Design of new finite state dynamical systems admitting a matrix representation : Application to cryptography

Hamid Boukerrou

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Design of New Finite State Dynamical Systems Admitting a Matrix Representation: Application to Cryptography

THÈSE

présentée et soutenue publiquement le 04 Avril 2023

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention Automatique, Traitement du signal et des images, Génie informatique)

par

Hamid Boukerrou

Composition du jury

Président :	Jamal Daafouz	Professeur, Université de Lorraine, CRAN
Rapporteurs :	Jean-Pierre Barbot	Professeur, QUARTZ EA 7393 / ENSEA, Cergy-Pontoise
	Gabriela Iuliana Bara	Professeure, Université de Strasbourg, Icube
Examinateurs :	Mirko Fiacchini	Chargé de Recherche, CNRS, GIPSA-lab
	Sihem Mesnager	Professeure, Université Paris VIII, LAGA
	Julien Francq	Expert cybersécurité, Naval Group, Naval Cyber Laboratory
Directeurs de thèse :	Gilles Millerioux	Professeur, Université de Lorraine
	Marine Minier	Professeure, Université de Lorraine

Centre de Recherche en Automatique de Nancy — UMR 7039
Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Remerciements

Je souhaite que celles et ceux qui liront ces lignes comprennent que se livrer, pour moi, est un exercice difficile. Je me livrerai quand même davantage, mais avec la pudeur qui sied à un Berbère !
Mes plus vifs remerciements vont à mes directeurs de thèse Gilles Millerioux et Marine Minier pour leur soutien inlassable, leur dévouement incomparable, leur engagement et la minutie dont ils ont fait preuve tout au long de ma thèse. Je leur dois une grande reconnaissance pour la confiance qu'ils m'ont accordée durant l'aventure. Je souhaite également exprimer ma profonde gratitude à Jean-Pierre Barbot et Gabriela Iuliana Bara d'avoir accepté de rapporter cette thèse et de faire partie de mon jury. Je tiens à remercier sincèrement Mirko Fiacchini, Sihem Mesnager, Jamal Daafouz et Julien Francq pour leur présence lors de mon jury de thèse. Je remercie plus particulièrement Julien et Mirko qui m'ont permis par leur relecture attentive d'améliorer le présent manuscrit. Je tiens également à exprimer ma gratitude aux co-auteurs des publications avec qui j'ai eu le plaisir de collaborer. Enfin, et non le moins important, mes remerciements les plus chaleureux vont à ma famille qui a tant prié pour moi durant toute la période de ma thèse.
Un regret. Je me sens particulièrement ému en pensant à mes deux oncles qui nous ont quittés récemment et regrette qu'ils ne puissent être là pour partager cet accomplissement.

Contents

Glossary

LPV	Linar Paramater Varying
$\mathbb{F}$	finite field
Σ_ρ	state space representation of LPV system
t	the discrete time
n	dimension of the system
x_t	state vector at time t
x_t^i	state vector component number i at time t
u_t	input of a dynamical system
y_t	output of a dynamical system
$\hat{x}_t$	state vector at time t of the left inverse dynamical system.
$\hat{u}_t$	output of the left inverse dynamical system
A_ρ	the dynamical matrix of LPV system
B_ρ	the input matrix of LPV system
C_ρ	the output matrix of LPV system
D_ρ	the direct transfer matrix of LPV system
$\rho(t)$	time varying parameter vector
$\rho^i(t)$	time varying parameter component number i
r	relative degree - inherent delay
Σ	structured linear systems
p	input size
$\mathbf{M}$	the set of p input vertices
I_A	dynamical matrix of the structured linear discrete-time system
I_B	input matrix of the structured linear discrete-time system
$\mathcal{A}$	set of alphabet
$\mathcal{M}$	message space
m	plaintext
c	ciphertext
m_t	symbol at time t of the plaintext
c_t	symbol at time t of the ciphertext
K	master key
K_A	Alice's key
K_B	Bob's key
k	key size
e	encryption function
d	decryption function
M	delay of synchronization
R_i	right halve of plaintext
L_i	left halve of plaintext
f	round function
n_r	the number of rounds

List of Figures

Résumé

Titre. Synthèse de nouveaux automates à états finis décrits par une représentation matricielle: application à la cryptographie.

Préambule. Cette thèse a été financée par le projet référencé ANR-15-IDEX-04-LUE. Il s'agit du projet Citizen Trust in the Digital World (acronyme DigiTrust) qui fait partie de la dernière vague des projets IMPACT au sein de l'initiative Lorraine Université d'Excellence (LUE) proposée dans le cadre de l'appel d'offres PIA2 IDEX/I-SITE. Il été lancé en avril 2019 et son ambition est de construire la confiance des citoyens dans le monde numérique autour de quatre axes de recherche. Le projet LUE DigiTrust vise à mener des recherches autour de la sécurité des données et des systèmes. DigiTrust encourage la recherche interdisciplinaire basée sur la complémentarité des expertises présentes dans les entités de l'Université de Lorraine et de ses partenaires. Pour cette thèse, il s'agit du CRAN (UMR 7039) pour l'aspect théorie du contrôle et Loria (UMR 7503) pour l'aspect cryptographie.

Cryptographie de l'Antiquité à nos jours. L'écriture est l'invention la plus importante de l'histoire de l'humanité, mais depuis que les humains ont la possibilité de partager des informations, ils ont également besoin de les dissimuler. Cela a conduit à l'invention de la cryptographie. Le mot cryptographie tire son nom du mot grec *"kryptos"* et *"graphein"* qui signifie écriture cachée, et désigne un domaine d'étude ancien concernant la façon de préserver des messages secrets à l'abri des tiers malveillants.

Le besoin d'une communication cryptographiquement sécurisée remonte à l'Antiquité. L'une des premières instanciations de la cryptographie, les hiéroglyphes, remonte à 1900 av. J.-C. en Égypte. Bien que les hiéroglyphes ne soient pas le meilleur exemple de communication secrète, ils peuvent être considérés comme la première tentative de mystère entourant la communication. En 486 av. J.-C., en Grèce, les Spartiates utilisaient le *scytale* pour envoyer des messages sensibles en période de combat. Les messages étaient écrits sur une bande de cuir enroulée autour d'une tige de bois d'une taille particulière, ce qui les rendait indéchiffrables jusqu'à ce qu'ils soient enroulés autour d'une tige de bois de taille similaire par les destinataires. Une autre méthode cryptographique a été inventée par les Romains. L'empereur Jules César (100 av. J.-C. - 44 av. J.-C.) utilisait des messages cryptographiques pour communiquer avec ses généraux. Le cryptogramme de César était l'un des premiers systèmes cryptographiques avancés. Dans ce système, chaque lettre du message est remplacée par une autre, suivant un décalage de positions de l'alphabet. Cette quantité était la "clé" nécessaire pour déchiffrer chaque message. Si les ennemis de César récupéraient le message sans avoir la clé, ils penseraient qu'il a été écrit dans une langue étrangère incompréhensible. Le chiffrement de César est un type de *chiffrement par substitution monoalphabétique*. Prenons le mot CAT et avançons dans l'alphabet, trois lettres de chaque lettre existante. Donc la première lettre C sera chiffrée en F et A en D et ainsi de suite. Voilà ce qu'est le chiffrement César. Il n'y a que 26 lettres dans l'alphabet et donc seulement 26 clés possibles. C'est pourquoi le chiffre de César est considéré comme l'un des systèmes de chiffrement les plus simples, car il est relativement facile à casser. En effet, vers 800 ap. J.-C., le mathématicien Al-Kindi a écrit le premier traité de cryptanalyse et a introduit la technique de l'analyse de fréquence, déterminant la lettre la plus fréquente en arabe, ainsi que les lettres qui ne peuvent pas apparaître ensemble. Dans les chiffrements par substitution monoalphabétique, ces propriétés de la langue naturelle sont préservées dans le texte chiffré, ce qui permet à un attaquant de déterminer le décalage approprié et de récupérer le message original. Cette technique a été rendue inefficace après l'invention du *chiffrement par substitution polyalphabétique*. L'un des premiers chiffrements polyalphabétiques était le *chiffre de Vigenère*, développé au 16ème siècle. Ce chiffre utilise une substitution décalée déterminée par un mot clé répété plusieurs fois, couvrant l'ensemble du message.

La cryptographie était utilisée par les gouvernements comme un outil puissant pour atteindre des objectifs diplomatiques et militaires afin d'assurer la sécurité de l'État. Mais cette même technologie pouvait

également constituer une menace lorsqu'elle était entre les mains des ennemis de l'État. Le conflit entre Elisabeth, reine d'Angleterre, et Marie, reine d'Ecosse, en est un exemple. Au milieu du 16e siècle, Marie était arrêtée en Angleterre et complotait pour renverser et assassiner Elisabeth. Et elles transmettaient des messages chiffrés. Le code de Mary était une nomenclature dans laquelle les lettres et les mots étaient remplacés par des symboles spéciaux. Finalement, les espions d'Elizabeth ont réussi à intercepter la correspondance de Mary et à déchiffrer son contenu, découvrant ainsi la clé. Le complot est découvert et la reine Marie est condamnée à mort.

De nouvelles percées dans les techniques cryptographiques en ont fait un outil clé de la guerre moderne. Pendant la seconde guerre mondiale l'Allemagne a déployé la machine Enigma, le dispositif de chiffrement le plus sophistiqué de l'époque. Elle se composait d'un certain nombre de rotors qui pouvaient chiffrer des messages en brouillant les 26 lettres de l'alphabet. La machine Enigma était considérée comme incassable jusqu'à ce qu'un mathématicien britannique, Alan Turing, découvre une faiblesse dans sa mise en œuvre. Turing a inventé la bombe, une machine de décryptage capable de craquer Enigma. Cela a permis aux alliés de remporter la bataille de l'Atlantique, ce qui a permis à l'Amérique de fournir au Royaume-Uni tout l'équipement nécessaire, d'envoyer toutes les troupes.

César a utilisé la cryptographie pour protéger l'empire romain. La reine Mary l'a utilisée pour tenter de renverser la couronne d'Angleterre. Le rôle clé joué par la cryptographie pendant la seconde guerre mondiale a souligné une fois de plus l'importance de cette technologie en matière de sécurité nationale.

Il est clair qu'un solide fondement théorique décrivant les exigences d'un système cryptographique sûr est nécessaire. En 1949, Claude Shannon, considéré comme le père de la cryptographie moderne, a rédigé un article révolutionnaire ([Sha49]) qui a établi la base mathématique de la théorie de l'information et formalisé les principaux objectifs de la cryptographie moderne. Une théorie solide a été développée pour aborder ce que signifie la sécurité d'un système de chiffrement. Cet article a notamment introduit le concept de secret parfait (également appelé sécurité inconditionnelle). Le secret parfait est la notion selon laquelle, étant donné un message chiffré ou un *ciphertext* provenant d'un schéma de chiffrement ou d'un système de chiffrement parfaitement sûr, absolument rien ne sera révélé sur le message ou le *plaintext* par le ciphertext. Shannon a fourni la première preuve mathématique expliquant comment et pourquoi le *one-time pad* est *parfaitement secret*. Le One-time pad - également connu sous le nom de chiffrement de Vernam - est un système de chiffrement inventé au vingtième siècle qui consiste simplement à effectuer un OU Exclusif (XOR) sur chaque texte clair avec une clé à usage unique de même longueur. Le One-time pad fournit un secret parfait si les clés utilisées sont totalement aléatoires, comme l'a prouvé Shannon. Cependant, ce chiffrement est peu pratique en raison du problème de gestion des clés. Pendant la guerre froide, les nouvelles avancées dans ce domaine ont été jalousement gardées secrètes par les superpuissances concurrentes : l'URSS et les États-Unis. Cela a conduit les États-Unis à inclure les technologies cryptographiques dans la liste des munitions des États-Unis, les classant ainsi comme une arme.
La cryptographie n'aurait pas été aussi révolutionnaire sans l'avènement des ordinateurs et de l'internet. Ces technologies ont permis une communication publique chiffrée à l'échelle mondiale. La cryptographie elle-même est devenue un élément fondamental du fonctionnement de l'internet. Du commerce électronique aux courriers électroniques, toutes sortes d'échanges de données privées sur l'internet sont désormais possibles, grâce à la cryptographie. Cela a donné naissance à des algorithmes cryptographiques modernes comme DES [Sta77] en 1977, RSA [RSA78] en 1978, AES [DR99] en 2001, Keccak [Ber+09] et Ascon [Dob+16].

Le développement considérable des nouvelles technologies de communication conduit aujourd'hui à un besoin croissant de sécurisation des échanges d'informations. Dans ce contexte, les systèmes cyber-physiques (CPS), qui relient le monde physique et le monde numérique, méritent une attention particulière car ils sont omniprésents. En effet, tous les opérateurs d'importance vitale (OIV) s'appuient sur les CPS. Les télécommunications, les transports, les banques, l'industrie manufacturière, la production et la distribution d'énergie, les systèmes de contrôle et d'acquisition de données (SCADA), les systèmes médicaux en sont des exemples typiques. L'échange d'informations s'effectue par le biais de réseaux publics. Et donc,

la garantie de la confidentialité de la communication est de première importance.

Ces dernières années, l'automatique a fait une entrée sans précédent sur la scène de la sécurité et la littérature sur le sujet s'accroît rapidement. La notion de *Covert channel* créé en 1973 par Butler Lampson, est un type d'attaque qui crée une capacité à transférer des objets d'information entre des processus qui ne sont pas censés être autorisés à communiquer par la politique de sécurité informatique. À cet égard, l'article [ALY21] (et ses références) explore dans ce contexte l'utilisation d'une approche de la théorie du contrôle pour étudier le problème d'un contrôleur en réseau compromis qui divulgue des informations à un espion ayant accès au canal de mesure. Par ailleurs, les méthodes basées sur le diagnostic se sont révélées efficaces pour la sécurité des CPS. En effet, une stratégie possible pour décider de l'existence d'une attaque adverse est d'exploiter un modèle mathématique précis de la dynamique du système physique sous contrôle et d'analyser toute divergence entre les mesures réelles des capteurs et celles fournies par les modèles. L'estimation des sorties peut être réalisée par des reconstructeurs. À cette fin, les notions de reconstruction d'état, de platitude et d'observabilité de la théorie du contrôle, telles que détaillées dans [Sho+15], jouent un rôle important. La revue bibliographique présentée dans l'article [Sán+19] aborde les perspectives orientées contrôle pour la sécurité des CPS, en complément des approches orientées information ou communication. Depuis des années, une autre approche à part entière pour sécuriser les CPS est la cryptographie, telle que discutée dans [MOV96]. L'objectif est de protéger directement les données véhiculées par des canaux publics. Depuis 2015, plusieurs tentatives d'intégration de la cryptographie dans les systèmes de contrôle en réseau sont apparues pour renforcer la cybersécurité. En se concentrant sur les calculs homomorphes des multiplications dans les schémas de chiffrement RSA et ElGamal, le concept de contrôle chiffré est séduisant, comme rapporté par exemple dans [FZ21]. Il est approprié lorsque des systèmes de contrôle distribués et basés sur le cloud sont recherchés.

En cryptographie, on peut distinguer deux classes de chiffrement : les chiffrements *à clé publique* (ou chiffrements à clé asymétrique) et les chiffrements *à clé secrète* (également appelés chiffrements à clé symétrique). Le présent travail porte sur la cryptographie symétrique. Parmi les chiffrements à clé symétrique, les chiffrements à flot présentent un intérêt particulier pour le chiffrement à haut débit dans les communications par satellite, la diffusion de chaînes de télévision privées, la RFID, les systèmes embarqués en réseau. Soulignons qu'au cours des dernières décennies, au moins deux projets européens ont été consacrés aux chiffrement à flot : le projet NESSIE dans le cadre du programme des technologies de la société de l'information de la Commission européenne qui avait démarré en 2000, suivi par ECRYPT lancé en 2004. Parrainé par ECRYPT, eSTREAM visait à identifier les cryptosystèmes symétriques prometteurs, tant logiciels que matériels, avec des propositions de l'industrie et des universités. Parmi les chiffrements à flot, les chiffrements à flot auto-synchronisants (SSSC), brevetés en 1946, possèdent des propriétés d'auto-synchronisation intrinsèques et sont particulièrement pertinentes pour les communications de groupe. Depuis 1960, des SSSCs spécifiques ont été conçus et sont toujours utilisés pour fournir un chiffrement de masse (pour la ligne Hertzian line, RNIS link, etc.) dans les applications militaires ou les réseaux radio mobiles gouvernementaux.

Au début des années 90, des modèles de SSSC ont été proposés dans [Mau91; DGV92] avec des constructions efficaces [Haw+04; Sar03; DP06]. Jusqu'à présent, tous ces SSSCs ont été cassés comme indiqué dans [JM03; JM05; JM06b; Käs+08; Klí05]. Cela motive une nouvelle méthodologie de conception des SSSCs. D'autre part, comme un débit efficace est une caractéristique importante attendue des SSSCs, il est essentiel qu'une telle considération puisse être prise en compte dans la conception et l'architecture proposée. En guise de contre-exemple, le SSSC peut être naturellement construit en utilisant un chiffrement par blocs en appliquant le mode Cipher Feedback (CFB). Cependant, le coût de calcul de CFB est une opération complète de chiffrement par bloc par bit. Par conséquent, pour les chiffres d'un seul bit, il est L fois moins efficace que les modes de chiffrement à flux synchrone tels que le mode Output Feedback (OFB) ou Counter (CTR), avec L la longueur du bloc. Par exemple, AES en mode CFB à un seul bit (tel que défini dans CFB1-AES128 dans le NIST SP 800-38a [Dwo01]) est 128 fois moins efficace que AES en mode compteur. En conséquence, il semble intéressant de proposer des SSSCs dédiés.

Les automates à états finis sont des primitives particulières largement utilisées en cryptographie symétrique,

en particulier les chiffrement à flot. Ces automates prennent la forme de systèmes dynamiques. Ces objets mathématiques sont également utilisés en automatique car ils peuvent décrire le comportement de systèmes physiques - dans ce cas, ils opèrent sur le champ des nombres réels - ou de systèmes à événements discrets. Dans ce dernier cas, ils opèrent, de manière similaire à la cryptographie, sur des champs finis. À des fins cryptographiques, la conception des machines à états finis doit être guidée par le compromis difficile entre les bonnes propriétés en matière de sécurité et la facilité et l'efficacité de la mise en œuvre. Il s'avère que des machines à états finis admettant de nouvelles représentations matricielles, appelées RLFSM (Rational Linear Finite State Machines) et FCSR (Feedback with Carry Shift Registers), ont été proposées dans [Arn+09; Arn+11a]. Il s'agit d'une généralisation des registres à décalage à rétroaction linéaire (LFSRs Linear Feedback Shifts Register) en étendant l'ensemble des coefficients possibles pour la matrice de transition aux fractions rationnelles. Cette nouvelle approche est un outil intéressant pour construire des circuits plus complexes à partir de LFSMs plus petits avec des propriétés intéressantes.

Cette thèse propose une construction de SSSC dédiés avec une approche dans une veine assez similaire, à savoir l'utilisation de Machines à Etats Finis. Ces machines à états auront une forme appelée Linear Parameter Varying (LPV). Les systèmes linéaires à paramètres variants (LPV) définissent une classe habituelle de systèmes dynamiques rencontrés en contrôle. Les systèmes dynamiques LPV sont décrits par des matrices de transitions d'état où certaines des entrées sont remplacées par des paramètres variant dans le temps. Les systèmes LPV sont attrayants pour leurs non-linéarités inhérentes tandis que la représentation matricielle permet d'aborder de manière efficace les problèmes de synchronisation. Or, il sera montré que la synchronisation est un problème central dans le contexte étudié dans cette thèse, à savoir la conception de nouvelles architectures de chiffrement/déchiffrement. Deux concepts principaux empruntés à la théorie du contrôle seront abordés : la platitude et l'analyse structurelle.

Le contenu de cette thèse est divisé en trois parties : La partie I vise à donner les bases nécessaires sur la théorie du contrôle et sur la cryptographie. Les parties II et III décrivent mes contributions. Tout au long de ce manuscrit, seuls les systèmes dynamiques à temps discret sont considérés.

Résumé du contenu des chapitres. Le chapitre 1 et le chapitre 2 sont principalement consacrés à quelques rappels nécessaires de la théorie du contrôle et de la cryptographie respectivement. Ils facilitent la compréhension des contributions détaillées dans les chapitres 3, 4 et 5.

Chapitre 1. Les fondamentaux des systèmes dynamiques LPV ainsi que des illustrations ont été rappelés. En tant que réalisation particulière d'un système linéaires invariant en temps (LTI) structuré, il a été montré comment un système LPV peut être décrit en termes de graphes. Ensuite, les conditions graphiques données dans le Théorème 1.3 et le Théorème 1.4 ont été rappelées pour caractériser avec une complexité polynomiale les sorties plates d'un système LPV à une entrée et une sortie (SISO) ou à plusieurs entrées et plusieurs sorties (MIMO). Ces conditions sont pratiques, comme on le voit dans la partie II, pour construire des graphes qui donnent des systèmes LPV plats et donc, pour concevoir des SSSCs (Chapitre 3 et Chapitre 4).

Chapitre 2. Dans ce chapitre, les généralités sur la cryptographie symétrique principalement sur les chiffrements à flot ont été rappelées. Les différentes architectures liées aux chiffrement à flot auto-synchronisants ont été détaillées. La cryptanalyse consacrée à l'évaluation de la sécurité des primitives cryptographiques a été abordée. Les principes généraux des attaques connues comme les attaques algébriques, les attaques différentielles et la cryptanalyse linéaire ont été donnés. Ce chapitre a permis d'expliquer comment la construction de systèmes dynamiques, ayant une propriété spécifique, peut être utilisée pour construire une classe spécifique de SSSC.

Chapitre 3. Nous avons fourni une nouvelle méthodologie empruntée à la théorie des graphes et du contrôle pour construire une famille de SSSC déterministes et auto-synchronisants qui peuvent donner des fonctions de transition d'état non triangulaires. Les SSSC font appel à des automates LPV à entrées multiples et sorties multiples pour accélérer le processus de chiffrement. La question d'un compromis entre la synchronisation, la complexité et la sécurité a été analysée. Si la charge de calcul est négligée,

la sécurité concernant l'existence d'une fonction de transition d'état triangulaire (dénommée T-fonction) est garantie. En effet, avec un système dynamique admettant un retard inhérent supérieur à 1, il a été montré qu'une fonction de transitions d'état non triangulaire peut être obtenue. Cependant, la réduction du calcul à son coût le plus bas (délai inhérent égal à 1) conduit nécessairement à des fonctions de transition d'état conjuguées aux T-fonctions et constitue une limitation en matière de sécurité. Dans tous les cas, la diffusion est montrée mauvaise mais est inhérente à un SSSC déterministe. Pour résoudre ce problème, une architecture hybride qui conduit au concept de SSSC statistique a été proposée dans le chapitre suivant.

Chapitre 4. Nous avons d'abord proposé une nouvelle architecture de chiffrement de flux auto-synchronisé statistique. Il s'agit d'une architecture hybride avec deux modes de fonctionnement, chacun étant régi par un modèle LPV présentant une dynamique non linéaire. Il a été expliqué comment le problème de conception peut être reformulé comme une conception de système commuté sous des contraintes de sécurité. Nous avons montré que les concepts de la théorie du contrôle, comme la platitude et l'analyse structurelle, sont toujours pertinentes à cette fin. Nous nous sommes concentrés sur le critère de sécurité spécifique, à savoir la diffusion. Il s'avère qu'avec une telle architecture, la diffusion a été clairement améliorée alors que la complexité de calcul reste réduite. D'autre part, il a été montré comment le concept de *dead-beat stabilizability* (un concept utilisé en théorie du contrôle dans le cadre de la stabilité des systèmes dynamiques) peut donner une approche pour la conception de chiffrement de flot auto-synchronisant impliquant des systèmes linéaires commutés. La sécurité sera abordée dans les travaux à venir. En particulier, la manière d'incorporer la clé secrète est également une question importante.

Chapitre 5. Ce chapitre a étudié le calcul de la probabilité d'un switch boomerang pour les chiffrements de Feistel, un problème qui n'avait pas été abordé jusqu'à présent malgré l'importance de telles constructions. La contrepartie de la BCT (Boomerang Connectivity Table), à savoir la FBCT (Feistel Boomerang Connectivity Table), a été présentée. Les principales propriétés de la FBCT et ses relations avec d'autres tables cryptographiques ont été établies. Nous avons également montré comment étendre un switch boomerang à un nombre arbitraire de tours et fourni une formule générique pour calculer la probabilité correspondante. Cependant, l'application de cette formule à un grand nombre de tours nécessite rapidement une puissance de calcul trop importante. La FBCT reste un outil utile pour trouver les S-boxes ayant les meilleures propriétés ou qui sont les plus susceptibles de conduire à des incompatibilités pour un chiffrement Feistel. Ce chapitre a présenté le travail effectué en master, avec quelques finalisations faites au début de la thèse. Il s'agit d'une vue d'ensemble ; de plus amples détails sont dans [Bou+20].

Annexe A. La nouvelle architecture SSSC statistique proposée dans le Chapitre 4 se base sur des automates linéaires commutés ayant une propriété de dead-beat stabilizability. L'idée présentée consistait à concevoir directement la matrice dynamique P qui définit le déchiffrement. Le code SageMath de l'algorithme qui permet de construire les matrices P est fourni. Les entrées sont la dimension du système n, le nombre de matrices singulières et non singulières. La sortie est la séquence mortelle γ des matrices P impliquées dans le déchiffreur.

Annexe B. Une instanciation d'un chiffrement auto-synchronisant (SSSC) déterministe embarquant un automate LPV MIMO, introduite dans [BMM22], a été mise en œuvre dans une plateforme qui implique des cartes Arduino MEGA. Les informations à chiffrer peuvent être des symboles d'un texte capturé par un écran tactile ou des signaux numérisés délivrés par un capteur de température. Chaque carte peut chiffrer ou déchiffrer des données en fonction de la source d'information utilisée et du canal par lequel les données chiffrés sont acheminées. Les cartes Arduino MEGA impliquent un processeur ATMEGA 2560. Il a une vitesse d'horloge de 16 MHz et une mémoire flash de 256 Ko. L'auto-synchronisation a été mise en évidence dans cet environnement matériel. L'Arduino est équipé de deux cartes configurées pour être connectées l'une à l'autre. La carte 1 est connectée aux capteurs de température qui recueille les données en clair. Les données sont ensuite chiffrés par la carte 1 et envoyées à la carte 2, qui se charge de les déchiffrer. La plateforme permet également d'effectuer une désynchronisation entre les cartes en fixant aléatoirement l'état interne de la carte qui joue le rôle de chiffreur. Il est également possible de modifier

le texte chiffré qui est envoyé au déchiffreur : cela produit également une désynchronisation.

