) $m \in \mathcal{M}$ ($\mathcal{M}$ is the message space) consisting of a string of symbols $m_k \in A$ is encrypted according to an encryption function e which depends on a so-called running key (also called keystream) z_k which is invertible for any prescribed z_k. Hence, the ciphering is performed with

$$c_{k+b} = e(z_{k+b}, m_k) \tag{2.1}$$

where e is the ciphering function, m_k is the plaintext symbol and $c_k \in B$ is the ciphertext symbol which belongs to an alphabet B usually (and assumed hereafter) identical to A. The integer $b \geq 0$ stands for a potential delay between the plaintext m_k and the corresponding ciphertext c_{k+b}. This is explained by computational reasons, for instance pipelining (see [29] for instance). Consequently, for stream ciphers, the way how to encrypt each plaintext symbol changes on each iteration. The resulting ciphertext $c \in \mathcal{C}$ ($\mathcal{C}$ is called the ciphertext space), a string of symbols c_k, is conveyed through a public channel to the *receiver*.
At the receiver side, the ciphertext c is decrypted according to a decryption function d which depends on the running key $\hat{z}_k$. The decryption function d obeys the following rule. For any two keystream symbols $\widehat{z}_{k+b}, z_{k+b}$, it holds that

$$\hat{m}_{k+b} := d(c_{k+b}, \widehat{z}_{k+b}) = m_k \text{ whenever } \hat{z}_{k+b} = z_{k+b}. \tag{2.2}$$

From (2.2), it is clear that, beyond the equality of the secret keys, the running keys z_k and $\hat{z}_k$ must be synchronized for a proper decryption. The distinct classes of stream ciphers differ from each other by the way on how the keystreams are generated and synchronized. The generators delivering the keystreams will be parametrized by a secret key denoted in the sequel by θ.

2.2.2.1 Synchronous Stream Ciphers

Synchronous Stream Ciphers admit the equations:

$$\begin{cases} q_k = & \sigma^s(q_{k-1}) \\ z_k = & s(q_k) \\ c_k = & e(z_k, m_k) \end{cases} \tag{2.3}$$

where σ^s is the next-state transition function, s acts as a filter and generates the keystream z_k. The value q_k is the internal state of the cipher whose initial value is generated from a secret key. This nonce is generated by applying a key scheduling process or a Linear Feedback Shift

Register.

Among the huge variety of synchronous stream ciphers that exist, we can mention RC4, SOSEMANUK [9], GRAIN [53], TRIVIUM [21].

2.2.2.2 Self-Synchronizing Stream Ciphers

Self-Synchronizing Stream Ciphers admit the equations:

$$\begin{cases} z_k = & \sigma_\theta^{ss}(c_{k-l-M+1}, \ldots, c_{k-l}) \\ c_k = & e(z_k, m_k) \end{cases} \tag{2.4}$$

where σ_θ^{ss} is the function that generates the keystream z_k; l is a non-zero positive integer standing for a possible delay. The keystream generator σ_θ^{ss} depends on past values of c_k. The number of past values is most often bounded and equals M, the delay of memorization. Equation 2.4 is also known as the canonical form of the SSSC and is illustrated by Figure 2.1.

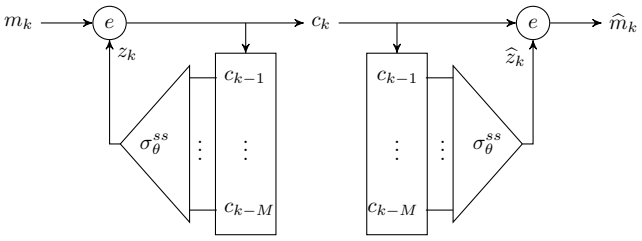

Figure 2.1: Canonical form of SSSC for $l = 1$.

To decrypt a ciphertext c_k and recover the right plaintext, it is necessary to have the right past previous M ciphertexts. Hence, there is a synchronization delay that is exactly M. The decipher equation is then given by:

$$\begin{cases} \widehat{z}_k = & \sigma_\theta^{ss}(c_{k-l-M+1}, \ldots, c_{k-l}) \\ \widehat{m}_k = & e(z_k, c_k) \end{cases} \tag{2.5}$$

The transmission of encrypted data through free-error channel are the ideal solution when dealing with cryptographic protocols. However, it is difficult to have in reality such a channel, and transmissions of ciphertexts are subject to errors such as bit tampering, bit insertion or deletion. When faced to bit tampering, synchronous stream cipher can be used without need of additional synchronization protocol, since a 1-bit error in the ciphertext will result in 1-bit error in the recovered plaintext. However, this problem is catastrophic to block cipher that handle data which are sequence of bits. On the other hand, insertion or deletion of one bit of the ciphertext make a loss of synchronization both for block cipher and synchronous stream cipher. The use of SSSC when facing this situation allows automatic resynchronization after a

transient time, without the use of any synchronization mechanism.

We can summarize advantages of SSSC as :

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

We detail in the next section the different architecture related to Self-Synchronizing Stream Ciphers.

2.3 Architectures of Self-Synchronizing Stream Ciphers

The defining property of self-synchronizing stream encryption is that a keystream bit depends on a limited number M (the memory) of past keystream bits, and hence it suffices to have received the last $M + 1$ ciphertext bits to correctly decrypt the current ciphertext bits. A first architecture calls for a block cipher (AES,DES) in a 1-bit CFB mode (see Figure A.3) as illustrated in Figure 2.2.

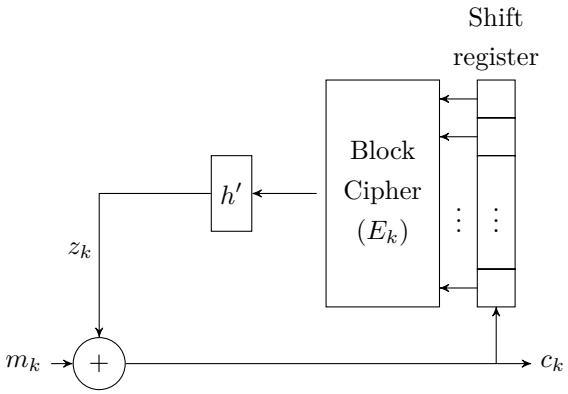

Figure 2.2: SSSC in CFB mode. A shift register is used to gather the previous ciphertexts. The output of the encryption algorithm E_k is filtered with a function h' to produce a 1-bit keystream z_k that is XORed to the plaintext m_k.

As alternative to the SSSC architecture in CFB mode, U. Maurer proposed in [74] several design approaches for SSSC based on automata. The design replaces the shift register in CFB mode based SSSC with serial and parallel combination of automata where each automaton consists on a function associated to a shift register of reduced size (see Figure 2.3). This architecture had led to the first one with finite state machine instead of simple shift register, that provides secure and high speed SSSC.

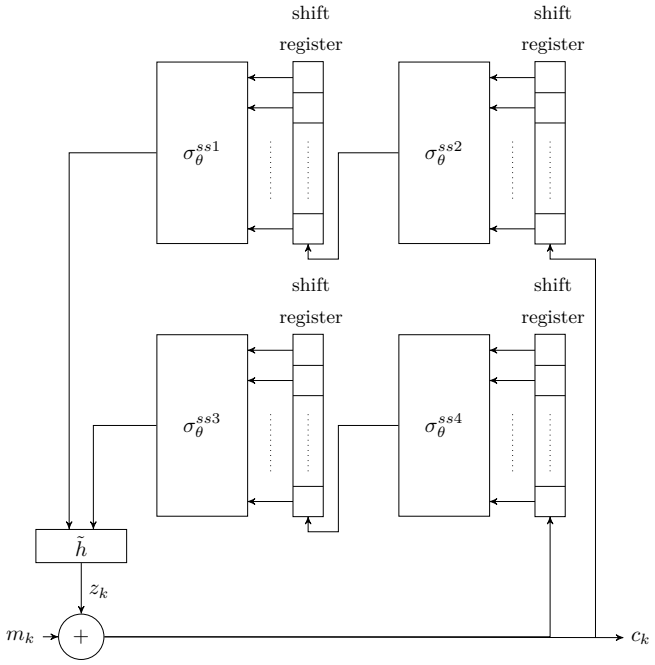

Figure 2.3: Serial and parallel automata in the architecture proposed by Maurer.

Later, Daemen and al. in [27] proposed another design based on the approach of Maurer where the use of serial and parallel automata had been lighten in structure involving boolean functions. In their design each component of the internal state is defined as a function of previous components; the first component being defined as the last ciphertext generated by the cipher. This gives rise to the so called Conditional Complementing Shift Register that was used to design the SSSC called KNOT. The architecture of KNOT [27] is implemented in a pipelined block cipher construction as a number of relatively simple rounds, called stages. This architecture allows very high speed encryption for hardware. In [57], it had been shown that KNOT presented some weaknesses that have been eliminated by improving it in a new SSSC called ϒΓ. We can also mention other known SSSCs that admit a similar architecture: HBB [96], SSS [51] and the family of MOUSTIQUE [29, 31]. They are described in Appendix A.2.

The resulting updating functions of all those constructions of SSSC were T-functions (see Subsection 2.3.2) or conditionally complementing shift registers (as in MOUSTIQUE).

The keystream bit is computed as a Boolean function of these statebits, possibly making use of pipelining (effectively adding bits to the internal state). This architecture guarantees that the keystream bit is completely determined by the previous M ciphertext bits. The situation of these architectures does not look very good as all concrete proposals have been broken and for the moment there is not a single unbroken dedicated self-synchronizing stream cipher. This motivates further investigations as proposed here.

2.3.1 Keystream generators for Self-Synchronizing Stream Ciphers

A well-admitted approach to generate the keystreams has been first suggest in [74]. It is based on the use of state automata with finite input memory. This is typically the case in the cipher MOUSTIQUE [31]. At the ciphering side, the automaton delivering the keystream takes the form:

$$\begin{cases} q_{k+1} = g_\theta(q_k, c_{k+b}) \\ z_{k+b} = h_\theta(q_k) \end{cases} \tag{2.6}$$

where q_k is the internal state. As previously stressed, the delay b is due to the fact that the output (also called filtering) function h is pipelined with an a architecture involving b layers.

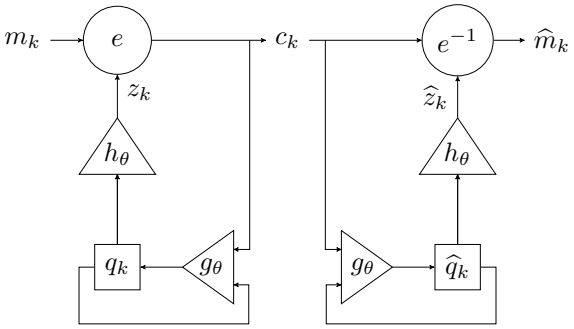

Figure 2.4: Automata architecture of SSSC.

If such an automaton has a finite input memory, it means that by iterating (2.6) a finite number of times, there exists a function l_θ and a finite integer M such that

$$q_k = l_\theta(c_{k+b-1}, \ldots, c_{k+b-M}) \tag{2.7}$$

and thus

$$z_{k+b} = h_\theta(l_\theta(c_{k+b-1}, \ldots, c_{k+b-M})) \tag{2.8}$$

Actually, the fact that the keystream symbol can be written in the general form involving a function σ_θ^{ss}

$$z_{k+b} \quad = \quad \sigma_\theta^{ss}(c_{k-l}, \ldots, c_{k-l'}) \tag{2.9}$$

is a common feature of all the SSSC. The quantities l and l' are integers in $\mathbb{Z}$. Equation (2.9) is the canonical equation.

Remark 2.3.1 *The outcome of implementing the recursive form (2.6) instead of directly implementing the canonical form (2.9) is that we can obtain complex nonlinear functions σ_θ^{ss} by implementing simpler nonlinear functions g_θ. The complexity results from the successive iterations which act as composition operations.*

At the deciphering side, the automaton takes the form:

$$\begin{cases} \widehat{q}_{k+1} = g_\theta(\widehat{q}_k, c_{k+b}) \\ \widehat{z}_{k+b} = h_\theta(\widehat{q}_k) \end{cases} \tag{2.10}$$

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

$$\widehat{q}_k = q_k \text{ and } \widehat{z}_{k+b} = z_{k+b} \tag{2.11}$$

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

To obtain a finite input memory feature, the solutions proposed in the open literature (see [26] for example), call for state transition functions g_θ in the form of shifts or T–functions (T for Triangle), which are functions that propagate dependencies in one direction only. One of the aim of this thesis is to show that it is possible to construct automata with finite input memory and state transition functions that admit a more general form than T–functions and to provide a systematic methodology of construction. The construction is based on the property of flatness, a structural property borrowed from control theory. This property is considered for the class of Linear Parameter-Varying (LPV) systems as motivated in Section 2.3.3.

2.3.2 Strict T-function

Design of SSSC which next state function are not strict T-function [63] remains a challenge. In [81], the authors established promising new ideas regarding the design of a new class of SSSC with a more general state resursive form (2.6). These ideas have been widely developed in Parriaux's Thesis [84] and led to a new design of SSSC. An effective construction of this class based on nilpotent semi-group approach had been proposed in [88]. An interesting classification of SSSC has been also established, showing that the next state function of an SSSC belongs to three categories of function:

1. strict T-functions
2. conjugate of strict T-functions
3. functions not based on strict T-functions

For T-functions, each variable depends only on previous variables. Hence, the function in (2.6) and (2.10) $g_\theta : \mathbb{F}^{n+1} \mapsto \mathbb{F}^n$ is a strict T-function if the output of its coordinate function g_θ^j depends only on the $j+1$ variables c_{k+b} and q_k^i, $i = 0, \ldots, j-1$:

$$g_\theta(q_k, c_{k+b}) = \begin{pmatrix} g^0(c_{k+b}) \\ g^1(q_k^0, c_{k+b}) \\ g^2(q_k^0, q_k^1, c_{k+b}) \\ \vdots \\ g^{n-1}(q_k^0, q_k^1, \ldots, q_k^{n-2}, c_{k+b}) \end{pmatrix} \tag{2.12}$$

T-functions are in general functions where the only Boolean operations are bit-oriented operations (additions, multiplications, subtractions, xor, or, and) and for which right shifts[1] or rotate operations are excluded.

[1] Note that left shifts are allowed since there are multiplication by 2.

The aim of the next section is to show that it is possible to design SSSC without T-functions when considering LPV systems as automata.

2.3.3 Particular case of LPV systems

The next proposition is essential to establish the correspondence between flat LPV systems and SSSC and is a central point to show that we can design a new architecture of SSSC.

Proposition 2.3.1 ([36]) *If the LPV system (1.1) has relative degree r and is flat, then the finite state automata given by*

$$\begin{cases} q_{k+1} &= P_{\rho(k:k+r)}q_k + B_{\rho(k)}(\mathfrak{T}^{r,0}_{\rho(k)})^{-1}y_{k+r} \\ z_{k+r} &= C_{\rho(k+r)}\prod_{l=k+r-1}^{k} A_{\rho(l)}q_k \end{cases} \tag{2.13}$$

along with

$$y_{k+r} = z_{k+r} + \mathfrak{T}^{r,0}_{\rho(k)}u_k \tag{2.14}$$

and

$$\begin{cases} \widehat{q}_{k+1} &= P_{\rho(k:k+r)}\widehat{q}_k + B_{\rho(k)}(\mathfrak{T}^{r,0}_{\rho(k)})^{-1}y_{k+r} \\ \widehat{z}_{k+r} &= C_{\rho(k+r)}\prod_{l=k+r-1}^{k} A_{\rho(l)}\widehat{q}_k \end{cases} \tag{2.15}$$

along with

$$\widehat{u}_{k+r} = (\mathfrak{T}^{r,0}_{\rho(k)})^{-1}(y_{k+r} - \widehat{z}_{k+r}) \tag{2.16}$$

define an SSSC.

Proof. If (1.1) has relative degree r and is flat, (1.8) holds and thus, (2.13) is well defined and can be identified with (2.6) while (2.15) is also well defined and can be identified with (2.10). The property (1.8), that is flatness, ensures that (2.13) and (2.15) are automata with finite input memory. The identification of (2.14) with (2.1) and the identification of (2.16) with (2.2) gives respectively the encryption and decryption functions. □ The following correspondences hold:

- u_k plays the role of m_k (plaintext symbol)
- y_k plays the role of c_k (ciphertext symbol)
- $z_{k+r} = C_{\rho(k+r)}\prod_{l=k+r-1}^{k} A_{\rho(l)}q_k$ is the keystream symbol of the cipher
- $\widehat{z}_{k+r} = C_{\rho(k+r)}\prod_{l=k+r-1}^{k} A_{\rho(l)}\widehat{q}_k$ is the keystream symbol used for deciphering.

- r plays the role of b (delay)
- the function $(z_{k+r}, u_k) \mapsto z_{k+r} + \mathcal{T}^{r,0}_{\rho(k)} u_k$ plays the role of e (encryption function)
- the function $(\widehat{z}_{k+r}, y_{k+r}) \mapsto (\mathcal{T}^{r,0}_{\rho(k)})^{-1}(y_{k+r} - \widehat{z}_{k+r})$ plays the role of d (decryption function)
- K plays the role of M (synchronization delay).

The nonlinearity is obtained by defining the values of the varying parameters $\rho^i(k)$ as nonlinear functions of the output y_k (or a finite number of shifts), implemented in the form of so-called S-Boxes φ_i: $\rho^i(k) = \varphi_i(y_k, y_{k-1}, \cdots)$

Remark 2.3.2 *Let us consider, from the system (1.1) the input u_k as the plaintext and the output y_k as the ciphertext of a stream cipher. The first equality in (1.2) shows that if the output y_k of (1.1) is flat, an interesting correspondence with (2.7) can be made. Indeed, x_k can be used as a quantity from which, through a filtering function, the keystream symbol z_k of an SSSC can be derived. The second equation shows that it is possible to recover the input u_k from a finite number of shifted outputs y_k. This is typically the central notion behind the decryption part of an SSSC.*

Remark 2.3.3 *The encryption and decryption functions e and d are quite simple. This is a common feature for SSSC. For example in the Boolean case, those functions are often nothing but the exclusive or. Actually, the security is essentially based on the properties of the keystream sequences.*

The next state transition function g_θ is not conjugated to a strict T-functions if the matrix $P_{\rho(k:k+r)}$ is not triangular. However, as stressed in Subsection 1.2.3, verifying and choosing non triangular matrices fulfilling (2.13) (or also (1.8)) is an intricate problem which is closely related to the notion of *mortality*. Furthermore, the problem under consideration here is more intricate since the matrices $P_{\rho(k:k+r)}$ are not given but should be chosen. In other words, we are not concerned with analysis but with synthesis. The method proposed here, that is the design a flat LPV system (1.1) followed by deriving the automaton (2.13) (which will be, because of its flatness property, with finite input memory and so self-synchronizing) constitutes an efficient alternative to a very challenging direct design of the automaton (2.13). This approach is new in cryptography and provides a general framework. However, from a control theory point of view, the issue of designing a flat LPV system is a new paradigm. Indeed, in automatic control, we are usually given a system and we have to check whether a system is flat or not and to check for the flat outputs.

The aim of the next section is to show that, using the graph approach, developed in Section 1.3 of Chapter 1, we can design SSSC based on flat LPV system. It is worth pointing out that those graphs are considered in the case where the relative degree of the LPV system is greater or equals 2. When $r = 1$ the only graphs that we can obtain correspond to systems whose next state functions are strict T-functions or conjugates of a strict T-function.

2.4 Construction of SSSC Based on Digraph

In this section, we tackle the problem of construction of SSSC by applying results that were developed in Chapter 1 and having in mind the correspondence between SSSC and LPV systems as shown in Subsection 2.3.3. More, we aim at designing a new class of SSSC where the state transition matrices P_ρ are not conjugated to a T-function.

2.4.1 Construction of flat LPV systems

Conditions **C0**-**C2** are instrumental for the construction of flat dynamical systems. Indeed, a systematic construction of digraphs fulfilling **C0**-**C2** can be derived. It had been published in [36] and is detailed below.

A digraph is parametrized by the triplet (n, r, n_a). The dimension n of the system (1.1) corresponds to the number of vertices of the graph minus one (the vertex assigned to the input). The relative degree r fulfilling (1.6) gives the number of edges in the main direct path. The integer n_a defines the number of edges in the digraph $\mathcal{G}(\Sigma_\rho)$ and can be freely chosen provided it is less than the maximal number n_M of edges. The expression of n_M depends on the construction and thus, will be given later on. The proposed construction of the digraph involves the following steps.

Step 1: The digraph $\mathcal{G}(\Sigma_\rho)$ corresponding to the system Σ_ρ of dimension n involves $n+1$ vertices. The input is assigned to the vertex denoted by $\mathbf{v^0}$. The other n vertices are denoted by $\mathbf{v^1}, \ldots, \mathbf{v^n}$. We want $\mathbf{v^r}$ to be the vertex that corresponds to the flat output.

Step 2: Add the edges $(\mathbf{v^i}, \mathbf{v^{i+1}})$ with $i = 0, \ldots, r-1$. The relative degree being r, there must have r edges which link $\mathbf{v^0}$ to $\mathbf{v^r}$. The resulting path will be called main direct path.

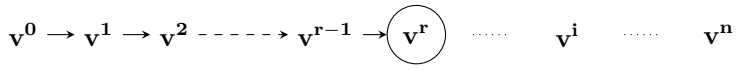

Figure 2.5: Digraph obtained after completion of Steps 1-2. The vertex $\mathbf{v^r}$ corresponds to the flat output.

After Step 2, Conditions **C0-C2** are fulfilled for $\mathbf{v^F} = \mathbf{v^r}$ and the resulting digraph is depicted in Figure 2.5. However, the corresponding dynamical system is quite trivial. The following steps provide a way of adding edges $(\mathbf{v^i}, \mathbf{v^j})$ while guaranteeing that Conditions **C0-C2** are still fulfilled.

Step 3: Add the edges $(\mathbf{v^i}, \mathbf{v^{i+1}})$ for $i = r, \ldots, n-1$ to avoid the situation with a vertex $\mathbf{v^j}$, $j = r+1, \ldots, n$ without predecessor. Indeed, if so, the dynamics of the corresponding vertex $\mathbf{v^j}$ would reduce to $x^j_{k+1} = 0$ and clearly would be useless.

Step 4: Add the edges that link the vertex $\mathbf{v^r}$ to any of the vertices of the graph (except the vertex related to the input), that is the edges $(\mathbf{v^r}, \mathbf{v^i})$, $i = 1, \ldots, n$.

Step 5: Consider the vertices $\mathbf{v^i}$ $i = 1, \ldots, r-1$, as starting vertices. For every vertex $\mathbf{v^i}$, $i = 1, \ldots, r-1$, add the directed edge $(\mathbf{v^i}, \mathbf{v^j})$ with $j = 1, \ldots, i$. Note that those edges introduce cycles, including cycles of order 1, but those cycles satisfy Condition **C2** since $\mathbf{v^i}$, $i = 0, \ldots, r-1$ belong to $V_{ess}(\mathbf{u}, \mathbf{v^r})$.

The graph obtained after Steps 1-5 is depicted in Figure 2.6.

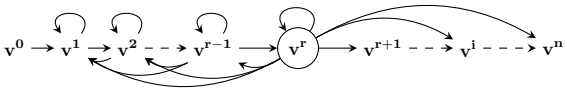

Figure 2.6: Graph obtained after Steps 1-5.

Step 6: Consider the vertices $\mathbf{v_i}$, $i = r+1, \ldots, n$. For every vertex $\mathbf{v_i}$, $i = r+1, \ldots, n$, add the directed edge $(\mathbf{v_i}, \mathbf{v_j})$ with $j = 1, \ldots, r$ and $j = i+2, \ldots, n$. Let us notice that this step generates cycles but Conditions **C2** is still satisfied.

The resulting digraph after completion of Steps 1-6 is depicted in Figure 2.7.

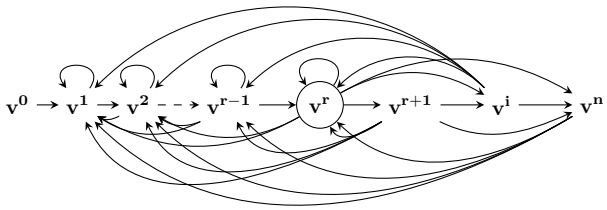

Figure 2.7: Graph obtained after completion of Steps 1-6.

A simple counting leads to the maximal number of edges n_M that can be added by following the construction at Steps 1-6. One has:

$$n_M = \frac{n \cdot (n+1)}{2} + r. \tag{2.17}$$

The number of edges n_a of the triplet (n, r, n_a) must satisfy $n_a \leq n_M$.

Step 7: Remove $n_M - n_a$ edges. Insofar as Steps 1-6 guarantee that Conditions **C0-C2** are fulfilled regardless whether the edges are actually added or not, Conditions **C0-C2** are still preserved after arbitrarily removing edges of the graph.

Remark 2.4.1 *By considering the flat LPV system related to the graph of Figure 2.7, it follows from Remark 1.3.6 that the integer K involved in Theorem 1.2.1 is bounded by the dimension n*

of this flat LPV system. Hence the synchronization delay M of the flat LPV system is bounded by n.

2.4.2 Completion of the design

Let $\mathcal{G}(\Sigma_\rho)$ be a digraph characterized by the triplet (n, r, n_a). We can derive the matrices I_A and I_B of the structured linear system $\mathcal{G}(\Sigma_\Lambda)$, from the adjacency matrix. The adjacency matrix, denoted with $\mathcal{I}$, of the digraph $\mathcal{G}(\Sigma_\rho)$, is the $(n+1)\times(n+1)$ matrix whose each entry $(a)_{ij}$ is defined as follows for $1 \leq i, j \leq n$

$$(a)_{ij} = \begin{cases} 1 \text{ if there exists an edge from } \mathbf{v^j} \text{ to } \mathbf{v^i} \\ 0 \text{ otherwise} \end{cases} \tag{2.18}$$

Hence, the adjacency matrix associated to $\mathcal{G}(\Sigma_\rho)$ is given by:

$$\mathcal{I} = \left(\begin{array}{c|c} 0 & I_B^T \\ \hline 0 & \\ \vdots & I_A^T \\ 0 & \end{array}\right) \tag{2.19}$$

where I_A^T and I_B^T stand respectively for the transpose of I_A and I_B.

The varying parameter $\rho^4(k)$ is in practice implemented in the form of a so-called S-Box whose entry is the flat output y_k and possibly a finite number of backwards iterates. By construction, any nonlinearity would lead to a flat LPV system and so to an SSSC. The approach proposed here gives thereby a family of SSSC. The design of a specific SSSC is completed by deriving the equations of Proposition 2.3.1. Actually, it can be shown (see next subsection) that even for this simple example, the state transition matrix $P_{\rho(k:k+2)}$ is non triangular, which corroborates that this method allows to provide a novel class of SSSC. Furthermore those SSSC are really simple to build owing to the graph approach associated to LPV system.

2.4.3 Basic construction of a flat LPV-based SSSC

We now show that an SSSC involving state transition functions more general than T–functions can be obtained even in a very simple case. Let us consider the graph depicted in Figure 2.8 which results from the construction given in the Subsection 2.4.1 for $n = 2$ and $n_a = n_M = 5$ (maximal number of edges) and $r = 2$. The relative degree r being equal to 2, it means that the component x^2 of the state vector of the corresponding LPV system (1.1) will be the flat output y_k, that is the ciphertext. Hence, $C_{\rho(k)} = C = [0\ 1]$ and $D_{\rho(k)} = 0$.