General Introduction

The considerable development of new technologies of communication leads nowadays to an increasing need for security of the exchanges of information. In this context, Cyber-Physical Systems (CPS) that connect the physical and the digital worlds, deserve a special attention since they are ubiquitous. Indeed, all the operators of vital importance (OIVs) rely on CPS. Telecommunication, transportation, banking, manufacturing, power generation and distribution, Supervisory Control And Data Acquisition (SCADA) systems, medical systems are typical examples. Information exchange is carried out through public networks. And thus, guaranteeing privacy in the communication is of first importance.

Since recent years, automatic control has entered the scene of security in an unprecedented fashion and the literature grows up rapidly. *Covert channel*, originated in 1973 by Butler Lampson, is a type of attack that creates a capability to transfer information objects between processes that are not supposed to be allowed to communicate by the computer security policy. In this regard, the paper [ALY21] (and references therein) explores in this context the use of a control-theoretic approach to investigate the issue of a compromised networked controller that leaks information to an eavesdropper who has access to the measurement channel. Besides, diagnosis-based methods have revealed efficiency for the sake of CPS security. Indeed, a possible strategy to decide about the existence of an adversarial attack is to exploit an accurate mathematical model of the dynamics of the physical system under control and to analyze any discrepancy between the actual sensor measurements and the ones delivered by the models. The estimation of the outputs can be carried out by reconstructors. To this end, the control-theoretic notions of state reconstruction, flatness and observability as detailed in [Sho+15] play an important role. The bibliographical review presented in the paper [Sán+19] addresses control-oriented perspectives for CPS security, complementary to information or communication oriented approaches.
For years, another fully-fledged approach to secure CPS is cryptography as discussed in [MOV96]. The aim is to directly protect data conveyed through public channels. Since 2015, several attempts to incorporate cryptography into networked control systems have appeared to enhance cybersecurity. Focusing on the homomorphic computations of multiplications in the RSA and ElGamal encryption schemes, the concept of encrypted control is appealing as reported for example in [FZ21]. It is appropriate when cloud-based and distributed control systems are sought.

In cryptography, two classes of ciphers can be distinguished: *public-key* ciphers (or asymmetric-key ciphers) and *secret-key* ciphers (also called symmetric-key ciphers). The present work is concerned with symmetric cryptography. Among symmetric-key ciphers, stream ciphers are of special interest for high speed encryption in satellite communications, private TV channels broadcasting, RFID, networked embedded systems. Let us stress that in the past decades, at least two European projects have been devoted to stream ciphers: the project NESSIE within the Information Society Technologies Program of the European Commission which had started in 2000, followed by ECRYPT launched in 2004. Sponsored by ECRYPT, eSTREAM aimed at identifying promising both software and hardware oriented symmetric cryptosystems with proposals from industry to academia. Among stream ciphers, Self-Synchronizing Stream Ciphers (SSSC) that were patented in 1946 get inherent self-synchronization properties and are 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, etc.) in military applications or governmental radio mobile networks.

In the early 90s, designs of SSSCs have been proposed in [Mau91; DGV92] with effective constructions [Haw+04; Sar03; DP06]. Until now, all of these SSSC have been broken as reported in [JM03; JM05; JM06b; Käs+08; Klí05]. That motivates a new SSSC design methodology. On the other hand, since an efficient throughput is an important expected feature of SSSC, it is essential that such a consideration can be taken into account in the design and the proposed architecture. As a counterexample, SSSC can be naturally built using a block cipher by applying the Cipher Feedback (CFB) mode. However, the computational cost of CFB is one full block cipher operation per digit. Hence, for single-bit digits, it is L times less efficient than synchronous stream encryption modes such as Output Feedback (OFB) or Counter (CTR) Mode, with L the block length. For example, AES in single-bit CFB mode (as defined as CFB1-AES128 in NIST SP 800-38a [Dwo01]) is 128 times less efficient than AES in counter mode. As a consequence, it seems interesting to propose dedicated SSSC.

Finite State Machines are particular primitives widely used in symmetric cryptography, in particular stream ciphers. Those automata take the form of dynamical systems. Those mathematical objects are commonly used in automatic control as well since they can describe the behavior of physical systems - in that case they operate on the field of real numbers - of discrete-events systems. In the last case, they operate, similarly to cryptography, on finite fields. For cryptographic purposes, the design of Finite State Machines must be guided by the challenging trade-off good properties with respect to security consideration versus ease and efficiency of implementation. It enters the cost effective design paradigm. It turns out that, Finite State Machines admitting new matrix representations, called Rational Linear Finite State Machines and Feedback with Carry Shift Registers, have been proposed in [Arn+09; Arn+11a]. It is a generalization of Linear Feedback Shifts Registers by extending the set of possible coefficients for the transition matrix to rational fractions. This new approach is an interesting tool for constructing more complex circuits from smaller LFSMs with nice properties.

This thesis proposes a construction of dedicated SSSC with an approach in a quite similar vein, that is, the use of Finite State Machines. Those State Machines will get a so-called Linear Parameter Varying (LPV) form. Linear Parameter Varying (LPV) systems define a usual class of dynamical systems encountered in control. LPV dynamical systems are described by state transitions matrices where some of the entries are replaced by time varying parameters. LPV systems are appealing for their inherent nonlinearties while the matrix representation allows to tackle in an efficient way synchronization issues. And yet, it will be shown that synchronization is an issue that is central in the context under consideration in this thesis, namely the design of new cipher/decipher architectures. Two main concepts borrowed from control theory will be addressed: flatness and structural analysis.

The content of this thesis is split into three parts: Part I aims at giving necessary background on control theory and on cryptography. Part II and Part III describe my contributions. All along this manuscript, only discrete-time dynamical systems are considered.

Part I: Background and Preliminaries

The purpose of the first part is to provide the reader with an understanding of both the fundamental concepts of control theory and symmetric cryptography.

Chapter 1. In this chapter, the characterization of flat output of LPV systems is discussed. We start by defining and introducing linear parameter varying (LPV) systems, highlighting their main features. Graph approach can be used to characterize a system by representing it as a directed graph of nodes and edges. The use of the graph approach is shown as a way to characterize flat outputs.

Chapter 2. This chapter reviews the history of cryptography and discusses the foundations of the field. In particular, this chapter focuses on self-synchronization as a key concept in cryptography, specifically in symmetric cryptography involving Self-Synchronizing Stream Ciphers (SSSC). Cryptanalysis, a field of cryptology devoted to the security evaluation of cryptographic primitives, is also discussed.

Part II: Contribution from an automatic control point of view

The second part of this thesis presents the contributions related to the deterministic and statistical Self-Synchronizing Stream Ciphers.

Chapter 3. A new methodology borrowed from graph and control theory is proposed to construct a family of Self-Synchronizing Stream Ciphers. This family involves so-called Linear Parameter Varying (LPV) automata described by matrix equations and having the property of flatness. Flatness is a specific property of dynamical systems that guarantees in this context the construction of automata with finite input memory and thus self-synchronization. As an extension, a Multiple Input Multiple Output (MIMO) LPV systems is investigated to accelerate the encryption process. Indeed, instead of constructing Single Input Single Output (SISO) LPV systems that are used to perform symbol-by-symbol encryption, MIMO LPV systems are sought to allow encrypting several symbols at a time. The performance of the SSSC is evaluated taking into consideration synchronization, complexity and security criteria.

Chapter 4. This chapter discusses statistical Self-Synchronizing Stream Ciphers to resolve the complexity and security issues. This class of SSSC admits a hybrid architecture involving switched LPV systems. In this class of SSSC, the delay of synchronization is not constant. In addition, the chapter deals with statistical SSSC based on switched linear systems. It is shown how the control-theoretic concept of deadbeat stabilizability plays a central role.

Part III: Evaluation of Feistel Structures and its Applications

This part of the discussion focuses on Feistel schemes, which are one of the main constructions used for block ciphers.

Chapter 5. This chapter introduces the Feistel Boomerang Connectivity Table (FBCT), a new tool for computing the probability of the middle round of a boomerang distinguisher for Feistel ciphers. The properties of the FBCT are compared to those of the Boomerang Connectivity Table (BCT), its equivalent for Substitution-Permutation Networks (SPNs), and a general formula used to determine the probability of a boomerang switch over multiple rounds is provided. This approach is then applied to LBlock to build a 16-round distinguisher.

List of Publications

Papers in international conferences

- *H. Boukerrou, G. Millérioux and M. Minier.* Hybrid architecture of LPV dynamical systems in the context of cybersecurity, In 4th IFAC Workshop on Linear Parameter Varying Systems, (LPV 2021), Milano, Italy, July 2021.
- *H. Boukerrou, G. Millérioux, M. Minier and M. Fiacchini.* Construction of dead-beat switched automata: application to cryptography, 10th International Conference on Systems and Control, Marseille, France, November 2022.
- *H. Boukerrou, G. Millérioux and M. Minier.* Towards a New Design of Ciphers to Secure CPS: The Role of Control Theory, 26th International Conference on System Theory, Control and Computing October 19−21 (ICSTCC 2022), 2022, Sinaia, Romania.

Workshop

- *H. Boukerrou, G. Millérioux and M. Minier.* LPV dynamical systems and Self-Synchronizing Stream Ciphers : Stanislas and further. In Journées C2 Codage et Cryptographie, Hendaye, France, April 2022.

Papers in referred journals

- *H. Boukerrou, G. Millérioux, and M. Minier.* Design of hybrid automata for data encryption in the context of cybersecurity. Nonlinear Analysis : Hybrid Systems, 2022, (submitted on July 2022).
- *H. Boukerrou, P. Huynh, V. Lallemand, B. Mandal, M. Minier.* On the Feistel Counterpart of the Boomerang Connectivity Table: Introduction and Analysis of the FBCT, IACR Transactions on Symmetric Cryptology, 2020, pp.331-362.

Part I

Background and Preliminaries

Chapter 1

LPV systems and Characterization of Flatness

Throughout this chapter, we introduce the notions and concepts of control theory. Pursuing this goal, we propose to investigate the class of Linear Parameter Varying (LPV) systems, a usual class of dynamical systems encountered in control theory. The framework of LPV systems concerns linear dynamical systems whose state-space representations depend on parameters expressed as a function of some scheduling variables that can be measured in real-time operation. They are appealing for their inherent non-linearties while the matrix representation allows to tackle in an efficient way synchronization issues. Once the definition of LPV dynamical systems is established, the notion of flatness, a crucial notion in control theory, of a dynamical system is addressed. To characterize the flatness, several approaches will be detailed. The first one is algebraic and based on the state space equations. The second one is graphical which is based on the interpretation of the flatness in terms of the structure of a graph associated to the LPV systems.

1.1 Linear Parameter Varying system

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 [AG], [Pfi+15], [LL00], [Sch01], [LJ07], [FD08] and in observation or filtering [Kan+12a], [HDM10], [Tót+07], [Sat06]. The interest of LPV models is that they provide a systematic procedure to design gain-scheduled controllers like in the area of aerospace control [RS00] or vehicle control [SGB13]. Besides, LPV models can represent nonlinear systems under appropriate conditions. The LPV system of dimension n denoted by Σ_ρ, defined over $\mathbb{R}$, is described by the following state space representation

$$\Sigma_\rho : \left\{ \begin{array}{lcl} x_{t+1} & = & A_{\rho(t)}\ x_t + B_{\rho(t)}\ u_t \\ y_t & = & C_{\rho(t)}\ x_t + D_{\rho(t)}\ u_t \end{array} \right. \tag{1.1}$$

where $t \in \mathbb{N}$ represents the discrete time, $x_t \in \mathbb{R}^n$ is the state vector, $u_t \in \mathbb{R}^{p'}$ is the input, $y_t \in \mathbb{R}^p$ is the output. The matrices $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times p'}, C \in \mathbb{R}^{p\times n}$ and $D \in \mathbb{R}^{p\times p'}$ are the dynamical matrix, the input matrix, the output matrix and the direct transfer matrix, respectively (n, p' and p are integers). The matrix B is the input matrix and defines the component x_t^i on which the input component $u_t^j (j \in \{1, \ldots, p'\})$ is added. Such a system is called Linear Parameter Varying because it is written with a linear dependence on the state vector. The set of all possibly varying parameters of A, B, C and D are collected on a vector denoted by

$$\rho(t) = \left[\ \rho^1(t), \rho^2(t), \ldots, \rho^L(t)\ \right] \in \mathbb{R}^L$$

where $L \in \mathbb{N}$ is the total number of non-zero varying entries. Such automata can exhibit nonlinear dynamics. Indeed, the nonlinearity is obtained by defining the varying parameters $\rho^i(t)$ as functions of

x_t and/or u_t. Let us notice that this notation can be considered as abusive since ρ does not depend explicitly on t. We can use the notation $\rho^i(t)$ even for constant parameters if they are possibly time-varying. The matrices A, B, C and D do not necessarily depend on all the parameters $\rho^i(t)$. The LPV systems are classified according to the source of the varying parameters $\rho^i(t)$. The special systems where the varying parameters $\rho^i(t)$ depend on the system state are referred to as *quasi-LPV* systems [LL99]. For a non-negative integer t_0, a sequence $\{\rho(t_0), \rho(t_0+1), \dots\}$ will be called realization and denoted ρ. Examples of single input and single output (SISO) and multiple input and multiple output (MIMO) systems are considered for illustration.

Example 1.1 *Let us consider the system described by*

$$\left\{\begin{array}{lcl} x^1_{t+1} &=& x^1_t + \rho^1(t)\ x^2_t \\ x^2_{t+1} &=& \rho^2(t)\ x^1_t + \rho^3(t)\ x^2_t + \rho^4(t)\ x^3_t \\ x^3_{t+1} &=& \rho^5(t)\ x^2_t + \rho^6(t)\ x^3_t + \rho^7(t)\ u_t \\ y_t &=& x^1_t \end{array}\right. \tag{1.2}$$

It admits the matrix representation (1.1) *with*

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

It is a single input single output (SISO) LPV system.

Example 1.2 *Let us consider the discrete-time dynamics of an unbalanced nonlinear motor studied in [SB90] described by*

$$\left\{\begin{array}{lcl} \theta_{t+1} &=& \theta_t + T_s\ \omega_t \\ \omega_{t+1} &=& (p(t) \cdot T_s Mgl/J)\ \theta_t + (1 - T_s b/J)\ \omega_t + (T_s K/J)\ I_t \\ I_{t+1} &=& (-T_s K/L)\ \omega_t + (1 - T_s R/L)\ I_t + (T_s/L)\ u_t \end{array}\right. \tag{1.3}$$

where T_s, M, g, l, b, J, K are the parameters of the physical system. The parameter T_s is the sampling time, K is the Motor torque constant, R is the Motor resistance, L is the Motor impedance, J is the Complete disk inertia, b is the Friction coefficient, M is the Additional mass, ℓ is the Mass distance from the disk center and $p(t) \in [0.1, 1]$ is a scheduling time-varying parameter. The system can be written as an LPV system like (1.1) *with state variables x^1_t, x^2_t and x^3_t respectively corresponding to θ_t, ω_t and I_t.*

$$\begin{pmatrix} \theta_{t+1} \\ \omega_{t+1} \\ I_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & T_s & 0 \\ p(t) \cdot T_s Mgl/J & 1 - T_s b/J & T_s K/J \\ 0 & -T_s K/L & 1 - T_s R/L \end{pmatrix} \begin{pmatrix} \theta_t \\ \omega_t \\ I_t \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ T_s/L \end{pmatrix} u_t \tag{1.4}$$

Letting $\rho^1(t) = T_s$, $\rho^2(t) = p(t) T_s Mgl/J$, $\rho^3(t) = 1 - T_s b/J$, $\rho^4(t) = T_s K/J$, $\rho^5(t) = -T_s K/L$, $\rho^6(t) = 1 - T_s R/L$ and $\rho^7(t) = T_s/L$, one has

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

According to the value of y_t, matrix C is defined as follows

- *in the case of $y_t = \theta_t$, the LPV system is single input and single output (SISO)*

$$C = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$$

- *in the case of $y_t = (\theta_t, I_t)$, the LPV system is single input and multiple output (SIMO)*

$$C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Example 1.3 *Let us consider the system described by*

$$\left\{\begin{array}{lcl} x^1_{t+1} &=& \rho^1(t)\ x^3_t + x^4_t \\ x^2_{t+1} &=& x^2_t + x^4_t \\ x^3_{t+1} &=& x^1_t + u^1_t \\ x^4_{t+1} &=& \rho^2(t)\ x^1_t + u^1_t + u^2_t \\ x^5_{t+1} &=& x^2_t + u^2_t \\ y^1_t &=& x^1_t \\ y^2_t &=& x^2_t \end{array}\right. \tag{1.5}$$

The system admits the multiple input multiple output (MIMO) LPV state space description (1.1) *with* $n = 5$ *and* $p = 2$

$$A_{\rho(t)} = \begin{pmatrix} 0 & 0 & \rho^1(t) & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ \rho^2(t) & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}, B_{\rho(t)} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}, C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}, D = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

As an example of quasi-LPV systems, we consider the following autonomous system.

Example 1.4 *We consider the Hénon recurrence [Hén76], of dimension 2, admitting the following state representation:*

$$\left\{\begin{array}{ll} x^1_{t+1} & = -1.4\ (x^1_t)^2 + x^2_t + 1 \\ x^2_{t+1} & = 0.3\ x^1_t \end{array}\right. \tag{1.6}$$

It admits an affine LPV representation, as an extension of the LPV representation (1.1)*, that reads*

$$x_{t+1} = A_{\rho(t)}\ x_t + B_{\rho(t)}\ u_t + E$$

with $A_{\rho(t)} = \begin{pmatrix} \rho^1(t) & 1 \\ 0.3 & 0 \end{pmatrix}$, $B_{\rho(t)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ *and* $E = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ *where* $\rho^1(t) = -1.4\ x^1_t$.

We are interested in the structural properties of LPV automaton defined for any realization of ρ, in other words, these properties can be checked independently of the value of the parameters $\rho^i(t)$. Hence the following section gives a description of structured linear systems.

1.2 Structured linear systems

A structured linear dynamical system denoted by Σ is a linear dynamical system only defined by the sparsity pattern of the state space realization matrices. It admits the state matrix representation

$$\Sigma: \quad x_{t+1} \quad = \quad I_A x_t + I_B u_t. \tag{1.7}$$

where I_A is $n \times n$ matrix and I_B a $n \times p$ matrix. In other words, for a structured linear system, we distinguish between the entries of I_A and I_B that are fixed to zero and the other ones that can take any value in $\mathbb{R}$. The entries of the matrices of (1.7) are symbolically denoted by '0' or '1'. The entries $A(i,j)$ of I_A (*resp.* $B(i,j)$ of I_B) that are '1' mean that there is a relation (dynamical interaction) between the state x^i_{t+1} at time $t+1$ and the state x^j_t at time t (*resp.* the state x^i_{t+1} at time $t+1$ and the input u^j_t at time t). The entries that are '0' mean that there is no relation.

Example 1.5 *Let us consider the LPV system given in Example 1.2 with the setting*

$$A_{\rho(t)} = \begin{pmatrix} 1 & T_s & 0 \\ p(t) \cdot T_s Mgl/J & 1 - T_s b/J & T_s K/J \\ 0 & -T_s K/L & 1 - T_s R/L \end{pmatrix}, B_{\rho(t)} = \begin{pmatrix} 0 \\ 0 \\ T_s/L \end{pmatrix}$$

Let us stress that in the structural analysis framework, all non-zero entries should be assumed independent to avoid pathological cases due to possible singularities. The dynamical matrix and the input matrix I_A and I_B of the corresponding structured linear system read:

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

For Example 1.3, the dynamical matrix and the input matrix I_A and I_B of the corresponding structured linear system read:

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

The structured systems admit a graphical representation. The structural properties can be characterized, in general, by considering necessary and sufficient conditions. The conditions are expressed in terms of graphical conditions that apply to a digraph associated to the structured systems. As a result, the proposed characterization is simple to implement and has polynomial complexity.

1.2.1 Background on digraphs

The structured linear system Σ (1.7) admits an equivalent representation in the form of a digraph $\mathcal{G}$. The digraph $\mathcal{G}$ describing the structured linear system Σ (1.7) is the combination of a vertex set $\mathcal{V}$ and an edge set $\mathcal{E}$. The vertices represent the state and the input components of Σ while the edges describe the dynamic relations between these variables. One has $\mathcal{V} = \mathbf{X} \cup \mathbf{U}$ where $\mathbf{X}$ is the set of state vertices defined as $\mathbf{X} = \{\mathbf{x^1}, \ldots, \mathbf{x^n}\}$ and $\mathbf{U}$ is the set of input vertices, $\mathbf{U} = \left\{\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^{\mathbf{p'}}\right\}$. The edge set is $\mathcal{E} = \mathcal{E}_A \cup \mathcal{E}_B$, with $\mathcal{E}_A = \{(\mathbf{x^i}, \mathbf{x^j}) \,|A(i,j) \neq 0\}$ and $\mathcal{E}_B = \{(\mathbf{x^i}, \mathbf{u^j}) \,|B(i,j) \neq 0\}$. For convenience, we can denote by $\mathbf{v^j}$, $(j = 1, \ldots, p' + n)$ a vertex of the digraph $\mathcal{G}$ regardless whether it is an input vertex (p' vertices) or a state vertex (n vertices).

The connection between the matrices I_A and I_B of the structured linear system Σ (1.7) and the digraph $\mathcal{G}$ can be carried out from the adjacency matrix, denoted by $\mathcal{I}$. It is the $(n + p') \times (n + p')$ matrix

$$\mathcal{I} = \left(\begin{array}{c|c} \mathbf{0}_{p' \times p'} & I_B^T \\ \hline \mathbf{0}_{n \times p'} & I_A^T \end{array}\right) \tag{1.8}$$

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

$$\mathcal{I}_{ij} = \begin{cases} 1 \text{ if there exists an edge from } \mathbf{v^j} \text{ to } \mathbf{v^i} \\ 0 \text{ otherwise.} \end{cases} \tag{1.9}$$

Let us define the set of edges and vertices on the following digraphs of the linear structured system associated to the LPV system (1.3).

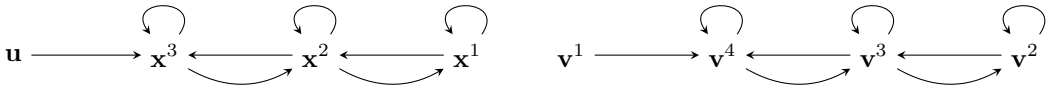


Figure 1.1: Left: digraph involving the vertices $\mathbf{u}$ and $\mathbf{x^i}$. Right: Equivalent digraph involving vertices $\mathbf{v^i}$. The vertex $\mathbf{v^i}$, $i = 1, 2, 3$ is associated to the component x_t^i of the state x_t of (1.3) and $\mathbf{v}^1$ is associated to the input u_t.

The digraph depicted on Figure 1.1 admits the following adjacency matrix

$$\mathcal{I} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Dynamical matrix I_A and the input matrix I_B are given in the corresponding structured linear system.

$$\Sigma: \quad x_{t+1} \quad = \quad \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} x_t + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} u_t. \tag{1.10}$$

Let us recall basic definitions in graph theory.

The SISO case

Example 1.6 *Let us consider the digraph $\mathcal{G}$ depicted in Figure 1.2.*

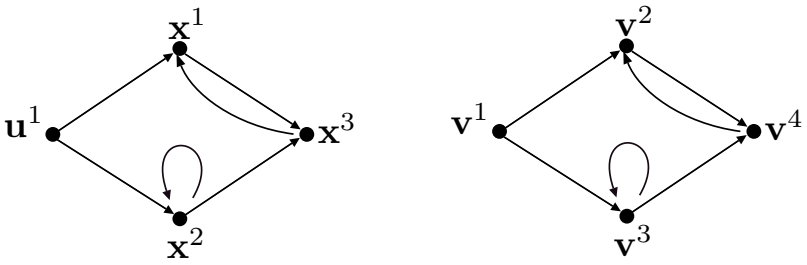


Figure 1.2: The vertex $\mathbf{v^i}$, $i = 1, 2, 3, 4$ is associated to the component x_t^i of the state x_t and $\mathbf{v}^1$ is associated to the input u_t.

The dynamical matrix I_A and the input matrix I_B are given in the corresponding structured linear system Σ (1.11)

$$\Sigma: \quad x_{t+1} \quad = \quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} x_t + \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} u_t. \tag{1.11}$$

The following definitions apply to the digraph depicted on Figure 1.2 (right).

- A directed path $\mathbf{P}$ is a sequence of successive edges directed in the same direction which connect a sequence of vertices. It is said that the path $\mathbf{P}$ covers a vertex if this vertex is the begin or the end vertex of one of the edges of $\mathbf{P}$;
  Example: $\{\mathbf{v^1} \to \mathbf{v^2} \to \mathbf{v^4}\}$ is a path which covers $\mathbf{v^2}$.

- 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 $\mathbf{v^j}$;
  Example: $\mathbf{v^2}$ is a successor of $\mathbf{v^1}$. $\mathbf{v^2}$ and $\mathbf{v^3}$ are both predecessors of $\mathbf{v^4}$.

- A simple path is a directed path where every vertex occurs only once in this path;
  Example: $\{\mathbf{v^1} \to \mathbf{v^2} \to \mathbf{v^4}\}$ is a simple path.

- The length of a directed path $\mathbf{P}$ is equal to the number of edges involved in $\mathbf{P}$. We let $\ell(\mathbf{v^i}, \mathbf{v^j})$ denote the minimal length of a path connecting $\mathbf{v^i}$ to $\mathbf{v^j}$;
  Example: $\{\mathbf{v^1} \to \mathbf{v^2} \to \mathbf{v^4}\}$ connect $\{\mathbf{v^1}$ to $\mathbf{v^4}\}$. One has $\ell(\mathbf{v^1}, \mathbf{v^4}) = 2$.

- A cycle is a simple path linking a vertex $\mathbf{v^i}$ to $\mathbf{v^i}$ with length $\ell(\mathbf{v^i}, \mathbf{v^i}) > 0$;
  Example: $\{\mathbf{v^2} \to \mathbf{v^4} \to \mathbf{v^2}\}$ is a cycle of length 2.

- $V_{ess}(\mathbf{v^i}, \mathbf{v^j})$ is the set of vertices, called essential vertices from $\mathbf{v^i}$ to $\mathbf{v^j}$, which are common to all the paths connecting $\mathbf{v^i}$ to $\mathbf{v^j}$.
  Example: $V_{ess}(\mathbf{v^1}, \mathbf{v^4}) = \{\mathbf{v^1}, \mathbf{v^4}\}$.