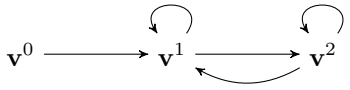

Figure 2.8: Digraph obtained for $n = 2$, $r = 2$, $n_a = n_M = 5$. The flat output is $y^F = x^2$ and corresponds to the vertex $\mathbf{v^2}$.

The adjacency matrix $\mathfrak{I}$ of this graph and the structured matrices I_A and I_B are

$$\mathfrak{I} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \; I_A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \; I_B = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{2.20}$$

Each of the entries of I_A and I_B can be potentially replaced by a realization $\rho^i(k)$ to construct the matrices $A_{\rho(k)}$ and $B_{\rho(k)}$ of (1.1). Let us choose the functions ρ^i $i = 1, 2, 3$ of the first three entries of A equal to 1. This finally leads to the following matrices $A_{\rho(k)}$ and $B_{\rho(k)}$:

$$A_{\rho(k)} = \begin{pmatrix} 1 & 1 \\ 1 & \rho^4(k) \end{pmatrix} \text{ and } B_{\rho(k)} = B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

As stressed in Section 1.1, the varying parameter $\rho^4(k)$ is the output of an S-Box whose entry is the flat output y_k and possibly a finite number of backwards iterates.

To complete the design of the decipher (see (2.13) of Proposition 2.3.1), let us calculate the matrix $P_{\rho(k)}$ (1.9) governing the state transition function of the automaton (2.13). One has $P_{\rho(k:k+2)} = A_{\rho(k)} - B \cdot C \cdot A_{\rho(k+1)} \cdot A_{\rho(k)}$. One obtains

$$P_{\rho(k:k+2)} = \begin{pmatrix} -\rho^4(k+1) & -\rho^4(k) \cdot \rho^4(k+1) \\ 1 & \rho^4(k) \end{pmatrix}$$

Let us point out that (1.8) holds for $K = 2$. Indeed, it's a simple matter to see that $P_{\rho(k+1:k+3)} P_{\rho(k:k+2)} = 0$. It corroborates that y_k is a flat output (but it is, by construction of the digraph).

It remains to show that the matrix $P_{\rho(k:k+2)}$ is not conjugate to a shift or to a T–function. To this end, it must be shown that the matrices $P_{\rho(k:k+2)}$, for almost every realization ρ, cannot be simultaneously triangularizable. The characteristic polynomial of $P_{\rho(k:k+2)}$ is given by $N(X) = X\big(X + (\rho^4(k+1) - \rho^4(k))\big)$. Hence, the eigenvalues of $P_{\rho(k:k+2)}$ are 0 and $\rho^4(k) - \rho^4(k+1)$. It follows that the related eigenspace to the eigenvalues of $P_{\rho(k:k+2)}$ is spanned by:

$$V_1 = \begin{pmatrix} -\rho^4(k) \\ 1 \end{pmatrix} \quad \text{and} \quad V_2 = \begin{pmatrix} -\rho^4(k+1) \\ 1 \end{pmatrix}$$

And yet, a necessary condition for simultaneous triangularization (see Theorem B.1.2) is that the matrices $P_{\rho(k:k+2)}$ share a common eigenvector for any realization ρ. As a result, if the

parameter $\rho^4(k)$ varies, the matrices $P_{\rho(k:k+2)}$ do not fulfill such a requirement. Clearly, even in this very trivial example, we have shown that an SSSC with state transition function different from a T–function can be obtained.

Let us notice that the graph approach that is proposed here provides T-function-based next-state transition function when the relative degree r of the LPV system equals 1.

The next-state function g_θ of a LPV-based SSSC is a T-function if and only if the inverse transition matrix $P_{\rho(k:k+r)}$ is triangular. One has:

$$q_{k+1}^i = g_\theta^i(q_k, c_k) = P_{\rho(k:k+r)}[i] \cdot q_k + \delta_r^i c_k$$

where δ_r^i equals 1 if $r = i$ and 0 if $r \neq i$. $P_{\rho(k:k+r)}[i]$ denotes the i^{th} row of the matrix $P_{\rho(k:k+r)}$. It follows that the output of each $g_\theta^i(c_k, q_k),\ i = 1, \ldots, n$ depends only on $q_k^0, \ldots, q_k^{i-1}$ if and only if $P_{\rho(k:k+r)}$ is a triangular matrix.

Theorem 2.4.1 *Let Σ be a flat LPV system (1.1) of relative degree $r = 1$. Then the next-state function f of an LPV-based SSSC (1.1) is conjugated to a T-function.*

Proof. From (1.9), if $r = 1$, it follows that:

$$P_{\rho(k:k+1)} = A_{\rho(k)} - (\mathfrak{T}_{\rho(k)}^{1,0})^{-1} B_{\rho(k)} C_{\rho(k+1)} A_{\rho(k)}.$$

As a consequence, the rows of $P_{\rho(k:k+1)}$ are the same as the ones of $A_{\rho(k)}$ except the first row which coefficients are zero:

$$P_{\rho(k:k+1))} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & a_{2,2} & \cdots & a_{2,n} \\ a_{31} & a_{3,2} & \cdots & a_{3,n} \\ & & \vdots & \\ a_{n1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix}$$

with $A_{\rho(k)} = (a_{i,j})_{1 \leq i,j \leq n}$.

Each vertex $\mathbf{v}^i$ is associated to the component x^i. Since $r = 1$, according to the graph construction of Subsection 2.4.1, the only possible edges in the graph $\mathcal{G}(\Sigma_\rho)$ are the ones that start from $\mathbf{v}^1$ or end at $\mathbf{v}^1$, and those that starting from each vertex towards its following vertices. Thus, $\mathcal{G}(\Sigma_\rho)$ has the form:

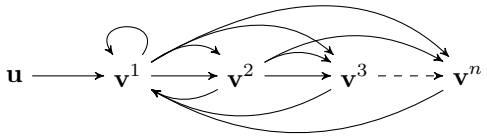

Figure 2.9: Graph construction related to a flat LPV system of relative degree $r = 1$.

It follows then that the reduced $(n-1) \times (n-1)$ matrix obtained by removing the first row of $A_{\rho(k)}$ is strictly lower triangular: that is $P_{\rho(k:k+1)}$ is strictly lower triangular. □ □

We give in the next section, a more general example of matrix P_ρ that is not conjugated to a T-function. To show that, we use Algorithm 1 known as Simultaneous Triangularization Algorithm (STA) provided in [39].

2.5 Examples of construction of LPV-based SSSC without T-functions

2.5.1 Example 1

Let us consider the graph of Figure 2.8, we give an example of matrix $A_{\rho(k)}$ with non-constant coefficients in a finite field $\mathbb{F} = \mathrm{GF}(16)$. The state matrix $A_{\rho(k)}$ related to this graph is a matrix of dimension 2 given by

$$A_{\rho(k)} = \begin{pmatrix} \rho^1(k) & \rho^2(k) \\ 1 & \rho^3(k) \end{pmatrix}$$

where

$$\begin{array}{rccc} \rho^i(k): & \mathrm{GF}(16)^s & \longrightarrow & \mathrm{GF}(16) \\ & c_{k-1}, \ldots, c_{k-s} & \mapsto & y_k \end{array}$$

is assumed to be a surjective non-linear function. The inverse transition matrix $P_{\rho(k:k+2)}$ (1.9) that characterizes the decryption and then the self-synchronization property, is given by

$$P_{\rho(k:k+2)} = A_{\rho(k)} + B \cdot C \cdot A_{\rho(k+1)} \cdot A_{\rho(k)}$$

Then, the rows of the matrix $P_{\rho(k)}$ are inferred as

$$P_{\rho(k:k+2)}[1] = A_{\rho(k)}[1] + A_{\rho(k+1)} \cdot A_{\rho(k)}[2]$$

$$P_{\rho(k:k+2)}[2] = A_{\rho(k)}[2]$$

Hence,

$$P_{\rho(k:k+2)} = \begin{pmatrix} \rho^3(k+1) & \rho^3(k)\cdot\rho^3(k+1) \\ 1 & \rho^3(k) \end{pmatrix}$$

Since, $\rho^3(k)$ is surjective, we can assume that, after a long transient time, $P_{\rho(k:k+2)}$ takes all the value in GF(16). Thus we need to find two instances of $P_{\rho(k:k+2)}$ that are not simultaneous triangularizable to prove that $P_{\rho(k:k+2)}$ is not simultaneous triangularizable regardless to $\rho(k)$.

Let

$$A_1 = \begin{pmatrix} \alpha^3+1 & \alpha+1 \\ 1 & \alpha^2+\alpha \end{pmatrix} \qquad A_2 = \begin{pmatrix} \alpha^2+\alpha+1 & \alpha^3+\alpha^2+1 \\ 1 & \alpha^3 \end{pmatrix}$$

We use a primitive element of GF(16) given by α with $\alpha^4 = \alpha+1$. The matrix A_1 is obtained by considering $\rho^3(k) = \alpha^2+\alpha,\ \rho^3(k+1) = \alpha^3+1$ and the matrix A_2 is obtained for $\rho^3(k) = \alpha^3,\ \rho^3(k+1) = \alpha^2+\alpha+1$. We show that A_1 and A_2 are not simultaneous triangularizable. Note that we need to iterate the STA (see Algorithm 1 of Appendix B) until we find no common eigenvectors for A_1 and A_2. The example is minimalist due to the considered graph.

For $i = 1, 2$, we denote by $P_{A_i}(X)$ the characteristic polynomial, $Sp(A_i)$ the spectrum and $\mathcal{V}(A_i)$ the eigenvectors of A_i.

- **Compute the eigenvectors of A_1:** $P_{A_1}(X) = X(X+\alpha^3+\alpha^2+\alpha+1)$. Then $Sp(A_1) = \{0, \alpha^3+\alpha^2+\alpha+1\}$ and

$$\mathcal{V}(A_1) = \left\{ \lambda_1 \begin{pmatrix} \alpha^2+\alpha \\ 1 \end{pmatrix},\quad \lambda_2 \begin{pmatrix} \alpha^3+1 \\ 1 \end{pmatrix} \right\} \qquad \lambda_1, \lambda_2 \in \text{GF(16)}$$

- **Compute the eigenvectors of A_2:** $P_{A_2}(X) = X(X+\alpha^3+\alpha^2+\alpha+1)$. Then $Sp(A_1) = \{0, \alpha^3+\alpha^2+\alpha+1\}$ and

$$\mathcal{V}(A_2) = \left\{ \lambda_3 \begin{pmatrix} \alpha^3 \\ 1 \end{pmatrix},\quad \lambda_4 \begin{pmatrix} \alpha^2+\alpha+1 \\ 1 \end{pmatrix} \right\} \qquad \lambda_3, \lambda_4 \in \text{GF(16)}$$

It follows that A_1 and A_2 do not admit any common eigenvector. As a conclusion the transition function related to the LPV-based SSSC is not conjugate to a T-function.

2.5.2 Example 2

The following example is to show how we can apply the STA algorithm 1 on a set of matrices $P_{\rho(k:k+r)}$, obtained from a graph that generates a flat system, to show that the set is not simultaneous triangularizable. For that, we show that there exist two instances of $P_{\rho(k:k+r)}$, that do not have a common eigenvector. The LPV system is of dimension $n = 6$, relative degree $r = 3$. The matrix $P_{\rho(k:k+3)}$ is obtained from the graph on Figure 2.10. For each $k \in \mathbb{N}$, the entries of $P_{\rho(k:k+3)}$ are given by polynomials $\varphi_i^j : \mathbb{F}^s \longrightarrow \mathbb{F}$, with $\mathbb{F} = GF(16)$. The expression of $P_{\rho(k:k+3)}$ is given by:

$$P_{\rho(k:k+6)} = \begin{pmatrix} \varphi_1^1 & \varphi_1^2 & \varphi_1^3 & \varphi_1^4 & \varphi_1^5 & \varphi_1^6 \\ 1 & \varphi_2^2 & \varphi_2^3 & 0 & \varphi_2^5 & \varphi_2^6 \\ 0 & 1 & 0 & \varphi_3^4 & \varphi_3^5 & \varphi_3^6 \\ 0 & 0 & \varphi_4^3 & \varphi_4^4 & 0 & 0 \\ 0 & 0 & 0 & \varphi_5^4 & 0 & 0 \\ 0 & 0 & 0 & \varphi_6^4 & \varphi_6^5 & 0 \end{pmatrix}$$

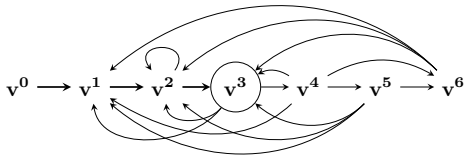

Figure 2.10: Graph structure related to the matrix $P_{(\rho:k+6)}$

As φ_i^j take its values in $\mathbb{F}$, it suffices to produce two instances of $P_{(\rho:k+6)}$ and show that these two instances do not share a common vector.
For example, consider the following two instances of $P_{(\rho:k+6)}$:

$$P_{\rho(k_1)} = \begin{pmatrix} \alpha^3{+}1 & 0 & \alpha^3{+}\alpha^2{+}\alpha{+}1 & \alpha{+}1 & \alpha^2 & \alpha^3{+}\alpha^2{+}\alpha \\ 1 & \alpha^3{+}1 & 1 & 0 & \alpha^3{+}\alpha^2{+}\alpha & 0 \\ 0 & 1 & 0 & \alpha^2{+}\alpha & \alpha^3{+}\alpha^2 & \alpha^3{+}\alpha^2{+}1 \\ 0 & 0 & \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha^3{+}\alpha^2{+}1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha{+}1 & 0 \end{pmatrix}$$

and

$$P_{\rho(k_2)} = \begin{pmatrix} \alpha^3{+}\alpha^2{+}\alpha{+}1 & \alpha^3 & \alpha^3{+}\alpha^2{+}\alpha{+}1 & \alpha^3{+}\alpha{+}1 & 1 & \alpha^2 \\ 1 & \alpha^2{+}\alpha{+}1 & \alpha^3{+}\alpha^2{+}1 & 0 & \alpha^3{+}\alpha^2{+}\alpha & 0 \\ 0 & 1 & 0 & \alpha^3{+}\alpha^2{+}\alpha{+}1 & \alpha^2{+}\alpha & \alpha^3{+}\alpha^2{+}1 \\ 0 & 0 & \alpha^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha{+}1 & 0 & 0 \\ 0 & 0 & 0 & \alpha^2 & \alpha^2{+}\alpha{+}1 & 0 \end{pmatrix}$$

it can be checked that the only eigenvalue of $P_{\rho(k_1)}$ is zero which eigenspace is spanned by the vector $v = (1, \alpha, 0, 0, 0, \alpha^3)$. But v is not an eigenvector of $P_{\rho(k_2)}$.
Indeed, $P_{\rho(k_2)} \times v = (\alpha^3 + \alpha, \alpha^3 + \alpha^2 + \alpha + 1, 0, 0, 0, 0)$. The characteristic polynomial of the matrix $P_{\rho(k_2)}$ is not split on $\mathbb{F}$ and its eigenvalues belong to an extension of $\mathbb{F}$.

2.6 Test platform and specifications

The graph approach used to derive a flat LPV system provides a convenient way to design an SSSC with arbitrary size of the internal state. Let us recall that the size of the SSSC internal state is the dimension n of the LPV system. From a practical point of view, a large dimension must be chosen to allow a sufficiently complex architecture for the sake of security. The proposed graph-based approach is well suited to deal with high dimensions. The dimension $n = 40$, which is compatible with security issues has been tested with success on an arduino MEGA 2560 card (see Figure 2.12). Indeed, as the internal state components are 4-bit size, a dimension 40 would provide a security level of 80 bits against time-memory trade-off attack.

The relative degree is given by $r = 3$. The choice of the value of the relative degree r is such that, the powers of the matrix $P_{\rho(k:k+r)}$ offer a good diffusion delay. By definition the least value s_0 such that the matrix $P^{s_0}_{\rho(k:k+r)}$ does not have any zero coefficient is called diffusion delay

[4, 10], where $P^{s_0}_{\rho(k:k+r)}$ stands for the power of $P_{\rho(k:k+r)}$ to s_0. Another crucial point concerning the relative degree is the increase in the algebraic degree of the polynomial coefficients of the powers of the matrix $P_{\rho(k:k+r)}$. The higher the value of r and the faster the increase of the algebraic degree. However, according to (1.9), a higher value of r would make the computation of the coefficients of $P_{\rho(k:k+r)}$ prohibitive for hardware implementation, since this computation involves products of S-Box. A value $r = 3$ provides a better compromise diffusion delay and coefficients complexity of the matrix $P_{\rho(k:k+r)}$.

The number of edges is given by $n_a = 782$, that is the number of non-zero coefficients in the matrix $A_{\rho(k)}$ minus 1. Hence, we set 80 coefficients of the matrix $A_{\rho(k)}$ as variable coefficients (or polynomial coefficients) that correspond to the same S-Box parametrized by 4-bit secret keys. Then, the subkeys size is 320 bits. The remaining 701 non-zero coefficients of the matrix are set to 1. The S-Box is defined as the inverse function on the Galois field $GF(16)$ plus α^2 that is $x \longmapsto \dfrac{1}{x} + \alpha^2$ where α is a primitive element of $GF(16)$.

Figure 2.11 illustrates the required number n_a of edges (that is the number non-zero coefficients) to avoid coefficients equal zero in the matrix $P_{\rho(k:k+r)}$ after n iterations.

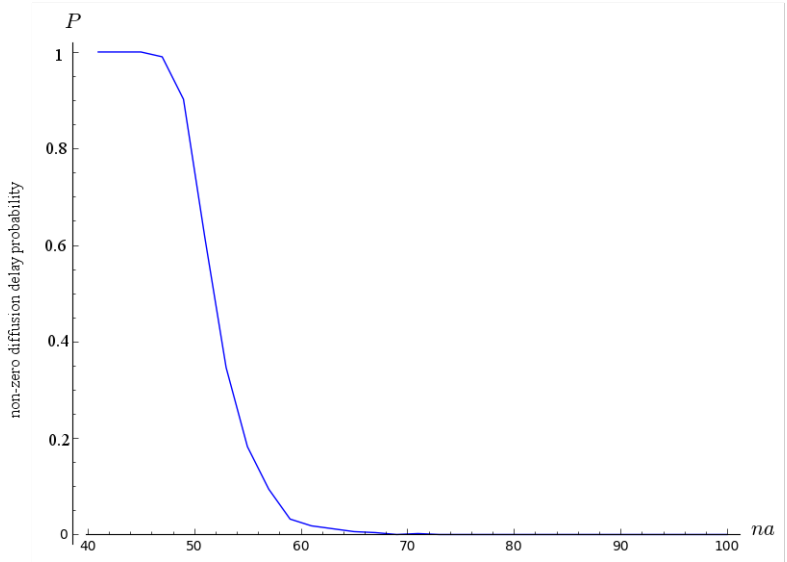

Figure 2.11: Non-zero diffusion delay probability with respect to the number n_a of non-zero coefficients in the inverse transition matrix $P_{\rho(k:k+3)}$.

We do not provide the complete specification here. The test platform is made of three wired arduino MEGA 2560 card.

Figure 2.12: Arduino mega 2560 card

The supervisor depicted in Figure 2.13 describes the interconnection between the cards. Card 1 and card 3 are wired to temperature sensors that collect the plaintext data. Those cards can encrypt the data and send them to the card 2 that decrypts them. The platform also allows to perform desynchronization between the cards by setting randomly the internal state of the card that plays the role of cipher. It is also possible to alter the ciphertext that is sent to the decipher: this also produces a desynchrozination. However there is a maximum delay of 40 iterations to reach a resynchronization between the cards.

2.7 Conclusion

A systematic and general construction of Self Synchronizing Stream Ciphers based on flat Linear Parameter Varying (LPV) dynamical systems has been proposed. It is based on algebraic conditions guaranteeing flatness of the LPV system and so the self-synchronizing property, those conditions being interpreted in terms of the structure of the graph associated to the LPV dynamical system. It has been shown that such an approach allows to enlarge the existing classes of SSSC, more precisely, to obtain non triangular SSSC. Two control-theoretical issues have been treated to this end: as a new paradigm, the construction of flat dynamical systems and the notion of mortality.

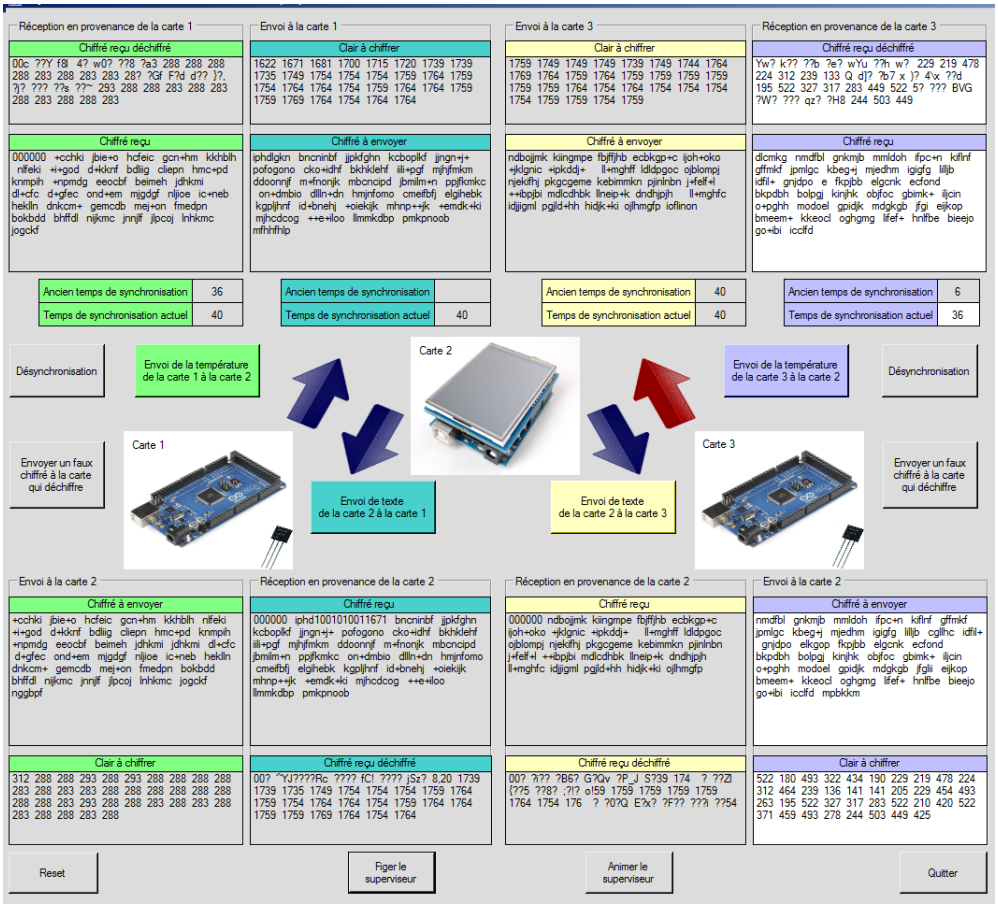

Figure 2.13: Supervisor

Chapter 3

Spectral Approach for Correlation Power Analysis

This chapter provides a new approach to perform Correlation Power Analysis (CPA) attack. Power analysis attacks are side channel attacks based on power consumption measures on a device running a cryptographic algorithm with a CMOS-technology-based circuitry. Unlike most of CPA attacks that are based on statistical attacks, we propose a new approach based on spectral analysis. The interest lies in the reduction of the attack complexity. The complexity is quasi linear in the size of the table of values of the S-Box whereas it is quadratic with statistical attacks. It is shown that it can be easily extended to a so-called multidimensional attack. The results in this chapter are published in [50].

Contents

3.1 Introduction

The Correlation Power Analysis (CPA) is a method that allows to recover the secret information (usually the secret key) embedded in the silicon of an electronic component [18]. It consists in measuring the power consumption while running operations that involve the secret information. This method has been introduced in 2004 by researchers of Gemplus Company (Eric Brier, Christophe Clavier and Francis Olivier) in [18]. The attack follows the work of Paul Kocher proposed in 1999 [67]. The electric circuits that perform the computation in the processor are designed from a technology known as CMOS (Complementary Metal Oxide Semiconductor) [5]. A general description of this technology is provided in [73]. The outcome of CMOS architectures (see Figure 3.1) lies in fast transitions but, on the other hand, they are sensitive to power consumption or electromagnetic leakage.

Power analysis attacks are based on the general principle that the instantaneous power consumption of a cryptographic device, based on CMOS technology, depends on the data it processes and on the operation it performs [77]. During a symmetric encryption, those operations are in general processed by a nonlinear function called S-Box parametrized by a secret parameter [33, 1]. Usually, CPA attacks computes a Pearson correlation coefficient for each subkey hypothesis that leads to a quadratic overall complexity [100, 91]. The attack can be applied to any encryption algorithm involving S-Boxes. This is usually the case of Self-Synchronizing Stream Cipher whose next-state transition function involves an S-Box and that guarantees confusion properties. To show the feasibility of the attack, we experiment it on the AES S-Box and the LPV-based SSSC that we have designed.

3.2 Preliminaries on Spectral Analysis

Having in mind a CPA attack based on spectral considerations, preliminaries on Fourier analysis must be recalled. It is worth pointing out that the suggested attack is based on a power consumption measurement on a component that processes binary data as input. Hence, the measurement can be modeled as a real-valued function over the set of binary words. This section is devoted to prerequisites on Fourier analysis of this class of functions.

Let Φ be the set of real valued functions over the set of n-dimensional binary words:

$$\Phi = \left\{\varphi : \{0,1\}^n \to \mathbb{R}\right\}$$

For any two functions φ and ψ in Φ, let us define the scalar product of φ and ψ as:

$$\langle \varphi, \psi \rangle = \sum_{x \in \{0,1\}^n} \varphi(x)\psi(x).$$

This scalar product is a symmetric bilinear form that confers to this set the structure of a 2^n-dimensional Euclidean vector space over $\mathbb{R}$.

The norm associated to this scalar product is:

$$\|\varphi\| = \sqrt{\langle \varphi, \varphi \rangle} = \sqrt{\sum_{x \in \{0,1\}^n} \varphi(x)^2}$$

The norm of a function φ in Φ is called the energy of φ.

The well-known Cauchy-Schwarz inequality holds:

$$\forall \varphi, \psi \in \Phi, \ \left|\langle \varphi, \psi \rangle\right| \leq \|\varphi\| \times \|\psi\| \tag{3.1}$$

where $\left|\cdot\right|$ stands for the absolute value.

The canonical basis of the space Φ is the family of characteristic functions of singletons which are by definition, for all vectors $u \in \{0,1\}^n$, the functions denoted by δ_u and defined by:

$$\delta_u : \ x \mapsto \begin{cases} 1 & \text{if } x = u \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

This basis is clearly orthonormal according to the above scalar product.

Each function $\varphi \in \Phi$ can be expressed in this basis as:

$$\varphi = \sum_{u \in \{0,1\}^n} \varphi(u) \delta_u.$$

Another basis of the space Φ is the basis of the so-called Walsh functions.