Remark 1.1 *If we deal with a MIMO case, additional definitions must be introduced and deserve to be treated in a specific way.*

The MIMO case

Example 1.7 *Let us consider the digraph $\mathcal{G}$ depicted in Figure 1.3*

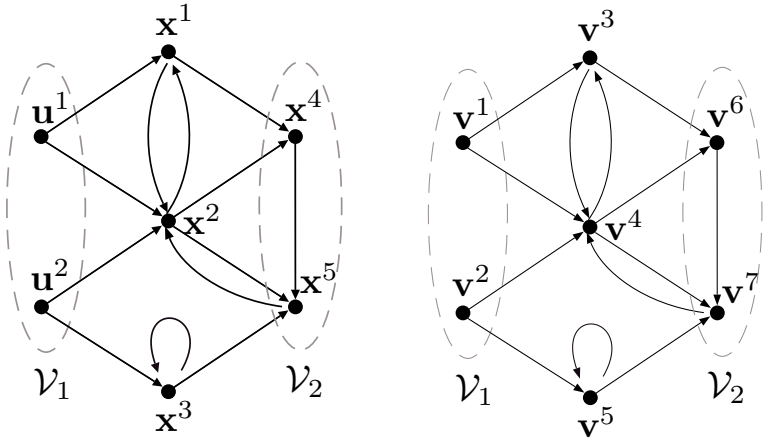


Figure 1.3: Example of digraph $\mathcal{G}$ of a MIMO structural system

The dynamical matrix I_A and the input matrix I_B are given in the corresponding structured linear system Σ (1.12)

$$\Sigma: \quad x_{t+1} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}, x_t + \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} u_t. \tag{1.12}$$

Further definitions are given below. The definitions apply for two sets $\mathcal{V}_1 = \{\mathbf{v^1}, \mathbf{v^2}\}$ and $\mathcal{V}_2 = \{\mathbf{v^6}, \mathbf{v^7}\}$ of vertices. They are illustrated from the digraph depicted on Figure 1.3

- Two paths are disjoint if they have no common vertex;
  Example: $\{\mathbf{v^1} \to \mathbf{v^3} \to \mathbf{v^6}\}$ and $\{\mathbf{v^2} \to \mathbf{v^5} \to \mathbf{v^7}\}$

- A simple path $\mathbf{P}$ is said a $\mathcal{V}_1 - \mathcal{V}_2$ path if its begin vertex belongs to $\mathcal{V}_1$ and its end vertex belongs to $\mathcal{V}_2$. If the only vertices of $\mathbf{P}$ belonging to $\mathcal{V}_1 \cup \mathcal{V}_2$ are its begin and its end vertices, $\mathbf{P}$ is a direct $\mathcal{V}_1 - \mathcal{V}_2$ path;

  Example: The simple paths $\{\mathbf{v^1} \to \mathbf{v^3} \to \mathbf{v^6}\}$ and $\{\mathbf{v^2} \to \mathbf{v^4} \to \mathbf{v^6} \to \mathbf{v^7}\}$ are some examples of $\mathcal{V}_1 - \mathcal{V}_2$. Only the first path is a direct one.

- A cycle is a simple path linking a vertex $\mathbf{v^i}$ to $\mathbf{v^i}$ with length $\ell(\mathbf{v^i}, \mathbf{v^i}) > 0$;
  Example: $\{\mathbf{v^3} \to \mathbf{v^4} \to \mathbf{v^3}\}$ is a cycle of length 2.

- A $\mathcal{V}_1 - \mathcal{V}_2$ linking is a set of disjoints $\mathcal{V}_1 - \mathcal{V}_2$ paths;

  Example 1: $\{\mathbf{v^1} \to \mathbf{v^3} \to \mathbf{v^6}\}$ or $\{\mathbf{v^2} \to \mathbf{v^4} \to \mathbf{v^7}\}$ are examples of $\mathcal{V}_1 - \mathcal{V}_2$ linkings with cardinality equal to 1.

  Example 2: An example of $\mathcal{V}_1 - \mathcal{V}_2$ linking with cardinality equal to 2 is $\{\mathbf{v^1} \to \mathbf{v^3} \to \mathbf{v^6}, \mathbf{v^2} \to \mathbf{v^4} \to \mathbf{v^7}\}$. The set $\{\mathbf{v^1} \to \mathbf{v^3} \to \mathbf{v^6}, \mathbf{v^2} \to \mathbf{v^4} \to \mathbf{v^6} \to \mathbf{v^7}\}$ is not admissible because they are not disjoint.

- A set of maximal number of disjoint $\mathcal{V}_1 - \mathcal{V}_2$ paths is called maximum $\mathcal{V}_1 - \mathcal{V}_2$ linking. The maximal number of disjoint $\mathcal{V}_1 - \mathcal{V}_2$ paths is called the size of the maximum linking and is denoted by $\eta(\mathcal{V}_1, \mathcal{V}_2)$ (Remark: there are possibly several maximum linkings but clearly, they admit the same size $\eta(\mathcal{V}_1, \mathcal{V}_2)$). The number of maximum linkings is denoted by n_{max}. The length of a maximal $\mathcal{V}_1 - \mathcal{V}_2$ linking, denoted by $\ell(\mathcal{V}_1, \mathcal{V}_2)$ is the sum of the length of each disjoint $\mathcal{V}_1 - \mathcal{V}_2$ path;

  Example: The maximum number of disjoint $\mathcal{V}_1 - \mathcal{V}_2$ paths is equal to 2. Hence, the maximum linkings are of size $\eta(\mathcal{V}_1, \mathcal{V}_2) = 2$. The number of $\mathcal{V}_1 - \mathcal{V}_2$ maximum linkings is 4. They are $\{\mathbf{v^1} \to \mathbf{v^3} \to \mathbf{v^6}, \mathbf{v^2} \to \mathbf{v^4} \to \mathbf{v^7}\}$, $\{\mathbf{v^1} \to \mathbf{v^3} \to \mathbf{v^6}, \mathbf{v^2} \to \mathbf{v^5} \to \mathbf{v^7}\}$, $\{\mathbf{v^1} \to \mathbf{v^3} \to \mathbf{v^4} \to \mathbf{v^6}, \mathbf{v^2} \to \mathbf{v^5} \to \mathbf{v^7}\}$, $\{\mathbf{v^1} \to \mathbf{v^4} \to \mathbf{v^3} \to \mathbf{v^6}, \mathbf{v^2} \to \mathbf{v^5} \to \mathbf{v^7}\}$ and $\{\mathbf{v^1} \to \mathbf{v^4} \to \mathbf{v^6}, \mathbf{v^2} \to \mathbf{v^5} \to \mathbf{v^7}\}$. The respective lengths $\ell(\mathcal{V}_1, \mathcal{V}_2)$ are 4,4, 5, 5 and 4.

- $\mu(\mathcal{V}_1, \mathcal{V}_2)$ is the minimal number of vertices covered by a maximum $\mathcal{V}_1 - \mathcal{V}_2$ linking.

  Example: The maximum $\mathcal{V}_1 - \mathcal{V}_2$ linkings with the minimal number of vertices $\mu(\mathcal{V}_1, \mathcal{V}_2) = 6$ are $\{\mathbf{v^1} \to \mathbf{v^3} \to \mathbf{v^6}, \mathbf{v^2} \to \mathbf{v^4} \to \mathbf{v^7}\}$, $\{\mathbf{v^1} \to \mathbf{v^3} \to \mathbf{v^6}, \mathbf{v^2} \to \mathbf{v^5} \to \mathbf{v^7}\}$ and $\{\mathbf{v^1} \to \mathbf{v^4} \to \mathbf{v^6}, \mathbf{v^2} \to \mathbf{v^5} \to \mathbf{v^7}\}$.

- The vertices which are covered by all the maximum $\mathcal{V}_1 - \mathcal{V}_2$ linkings are called the essential vertices for the $\mathcal{V}_1 - \mathcal{V}_2$ linkings. These vertices constitute a specific subset denoted $V_{ess}(\mathcal{V}_1, \mathcal{V}_2)$ which is defined as $V_{ess}(\mathcal{V}_1, \mathcal{V}_2) = \{\mathbf{v} \in \mathcal{V} | \mathbf{v}$ is covered by any maximum $\mathcal{V}_1 - \mathcal{V}_2$ linking$\}$; Otherwise characterized, if $\mathcal{V}^i_{max}(\mathcal{V}_1, \mathcal{V}_2)$ denotes the set of vertices of the $i - th$ $\mathcal{V}_1 - \mathcal{V}_2$ maximum linking $(i = 1, \ldots, n_{max})$ then $V_{ess}(\mathcal{V}_1, \mathcal{V}_2) = \bigcap_{i=1}^{n_{max}} \mathcal{V}^i_{max}(\mathcal{V}_1, \mathcal{V}_2)$.

  Example: $V_{ess}(\mathbf{v^1}, \mathbf{v^3}) = \{\mathbf{v^1}, \mathbf{v^2}, \mathbf{v^6}, \mathbf{v^7}\}$

Now, let us focus on the special property of flatness of the LPV system (1.1). This property is essential in our context as seen in Part II. The next section introduces the definition of difference flatness.

1.3 Difference Flatness for LPV system

Before addressing flatness, the property of left invertibility must be recalled since it is a necessary condition for an output to be flat. There exist different definitions of left invertibility. We recall here a general definition, which is in accordance with the ones proposed in [SM69; Sil69; FB06] in a linear context and in [Sin81] in a nonlinear context.

1.3.1 Left invertibility

Notation: a sequence of vectors z_t indexed by the discrete-time t is denoted by (z).

Definition 1.1 *(Left invertibility) The dynamical system Σ_ρ (1.1) is said to be left invertible if, for a given (infinite length) output sequence (y) produced by Σ_ρ (1.1), the (infinite length) input sequence (u) that produces (y) is unique.*

In other words, the function that defines the output sequence (y) from the input sequence (u) is injective. The characterization of left invertibility can be carried out with the notion of relative degree for LPV SISO systems and inherent delay for LPV MIMO systems.

The SISO case

Definition 1.2 *The relative degree of a discrete-time dynamical system is the minimal number r of iterations such that its output y_{t+r} at time $t+r$ can be written in the explicit way with respect to the input u_t.*

Now, let us particularize this general definition to the system Σ_ρ defined by equation (1.1). To this end, to characterize left invertibility from an algebraic point of view.
In the equation above, we use the quantity $\mathcal{T}_{\rho(t)}$ defined for $j \leq i$ as

$$\begin{array}{ll} \mathcal{T}^{i,j}_{\rho(t)} & = C_{\rho(t+i)} \prod_{l=t+i-1}^{t+j+1} A_{\rho(l)} B_{\rho(t+j)} \text{ if } j \leq i-1 \text{ and} \\ \mathcal{T}^{i,i}_{\rho(t)} & = D_{\rho(t+i)} \end{array} \tag{1.13}$$

From (1.1), it holds that $y_{t+1} = C_{\rho(t+1)} A_{\rho(t)} x_t + \mathcal{T}^{1,0}_{\rho(t)} u_t + \mathcal{T}^{0,0}_{\rho(t+1)} u_{t+1}$. Then, by iterating the output y_t for $i \geq 1$ and noticing that $\mathcal{T}^{i,j}_{\rho(t+1)} = \mathcal{T}^{i+1,j+1}_{\rho(t)}$, it follows that

$$y_{t+i} = C_{\rho(t+i)} \prod_{l=t+i-1}^{t} A_{\rho(l)} x_t + \sum_{j=0}^{i} \mathcal{T}^{i,j}_{\rho(t+1)} u_{t+j} \tag{1.14}$$

Hence, if (1.1) admits a finite relative degree r, it follows from (1.2) that:

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

$$\begin{array}{ll} \mathcal{T}^{i,j}_{\rho(t)} & = 0 \text{ for } i = 0, \ldots, s-1 \text{ and } j = 0, \ldots, i \\ \mathcal{T}^{s,0}_{\rho(t)} & \neq 0 \end{array} \tag{1.15}$$

Hence, it holds that

$$y_{t+r} = C_{\rho(t+r)} \prod_{l=t+r-1}^{t} A_{\rho(l)} x_t + \mathcal{T}^{r,0}_{\rho(t)} u_t$$

The property of left invertibility of (1.1) is equivalent to that (1.1) admits a finite relative degree r. Let us stress that it is well-known that r is bounded by n for Linear Time Invariant systems.

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

$$\widehat{x}_{t+r+1} \quad = \quad P_{\rho(t:t+r)} \widehat{x}_t + B \mathcal{T}^{-1} y_{t+r} \tag{1.16}$$

with

$$P_{\rho(t:t+r)} = A_{\rho(t)} - B \mathcal{T}^{-1} C \prod_{l=t+r-1}^{t} A_{\rho(l)} \tag{1.17}$$

and the function retrieving the input obeys

$$\hat{u}_{t+r} = (\mathcal{T}^{r,0}_{\rho(t)})^{-1} C_{\rho(t+r)} \prod_{l=t+r-1}^{t} A_{\rho(l)} \hat{x}_t - (\mathcal{T}^{r,0}_{\rho(t)})^{-1} y_{t+r} \tag{1.18}$$

Example 1.8 *For Equation* (1.3)*, we show that the system considered is left invertible. Let us check the condition given by Equations* (1.15)

$$\begin{array}{ll} \mathcal{T}^{i,j}_{\rho(t)} & = 0 \text{ for } i = 0, \ldots, 2 \text{ and } j = 0, \ldots, i \\ \mathcal{T}^{3,0}_{\rho(t)} & = C_{\rho(t+3)} \cdot A_{\rho(t+2)} \cdot A_{\rho(t+1)} \cdot B_{\rho(t)} = \rho^1(t+2)\rho^4(t+1)\rho^7(t) \end{array} \tag{1.19}$$

Since $\rho^1(t) = T_s$, $\rho^4(t) = T_s K/J$ and $\rho^7(t) = T_s/L$ are non-zero constants, it turns out that the system is left invertible with relative degree $r = 3$.

The MIMO case

Before recalling a theorem that characterizes the left invertibility of Σ_ρ (1.1) in terms of an equality involving its state space matrices, let us introduce the subsequent vectors and matrices notation:

$$u_{t:t+i} = \begin{pmatrix} u_t \\ u_{t+1} \\ \vdots \\ u_{t+i} \end{pmatrix}, y_{t:t+i} = \begin{pmatrix} y_t \\ y_{t+1} \\ \vdots \\ y_{t+i} \end{pmatrix} \tag{1.20}$$

$$\mathcal{O}_{\rho(t:t+i)} = \begin{pmatrix} C_{\rho(t)} \\ C_{\rho(t+1)}A_{\rho(t)} \\ \vdots \\ C_{\rho(t+i)}A_{\rho(t)}^{\rho(t+i-1)} \end{pmatrix} \tag{1.21}$$

Finally, we recursively define the matrix

$$M_{\rho(t:t+i)} = \begin{pmatrix} D_{\rho(t)} & 0 \\ \mathcal{O}_{\rho(t+1:t+i)}B_{\rho(t)} & M_{\rho(t+1:t+i)} \end{pmatrix} \tag{1.22}$$

with

$$M_{\rho(t:t)} = D_{\rho(t)} = \begin{pmatrix} 0_{p\times p} \end{pmatrix}$$

By successively iterating the output y_t, it holds that for $i \geq 0$:

$$y_{t:t+i} = \mathcal{O}_{\rho(t:t+i)}\ x_t + M_{\rho(t:t+i)}\ u_{t:t+i} \tag{1.23}$$

Hence, the output at time $t+r$ reads:

$$y_{t:t+r} = \mathcal{O}_{\rho(t:t+r)}\ x_t + M_{\rho(t:t+r)}\ u_{t:t+r} \tag{1.24}$$

Left multiplying (1.24) by a matrix $Q \in \mathbb{R}^{p\times(r+1)p}$ yields

$$Qy_{t:t+r} = Q\ \mathcal{O}_{\rho(t:t+r)}\ x_t + Q\ M_{\rho(t:t+r)}\ u_{t:t+r} \tag{1.25}$$

Therefore, Σ_ρ (1.1) is left invertible if and only if there exists a matrix Q such that:

$$Q\ M_{\rho(t:t+r)} = I_{p\times r} \tag{1.26}$$

where $I_{p\times r} = \begin{pmatrix} \mathbf{1}_p & \mathbf{0}_{p\times(p\cdot r)} \end{pmatrix}$ for $r > 0$ and $I_{p\times 0} = \mathbf{1}_p$.

The following theorem, first proved in [SH06], then adapted for LPV systems in [PM13], gives a necessary and sufficient condition for the system Σ_ρ (1.1) to be left invertible with respect to y_t. Then, the expression of the left inverter and its property are given.

Theorem 1.1 *The system Σ_ρ (1.1) is left invertible with respect to the output y_t if and only if there exists a non negative integer $r < \infty$ such that, for all sequences (ρ),*

$$rank \begin{pmatrix} M_{\rho(t:t+r)} \\ I_{p\times r} \end{pmatrix} = rank(M_{\rho(t:t+r)}) \tag{1.27}$$

The least integer r such that Σ_ρ (1.1) is left invertible is called the left inherent delay, a terminology first introduced in [SM69] for Linear Time Invariant (LTI) systems. For a Single Input Single Output system, the left inherent delay and the relative degree coincide one another.

Remark 1.1 *For $r = 0$, Equality* (1.27) *reduces to*

$$rank\ D_{\rho(t)} = p \tag{1.28}$$

Assuming that the system defined by Equations Σ_ρ (1.1) is left invertible with left inherent delay r, the system that allows to recover the input sequence (u) from the output sequence (y) in a sequential way, that is the left inverter reads

$$\left\{ \begin{array}{lcl} \hat{x}_{t+r+1} & = & P_{\rho(t:t+r)}\ \hat{x}_{t+r} + B_{\rho(t)}\ I_{p\times r}\ (M_{\rho(t:t+r)})^\dagger\ y_{t:t+r} \\ \hat{u}_{t+r} & = & -I_{p\times r}\ (M_{\rho(t:t+r)})^\dagger\ \mathcal{O}_{\rho(t:t+r)}\ \hat{x}_{t+r} + I_{p\times r}\ (M_{\rho(t:t+r)})^\dagger y_{t:t+r} \end{array} \right. \tag{1.29}$$

with

$$P_{\rho(t:t+r)} = A_{\rho(t)} - B_{\rho(t)} I_{p\times r} (M_{\rho(t:t+r)})^\dagger \mathcal{O}_{\rho(t:t+r)} \tag{1.30}$$

The matrix $(M_{\rho(t:t+r)})^\dagger$ is the classical Moore-Penrose generalized inverse of $M_{\rho(t:t+r)}$. The matrix $P_{\rho(t:t+r)}$ is the dynamical matrix of the left-inverse dynamical system (1.29). It also governs the dynamics of the error defined by $\varepsilon_t = x_t - \hat{x}_{t+r}$ verifying $\varepsilon_{t+1} = P_{\rho(t:t+r)}\varepsilon_t$. Let us recall that for a given matrix Z, the Moore-Penrose generalized inverse $Z^\dagger$ is a matrix of the same dimension as Z' so that $ZZ^\dagger Z = Z$, $Z^\dagger ZZ = Z^\dagger$, $ZZ^\dagger$ and $Z^\dagger Z$ are Hermitian.

Example 1.9 *For Equations* (1.5)*, the system considered is left invertible. Let us check the condition given by Equations* (1.27)*. The matrices $\mathcal{O}_{\rho(t:t+2)}$, $M_{\rho(t+1:t+2)}$ respectively defined as in Equation* (1.21)-(1.22) *read:*

$$\mathcal{O}_{\rho(t:t+2)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \rho^1(t) & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & \rho^2(t) & 0 \\ 0 & 1 & 0 & \rho^2(t) & 0 \end{pmatrix} \mathcal{M}_{\rho(t:t+2)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \rho^1(t) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Since (1.27) *is verified for $r = 2$, it turns out that the system is left invertible with inherent delay $r = 2$.*

1.3.2 Flatness

Differential flatness is an idea associated with systems of differential equations. It represents the ability to completely parameterize each variable of the system in terms of a finite set of outputs of the system. Flatness was introduced by Fliess et al [Fli+95] in 1995. The counterpart of differential flatness for discrete-time systems is called difference flatness and its treatment for LTI systems can be found in the book [SA04]. In this thesis, only difference flatness will be addressed. Hence, without any confusion, we will merely use the vocable flatness. Flatness is an important notion that admits various and diverse application domains such as path planning [Cha+12], [Meu11], predictive control or constraint management [De +09], [FM00], [Kan+12b]. We say that a system is flat if it admits a flat output. Beyond a simple verification of the flat outputs of an LPV system, the construction of LPV systems that admit flat outputs is provided in [Dra17].

Definition 1.3 *(flat output) The system Σ_ρ* (1.1) *is said to be generically flat if there exists a non negative integer t_0 such that every variable of the system can be expressed as a function of $y_t \in \mathbb{R}^p$ and a finite number of its backward and/or forward iterates for $t \geq t_0$. In particular, there exist two functions $\mathcal{F}_\rho$ and $\mathcal{G}_\rho$ parameterized by ρ such that:*

$$\left\{ \begin{array}{lcl} x_t & = & \mathcal{F}_\rho(y_{t+t_\mathcal{F}}, \ldots, y_{t+t'_\mathcal{F}}), \\ u_t & = & \mathcal{G}_\rho(y_{t+t_\mathcal{G}}, \ldots, y_{t+t'_\mathcal{G}}) \end{array} \right. \tag{1.31}$$

where $t_\mathcal{F}$, $t'_\mathcal{F}$, $t_\mathcal{G}$ and $t'_\mathcal{G}$ are $\mathbb{Z}$-valued integers satisfying $t_\mathcal{F} \leq t'_\mathcal{F}$ and $t_\mathcal{G} \leq t'_\mathcal{G}$.

Example 1.10 *For Equations* (1.2)*, y_t is a flat output. Indeed, Definition 1.3 is satisfied with* (1.31) *which reads*

$$\begin{aligned}
x_t^1 =& y_t \\
x_t^2 =& \frac{y_{t+1}}{T_s} + \left(\frac{1 - T_s b/J}{T_s} - (2 - T_s b/J)\right) T_s (K/J)\, y_t \\
x_t^3 =& -\left(1 + T_s^2 p_{t+1} Mgl/J\right) + \left(2 - T_s b/J\right) T_s/L + T_s^2 \left(2 - T_s b/J\right)^2 K/J
\end{aligned}$$

$$u_t = \frac{JL}{T_s^3 K}\left(y_{t+3} - P_1(t)\, x_t^1 - \left(P_2(t) + \tfrac{T_s^3 K^2}{LJ}\right) x_t^2 + \left(P_3(t) + \tfrac{T_s^2 K}{J}\tfrac{1 - T_s R}{L}\right) x_t^3\right)$$

with

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

1.4 Flat Outputs Characterization

1.4.1 Inverse system approach

A characterization of the flat output based on the inverse system has been proposed for switched linear systems in [MD09]. The extension of this characterization to LPV systems has been provided in [PM13] by considering the switching function as a function that takes values in a continuum rather than a finite set (the modes). The characterization of a flat output for LPV systems is given by the following theorem recalled from [PM13].

Theorem 1.2 *An output y_t of the system* (1.1)*, assumed to be left invertible with left relative degree (inherent delay) r, is a flat output if and only if there exists a non negative integer t_0 and a positive integer L, such that, for all $t \geq t_0$ and all sequences (ρ):*

$$P_{\rho(t+L-1:t+L-1+r)} P_{\rho(t+L-2:t+L-2+r)} \cdots P_{\rho(t:t+r)} \quad = 0 \tag{1.32}$$

with

$$P_{\rho(t:t+r)} = A_{\rho(t)} - B_{\rho(t)} (\mathcal{T}^{r,0}_{\rho(t)})^{-1} C_{\rho(t+r)} \prod_{l=t+r-1}^{t} A_{\rho(l)} \quad \textit{(SISO)} \tag{1.33}$$

$$P_{\rho(t:t+r)} = A_{\rho(t)} - B_{\rho(t)} I_{p\times r} (M_{\rho(t:t+r)})^{\dagger} \mathcal{O}_{\rho(t:t+r)} \quad \textit{(MIMO)} \tag{1.34}$$

Example 1.11 *For Equations* (1.5)*, we want to show that $y_t = Cx_t = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} x_t$ is a flat output. Let us address the left invertibility property since it is a necessary condition for flatness. To this end, let us check the condition given by Equations* (1.27)*. It turns out that the system is left invertible with inherent delay $r = 2$. The matrices $\mathcal{O}_{\rho(t:t+2)}$, $M_{\rho(t+1:t+2)}$ respectively defined as in Equation* (1.21) *and Equation* (1.22) *read:*

$$\mathcal{O}_{\rho(t:t+2)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \rho^1(t) & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & \rho^2(t) & 0 \\ 0 & 1 & 0 & \rho^2(t) & 0 \end{pmatrix} \mathcal{M}_{\rho(t:t+2)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \rho^1(t) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Let us compute the matrix P as defined in Equation (1.34) *which reads, for $r = 2$ as*

$$P_{\rho(t:t+2)} = \begin{pmatrix} 0 & 0 & \rho^1(t) & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\rho^2(t) & 0 \end{pmatrix}$$

The matrix $P_{\rho(t:t+2)}$ verifies $P_{\rho(t)} \cdot P_{\rho(t+1)} = 0$ and thus, Theorem 1.2 is fulfilled with $L = 2$. Thus, the system is flat.

Checking that y_t of the automaton is a flat output is like checking the product (1.32) for an infinite number of realizations ρ, which is difficult in practice. From this perspective, the graph-oriented approach proposed in [MB15] can act as an alternative.

1.4.2 Necessary and sufficient conditions for flatness based on digraph

The graph-oriented approach proposed in [MB15] provides the necessary and sufficient conditions to characterize flat outputs. Let us notice that this characterization applies to a restricted set of outputs, the outputs that consist of a set of state components, $y_t^j = x_t^i$, $j \in \{1, \ldots, p\}$ and $i \in \{1, \ldots, n\}$. Hereafter, such outputs will be called canonical outputs. Let us recall the necessary and sufficient conditions which must be satisfied for any vertex $\mathbf{v^i}$ ($i \in \{1, \ldots, n\}$) of the digraph $\mathcal{G}$ to be associated to a flat output.

The SISO case

Theorem 1.3 *The output $y_t^F = x_t^i$ ($i \in \{1, \ldots, n\}$) of the structured linear system* (1.7)*, associated to the vertex $\mathbf{v^F} \in \mathbf{X}$ in the associated digraph $\mathcal{G}$, is generically a flat output if and only if, the three following conditions hold:*
C0. *$\mathbf{v^F}$ is a successor of $\mathbf{u^1}$;*
C1. *Any simple paths from $\mathbf{u^1}$ to $\mathbf{v^F}$ have the same length equal to $\ell(\mathbf{u^1}, \mathbf{v^F})$;*
C2. *Any cycles cover at least an element of $V_{ess}(\mathbf{u^1}, \mathbf{v^F})$.*

Example 1.12 *(Search for the flat outputs with the graph-based condition). The digraph of the linear structured system associated to the LPV system* (1.3) *is depicted on Figure 1.4.*

Figure 1.4: Example of digraph of the linear structured system associated to the LPV system (1.3).

The aim is to know if $\mathbf{x^1}$ is a flat output of (1.7) *by checking the conditions **C0-C2**. It follows that:*

C0. *Condition **C0** is satisfied since there exists a path linking $\mathbf{u^1}$ to $\mathbf{x^1}$.*

C1. *Condition **C1** is also satisfied because there exists one simple path linking $\mathbf{u^1}$ to $\mathbf{x^1}$ have the length $\ell(\mathbf{u^1}, \mathbf{x^1}) = 3$.*

C2. *Furthermore, Condition **C2** is satisfied since the cycles involve $\mathbf{x^1}$, $\mathbf{x^2}$ and $\mathbf{x^3}$ which are all elements of $V_{ess}(\mathbf{u^1}, \mathbf{x^1})$. Thus, $\mathbf{x^1}$ is a flat output.*

The MIMO case

Theorem 1.4 *Consider the structured linear discrete-time system described by Equation* (1.7)*. The output vector $y_t \in \mathbb{R}^p$ associated to a specific set of vertices $\mathbf{V^F} \subseteq \mathbf{X}$ is a flat output if and only if, in the*

associated digraph $\mathcal{G}$, the following three conditions hold:
C0. *$\eta(\mathbf{U}, \mathbf{V^F}) = card(\mathbf{U})$;*
C1. *All the maximum $\mathbf{U} - \mathbf{V^F}$ linkings have the same length $\ell(\mathbf{U}, \mathbf{V^F})$;*
C2. *Every cycle in the digraph covers at least an element of $V_{ess}(\mathbf{U}, \mathbf{V^F})$.*

Hence, Conditions **C0-C2** are instrumental for the construction of structural flat dynamical systems as it will be seen in Part II.

Example 1.13 *Let us consider the digraphs $\mathcal{G}$ depicted on Figure 1.5.*

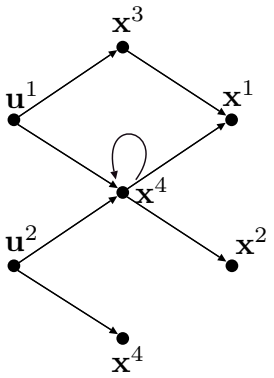


Figure 1.5: Example of digraph of the linear structured system associated to the LPV system (1.7)

Let us show that the set $\{\mathbf{x^1}, \mathbf{x^2}\}$ defines a flat output y_t. To this end, let us consider $\mathbf{U} = \{\mathbf{u}^1, \mathbf{u}^2\}$ and $\mathbf{V^F} = \{\mathbf{x}^1, \mathbf{x}^2\}$. One has card$(\mathbf{U}) = 2$.

C0. *There is only one $\mathbf{U} - \mathbf{V^F}$ maximum linking. It is $\{\mathbf{u^1} \to \mathbf{x^3} \to \mathbf{x^1}, \mathbf{u^2} \to \mathbf{x^4} \to \mathbf{x^2}\}$. The size of the $\mathbf{U} - \mathbf{V^F}$ maximum linking is $\eta(\mathbf{U}, \mathbf{V^F}) = 2$ and Condition **C0** is fulfilled.*

C1. *The length of the $\mathbf{U} - \mathbf{V^F}$ maximum linking is $\ell(\mathbf{U}, \mathbf{V^F}) = 4$. Since the $\mathbf{U} - \mathbf{V^F}$ maximum linking is unique, it is clear that Condition **C1** is fulfilled.*

C2. *The set $V_{ess}(\mathbf{U}, \mathbf{V^F}) = \{\mathbf{u^1}, \mathbf{u^2}, \mathbf{x^1}, \mathbf{x^2}, \mathbf{x^3}, \mathbf{x^4}\}$. There is one cycle of length $\ell(\mathbf{x^4}, \mathbf{x^4}) = 1$. It turns out that this cycle covers an element of $V_{ess}(\mathbf{U}, \mathbf{V^F})$, it is the element $\mathbf{x^4}$. Hence, Condition **C2** is fulfilled.*

1.5 Conclusion

In this chapter, background on LPV dynamical systems have been recalled. Thus a graph-oriented approach, through three conditions given in Theorem 1.3 and Theorem 1.4 have been recalled to characterize with polynomial complexity the flat outputs of a SISO or MIMO LPV system. And based on these conditions, we will see in Part II that it is convenient to construct digraphs that yield flat LPV systems, which the algebraic approach did not allow. This approach will be central for the purpose of designing ciphers as we will see in Chapter 3 and Chapter 4.

Chapter 2

Generalities on Cryptology

The written word is the most important invention in human history but as long as humans have had the ability to share information they have also had the need to conceal that information as well this need led to the invention of cryptography. The word cryptography comes from the greek *"kryptos"* and *"graphein"* meaning hidden writing, is an ancient field of study concerning how to hide secret messages from a malevolent third party.

2.1 Cryptography from ancient to modern

The need for cryptographically secure communication dates back to ancient times. One of the earliest instantiation of cryptography, the hieroglyphs, can be traced back to 1900 BC in Egypt. Although hieroglyphs are not the best example of secret communication they can be considered as the first attempt at mystery surrounding communication. In 486 BC in Greece, the Spartans used *scytales* to send sensitive messages during times of battle. Messages were written on a strip of leather wound around a wooden rods of a particular size, making them undecipherable until they were wrapped around a similar-sized wooden rods by the recipients.
An other cryptographic method was invented by the Romans. Emperor Julius Caesar (100 BC - 44 BC) used cryptographic messages to communicate with his generals. The *Caesar cipher* was one of the earliest advanced cryptographic systems. In this system, each letter in the message is replaced with another one, which is certain amount of positions down the alphabet. This certain amount was the *key* that was needed to decipher each message.
As Caesar's enemies retrieve the message without having the key, they would think it was written in some incomprehensible foreign language. Caesar's cipher is a type of *monoalphabetic substitution cipher*. Let's have the word CAT and then walk along in the alphabet, three letters from every existing letter. So the first letter C will be encrypted to F and A to D and so on. So that is what the Caesar cipher is. Now, what that's doing is actually adding the key three to every letter. There are only 26 letters in the alphabet and therefore only 26 possible keys. That is why the Caesar cipher is considered among the simplest encryption systems as it relatively easy to break. Indeed, around 800 AD, mathematician Al-Kindi wrote the first treatise on cryptanalysis and introduced the frequency analysis technique, determining the most occurring letter in Arabic, as well as which letters could not occur together. In monoalphabetic substitution ciphers, such properties of the natural language are preserved in the ciphertext, allowing an attacker to determine the appropriate shift and recover the original message. This technique was rendered ineffective after the invention of *polyalphabetic substitution ciphers*. One of the earliest polyalphabetic ciphers was the *Vigenère cipher*, developed in the 16th century. This cipher used a shifting substitution determined by a keyword repeated multiple times, spanning the entire message.

Still, in these early times, cryptography was used by governments as a powerful tool for achieving diplomatic and military goals to ensure the security of the state. But the same technology could also be a threat when in the hands of the enemies of the state. One example of this was the conflict between

Elizabeth, Queen of England, and Mary, Queen of Scots, in the mid 16th century, Mary was under arrest in England and was plotting to overthrow and assassinate Elizabeth. And they were transmitting encrypted messages. Mary's cipher was a nomenclature in which both letters and words were replaced by special symbols. Eventually, Elizabeth's spies managed to intercept Mary's correspondence and decipher its content, thus discovering the key. The plot was uncovered and queen Mary sentenced to death.
New breakthroughs in cryptographic techniques made them a key tool in modern warfare. During the second world war Germany deployed the Enigma machine, the most sophisticated encryption device at the time. It consisted of a number of rotors which could encrypt messages by scrambling the 26 letters of the alphabet. The Enigma machine was considered to be unbreakable until British mathematician, Alan Turing, found a weakness in its implementation. Turing invented the bomb, a decryption machine capable of cracking Enigma. So this allowed the allies to win the battle of the Atlantic, which allowed America to supply the U.K. with all of the necessary equipment, send all the troops over and then we have the D-day.
Caesar used cryptography to protect the roman empire. Queen Mary used it in an attempt to overthrow the English crown. The key role played by cryptography during second world war once more underscored the importance of this technology in matters of national security.
It is clear that a strong theoretical background describing the requirements of a secure cryptographic system is needed. In 1949, Claude Shannon who is as the father of modern cryptography authored a groundbreaking paper [Sha49] that established the mathematical basis of information theory and formalized the main goals of modern cryptography. A solid theory has been developed for addressing what it means for a cipher to be secure. Notably, this paper introduced the concept of perfect secrecy (also referred to as unconditional security). Perfect secrecy is the notion that given an encrypted message or *ciphertext* from a perfectly secure encryption scheme or cipher system absolutely nothing will be revealed about the message or *plaintext* by the ciphertext. Shannon gave the first mathematical proof for how and why the *One-Time Pad* is *perfectly secret*. One-Time Pad — also known as the Vernam cipher — is an encryption algorithm invented in the 20th century that simply consists in Xoring every plaintext with a single-use key of the same length. The One-Time Pad provides perfect secrecy if the keys used are fully random, as proven by Shannon, however, this cipher is impractical due to the key management problem.
During the cold war, new advancements in this field were jealously kept secret by the competing superpowers: USSR and US. This led the US to include cryptographic technologies in the United States munitions list, thus classifying it as a weapon.
Cryptography would not have been so groundbreaking if it was not for the advent of computers in the Internet. These technologies enabled public encrypted communication on a global scale. This gave rise to modern cryptographic algorithms like DES [Sta77] in 1977, RSA [RSA78] in 1978, AES [DR99] in 2001, Keccak [Ber+09] and Ascon [Dob+16].

2.2 General principles of cryptography

The fundamental purpose of cryptography is to hide information communicated between two entities, traditionally called Alice and Bob, over an insecure channel from the eyes of *attackers*. In addition, cryptography is used to provide the following security services:

- *Confidentiality.* Guarantees that the content of the information is kept secret, even if the communication is intercepted, and it should only be accessible to authorized persons, Alice and Bob here.
- *Integrity.* Guarantees that the content of the information has not be altered or erased, malevolently or by mistake, during the transmission by an unauthorized party.
- *Authenticity.* Guarantees the identity of the sender of the information. If Bob receives a message from Alice, he must be able to confirm that Alice was indeed the real sender. Moreover, it must be impossible for attackers to impersonate Alice, making it appear for Bob as if a normal exchange of information is underway.

- *Non-Repudiation.* Guarantees that no party, Alice or Bob, can deny that it sent or received an information.

If Alice wants to send a message to Bob over an insecure channel eavesdropped by an attacker, the following mechanism must be used (see Figure 2.1):

- Alice encrypt the message m using her key K_A and the encryption function e:
$$c = e_{K_A}(m)$$
- Alice sends the ciphertext c to Bob;
- Bob receives and decrypts c using his key K_B and the decryption function D:
$$m = d_{K_B}(c)$$

The attackers, snoop on the channel of transmission and intercept the ciphertext c which is unintelligible without the key.

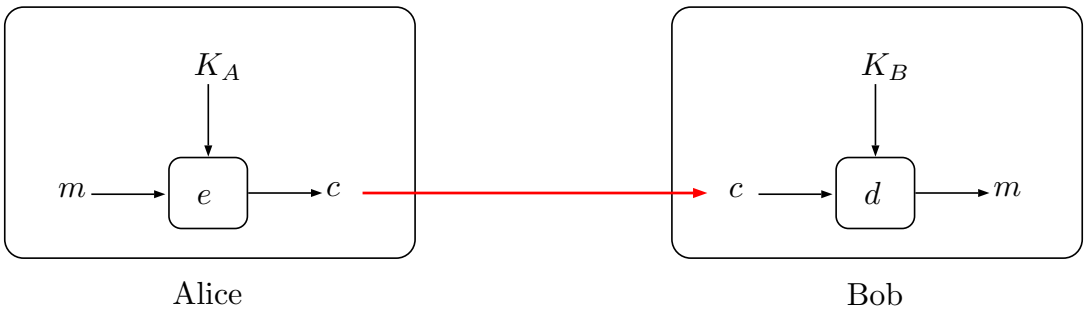


Figure 2.1: Concept of encryption: Alice send a message to Bob.

Cryptography can be divided into two main areas: *secret-key* (or symmetric) cryptography and *public-key* (also called asymmetric) cryptography.

2.2.1 Symmetric encryption

Alice and Bob share a *unique* key for both encryption and decryption: $K_A = K_B$. The biggest problem with symmetric encryption is that the key have to be kept secret. For this end, Bob need to have a way to get the key of Alice with whom he shares the message. Symmetric encryption is particularly useful when rapidity is a bottleneck and when resources are constraints because symmetric key cryptography is faster and smaller in implementations than public key systems.

2.2.2 Asymmetric encryption

Alice and Bob use *two distinct* keys to encrypt and decrypt: $K_A \neq K_B$. In Figure 2.1, Bob sets up in advance K_A (called public key) and K_B (called secret key). Alice use the public key K_A, distributed to everyone, for encrypting message. The Private Key K_B, as implied in the name, is intended to be secret (never needs to be shared) so that only Bob can decrypt the message. Asymmetric algorithms are based on hard problems such as the factorization of large Integers (for RSA [RSA78]) or the discrete logarithm computation in a finite group (for Diffie–Hellman [WM76]). The major drawback of these ciphers is their slowness and cost.
In practice, asymmetric cryptography is often used to exchange the secret key of a symmetric key system to encrypt data. Asymmetric encryption is used in key exchange, email security, Web security, and other encryption systems that require key exchange over the public network.

This thesis focuses on the so-called Self-Synchronizing Stream Ciphers, that will be developed in Section 2.3.2.2. The next section gives a general description on symmetric cryptography.

2.3 Generalities on symmetric cryptography

In symmetric cryptography, two families of ciphers are used: block ciphers and stream ciphers. Block ciphers specify sizes of blocks to encrypt with a specific secret key. While the data is encrypted, the system keeps it in memory waiting for the blocks to be complete through a mode. For stream ciphers, on the other hand, data is encrypted as it streams instead of being held in system memory. Block ciphers and stream ciphers provide confidentiality.

2.3.1 Block ciphers

As implied in the name, a block cipher divides each message into fixed-size blocks of n bits (usually 64/128/256 bits). The encryption is then a permutation of the block by using the key as can be seen in Definition 2.1.
We will denote by K the key of k bits, m the *plaintext* of size n bits and c the *ciphertext* of n bits.

Definition 2.1 *(Block cipher). A block cipher is a function e:*

$$\begin{array}{rccc} e_K & : & \mathbb{F}_2^n \times \mathbb{F}_2^k & \rightarrow & \mathbb{F}_2^n \\ & & (m, K) & \mapsto & c \end{array}$$

such that for any secret key $K \in \mathbb{F}_2^k$, the function e is a permutation.

For each $K \in \mathbb{F}_2^k$, the function e is required to be an invertible mapping on $\mathbb{F}_2^n$. The inverse for e is defined as a function:

$$\begin{array}{rccc} d_K & : & \mathbb{F}_2^n \times \mathbb{F}_2^k & \rightarrow & \mathbb{F}_2^n \\ & & (c, K) & \mapsto & m \end{array}$$

such that $d_K(e_K(m)) = m$, for all $K \in \mathbb{F}_2^k$.

2.3.1.1 Modes of operation

A block cipher processes the data blocks of length n. Usually, the size of a message is larger than the block size. Hence, the long message is first split into a series of sequential n-bit blocks. In the cases where the length of the message is not a multiple of the length of a block, this is called padding, the last block will be completed according to a specific rule until the block length n is reached. Once this is done, there are several methods or modes for acting on this set of blocks.

ECB (Electronic Code Book). This mode is the most simple way of processing a series of sequentially listed message blocks. The first block of plaintext encrypts it with the key to produce the first block of ciphertext. The second block of plaintext follows the same process with the same key and so on so forth. That means that each block is encrypted independently at a time. Encryption in the ECB mode of operation is depicted in Figure 2.2.

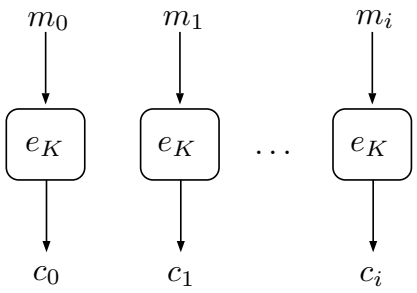


Figure 2.2: Encryption in the ECB mode of operation.

The drawback of this method is that two identical message blocks will give identical ciphertext block, which could allow an attacker to reveal information on the inner structure of the original message.

CBC (Cipher Chaining Block). The CBC mode avoids the problem evoked in ECB mode since each CBC mode of operation provides plaintext dependence for generating ciphertext and makes the system non-deterministic. More precisely, each block of plaintext is XORed immediately with the preceding ciphertext block before applying the block cipher. For the first block, the initialization vector acts as the preceding ciphertext block, see Figure 2.3.

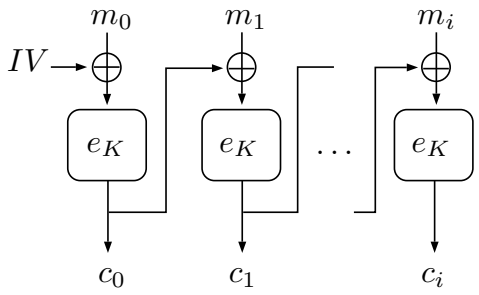


Figure 2.3: Encryption in the CBC mode of operation.

CFB (Cipher Feedback). The CFB mode uses an initial chaining vector (IV) in its processing. CFB mode performs cipher feedback encryption. The initial vector (IV) is encrypted and the encryption result is used as keystream to XOR with the plaintext. The resulting ciphertext is then fed back to the state.

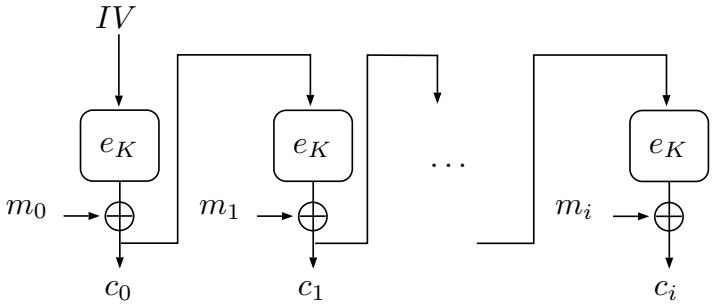


Figure 2.4: Encryption in the CFB mode of operation.

There are other modes such as OFB (Output Feedback) and Counter (CTR) mode. Some of which like can be seen as modes allowing to use a block cipher to design a stream cipher (see section 2.4.1).

If we refer to the definition given in 2.1, constructing directly a block cipher is hard. Indeed, an ideal block cipher would allow every possible permutation of the plaintext values, with each permutation being selected by an encryption key K. Furthermore, each permutation e, for $K \in \mathbb{F}_2^k$, should be complex enough for it to ensure a certain cryptographic security. This task is not easy, since for an n-bit block, oftentimes $n = k = 128$, there are $2^n!$ such permutations in total. The key specifies which mapping to use, then the key space should be as big as the number of possibilities. This size is absolutely not achieved since $2^k < 2^n!$. For this reason, modern constructions mostly adopt an iterative process, see Figure 2.5.

2.3.1.2 Iterated block ciphers

Most block ciphers are designed by repeatedly applying a simpler key-dependent transformation called the *round function* that is applied n_r times in an iterative fashion to compute the output.

Definition 2.2 *(Iterated block cipher). An iterated block cipher is a block cipher obtained by iterating n_r times an invertible permutation f called the round function, each time with a round key K_i derived from the master key K by means of a key scheduling algorithm. Denoting the n-bit plaintext with m, the encryption operation can be written as:*

$$e_K(m) = f_{K_{n_r-1}} \circ f_{K_{n_r-2}} \circ \cdots \circ f_{K_0}(m) = c$$

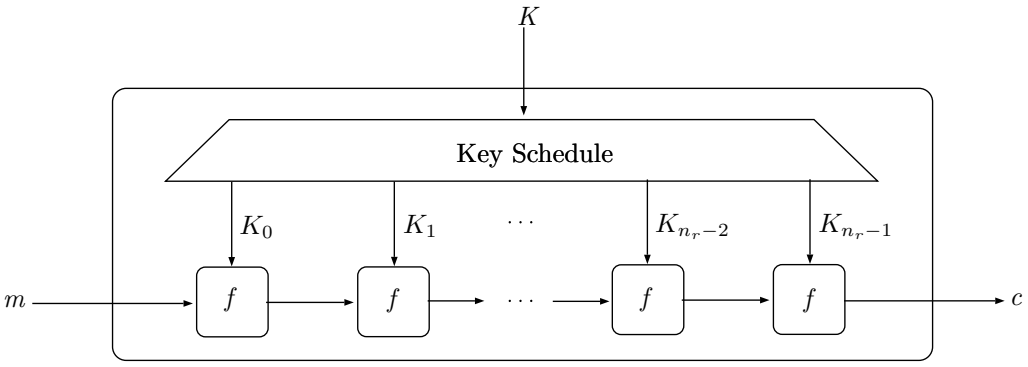


Figure 2.5: An iterative construction of a block cipher: the round function f is repeatedly applied, each time using a different round-key K_i derived from the master key K.

In such construction, the master key K is used to first generate a sequence K_i of round keys that will be involved in the associated round function. The algorithm which allows to obtain the keys K_i is called *Key Schedule*. While the individual round function might be cryptographically weak, its repeated application gives a the construction of a complex encryption function and achieve the required security level.

Iterated block ciphers may be divided into two large categories: Feistel networks and Substitution-Permutation Networks (SPN).

Feistel networks. A Feistel cipher is a symmetric structure used in the construction of block ciphers, named after the physicist and cryptographer Horst Feistel introduced them in 1974. Feistel network was first seen commercially in IBM's Lucifer cipher [Fei74].
An example of a Feistel cipher is the DES[1][Sta77] which was adopted by the U.S. government as an official standard from 1977 to 2000.

Let f be the *Feistel function* and let $K_0, K_1, \ldots, K_{n_r}$ be subkeys for the rounds $\{0, 1, \ldots, n_r\}$ respectively. The round function of a Feistel network proceeds by mapping the plaintext block of n-bits, split into two equal halves (L_i, R_i), to an output (L_{i+1}, R_{i+1}). For each round $i = 0, 1, \ldots, n_r$, the Feistel scheme works as follows:

$$\begin{array}{rccl} e_{K_i} & : & \mathbb{F}_2^{\frac{n}{2}} \times \mathbb{F}_2^{\frac{n}{2}} & \rightarrow & \mathbb{F}_2^{\frac{n}{2}} \times \mathbb{F}_2^{\frac{n}{2}} \\ & & (L_i, R_i) & \mapsto & (L_{i+1}, R_{i+1}) = (R_i, L_i \oplus f(R_i, K_i)) \end{array} \tag{2.1}$$

The right part has not been transformed (just sent to the left). The Feistel structure has the advantage that the Feistel function used for encryption and decryption are the same up to the final permutation and a reversal of the subkeys. Then the ciphertext is (R_{n+1}, L_{n+1}). Decryption of a ciphertext (R_{n+1}, L_{n+1}) is accomplished by deriving for $i = n_r, n_r - 1, \ldots, 0$:

$$\begin{array}{rccl} d_{K_i} & : & \mathbb{F}_2^{\frac{n}{2}} \times \mathbb{F}_2^{\frac{n}{2}} & \rightarrow & \mathbb{F}_2^{\frac{n}{2}} \times \mathbb{F}_2^{\frac{n}{2}} \\ & & (L_{i+1}, R_{i+1}) & \mapsto & (R_i, R_{i+1} \oplus f(L_{i+1}, K_i)) \end{array} \tag{2.2}$$

[1]Data Encryption Standard

Then (L_0, R_0) is the plaintext again. As a result, the iterative nature of the Feistel construction makes implementing the decryption on top of encryption very cost-efficient (particularly on the hardware available at the time of DES' design). A Feistel round is depicted in Figure 2.6.

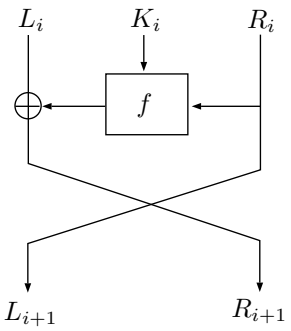


Figure 2.6: A Feistel round.

Many modern and also some old symmetric block ciphers based on Feistel networks in the literature include LBlock [WZ11], Simon [Bea+15] and Lilliput [Ber+15].
The structure and properties of Feistel ciphers have been extensively explored by cryptographers. Several theoretical works has generalized the construction somewhat, and gives more precise bounds for security such as unbalanced Feistel networks [SK96], where the two branches are of different sizes. In the case of Misty-like structures—named after the Misty block cipher [Mat97]—the Feistel function has to be a permutation and is directly applied to a branch rather than a copy of it. Finally, *Generalized Feistel Networks* extend the Feistel network to a larger number of branches.
Substitution-Permutation networks. A Substitution-Permutation network or an SP network is an iterative construction of block cipher. It forms the basis of the famous AES[2] algorithm proposed by Daeman and Rijmen [DR99] and consists in following accurately the two design criteria stated by Claude Shannon [Sha49], better known as Shannon's confusion and diffusion properties. These two properties were designed to provide resistance against statistical analysis, and can be described as follows:

- Confusion refers to making the relationship between the key and the ciphertext as complex and involved as possible;
- Diffusion refers to the property that the redundancy in the statistics of the plaintext is "dissipated" in the statistics of the ciphertext.

As shown in Figure 2.7, a Substitution-Permutation Network round is composed (in addition to the usual key addition) of two main operations: application of small nonlinear functions—usually operating on 4 or 8 bits—called *S-boxes*[3]providing a confusion, and application of linear mappings L_m to reach better diffusion. The SP networks structure do not guarantee invertibility, opposed to Feistel ciphers, it requires that all inner components be invertible.
Note that the substitution and permutation boxes are not kept hidden in SP networks. Only the round keys are kept secret for security.
In addition to the Advanced Encryption Standard [DR99] (the current NIST[4] standard), there are also examples of SPN algorithms such as Klein [GNL11], Midori [Ban+15], Noekeon [Dae+00], Present [Bog+07], Robin [Gro+14], Skinny [Bei+16] and Zorro [Gér+13].

2.3.2 Stream ciphers

Stream ciphers are based on the so-called Vernam cipher where the plaintext (a binary string of some length) is XORed with a sequence of bits of the same length called the keystream, in order to produce the

[2]Advanced Encryption Standard.
[3]Substitution Box.
[4]National Institute of Standards and Technology, American standardization authority.