Definition 3.2.1 (Walsh functions [49]) *The Walsh functions are the functions of Φ defined for any $u \in \{0,1\}^n$ by :*

$$\chi_u : \ x \mapsto \frac{1}{\sqrt{2^n}} (-1)^{u \cdot x}$$

where $x \cdot u = u_1 x_1 + u_2 x_2 + \cdots + u_n x_n$ is the dot product over the space $\mathbb{F}_2^n$ of n–dimensional binary words over the two elements field $\mathbb{F}_2$.

The Walsh functions are pairwise orthogonal. They are presented here with a normalization coefficient equal to $1/\sqrt{2^n}$ such that they provide an orthonormal basis as stated in the following proposition.

Proposition 3.2.1 *The family $(\chi_u)_{u \in \{0,1\}^n}$ of Walsh functions is an orthonormal basis of Φ.*

Proof. Let u and v be two n-dimensional binary vectors. One has:

$$\langle \chi_u, \chi_v \rangle = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot (u+v)}.$$

If $u = v$ then $u + v = 0$. The above sum has 2^n terms, all equal to 1. Then $\langle \chi_u, \chi_u \rangle = 1$.

If $u + v \neq 0$, it has a non-zero component. Let $u_i + v_i$ be this component. Then, the sum over all the vectors x for which $x_i = 0$ is the opposite of the sum over all the vectors x for which $x_i = 1$. It results that if $u \neq v$ then $\langle \chi_u, \chi_v \rangle = 0$. □

This family allows to express any function $\varphi \in \Phi$ in the basis of Walsh functions

$$\varphi = \sum_{u \in \{0,1\}^n} \langle \varphi, \chi_u \rangle \chi_u.$$

Definition 3.2.2 (Fourier Transform) *The Fourier spectrum of $\varphi \in \Phi$ is the family of coefficients $\left(\langle \varphi, \chi_u \rangle\right)_{u \in \{0,1\}^n}$, of the expression of φ in the basis of Walsh functions, and the Fourier transform of φ is the function in Φ defined on $\{0,1\}^n$ as:*

$$u \mapsto \widehat{\varphi}(u) = \langle \varphi, \chi_u \rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} \varphi(x)(-1)^{u \cdot x}.$$

The Fourier transform expresses a change of basis. Thus, the transformation is linear. Moreover, it is isometric as stated in the following proposition.

Proposition 3.2.2 *For any functions φ and ψ in Φ, one has:*

$$\langle \varphi, \psi \rangle = \langle \widehat{\varphi}, \widehat{\psi} \rangle.$$

Proof.

$$\langle \widehat{\varphi}, \widehat{\psi} \rangle = \sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{2^n}} \sum_{u \in \{0,1\}^n} \varphi(u)(-1)^{u \cdot x} \frac{1}{\sqrt{2^n}} \sum_{v \in \{0,1\}^n} \psi(v)(-1)^{v \cdot x}$$

By inverting the summation order, it follows that

$$\langle \widehat{\varphi}, \widehat{\psi} \rangle = \frac{1}{2^n} \sum_{u \in \{0,1\}^n} \sum_{v \in \{0,1\}^n} \varphi(u)\psi(v) \sum_{x \in \{0,1\}^n} (-1)^{(u+v) \cdot x}.$$

As this latter sum equals 2^n if $u = v$ and equals 0 elsewhere, the result holds. □

As a direct consequence of this proposition, it follows that for all functions φ in Φ, the so-called Parseval equality holds (see [22])) and expresses the energy conservation law:

$$\|\varphi\| = \|\widehat{\varphi}\| \tag{3.3}$$

Proposition 3.2.3 (Effect of a translation [50]) *For any vector a in $\{0,1\}^n$, let $\tau_a : t \longmapsto t + a$ be the translation of vector a. Let φ be a function in Φ, then for all vectors $u \in \{0,1\}^n$, one has:*

$$\widehat{\varphi \circ \tau_a}(u) = (-1)^{u \cdot a} \widehat{\varphi}(u).$$

Proof.

$$\widehat{\varphi \circ \tau_a} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} \varphi(x+a)(-1)^{u \cdot x} = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} \varphi(y)(-1)^{u \cdot (y+a)}$$
$$= (-1)^{u \cdot a} \widehat{\varphi}(u) \tag{3.4}$$

□

Proposition 3.2.4 (Fourier Transform of a constant [50]) *Let $k \in \{0,1\}^n$. The Fourier transform of the constant function $x \longmapsto k$ is:*

$$\widehat{k} = k\sqrt{2^n}\delta_0.$$

Proof. By definition,

$$\widehat{k}(u) = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} k(-1)^{u \cdot x} = \frac{k}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{u \cdot x}.$$

The latter sum equals 2^n if $u = 0$ and 0 elsewhere, and the result holds. □

3.3 Modelling the Attack

3.3.1 Physical principles

Nowadays, digital circuits are often designed with CMOS (Complementary Metal Oxide Semiconductor) technology. The main characteristic lies in the output stage of the logical gates that involves a pair of Field Effect Transistors (FET) with opposite polarity. The transistors are combined symmetrically (push-pull architecture) such that they switch between two states: on and off. The outcome of such architecture is that in a steady state mode, the current consumption is almost null. On the other hand, when the output state of a gate changes, a parasitic capacitor discharges in the complementary parasitic capacitor. This leads to power consumption while a transition occurs.

As a consequence, the following assumption, which is the core idea of CPA is well-admitted:

Assumption 1 *The power consumption of CMOS circuit is proportional to the number of logic gates that switch [5].*

From this assumption, two models of consumption can be considered: the Hamming weight model and the Hamming distance model [18]. For a given calculus, the first model assumes that the gates switch from the state 0 to the result of the calculus. It turns out that the model is well suited for software implementations. The second model assumes that the gates switch from an initial state that corresponds to the former calculus result to the state corresponding

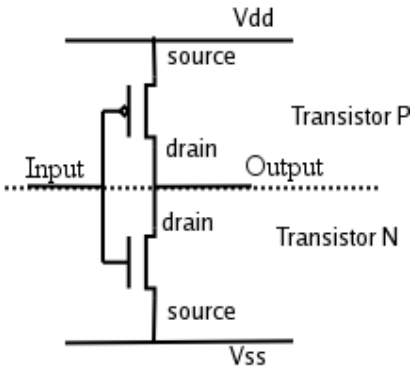

Figure 3.1: CMOS technology

to the result of the current calculus. It turns out that this model is more suitable for hardware implementations.

3.3.2 Target of the Attack

Many symmetric ciphering algorithms such as DES [33] or AES [1] for instance are based on alternate stages of linear and nonlinear calculations. The nonlinear stage is most often implemented in the form of S-Boxes, each of one corresponding to a nonlinear function f from $\{0,1\}^n$ to $\{0,1\}^m$, whose input is the exclusive or (XOR) of a data $d \in \{0,1\}^n$ and of an unknown secret subkey $k^\star \in \{0,1\}^n$, and returns a quantity $y \in \{0,1\}^m$, that is

$$y = f(d + k^\star).$$

Many strategies exist and allow to recover the secret key $k^\star$. A well know approach provided in [73] is based on statistical means and uses Pearson correlation coefficients.

3.3.3 General CPA approach

The strategy used in the CPA attack consists of performing the attack in 5 steps [73] as illustrated in Figure 3.2.:

Step 1: choose intermediate result of the executed algorithm and hypothetical keys

Step 2: measure the power consumption and record the *traces*

Step 3: calculate hypothetical values from the intermediate values

Step 4: mapping the hypothetical values to power consumption values

Step 5: correlate the hypothetical power consumption values with the power traces.

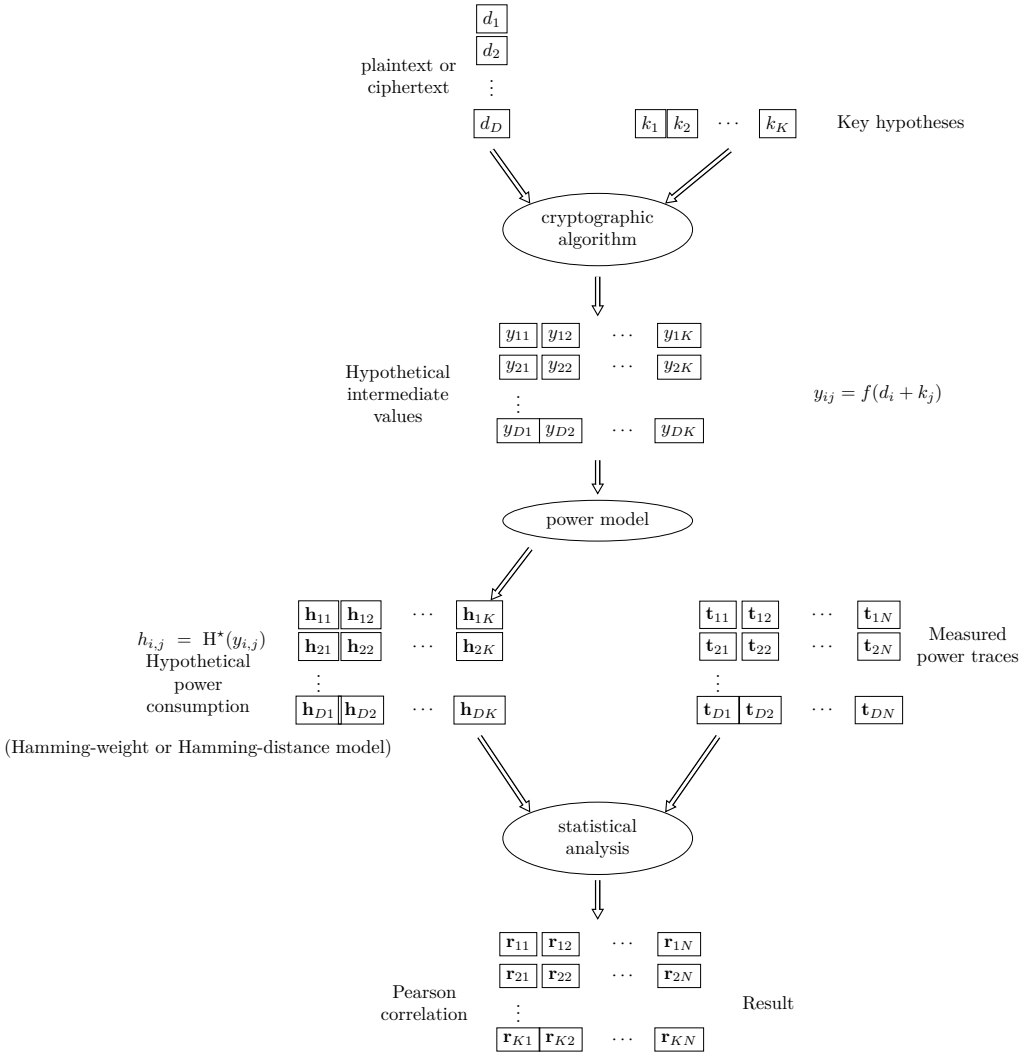

Figure 3.2: Block diagram of the different Steps of a general CPA attack.

In **Step 1**, the adversary chooses D random values $(d_1, d_2, \ldots, d_D)$ (plaintext or ciphertext) that will be used as input of the executed algorithm. He also chooses hypothetical values of keys $k_1, \ldots, k_K$ that will be correlated to the right key $k^\star$.

In **Step 2**, the adversary records, using an oscilloscope, the power consumptions or *traces* related to each random value d_i. Each trace is made of N samples and is denoted by $t_{i,j}$, $i = 1, \ldots, D$, $j = 1, \ldots, N$. This provides a matrix $T = (t_{i,j})$.

In **Step 3**, for each random value d_i and hypothetical key k_j, the adversary computes hypothetical values $y_{i,j} = f(d_i + k_j)$.

In **Step 4**, the adversary computes hypothetical power consumption values $h_{i,j}$ using either the Hamming weight or Hamming distance model, with $h_{i,j} = \mathrm{HW}(f(d_i + k_j))$ in the case of

Hamming weight model or $h_{i,j} = \mathrm{HD}(f(d_i + k_j))$ in the case of Hamming distance model. This provides a matrix $H = (h_{i,j})$.

Finally in **Step 5**, the adversary correlates column by column the entries of the matrices H and T above by applying Pearson correlation (3.5) formula.

We recall that for two random variables $X = (X_1, \ldots, X_D)$, $Y = (Y_1, \ldots, Y_D)$ the Pearson correlation formula is given by:

$$\rho(X,Y) = \frac{\sum_{i=1}^{D}(X_i - \bar{X}) \cdot (Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{D}(X_i - \bar{X})^2 \cdot \sum_{i=1}^{D}(Y_i - \bar{Y})^2}} \tag{3.5}$$

where $\bar{X}$ corresponds to the mean of the random variable X.

In **Step 5** the adversary computes then the correlation coefficients related to each hypothetical key value: $r_{i,j} = \rho(H_{.,i}, T_{.,j})$, $i = 1, \ldots, K, j = 1, \ldots, N$, where $H_{.,i}$ and $T_{.,j}$ are respectively the column i of H and the column j of T. The secret key $k^\star$ is recovered from the values of $r_{i,j}$. For example, if k_{i_0} is a right hypothesis key for an index i_0, then the values $r_{i_0,j}$, $j = 1, \ldots, N$ or their mean are higher (i.e exceed a threshold value) than the other correlation coefficients $r_{i,j}$, $i \neq i_0$.

In this chapter, the spectral approach based on Fourier transform allows to recover the secret key $k^\star$ without computing Pearson coefficient, but only to estimate the relevance of the secret key that had been found. For that, we try to minimize the value of the error related to the measurement of the power consumption or to the accuracy of the leakage model. In the sequel we will consider the Hamming-Weight model for the leakage, as we are concerned with software implementation. Indeed the implementation is realized on smart card based on an AVR processor.

3.3.4 Spectral Approach

Like in the general case, the attack consists in performing ciphering operations with various random inputs $d_1, d_2, \ldots, d_D$, chosen by the adversary, then, measuring the power consumption during the calculation and finally, trying to infer the values of the secret subkey $k^\star$. In the sequel we will denote the known data by d. Having in mind a smart card software implementation, according to the discussion in Section 3.3.1, the first model of leakage will be hereafter considered and Assumption 1 applies. Let us notice that this is not restrictive because it is easy to adapt the proposed attack to the second model. Thus, for a given calculus of an S-Box with input d and secret subkey $k^\star$, the power consumption is proportional to the quantity $\varphi(d)$

which admits the following expression:

$$\varphi(d) = \sum_{i=1}^{m} f_i(d + k^\star) + \varepsilon(d) + C, \tag{3.6}$$

where:

- the first term is the model leakage, i.e. the Hamming weight of $f(d+k^\star)$, with f_i the i-th Boolean component of the function f
- the second term is a random noise denoted $\varepsilon(d)$. This noise is due to errors in measurement and also on the accuracy of the model
- the third term is a constant C that corresponds to the power consumption of the device which does not depend on d.

Let g be the function $d \mapsto \sum_{i=1}^{m} f_i(d)$. Equation (3.6) is rewritten as:

$$\varphi(d) = g(d + k^\star) + \varepsilon(d) + C.$$

However, the subkey k assumed by the adversary may not be equal to the right secret key $k^\star$ and the leakage model may not be the right one. Hence, we must introduce an error depending in particular on k. It is denoted by $\varepsilon_k(d)$ and is defined as:

$$\varepsilon_k(d) = \varphi(d) - g(d + k) - C. \tag{3.7}$$

In an ideal situation, that is no noise, no mismatch, and in particular when $k = k^\star$, it should be zero for any $d \in \{0,1\}^n$. Hence, we can define the objective of the attack as finding the value of k which minimizes the quadratic error

$$E(k)^2 = \|\varepsilon_k\|^2 = \sum_{d \in \{0,1\}^n} \varepsilon_k(d)^2.$$

Based on Parseval's Equality (3.3), the problem is equivalent to find k which is solution to:

$$\arg\min_k E(k)^2 = \sum_{u \in \{0,1\}^n} \widehat{\varepsilon}_k(u)^2 \tag{3.8}$$

In the following, it is shown that a spectral approach to solve Equation (3.8) will be relevant in terms of complexity of the underlying attack.

By applying the Fourier transform to Equation (3.7), it comes, for all vectors $u \in \{0,1\}^n$:

$$\widehat{\varepsilon}_k(u) = \widehat{\varphi}(u) - (-1)^{u \cdot k} \widehat{g}(u) - C\sqrt{2^n}\delta_0(u). \tag{3.9}$$

The interest of considering the Fourier transform is that discarding the value at the zero vector eliminates the last term which corresponds to the consumption that does not depend on the value of d. Hence, we define the functions $\widehat{\varphi}^\star$ and $\widehat{g}^\star$ by zeroing the value at the zero vector, that is:

$$\widehat{\varphi}^\star(u) = \begin{cases} \widehat{\varphi}(u) & \text{if } u \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \widehat{g}^\star(u) = \begin{cases} \widehat{g}(u) & \text{if } u \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

then finding a solution k of Equation (3.8) is equivalent to finding a solution k of

$$\arg\min_k E(k)^2 = \sum_{u \in \{0,1\}^n} \left(\widehat{\varepsilon}_k^\star(u)\right)^2 \tag{3.11}$$

with

$$\widehat{\varepsilon}_k^\star(u) = \widehat{\varphi}^\star(u) - (-1)^{u \cdot k} \widehat{g}^\star(u).$$

Expanding the terms in $E(k)^2$ yields

$$E(k)^2 = \sum_{u \in \{0,1\}^n} \widehat{\varphi}^\star(u)^2 + \sum_{u \in \{0,1\}^n} \widehat{g}^\star(u)^2 - 2 \sum_{u \in \{0,1\}^n} \widehat{\varphi}^\star(u) \widehat{g}^\star(u) (-1)^{u \cdot k} \tag{3.12}$$

On the right-hand side, only the last term depends on k. Hence and finally, the problem is to find a solution k to

$$\arg\max_k F(k) \tag{3.13}$$

with

$$F(k) = \sum_{u \in \{0,1\}^n} \widehat{\varphi}^\star(u) \widehat{g}^\star(u) (-1)^{u \cdot k}. \tag{3.14}$$

Remark 3.3.1 *The function F is, up to a factor $1/\sqrt{2^n}$, nothing but the Fourier transform of the function:*

$$u \mapsto \widehat{\varphi}^\star(u) \widehat{g}^\star(u)$$

3.3.4.1 Assessing the estimation reliability

From the above section, we assume that the attacker can choose any $d \in \{0,1\}^n$ and knows the exact instant when the operation $f(d + k)$ is performed. To circumvent such a difficulty which arises in practice, an enhancement of the attack should be proposed. We must memorize the consumption $\varphi(d)$ during a sufficient large time window Δ_t (see Figure 3.5) to guarantee that the computation of $f(d + k^\star)$ will be actually performed. The resulting signal is called a trace (see Figure 3.6). A trace has to be measured for all 2^n values of d. Let us denote by $\varphi_t(d)$ the consumption at time t for the input value d. For a finite number of sample times t in the time window Δ_t, an estimation k of the secret subkey $k^\star$ is computed from (3.13). However, if $k^\star$ is not involved in the operation performed at this time or if $k^\star$ is

actually involved but k is not equal to $k^\star$, the estimation of the right subkey $k^\star$, that is the result of (3.13) will not be reliable. Thus, it is required to find a way of assessing this reliability.

To this end, similarly as in Subsection 3.3.2, let us introduce the quantity defined as:

$$F_t(k) = \sum_{u \in \{0,1\}^n} \widehat{\varphi}_t^\star(u)\widehat{g}^\star(u)(-1)^{u \cdot k}. \tag{3.15}$$

where

$$\widehat{\varphi_t}^\star(u) = \begin{cases} \widehat{\varphi_t}(u) & \text{if } u \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

For brevity, let us denote by g_k the function $d \mapsto g(d+k)$. It is recalled that this function stands for the leakage model. Let us denote by $\widehat{g_k}$ the Fourier transform of g_k and with the same motivation as in Subsection 3.3.2, let us introduce $\widehat{g_k}^\star$ as the Fourier transform of $g_k^\star$ defined as

$$\widehat{g_k}^\star(u) = \begin{cases} \widehat{g_k}(u) & \text{if } u \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

which allows to disregard the zero vector. According to Proposition 3.2.3, for all n–dimensional binary vectors u, we have that $\widehat{g_k}^\star(u) = (-1)^{u \cdot k}\widehat{g}^\star(u)$. Hence, Equation (3.15) can be rewritten as

$$F_t(k) = \langle \widehat{\varphi}_t^\star, \widehat{g_k}^\star \rangle \tag{3.18}$$

Hence, for every sample time t in Δ_t, it is aimed at finding k, that is finding the solution of

$$\arg\max_k F_t(k) \tag{3.19}$$

Any time t leads to a possible estimation k of $k^\star$. It is then necessary to find at which time, this value k is relevant, in other terms, the time where there is a highly matching of k with $k^\star$.

The scalar product $F_t(k)$ being computed for all sample times t in Δ_t, we must detect peaks. The more the matching between the measure and the model, including the key k and the higher the peaks. Hence, the orthogonality is a way of assessing the matching. However, since the detection of the peaks requires a comparison of $F_t(k)$ for all sample times t in Δ_t, a normalization must be done. To this end, we introduce the following quantity which will be called reliability coefficient.

$$r_t(k) = \frac{F_t(k)}{\|\widehat{\varphi}_t^\star\| \cdot \|\widehat{g_k}^\star\|} = \frac{\langle \widehat{\varphi}_t^\star, \widehat{g_k}^\star \rangle}{\|\widehat{\varphi}_t^\star\| \cdot \|\widehat{g_k}^\star\|}$$

Noticing that $\|\widehat{g_k^\star}\| = \|\widehat{g^\star}\|$, the reliability coefficient turns into

$$r_t(k) = \frac{F_t(k)}{\|\widehat{\varphi}_t^\star\| \cdot \|\widehat{g^\star}\|}, \tag{3.20}$$

with a normalization which is independent from k. According to Cauchy-Schwarz inequality (3.1) it follows that $r_t(k) \in [0,1]$

Finally, the attack consists in the following steps:
Steps 1 : for every sample time t in Δ_t, find the solution k of (3.19);
Steps 2 : for every solution k associated to a given sample time t in Δ_t, compute $r_t(k)$ (see Equation (3.20)) and detect the peaks among all the $r_t(k)$ with $t \in \Delta_t$. Those peaks correspond to the times where the key $k^\star$ match with the solution k of Steps 1.

Remark 3.3.2 *If the time where $f(d+k^\star)$ is computed is exactly known, there is no need to perform Steps 2.*

Let us comment on the complexity of the attack. The Fourier Transform of the function $u \mapsto g^\star(u)$ can be computed off-line once and for all from the S-Box table of the leakage model. The Fourier transform $\widehat{\varphi}_t^\star$ must be computed from experimental data for every time t in Δ_t. For every time t in Δ_t, according to Remark 3.3.1, $F_t(k)$ is computed with the Fourier transforms of the functions $u \mapsto \widehat{\varphi}_t^\star(u)\widehat{g}^\star(u)$ and looking for the value of k that maximizes the function F_t. Finally, the peaks are detected by computing $r_t(k)$ (see Equation (3.20)). The Fourier transform calculation can be carried out by using a fast algorithm. A simple divide-and-conquer butterfly algorithm exists for that purpose and is called the Fast Fourier Transform (FFT) (see for example [22]). The complexity of this algorithm equals $n\,2^n$, which is quasi linear in the size of the table of values of the involved function. As a result, the overall complexity of the attack is quasi linear in the size of the S-Box table.

Remark 3.3.3 *It is worth pointing out that the quantity in Equation (3.20) is related to the usual Pearson correlation coefficient (3.5). This coefficient is widely used to evaluate relationship between data. Hence, it is a popular choice for statistical analysis when it comes to perform CPA attacks. The Pearson correlation coefficient of the functions φ and ψ in Φ is then given by:*

$$c(\varphi,\psi) = \frac{\sum_{x\in\{0,1\}^n}\big(\varphi(x)-m_\varphi\big)\times\big(\psi(x)-m_\psi\big)}{\sqrt{\sum_{x\in\{0,1\}^n}\big(\varphi(x)-m_\varphi\big)^2}\times\sqrt{\sum_{x\in\{0,1\}^n}\big(\psi(x)-m_\psi\big)^2}}, \tag{3.21}$$

where m_φ and m_ψ are the means of functions φ and ψ,given by:

$$m_\varphi = \frac{1}{2^n}\sum_{x\in\{0,1\}^n}\varphi(x) \quad \textit{and} \quad m_\psi = \frac{1}{2^n}\sum_{x\in\{0,1\}^n}\psi(x)$$

In other words, Pearson coefficient $c(\varphi,\psi)$ can be expressed as:

$$c(\varphi,\psi) = \frac{\langle\varphi-m_\varphi,\psi-m_\psi\rangle}{\|\varphi-m_\varphi\|\cdot\|\psi-m_\psi\|}$$

As it can be noticed that $\widehat{\varphi - m_\varphi} = \widehat{\varphi}^\star$*, and as the Fourier transform is isometric, it follows that the coefficient given by Equation (3.20) is nothing but the Pearson correlation coefficient of the functions* φ_t *and* g_k*. Thus, it results that the value of* k *given by maximizing* $F(k)$ *in Equation (3.14) is the same value that maximizes the Pearson correlation coefficient of* φ *and* g_k*.*

The interest of the spectral analysis lies in low complexity calculation but also in the fact that it can be easily extended to a so-called multidimensional attack as explained in the next subsection.

3.3.4.2 Multidimensional attack

When computing the value of $f(d + k^\star)$ in a software device, the processor computes sequentially: first $d + k^\star$ and later the value of $f(d + k^\star)$. Thus, the unknown $k^\star$ is involved at least twice within a sufficient time window Δ_t. We call multidimensional attacks, the attacks that take into account the consumption at several instants. It can be expected that more reliable results can be obtained.

The two-dimensional attack is presented thereafter because it is the one which will be used in the experiments presented in Section 3.4 but it is straightforward to generalize it to any higher finite dimension.

Let us consider two instants t_1 and t_2. Let us denote by $f^1(d+k^\star)$ and $f^2(d+k^\star)$ the values computed respectively at times t_1 and t_2. In our case, the function f^1 is the identity since the computation $d + k^\star$ is considered and the function f^2 is the S-Box implementing $f(d + k^\star)$. Let φ^1 (resp. φ^2) be the chip consumption at time t_1 (resp. at time t_2). We consider the two dimensional vector of functions $\overrightarrow{\varphi} = (\varphi^1, \varphi^2)$ in a set Φ_2 of functions $\{0,1\}^n \to \mathbb{R}^2$. Let E^1 the error value given by Equation (3.12) at time t_1 and E^2 be the error value at time t_2. The most likely value of k is the one minimizing the Euclidean norm of the two-dimensional vector $\overrightarrow{E} = (E^1, E^2)$. A direct computation shows that the value that minimizes this norm is the one maximizing the value of

$$F(k) = \sum_{u \neq 0} \left(\widehat{\varphi^1}(u)\widehat{g^1}(u) + \widehat{\varphi^2}(u)\widehat{g^2}(u)\right)(-1)^{u \cdot k},$$

where g^1 and g^2 are the sum of the components of f^1 and f^2. The leakage model at time t_1 is $g_k^1(d) = g^1(d + k)$ and at time t_2 is $g_k^2(d) = g^2(d + k)$.