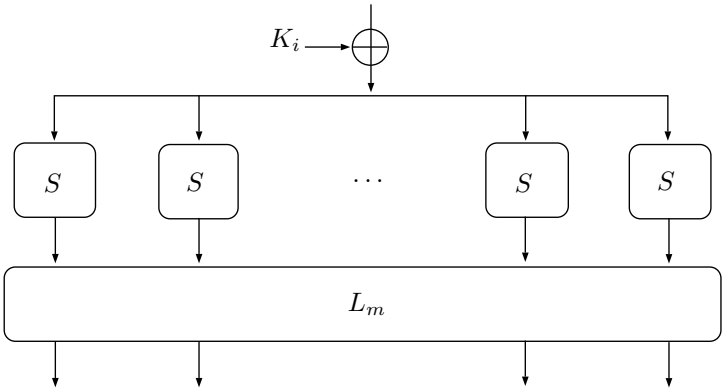


Figure 2.7: One SPN round.

ciphertext. The Vernam cipher is also known as the one-time-pad (1917) because it uses a new random secret key for every encryption.
For a stream cipher, it must be given an alphabet $\mathcal{A}$, that is, a finite set of basic elements named symbols. The set $\mathcal{A}$ stands in this paragraph as a general notation without any specific alphabet. Typically, $\mathcal{A}$ could be composed of 1 or several bits elements. Hereafter, the index $t \in \mathbb{N}$ will stand for the discrete-time. On the ciphering part, the plaintext (also called information or message) $m \in \mathcal{M}$ ($\mathcal{M}$ is the message space) is a string of plaintext symbols $m_t \in \mathcal{A}$.
Each plaintext symbol is encrypted, by means of an encryption (or ciphering) function e, according to:

$$c_{t+b} = e(z_{t+b}, m_t), \tag{2.3}$$

where $z_t \in \mathcal{A}$ is the so-called keystream (or running key) symbol delivered by a keystream generator. The function e is invertible for any prescribed z_{t+b}. The resulting symbol $c_{t+b} \in \mathcal{A}$ is the ciphertext symbol. The integer $b \geq 0$ stands for a delay between the plaintext m_t and the corresponding ciphertext c_{t+b} symbol that is sometimes introduced for computational or implementation reasons. Consequently, for stream ciphers, the way how to encrypt each plaintext symbol changes at each iteration and stands as an asset for this class of ciphers. The resulting ciphertext $c \in \mathcal{C}$ ($\mathcal{C}$ is called the ciphertext space) is the string of symbols $c_t \in \mathcal{A}$.

At the deciphering side, the ciphertext c_t is processed through a decryption function d which depends on a running key $\widehat{z}_t \in \mathcal{A}$ delivered, similarly to the ciphering part, by a keystream generator.
The decryption function d obeys the following rule. For any two keystream symbols $\widehat{z}_{t+b}, z_{t+b} \in \mathcal{A}$, it holds that

$$\widehat{m}_{t+b} := d(c_{t+b}, \widehat{z}_{t+b}) = m_t \text{ whenever } \widehat{z}_{t+b} = z_{t+b}. \tag{2.4}$$

where d must be introduced for causality sake. Equation (2.4) means that the running keys z_t and $\widehat{z}_t$ must be synchronized to guarantee correct decryption. The generators delivering the keystreams are parameterized by a secret key denoted by $K \in \mathcal{K}$ ($\mathcal{K}$ is the secret key space). The distinct classes of stream ciphers (synchronous or self-synchronizing) differ each other by the way on how the keystreams are generated and synchronized.

2.3.2.1 Synchronous Stream Ciphers

The following equations describe the synchronous stream ciphers:

$$\begin{cases} x_t &= g_K(x_{t-1}) \\ z_t &= h_K(x_t) \\ c_t &= e(z_t, m_t) \end{cases} \tag{2.5}$$

where g_K is the *next-state transition function*. The keystream z_t is obtained by applying a *filtering function* h_K to the internal state. The value x_t is the internal state whose initial state is obtained not only from the key but also from a public initialization vector IV. This nonce is generated by applying a key scheduling process or a Linear Feedback Shift Register. A synchronous stream ciphers is depicted in Figure 2.8.

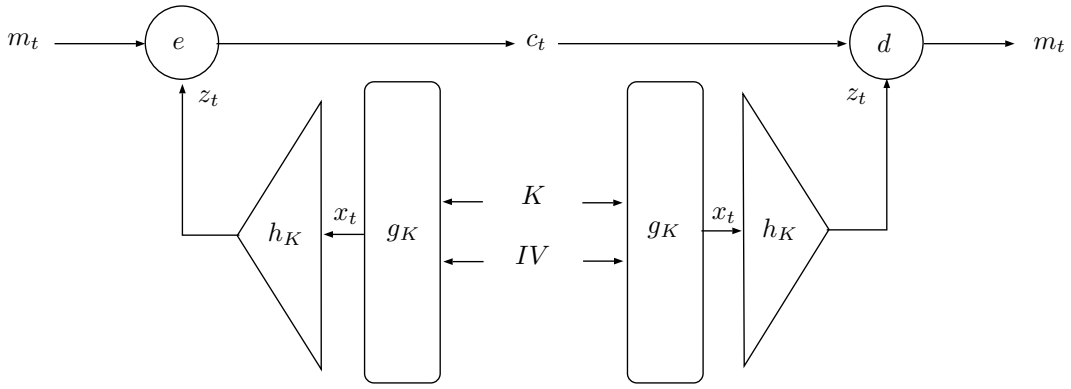


Figure 2.8: Synchronous stream encryption.

There are examples of synchronous stream ciphers algorithms such as RC4, SOSEMANUK [Ber+08], GRAIN [HJM07], TRIVIUM [Can06].

Next, we detail the special class of stream ciphers called Self-Synchronizing Stream Ciphers and denoted for brevity SSSC.

2.3.2.2 Self-Synchronizing Stream Ciphers (SSSC)

The following equations (assuming the delay $b = 0$) describe the self-synchronous stream ciphers:

$$\begin{cases} z_t &= \alpha_K(c_{t-l-M+1}, \ldots, c_{t-l}) \\ c_t &= e(z_t, m_t) \end{cases} \tag{2.6}$$

where K is the secret key shared between the ciphering and deciphering side. The keystream z_t is obtained by applying a *filtering function* α_K. l is a non-zero positive integer standing for a possible delay. The keystream generator α_K depends on past values of c_t whose the number is most often bounded and equals M, the delay of memorization. Equation (2.6) is also known as the canonical form of the SSSC and is depicted in Figure 2.9.

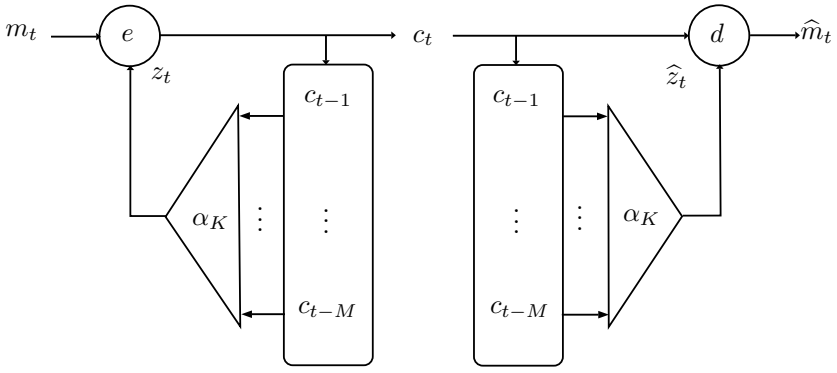


Figure 2.9: Canonical form of SSSC for $l = 1$.

To decrypt a ciphertext c_t 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:

$$\begin{cases} \hat{z}_t & = \alpha_K(c_{t-l-M+1}, \ldots, c_{t-l}) \\ \hat{m}_t & = d(\hat{z}_t, c_t) \end{cases} \tag{2.7}$$

Properties of SSSC. As far as SSSC are concerned, the ability to self-synchronizing provides many advantages:

- 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; and so on;
- 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;
- it had been shown in [DGM17] 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;
- 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.

In the next section, we detail the different architecture related to Self-Synchronizing Stream Ciphers.

2.4 Design of Self-Synchronizing Stream Ciphers

Actually, the model (2.6) of an SSSC is a conceptual model, called canonical representation, that can correspond to different architectures and that results from different design approaches. In the open literature, few designs methods have been proposed. They are detailed below.

2.4.1 Block ciphers in CFB mode

This SSSC design approach resorts to a length M shift register and a block cipher (DES for instance) both inserted in a closed-loop architecture. It is a very special mode of operation involving block ciphers naturally called Cipher FeedBack (CFB) mode. The block cipher's input is the shift register state. Usually a limited number of the block cipher output bits are retained, the selection being performed through a filtering function denoted h on the Figure 2.10. Such a configuration is often used in 1-bit CFB mode. In such a case, the encryption function e is a XOR (modulo 2 addition over $\{0, 1\}$). The keystream generator h of the corresponding canonical form (2.6) results from the composition of three functions: the state transition function of the shift register, the block cipher and the filter function h. This mode is quite inefficient in terms of encryption speed since one block cipher operation, and so multiple rounds, are required for enciphering a single plaintext m_t.

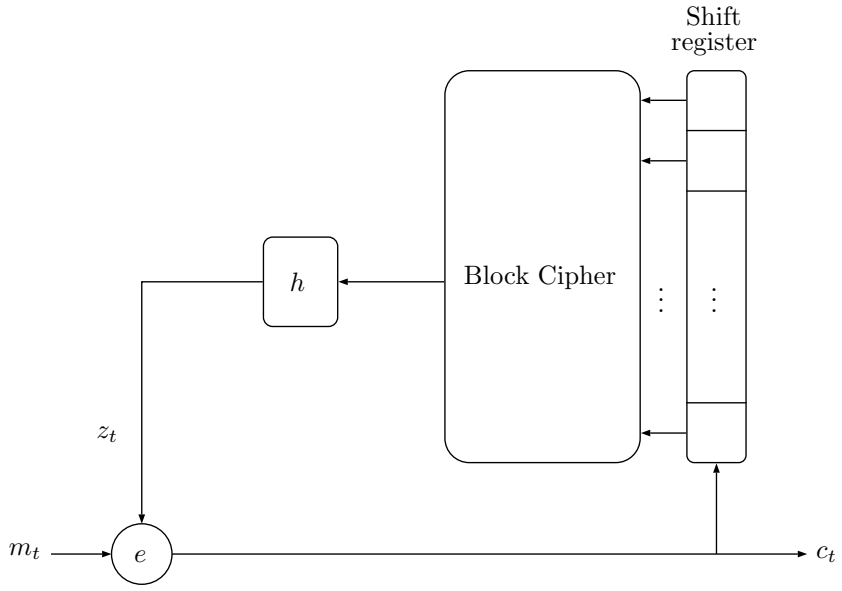


Figure 2.10: SSSC in CFB mode. A shift register is used to gather the previous ciphertexts. The output of the block cipher is filtered with a function h to produce a 1-bit keystream z_t that is XORed to the plaintext m_t.

2.4.2 Maurer's approach

In [Mau91], it is suggested an alternate design approach exclusively dedicated to SSSC. It includes two main ideas.

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

$$\begin{cases} x_{t+1} &= g_K(x_t, c_t) \\ z_t &= h_K(x_t) \end{cases} \tag{2.8}$$

The function g_K is next state transition function while h_K is called output function.
The function g_K is called T-functions [KS04] if each variable depends only on previous variables. Hence, the function g_K is a T-function if for each state component i, $1 \leq i < n$ of the output can depend only on component $0, 1, \ldots, i-1$ of the input.

$$\begin{aligned} x^0_{t+1} &= g_K(c_t) \\ x^1_{t+1} &= g_K(x^0_t, c_t) \\ x^2_{t+1} &= g_K(x^0_t, x^1_t, c_t) \\ &\vdots \\ x^{n-1}_{t+1} &= g_K(x^0_t, x^1_t, \ldots, x^{n-2}_t, c_t) \end{aligned} \tag{2.9}$$

The automaton must have a finite input memory of size M meaning that the state x_t must be expressed by mean of a function l_K which depends on a finite number of past ciphertexts c_{t-i} :

$$x_t = l_K(c_{t-M}, \ldots, c_{t-1}) \tag{2.10}$$

Substituting the above expression of x_t into the second equation of (2.8) gives the function α_K of the canonical form (2.6). One has the following composition : $\alpha_K = h_K \circ l_K$.
Let us notice that the CFB mode can be rewritten into the form (2.8)-(2.10). The function l_K is very simple since it merely reduces to a shift. The output function h_K results from the composition of the

block cipher (parameterized by its secret key K) and the filter function h.

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

The second idea of the Maurer's principle consists in increasing the complexity by combining several finite automata in serial or in parallel or more generally by performing composition. As a result, many components that are relatively simple in terms of implementation complexity and memory size can be combined to form an SSSC realizing a very complicated function g_K in the corresponding canonical representation (2.6). For a serial composition of multiple automata, the resulting memory size equals the sum of the memory size of each automaton. For a parallel composition of multiple automata, the resulting memory size equals the upper memory size. When implemented in hardware, parallelization leads to very high achievable encryption speed. An example of architecture involving four automata is depicted in Figure 2.11.

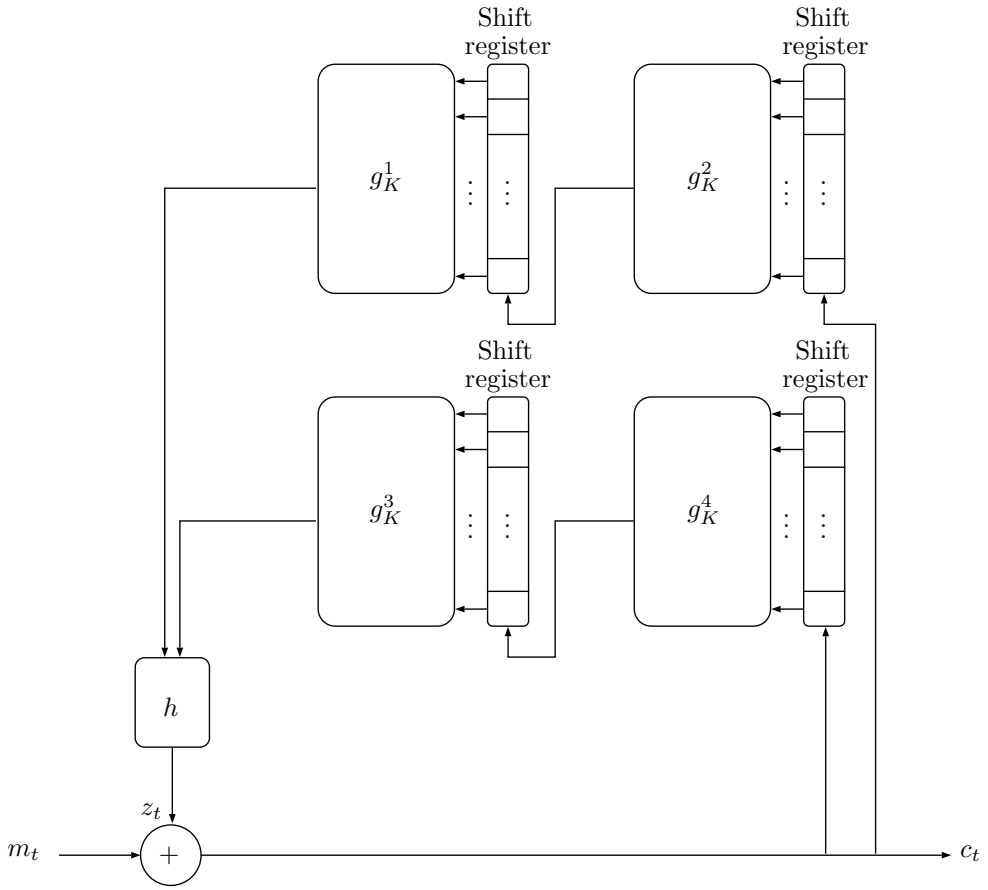


Figure 2.11: Serial and parallel automata in the architecture proposed by Maurer.

Throughout the eSTREAM[5] project, two fully specified algorithms have retained attention: SSS [DGV04] and Moustique [DP06]. They are shortly described to illustrate how the general principle of Maurer is taken into account. As a matter of fact, only the first idea of Maurer consisting in resorting to an automaton with finite input memory has been adopted throughout these two examples. Indeed, as it

[5] https://www.ecrypt.eu.org/stream/

turns out, the second idea is too general as is. These examples are also interesting in that they give us a better understanding in the way how the dynamical systems are "shaped" to guarantee the self-synchronization property.

2.5 General Principles of Classical Cryptanalysis

Cryptanalysis is the science that aims at finding weaknesses in cryptographic algorithms and using them to gain access to the contents of encrypted messages without knowing the secret key or, at best, to recover the secret key. It helps us to better understand the cryptographic algorithms and also helps cryptographer to improve the cryptographic schemes by finding any weak point and thus work on the algorithm to create a more secure cryptographic algorithm. In symmetric cryptography, and when the studied primitive is an encryption, its objective will most often be to find the key thanks to a less expensive procedure than the exhaustive research. Other results such as the highlighting of a statistical bias in the cipher suite produced by a stream cipher will also be considered as cryptanalysis, insofar as they highlight undesired properties of a cipher.

2.5.1 Attack Models

The impact of a cryptanalysis can be defined depending on data required for an adversary to perform the attack. We can distinguish several categories of attacks, sort from least constrained to most constrained.

- *Ciphertext-only attack.* In this method, the attacker is assumed to have access to a set of ciphertexts. He does not have access to corresponding plaintexts. For such an attack to succeed, we need to have enough knowledge about the plaintext distribution. A cipher which is vulnerable to this attack is very weak.
- *Known-plaintext attack.* The attacker has access to some plaintext/ciphertext pairs encrypted under the same key.
- *Chosen-plaintext attack.* In this method, the attacker has the ability to encrypt plaintexts of his choice. So he has the ciphertext-plaintext pair of his choice.
- *Chosen-ciphertext attack.* The attacker has access to a decryption oracle and can select different ciphertexts to be decrypted.
- *Related-key attack.* In this attack model, the attacker is assumed to have an access to plaintext and its different encryptions under different unknown related keys that have a certain known or chosen relation with the target key to be recovered.

2.5.2 The complexity of an attack

To assess the efficiency of an attack, it is required to evaluate the minimum number of operations that would be needed to break the cipher. For this purpose, we use the following three complexities:

- *Data complexity.* Means the number of data (ciphertexts, known-plaintext, chosen-plaintext etc.) required to perform successfully an attack. Since these data must be obtained from the key keeper, this has an immediate influence on the communication complexity.
- *Time complexity* of attack. It corresponding to the amount of computational processing required to perform this attack successfully. It can be expressed such that one can compare the attack to an exhaustive[6] key search. The computational unit is often expressed either in elementary operations, or in number of encryptions.
- *Memory complexity.* It measures the amount of memory necessary to store the computed data to perform the attack.

[6]Exhaustive search (or brute force) is the method of trying all possible elements in the search space until the correct one is found.

2.5.3 Cryptanalysis of stream ciphers

In this section we give a set of attacks of stream ciphers that received much attention in the literature. This attacks should be considered during the design of the stream ciphers.

2.5.3.1 Algebraic attacks

Algebraic attacks on stream ciphers based on linear feedback shift registers (LFSR's) have been introduced by N. Courtois and W. Meier in [CM03]. The main idea of algebraic attacks is to recover the secret key by generating and solving nonlinear equations involving plaintext, ciphertext and key bits. This idea dates back to the seminal work of Shannon. To foil algebraic attacks, designers try to increase the non-linearity of their system and improve the linear diffusion throughout the cipher.
The keystream bit z_t can be expressed in the form of a system of N equations involving the initial state x_0, the next-transition function g_K and the filtering function h_K:

$$\begin{cases} z_0 = h_K(x_0) \\ z_1 = h_K \circ g_K(x_0) \\ \quad \vdots \\ z_N = h_K \circ g_K^N(x_0) \end{cases} \tag{2.11}$$

The knowledge of any N keystream bits lead to an algebraic system of N equations if the g_K function is linear. The degree of these equations correspond to the algebraic degree of the filtering function. It is possible to solve the obtained system using a relinearization technique (see [CW82]) or a dedicated algorithm using the Gröbner basis.
An improvement of this attack was proposed in [Cou03] then in [Arm04]. The attack described above improved in [Cou03] then in [Arm04] under the name of fast algebraic attack. It enables to cryptanalyze some systems whose transition function is linear even when the filtering function has a high algebraic degree [FA03]. The main idea is to solve equations once there exist relations of low degree degree between the input and the output of the filtering function. Thus, if there exists a function Φ of the small degree such that

$$\forall x \in \mathbb{F}_2^n \quad \Phi(x) \cdot h_K(x) = 0$$

we deduce that

$$\Phi(g_K^t(x_0)) \cdot z_t = 0$$

This means that once we observe a bit z_t equals to 1, the initial state x_0 verifies the equation

$$\Phi(g_K^t(x_0)) = 0$$

which is of degree $deg(\Phi)$, low degree when the transition function is linear. These attacks have given rise to a criterion called the *algebraic immunity* $AI(g)$. The algebraic immunity is the minimum value of d such that g or $g+1$ admits an *annihilating function* of degree d. An annihilator of g is a non-zero Boolean function g such that $\Phi \cdot g = 0$.

Solving a system of multivariate polynomials using a relinearization method i.e. the inversion of this system by a Gaussian elimination, allows to find the initial state in a number of operations of the order of [CW87]

$$\left(\sum_{i=0}^{d} \binom{n}{i} \right)^3 \approx n^{3d}$$

2.5.3.2 Guess-and determine attack

Many stream ciphers do not use their whole internal state to compute a keystream bit. Such cipher design allows the attacker to mount a guess-and determine attack [HR02]. This attack exploits the fact that only a few internal state bits are defined as input to the filter function. Therefore, only these bits determine

the value of the generated keystream bit. To mount such an attack, the attacker only guesses used bits, computes the output and evaluates it against the corresponding keystream bit that was recovered from an eavesdropped trace. The evaluation immediately leads to a contradiction for many of the guessed candidates. After evaluation, the internal state is updated accordingly and only additional required bits are guessed.

2.5.3.3 Time-Memory Trade-Off for Stream Ciphers

Cryptanalysis attacks based on exhaustive key search are the typical context where time-memory trade-offs are applicable. Due to large key sizes, exhaustive key search usually needs unreal computing powers. Time-Memory Trade-Off (TMTO) attacks offer a generic technique to reverse one-way functions.
Assume that an attacker tries to break a known cryptographic function e. On stream ciphers, this amounts to find the internal state used to encrypt a known plaintext m_t. The goal is to reverse the function h_K, i.e. to find an input x_t such that $z_t = h_K(x_t)$. Here x is the internal state of cipher e_K and z_t is the corresponding keystream. So $h_K(x_t) = z_t$ and c_t is obtained by XORing keystream and known plaintext. This section is concerned with finding the internal state of a stream cipher. The attack was originally presented by Hellman in [Hel80].
A typical TMTO attack is composed of two phases: the first is the precomputation phase, or offline phase, while the second is referred to as the real-time, or online phase.
In the offline phase, the attacker precomputes a large table using the function h_K he is trying to break, while in the online phase the attacker captures a sample of keystream and checks if this happens to be in his tables. If this attack is successful the attacker can learn the internal state x_t for which $z_t = h_K(x_t)$. To evaluate these types of attacks different parameters and costs are considered.

- N : the size of the state space.
- T (Attack time): this can be subdivided between the time for the offline an online phases.
- M_c (Memory): memory cost of the attack.

Hellman's attack on stream ciphers then goes as follows. In order to reverse the function h_K, the table above is precomputed in the offline phase, for a single known plaintext.

$$\begin{array}{ccccccccc}
x_0 & \to & h_K(x_0) & \to & h_K(h_K(x_0)) & \to & \dots & \to & h_K^\theta(x_0) \\
x_1 & \to & h_K(x_1) & \to & h_K(h_K(x_1)) & \to & \dots & \to & h_K^\theta(x_1) \\
x_2 & \to & h_K(x_2) & \to & h_K(h_K(x_2)) & \to & \dots & \to & h_K^\theta(x_2) \\
\vdots & \vdots & \vdots & & \vdots & & \ddots & \vdots\vdots & \vdots \\
x_{m_c} & \to & h_K(x_{m_c}) & \to & h_K(h_K(x_{m_c})) & \to & \dots & \to & h_K^\theta(x_{m_c})
\end{array}$$

In order to cover as much of the N points of the search space as possible, an $m_c \times \theta$ matrix is computed, where the m_c rows consist of chains of length θ and where each point in the chain is a new iteration of h_K on the result of the previous point. Finally, only the begin point and end point of each chain are stored (ordered by the endpoints) as the precomputation table.
During the online phase, the attacker obtains keystream samples. He then makes another chain of at most θ iterations of applying the function h_K and for each iteration checks if the result matches one of the endpoints stored in his table. If this happens, he recomputes the chain starting from the corresponding begin point until the preimage of the ciphertext, thereby reversing function g_K in an attack time of order θ in the online phase. Hellman proves that this lower bound can be approximated to 3/4 for tables for which $m_c \cdot \theta^2 = N$.
He argues that increasing m_c and θ beyond $m_c\theta^2 = N$ is ineffective, since the chance of overlap only increases as m_c and θ increase. Therefore Hellman continues his analysis of using $m_c\theta^2$ matrices satisfying $m_c\theta^2 = N$, $M_c = m_c\theta$, $T = \theta^2$ and $TM_c^2 = \theta^2 m_c^2\theta^2 = (m_c\theta^2)^2 = N^2$.

2.5.3.4 Differential attacks

The ideas of Differential cryptanalysis were used in various cryptanalytic attacks on stream ciphers. In [BD07], it was shown a key difference, or even an initial value difference, can be used to predict the stream differences with some probability. Furthermore, the terms *differential characteristics* and *differential* for stream ciphers was defined. For self synchronizing stream ciphers, the possible characteristics in this case are:

- $(\Delta K, \Delta IV) \rightarrow \Delta x_{t+1}$ where a difference in the key or the IV generates a difference in the internal state,
- $(\Delta x_t, \Delta m_t) \rightarrow \Delta x_t$ is the differential that predicts the difference in the internal state given the current difference of the internal state and the difference in the plaintext,
- $(\Delta x_t, \Delta m_t) \rightarrow \Delta c_t$ is the differential that predicts the ciphertext difference given the internal state difference and the plaintext difference.
- $(\Delta x_t, \Delta c_t) \rightarrow (\Delta x_t, \Delta m_t)$ is the differential that predicts the plaintext difference given the internal state difference and the ciphertext difference.

2.5.3.5 Linear cryptanalysis

A linear cryptanalysis technique for stream ciphers was presented by Golić in [Gol94]. It relies on the same basic principles as the linear cryptanalysis for block ciphers introduced by Matsui. The linear cryptanalysis consists in finding some linear functions of the keystream bits which are not uniformly distributed. Such linear correlations are used for distinguishing the keystream sequence from a random sequence by a classical statistical test. Efficient techniques for finding biased linear relations between the keystream bits are presented in [Gol94; CHJ02].

2.5.3.6 Cube Attacks

As established in [DS09], a cipher is vulnerable to cube attacks if an output bit can be represented as a sufficiently low degree polynomial over $GF(2)$ of key and input bits. It works by summing an output bit value for all possible values of a subset of public input bits, chosen such that the resulting sum is a linear combination of secret bits. Repeated application of this technique gives a set of linear relations between secret bits that can be solved to discover these bits.

2.6 Conclusion

In this chapter, we recalled generalities on symmetric cryptography mainly on stream ciphers. The different architecture related to Self-Synchronizing Stream Ciphers was detailed. One of the fundamental purposes of cryptography is security. This is why cryptanalysis devoted to the security evaluation of cryptographic primitives was discussed. The general principles of the known attacks like algebraic attacks, differential attacks and linear cryptanalysis are given. A systematic and general new construction of Self Synchronizing Stream Ciphers based on flat LPV systems will be motivated and addressed in Chapter 3.

Part II

Contribution from an automatic control point of view

Introduction

Being all the necessary control-theoretical background recalled, the second part of this thesis regroups all the works related to the design of Self-Synchronizing Stream Ciphers (written as SSSC for brevity) based on LPV automata. In the introduction of this part, the equations of the Self-Synchronizing Stream Ciphers are presented. The distinction between deterministic and statistical SSSC is made. Then, the motivation on the use of LPV dynamical systems as appropriate candidates for the design of SSSC is given. Finally, we focus on the performances that an SSSC should get; self-synchronization ability, computational complexity amenable for practical use and security. Chapter 3 deals with the design of deterministic SSSCs based on LPV systems and the structural approach. Chapter 4 focuses on the design of statistical SSSCs. The structural approach is dedicated to LPV systems while an alternative based on an algebraic approach is proposed for the special case of switched linear systems. Benefits and shortcomings of deterministic or statistical structures are discussed.

Self Synchronizing Stream Ciphers and LPV automata

Stream Ciphers Overview

For a stream cipher, it must be given an alphabet $\mathcal{A}$, that is, a finite set of basic elements named symbols. The set $\mathcal{A}$ stands in this part as a general notation without any specific alphabet. Typically, $\mathcal{A}$ could be composed of 1 or several bits elements. On the ciphering part, the plaintext (also called information or message) $m \in \mathcal{M}$ ($\mathcal{M}$ is the message space) is a string of plaintext symbols $m_t \in \mathcal{A}$. Each plaintext symbol is encrypted, by means of an encryption (or ciphering) function e, according to:

$$c_{t+r} = e(z_{t+r}, m_t), \tag{2.12}$$

where $z_t \in \mathcal{A}$ is the so-called keystream (or running key) symbol delivered by a keystream generator. The function e is invertible for any prescribed z_{t+r}. The resulting symbol $c_{t+r} \in \mathcal{A}$ is the ciphertext symbol. The integer $r \geq 0$ stands for a delay between the plaintext m_t and the corresponding ciphertext c_{t+r} symbol that is sometimes introduced for computational or implementation reasons. Consequently, for stream ciphers, the way how to encrypt each plaintext symbol changes on each iteration and stands as an asset for this class of ciphers. The resulting ciphertext $c \in \mathcal{C}$ ($\mathcal{C}$ is called the ciphertext space) is the string of symbols c_t.
At the deciphering side, the ciphertext c_t is processed through a decryption function d which depends on a running key $\widehat{z}_t \in \mathcal{A}$ delivered, similarly to the ciphering part, by a keystream generator. The decryption function d obeys the following rule. For any two keystream symbols $\widehat{z}_{t+r}, z_{t+r} \in \mathcal{A}$, it holds that

$$\widehat{m}_{t+r} := d(c_{t+r}, \widehat{z}_{t+r}) = m_t \text{ whenever } \widehat{z}_{t+r} = z_{t+r}. \tag{2.13}$$

where r must be introduced for causality sake. Equation (2.13) means that the running keys z_t and $\widehat{z}_t$ must be synchronized to guarantee correct decryption. The generators delivering the keystreams are parameterized by a secret key denoted by $K \in \mathcal{K}$ ($\mathcal{K}$ is the secret key space). The distinct classes of stream ciphers (synchronous or self-synchronizing) differ each other by the way on how the keystreams are generated and synchronized. Next, we detail the special class of stream ciphers called Self-Synchronizing Stream Ciphers.

Keystream Generators for Self-Synchronizing Stream Ciphers

A keystream generator based on automata in the form of dynamical systems has been first suggested in [Mau91]. It is based on the use of so-called finite state automata with finite memory as described below. This is typically the case in the cipher Moustique [Kas+04]. At the ciphering side, the automaton delivering the keystream takes the form:

$$x_{t+1} = f_K(x_t, m_t), \tag{2.14}$$

where $x_t \in X$ is the internal state, X is the set where the internal state belongs, f is the next-state transition function parameterized by the secret key $K \in \mathcal{K}$. The message symbol denoted m_t acts as an input usually denoted u_t for the dynamical system (2.14). In this context, m_t is more appropriate than u_t since it is not a control input as defined in control theory. The output of the automaton (2.14) is defined by a function h of the state x_t as follows.

$$h : x_t \in X \mapsto h(x_t) \in \mathcal{A} \tag{2.15}$$

Some SSSCs do not use the output $h(x_t)$ as the keystream but r successive operations from x_t are performed at time instants $t, \ldots, t+r$ to deliver, with a delay r, the keystream denoted by z_{k+r}. The keystream is thus defined by means of a function h^r defined as:

$$h^r : x_t \in X \mapsto z_{t+r} = h^r(x_t) \in \mathcal{A} \tag{2.16}$$

When $r = 0$ (no delay), both functions h and h^r coincide one another.
According to (2.13), m_t is a function of z_{t+r} and c_{t+r}. Besides, from (2.16), it holds that z_{t+r} is a function of x_t. Hence, the automaton described by Equation (2.14) can be equivalently rewritten as

$$\begin{cases} x_{t+1} = g_K(x_t, c_{t+r}) \\ z_{t+r} = h^r(x_t) \end{cases} \tag{2.17}$$

By iterating (2.17) a finite number of times, if there exists a function ℓ_K and two finite integers ℓ and ℓ' (with $\ell, \ell' \in \mathbb{Z}$ and $|\ell'| \geq |\ell|$, $|\cdot|$ stands for the absolute value) such that

$$x_t = \ell_K(c_{t-\ell}, \ldots, c_{t-\ell'}). \tag{2.18}$$

then,

$$z_{t+r} = h^r(\ell_K(c_{t-\ell}, \ldots, c_{t-\ell'})). \tag{2.19}$$

Otherwise stated, if after a finite number of iterations, the current state of the automaton (2.17) does no longer depend on the initial state, the current state will only depend on shifted cryptograms. Such an automaton is called a finite memory automaton according to [Mau91].
Actually, the fact that the keystream symbol can be written in the general form

$$z_{t+r} \quad = \quad \alpha_K(c_{t-\ell}, \ldots, c_{t-\ell'}), \tag{2.20}$$

with α_K a function involving a finite number of shifted ciphertexts from time $t-\ell$ to $t-\ell'$ ($\ell, \ell' \in \mathbb{Z}$), is a common feature of the SSSC. Equation (2.20) is called the canonical equation. The integer $M = |\ell'| - |\ell| + 1$ is called the delay of memorization.
At the deciphering side, the automaton takes the form

$$\begin{cases} \widehat{x}_{t+r+1} = g_K(\widehat{x}_{t+r}, c_{t+r}), \\ \widehat{z}_{t+r} = h^r(\widehat{x}_t) \end{cases} \tag{2.21}$$

where $\widehat{x}_t$ is the internal state. It is worthwhile stressing that such a state space representation is nothing but the left inverter of (2.14) in control theory. Similarly to the ciphering part, the automaton having a finite memory since it gets the same dynamics than the cipher, it means that, iterating Equation (2.21) a finite number of times also yields

$$\hat{x}_{t+r} = \ell_K(c_{t-\ell}, \ldots, c_{t-\ell'}),$$

and thus,

$$\hat{z}_{t+r} = h^r(c_{t-\ell}, \ldots, c_{t-\ell'}).$$

Hence, it is clear that after a transient time of maximal length equal to M, it holds that, for $t \geq M$,

$$\widehat{x}_{t+r} = x_t \text{ and } \widehat{z}_{t+r} = z_{t+r}. \tag{2.22}$$

In other words, the keystream generators synchronize automatically after at most M iterations. Hence, the decryption is automatically and properly achieved after at most M iterations too. No specific synchronizing protocol between the cipher and the decipher is needed. This explains the terminology Self-Synchronizing Stream Ciphers and it is one of its practical interest.

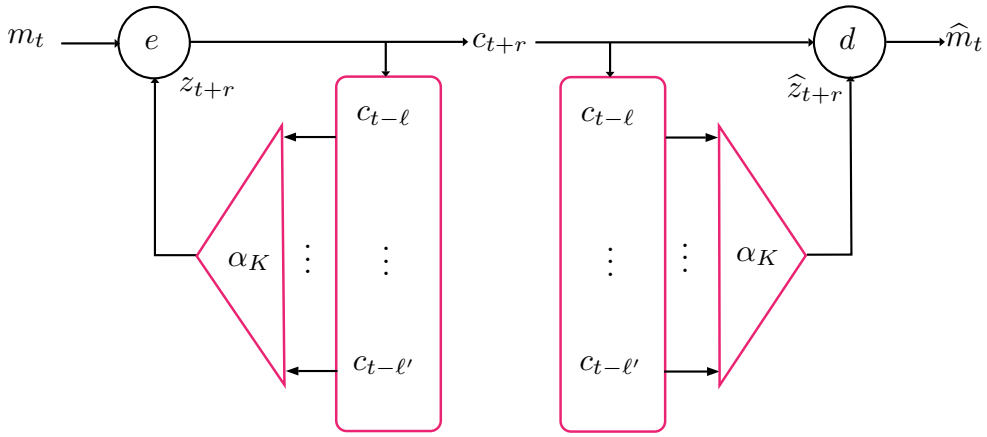


Figure 2.12: Canonical form of SSSC.

Following the methodology of Maurer, the purpose of this thesis is to propose an architecture based on dynamical systems to compute the keystream z_t as depicted on the Figure 2.13.

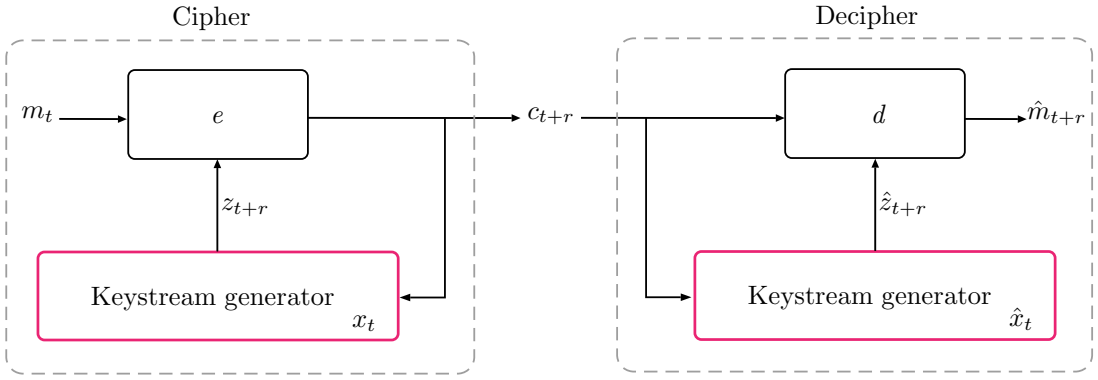


Figure 2.13: Flat LPV-based Self-Synchronizing Stream Ciphers block diagram.

Two classes of SSSC can be defined according to the delay of synchronization:

- *Deterministic*: the delay of synchronization is bounded by the constant M a priori fixed. This class of SSSC is discussed in Chapter 3.
- *Statistical*: the bound of the delay of synchronization is not constant but is a random variable with respect to the sequence of ciphertexts or the initial state vector. This class of SSSC is discussed in Chapter 4.

Next paragraph is devoted to the motivation of resorting to LPV systems to design SSSC.

The interest of flat LPV automata for the design of SSSC

For SSSC, the state vector x_t of the automaton (2.14) must be expressed as a function of a finite number of its shifted outputs as described by Equation (2.18). Furthermore, from the equalities (2.13) and (2.22), the same property holds for the input m_t. And yet, from a control-theoretical point of view, it is nothing but the property of difference flatness of (2.14). As a result, it will be stressed later than flat systems enlarge the existing class of all known SSSC ciphers found in the literature. Indeed, they may lead to automata with state transitions functions with non triangular form (see Section 3.3.1). It can be claimed that any flat automaton is an admissible candidate for the design of an SSSC. Besides, it is central to stress that flatness must be guaranteed for any value of the parametrization K since K acts the secret key. Otherwise stated, flatness must be guaranteed from a structural point of view. And yet, LPV automata getting a linear structure, they are appropriate candidates since structural flatness can be efficiently characterized for LPV systems as seen in Section 1.4.2. Besides, since for cryptographic purposes, automata must be necessarily nonlinear, LPV automata are interesting in that the nonlinearity can be incorporated in a flexible way, by choosing the varying parameters $\rho(t)$ as parametrized nonlinear functions. In the context of cryptography, those functions are implemented in the form of S-boxes.

$$\begin{array}{rccc} \rho^i : & \mathbb{F}^{s+1} & \to & \mathbb{F} \\ & (c_t, c_{t-1}, \cdots, c_{t-s}) & \mapsto & \rho^i(c_t, c_{t-1}, \cdots, c_{t-s}, SK_i) \end{array} \tag{2.23}$$

where SK_i is the subkey number i derived from the secret key K. As a conclusion, any realization of a structured linear system in the form of a flat LPV system involving arbitrary nonlinearities ρ^i may act as an SSSC. The point is how to systematically design a structured flat system. Then, from the family of resulting flat systems, the aim is to select realizations that are the more efficient regarding some performances criteria. Those issues are discussed in this Part II of the manuscript.

Performances of SSSC based on LPV automata and design

From the perspective of designing SSSC, three criteria regarding their performances are considered.

i) Self-synchronization property: it is related to flatness.

ii) Computational complexity: a higher value of the inherent degree r would make the computation of the coefficients of $P_{\rho(t:t+r)}$ (1.34) excessive for hardware implementation, since this computation involves products of S-Box. Hence the inherent delay impacts the complexity.

iii) Security: both criteria will be examined: the diffusion delay and the property of the state transition function of the decipher (triangular or not).

The developments in this part II highlight the fact that those criteria interplay one another.

Chapter 3

Design of deterministic Self-Synchronizing Stream Ciphers based on LPV automata

This chapter aims at proposing a new design methodology of deterministic Self-Synchronizing Stream Ciphers based on the construction of a structured flat automata. The interest is that, till now, any of the proposed cipher have triangular state transitions functions as opposed to the proposed approach. We show how the performances of SSSC, namely self-synchronization ability, computational complexity and security, can be expressed or are related to control-theoretical concepts. A Proof-of-Concept example is proposed as an illustration.

3.1 Construction of a structured flat linear automaton

The first step towards the design of an SSSC is the construction of a structured flat linear system in order to guarantee the self-synchronization property (criterion *i)*) as pointed out in the introduction. This subsection aims at giving a construction of a digraph $\mathcal{G}$, describing the equations of a structured system

$$\Sigma: \quad x_{t+1} \quad = \quad I_A \; x_t + I_B \; m_t \tag{3.1}$$

that leads to a flat LPV automaton with a given flat output. Since Theorem 1.4 ensures that the flat outputs are components x_t^i of x_t, otherwise stated, $c_t^j = x_j^i$, $j \in \{1, \ldots, p\}$ and $i \in \{1, \ldots, n\}$, the output function h can be written in a matrix way

$$h : x_t \in \mathbb{F}^n \mapsto h(x_t) = Cx_t \tag{3.2}$$

with C a $p \times n$ matrix with entries equal to 0 or 1.

The digraph $\mathcal{G}$ is parameterized by a 4-tuple of integers (n, p, n_a, r) where n is the dimension of the LPV system, p is the dimension of the input (the same as the output by construction of a flat system) such that $p \leq n$, n_a is the number of edges and r is the left inherent delay.

The main line of the construction proposed below consists in considering successively sets of edges and showing that the resulting digraph $\mathcal{G}$ that involves $n + p$ vertices fulfill Conditions **C0-C2**. For $r > 0$, $\mathbf{M} = \{\mathbf{m^1}, \ldots, \mathbf{m^p}\}$ is the set of p input vertices, $\mathbf{X} = \bigcup_{k=0}^{r-1} \mathbf{X_k} \cup \mathbf{X_r}$ is the set of the other n state vertices where $\mathbf{X_k} = \{\mathbf{x^{k \cdot p+1}}, \ldots, \mathbf{x^{(k+1) \cdot p}}\}$ and $\mathbf{X_r} = \{\mathbf{x^{r \cdot p+1}}, \ldots, \mathbf{x^n}\}$. The flat state vertices are clustered in $\mathbf{X_{r-1}}$ i.e. $\mathbf{V^F} = \mathbf{X_{r-1}}$. The digraph's vertices are depicted on Figure 3.1.

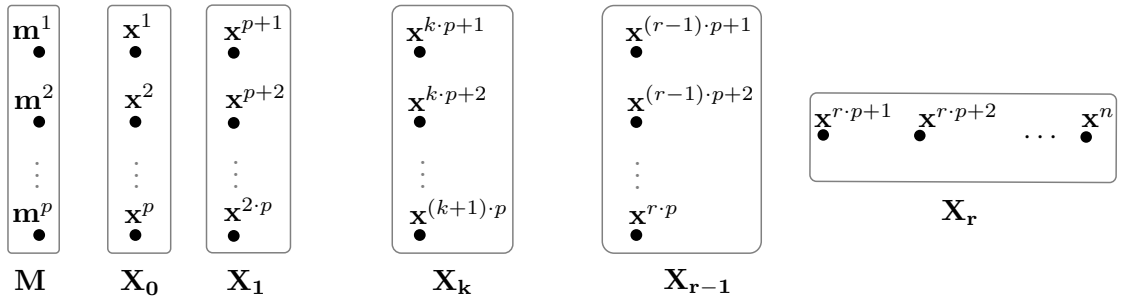


Figure 3.1: The digraph's vertices.

The special case $r = 0$ is ignored here. Indeed, for $r = 0$, the digraph would only involve the set $\mathbf{M}$ and the set $\mathbf{X_0}$ while the set $\mathbf{V^F}$ would coincide with $\mathbf{M}$. In our context, it would mean that the flat output c_t would directly depend on the plaintext symbol m_t and so would make no sense. Hence, it will be assumed hereafter that $r > 0$.

1. Let us define the following sets of edges in the digraph $\mathcal{G}$:

- $\mathcal{E}_M = \{(\mathbf{m^i}, \mathbf{x^i}),\ i = 1, \ldots, p\}$
- $\mathcal{E}_0 = \{(\mathbf{x^i}, \mathbf{x^{i+p}}),\ i = 1, \ldots, (r-2)\cdot p\}$ if $r > 1$ and $\mathcal{E}_0 = \emptyset$ if $r = 1$.

The digraph $\mathcal{G}$ obtained with $\mathcal{E}_M$ and $\mathcal{E}_0$ is depicted on Figure 3.2.

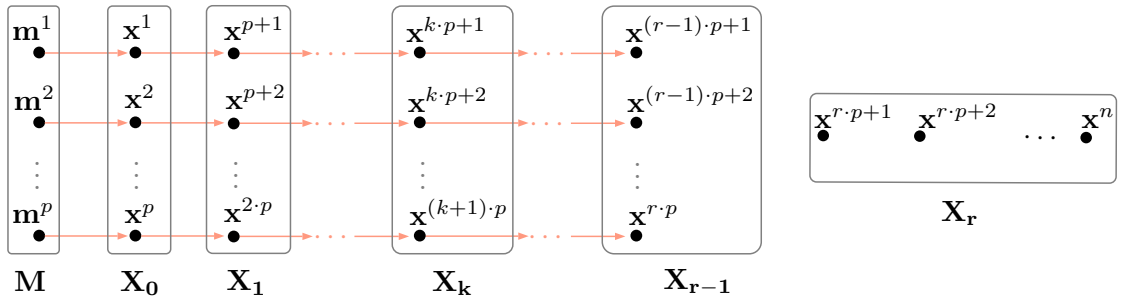


Figure 3.2: Digraph $\mathcal{G}$ obtained with $\mathcal{E}_M$ and $\mathcal{E}_0$.

The resulting digraph generates a set denoted by $\mathbf{P_{M-V_F}}$ of disjoints $\mathbf{M - V}_F$ paths of the same length. It is defined by

$$\begin{aligned}\mathbf{P_{M-V_F}} = \{&\mathbf{m^1} \to \mathbf{x^1} \to \mathbf{x^{p+1}} \to \cdots \to \mathbf{x^{k\cdot p+1}} \to \cdots \to \mathbf{x^{(r-1)\cdot p+1}};\\ &\mathbf{m^2} \to \mathbf{x^2} \to \mathbf{x^{p+2}} \to \cdots \to \mathbf{x^{k\cdot p+2}} \to \cdots \to \mathbf{x^{(r-1)\cdot p+2}};\\ &\vdots\\ &\mathbf{m^p} \to \mathbf{x^p} \to \mathbf{x^{2\cdot p}} \to \cdots \to \mathbf{x^{k\cdot p}} \to \cdots \to \mathbf{x^{r\cdot p}}\}\end{aligned}$$

There is only one $\mathbf{M - V^F}$ maximum linking. It coincides with $\mathbf{P_{M-V_F}}$. Thus, the vertices of the the set $\mathbf{M} \cup \mathbf{X_0} \cup \cdots \cup \mathbf{X_{r-1}}$ which are covered by all the disjoints $\mathbf{M - V}_F$ paths are essential vertices *i.e.* $V_{ess}(\mathbf{M}, \mathbf{V^F}) = \mathbf{M} \cup \mathbf{X_0} \cup \cdots \cup \mathbf{X_{r-1}}$.

The size of the $\mathbf{M - V^F}$ maximum linking is $\eta(\mathbf{M}, \mathbf{V^F}) = p$ and Condition **C0** is fulfilled. The length of the $\mathbf{M - V^F}$ maximum linking is $\ell(\mathbf{M}, \mathbf{V^F}) = p \cdot r$. Since the $\mathbf{M - V^F}$ maximum linking is unique, then

Condition **C1** is fulfilled. Since there are not any cycle in the digraph, Condition **C2** is clearly fulfilled.

2. The digraph is supplemented by a set of edges that link the vertices of $\mathbf{X_{r-1}}$ to any of the vertices of $\mathbf{X_r}$, that is the set:

- $\mathcal{E}_1 = \{(\mathbf{x^i}, \mathbf{x^j}),\ i = (r-1)\cdot p + 1, \ldots, r\cdot p,\ j = r\cdot p + 1, \ldots, n\}$.

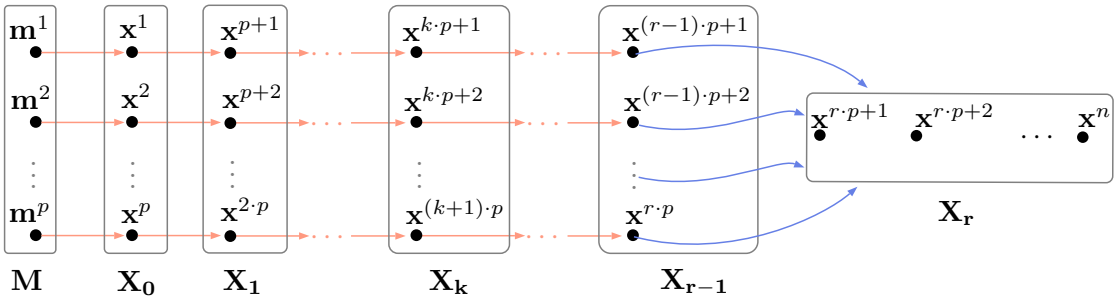


Figure 3.3: Digraph $\mathcal{G}$ obtained with $\mathcal{E}_M$ and $\mathcal{E}_0$ supplemented with the set $\mathcal{E}_1$.

The digraph $\mathcal{G}$ obtained with $\mathcal{E}_M$, $\mathcal{E}_0$ supplemented with the set $\mathcal{E}_1$ is depicted on Figure 3.3. There are no new maximum linking and there are not any cycle. Hence, Conditions **C0-C2** are fulfilled.

3. The digraph is supplemented with the set of edges that link the vertices of $\mathbf{X_r}$ to each other without generating cycles, that is the set of edges:

- $\mathcal{E}_2 = \{(\mathbf{x^i}, \mathbf{x^j}),\ i = r\cdot p + 1, \ldots, n-1,\ j = r\cdot p + 2, \ldots, n, j > i\}$

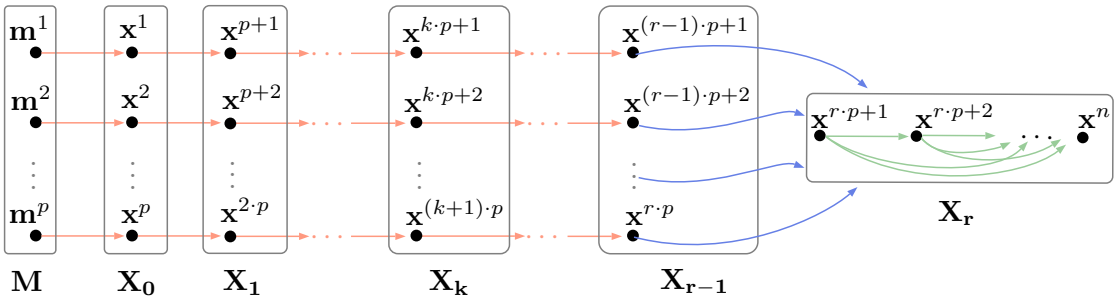


Figure 3.4: Digraph $\mathcal{G}$ obtained with $\mathcal{E}_M$, $\mathcal{E}_0$, $\mathcal{E}_1$ supplemented with the set $\mathcal{E}_2$.

The digraph $\mathcal{G}$ obtained with $\mathcal{E}_M$, $\mathcal{E}_0$, $\mathcal{E}_1$ supplemented with the set $\mathcal{E}_2$ is depicted on Figure 3.4. There are no new maximum linking and there are not any cycle. Hence, Conditions **C0-C2** are fulfilled.

4. The digraph is supplemented with the set of edges that link the vertices of $\mathbf{X_r}$ to any of the vertices of $\mathbf{X_0} \cup \mathbf{X_1} \cup \cdots \cup \mathbf{X_{r-1}}$, that is the set of edges:

- $\mathcal{E}_3 = \{(\mathbf{x^i}, \mathbf{x^j}),\ i = r\cdot p + 1, \ldots, n,\ j = 1, \ldots, r\cdot p\}$

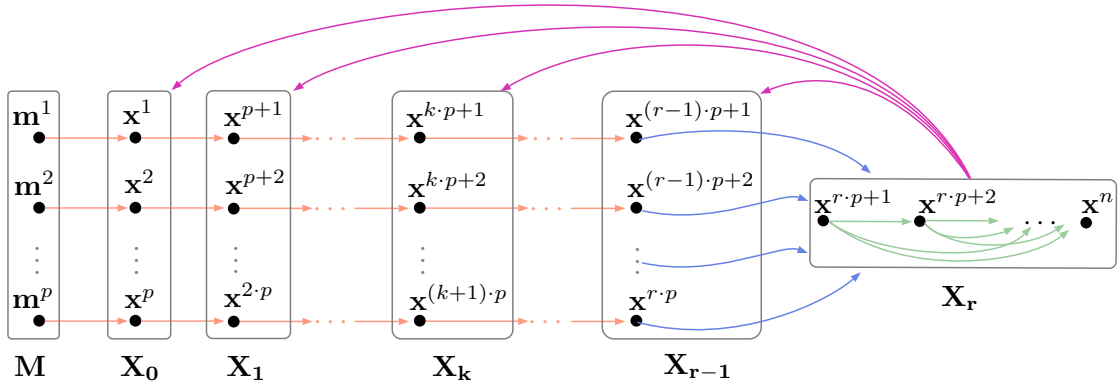


Figure 3.5: Digraph $\mathcal{G}$ obtained with $\mathcal{E}_M$, $\mathcal{E}_0$, $\mathcal{E}_1$, $\mathcal{E}_2$ supplemented with the set $\mathcal{E}_3$.