The scalar product to consider in Φ_2 is the following:

$$\forall \overrightarrow{\varphi} = (\varphi^1, \varphi^2),\ \overrightarrow{\psi} = (\psi^1, \psi^2) \in \Phi_2,\ \langle \overrightarrow{\varphi}, \overrightarrow{\psi} \rangle = \langle \varphi^1, \psi^1 \rangle + \langle \varphi^2, \psi^2 \rangle.$$

The norm associated to this scalar product is:

$$\forall \overrightarrow{\varphi} = (\varphi^1, \varphi^2) \in \Phi_2,\ \|\overrightarrow{\varphi}\|^2 = \|\varphi^1\|^2 + \|\varphi^2\|^2.$$

Clearly, the multidimensional attack can be combined with the estimation approach described above. Let $\overrightarrow{\varphi_t}$ be the two-dimensional power consumption vector at time t. Then $\overrightarrow{\varphi_t} = (\varphi_t^1, \varphi_t^2)$, where φ_t^1 is the consumption at time t and φ_t^2 is the consumption at time $t + t_2 - t_1$. The reliability coefficient between the power consumption $(\varphi_t^1, \varphi_t^2)$ and the model (g_k^1, g_k^2) is required to assess the quality of the estimation. This reliability coefficient for the estimated value k at time t is computed as :

$$r_t(k) = \frac{\langle (\widehat{\varphi_t^1}^\star, \widehat{\varphi_t^2}^\star), (\widehat{g_k^1}^\star, \widehat{g_k^2}^\star) \rangle}{\|(\widehat{\varphi_t^1}^\star, \widehat{\varphi_t^2}^\star)\| \cdot \|(\widehat{g^1}^\star, \widehat{g^2}^\star)\|},$$

where, for $i \in \{1, 2\}$, $\widehat{\varphi_t^i}^\star$ and $\widehat{g_k^i}^\star$ are obtained by discarding the value at the zero vector, as in (3.10).

By using fast Fourier transform algorithm, the multidimensional attack complexity still remains quasi linear in the size of the S-Box.

3.4 Experimental Results

The Challenge An ATMega 163 smart card, involving an 8-bit AVR type processor, has been programmed to process a cipher operation that involves $f(d_i + k_i)$, where f is the AES S-Box, the k_i's are 8-bit secret key elements previously introduced in the card, the d_i's are 8-bit parameters introduced in the card by the adversary. The operation + denotes the bitwise xor of bytes. The challenge of the adversary is to retrieve the secret values k_i by measuring the smart card consumption during calculations. For this purpose, the adversary uses a test bench.

The test bench The test bench (see Figure 3.3) involves the smart card, a smart card reader and an oscilloscope. The oscilloscope is a picoscope 5444b with a 200 MHz bandwidth and a sample rate of $1GS$ per second. The pins of the smart card are connected to the oscilloscope via an adapter. The chip consumption is measured through the potential drop at a resistor connected between the Vss of the card and the ground (see Figure 3.4). All these devices are driven by a computer that implements the attack algorithm.

3.4.1 Application of the attack on a simple S-Box

Experimental protocol The attack consists in recovering four secret keys k_1, k_2, k_3 and k_4 during a time window when the card processes sequentially $f(d + k_1)$, $f(d + k_2)$, $f(d + k_3)$

Figure 3.3: Test bench.

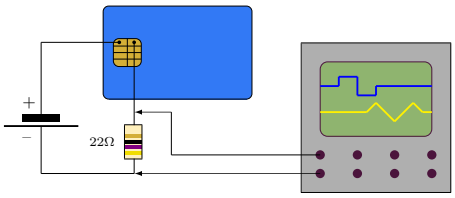

Figure 3.4: Schematic of the power consumption measurement.

and $f(d + k_4)$ for 256 values of d ranging from 0 to 255.

In all subsequent figures, the upper signal is the I/O signal corresponding to the data exchanges and the lower signal is the chip consumption.

Figure 3.5 is an example of consumption trace and the corresponding consumption signal. The time window Δ_t during the computation of the four successive operations $f(d + k_i)$ $(i = 1, \ldots, 4)$ is the range of time when the 256 traces of consumption are memorized in the oscilloscope. The 256 trace records are each composed of 40 000 samples. Figure 3.6 is a magnified part of the time window Δ_t.

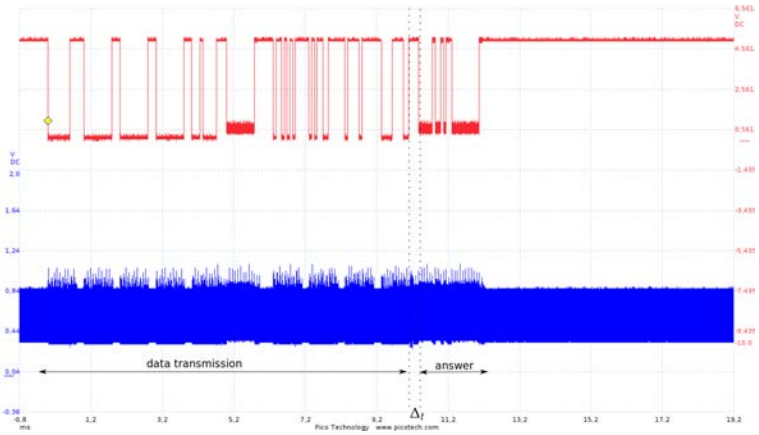

Figure 3.5: Whole data exchanges (up) and the corresponding consumption signal (down).

It is crucial to synchronize properly the 256 consumption traces. For this purpose, the consumption traces are synchronised with the instant given by the first falling edge of the I/O signal that follows the calculation.

For every sample t, $t \in \{1, \ldots, 40\,000\}$ of each 256 traces, the values $\varphi_t(d)$, $d \in \{0, \ldots, 255\}$ are extracted. Then, for every t, $t \in \{1, \ldots, 40\,000\}$, the Fourier transform of φ_t is computed.

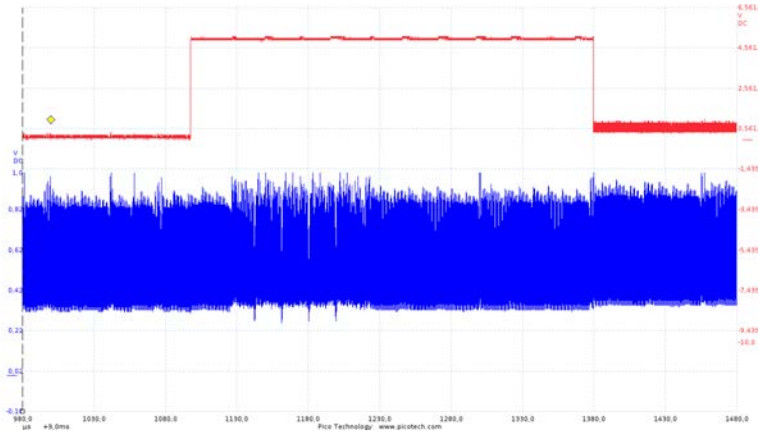

Figure 3.6: Zoom on the time range Δ_t highlighting the first falling edge of the card response used for the synchronization of the traces.

The secret key k_i is evaluated by maximizing the function F given by Equation (3.14). Finally, for this value of k_i, the correlation coefficient is computed according to Equation (3.20).

Experimental results Three distinct scenarios have been considered:

i) an attack when the card computes the exclusive or of the secret k and the data d, that is the operation $d + k$

ii) an attack when the card computes the output of the S-Box, that is the operation $f(d+k)$

iii) a two dimensional attack that combines the two previous attacks involving the operation $d + k$ followed by $f(d + k)$

Figure 3.7, Figure 3.8 and Figure 3.9 show the reliability coefficient $r_t(k)$ given by equation (3.20) with respect to the time t, for the three respective situations. The attack is successful. Indeed, the correlation peaks correspond to the instants when the operations are actually performed and with the right secret key k_i. Let us notice that in Figure 3.7 and Figure 3.9, there are ghost peaks. However, they can be easily disregarded since they correspond to $k = 0$. One possible explanation is that the value of d is loaded in a register previously containing the zero value.

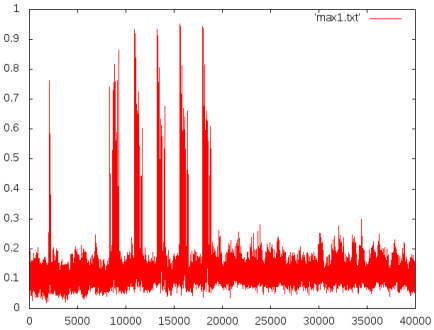

Figure 3.7: Correlation peaks that correspond to scenario *i)*.

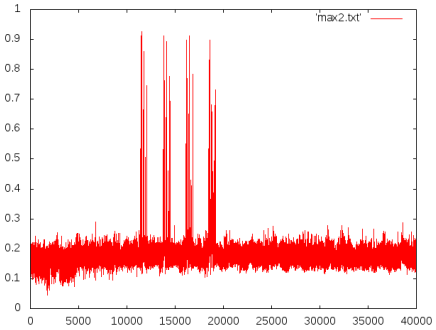

Figure 3.8: Correlation peaks that correspond to scenario *ii)*.

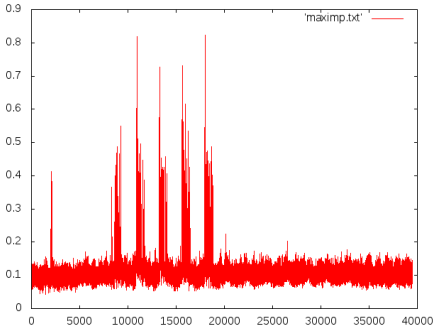

Figure 3.9: Correlation peaks that correspond to scenario *iii)*. The two-dimensional attack considers the times instants t and $t + 4.2\mu s$.

Remark 3.4.1 *Note that in the case where no key is found during the analysis, we have low correlation peaks as illustrated in Figure 3.10. This can happen when the traces are not well synchronised.*

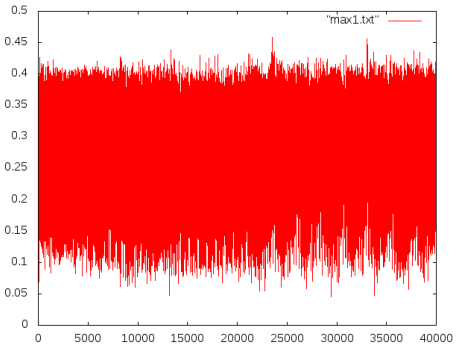

Figure 3.10: Correlation peaks when no key is found.

3.4.2 Application of the attack to the algorithm THE CASCADE

We consider the following cipher equation for a LPV-based SSSC that has be designed. The plaintext at the discrete time k is denoted by m_k and the ciphertext at discrete time k by $d_k = x_k[2]$. We denote the S-Box by S and the subkeys by $\mathrm{SK}_i,\ i = 0, \ldots, 21$.

$$\begin{aligned}
x_{k+1}[0] &= \mathrm{S}(d_k + \mathrm{SK}_0) \cdot x_k[0] + \mathrm{S}(d_k + \mathrm{SK}_1) \cdot x_k[1] + \mathrm{S}(d_k + \mathrm{SK}_2) \cdot x_k[2] + \mathrm{S}(d_k + \mathrm{SK}_3) \cdot x_k[3] \\
&\quad + \mathrm{S}(d_k + \mathrm{SK}_4) \cdot x_k[4] + \mathrm{S}(d_k + \mathrm{SK}_5) \cdot x_k[5] + \mathrm{S}(d_k + \mathrm{SK}_6) \cdot x_k[6] + m_k \\
x_{k+1}[1] &= x_k[0] + x_k[1] + \mathrm{S}(d_k + \mathrm{SK}_7) \cdot x_k[2] + x_k[6] \\
x_{k+1}[2] &= x_k[1] + x_k[2] \\
x_{k+1}[3] &= \mathrm{S}(d_k + \mathrm{SK}_8) \cdot x_k[1] + \mathrm{S}(d_k + \mathrm{SK}_9) \cdot x_k[2] \\
x_{k+1}[4] &= \mathrm{S}(d_k + \mathrm{SK}_{10}) \cdot x_k[1] + \mathrm{S}(d_k + \mathrm{SK}_{11}) \cdot x_k[2] + \mathrm{S}(d_k + \mathrm{SK}_{12}) \cdot x_k[3] \\
x_{k+1}[5] &= \mathrm{S}(d_k + \mathrm{SK}_{13}) \cdot x_k[1] + \mathrm{S}(d_k + \mathrm{SK}_{14}) \cdot x_k[2] + \mathrm{S}(d_k + \mathrm{SK}_{15}) \cdot x_k[3] \\
&\quad + \mathrm{S}(d_k + \mathrm{SK}_{16}) \cdot x_k[4] \\
x_{k+1}[6] &= \mathrm{S}(d_k + \mathrm{SK}_{17}) \cdot x_k[1] + \mathrm{S}(d_k + \mathrm{SK}_{18}) \cdot x_k[2] + \mathrm{S}(d_k + \mathrm{SK}_{19}) \cdot x_k[3] \\
&\quad + \mathrm{S}(d_k + \mathrm{SK}_{20}) \cdot x_k[4] + \mathrm{S}(d_k + \mathrm{SK}_{21}) \cdot x_k[5]
\end{aligned} \tag{3.22}$$

Let us recall that, during the first and second iterations of the encryption, the ciphertext $x[2]$ depends only on the initial internal state x_0. The plaintext m_k influences $x[0]$ at the first iteration, $x[1]$ at the second iteration and $x[2]$ at the third iteration. Then the corresponding ciphertext of m_k is $x_{k+3}[2]$.

Considering the encryption algorithm THE CASCADE, each ciphertext and plaintext are nibble (4 bits). Then we need 16 traces of power consumption. The measurements are performed for traces with 100 000 samples in order to cover the whole encryption operations (3.22). Note that unlike an attack on a simple S-Box, the inputs $d_i, i = 1, \ldots, D$ of the attack are plaintext and the inputs of the S-Box which is targeted within the cipher function are ciphertext that are produced by the encryption function.

To perform the measurement, we then generate several random plaintexts by applying the coupon collector's context solution [41]. Then in order to get 16 different values of ciphertexts, we need to generate about $16 \log(16) \simeq 37$ random plaintexts. Then we encrypt these plaintexts by measuring the power consumption during the encryption operations.

The correlation peaks obtained from the attack are illustrated in Figure 3.11.

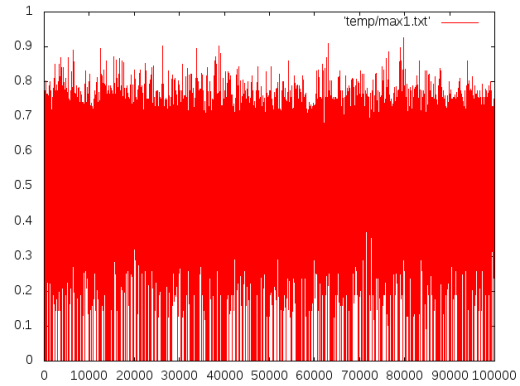

Figure 3.11: Correlation peaks when performing CPA attack on the encryption algorithm THE CASCADE.

Let us comment on the result. The correlation peaks show that there is high correlation even for times t where no calculation of S-Box is performed. And the attack returns wrong secret keys for several times t. These wrong secret keys result from intermediate multiplication operations that are performed during the encryption. Indeed, the operations can be also seen as nonlinear operations $f(d + k)$ where k is an unknown value. Note that the multiplication operations are implemented in constant time operations. This makes it possible to synchronize properly the different traces. Otherwise, the synchronization of the traces would be impossible to achieve.

A solution to find the right secret keys, would be to synchronize the traces by considering the power consumption within an interval of time Δ_t that matches exactly with calculation of the output of the S-Box. Which is actually difficult to achieve.

Note also that we have implemented the multiplication operations as constant time operation to make the attack easy, but in the case where these operations are not implemented in constant time, it would be very complicated to perform the attack by any means since it will be very difficult to distinguish times when multiplication operations and S-Box computation are performed.

3.5 Conclusion

A new approach of CPA attack has been proposed. Unlike usual approaches based on statistical analysis, a spectral approach has been provided. It is based on a correlation quantity derived from the Fourier transform of the power consumption signals. The attack can be applied to any algorithms that involve S-Boxes whose input is the exclusive or of data with a secret subkey. A Hamming weight leakage model has been used. The interest is the complexity of the attack is quasi linear in the size of the S-Box table. Furthermore, it has been shown that it can be easily extended to a so-called multidimensional attack. The attack has been successfully applied to

the AES S-Box, but was less effective on the SSSC THE CASCADE owing to intermediate operations in the update function of the encryption.

Chapter 4

Security Proof of the Canonical Form of Self-Synchronizing Stream Ciphers

The exiting SSSC mentioned in Chapter 2 have an issue related to security: they had been either totally broken (KNOT, ϒΓ, MOSQUITO, HBB, SSS) or partially broken (MOUSTIQUE). Furthermore the attacks were performed by applying Chosen Ciphertext Attacks (CCA). From this, a natural question is raised: *can we expect the design of secure SSSC ?* and if yes *which security level can we expect for this design ?*

In this chapter, we tackle the problem of security of SSSC by considering their canonical form. And we introduce a weaker property to characterize and study the indistinguishablity security of SSSC. The results in this chapter are published in [38].

Contents

4.1 Introduction

In this chapter, we derive a result on the security of the canonical form of the Self-Synchronizing Stream Ciphers. Let us recall that the canonical form of the Self-Synchronizing Stream Cipher (SSSC) is constituted by the combination of a shift register (Figure 4.1), which acts as a state register with the ciphertext as input, together with a filtering function that provides the running key stream. It had been shown in [35] that the canonical form of the Self-Synchronizing Stream Cipher is not resistant against chosen ciphertext attack (IND-CCA security) but can reach the resistance against chosen plaintext attack (IND-CPA security) provided that the filtering function is pseudo random. We introduced a new family of functions in order to characterize the security of the canonical SSSC from its filtering function. Those functions are called Weak Pseudo Random Functions (WPRF) and provide a weaker property than pseudo-randomness. A connection with the *left-or-right indistinguishability* (LOR-IND) is made. This property provides a necessary and sufficient condition to characterize the indistinguishablity of SSSC. The technical developments used to establish the security proof follow similar lines than those used when dealing with block cipher symmetric encryption schemes.

4.2 Canonical Form of the Self-Synchronizing Stream Cipher

4.2.1 Generalities on SSSC

A conventional way for designing an SSSC is to resort to a shift-register-like architecture, giving the so-called canonical representation of the SSSC [76]. It has also been studied in [84]. We recall the canonical representation as depicted in Figure 4.1 and the equations that characterize the cipher and decipher respectively (see also 2.2.2.2 of Chapter 2):

$$\text{cipher: } \begin{cases} z_k = \sigma_\theta^{ss}(c_{k-1}, \ldots, c_{k-n}) \\ c_k = z_k \oplus m_k \end{cases} \qquad \text{decipher: } \begin{cases} \widehat{z}_k = \sigma_\theta^{ss}(c_{k-1}, \ldots, c_{k-n}) \\ \widehat{m}_k = \widehat{z}_k \oplus c_k \end{cases} \tag{4.1}$$

where n is the dimension of the shift register, $\sigma_\theta^{ss} : \{0,1\}^n \to \{0,1\}$ denotes the filtering function parametrized by the secret key θ and for every instant t, $m_k \in \{0,1\}$ is the plaintext symbol, $c_k \in \{0,1\}$ is the ciphertext symbol, $z_k \in \{0,1\}$ is the key stream symbol. The quantity

$\widehat{z}_k \in \{0,1\}$ is the key stream symbol on the deciphering side and $\widehat{m}_k \in \{0,1\}$ is the recovered plaintext symbol.

It is worth pointing out that all the results are established here in the Boolean case but still hold when considering any other finite alphabet.

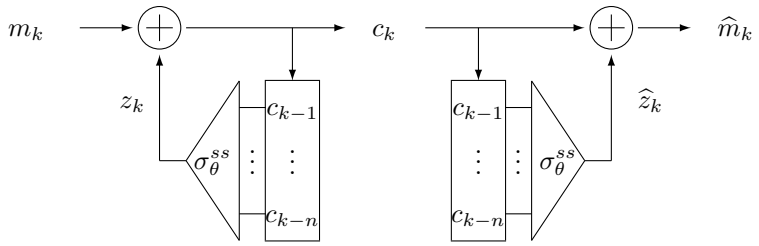

Figure 4.1: Canonical form of a SSSC. Left: the encryption. Right: the decryption.

The secret key θ corresponds to some suitable parameters that select the function σ^{ss} among a family of filtering functions. The dimension of the shift register is given by the integer n. The key stream symbol z_k is the output of the filtering function. It only depends on the secret key θ shared by the emitter (encryption side) and the receiver (decryption side) and on n past values of the ciphertext. The ciphertext c_k is worked out from an exclusive or of the plaintext symbol m_k and of the key stream symbol z_k, and is conveyed through the public channel. Since the generator function σ^{ss} shares, at the transmitter and receiver sides, the same values, namely the n past ciphertexts, then the receiver recovers the plaintext when it properly received the last n ciphertext symbols. Indeed $\widehat{m}_k = m_k$ whenever $\widehat{z}_k = z_k$. Finally, the vector $q_k = (c_{k-1}, \ldots, c_{k-n})$ stands for the internal state. Then, it is clear that the generators synchronize automatically after a finite transient time of length n. As a result, the dimension n of the shift register defines the synchronization delay of the decipher.

The SSSC model defined by Equations (4.1) and depicted on Figure 4.1 remains a conceptual model that can be implemented by different architectures resulting from different design approaches. The state transition function is described by the shift register. It is very simple and is secret key independent. Therefore, the security relies entirely on the security of the filtering function. Maurer's approach is suggested in [74] as an alternative.

4.2.2 Encryption/decryption mechanisms of SSSC

In order to assess the security level reached by the canonical SSSC, it is necessary to formally define the encryption and the decryption mechanisms.

Setup. A random secret key θ is randomly chosen from the key space for both the encryption and the decryption algorithms.

Encryption. For encrypting a message m consisting of ℓ binary symbols, the steps are the following:

1. Randomly choose an n-dimensional binary vector as an initial state q_0 of the shift register.
2. Randomly choose n binary symbols to build the synchronization sequence and concatenate it with the message to be encrypted yielding a binary sequence $m_0, \ldots, m_{n+\ell-1}$. The first n symbols are those of the synchronization sequence, and the other ones are those of the message to be encrypted.
3. For each symbol at instant k,
   (a) Compute the keystream symbol as $z_k = \sigma_\theta^{ss}(q_k)$.
   (b) Compute the ciphertext symbol as $c_k = m_k \oplus z_k$.
   (c) Update the shift register state by shifting its components and feeding it by the computed ciphertext symbol c_k to obtain the next state q_{k+1}.

Decryption. For decrypting the cryptogram consisting of $n + \ell$ binary symbols, the steps are the following:

1. Initialize the shift register state to any arbitrary value, for example the n-dimensional zero vector $\widehat{q}_0 = 0$.
2. For each symbol at instant k,
   (a) Compute the keystream symbol as $\widehat{z}_k = \sigma_\theta^{ss}(\widehat{q}_k)$.
   (b) Compute the plaintext symbol as $\widehat{m}_k = c_k \oplus \widehat{z}_k$.
   (c) Update the shift register state by shifting its components and feeding it with the computed ciphertext symbol c_k to obtain the next state $\hat{q}_{k+1}$.
3. The first n decrypted symbols from the synchronization sequence are ignored. The decrypted message consists of the ℓ last decrypted symbols.

It is easy to check from (4.1) that the decryption procedure recovers the message m when encrypted with the same secret key θ.

4.3 Security Criteria on the Filtering Function

In this section, we first recall what a *Pseudo Random Function* is, and then we define a new security criterion on the filtering function that characterizes the security of the SSSC.

4.3.1 Pseudo Random Functions

A family of functions is said to be Pseudo Random Function (PRF), [46, 8], if it is computationally indistinguishable from the set of all the functions. This means that for polynomial complexity adversaries, a Pseudo Random Function behaves as if it was a true randomly chosen function.

The Pseudo Random property is assessed by the so-called PRF-game that involves an adversary $\mathcal{B}$, challenged to guess whether a given function is either a true random function, randomly chosen from the set of all the functions, or is an element of the family. The entries of the algorithm $\mathcal{B}$ are an integer n together with an oracle that, on request for an n-dimensional vector, returns the value of the function.

The PRF-game. For the parameter n, let us consider a family of functions $f_\theta : \{0,1\}^n \to \{0,1\}$, $\theta \in \{0,1\}^n$. The steps of the game are the followings:

1. The oracle chooses a random bit $b \in \{\text{rf}, \text{prf}\}$. If $b = \text{prf}$ (pseudo random world) then it randomly chooses a function $f = f_\theta$ in the family of pseudo random functions. If $b = \text{rf}$ (random world), it randomly chooses a Boolean function f in the set of all 2^{2^n} Boolean functions $\{0,1\}^n \to \{0,1\}$.

2. The challenger asks the oracle for the values of $f(q_i)$ for inputs $q_1, \ldots, x_\eta$. The number η of queries the challenger is allowed to perform is bounded by a polynomial in n.

3. After the η queries, the challenger must answer a binary value $\widehat{b}$ that means that the challenger guesses that the chosen function was pseudo random ($\widehat{b} = \text{prf}$) or was purely random ($\widehat{b} = \text{rf}$). If $\widehat{b} = b$ then the challenger wins.

The adversary $\mathcal{B}$ does not know in which world she plays: pseudo random or random world? Let $\Pr_{\text{prf}}(\mathcal{B} = \text{prf})$ be the probability that $\mathcal{B}$ answers prf given that it plays in the pseudo random world and let $\Pr_{\text{rf}}(\mathcal{B} = \text{prf})$ be the probability that $\mathcal{B}$ answers prf given that it plays in the random world. In the first case, the answer of $\mathcal{B}$ is correct, and in the second case, it is wrong. The advantage of $\mathcal{B}$ in the PRF-game is by definition:

$$\text{A}dv_{\text{prf}}(\mathcal{B}) = \left|\Pr_{\text{prf}}(\mathcal{B} = \text{prf}) - \Pr_{\text{rf}}(\mathcal{B} = \text{prf})\right|$$

A family of Boolean functions is said to be pseudo random if the maximum advantage of any polynomial adversary is negligible. And a function of the real number x is said to be *negligible* if it decreases faster that the inverse of any polynomial in x as x grows to infinity.

Remark 4.3.1 *For a family of functions to be PRF, it is necessary that the number of its elements increases faster than any polynomial in the security parameter n. If not, the exhaustive search algorithm has polynomial complexity. In a cryptographic context, the secret key selects*

an element of the family. So, the key size must be greater than the logarithm of any power of n. In practice, an n-bit long key is admissible.

4.3.2 Weak Pseudo Random Functions

In this section, we define a new family of functions with a weaker property than the pseudo random functions defined in the previous section and that is more suitable to assess the security of Self Synchronizing Stream Ciphers.

A *Weak Pseudo Random Function*, WPRF for short, is an element of a family of weak pseudo random functions. Such a family is computationally indistinguishable from the family of all functions. The WPRF property is defined by the so-called WPRF-game defined further.