The digraph $\mathcal{G}$ obtained with $\mathcal{E}_M$, $\mathcal{E}_0$, $\mathcal{E}_1$, $\mathcal{E}_2$ supplemented with the set $\mathcal{E}_3$ is depicted on Figure 3.5. There are no new maximum linking. Hence, Conditions **C0-C1** are fulfilled. On the other hand, the set $\mathcal{E}_3$ generates cycles. However, the cycles necessarily cover vertices belonging to $V_{ess}(\mathbf{M}, \mathbf{V^F})$. Hence, Condition **C2** is also fulfilled.

5. The digraph is supplemented with the set of edges that generates self-cycles on any of the vertices of $\mathbf{X_0} \cup \mathbf{X_1} \cup \cdots \cup \mathbf{X_{r-1}}$, that is the set of edges:

- $\mathcal{E}_4 = \{(\mathbf{x^i}, \mathbf{x^i}),\ i = 1, \ldots, r \cdot p\}$

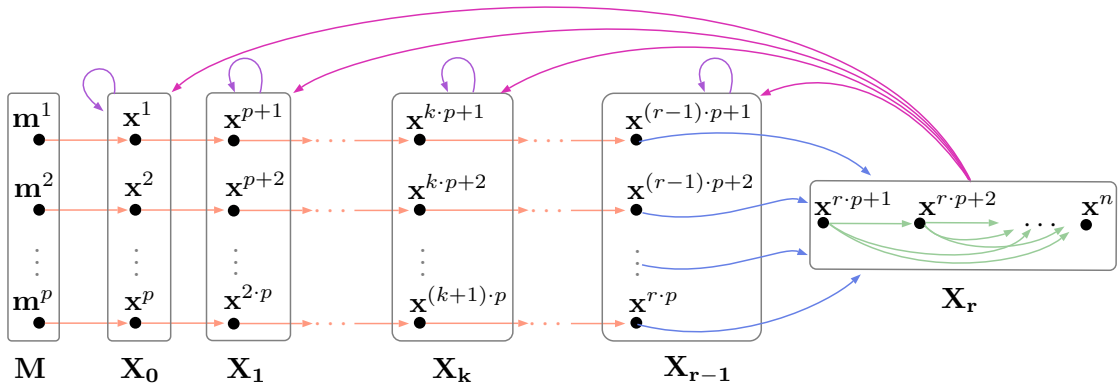


Figure 3.6: Digraph $\mathcal{G}$ obtained with $\mathcal{E}_M$, $\mathcal{E}_0$, $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$ supplemented with the set $\mathcal{E}_4$.

The digraph $\mathcal{G}$ obtained with $\mathcal{E}_M$, $\mathcal{E}_0$, $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$ supplemented with the set $\mathcal{E}_4$ is depicted on Figure 3.6. There are no new maximum linking. Hence, Conditions **C0-C1** are fulfilled. On the other hand, the set $\mathcal{E}_4$ generates cycles. However, the cycles necessarily cover the vertices belonging to $V_{ess}(\mathbf{M}, \mathbf{V^F})$. Hence, Condition **C2** is also fulfilled.

6. The digraph is supplemented with the set of edges that link the vertices of $\mathbf{X_k}$ to the vertices of $\mathbf{X_0} \cup \mathbf{X_1} \cup \cdots \cup \mathbf{X_{k-1}}$ $(1 \leq k \leq r-1)$, that is the set of edges:

- $\mathcal{E}_5 = \{(\mathbf{x^i}, \mathbf{x^j}),\ i = 1 + kp, \ldots, (k+1)p,\ j = 1 + k'p, \ldots, (k'+1)p,\ k = 1, \ldots, r-1,\ k' = 0, \ldots, r-2,\ k' < k\}$

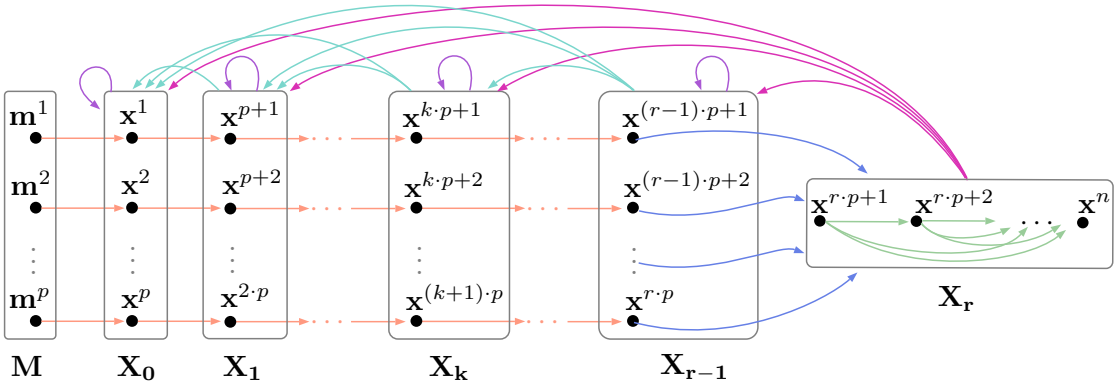


Figure 3.7: Digraph $\mathcal{G}$ obtained with $\mathcal{E}_M$, $\mathcal{E}_0$, $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$, $\mathcal{E}_4$ supplemented with the set $\mathcal{E}_5$.

The digraph $\mathcal{G}$ obtained with $\mathcal{E}_M$, $\mathcal{E}_0$, $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$, $\mathcal{E}_4$ completed with the set $\mathcal{E}_5$ is depicted on Figure 3.7. There are no new maximum linking. Hence, Conditions **C0-C1** are fulfilled. On the other hand, the set $\mathcal{E}_5$ generates cycles. However, the cycles necessarily cover vertices belonging to $V_{ess}(\mathbf{M}, \mathbf{V^F})$. Hence, Condition **C2** is also fulfilled.

7. The digraph is supplemented with the set of edges that link the vertices of $\mathbf{X_k}$ to any of the vertices of $\mathbf{X_{k+1}}$ for $0 \leq k \leq r-2$, that is the set of edges:

- $\mathcal{E}_6 = \{(\mathbf{x^i}, \mathbf{x^j}),\ i = 1 + kp, \ldots, (k+1)p,\ j = 1 + (k+1)p, \ldots, (k+2)p,\ k = 0, \ldots, r-2\}$

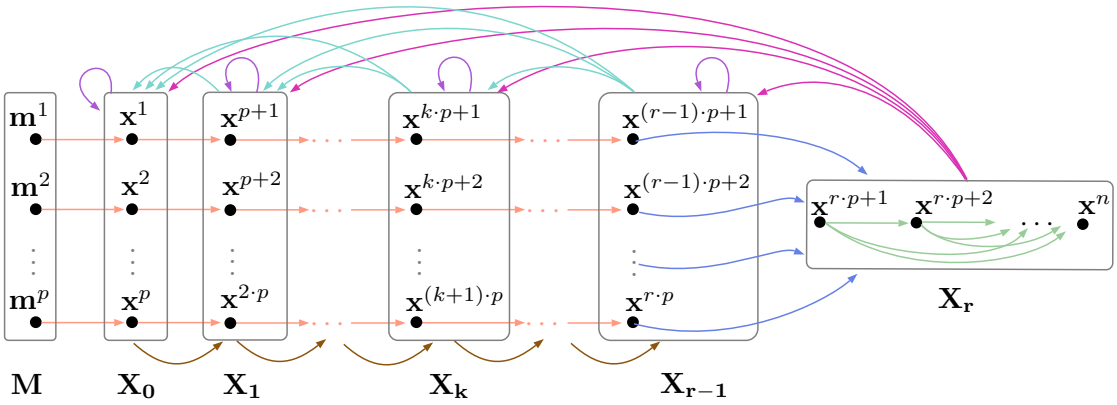


Figure 3.8: Digraph $\mathcal{G}$ obtained with $\mathcal{E}_M$, $\mathcal{E}_0$, $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$, $\mathcal{E}_4$, $\mathcal{E}_5$ supplemented with the set $\mathcal{E}_6$.

Finally, to complete the construction, the digraph $\mathcal{G}$ obtained with $\mathcal{E}_M$, $\mathcal{E}_0$, $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$, $\mathcal{E}_4$, $\mathcal{E}_5$ supplemented with the set $\mathcal{E}_6$ is depicted on Figure 3.8. The construction yields new disjoint paths and so new maximum linking. However, since the new edges link the vertices of $\mathbf{X_k}$ to $\mathbf{X_{k+1}}$, the length of any $\mathbf{M} - \mathbf{V^F}$ maximum linking is equal to r and each of them involves p disjoint paths. Hence, Conditions **C0-C1** are fulfilled. The set $V_{ess}(\mathbf{M}, \mathbf{V^F})$ remains unchanged and the set $\mathcal{E}_6$ does not generate new cycles. Hence, Condition **C2** is also fulfilled.

From a simple counting, it can be inferred that the maximal number n_M of edges in $\mathcal{G}$ resulting from the sets $\mathcal{E}_M$ and $\mathcal{E}_i$ for $i \in [0, 6]$ is

$$n_M = np(r+1) + p(r - p(r^2+1) + 1) + p^2\frac{(r-1)r}{2} + \frac{(n-pr)(n-pr-1)}{2}$$

However, any digraph with number $n_a \leq n_M$ of edges in the set $\mathcal{E}_M \cup \mathcal{E}_0 \cup \cdots \cup \mathcal{E}_6$ still fulfills Conditions **C0-C2**.

Proposition 3.1 *The LPV automaton Σ_ρ (1.1) corresponding to the digraph $\mathcal{G}$, parameterized by the 4-tuple of integers (n, p, n_M, r) and defined by the set $\mathcal{E}_M \cup \mathcal{E}_0 \cup \cdots \cup \mathcal{E}_6$, is structurally flat with flat outputs $c_t^j = x_t^i$, $j = 1, \ldots, p$ and $i = (r-1)p + 1, \ldots, r \cdot p$.*

Proof 3.1 *The proof is a direct consequence of the aforementioned construction and its analysis with respect to Conditions **C0-C2** of Theorem 1.4.*

In the next section, the general equations of SSSC given in the introduction of Part II are particularized when considering the special class of flat LPV automata. The next section has a particular focus on criterion *ii)* (computational complexity).

3.2 Equations of the LPV SSSC

Consider the LPV automaton described by

$$\Sigma_\rho : \left\{ \begin{array}{lcl} x_{t+1} & = & A_{\rho(t)}\ x_t + B_{\rho(t)}\ m_t \\ c_t & = & C_{\rho(t)}\ x_t \end{array} \right. \tag{3.3}$$

Let us recall that the set of flat outputs c_t are components x_t^i of x_t , that is $c_t^j = x_j^i$, $j \in \{1, \ldots, p\}$ and $i \in \{1, \ldots, n\}$. Thus, matrix C of Equation (3.3) verifies (3.2) and is a canonical $p \times n$ matrix.
In order to reduce the computational complexity, the following consideration is central.
Let us define $r \geq 1$ as the smallest integer such that

$$\mathcal{T} = C \prod_{l=t+r-1}^{l=t+1} A_{\rho(l)} B \tag{3.4}$$

is an invertible matrix. In such a case, it is worthwhile realizing that Equation (1.27) is necessarily fulfilled, meaning that system (3.3) is left invertible. Furthermore, if the following condition is also fulfilled,

$$C \prod_{l=t+s-1}^{l=t+1} A_{\rho(l)} B = 0 \tag{3.5}$$

for any integer $s < r$, then the dynamical equation of the left inverse system (1.29) simplifies and reads

$$\hat{x}_{t+r+1} \quad = P_{\rho(t:t+r)} \hat{x}_{t+r} + B_{\rho(t)} \mathcal{T}^{-1} c_{t:t+r} \tag{3.6}$$

with

$$P_{\rho(t:t+r)} = A_{\rho(t)} - B\mathcal{T}^{-1} C \prod_{l=t+r-1}^{t} A_{\rho(l)}. \tag{3.7}$$

The assumption that (3.5) holds allows the pseudo inverse involved in the equations of the left inverse system (1.29) to boil down to an inverse matrix. It handles a critical issue from a computational point of view since the computation of the pseudo-inverse would be redhibitory in practice.
The vector c_{t+r} is the output c_t that has been iterated r times. Hence, it reads

$$c_{t+r} = C \prod_{l=t+r-1}^{t} A_{\rho(l)} x_t + \mathcal{T} m_t. \tag{3.8}$$

Then, letting

$$z_{t+r} = C \prod_{l=t+r-1}^{t} A_{\rho(l)} x_t, \tag{3.9}$$

Equation (3.8) can be rewritten as

$$c_{t+r} = z_{t+r} + \mathcal{T} m_t. \tag{3.10}$$

Next, let us define

$$\widehat{z}_{t+r} \quad = \quad C \prod_{l=t+r-1}^{t} A_{\rho(l)} \widehat{x}_t \tag{3.11}$$

and

$$\widehat{m}_{t+r} = \mathcal{T}^{-1}(c_{t+r} - \widehat{z}_{t+r}). \tag{3.12}$$

It follows that

$$\widehat{m}_{t+r} = m_t \text{ if } \widehat{x}_{t+r} = x_t. \tag{3.13}$$

Let us recall that the left inverse system ensures that $\widehat{x}_{t+r} = x_t$ for $t \geq M$ according to the definition of the left inverter and the property of flatness (see Section 1.3.1 and 1.3.2). This being the case, since the state of the automaton (3.6) coincides with the state of the automaton (3.3) for $t \geq M$, the input m_t of (3.3) can be recovered after a finite transient time as well by performing (3.13). As a result, the dynamical system (1.16) acts as the deciphering part of an SSSC with $\widehat{z}_t$ playing the role of the keystream.
To sum up, by identification with the general equations given in Section II, the following conclusion can be drawn. The set of Equations (3.3) (cipher keystream generator), (3.9) (keystream) and (3.10) (encryption function e) describes the ciphering part involving an LPV automaton. This set particularizes the general equations (2.14) of the cipher, (2.17) (second equality) and (2.12) respectively. The set of Equations (3.6) (decipher keystream generator), (3.11) (keystream) and (3.12) (deciphering function) describes the deciphering part involving an LPV automaton. This set particularizes the general equations (2.21) and (2.13) respectively. The architecture is depicted on Figure 3.9.

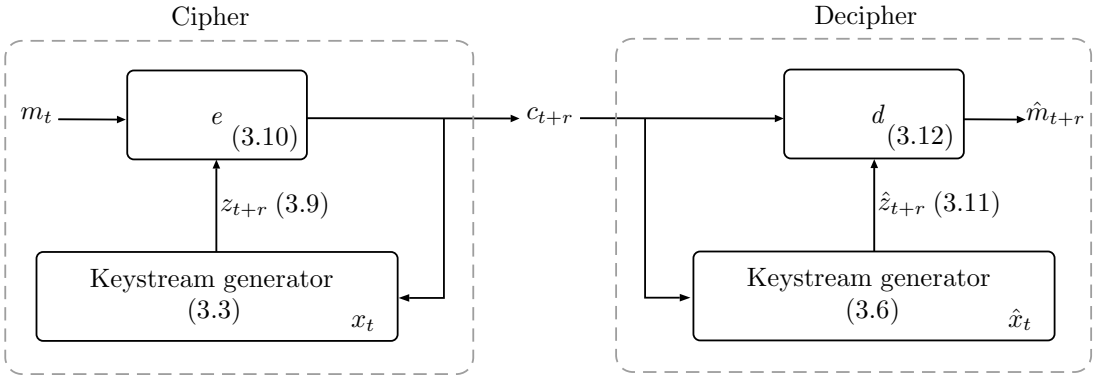


Figure 3.9: Flat LPV-based Self-Synchronizing Stream Ciphers block diagram.

From the 4-tuple (n, p, n_a, r) characterizing the digraph $\mathcal{G}$, it can be noticed that the parameter r, that is the left inherent delay, has a significant impact on the computational complexity. Due to the matrices multiplication involved in (3.7) at the deciphering side, the computational complexity grows up rapidly when r increases. Therefore, it is natural to attempt to reduce the left inherent delay to its minimum value, $r = 1$ ($r = 0$ would mean that the ciphertext c_t is equal to the plaintext m_t and so makes no sense). However, it will be shown below that it also impacts the security.

3.3 Considerations on the security

This subsection is devoted to the criterion *iii)* (security). In cryptography, state transition functions g_K of automata like (2.17) that propagate dependencies in one direction only are called T–functions (T for Triangle).

3.3.1 T–functions

T–functions are known to suffer from weaknesses [JM06a] because of the inherent propagation of differential properties that facilitate cryptanalysis (the work that consists in assessing weakness of cryptosystems). The following proposition holds.

Proposition 3.2 *If c_t is a flat output with a constant output matrix C as defined by* (3.2) *and $r = 1$, then the state transition function of the decipher* (3.6) *is necessarily a T-function up to a change of basis in the form of a permutation.*

Proof 3.2 *If c_t is a flat output, then, for any sequence (ρ), any product of matrices in* (1.32) *gives zero. In the case when $r = 1$ and C is constant, the matrix $P_{\rho(t:t+r)}$ depends only on the index t and the product* (1.32) *is therefore commutative. Hence, the set of matrices $P_{\rho(t:t+r)}$ for all sequences (ρ) generates a nilpotent semigroup*[7]*. According to Levintsky's theorem (Theorem 2.1.7 stated in [RR00]), this set of matrices can be simultaneously triangularized and thus, the state transition function is conjugate to a T-function.*

On the other hand, the following remark applies.

Remark 3.1 *For $r > 1$, the set of matrices $P_{\rho(t:t+r)}$ for all sequences (ρ) does not necessarily admit a simultaneous triangularization. For example, let us considering $r = 2$ and $n = 3$. Let have*

$$A_{\rho(t)} = \begin{pmatrix} \rho^1(t) & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & \rho^2(t) \end{pmatrix}, B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \text{ and } C = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

One has $P_{\rho(t:t+2)} = A_{\rho(t)} - B(CA_{\rho(t+1)}B)^{-1}CA_{\rho(t+1)}A_{\rho(t)}$. One obtains

$$P_{\rho(t:t+2)} = \begin{pmatrix} -\rho^2(t+1) & 0 & -\rho^2(t)\rho^2(t+1) \\ 0 & 0 & 0 \\ 1 & 0 & \rho^2(t) \end{pmatrix}$$

The characteristic polynomial of $P_{\rho(t:t+2)}$ is given by $N(X) = X^3 + (\rho^2(t+1) - \rho^2(t))X^2$.
Hence, the eigenvalues of $P_{\rho(t:t+2)}$ are 0 and $\rho^2(t) - \rho^2(t+1)$. It follows that the eigenspace related to the eigenvalues of $P_{\rho(t:t+2)}$ is spanned by

$$v_1 = \begin{pmatrix} 1 \\ 0 \\ \rho^2(t) \end{pmatrix}, v_2 = \begin{pmatrix} 1 \\ 0 \\ \rho^2(t+1) \end{pmatrix}, v_3 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

And yet, a necessary condition for simultaneous triangularization (see [Joc09]) is that the matrices $P_{\rho(t:t+2)}$ share a common eigenvector for any realization $\rho(t)$. As a result, since the parameter $\rho(t)$ varies, the matrices $P_{\rho(t:t+2)}$ do not fulfill such a requirement.

3.3.2 Diffusion delay

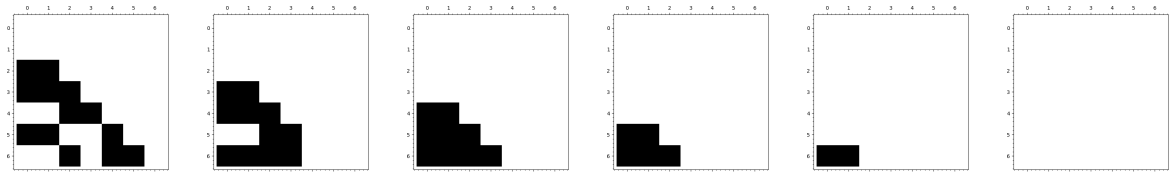
We focus here on a security criteria called diffusion and explained below. Let $\mathbf{p}$ denotes the power of a matrix $Z \in M_n(\mathbb{F})$. The diffusion delay, introduced in [Arn+11b], is the smallest value, denoted by d_0, of $\mathbf{p}$ such that Z^p does not have any zero coefficient. In other words, it is the smallest value of $\mathbf{p}$ such that each element of the initial internal state x_0 has influenced every element of x_t for $t \geq d_0$. Hence, the investigation of such properties relies on symbolic matrix P_S obtained by replacing any non zero entries of P (3.7) by the symbol 'S'. It is then a structural property. From design perspectives, the smaller d_0, the better the diffusion. In our context, we focus on the diffusion property of the decipher.

[7] 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.

Let us consider a digraph $\mathcal{G}(\Sigma_\Lambda)$ related to $r = 1$ as depicted on Figure 3.11. The matrix P_S obeys (3.7).

$$P_S = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ S & S & 0 & 0 & 0 & 0 & 0 \\ S & S & S & 0 & 0 & 0 & 0 \\ 0 & 0 & S & S & 0 & 0 & 0 \\ S & S & 0 & 0 & S & 0 & 0 \\ 0 & 0 & S & 0 & S & S & 0 \end{pmatrix}$$

The study of the diffusion must focus on the successive powers $P_S^{\mathbf{p}}$ as $\mathbf{p}$ increases. Indeed, the power of matrix results from the successive iterations of the deciphering equations given by (3.6).



P_S P_S^2 P_S^3 P_S^4 P_S^5 P_S^6

Figure 3.10: Powers of $P_S^{\mathbf{p}}$ for $\mathbf{p} \in [1, 6]$, $r = 1$. Black squares correspond to non zero entries while blank ones correspond to null entries.

The successive powers of $P^{\mathbf{p}}$ undergo a bad diffusion as illustrated in Figure 3.10. Indeed, it can be noticed that P is lower triangular. Actually, it has been shown in Section 3.3.1, by construction of the digraph fulfilling Conditions **C0**-**C2**, that it is a general property for $r = 1$. And yet, it can be easily proved that the product of two triangular matrices is a triangular matrix. Hence, P_S^7 reaches systematically the null matrix since P is nilpotent by construction.

Remark 3.2 *Actually, for any $r > 1$, the product of P_S will systematically reach the null matrix according to the flatness property* (1.32)*, no matter whether P_S is nilpotent or not.*

3.4 Design methodology of a deterministic SSSC

3.4.1 Overall procedure

The following Steps Si summarize the way how to construct a deterministic SSSC. The construction is presented for arbitrary left inherent delay r.
Step S0: Choose a 4-tuple (n, p, n_a, r) with n the dimension of the state, p the number of inputs (equal to the number of outputs), r the left inherent delay and n_a the number of edges in the digraph $\mathcal{G}$.

Step S1: Set n as the dimension of the state vector x_t. Choose p components x_t^i ($i \in \{1, \ldots, n\}$) as the desired flat output vector $c_t = Cx_t$ and a compatible matrix B of dimension $n \times p$ such that (3.4) and (3.5) hold for the left inherent delay r.

Step S2: Construct a digraph $\mathcal{G}$ fulfilling Conditions **C0**-**C2** with n_a edges according to Proposition 3.1. Derive the matrices I_A and I_B of the structured linear system Σ (1.7) from the adjacency matrix

Step S3: Replace some non-zero entries of I_A by nonlinear functions $\rho^i(t) = \rho^i(c_t, c_{t-1}, \cdots, c_{t-s})$ to construct the matrix $A_{\rho(t)}$ of (3.3). Not all '1' entries of I_A must be assigned to a non-linear function. Some of them can be merely constant. The choice must obey a trade-off between complexity of the architecture and security (that goes beyond the consideration on T-functions but this issue is not addressed

here, further remarks on security will be however given in the conclusion of this manuscript).

Step S4: Complete the design by calculating explicitly the equations assigned to the blocks and equations of Figure 3.9 depicting the cipher and the decipher parts of a deterministic SSSC.
The section below gives an example that illustrates the way how the LPV-based deterministic self-synchronization operates.

3.4.2 Proof-of-Concept Example

Let us consider an automaton operating over the finite field $\mathbb{F}_{16}$ defined by $\mathbb{F}_2[X]/(X^4+X+1)$. Then, let us consider a plaintext sequence (m) with symbols $m_t \in \mathbb{F}_{16}^{\ell}$ to be encrypted.

Step S0: Consider the 4-tuple $(n=7, p=2, n_a=19, r=1)$.
Step S1: Let $n=7$ be the dimension of the state x_t of the automaton and $p=2$ be the dimension of the output. Let B and C as

$$B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Thus, (3.4) and (3.5) hold for the left inherent delay $r=1$ since the product CB is the identity matrix. Thus, the output of the automaton fulfills $c_t = Cx_t = (x_t^1, x_t^2)$.

Step S2: For the 4-tuple considered in Step S0, construct the full digraph $\mathcal{G}$ according to the methodology presented in Subsection 3.1. It involves $n+p$ vertices and $n_M = 36$ edges and it necessarily fulfills Conditions **C0-C2**. Then, remove edges in the sets $\mathcal{E}_i$ for $i \in [1,6]$ to obtain a digraph with $n_a = 19$ edges. The selection of edges to be removed is here arbitrary. Actually, it should obey some criteria regarding the security of the expected cryptosystem which is not discussed here. The resulting digraph is depicted on Figure 3.11. The flat state vertices are clustered in $\mathbf{V^F} = \mathbf{X_0}$.

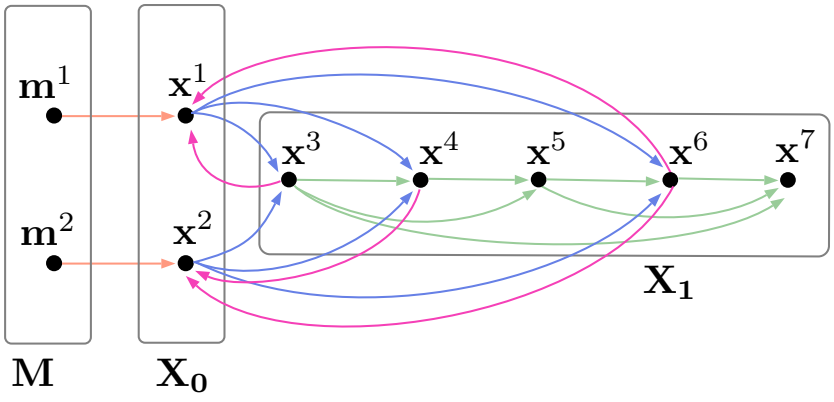


Figure 3.11: Digraph $\mathcal{G}$ with $n=7$, $p=2$, $n_a=19$, $r=1$. The vertices corresponding to the states x_i with $i=3,\ldots,7$.