Given a WPRF-family, the WPRF-game challenger is challenged to guess whether a given function presented to her is either an element of the family or a random function. In order to answer the challenge, the challenger can ask for help from an oracle. The difference with PRF-game lies in the form of the requests to the oracle.

The WPRF-game. For the parameter n let us consider a family of functions $f_\theta : \{0,1\}^n \to \{0,1\},\ \theta \in \{0,1\}^n$. The steps of the game are the following.

1. The oracle chooses a random bit $b \in \{\mathrm{rf}, \mathrm{wprf}\}$. If $b = \mathrm{rf}$, then it randomly chooses a function in the set of all 2^{2^n} functions $\{0,1\}^n \to \{0,1\}$. Otherwise, if $b = \mathrm{wprf}$, then it chooses a random parameter θ that defines a random element $f = f_\theta$ of the WPRF-family. The oracle keeps secret the chosen bit and the goal of the challenger is to guess it.

2. The challenger asks the oracle for responses to requests. The input of the request is a polynomial length string of bits $m_1 \cdots m_k$. The response of the oracle is the sequence of couples $\big(q_i, f(q_i)\big)_{i\in\{0,\ldots k\}}$, where:

$$\begin{aligned} &q_0 \text{ is chosen at random by the oracle} \\ &\text{for } i = 1 \text{ to } k,\ q_i = (q_{i-1} \ll 1) + \big(m_i \oplus f(q_i)\big), \end{aligned}$$

   $x \ll 1$ denotes the left shift of the binary vector x by one and the symbol + stands for concatenation.

3. After a polynomial number η of queries, the challenger answers a binary value $\widehat{b} \in \{\mathrm{rf}, \mathrm{wprf}\}$ which means that the function f chosen by the oracle is either random or weak pseudo random.

Remark 4.3.2 *The sequence of couples transmitted by the oracle at Step 2 corresponds exactly to the parameters and the values of the canonical SSSC filtering function during the encryption process of the message m. Given this sequence, the cryptogram is deduced by:*

$$c_i = m_i \oplus f(q_i).$$

Conversely, knowing the cryptogram of a given message, it is possible to infer the sequence transmitted by the oracle. The parameter q_i consists of the vector whose components are the latter cryptogram symbols and the value is given by:

$$f(q_i) = c_i \oplus m_i.$$

Remark 4.3.3 *As the initial value is randomly chosen by the oracle, the challenger cannot predict which parameters q_i will be returned by the oracle.*

The advantage of a challenger $\mathcal{B}$ in a WPRF game is defined in the same manner as in the PRF game:

$$\mathrm{A}dv_{\mathrm{wprf}}(\mathcal{B}) = \Big|\mathrm{Pr}_{\mathrm{wprf}}(\mathcal{B} = \mathrm{wprf}) - \mathrm{Pr}_{\mathrm{rf}}(\mathcal{B} = \mathrm{wprf})\Big|.$$

4.3.3 Pseudo random functions are weak pseudo random functions

Proposition 4.3.1 ([38]) *If a family of functions is pseudo random, then it is weak pseudo random.*

Proof. Given a family of functions $\mathcal{F}$, one has to prove that if a challenger against the weak pseudo random property of $\mathcal{F}$ exists, then a challenger against the pseudo random property can be derived.

Let $\mathcal{B}$ a challenger against the weak pseudo random function family. We construct a challenger $\mathcal{A}$ against the pseudo random function family, that will act for $\mathcal{B}$ as a weak pseudo random oracle. The precise definition of $\mathcal{A}$ is:

1. On request of a length k sequence $m_1 \cdots m_k$ from $\mathcal{B}$, the challenger $\mathcal{A}$ chooses a random value $q_0 \in \{0,1\}^n$, computes $q_1, \ldots, q_k$ as $q_i = (q_{i-1} \ll 1) + m_i$, asks the oracle for the values $f(q_i)$ and transmits them to $\mathcal{B}$.

2. After a polynomial number q of such queries, the challenger $\mathcal{B}$ answers. If $\mathcal{B}$ answers rf, then $\mathcal{A}$ answers rf, and if $\mathcal{B}$ answers wprf, then $\mathcal{A}$ answers prf.

We now prove that the challenger $\mathcal{A}$ so constructed has the same advantage as the challenger $\mathcal{B}$. This follows from the fact that the challenger $\mathcal{A}$ gives the same answer as $\mathcal{B}$. Thus $\mathcal{A}$ wins at the same time as $\mathcal{B}$. And yet, as $\mathcal{A}$ acts for $\mathcal{B}$ as a WPRF oracle, one has $\mathrm{Pr}_{\mathrm{prf}}(\mathcal{A} = \mathrm{prf}) = \mathrm{Pr}_{\mathrm{wprf}}(\mathcal{B} = \mathrm{wprf})$ and $\mathrm{Pr}_{\mathrm{rf}}(\mathcal{A} = \mathrm{rf}) = \mathrm{Pr}_{\mathrm{rf}}(\mathcal{B} = \mathrm{rf})$. It follows that $\mathrm{A}dv_{\mathrm{prf}}(\mathcal{A}) = \mathrm{A}dv_{\mathrm{wprf}}(\mathcal{B})$, and then, if $\mathcal{B}$ has a non negligible WPRF-advantage, then $\mathcal{A}$ has a non negligible PRF-advantage too. □

4.4 Indistinguishability and Security Games

The framework used here to assess the security of SSSC is the one defined in [8] for symmetric encryption schemes. The notions of security games is used within this framework. They are

based on security concepts introduced by Goldwasser and Micali [47, 48]. The first concept is the semantic security, which is a computational analogue to the Shannon perfect secrecy. It means that an adversary cannot recover any bit of information of the plaintext from the ciphertext. However, the semantic security does not allow to fluently deal with security proof. This leads the authors in [47, 48] to define another concept known as indistinguishability which better facilitates proving the security of practical ciphers.

In order to characterize the security of the SSSC, we need a variant of the indistinguishability property known as the *left-or-right indistinguishability* which implies the indistinguishability property against an adaptive chosen plaintext attack.

4.4.1 Indistinguishability and IND-game

A cryptographic scheme is said to be secure, in the sense of indistinguishability, if a polynomial complexity algorithm has a negligible advantage in distinguishing from which of two messages (m_0 and m_1 it has provided) is the cryptogram presented to it. This is formalized by the so-called IND-game.

This game involves two players: an algorithm $\mathcal{A}$ called adversary or challenger, and an oracle. The game consists of the following steps:

1. The challenger chooses two messages m_0 and m_1 and provides the oracle with them.
2. The oracle randomly chooses a binary digit $b \in \{0, 1\}$, encrypts the message m_b and returns the cryptogram to the challenger.
3. The challenger is challenged to guess the value of b, that is which of the two messages has been encrypted. The answer of the challenger is a binary value $\widehat{b}$. The challenger wins if $b = \widehat{b}$.

Before giving her answer, the challenger can be supported by the oracle to decrypt ciphertexts the challenger chooses (Chosen Ciphertext Attack, CCA) or to encrypt plaintexts the challenger chooses (Chosen Plaintext Attack, CPA). Of course, in a CCA attack, the challenger is not allowed to request the decryption of the challenge it has received.

For any b and $\widehat{b}$ in the set $\{0, 1\}$, let $\Pr_{m_b}(\mathcal{A} = \widehat{b})$ be the probability that the challenger $\mathcal{A}$ answers $\widehat{b}$ given the ciphertext which corresponds to the encryption of the message m_b. The advantage of $\mathcal{A}$ in this IND-game is by definition:

$$\mathrm{A}dv(\mathcal{A}) = \Big|\Pr_{m_0}(\mathcal{A} = 0) - \Pr_{m_1}(\mathcal{A} = 0)\Big|.$$

As m_0 and m_1 are randomly chosen with the same probability $1/2$, it holds that

$$\mathrm{A}dv(\mathcal{A}) = \Big|2\Pr(b = \widehat{b}) - 1\Big|,$$

where $\Pr(b = \widehat{b})$ denotes the probability that $\mathcal{A}$ wins.

If the challenger answers at random, then her advantage is null. If it always wins, or always looses, then her advantage equals 1.

A cryptographic scheme is defined by a security parameter, which is for example the key size. Such a scheme is said to be IND-secure if the advantage of any polynomial challenger is negligible. A cryptographic algorithm is said to be IND-CCA if it is IND-secure during a CCA attack. It is said to be IND-CPA if it is IND-secure during a CPA attack.

For the canonical SSSC, the security parameters are the size of the shift register which equals the number of inputs of the filtering function and the size of the secret key.

4.4.2 Left-or-right indistinguishability and LOR-IND-game

Likewise the IND-game, the LOR-IND-game, namely *left-or-right indistinguishability* is described in [7]. It involves two algorithms: the challenger $\mathcal{A}$ and an oracle. The game consists of the following steps:

1. First, the oracle randomly chooses a bit $b \in \{0,1\}$.
2. The challenger sends to the oracle a polynomial number of queries in the form of couples of messages (m_0^i, m_1^i). The oracle encrypts either the message m_0^i or the message m_1^i depending on the bit b chosen during Step 1. It returns to the challenger the cryptogram c^i of the message m_b^i.
3. After a polynomial number of queries, the challenger is challenged to guess the value of the bit b chosen by the oracle. The response of the challenger is a binary value $\widehat{b} \in \{0,1\}$. The challenger wins if $b = \widehat{b}$.

Likewise the IND-game, the advantage of a challenger $\mathcal{A}$ in the LOR-IND-game is defined by:

$$\begin{aligned} \mathrm{Adv}(\mathcal{A}) &= \left|\Pr_{b=0}(\mathcal{A}=0) - \Pr_{b=1}(\mathcal{A}=0)\right| \\ &= \left|2\Pr(b=\widehat{b}) - 1\right| \end{aligned}$$

A ciphering scheme is said to have the LOR-IND property if any polynomial adversary has a negligible advantage in the LOR-IND-game.

The following proposition states that the left-or-right indistinguishability introduced above is a stronger property that the indistinguishability against a chosen plaintext attack.

Proposition 4.4.1 ([38]) *If a ciphering scheme reaches the LOR-IND-CPA property, then it reaches the IND-CPA property.*

Proof. By contraposition, assume that a challenger $\mathcal{A}$ against the IND-CPA property exists. A challenger $\mathcal{B}$ against the LOR-IND property is constructed from $\mathcal{A}$:

1. The challenger $\mathcal{A}$ chooses a first couple of messages (m_0^0, m_1^0) that she sends to $\mathcal{B}$, and $\mathcal{B}$ sends directly theses messages to the LOR-IND oracle.

2. When $\mathcal{A}$ asks $\mathcal{B}$ for the cryptograms of a chosen plaintext message m^i, the algorithm $\mathcal{B}$ sends to the LOR-IND-oracle the couple with twice the same message : (m^i, m^i). The cryptogram returned by the oracle thus corresponds to m^i whatever the random bit b chosen by the oracle. This cryptogram is returned to $\mathcal{A}$.

3. After a polynomial number of queries, the challenger $\mathcal{A}$ returns a guess $\widehat{b}$ for b, and $\mathcal{B}$ returns the same value.

The algorithm $\mathcal{B}$ so defined simulates for $\mathcal{A}$ an IND-CPA-oracle, and the advantage of B equals the advantage of $\mathcal{A}$. Thus, if an adversary against the IND-CPA property exists, then an adversary against the LOR-IND property exists too. By contraposition, if the ciphering scheme is LOR-IND-CPA secure, then it is IND-CPA secure. □

4.5 Security proof of the canonical SSSC

4.5.1 IND-CCA security of the canonical SSSC

The ideal case for a canonical SSSC is when the filtering function is randomly chosen among all the 2^{2^n} Boolean functions. This ideal case is not realistic as it would imply an exponential key size. However, even in this situation, the following property holds:

Proposition 4.5.1 ([38]) *The canonical SSSC cannot reach the IND-CCA security*

Proof. Actually, this is due to the fact that the cryptograms produced by an SSSC are malleable. Indeed, they can be modified at their end without effect on the beginning of the plaintext.

Consider the special situation when the challenger provides $m_0 = 0 \cdots 0$, the message that only involves symbols 0, and $m_1 = 1 \cdots 1$, the message that only involves symbols 1. It receives a cryptogram c. It modifies only the last symbol of the cryptogram and asks the oracle to decrypt. If the answer of the oracle starts with a sequence of 0, then c is the encryption of m_0 and if it starts with a sequence of 1, then c is the encryption of m_1. Following this strategy, the challenger always wins. That completes the proof. □

On the other hand, it is shown in next session that the canonical SSSC can reach the IND-CPA security.

4.5.2 IND-CPA security of the canonical SSSC

The main result of this section is to prove that the WPRF property of the filtering function characterizes the IND-CPA security of the canonical SSSC.

Proposition 4.5.2 ([38]) *A canonical SSSC reaches the LOR-IND security if and only if the filtering function is WPRF-family.*

Proof. We first prove that if the filtering function of a canonical SSSC is WPRF-family, then the SSSC reaches the LOR-IND security. In order to prove this property, we prove that if a polynomial adversary $\mathcal{A}$ exists against the SSSC, then we can deduce a challenger $\mathcal{B}$ against the WPRF property of the filtering function. Let $\mathcal{B}$ be defined as follows:

1. $\mathcal{B}$ chooses a random bit $b \in \{0,1\}$.
2. For each received couple of messages (m_0^i, m_1^i), the algorithm $\mathcal{B}$ asks for the WPRF oracle the sequence of bits m_b^i and receives from the oracle the sequence $\left(x_k^i, f(x_k^i)\right)$. From this sequence, $\mathcal{B}$ computes a cryptogram c that it sends to $\mathcal{A}$.
3. After a polynomial times of queries, the adversary A answers a bit $\widehat{b} \in \{0,1\}$. If $b = \widehat{b}$ then $\mathcal{B}$ answers wprf, and if $b \neq \widehat{b}$ then she answers rf.

It must be shown that if the advantage of $\mathcal{A}$ is non negligible, then the advantage of $\mathcal{B}$ is non negligible too. From the definition of $\mathcal{B}$, its advantage is:

$$\mathrm{A}dv(\mathcal{B}) = \left|\Pr_{\mathrm{wprf}}(b = \widehat{b}) - \Pr_{\mathrm{rf}}(b = \widehat{b})\right|.$$

In the WPRF world, the challenger $\mathcal{B}$ acts for $\mathcal{A}$ as a true ciphering oracle. Thus, the advantage of $\mathcal{A}$ is:

$$\begin{aligned} \mathrm{A}dv(\mathcal{A}) &= \left|2\Pr_{\mathrm{wprf}}(b = \widehat{b}) - 1\right| \\ &\leq \left|2\Pr_{\mathrm{wprf}}(b = \widehat{b}) - 2\Pr_{\mathrm{rf}}(b = \widehat{b})\right| + \left|2\Pr_{\mathrm{rf}}(b = \widehat{b}) - 1\right| \end{aligned} \tag{4.2}$$

The first term of the right hand side of the inequality (4.2) is twice the advantage of $\mathcal{B}$. Moreover, as in the random world, the answers of the WPRF oracle are random, the answers of $\mathcal{B}$ to $\mathcal{A}$ are random too. As a consequence, the answer of $\mathcal{B}$ is random and the second term in the second member of inequality (4.2) is null. It follows that:

$$\mathrm{A}dv(\mathcal{A}) \leq 2\mathrm{A}dv(\mathcal{B}),$$

which proves the expected result.

It remains to prove the reciprocal: if the canonical SSSC reaches the LOR-IND-CPA security, then the filtering function achieves the WPRF property. By contraposition, it is equivalent to prove that if a WPRF challenger exists against the filtering function, then a LOR-IND adversary can be constructed against the canonical SSSC.

Given a WPRF challenger $\mathcal{B}$, then a LOR-IND adversary is constructed as follows:

1. On each query m^i from the challenger $\mathcal{B}$, the adversary $\mathcal{A}$ generates a random binary sequence a^i and sends to the LOR-IND oracle the couple (m^i, a^i).

2. The oracle encrypts either m^i or a^i according to the value of a random bit $b \in \{0,1\}$ chosen once and for all. The oracle returns to $\mathcal{A}$ the corresponding cryptogram c^i.

3. From the received cryptogram c^i, the adversary $\mathcal{A}$ computes the values of the filtering functions used by the oracle, assuming that the oracle has encrypted the message m^i, and transmits the sequence of couples $\left(x_k^i, f(x_k^i)\right)$ to $\mathcal{B}$.

4. After a polynomial number of queries, the challenger $\mathcal{B}$ returns an answer: rf or wprf. If $\mathcal{B}$ answers wprf, then $\mathcal{A}$ returns $\widehat{b} = 0$ and if $\mathcal{B}$ answers rf, then $\mathcal{A}$ returns $\widehat{b} = 1$.

If the LOR-IND oracle chooses $b = 0$, then its answers to $\mathcal{B}$ correspond to the encryption of the binary sequences provided by $\mathcal{A}$, and $\mathcal{B}$ acts as an oracle for $\mathcal{A}$ in the WPRF world. Conversely, if the LOR-IND oracle chooses $b = 1$, and as the sequences a^i are random, $\mathcal{B}$ acts as an oracle for A in the rf world. As a consequence, the advantage of $\mathcal{B}$ in the LOR-IND game equals exactly the advantage of $\mathcal{A}$ in the WPRF game and that achieves the proof. □

From the above propositions, the following result holds:

Corollary 4.5.1 ([38]) *If the filtering function of a canonical SSSC is PRF, then it achieves the IND-CPA security.*

Proof. Assume that the filtering function of a canonical SSSC is PRF. From Proposition 4.3.1, it is WPRF. From Proposition 4.5.2, the canonical SSSC also achieves the LOR-IND security, and finally, from Proposition 4.4.1, it achieves the IND-CPA security too. □

We can summarize all the security notions developed in this chapter by the following implications:

$$\text{prf} \Rightarrow \text{wprf} \Leftrightarrow \text{LOR-IND-CPA} \Rightarrow \text{IND-CPA}$$

4.6 Conclusion

The cryptological complexity of the canonical form of the Self-Synchronizing Stream Cipher lies in the filtering function. The maximum security is achieved when this function is a pure random function. It has been shown that, even in this ideal case, this kind of cipher cannot reach the IND-CCA security. This result consolidates the CCA attacks that have been performed on existing SSSC [56, 30, 57]. This is due to the fact that the ciphertexts are malleable. Thus, the maximum expected security is IND-CPA. In realistic implementation, the filtering is a pseudo random function. We have proved that for such practical ciphers, the IND-CPA security can be reached. The interest of the result lies in that it guarantees the existence of SSSC that can be IND-CPA secure.

Finally, a weaker property than pseudo-randomness has been proposed. It had been called WPRF. It allows to assess a stronger security level than indistinguishability, namely left-or-right indistinguishability, against chosen plaintext attacks.

Chapter 5

Matrix Representations of Vectorial Boolean Functions and Eigenanalysis

This chapter aims at giving a unified overview on the various representations of vectorial Boolean functions, namely the Walsh matrix, the correlation matrix and the adjacency matrix. A new representation called polynomial matrix is introduced. It is shown that those different representations are similar. For a vectorial Boolean function with the same number of inputs and outputs, an eigenanalysis of those representations is performed. It is shown how eigenvalues and eigenvectors are related to the structure of the graph associated to this function. Also, we give an example of LPV-based SSSC and, by considering the correlation matrix associated to its state transition function, we show that it is not based on a T-function. To simplify the reading of this chapter, the proof of the results are removed but can be found by the reader in [37].

Contents

5.1 Introduction

Vectorial Boolean functions [22] play an important role in cryptography as non-linear components of symmetric algorithms [75]. They are also used in control theory to model discrete dynamical systems [14]. Usual existing representations are presented such as the Walsh matrix, the correlation matrix and the adjacency matrix (related to graph representations) [28] often used in the analysis of cryptographic properties. Besides, we introduce a new representation based on algebraic properties. We call this representation polynomial matrix.

It is shown that the representations describe the same function in different bases and the bases are given explicitly. Then, the relations between these representations are proved. For square matrices representing the vectorial Boolean functions, we perform the eigenanalysis. Deep connections are established between the eigenvalues of those matrices, their related eigenspaces and the structure of the graphs associated to the vectorial Boolean functions.

This chapter provides further results to the work of Parriaux's Thesis [84]. They are useful to classify different categories of state transition function of SSSC when considering the correlation matrix of these functions. The results are also general and could be useful for people concerned with theoretical aspects regarding vectorial Boolean functions and with practical applications too. They are interesting in particular for cryptographic purposes.

5.2 Boolean functions representations

This section provides necessary prerequisites. The reader may refer to [22] for details. New results are also presented here as important complements.
A Boolean function is a function from the vector space $\mathbb{F}_2^n$ to the set $\{0,1\}$. Depending on the context, the set $\{0,1\}$ is considered either as the two element field $\mathbb{F}_2$ $(1+1=0)$ or as a subset of the field $\mathbb{C}$ of complex number $1+1=2$. We call such a function a (n)–function. The various usual representations are the truthtable, the Fourier and the Walsh transform and the polynomial representation. These are recalled below. It is clear that the convenience of a specific representation depends on the properties it is expected to characterize.

5.2.1 Fourier/Walsh transform

Let f be any complex-valued function on $\mathbb{F}_2^n$. We recall from Definition 3.2.1 of Chapter 3 the Fourier transform of f denoted by $\widehat{f}$, which is by definition the complex-valued mapping $\mathbb{F}_2^n \longrightarrow \mathbb{C}$ defined for $u \in \mathbb{F}_2^n$ by:

$$\widehat{f}(u) = \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{F}_2^n} f(x)(-1)^{x \cdot u}, \tag{5.1}$$

where $x \cdot u = x_0 u_0 + \cdots + x_{n-1} u_{n-1}$ is the dot product of the two vectors x and u. This transform is invertible and the inverse is given by:

$$\widehat{\widehat{f}} = f \tag{5.2}$$

And the Fourier transform is an expression of f in the orthogonal basis of the so-called Walsh functions (see Definition 3.2.1), defined for all $u \in \mathbb{F}_2^n$ by:

$$\chi_u : \begin{array}{ccc} \mathbb{F}_2^n & \longrightarrow & \mathbb{C} \\ x & \longmapsto & \frac{1}{\sqrt{2^n}}(-1)^{u \cdot x}. \end{array} \tag{5.3}$$

The expression of f in this basis is:

$$f = \sum_{u \in \mathbb{F}_2^n} \widehat{f}(u) \chi_u.$$

When f is represented by the vector corresponding to its truthtable, the Fourier transform (5.1) also admits a matrix expression:

$$\widehat{f} = Hf, \tag{5.4}$$

where H is the so-called Hadamard matrix whose coefficient at row $u \in \mathbb{F}_2^n$ and column $v \in \mathbb{F}_2^n$ is:

$$H_{u,v} = \frac{1}{\sqrt{2^n}}(-1)^{u \cdot v}. \tag{5.5}$$

The Hadamard matrix H is invertible and its inverse is given by:

$$H^{-1} = H. \tag{5.6}$$

As a result, it holds that

$$f = H\widehat{f}$$

When dealing with Boolean functions, it is better to use the Walsh transform that has 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 $f_\chi = (-1)^f = 1 - 2f$. The Walsh transform of f is then the function $\widehat{f_\chi}$ defined by:

$$\begin{array}{rccc} & \mathbb{F}_2^n & \longrightarrow & \mathbb{R} \\ \widehat{f_\chi} : & u & \longmapsto & \dfrac{1}{\sqrt{2^n}} \displaystyle\sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+x\cdot u} \end{array} \tag{5.7}$$

By applying the Parseval's equality (3.3) to the sign function of Boolean functions, one has:

$$\sum_{u \in \mathbb{F}_2^n} \left[\widehat{f_\chi}(u)\right]^2 = \left[\sum_{u \in \mathbb{F}_2^n} \widehat{f_\chi}(u)\right]^2 = 2^n \tag{5.8}$$

5.2.2 Polynomial representations

This section is devoted to polynomial representations of (n)–functions. Two representations are presented: Algebraic Normal Form (ANF) (see [71]) and Numerical Normal Form (NNF) (see [24]).

Due to the equality, $\forall a \in \{0,1\}, a = a^2$, distinct polynomials may represent the same Boolean function. In order to obtain the uniqueness of the representation, we only consider the polynomials in the ring of multivariate polynomials whose exponents for each indeterminate are at most one.

Let a and b be two elements of $\mathbb{F}_2$, it holds that $a^b = 1$ if $b \leq a$ and $a^b = 0$ elsewhere. Polynomial representations are expressions of the function in the so-called basis of monomials defined, for $u \in \mathbb{F}_2^n$, by:

$$\begin{array}{ccc} \mathbb{F}_2^n & \longrightarrow & \{0,1\} \\ x & \longmapsto & x^u \end{array} \tag{5.9}$$

where

$$x^u = x_1^{u_1} \cdots x_n^{u_n} \tag{5.10}$$

is called a monomial.

For any vector u in $\mathbb{F}_2^n$, the support of u is defined by:

$$\mathrm{supp}(u) = \big\{ i \in \{1, \ldots, n\} \mid u_i \neq 0 \big\}.$$

Definition 5.2.1 *When x and u are two n–dimensional binary vectors, the notation $x \preceq u$ means that the support of x is included in the support of u. The following equivalences holds:*

$$x \preceq u \iff u^x = 1 \iff \forall i \in \{1, \ldots, n\}, \; x_i \leq u_i. \tag{5.11}$$

5.2.2.1 Algebraic Normal Form (ANF)

Let us recall a multivariate polynomial representation of Boolean functions called Algebraic Normal Form (ANF for short).

The ANF coefficients of a function f are, by definition, for $u \in \mathbb{F}_2^n$,

$$a_u = \sum_{x \in \mathbb{F}_2^n} f(x) u^x, \tag{5.12}$$

where the sum is performed in the two-element field $\mathbb{F}_2$. They express the function f in the basis of monomials as:

$$\forall x \in \mathbb{F}_2^n,\ f(x) = \sum_{u \in \mathbb{F}_2^n} a_u x^u. \tag{5.13}$$

5.2.2.2 Numerical Normal Form (NNF)

Let us recall another multivariate polynomial representation of Boolean functions called Numerical Normal Form (NNF for short).

Unlike the ANF, the coefficients of the polynomial do not lie in the two element field $\mathbb{F}_2$ but in the field $\mathbb{C}$ of complex numbers. Notice that such a polynomial may not correspond to a $\{0,1\}$ valued function.

The NNF coefficients of a complex valued function f are the complex coefficients of f expressed in the basis of monomial functions. They are defined by:

$$\widetilde{f}(u) = \sum_{x \in \mathbb{F}_2^n} (-1)^{\mathrm{HW}(x)-\mathrm{HW}(u)} f(x) u^x = \sum_{x \in \mathbb{F}_2^n | x \preceq u} (-1)^{\mathrm{HW}(x)-\mathrm{HW}(u)} f(x),\ u \in \mathbb{F}_2^n$$

where HW denotes the Hamming weight function and the sum is performed in the field of complex numbers. The expression of f in the basis of monomials is:

$$\forall x \in \mathbb{F}_2^n,\ f(x) = \sum_{u \in \mathbb{F}_2^n} \tilde{f}(u) x^u = \sum_{u \in \mathbb{F}_2^n | u \preceq x} \tilde{f}(u) \tag{5.14}$$

Likewise for the Fourier transform, a matrix relation exists between a function f described by its truthtable and its NNF:

$$\widetilde{f} = Zf, \tag{5.15}$$

where Z is a 2^n-dimensional square matrix whose coefficient at row $u \in \mathbb{F}_2^n$ and column $v \in \mathbb{F}_2^n$ is given by:

$$Z_{u,v} = (-1)^{\mathrm{HW}(v)-\mathrm{HW}(u)} u^v.$$

x	000	001	010	011	100	101	110	111
$f_e(x)$	010	100	001	101	010	101	010	110

Table 5.1: Truthtable of function f_e

The matrix Z is invertible and the inverse is the so-called *monomial matrix* defined by:

$$Z^{-1} = M \tag{5.16}$$

where the coefficient at row $x \in \mathbb{F}_2^n$ and column $u \in \mathbb{F}_2^n$ of M is given by $M_{x,u} = x^u$. As a result, one has $f = M\tilde{f}$.

5.3 Vectorial Boolean functions representations

A vectorial Boolean function is, by definition, a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. It can be considered as a vector of m (n)–functions. We call such a function an (n, m)–function. Vectorial Boolean functions have been extensively discussed in [23]. For any vectorial Boolean function f, it is possible to define different matrix representations denoted by A_f, C_f, W_f, and P_f. They are respectively named adjacency, correlation, Walsh and polynomial matrices. For brevity, the superscript of the matrices is omitted when the corresponding function is clear. Let f_e be the function defined by the table of values of Table 5.1. This function is used throughout the rest of the paper to illustrate the results.

5.3.1 Adjacency matrix

The adjacency matrix of f is denoted by A. It is the expression of f in the basis of the characteristic functions $(\delta_u)_{u\in\mathbb{F}_2^n}$, defined by Equation (3.2).

Definition 5.3.1 (Adjacency matrix) *Let f be an (n, m)–function. Its adjacency matrix A is a $2^n \times 2^m$ dimensional matrix for which each row indexed by $x \in \mathbb{F}_2^n$ is null except the coefficient at the column $y = f(x)$, which equals* 1.

For example, the adjacency matrix of f_e is:

$$A_{f_e} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

The name *adjacency matrix* is inspired from graph theory [45]. When the number of inputs equals the number of outputs, the function can also be represented by a labeled directed graph $\mathcal{G}$. Hereafter, we only consider directed graphs. Thus, for brevity but without any ambiguity, we merely call them graphs. A graph is naturally associated to a (n, n)–function, where an arc relates an input of the function to its image as defined below.

Definition 5.3.2 (Graph associated to a function (see [93])) *Let f be an (n, n)–function. The graph associated to f is defined by the set of vertices $V = \mathbb{F}_2^n$ and the set of arcs E that are the ordered pairs $(x, y) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ such that $y = f(x)$.*

Now, let us recall some graph-theoretic terminology that is necessary in the further development. Some notions of this terminology have also been defined in Chapter 1. For each definition, the corresponding structure is illustrated for the function f_e in Figure 5.1.

Definition 5.3.3 (Graph-related terminology (see [45]))

- *An edge is an unordered pair of distinct vertices of the graph,*
- *An arc is a directed edge, i.e an ordered pair of distinct vertices. An arrow shows the direction of the edge,*
- *A vertex x is said to be incident to a vertex y (or to be a preimage of a vertex y) if there is an arc from x to y. The vertex 111 is incident to 110,*
- *The in-degree 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 for which this vertex is incident to. The out-degree of each vertex is 1, including vertex 101,*
- *A path is a sequence of vertices $(x_0, \ldots, x_k)$ such that, for each vertex, there is an arc from x_i to x_{i+1}. The length of the path is the number of arcs involved in the sequence. The sequence $(110, 010, 001)$ is a path of length 2,*
- *A cycle is a path such that the starting vertex and the ending vertex are the same. The sequence $(010, 001, 100, 010)$ 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,*
- *The 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 any other vertex but itself. Any sink defines 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 is a set of vertices such that there is always a path (not necessarily directed) that relates any two vertices of that set. The set of vertices* $\{111, 110, 000, 010, 100, 001\}$ *corresponds to one connected component and the set of vertices* $\{011, 101\}$ *corresponds to another connected component.*

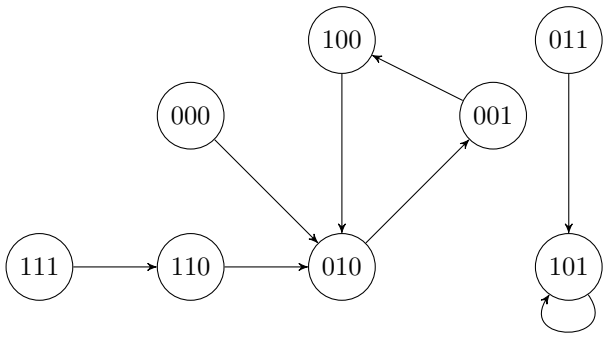

Figure 5.1: Graph associated to the function f_e.

The claims in the following Remarks 5.3.1, 5.3.2 and 5.3.3 are straightforward:

Remark 5.3.1

- *the graph associated to an* (n, n)*–function is such that there is one and only one arrow starting from any of its vertices,*
- *the graph associated to a permutation contains neither leaf nor junctions,*
- *if a graph contains no leaf then it is the graph associated to a permutation and thus, it contains no junction,*
- *if a graph contains no junction then it is the graph associated to a permutation and thus, it contains no leaf.*

Remark 5.3.2 *As the set of vertices of the graph of a* (n, n)*–function is finite, the graph contains at least one cycle.*

Remark 5.3.3 *If there is no leaf, then the graph is a union of cycles and it contains no junction. Besides, each new leaf, either adds a new junction or increases the multiplicity of an existing junction by one. As a consequence, the number of leaves equals the sum of the multiplicities of the junctions.*

5.3.2 Walsh/correlation matrix

Correlation matrices have been defined in [28]. They are related to Walsh matrices by a mere normalization coefficient.

Definition 5.3.4 (see [23]) *The Walsh matrix of an (n,m)–function is the $2^m \times 2^n$ dimensional matrix W whose coefficients are defined at indexes $u \in \mathbb{F}_2^m$ and $v \in \mathbb{F}_2^n$ by:*

$$W_{u,v} = \sum_{x \in \mathbb{F}_2^n} (-1)^{u \cdot f(x) + v \cdot x}. \tag{5.17}$$

For all $u \in \mathbb{F}_2^m$ and all $v \in \mathbb{F}_2^n$, the coefficient $W_{u,v}$ is the number of times the Boolean function $x \longmapsto u \cdot f(x)$ equals the linear Boolean function $x \longmapsto v \cdot x$, minus the number of times they differ.

For example, the Walsh matrix of the function f_e is:

$$W_{f_e} = \begin{pmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & -2 & -2 & 6 & -2 \\ 0 & -4 & 0 & -4 & 4 & 0 & -4 & 0 \\ -6 & -2 & 2 & -2 & 2 & -2 & 2 & -2 \\ 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & -2 & -2 & -2 & 6 \\ -4 & 0 & -4 & 0 & 0 & 4 & 0 & -4 \\ -2 & -6 & -2 & 2 & -2 & 2 & -2 & 2 \end{pmatrix}$$

The row $u \in \mathbb{F}_2^m$ of the matrix W is the Walsh transform (up to a coefficient $\sqrt{2^n}$) of the linear combinations of the coordinates of f defined by $x \longmapsto u \cdot f(x)$, $x \in \mathbb{F}_2^n$. The list of the coefficients of the Walsh matrix of a function is called the spectrum of the function.

Definition 5.3.5 (see [28]) *The correlation matrix of a (n,m)–function f is:*

$$C_f = 2^{-n} W_f. \tag{5.18}$$

Let us recall important results used further and which have been published in [86].
We are given an (n,m)–function g and a random variable $X \in \mathbb{F}_2^n$ whose value is described by the probability law $p : \mathbb{F}_2^n \longrightarrow \mathbb{R}$ that expresses the probability $p(x)$ that $X = x$. We are concerned with inferring 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 $q(y) = \Pr[g(X) = y]$. Without any 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 $q(y)$).

Proposition 5.3.1 (see [86]) *Let g be an (n,m)–function and X be a random variable described by the probability law p. Then the probability law q of the random variable $Y = g(X)$ is given by:*

$$q = H^{-1} C_g H p. \tag{5.19}$$

Remark 5.3.4 *Let $\widehat{p}$ and $\widehat{q}$ be the respective Fourier transform of p and q as defined in Proposition 5.3.1. It is also shown in [86] that*

$$\widehat{q} = C_g \widehat{p} \tag{5.20}$$

5.3.3 Reduced Walsh/correlation matrix

The reduced Walsh matrix is defined as follows.

Definition 5.3.6 (Reduced Walsh matrix) *For a Walsh matrix W of dimension $2^m \times 2^n$, its reduced matrix $W^\star$ of dimension $(2^m - 1) \times (2^n - 1)$ is the matrix deduced from W, where the first row and the first column have been removed.*

$$W^\star = \begin{pmatrix} W_{1,1} & \cdots & W_{1,2^n-1} \\ \vdots & & \vdots \\ W_{2^m-1,1} & \cdots & W_{2^m-1,2^n-1} \end{pmatrix}.$$

Remark 5.3.5 *The same definition holds for the correlation matrix C.*

It is interesting because it yields more homogeneous results. The purpose of the sequel is to show that no information on f is lost with the reduced matrix, except for constant functions.

First, let us note that the first row of the correlation matrix always equals the 2^n–dimensional vector $(2^n, 0, \ldots, 0)$. In the sequel, considerations on the first column are given.

Lemma 5.3.1 *Let f be an (n)–function. Then, the quantity $\sum_{u \in \mathbb{F}_2^n | u \neq 0} \widehat{f_\chi}(u)$ is null if and only if f is a constant function.*

Lemma 5.3.2 *An (n)–function f can be uniquely recovered from its last $2^n - 1$ Walsh coefficients provided that it is not a constant function.*

Finally, we have the following result.

Proposition 5.3.2 ([37]) *An (n,m)–function can be uniquely recovered from its reduced Walsh matrix provided that it is not a constant function.*

5.3.4 Polynomial matrices

The extension of the NNF to an (n, m)–function gives rise to a $2^m \times 2^n$ dimensional matrix denoted with P. We call it the *polynomial matrix* of f, and the entry at row indexed by $u \in \mathbb{F}_2^m$ and column indexed by $v \in \mathbb{F}_2^n$ is defined by:

$$P_{u,v} = \sum_{x \in \mathbb{F}_2^n} (-1)^{\mathrm{HW}(x)-\mathrm{HW}(v)} f(x)^u v^x.$$

Note that the rows indexed by $u \in \mathbb{F}_2^n$ correspond to the NNF of the function f^u and in particular the rows for which $\mathrm{HW}(u) = 1$ correspond to the NNF of the coordinate functions of f. The matrix P expresses f in the basis of the polynomials, $x \longmapsto (-1)^{\mathrm{HW}(x)-\mathrm{HW}(v)} v^x$, $v \in \mathbb{F}_2^n$. For example, the polynomial matrix of f_e is

$$P_{f_e} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

In the same way as the ANF can be obtained by performing a modulo two reduction of the NNF, we can define a modulo two reduction of the polynomial matrix P.

5.3.5 Reduced Polynomial matrix

Likewise for correlation matrix, we define the reduced form P^* of a polynomial matrix P associated to an (n, m)–function f. Assuming that f is not constant, we will prove that f can be recovered given the coefficients of $P^\star$.

Definition 5.3.7 (Reduced polynomial matrix) *For any polynomial matrix P of dimension $2^m \times 2^n$, its reduced matrix $P^\star$ of dimension $(2^m - 1) \times (2^n - 1)$ is the matrix deduced from P, where the first row and column have been removed.*

$$P^\star = \begin{pmatrix} P_{1,1} & \cdots & P_{1,2^n-1} \\ \vdots & & \vdots \\ P_{2^m-1,1} & \cdots & P_{2^m-1,2^n-1} \end{pmatrix}$$

The following lemma is similar to Parseval's identity, and is based on the fact that for any $\{0, 1\}$ valued function f, one has $\sum_{x \in \mathbb{F}_2^n} f(x) = \sum_{x \in \mathbb{F}_2^n} f^2(x)$.

Lemma 5.3.3 (see [49])
If f is a Boolean (n)–function then,

$$\sum_{x\in\mathbb{F}_2^n} f(x) = \sum_{x\in\mathbb{F}_2^n}\sum_{u\in\mathbb{F}_2^n}\sum_{v\in\mathbb{F}_2^n} \tilde{f}(u)\tilde{f}(v)x^u x^v. \tag{5.21}$$

The next lemma expresses orthogonality between monomial functions.

Lemma 5.3.4 (see [49]) *Let $s, u \in \mathbb{F}_2^n$ then,*

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

In particular, for $s = 0$, $\sum_{x\in\mathbb{F}_2^n} (-1)^{HW(x)} u^x = \begin{cases} 1 & \text{if } u = 0 \\ 0 & \text{else.} \end{cases}$

As a straightforward consequence of Lemma 5.3.4 and relation (5.14), the following remark holds.

Remark 5.3.6 *A Boolean function f is constant if and only if, for all nonzero vector $u \in \mathbb{F}_2^n$, $\tilde{f}(u) = 0$.*

Remark 5.3.7 *From Lemma 5.3.2, it can be inferred whether a Boolean function is a constant function or not, given its $2^n - 1$ Walsh coefficients at the nonzero vectors.*

The following result acts as a counterpart of Lemma 5.3.1 for NNF. They have been published in [37].

Proposition 5.3.3 *The Boolean function f is non constant if and only if the quantity* $\sum_{u\in\mathbb{F}_2^n | u\neq 0} \tilde{f}(u)2^{-HW(u)}$ *is non null.*

Proposition 5.3.4 *Let f be a non constant Boolean (n)–function such that all the NNF coefficients are known except $\tilde{f}(0)$. Then f can be entirely recovered.*

We are now able to prove that the reduced matrix $P^\star$ is sufficient to get the whole polynomial matrix P.

Proposition 5.3.5 *An (n, m)–function can be recovered from its reduced polynomial matrix coefficients, provided that it is not a constant function.*

5.3.6 Similarity relations between the matrix representations

We prove in this subsection, for $m = n$, similarity relations that relate the polynomial matrix P, the correlation matrix $C = 2^{-n}W$ and the adjacency matrix A (with complex coefficients). We also show that, when the coefficients of the adjacency matrix A and of the polynomial matrix P are considered in $\mathbb{F}_2$, there exists a similarity transform that relates them. This relation allows to simplify the analysis of the eigenstructures of these matrices. This is typically the case for the issue addressed in Section 5.4.

The result below allows to relate the correlation matrix (or the Walsh matrix) of a function to the adjacency matrix of its graph.

Proposition 5.3.6 ([37]) *Let f be an (n, n)–function, then its adjacency matrix A and its correlation matrix C are related by:*

$$A = H^{-1} C^T H, \tag{5.22}$$

where the matrix C^T is the transpose of C and H is the Hadamard matrix (see (5.5)).

The result below allows to relate the polynomial matrix of a function to the adjacency matrix of its graph.

Proposition 5.3.7 ([37]) *Let f be an (n, n)–function, then, its polynomial matrix P and its adjacency matrix A are related by :*

$$P = M^{-1} A M. \tag{5.23}$$

where M is the monomial matrix defined by (5.16).

Remark 5.3.8 *When they are reduced modulo 2, the entries of the matrices M and P can also be considered as elements on $\mathbb{F}_2$, and then Proposition 5.3.7 still holds.*

Considering Propositions 5.3.6 and 5.3.7 and taking into account the fact that a matrix and its transpose are similar, we conclude that A, A^T, P and C are similar matrices.

5.3.7 Matrix representation and composition

It has been noted in [28] that the correlation matrix of the composition of two functions is the product of the correlation matrices of these functions. From Propositions 5.3.6 and 5.3.7, it is clear that this property holds for the adjacency matrix and for the polynomial matrix. As a result, the following proposition holds.

Proposition 5.3.8 ([37]) *If f is a Boolean (n, m)–function and g is a (p, n)–function then, the matrix representations of the composition $f \circ g$, for the adjacency matrix A, correlation*

matrix C and polynomial matrix P, are given by:

$$A_{f\circ g} = A_g A_f \tag{5.24}$$
$$C_{f\circ g} = C_f C_g \tag{5.25}$$
$$P_{f\circ g} = P_f P_g. \tag{5.26}$$

The following relations between matrix product and function composition also hold for reduced correlation and polynomial matrices.

Corollary 5.3.1 *If f is an (n,m)–function and g is a (p,n)–function then:*

$$C^\star_{f\circ g} = C^\star_f C^\star_g \tag{5.27}$$
$$P^\star_{f\circ g} = P^\star_f P^\star_g, \tag{5.28}$$

where $C^\star_f$ (resp. $P^\star_f$) denotes the reduced correlation (resp. the reduced polynomial) matrix of f.

5.4 Eigenanalysis of the matrix representation

Let f be an (n,n)–function, the adjacency matrix A, the polynomial matrix P and the correlation matrix C are square matrices. This section is devoted to the eigenanalysis of these matrices. Due to the similarity relations, the eigenvalue analysis can be done on any of them. The study of the eigenvectors depends on the matrix under consideration. However, as the adjacency matrix has exactly one nonzero component equal to 1 per row, the study is easier on this matrix. As explained in Section 5.3.1, it is possible to associate a graph $\mathcal{G}$ to the function f. This section establishes connections between the eigenanalysis of the matrix representations of a vectorial Boolean function and its graph representation.

We show that the eigenvalues of the representation matrices are directly related to the number of cycles, to their length and to the number of leaves in the graph $\mathcal{G}$. It has been mentioned in Section 5.3.6 that, the adjacency matrix A can be considered either as a $\mathbb{C}$-valued matrix or an $\mathbb{F}_2$-valued matrix. The eigenanalysis below is performed in the field $\mathbb{C}$ of complex number, and thus, the eigenvectors are 2^n-dimensional complex vectors. Hence, each eigenvector can be indexed by the vertices of the graph $\mathcal{G}$ associated to the function f, since those vertices are elements of $\mathbb{F}_2^n$.

Section 5.4.1 is devoted to eigenvalues. In section 5.4.2, we show how to determine the corresponding eigenvectors of the adjacency matrix from the graph $\mathcal{G}$ of the function f.

5.4.1 Eigenvalues

Proposition 5.4.1 ([37]) *The eigenvalues of the matrices A, P and C are either zero or roots of unity.*

The following proposition makes a connection between the eigenvalue zero and the leaves of the graph $\mathcal{G}$.

Proposition 5.4.2 ([37]) *Let f be an (n,n)–function. Zero is an eigenvalue of the adjacency matrix A of f if and only if there exists a leaf in the graph $\mathcal{G}$ of f.*

Remark 5.4.1 *The proof of Proposition 5.4.2 gives a construction of eigenvectors related to the zero eigenvalue. There exists a simpler proof. Assuming that $x \in \{0,1\}^n$ is a leaf, the column x of the adjacency matrix A is null, which implies that the determinant of A is null too. Since this determinant equals the product of the eigenvalues, this means that 0 is an eigenvalue of A.*
Conversely, if 0 is an eigenvalue of A, the kernel of the endomorphism associated to A is not reduced to zero. Thus, this endomorphism is not surjective, which indicates that the graph of f has a leaf.

Remark 5.4.2 *If f is an (n,n)–function, let E_0 be the eigenspace of the eigenvectors associated to the eigenvalue zero of the adjacency matrix A of f. Then, the dimension of E_0 equals the number of leaves in the graph $\mathcal{G}$ of f.*
If v is an eigenvector of the eigenvalue 0 for the adjacency matrix A then, $Av = 0$, and this is equivalent to $v_{f(x)} = 0$ for all $x \in \mathbb{F}_2^n$. Hence, it follows that the support of v is included in the set of the leaves of the graph and then E_0 is spanned by the vectors e_y as defined in the proof of Proposition 5.4.2 where y is a leaf of the graph.

Remark 5.4.3 *The eigenvectors defined in the proof of Proposition 5.4.2 shows that eigenvectors can be interpreted as functions. Indeed, let f be an (n,n)–function and g an (n)–function. If $g \circ f = 0$ then the truthtable of g is an eigenvector for the eigenvalue 0 of A_f. Conversely, assume that g is an eigenvector of A_f associated to the eigenvalue 0 such that all its components are either zero or one then, $g \circ f = 0$.*
Note that whenever the eigenvectors of the adjacency matrix of f associated to 0 are obtained as explained in the proof of Proposition 5.4.2, we can determine all the Boolean functions g for which Remark 5.4.3 applies. They belong to the set of all linear combinations with $\{0,1\}$ coefficients of the eigenvectors associated to 0. There is not any other one. Hence, if there are ℓ leaves in the graph $\mathcal{G}$, there are exactly 2^ℓ (n)–functions g such that $g \circ f = 0$.

We are now interested in nonzero eigenvalues. From Proposition 5.4.1, those eigenvalues are roots of unity.

Proposition 5.4.3 ([37]) *Let f be an (n,n)–function, α be a nonzero eigenvalue of the adjacency matrix of f and let v be an eigenvector for the eigenvalue α. Let ℓ be the order of α and $x \in \{0,1\}^n$ be a n-dimensional binary vector. If the component v_x of vector v is nonzero, then the length of the ultimate cycle of the connected component of the graph $\mathcal{G}$ that contains x is multiple of ℓ.*

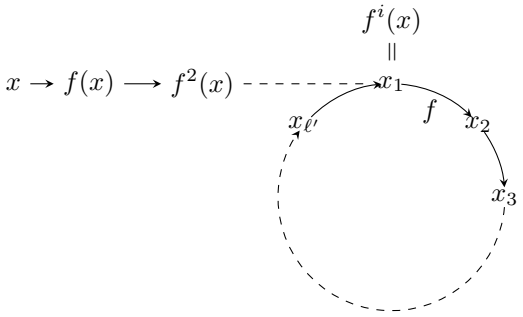

Figure 5.2: Cycle of vertices. The vertex x of the connected component is connected to its ultimate cycle by a path of length i whose elements are the iteration of the (n,n)–function f on the vertex x. This path is $\big(x, f(x), f^2(x), \ldots, f^i(x) = x_1\big)$

Now, let us deal with the dimension of the eigenspaces. Let us denote by E_α the eigenspace associated to the eigenvalue α of the adjacency matrix of the (n,n)–function f. As stated at the beginning of Section 5.4, each eigenvector v associated to α can be indexed by the vertices of the graph $\mathcal{G}$. Thus, the support of v is defined as:

$$\mathrm{supp}(v) = \{x \in \mathbb{F}_2^n \mid v_x \neq 0\}$$

Proposition 5.4.4 ([37]) *Let α be a nonzero eigenvalue of the adjacency matrix of the (n,n)–function f and $\mathcal{C}$ be a connected component of the graph $\mathcal{G}$ whose ultimate cycle length is multiple of the order of α. Then, the subspace of E_α of the vectors whose support is included in $\mathcal{C}$ is of dimension 1.*

The following proposition gives the dimension of the eigenspace E_α.

Proposition 5.4.5 ([37]) *Let α be a nonzero eigenvalue of the adjacency matrix of an (n,n)–function. The dimension of the vectorspace E_α equals the number of cycles in the graph $\mathcal{G}$ whose length are multiple of the order of α.*

According to Remark 5.3.2, there always exists a cycle in the graph and thus 1 is always an eigenvalue of the adjacency matrix.

Corollary 5.4.1 *Let f be an (n,n)–function. The function f is an involution if and only if the eigenvalues of its adjacency matrix are either -1 or 1.*

5.4.2 Eigenvectors

In this section, we are interested in identifying the eigenvectors associated to the eigenvalues for the adjacency matrix A, the polynomial matrix P and the correlation matrix C. Unlike eigenvalues, eigenvectors are basis-dependent. Due to Proposition 5.3.7, eigenvectors of P and

C are easily deduced from eigenvectors of the adjacency matrix A. For each matrix, there is a natural way to derive a basis of the eigenspaces from the graph of the function.

5.4.2.1 Eigenvectors of the adjacency matrix A

The following proposition shows that the eigenvectors of the adjacency matrix A corresponding to the zero eigenvalue are deduced from the junctions of the graph $\mathcal{G}$ of the function f.

Proposition 5.4.6 ([37]) *Assume that the vertex y is a junction of the (n,n)–function f and let x_1 and x_2 be two incident vertices of this junction. Let e_{x_1} (respectively e_{x_2}) be the 2^n-dimensional vector such that all its components are null except the one at coordinate x_1 (respectively x_2) which equals 1. Then, the vector $e = e_{x_1} - e_{x_2}$ is an eigenvector of the matrix A^T for the eigenvalue 0.*

Remark 5.4.4 *According to Remark 5.3.3, from each junction of multiplicity k in $\mathcal{G}$, it is possible to get k independent eigenvectors of A^T for the eigenvalue 0.*

Remark 5.4.5 *Conversely, if v is an eigenvector of A^T for the eigenvalue 0, then the support of v is included in the set of preimages of junctions in the graph $\mathcal{G}$ of f.*

Proposition 5.4.7 *Let $\mathcal{L} = (x_0, \ldots, x_{\ell-1})$ be a cycle of length ℓ of the graph $\mathcal{G}$ associated to an (n,n)–function, and α be a ℓ^{th} root of unity. Then, for i in $\{0, \ldots, \ell-1\}$, the complex number α^i is an eigenvalue of the adjacency matrix A. For every $i \in \{0, 1, \ldots, \ell-1\}$, an eigenvector v of α^i associated to the transpose matrix A^T of A is given by:*

$$v_{x_j} = \begin{cases} \alpha^{i(\ell-j)} & \text{if } x_j \in \mathcal{L} \\ 0 & \text{elsewhere.} \end{cases} \tag{5.29}$$

Proposition 5.4.8 ([37]) *The trace of the adjacency matrix of f is the number of cycles of length one.*

Remark 5.4.6 *As the trace is invariant under similarities, Proposition 5.4.8 also holds for polynomial and correlation matrices.*

Remark 5.4.7 *The result of Proposition 5.4.8 can be proved by noticing that for $x \in \mathbb{F}_2^n$, the coefficient $A_{x,x}$ belonging to the main diagonal of the adjacency matrix A equals 1 if and only if the vector $f(x) = x$. Then the result holds, as the trace of a matrix is the sum of the main diagonal coefficients.*

As an example, we show how to derive the eigenstructures of A_{f_e} based on the graph of Figure 5.1.
According to Proposition 5.4.6, the four eigenvectors associated to 0 are related to:

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

The eigenvectors are denoted by a_0, a_1, a_2, a_3 and can be derived as follows.

$$a_0 = e_{x_1} - e_{x_5} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad a_1 = e_{x_1} - e_{x_7} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \end{pmatrix}$$

$$a_2 = e_{x_5} - e_{x_7} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \\ 0 \end{pmatrix}, \qquad a_3 = e_{x_4} - e_{x_6} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \\ 0 \\ 0 \end{pmatrix}$$

According to Proposition 5.4.7, due to the cycle $(101, 101)$, the following vectors are eigenvectors for the eigenvalues 1.

$$a_4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad a_5 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

These two eigenvectors are obtained from the cycles of length one and three respectively in the graph.

According to Proposition 5.4.7, the following eigenvectors exist and are respectively associated to the eigenvalues $\jmath$ and $\jmath^2$, where $\jmath = \frac{1+\imath\sqrt{3}}{2}$ is a primitive cube root of unity.

$$a_6 = \begin{pmatrix} 0 \\ \jmath \\ \jmath^2 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \; a_7 = \begin{pmatrix} 0 \\ \jmath^2 \\ \jmath \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

The eigenvectors of the correlation and polynomial matrices can be respectively obtained by applying the Hadamard matrix H given by (5.5) and the monomial matrix M given by (5.16) to the vectors a_0, a_1, a_2, a_3, a_4, a_5, a_6.
It is shown in the sequel that, from the eigenvectors of the adjacency matrix A, the change of basis of Proposition 5.3.6 and of Proposition 5.3.7 can be used to determine respectively the eigenspaces of the correlation matrix C and the polynomial matrix P.

5.4.2.2 Eigenvectors of the correlation matrix C

From (5.22), the eigenvectors of C can be deduced from those of A^T. If v is an eigenvector for A^T, then, due to (5.4), the Fourier transform of v, denoted by $\widehat{v} = Hv$ is an eigenvector of the Walsh matrix W and so of the correlation matrix C. Therefore:

$$\forall y \in \mathbb{F}_2^n, \; \widehat{v}_y = \sum_{x \in \mathbb{F}_2^n} v_x (-1)^{x \cdot y}. \tag{5.30}$$

If v is an eigenvector of A^T associated to the eigenvalue 0, then from Remark 5.4.5, the support of v is included in the set of preimages of the junctions in the graph $\mathcal{G}$.

Due to Equation (5.18), the eigenvalues of the Walsh matrix W are merely the eigenvalues of the ones of the correlation matrix C times 2^n. Thus, the eigenvectors are the same.

5.4.2.3 Eigenvectors of the polynomial matrix P

The eigenvectors of the polynomial matrix P can also be deduced from those of the adjacency matrix A by applying (5.23). If v is an eigenvector of A then $\widetilde{v} = Zv$ is an eigenvector of P.

Therefore:

$$\forall y \in \mathbb{F}_2^n,\ \widetilde{v}_y = \sum_{x \in \mathbb{F}_2^n} (-1)^{\mathrm{HW}(x)-\mathrm{HW}(y)} y^x v_x.$$

In the case when v is an eigenvector of A associated to the eigenvalue 0, the support of v is included in the set of the leaves of the graph $\mathcal{G}$.

5.5 Application to Self-Synchronizing Stream Cipher

In Parriaux's thesis [84], it had been shown that the correlation matrix C_{g_θ} of the state transition function g_θ in (2.6) determines whether g_θ is based on a T-function or not. This provides a more general approach to characterize the category of state transition function. The approach provided in Section 2.5 of Chapter 2 is straightforward but works only for LPV systems, since in this case the expression of the inverse transition matrix $P_{\rho(k:k+r)}$ characterizes entirely the triangular structure of g_θ.

Note that g_θ can be expressed as a (n, m)-function when the base field is $\mathbb{F}_2$. For a sake for simplicity we will denote in the sequel g_θ by g.

Let us consider the partial function $g_c : \mathbb{F}_2^n \times \mathbb{F}_2^n$ with $g_c(q_k) = g(q_k, c)$, and $c \in \mathbb{F}_2^{n_s}$. We recall the following Proposition from [87].

Proposition 5.5.1 *Assume that the system (2.6)–(2.10) is finite-time self-synchronizing. And let us consider its state transition function g and the related partial functions g_c, $c \in \mathbb{F}_2^{n_s}$. Then g is not based on a T-function if all the reduced correlation matrices $C^\star_{g_c,}$ are not simultaneously triangularizable.*

We show in the following example that the LPV-based SSSC designed from the graph approach in Chapter 2 provides a state transition function g that is not based on a T-function.

Example 5.5.1 *Let us consider the system of dimension $n = 2$ in Example 2.5.1, where the parameters are elements of the finite field $\mathbb{F}_2$. The input matrix $A_{\rho(k)}$ is given by:*

$$A_{\rho(k)} = \begin{pmatrix} \rho^1(k) & \rho^2(k) \\ 1 & \rho^3(k) \end{pmatrix}$$

with

$$\rho^i(k) : \begin{array}{ccc} \mathbb{F}_2 & \longrightarrow & \mathbb{F}_2 \\ c_k & \mapsto & f(c_k) \end{array}$$

We consider all the functions ρ^i, $i = 1, 2, 3$ as the identity function. From Equation (1.9), we have $n_s = 3$ and the inverse transition matrix $P_{\rho(k:k+2)}$ is given by:

$$P_{\rho(k:k+2)} = \begin{pmatrix} c_{k+1} & c_k \cdot c_{k+1} \\ 1 & c_k \end{pmatrix}$$

Hence, from (2.13), the partial state transition function reads:

$$\begin{array}{rccc} g_c: & \mathbb{F}_2^2 & \to & \mathbb{F}_2^2 \\ & (q_k^1, q_k^2) & \longmapsto & \left(c_k \cdot q_k^1 + c_k \cdot c_{k+1} \cdot q_k^2 + c_{k+2}, q_k^1 + c_k \cdot q_k^2\right) \end{array}$$

The expressions of the eight partial functions $g_c, c \in \mathbb{F}_2^3$ are provided in Table 5.2.

c \ q_k	(0,0)	(0,1)	(1,0)	(1,1)
000	(0,0)	(0,0)	(0,1)	(0,1)
001	(1,0)	(1,0)	(1,1)	(1,1)
010	(0,0)	(0,0)	(0,1)	(0,1)
011	(0,1)	(0,1)	(1,1)	(1,1)
100	(0,0)	(0,1)	(1,1)	(1,0)
101	(1,0)	(1,1)	(0,1)	(0,0)
110	(1,0)	(1,1)	(0,1)	(0,0)
111	(0,0)	(0,1)	(1,1)	(1,0)

Table 5.2: Table of the partial functions g_c, with $c = c_k c_{k+1} c_{k+2}$ and $q_k = (q_k^1, q_k^2)$.

The reduced correlation matrices are given by:

$C^\star_{g_{000}}$	$\begin{pmatrix} 0 & 4 & 0 \\ 0 & 0 & 0 \\ 0 & 4 & 0 \end{pmatrix}$	$C^\star_{g_{100}}$	$\begin{pmatrix} 0 & 0 & 4 \\ 0 & 4 & 0 \\ 4 & 0 & 0 \end{pmatrix}$
$C^\star_{g_{001}}$	$\begin{pmatrix} 0 & 4 & 0 \\ 0 & 0 & 0 \\ 0 & -4 & 0 \end{pmatrix}$	$C^\star_{g_{101}}$	$\begin{pmatrix} 0 & 0 & 4 \\ 0 & -4 & 0 \\ -4 & 0 & 0 \end{pmatrix}$
$C^\star_{g_{010}}$	$\begin{pmatrix} 0 & 4 & 0 \\ 0 & 0 & 0 \\ 0 & 4 & 0 \end{pmatrix}$	$C^\star_{g_{110}}$	$\begin{pmatrix} 0 & 0 & 4 \\ 0 & -4 & 0 \\ -4 & 0 & 0 \end{pmatrix}$
$C^\star_{g_{011}}$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & -4 & 0 \end{pmatrix}$	$C^\star_{g_{111}}$	$\begin{pmatrix} 0 & 0 & 4 \\ 0 & 4 & 0 \\ 4 & 0 & 0 \end{pmatrix}$

Table 5.3: Reduced correlation matrices of the partial functions g_c associated to the transition function g.

And applying Algorithm 1 of Simultaneous Triangularization (see Appendix B) on the set of matrices $C^\star_{g_c}$, $c \in \mathbb{F}_2^3$ shows that these matrices are not simultaneously triangularizable. As

conclusion the state transition function g is not based on a T-function according to Proposition 5.5.1.

5.6 Conclusion

In this chapter, a unified overview on the various representations of vectorial Boolean functions, namely the Walsh matrix, the correlation matrix and the adjacency matrix, has been given. A new representation called polynomial matrix has been introduced with an interest when dealing with algebraic properties. It has been shown that those different representations are similar. Then, an eigenanalysis of those representations has been performed. It has been shown that, for all the representations, the eigenvalues are either zero or roots of unity. For a given vectorial Boolean function with the same number of inputs and outputs, a link has been made between the eigenvalues of its matrix representations and the structure of the graph assigned to this function. The distinction between zero and nonzero eigenvalues plays an important role for that purpose. Finally, the eigenspace associated to the eigenvalues of the matrix representations has been studied. For nonzero eigenvalues, the corresponding eigenvectors can be determined from the cycles of the graph. On the other hand, the eigenvector corresponding to the zero and unique eigenvalue can be determined by the junctions of the graph.

General Conclusions and Perspectives

The aim of this thesis was to propose a new architecture of Self-Synchronizing Stream Ciphers whose next-state function is not based on T-functions and by using LPV systems with flat outputs.

In Chapter 1, an algebraic property has been proposed to characterize LPV systems with flat outputs. A complementary approach based on digraph has been proposed and provides a systematic way to design such LPV systems with flat outputs.

Chapter 2 focuses on SSSC and their representation as automata is provided. Thus a complete framework has been proposed and it is shown how to design LPV-based SSSC and how to replace the parameters within the obtained LPV system with cryptographic primitives such as S-Boxes using secret keys.

Chapter 3 tackles physical security issues of the designed LPV-based SSSC. In particular a spectral approach based on Fourier Transform is proposed in order to perform a side-channel attack. This attack works on the S-Box used within the LPV-based SSSC and can be applied on any other encryption algorithm that involves S-Boxes. Also a multidimensional attack has been proposed from the Fourier transform approach and allows to perform the attack at different times of the encryption process.

Chapter 4 focuses on theoretical security aspects of SSSC by considering their canonical form. It gives a necessary and sufficient condition of the filtering function of the canonical form of SSSC in order to achieve IND-CPA security. Besides it has been show that SSSC cannot achieve IND-CCA security.

Finally Chapter 5 gives results on matrix representations of vectorial boolean functions. The representations that are given are similar. Especially one of the representations that is the correlation matrix characterizes the type of state transition function of a SSSC: that is whether this function is based on a T-function or not. Furthermore, we show that the LPV-based SSSC designed in this thesis admits a state transition function that is not based on a T-function. This provides interesting perspectives to check the class of the state transition function from other approaches that can provide SSSC, other than the graph approach.

The subjects on which we focused in this thesis still leave several questions open:

Statistical SSSC. Statistical SSSC had not been tackled in this thesis. However it remains an open topic in the literature [44, 54]. A statistical SSSC is such that the synchronization delay is not bounded by any value and is a random variable. It then may depend on a pattern of ciphertexts or/and the initial state of the scheme. In Chapter 1, the graph-oriented approach provides sufficient and necessary structural conditions and a systematic way to construct LPV systems with flat output that can be used as SSSC. One can break this flatness property by altering one of these conditions. Besides, the algebraic characterization of the flatness property ensures that a combination of flat LPV system and non-flat LPV system can be also used as an SSSC. Hence the combination of these two systems can be used to provide statistical SSSC. For example one can switch between the two systems by checking whether previous ciphertexts match with a pattern of given ciphertexts.

- **Matrix diffusion.** The synchronization property of the LPV-based SSSC is mainly characterized by the inverse transition matrix. And this matrix is central in the study of the security of the scheme. The non-zero coefficients of this matrix can be replaced by S-Boxes as shown in Chapter 2, to increase the confusion of the scheme and make it resistant against well-known attacks such as linear, differential or distinguishers attacks [11, 64, 13, 65]. Hence the settings of the S-Boxes to achieve such resistances should be studied in more details compared to the work of Section 2.6. The diffusion of the entries of the inverse transition matrix $P_{\rho(k:k+r)}$ should be as complex as possible when considering its successive powers.

- **Hardware implementation.** Specifications have been provided to the industrial partner (Airbus Defence and Space) and have been implemented successfully on FPGA cards. However it could be interesting to make the architecture more compact by improving the circuitry for hardware, and also to reduce the cost of implementation by optimizing the use of cryptographic primitives or subkeys generation in the hardware.

- **Correlation Power Analysis.** The CPA attack proposed in Chapter 3 requires the calculation of the Fourier Transform of the leakage model g and then a number of traces equals to 2^n where n is the size of the input word of g. It would be interesting to see whether the number of traces could be reduced. It means that the attack could be achieved with an approximate value of the Fourier transform. In this case, the multidimensional attack would have a further interest as it can combine reduced Fourier transform for several leakage models. Furthermore, it remains to check how these improvements behave with countermeasures that have been proposed to resist such attack [55], [72], [82], [105].

Appendix A

Basics on Block Ciphers and Self-Synchronizing Stream Ciphers

A.1 Block ciphers

Block ciphers are used for symmetric encryption. This is an extension of simple substitution. The data to encrypt is subdivided in blocks of the same length, for example 64 or 128 binary symbols. A keyed reversible transformation is then performed on each block, involving an encryption algorithm denoted E_k, to get the corresponding ciphertext block. The inverse transformation is performed in order to decrypt the ciphertext blocks. Block cipher is suitable to perform securely encryption or decryption on a single block of data. But encrypting a long plaintext for example requires to run repeatedly the block cipher within a mode of operation. We describe many mode of operations that use block ciphers. Some of the modes of operation act like SSSC.

The Electronic CodeBook mode (ECB) performs a simple substitution. The message to encrypt is subdivided in block of the same length and all the blocks are encrypted the same way. This mode of operation is not semantically secure [8].

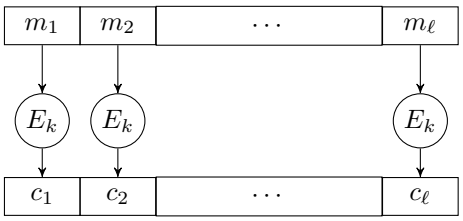

Figure A.1: Electronic CodeBook mode (ECB). The ciphertext is subdivided in blocks and every block is encrypted independently.

The Cipher Block Chaining (CBC) mode is another way to perform a block encryption. Here, the output block of the encryption algorithm E_k is XORed to the next block of the plaintext. Then the resulting block is encrypted to produce the next output (the ciphertext). Note that to decrypt a block m_i of the plaintext, we only need the ciphertexts c_i and c_{i-1} : $m_i = c_{i-1} \oplus E_k^{-1}(c_i)$. It turns out that this mode of operation can be used as SSSC on block of data.

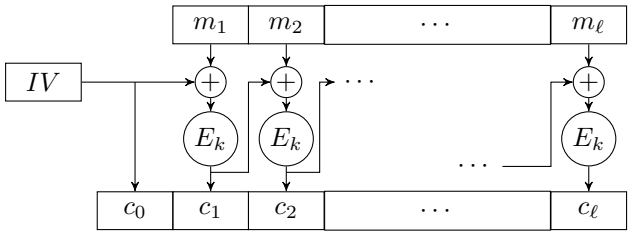

Figure A.2: Chaining mode or Cipher Block Chaining (CBC) mode. The ciphertext block is XORed to the next plaintext block such that every ciphertext block depends on the previous blocks. This mode requires an Initial Vector (IV) block to start the chaining. This IV should be generated randomly and shared through a secure channel between the emitter and the receiver.

In Cipher FeedBack (CFB) mode, the last ciphertext is used as input of the encryption algorithm E_k to produce an output that is XORed to the next plaintext to get the next ciphertext. This mode of operation is suitable to be used as SSSC to encrypt plaintext whose size is lower than the block size (see also Section 2.3 of Chapter 2). A shift register is used in this case to gather previous ciphertexts that are used as input of the encryption algorithm E_k.

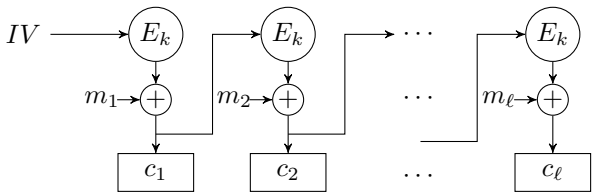

Figure A.3: Cipher FeedBack mode (CFB). The mask is obtained as the encryption of the previous ciphertext block. An IV is then used to compute the first mask. The cipheretxt block is the result of the mask XORed to the plaintext block.

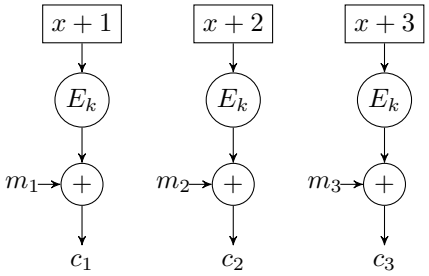

Figure A.4: CTR mode or counter mode. The mask is obtained as the encryption of a counter value that is initialized at given value x. The value of the counter is incremented, for each block, and used to produce another mask that is XORed to the plaintext block to get the ciphertext block.

A.2 Existing Self-Synchronizing Stream Cipher schemes

A.2.1 The Self-Synchronizing Stream Cipher HBB

HHB is a stream cipher designed by Palash Sarkar in [96]. The cipher operates in two modes: a basic mode as a synchronous stream cipher and a self-synchronizing mode. It operates on 128-bit data block and its internal state is partitioned into a linear core of 512 bits denoted LC and a non-linear core of 128 bits denoted NLC. The cores are updated by XORing a register that contains previous ciphertexts with an expansion of the secret key. Unlike others designed SSSC, HHB general structure is similar to a block cipher structure (internal state update in many rounds, input and output are blocks). The following algorithm describes the self-synchronizing mode of HBB: For a message $M = M_0 \parallel \cdots \parallel M_{n-1}$ to be encrypted with a key $\mathbf{K}$ of 128-bits, the corresponding ciphertext is given by the algorithm described below:

HBB(M,$\mathbf{K}$)

1. LC = EXP($\mathbf{K}$); F = Fold($\mathbf{K}$,64); NLC = $F \parallel \bar{F}$
2. for i = 0 to 15 do (T_{imod4}, LC,NLC) = Round(LC,NLC);
3. $\text{LC}_{-1} = \text{LC} \oplus (T_3 \parallel T_2 \parallel T_1 \parallel T_0)$, $\text{NLC}_{-1} = \text{NLC}; C_{-3} = T_3; C_{-2} = T_2; C_{-1} = T_1$;
4. for $i = 0$ to $n - 1$ do
5. $\quad (K_i, \text{LC}_i, \text{NLC}_i) = \text{Round}(\text{LC}_{i-1}, \text{NLC}_{i-1})$;
6. $\quad C_i = M_i \oplus \mathbf{K}_i$;
7. $\quad \text{LC}_i = \text{Exp}(\mathbf{K}) \oplus (C_i \parallel C_{i-1} \parallel C_{i-2} \parallel C_{i-3})$;
8. $\quad \text{NLC}_i = \text{Fold}(\mathbf{K}, 128) \oplus C_i \oplus C_{i-1} \oplus C_{i-2 \oplus C_{i-3}}$;
9. output $C_0 \parallel \cdots \parallel C_{n-1}$;

the function Fold(S,i) computes the sum on i-bit blocks of S, that is Fold(S,i) $= \sum_{j=1}^{k} S_j$ with $|S_j| = i$. And the function Exp($\mathbf{K}$) is an expansion of the key $\mathbf{K}$ that concatenates it with its opposite: Exp($\mathbf{K}$) = $\mathbf{K} \parallel \bar{\mathbf{K}}$. The lines 8 and 9 show that the internal state is a shift of the previous ciphertexts. The function Round is a non-linear transformation that performs

successive operations of permutation and transposition on the linear core and the non-linear core to produce the keystream. Hence, the round function can be identified to a filtering function. The corresponding figure is given in Figure A.5.

The keystream z_k corresponds to $\mathbf{K}_i$ and the keystream generator σ_θ^{ss} is a combination of the functions Round, Fold and Exp:

$$\begin{cases} z_i = & \sigma_\theta^{ss}(C_i, C_{i-1}, C_{i-2}, C_{i-3}) \\ C_i = & \mathbf{K}_i \oplus M_i \end{cases}$$

with $\sigma_\theta^{ss}(C_i, C_{i-1}, C_{i-2}, C_{i-3}) = \text{Round}\big(\text{Exp}(\mathbf{K}) \oplus (C_i \parallel C_{i-1} \parallel C_{i-2} \parallel C_{i-3}), \text{Fold}(\mathbf{K}, 128) \oplus C_i \oplus C_{i-1} \oplus C_{i-2 \oplus C_{i-3}}\big)$[1].

Two cryptanalysis of HBB have been proposed: a known-plaintext attack in [62] that recovers the whole key $\mathbf{K}$ in 2^{66} HBB operations and a differential attack using chosen ciphertexts in [58] that also recovers the whole key.

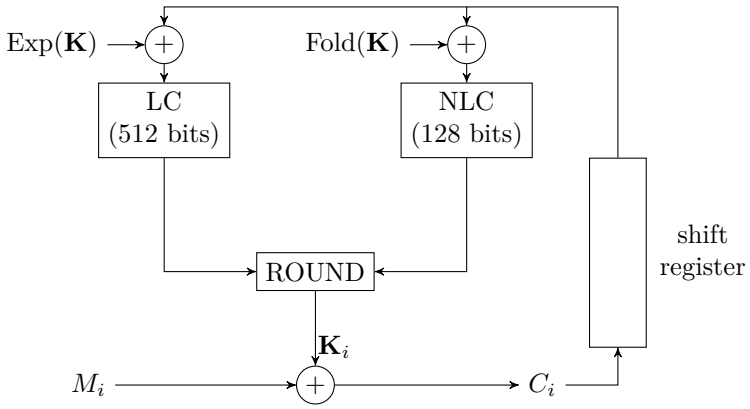

Figure A.5: The Self-Synchronizing Stream Cipher HHB. One round of HBB.

A.2.2 The Self-Synchronizing Stream Cipher SSS

SSS is a self-synchronizing that belongs to SOBER stream ciphers family proposed in [51]. The scheme allows self-synchronization, message authentication code and integrity. It is based on 16-bit operations and 16-bit blocks. Additions and multiplications related to the operations are performed in the finite field GF(2^{16}). The internal state of the scheme is made of 17 blocks of 16 bits called a vector of words. Update of the blocks relies on rotation operations, right shifting, addition on GF(2^{16}) and XOR as illustrated in Figure A.6. The scheme uses a nonlinear keyed function f. Despite the fact that the nonlinear function used in the scheme admits good properties as a high nonlinearity, balanced and pairwise uncorellated, a very efficient attack has been proposed against SSS in [30]. The way the function f is used and also the T-function-like structure of the internal state defaults the scheme. The attack that has been performed is a chosen ciphertext attack that can compute the output of the function f without knowing the

key and hence recovers the entire secret key.

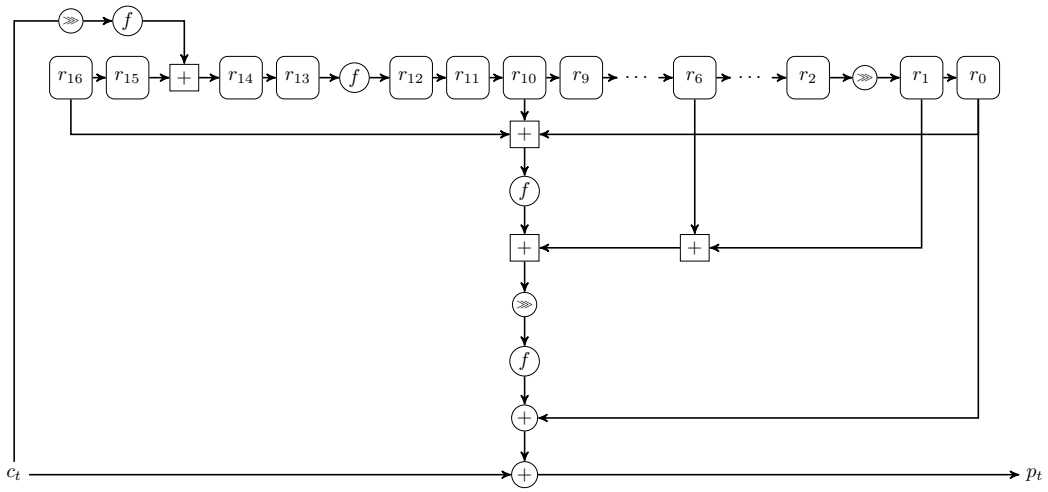

Figure A.6: The Self Synchronizing Stream Cipher SSS. The symbol >>> stands for rotation of the bits to the right, $\boxed{+}$ stands for the addition on GF(16) and $\oplus$ stands for XOR.

A.2.3 Family of the Self-Synchronizing Stream Cipher MOUSTIQUE.

The SSSC MOUSTIQUE [31] belongs to a family of SSSC that are KNOT, ϒΓ, MOUSQUITO and that had been designed by Daemen et al. They had been upgraded subsequently after a weakness was find in their design.

The SSSCs belonging to MOUSTIQUE SSSC family, admit all an architecture that relies on the one proposed by Maurer [74]. Daemen and al. pointed out that is was not very efficient to recompute the whole function g_θ based on serial and parallel automata as proposed by Maurer. And their schemes have been designed to be efficient in one hand relatively to security with regard to more general attacks than the ones considered by Maurer; and in the other hand relatively to speed and easy implementation for hardware.

In the new architecture that they propose, the update function g_θ in Equation (2.10) is implementing in a shift-register-like architecture called Complementing Shift Register (CCSR (see Figure (A.7))). It makes the computation of the internal state more complex. The architecture is pipelined providing a very fast gate delay. There is no key used in the filtering function, because the inputs of the filtering function are unknown values that depend on the key and have been calculated by the CCSR. Like the other designed SSSCs, the architecture is also based on a T-function. One can observe that the propagation goes always in the direction of increasing indexes. This guarantees that the influence of old ciphertext bits eventually vanishes, which makes the resynchronization possible.

KNOT had been tweaked in ϒΓ after the authors discovered later a statistical imbalance in the output function. Besides this security issue, another weakness has been discovered by Joux and Muller in [57]. They proposed an attack based on differential cryptanalysis that detects

collisions in the internal state register of the scheme. MOSQUITO [29] was an upgraded version of ϒΓ. Despite the efficiency of its structure, considering speed and the complexity of the update function, it has been broken in [56]. The weakness disclosed in MOSQUITO has been removed and the scheme has been upgraded to MOUSTIQUE.

The improvement to MOUSTIQUE offers resistance against attacks that make previous SSSC weak. However, in [61], the authors apply successfully a correlation keystream-based attack by exploiting a weakness in the output of some boolean function used in the design of MOUSTIQUE.

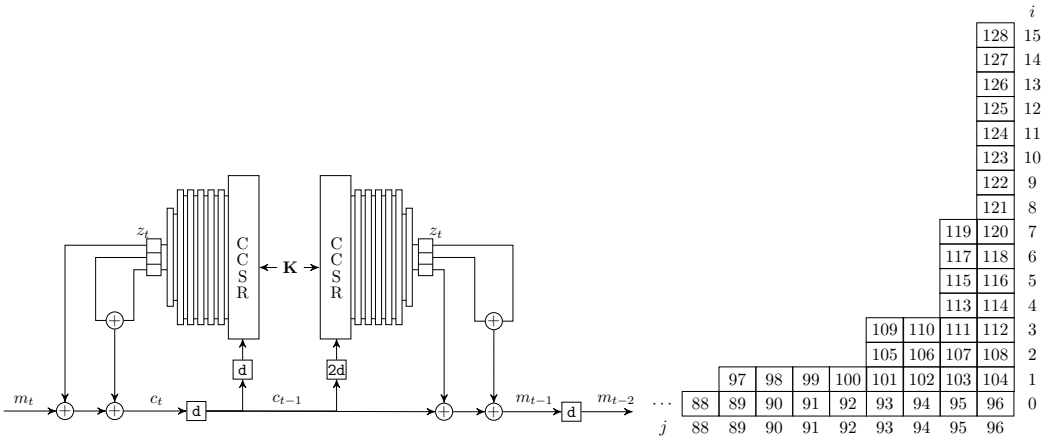

Figure A.7: MOUSTIQUE SSSC. Architecture of SSSC of the family of MOUSTIQUE. The CCSR is described on right.

Appendix B

Simultaneous Triangularization

In [39], the author gave an algorithm that computes for a set of matrices $\mathbb{A}$, provided that $\mathbb{A}$ is simultaneous triangularizable, a matrix basis that triangularizes $\mathbb{A}$. Furthermore, this algorithm aborts if such basis does not exist. A sufficient condition of non-existence of such a basis is that the matrices of $\mathbb{A}$ do not have a common eigenvector.
Before getting in details of the algorithm itself, we present some important definitions and useful results.

Let E an n-dimensional vector space. Let $\mathcal{B}, \mathcal{B}'$, two bases of E, f a linear application on E, A (resp. B) the matrix representation of f with respect to the basis $\mathcal{B}$ (resp. $\mathcal{B}'$). If P is the change of coordinate matrix from $\mathcal{B}$ to $\mathcal{B}'$ then $B = P^{-1}AP$.

B.1 Definition and property

Definition B.1.1 *Let $\mathbb{A} = \{A_1, \cdots, A_n\}$ a set of matrices. The set $\mathbb{A}$ is simultaneous triangularizable if there exists a basis S that triangularizes $\mathbb{A}$, that is*

$$S^{-1}A_jS = T_j \quad \forall j \in \{1, \ldots, n\}$$

where T_j is a triangular matrix. The matrix S is called a T-basis.

A set of matrices $\mathbb{A}$ is simultaneous triangularizable if and only if there exists nested subsets $\{0\} = V_1 \subset V_2 \subset \cdots \subset V_n = E$ such that:

1. $\dim V_i = i$
2. $\forall i$, A_j is V_i-invariant, that is $A_jV_i \subset V_i$

In this case the T-basis S is given by $S = (v_1, \ldots, v_n)$ with $v_i \in V_i$ and where (v_i) are linearly independent. The notation $S = (v_1, \ldots, v_n)$ denotes the change of coordinate matrix from a basis $\mathcal{B}$ (for instance the canonical basis) to the basis (v_i). Hence T_j is the matrix representation of the multiplication by b_j with respect to the basis (v_i).

Theorem B.1.1 ([39]) *Let $\mathbb{A}$ be a set of matrices that is V-invariant, where V is a k-dimensional linear subset spanned by $\{v_1, \ldots, v_k\}$. Let $\mathcal{B}$ be a basis of E containing (v_i) and the matrix $S = (v_1, \ldots, v_n)$. Then for each $A_j \in \mathbb{A}$*

$$S^{-1}A_jS = \begin{pmatrix} \overset{k}{\longleftrightarrow} & \overset{n-k}{\longleftrightarrow} & \\ A_{1j} & A_{2j} & \updownarrow k \\ 0 & A_{3j} & \updownarrow n-k \end{pmatrix}$$

And the set $\mathbb{A}$ is simultaneously triangularizable if and only if both the set $\mathbb{A}_1 = \{A_{11}, \ldots, A_{1n}\}$ and the set $\mathbb{A}_3 = \{A_{31}, \ldots, A_{3n}\}$ are simultaneously triangularizable.

Theorem B.1.2 ([39]) *If $\mathbb{A}$ is simultaneously triangularizable then, the matrices A_j admit a common eigenvector.*

B.2 Simultaneous Triangularization Algorithm (STA)

The algorithm takes as input the set of matrices for which we want to check whether they admit a basis S that triangularizes them together. The algorithm achieves the verification in at most n iterations where during iteration $k < n$, a search for common eigenvector is performed on a set of matrices of dimension $n - k$ (Theorem B.1.2). If a common eigenvector is found, another vector is computed from it and added to a family $\mathcal{B}$ of linearly independent vectors. Actually, Theorem B.1.1 is applied by computing $\mathbb{A}_3$ and checking whether it admits a common eigenvector. Here, one should mention that there is no need to check whether $\mathbb{A}_1$ is simultaneous triangularizable since the process of the algorithm allows to have at each iteration the matrix $\mathbb{A}_1$ as triangular matrix. If no common eigenvector is found at iteration $k < n$ then the algorithm aborts. Otherwise the algorithm is processed until iteration n where a T-basis is inferred from $\mathcal{B}$.

The algorithm requires a set of matrices on which the verification is performed and returns, in case it exists, a basis $\mathcal{B}$ that triangularizes this set. Otherwise the algorithm aborts.

Algorithm 1 Simulatneous Triangularization Algorithm

Require: $A_1, \ldots, A_N$
Ensure: $\mathcal{B}$
Setup: $\mathcal{B} = \emptyset$, $T_j = A_j$, $S_2 = I_n$, $k = 0$
Step 1: find a common eigenvector v of $T_1, \ldots, T_n$. If one does not exists then abort
Step 2: set $v_{k+1} = S_2 v$, $\mathcal{B} = \mathcal{B} \cup \{v_{k+1}\}$, $S_1 = (v_1, \ldots, v_{k+1})$
Step 3: if $k + 1 = n$ go to **Step 6**. Else complete $\mathcal{B}$ to a basis of $\mathbb{F}^n$ with $u_1, \ldots, u_\ell$
Step 4: set $S_2 = (u_1, \ldots, u_\ell)$ $S = (S_1\ S_2)$ and

$$T_j = \begin{pmatrix} 0 & I_{n-(k+1)} \end{pmatrix} S^{-1}A_jS \begin{pmatrix} 0 \\ I_{n-(k+1)} \end{pmatrix}$$

Step 5: set $k = k + 1$ and go to **Step 1**
Step 6: $\mathbb{A}$ is simultaneous triangularizable.

Remark B.2.1 *For a given set of matrices, it suffices to find one of the smallest subsets of matrices that are not simultaneously triangularizable in order to guarantee that the set is not simultaneously triangularizable.*

References

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

[2] P. Apkarian, P. Gahinet, and G. Becker. A convex characterization of gain-scheduled H_∞ Controllers. *IEEE Transactions on Automatic Control*, **40**[5]:853–864, May 1995.

[3] E. Aranda-Bricaire, Ü. Kotta, and C. H. Moog. Linearization of discrete-time systems. *SIAM J. Control Optim.*, **34**[6], November 1996.

[4] François Arnault, Thierry P. Berger, Marine Minier, and Benjamin Pousse. Revisiting LFSRs for cryptographic applications. *IEEE Trans. Information Theory*, **vol. 57**[12]:8095–8113, 2011.

[5] R Jacob Baker. Cmos: circuit design, layout, and simulation, 2011.

[6] G. I. Bara, J. Daafouz, F. Kratz, and J. Ragot. Parameter-dependent state observer design for affine LPV systems. *International Journal of Control*, **74**[16]:1601–1611, 2001.

[7] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE Computer Society Press, 1997.

[8] Mihir Bellare and Phillip Rogaway. Introduction to modern cryptography. In *UCSD CSE 207 Course Notes*, page 207, 2005.

[9] Côme Berbain, Olivier Billet, Anne Canteaut, Nicolas Courtois, Henri Gilbert, Louis Goubin, Aline Gouget, Louis Granboulan, Cédric Lauradoux, Marine Minier, Thomas Pornin, and Hervé Sibert. Sosemanuk, a fast software-oriented stream cipher. In Robshaw and Billet [94], pages 98–118.

[10] Thierry P. Berger, Marine Minier, and Gaël Thomas. Extended generalized Feistel networks using matrix representation. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, **vol. 8282** of *Lecture Notes in Computer Science*, pages 289–305. Springer, 2013.

[11] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. *J. Cryptology*, **vol. 18**[4]:291–311, 2005.

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

[13] Andrey Bogdanov and Vincent Rijmen. Zero-correlation linear cryptanalysis of block ciphers. *IACR Cryptology ePrint Archive*, **vol. 2011**:123, 2011.

[14] Dorothy Bollman, Omar Colón-Reyes, Victor A. Ocasio, and Edusmildo Orozco. A control theory for Boolean monomial dynamical systems. *Discrete Event Dynamic Systems*, **vol. 20**[1]:19–35, 2010.

[15] T. Boukhobza. Partial state and input observability recovering by additional sensor implementation: a graph-theoretic approach. *Intern. J. Syst. Sci.*, **41**[11]:1281–1291, November 2010.

[16] T. Boukhobza and G. Millérioux. Graphical characterization of the set of all flat outputs for structured linear discrete-time systems. In *6th Symposium on System Structure and Control, (SSSC 2016)*, Istanbul, Turkey, June 2015.

[17] Olivier Bournez and Michael Branicky. The mortality problem for matrices of low dimensions. *Theory of Computing Systems*, **35**[4]:433–448, 2002.

[18] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, pages 16–29, 2004.

[19] F. Bruzelius. *Linear Parameter-Varying Systems: an approach to gain scheduling.* PhD thesis, Department of Signals and Systems, Chalmers University of Technology, Göteborg, Sweden, 2004.

[20] B. Buchberger. Gröbner bases: an algorithmic method in polynomial ideal theory. In *Multidimensinal Systems Theory - Progress, Directions and Open Problems in Multidimensional Systems*, pages 184–232. Reidel Publishing Company, 1985.

[21] Christophe De Cannière. Trivium: A stream cipher construction inspired by block cipher design principles. In *Information Security, 9th International Conference, ISC 2006, Samos Island, Greece, August 30 - September 2, 2006, Proceedings*, pages 171–186, 2006.

[22] C. Carlet. Boolean functions for cryptography and error-correcting codes. In *Boolean Models and Methods in Mathematics, Computer Science, and Engineering.* Cambridge Press, 2010.

[23] C. Carlet. Vectorial Boolean functions for cryptography. In *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*. Cambridge Press, 2010.

[24] 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*, **vol. 1719** of *Lecture Notes in Computer Science*, pages 731–731. Springer Berlin / Heidelberg, 1999. 10.1007/3-540-46796-3_10.

[25] A. Chamseddine, Y. Zhang, C.A. Rabbath, C. Join, and D. Theillol. Flatness-based trajectory planning/replanning for a quadrator unmanned aerial vehicle. *IEEE Trans. on Aerospace and electronic systems*, 2012.

[26] J. Daemen. *Cipher and Hash function design, strategies based on linear and differential cryptanalysis.* PhD Thesis, Katholieke Universiteit Leuven, 1995.

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

[28] J. Daemen, R. Govaerts, and J. Vandewalle. Correlation matrices. In *Fast Software Encryption : Second International Workshop, LNCS 1008*, pages 275–285. Springer-Verlag, 1994.

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

[30] J. Daemen, J. Lano, and B. Preneel. Chosen ciphertext attack on SSS. *eSTREAM, ECRYPT Stream Cipher Project, Report 2005/044*, June 2005. Available online at http://www.ecrypt.eu.org/stream/papers.html/044.pdf.

[31] Joan Daemen and Paris Kitsos. The self-synchronizing stream cipher moustique. In Robshaw and Billet [94], pages 210–223.

[32] J.A. De Donà, F. Suryawan, M.M. Seron, and J. Lévine. A flatness-based iterative method for reference trajectory generation in constrained NMPC. In Lalo Magni, Davide Martino Raimondo, and Frank Allgöwer, editors, *Nonlinear Model Predictive Control*, **vol. 384** of *Lecture Notes in Control and Information Sciences*, pages 325–333. Springer Berlin Heidelberg, 2009.

[33] NIST DES. Fips publication 46-3 - Data Encryption Standard, October 25, 1999.

[34] B. Dravie, T. Boukhobza, and G. Millérioux. A mixed algebraic/graph-oriented approach for flatness of SISO LPV discrete-time systems. In *American Control Conference - ACC 2017- The 2017 American Control Conference, Seattle, WA, USA, May 24-26, 2017*, 2017.

[35] B. Dravie, P. Guillot, and G. Millérioux. Security proof of the canonical form of self-synchronizing stream ciphers. In *Ninth International Workshop on Coding and Cryptography 2015*, Paris, France, April 13-17 2015.

[36] B. Dravie, P. Guillot, and G. Millérioux. Design of self-synchronizing stream ciphers: a new control-theoretical paradigm. In *International Federation of Automatic Control - IFAC 2017- The 20th IFAC World Congress, Toulouse, France, July 9-14, 2017*, 2017.

[37] B. Dravie, J. Parriaux, P. Guillot, and G. Millérioux. Matrix representations of vectorial Boolean functions and eigenanalysis. *Cryptography and Communications*, **vol. 8**[4]:555–577, 2016.

[38] Brandon Dravie, Philippe Guillot, and Gilles Millérioux. Security proof of the canonical form of self-synchronizing stream ciphers. *Des. Codes Cryptography*, **vol. 82**[1-2]:377–388, 2017.

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

[40] M. Farhood and G.E. Dullerud. Control of nonstationary LPV systems. *Automatica*, **44**:2108–2119, 2010.

[41] Marco Ferrante and Nadia Frigo. A note on the coupon - collector's problem with multiple arrivals and the random sampling. arXiv:1209.2667 [math.PR], 12 Sep 2012.

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

[43] M. Fliess and R. Marquez. Toward a module theoretic approach to discrete-time linear predictive control. *International Journal of Control*, **73**:606–623, 2000.

[44] A. Geraldy, B. Pfitzmann, and Ahmad-Reza Sadeghi. Optimized self-synchronizing mode of operation. In *Proceedings of Fast Software Encryption International Workshop (FSE'2001)*, 2001.

[45] C. Godsil and G. Royle. *Algebraic Graph Theory*. Springer, 2001.

[46] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, **vol. 33**[4]:792–807, 1986.

[47] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 365–377, 1982.

[48] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, **vol. 28**[2]:270–299, 1984.

[49] P. Guillot. *Fonctions courbes binaires et transformation de Möbius*. PhD thesis, Université de Caen/ Basse Normandie, 1999.

[50] P. Guillot, G. Millérioux, B. Dravie, and N. El Mrabet. Spectral approach for correlation power analysis. In Said El Hajji, Abderrahmane Nitaj, and El Mamoun Souidi, editors, *Codes, Cryptology and Information Security - Second International Conference, C2SI 2017, Rabat, Morocco, April 10-12, 2017, Proceedings - In Honor of Claude Carlet*, **vol. 10194** of *Lecture Notes in Computer Science*, pages 238–253. Springer, 2017.

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

[52] M. Heemels, J. Daafouz, and G. Millérioux. Observer-based control of discrete-time LPV systems with uncertain parameters. *IEEE Trans. Autom. Contr.*, **55**[9]:2130–2135, September 2010.

[53] Martin Hell, Thomas Johansson, and Willi Meier. Grain: a stream cipher for constrained environments. *IJWMC*, **vol. 2**[1]:86–93, 2007.

[54] Howard M. Heys. An analysis of the statistical self-synchronization of stream ciphers. In *Proceedings of IEEE INFOCOM 2001*, pages 22–26, 2001.

[55] Kouichi Itoh, Jun Yajima, Masahiko Takenaka, and Naoya Torii. DPA countermeasures by improving the window method. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pages 303–317, 2002.

[56] A. Joux and F. Muller. Chosen-ciphertext attack against mosquito. *Lecture Note in Computer Science*, 2006.

[57] Antoine Joux and Frédéric Muller. Loosening the KNOT. In *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*, pages 87–99, 2003.

[58] Antoine Joux and Frédéric Muller. Two attacks against the HBB stream cipher. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, **vol. 3557** of *Lecture Notes in Computer Science*, pages 330–341. Springer, 2005.

[59] A. Kaldmae and U. Kotta. On flatness of discrete-time nonlinear systems. In *Proc. of the 9th IFAC Symposium on Nonlinear Control Systems*, Toulouse, France, 2013.

[60] C. Kandler, S.X Ding, T. Koenings, and N. Weihold. A differential flatness based model predictive control approach. In *Proc. of IEEE International Conference on Control Applications (CCA)*, Dubrovnik, 2012.

[61] Emilia Käsper, Vincent Rijmen, Tor E. Bjørstad, Christian Rechberger, Matthew J. B. Robshaw, and Gautham Sekar. Correlated keystreams in moustique. In *Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings*, pages 246–257, 2008.

[62] Vlastimil Klíma. Cryptanalysis of hiji-bij-bij (HBB). *IACR Cryptology ePrint Archive*, **vol. 2005**:3, 2005.

[63] Alexander Klimov and Adi Shamir. New cryptographic primitives based on multiword t-functions. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, **vol. 3017** of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2004.

[64] Lars Knudsen. Deal - a 128-bit block cipher. In *NIST AES Proposal*, 1998.

[65] Lars R. Knudsen and David Wagner. Integral cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers*, **vol. 2365** of *Lecture Notes in Computer Science*, pages 112–127. Springer, 2002.

[66] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, **vol. 1109** of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[67] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, **vol. 1666** of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[68] Balázs Kulcsár and Michel Verhaegen. Robust inversion based fault estimation for discrete-time LPV systems. *IEEE Trans. Automat. Contr.*, **vol. 57**[6]:1581–1586, 2012.

[69] S. M. Lee and J. H. Park. Output feedback model predictive control for LPV systems using parameter-dependent Lyapunov-function. *Applied Mathematics and Computation*, **190**:671–676, 2007.

[70] D.J. Leith and W.E. Leithead. Survey of gain scheduling analysis and design. *International Journal of Control*, **73**:1001–1025, 2000.

[71] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. Northholland Publishing Company, 2nd edition, 1978.

[72] Hideyo Mamiya, Atsuko Miyaji, and Hiroaki Morimoto. Efficient countermeasures against RPA, DPA, and SPA. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, **3156** of *Lecture Notes in Computer Science*, pages 343–356. Springer, 2004.

[73] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[74] Ueli M. Maurer. New approaches to the design of self-synchronizing stream ciphers. In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, **vol. 547** of *Lecture Notes in Computer Science*, pages 458–471. Springer, 1991.

[75] Willi Meier and Othmar Staffelbach. Nonlinearity criteria for cryptographic functions. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings*, **vol. 434** of *Lecture Notes in Computer Science*, pages 549–562. Springer, 1989.

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

[77] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers*, **51**[5]:541–552, May 2002.

[78] T. Meurer. Flatness-based trajectory planning for diffusion-reaction systems in a parallelepipedon - a spectral approach. *Automatica*, **47**[5]:935 – 949, 2011.

[79] G. Millerioux, F. Anstett, and G. Bloch. Considering the attractor structure of chaotic maps for observer-based synchronization problems. *Mathematics and Computers in Simulation*, **68**[1]:67–85, February 2005.

[80] G. Millérioux and J. Daafouz. Flatness of switched linear discrete-time systems. *IEEE Trans. on Automatic Control*, **54**[3]:615–619, March 2009.

[81] G. Millérioux and P. Guillot. Self-synchronizing stream ciphers and dynamical systems: state of the art and open issues. *International Journal of Bifurcation and Chaos*, **20**[9], September 2010.

[82] Amir Moradi and Axel Poschmann. Lightweight cryptography and DPA countermeasures: A survey. In Radu Sion, Reza Curtmola, Sven Dietrich, Aggelos Kiayias, Josep M. Miret, Kazue Sako, and Francesc Sebé, editors, *Financial Cryptography and Data Security, FC 2010 Workshops, RLCPS, WECSR, and WLC 2010, Tenerife, Canary Islands, Spain, January 25-28, 2010, Revised Selected Papers*, **6054** of *Lecture Notes in Computer Science*, pages 68–79. Springer, 2010.

[83] A. Packard and G. Balas. Theory and application of linear parameter-varying control techniques. *Proceedings of the American Control Conference, Tutorial workshop*, 1997.

[84] J. Parriaux. *Control, synchronization and encryption.* PhD thesis, Université de Lorraine, 2012.

[85] J. Parriaux, P. Guillot, and G. Millérioux. A spectral approach for characterizing the self-synchronization of stream ciphers. In *Ecrypt Workshop on Symmetric Encryption (SKEW 2011)*, Lyngby, Denmark, February 2011. paper accessible at http://skew2011.mat.dtu.dk/program.html.

[86] 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.

[87] J. Parriaux and G. Millérioux. A constructive approach for the design of self-synchronizing dynamical systems: an application to communications. In *Proc. of the 8th IFAC World Congress*, Milan, September 2011.

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

[89] J. Parriaux and G. Millérioux. Nilpotent semigroups for the characterization of flat outputs of switched linear and lpv discrete-time systems. *Systems and Control Letters*, **62**[8]:679–685, 2013.

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

[91] Emmanuel Prouff, editor. *Constructive Side-Channel Analysis and Secure Design - 4th International Workshop, COSADE 2013, Paris, France, March 6-8, 2013, Revised Selected Papers*, **vol. 7864** of *Lecture Notes in Computer Science*. Springer, 2013.

[92] H. Radjavi and P. Rosenthal. *Simultaneous Triangularization*. Springer, 2000.

[93] François Robert and J. Jon Rokne. *Discrete iterations : a metric study*. Springer series in computational mathematics. Springer-Verlag, Berlin, New York, Tokyo, 1986.

[94] Matthew J. B. Robshaw and Olivier Billet, editors. *New Stream Cipher Designs - The eSTREAM Finalists*, **vol. 4986** of *Lecture Notes in Computer Science*. Springer, 2008.

[95] W. Rugh and J. Shamma. Research on gain scheduling. *Automatica*, **36**:1401–1425, 2000.

[96] Palash Sarkar. Hiji-bij-bij: A new stream cipher with a self-synchronizing mode of operation. *IACR Cryptology ePrint Archive*, **vol. 2003**:14, 2003.

[97] K. Sato. On an algorithm for checking whether or not a nonlinear discrete-time system is difference flat. In *Proc. of 20th International Symposium on Mathematical Theory of Networks and Systems*, Melbourne, Australia, July 2012.

[98] M. Sato. Filter design for LPV systems using quadratically parameter-dependent lyapunov functions. *Automatica*, **42**:2017–2023, 2006.

[99] C. Scherer. LPV control and full block multipliers. *Automatica*, **37**:361–375, 2001.

[100] O. Schimmel, P. Duplys, E. Bohl, J. Hayek, and W. Rosenstiel. Correlation power analysis in frequency domain, 2010.

[101] O. Sename, P. Gaspar, and J. Bokor, editors. *Robust Control and Linear Parameter Varying Approaches, Application to Vehicle Dynamics*, **437** of *Lecture Notes in Control and Information Sciences*. Springer, 2013.

[102] Akihiro Shimizu and Shoji Miyaguchi. Fast data encipherment algorithm FEAL. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology - EUROCRYPT '87, Workshop on the Theory and Application of of Cryptographic Techniques, Amsterdam, The Netherlands, April 13-15, 1987, Proceedings*, **vol. 304** of *Lecture Notes in Computer Science*, pages 267–278. Springer, 1987.

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

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

[105] Stefan Tillich and Christoph Herbst. Attacking state-of-the-art software countermeasures-a case study for AES. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International*

Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings, **vol. 5154** of *Lecture Notes in Computer Science*, pages 228–243. Springer, 2008.

[106] R. Toth, F. Felici, P.S.C Heuberger, and P. M. J. Van den Hof. Discrete time LPV I/O and state-space representations, differences of behavior and pitfalls of interpolation. In *Proc. of the European Control Conference, (ECC 2007)*, Kos, Greece, 2007. IEEE.

Résumé

Nous présentons dans le cadre de cette thèse une construction effective de chiffreurs par flot auto-synchronisants centrée autour de la classe particulière des systèmes dynamiques Linear Parameter Varying (LPV). Il s'agit de systèmes dont la représentation d'état admet une écriture affine par rapport à l'état et l'entrée mais dont les matrices de la représentation dépendent de paramètres variants dans le temps. Cette dépendance peut se traduire par des fonctions non linéaires de la variable de sortie. La dynamique résultante est donc non linéaire. Nous montrons que la propriété d'auto-synchronisation est liée à une propriété structurelle du système dynamique à savoir la platitude. La platitude est une propriété algébrique qui permet d'exprimer lorsque cela est possible les paramètres d'entrée et sortie d'un système dynamique en fonction de sa sortie qui est appelée dans ce cas une sortie plate. Une caractérisation de la platitude est exprimée en termes des matrices d'état du système dynamique. Une caractérisation complémentaire est proposée en termes de propriétés d'un graphe d'adjacence associé. L'utilisation conjointe de la caractérisation algébrique et graphique donne lieu à une construction systématique d'une nouvelle classe de chiffreurs auto-synchonisants.

Dans la deuxième partie de la thèse, nous nous intéressons à la sécurité de chiffreurs auto-synchronisants. Nous proposons dans un premier temps une approche spectrale pour réaliser une attaque par canaux cachés. Cette approche offre une complexité réduite par rapport aux approches classiques utilisées pour les attaques par canaux cachés. Nous donnons ensuite une preuve de sécurité de la forme canonique d'un chiffreur auto-synchronisant basée sur la notion d'indistinguabilité. Une condition nécessaire et suffisante pour caractériser l'indistinguabilité des chiffreurs auto-synchronisants est proposée. Finalement, nous avons établi des résultats sur les propriétés de fonctions vectorielles booléennes qui permettent de caractériser d'une façon générale les chiffreurs auto-synchronisants.

Mots clés: Systèmes LPV, platitude, chiffreur par flot auto-synchronisant, preuve de sécurité, attaque par canaux cachés

Abstract

In this thesis, we present an effective construction of self-synchronizing stream ciphers based on the class of Linear Parameter-Varying (LPV) dynamical systems. For such systems, the state-space representation admits an affine expression regarding the input and the state but the state matrices depend on time varying parameters. This dependence can be expressed using nonlinear functions of the output variable. Hence, the resulting dynamics of the system are nonlinear. We show that the self-synchronization property is related to a structural property of the dynamical system known as flatness. Flatness is an algebraic property that allows, when possible, the expression of the input and state parameters of a dynamical system as functions of its outputs which is then called flat output. A characterization of the flatness is expressed in terms of state matrices of the dynamical matrix. A complementary characterization is given in terms of properties of the related adjacency graph. The combination of the algebraic and graph theory characterization gives a systematic construction of a new class of self-synchronizing stream ciphers.

In the second part of the thesis, we tackle security aspects of self-synchronizing stream ciphers. We propose a spectral approach to performing side channel attacks. This approach offers reduced complexity when compared with standard approaches used for side channel attacks. We also give a security proof, based on the notion of indistinguishability, for the canonical form of self-synchronizing stream ciphers. A neccessary and sufficient condition is proposed in order to characterize the indistinguishability. Finally, we establish some results on vectorial boolean functions and properties they can be achieved when trying to design Self-Synchronizing Stream Ciphers.

Keywords: LPV systems, flatness, self-synchronizing stream cipher, proof of security, side channel attack