The matrices I_A and I_B of the structured linear system Σ (3.1) are derived from the adjacency matrix (1.8) and read

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

Step S3: The nonlinearity defined by the S-box is a unique function ρ ($i = 1$, $s = 0$ and $c_t = Cx_t = (x_t^1, x_t^2)$, see (2.23)) given by

$$\begin{array}{rccc} \rho: & \mathbb{F}_{16} & \to & \mathbb{F}_{16} \\ & c_t^1 & \mapsto & S(c_t^1 \oplus SK) \end{array} \tag{3.14}$$

where the key SK is a 4-bit word chosen randomly for each $\rho(t)$. The function S is the bijective S-Box borrowed from Piccolo [Shi+11].
As a Proof-Of-Concept example, the selection of the entries of I_A that are replaced by the S-box is here arbitrary. Let $A_{\rho(t)}$ read as follows.

$$A_{\rho(t)} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ \rho(t) & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \rho(t) & 0 & 0 & 0 \\ \rho(t) & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \rho(t) & 0 & 1 & \rho(t) & 0 \end{pmatrix}$$

Step S4: To complete the design, we calculate explicitely the equations assigned to the blocks and variables of Figure 3.9 depicting the cipher and decipher part of a deterministic SSSC. In particular, $\mathcal{T}$ is the identity matrix of dimension 2 and the dynamical matrix of the left-inverse system (1.29) obeys Equation (3.7) that reads, for $r = 1$ and taking into account $\mathcal{T}$

$$P_{\rho(t:t+1)} = A_{\rho(t)} - BCA_{\rho(t)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \rho(t) & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \rho(t) & 0 & 0 & 0 \\ \rho(t) & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \rho(t) & 0 & 1 & \rho(t) & 0 \end{pmatrix}$$

3.5 Conclusion

We have provided a method for constructing SSSCs that possibly yield non-triangular state transition functions. The SSSC are based on flat LPV automata. Proposition 3.2 together with Remark 3.1 and Remark 3.2 point out the issue of a trade-off between computational complexity (criterion *ii)*) and security (criterion *iii)*). If the computational burden is neglected, the security regarding the consideration on T-function can be guaranteed by setting $r > 1$. On the other hand, it is clear that $r = 1$ is a good setting from the computational point of view. However, the fact that it necessarily leads to state transition functions conjugate to T-functions is a limitation regarding the security. In every cases regarding the value of r, the diffusion will be bad. To handle the problem, a hybrid architecture that leads to the concept of statistical SSSC can be proposed. Those issues are detailed in Chapter 4.

Chapter 4

Design of statistical Self-Synchronizing Stream Ciphers based on switched automata

In Section 4.1, we propose a methodology to design a statistical SSSC based on switched LPV automata. It will be shown that it is a solution to fix the point of bad diffusion inherent to flat systems as pointed out in previous chapter. Switched automata have been motivated for the first time in [JR99] in the context of SSSC. Essentially, this concept calls for two modes operations, each one benefiting from a specific outcome. A switching rule orchestrates the successive modes of operation. The price to pay is that the overall architecture does not admit a deterministic self-synchronization property but a statistical one. The LPV-based hybrid architecture presented in this subsection is in the same vein. It is motivated by achieving a trade-off between self-synchronization property, computational complexity and security. Indeed, one mode will correspond to a flat system, hence ensuring self-synchronization but with bad diffusion. The other mode will correspond to a non flat system, with non necessarily triangular state transition function and with good diffusion. The purpose of Section 4.2 is to propose another statistical architecture. It is based on switched linear automata having a dead-beat stabilizability property. The aim is to design directly the matrix P that defines the decipher. We tackle the peculiarities due to the consideration of finite fields for which the notion of orthogonality deserves a special treatment. We propose a code written in SageMath to illustrate the principle. Whether this approach has a potential interest in the context of cryptography is left to future investigation.

4.1 Design of a statistical SSSC based on switched LPV automata

4.1.1 General principle

The main idea is to define two modes of operations. Both modes involve LPV systems with a left inherent delay $r = 1$ for the sake of computational complexity as stressed in Subsection 3.2.

The hybrid architecture with switching rule σ obeys the following state space representation at the cipher part:

$$x_{t+1} = \begin{cases} A_{\rho(t)}x_t + Bm_t \text{ if } \sigma(t) = 1 \\ A'_{\rho(t)}x_t + Bm_t \text{ if } \sigma(t) = 2 \end{cases} \tag{4.1}$$

with $x_t \in \mathbb{F}^n$ is the state vector of the cipher, $m_t \in \mathbb{F}^p$ is the plaintext input. The matrices $A \in \mathbb{F}^{n\times n}$ and $A' \in \mathbb{F}^{n\times n}$ are the dynamical matrices of respective modes 1 and 2, $B \in \mathbb{F}^{n\times p}$ is the input matrix of

the cipher.

The hybrid architecture obeys the following state space representation at the decipher part:

$$\widehat{x}_{t+2} = \begin{cases} P_{\rho(t:t+1)}\widehat{x}_{t+1} + Bc_{t+1} \text{ if } \sigma(t) = 1 \\ P'_{\rho(t:t+1)}\widehat{x}_{t+1} + Bc_{t+1} \text{ if } \sigma(t) = 2 \end{cases} \tag{4.2}$$

All the equations of the cipher and decipher given in Subsection 3.2 still hold for the mode 2 except that A is replaced by A'.
Now, let us characterize each mode.

- For the mode $\sigma(t) = 1$, the system with dynamical matrix A must be flat. The corresponding digraph $\mathcal{G}$ fulfills Conditions **C0**-**C2** and the system gets the self-synchronization property. In other words, for any initial conditions of the cipher and the decipher and any plaintext message, the synchronization is guaranteed with a delay bounded by the dimension n of the system. However, it necessarily involves a state transition function in the form of a T–function according to Proposition 3.2.
- For the mode $\sigma(t) = 2$, the system with dynamical matrix A' is designed so as it is not flat. The corresponding digraph $\mathcal{G}'$ does not fulfill Conditions **C0**-**C2** and since those conditions are necessary and sufficient for an output to be flat, the system does not get the self-synchronization property. In such a case, a state-transition function not necessarily in the form of a T–function is admissible. To complete the design of a statistical SSSC by supplemented the deterministic SSSC with a non flat mode, the following step yielding a hybrid architecture is added. (see Section 3.4.1 Design methodology of a deterministic SSSC)

  Switching rule
  Both modes are orchestrated according to the switching rule σ. In our context, it means that the hybrid architecture is equipped with a supervisor at both sides that manages the synchronization. The switching rule σ is detailed as follows. At the initialization, both the cipher and the decipher are in the mode $\sigma(t) = 1$. After M iterations, since the self-synchronization is ensured because of the flatness property characterizing mode $\sigma(t) = 1$, the cipher and the decipher switch on the mode $\sigma(t) = 2$. In this mode, since the synchronization had already occurred, although the system is not flat, the synchronization is preserved (due to the inherent property of the left-inverse system). However, in case of a transmission error, a corrupted cryptogram c_t may be received by the decipher while the cryptogram c_t is not altered for the cipher (no remote transmission for the cipher). Recalling that the dynamical matrices A and P of both the cipher and the decipher depend on the cryptogram (through time-varying entries in the form of S-boxes), they will not coincide and a desynchronization will occur. To overcome this issue, in the mode $\sigma(t) = 2$, the supervisor at both sides is scanning on-line the ciphertext symbols of the sequence (c). Let us notice that both the cipher and the decipher access the past and present ciphertext symbols. Those symbols are compared on-line with a finite number of symbols of reference defining a so-called sync pattern of length $l \in \mathbb{N}$. It is denoted by $(c_0^*, \ldots, c_{l-1}^*)$ with $c_i^* \in \mathbb{F}$. Then, when they coincide, the supervisor of both the cipher and the decipher switches in mode $\sigma(t) = 1$.

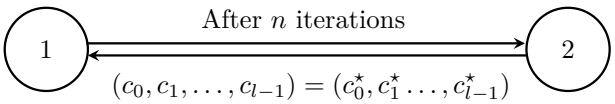


Figure 4.1: Representation of the automaton that defines the switching rule σ and its two corresponding modes.

Step S5: Compute the matrices of the cipher and decipher of the mode $\sigma(t) = 2$. In this perspective, choose a matrix A' so that the output $c_t = Cx_t$ is no longer flat. In this regard, we must

consider the digraph $\mathcal{G}$ and add edges so that Conditions **C0**-**C2** are no longer fulfilled. An admissible option is to add edges on non essential vertices. Indeed, in such a case, **C2** is not fulfilled. Besides, choose a sync pattern $(c_0^*, \ldots, c_{l-1}^*)$ that will trigger the local supervisor to switch from mode $\sigma(t) = 2$ to mode $\sigma(t) = 1$.

It must be pointed out that the sync pattern occurs in a statistically distributed way in the cryptogram sequence (c). Hence, the time before resynchronization is statistical and no longer deterministic, it is the price to pay. This being the case, it is worthwhile deriving the probability law of the synchronizaion.

Proposition 4.1 *Consider the sequence (c) as random and independent cryptogram symbols which are uniformaly distributed with probability p. The probability $\mathbb{P}$ of the first occurrence of a sync pattern before or at time t_s (t_s a whole integer) is given by:*

$$\mathbb{P} = 1 - (1-p)^{t_s} \tag{4.3}$$

with $p = 1/q^l$ where q is the cardinality of $\mathbb{F}$.

Proof 4.1 *The cryptogram symbols are uniformaly distributed and independent. Hence, considering a sync pattern of length l, the probability that it occurs is $p = 1/q^l$. Let E_i be the event "the sync pattern appears at time i" (with i a whole integer). Hence, the probability of E_0 is p. Let F_i be the event "the sync pattern appears before or at time i". Consequently, F_0 occurs if E_0 occurs, and then the probability of F_0 is p. Next, F_1 occurs if E_0 occurs or if E_1 occurs and E_0 does not occur. Hence, the probability of F_1 is $p + p(1-p)$. Following the same reasoning to the event F_{t_s} yields*

$$\begin{aligned}\mathbb{P} &= p + p(1-p) + p(1-p)^2 \cdots + p(1-p)^{t_s} \\ &= p\big(1 + (1-p) + (1-p)^2 \cdots + (1-p)^{t_s}\big) \\ &= p(1-(1-p)^{t_s})/p \\ &= 1 - (1-p)^{t_s}\end{aligned}$$

4.1.2 Proof-of-Concept Example

In addition to **Steps S0-S4** given in Section 3.4.2, let us consider **Step S5** that defines the hybrid aspect of the architecture.

Step S5: For defining the mode $\sigma(t) = 2$, the matrix A' is derived from the digraph $\mathcal{G}$ of mode 1 where an edge on the vertex of x^7 has been added. The vertex of x^7 is not essential. Consequently, **C2** is not fulfilled and the output $c_t = Cx_t$ is no longer flat. The new digraph $\mathcal{G}'$ is depicted on Figure 4.2.

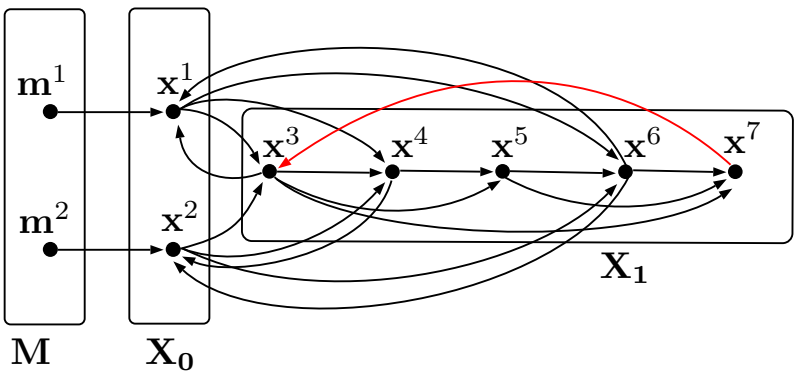
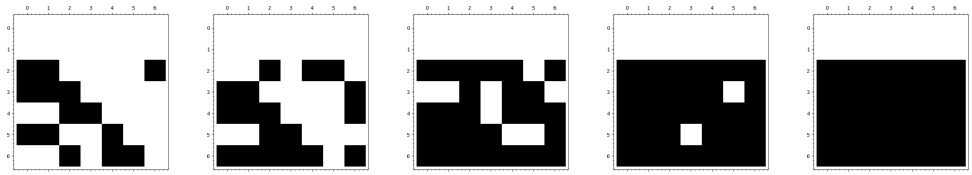


Figure 4.2: Digraph $\mathcal{G}'$ related to the new LTI system with relative degree $r = 1$. The red edge is added.

The matrices $A'_{\rho(t)}$ and $P'_{\rho(t:t+1)}$ are obtained and given below.

$$\underbrace{\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ \rho(t) & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \rho(t) & 0 & 0 & 0 \\ \rho(t) & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \rho(t) & 0 & 1 & \rho(t) & 0 \end{pmatrix}}_{A'_{\rho(t)}} \quad \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \rho(t) & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \rho(t) & 0 & 0 & 0 \\ \rho(t) & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \rho(t) & 0 & 1 & \rho(t) & 0 \end{pmatrix}}_{P'_{\rho(t:t+1)}}$$

Figure 4.3 gives the diffusion property in this new mode of operation. Clearly, the diffusion is now much better achieved.



P'_S P'^2_S P'^3_S P'^4_S P'^5_S

Figure 4.3: Powers of $P^{\mathbf{p}}_S$ for $\mathbf{p} \in [1, 5]$, $r = 1$. Black squares correspond to non zero entries while blank ones correspond to null entries.

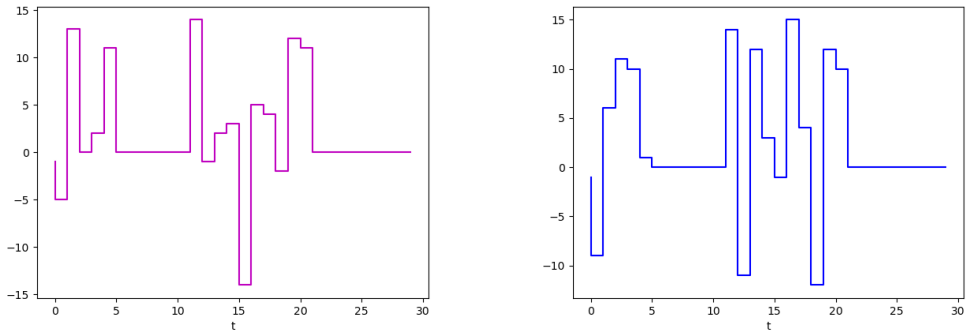


Figure 4.4: Time evolution of the error $m_t - \hat{m}_{t+1}$ for the first component (left) and right component (right).

The statistical SSSC has been implemented with . Figure 4.4 gives the result by depicting the error $m_t - \hat{m}_{t+1}$ for both components.
A time $t = 0$, the cipher and the decipher are in mode $\sigma(t) = 1$. As expected, the synchronization is performed after $t = 7$. Indeed, the flatness guarantees synchronization after a transient time no greater than $n = 7$. Then, for $t \geq 7$, the local supervisor of both the cipher and the decipher switches to mode $\sigma(t) = 2$. At $t = 12$, mimicking a disturbance (bit slip for example), the cipher c_t is corrupted by changing the value delivered by the cipher. As expected, that causes a desynchronization and no self-synchronization can occur since the mode $\sigma(t) = 2$ is not flat. The sync pattern appears at $t = 16$. Then, the local supervisor switches the cipher and the decipher in mode $\sigma(t) = 1$. After $t = 23$, the self-synchronization is again recovered. If no disturbance occurs, the synchronization error remains indefinitely at zero and the deciphering is achieved properly.
Finally, to illustrate the impact of the length of the sync pattern and the probability of synchronization, for arbitrary messages and arbitrary initial conditions, 2000 runs have been performed. The ratio between

the number of successful resynchronizations after a time t and the total number of runs has been reported in Figure 4.5 for a pattern's length l equal to 1, 2 and 3 respectively. As expected, the plots show that the more the length of the sync pattern, the more the time before resynchronization. Indeed, the probability that the sync pattern occurs decreases with respect to the length l. Besides, the plots tend towards 1 as the time t before synchronization tends towards infinity.

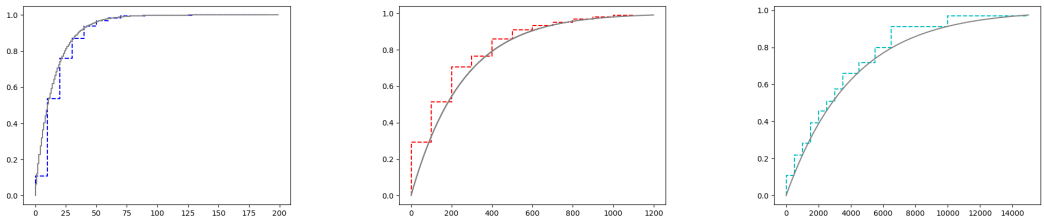


Figure 4.5: Theoretical probability of successful resynchronizations after a time t (solid line) and experimental result (dashed line) for time windows. Length $l = 1$ (left), length $l = 2$ (middle), length $l = 3$ (right).

The stairs are explained by the fact that the ratio between the number of successful resynchronizations after a time t and the total number of runs has been not calculated for every t and is kept constant between two time windows.
The next section deals with statistical SSSC based on switched linear systems. We show that dead-beat stabilizability, closely related to the problem of mortality[8]of a set of matrices, can be interesting to propose design methodologies leading to statistical self-synchronizing architectures. We improve an existing algorithm for building dead-beat stabilizable systems.

4.2 Design of a statistical SSSC based on switched Linear automata

Specific notation for this section: For a vector z_t indexed by the time $t \in \mathbb{N}$, z_t^i denotes its i-th component. Given $n \in \mathbb{N}$, define $\mathbb{N}_n = \{j \in \mathbb{N} : 1 \leq j \leq n\}$. The set of q switching modes is $\mathcal{I} = \mathbb{N}_q$ and the related matrices forms a finite collection $\mathcal{A} \subseteq \mathbb{R}^{n\times n}$, whose i-th element is denoted with A_i, i.e. $\mathcal{A} = \{A_i\}_{i\in\mathcal{I}}$, with $A_i \in \mathbb{R}^{n\times n}$ for all $i \in \mathcal{I}$. The mode at time t of a switched system is denoted $\sigma(t)$. All the possible sequences σ of modes of length N is $\mathcal{I}^N = \prod_{j=1}^{N} \mathcal{I}$, with $\mathcal{I}^N = \emptyset$ if $N = 0$. The length of a sequence σ is denoted by $|\sigma|$ and $|\sigma| = N$ if $\sigma \in \mathcal{I}^N$. Given σ, δ sequences of modes, (σ, δ) is their concatenation. Given $\sigma \in \mathcal{I}^N$, define $\mathbb{A}_\sigma = \prod_{j=1}^{N} A_{\sigma(j)} = A_{\sigma(N)} \cdots A_{\sigma(1)}$ and $\prod_{j=n_0}^{n_1} A_{\sigma(j)} = I_n$ if $n_0 > n_1$ where I_n stands for the identity matrix.
Hereafter, the automaton operates on a finite field $\mathbb{F}$. Necessary basics on finite field should be recalled. It is the purpose of the following subsection.

4.2.1 Basics on finite fields

A field is a 3-uplet $(\mathbb{F},+,\cdot)$ where $\mathbb{F}$ is a set, $+$ an operation usually called *addition* and $\cdot$ an operation usually called *multiplication*. The following properties apply

1. For all a and b in $\mathbb{F}$ both $a + b$ and $a \cdot b$ are $\mathbb{F}$,

[8]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.

2. For all a, b and c in $\mathbb{F}$, associativity holds: $a + (b + c) = (a + b) + c$ and $a \cdot (b \cdot c) = (a \cdot b) \cdot c$,

3. For all a and b in $\mathbb{F}$, commutativity holds: $a + b = b + a$ and $a \cdot b = b \cdot a$,

4. There exists an element of $\mathbb{F}$, called the additive identity element and denoted by 0 such that for all a in $\mathbb{F}$, $a+0 = 0+a = a$. Likewise, there is an element called the multiplicative identity element and denoted by 1, such that for all a in $\mathbb{F}$, $a \cdot 1 = a$. The identity elements 0 and 1 have to be different,

5. For every a in $\mathbb{F}$, there exists an element $-a$ in $\mathbb{F}$ such that $a + (-a) = 0$. Similarly, for any a in $\mathbb{F}$ other than 0, there exists an element a^{-1} in $\mathbb{F}$ such that $a \cdot a^{-1} = 1$,

6. For all a, b and c in $\mathbb{F}$ distributivity of the multiplication over the addition holds: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.

The number of elements of a finite field is called its order. A finite field of order Z exists if and only if Z is a prime power Q^m (where Q is a prime number and m is a positive integer). Thus, the field $\mathbb{F}_{Q^m} = GF(Z)$ (where GF stands for Galois Field) may be explicitly constructed in the following way. One first chooses an irreducible polynomial P in the ring of polynomials defined by the variable X, $GF(Q)[X]$ of degree m (such an irreducible polynomial always exists). Then, the elements of $GF(Z)$ are the polynomials over $GF(Q)$ whose degree is strictly less than m. The addition and the subtraction are those of polynomials over $GF(Q)$. The product of two elements is the remainder of the Euclidean division by P of the product in $GF(Q)[X]$. The multiplicative inverse of a non-zero element may be computed with the extended Euclidean algorithm.
For SSSC, it has been stressed that the state vector x_t of the automaton (2.17) must be expressed as a function of a finite number of its shifted outputs as described by Equation (2.18). In next section, the connection between the design of SSSC, that are dynamical systems having this specific property, and the construction of dead-beat stabilizable switched systems is brought out.

4.2.2 The connection between the properties of finite input memory automata and dead-beat stabilizability

Switched linear systems correspond to the Maiorana McFarland construction which has proved to produce functions that have many interesting cryptographic properties like high non linearities, high correlation immunity and good propagation characteristics (see [Car10] when considering Boolean functions defined over the two-element field). Hence, as a special case of the state transition function g_K of (2.17), it is interesting to consider the nonlinear automaton in the form

$$x_{t+1} = P_{\sigma(t)}x_t + R_{\sigma(t)}c_{t+r} \tag{4.4}$$

where $x_t \in \mathbb{F}^n$ is the state at time $t \in \mathbb{N}$ and σ is a switching rule which depends in a nonlinear way on a finite sequence of past ciphertexts $c_t, \ldots, c_{t-s}$ with $s \in \mathbb{N}$. Hence, at each time t, the switching function selects the mode by performing $\sigma(t) = \varphi(c_t, c_{t-1}, \cdots, c_{t-s})$ in $\mathcal{I}$. The n-dimensional square matrix $P_{\sigma(t)}$ belongs to a finite set of q matrices with entries in $\mathbb{F}$ and $R_{\sigma(t)}$ is a $(n \times m)$-dimensional vector with entries in $\mathbb{F}$. We consider the case when $r = 0$.

As a clue to tackle the design of statistical SSSCs, it is worthwhile realizing that the property of finite input memory with a statistical delay of memorization expressed by Equation (2.18) is obtained for (4.4) whenever there exists at least a switching sequence of length N so that $\prod_{i=1}^{N} P_{\sigma(t+i-1)} = 0$. It turns out that it is related to the property of dead-beat stabilizability. Indeed, let us consider the so-called auxiliary system of (4.4) defined by

$$x_{t+1} = A_{\sigma(t)}x_t \tag{4.5}$$

where $x_t \in \mathbb{F}^n$ is the state at time $t \in \mathbb{N}$ and $A_{\sigma(t)}$ is the n-dimensional square matrix that verifies $A_{\sigma(t)} = P_{\sigma(t)}$ for all $t \in \mathbb{N}$. The following definition is recalled from [FM19]. It means that the dynamical matrices of the auxiliary system coincide with the matrices of the decipher (4.4)

Definition 4.1 *The system (4.5) is dead-beat stabilizable if and only if*

$$\exists N \in \mathbb{N}, \quad \exists \gamma \in \mathcal{I}^N \quad \text{s.t.} \quad \mathbb{A}_\gamma = 0. \tag{4.6}$$

All in all, the automaton defined by Equation (4.4) has a finite input memory whenever the auxiliary system (4.5) is dead-beat stabilizable. Since the switching rule σ depends on past ciphertexts and that the ciphertexts are assumed to be random and uniformly distributed (a common feature of the cipher), the time before a stabilizing switching sequence appears is also random. As a result, the upper bound M of the memorization delay is a random variable. The resulting SSSC belongs to the class of statistical, what is precisely our objective.

4.2.3 Methodology to build a SSSC based on switched Linear automata

We recall a Necessary and Sufficient Condition for dead-beat stabilizability and the algorithm that allows to construct a dead-beat stabilizable system. Then, we highlight the pecularities due to the consideration of finite fields. The notion of orthogonality deserves a special treatment. Finally, we propose a code written in SageMath along with an illustrative example.

4.2.3.1 Dead-beat stabilizability and algorithm

Let us consider the systems described by Equation (4.5). For this system, let us introduce the following notation

$$\mathcal{I}_s = \{i \in \mathcal{I} : \ \det A_i = 0\}, \ \mathcal{I}_{ns} = \{i \in \mathcal{I} : \ \det A_i \neq 0\}, \tag{4.7}$$

the sets of modes of singular and nonsingular matrices A_i in $\mathcal{A}$ and by q_s and q_{ns} the number of elements of $\mathcal{I}_s$ and $\mathcal{I}_{ns}$, respectively. Clearly, one has $q_s + q_{ns} = q$ where it is recalled that q is the number of modes of the system (4.5).
The aim of this section is to illustrate how to build the set $\mathcal{A}$ of matrices such that condition (4.6) is satisfied for a given, desired $N \in \mathbb{N}$ but it is not for any $N' < N$. This means that the set $\mathcal{A}$ of matrices would allow to generate sequences γ with length greater than or equal to N such that $\mathbb{A}_\gamma = 0$, but not shorter ones, and then there exists only one sequence such that $\mathbb{A}_\gamma = 0$ among the q^N of length N at most. This property is useful to design a SSSC whose upper bound M of the memorization delay is a design parameter determined by q and N.
The main underlying idea of the proposed method relies on results from [FM17; FM19], in which it is proved that γ such that $\mathbb{A}_\gamma = 0$ exists, and hence (4.5) is dead-beat stabilizable, if and only if there is a set of $\tilde{m}$ switching sequences σ^p of finite length r^p, with $p \in \mathbb{N}_{\tilde{m}}$ and $1 \leq \tilde{m} \leq n$, whose last element is related to a singular matrix, and such that the intersection of the kernel of $\mathbb{A}_{\sigma^p}$ and the image of the product $\prod_{k=1}^{p-1} \mathbb{A}_{\sigma^k}$ of matrices has a dimension strictly greater than zero, that is:

$$\dim \Big(\mathrm{im} \Big(\prod_{k=1}^{p-1} \mathbb{A}_{\sigma^k} \Big) \cap \ker(\mathbb{A}_{\sigma^p}) \Big) > 0 \tag{4.8}$$

Denoting with $X_{p-1} \in \mathbb{R}^{n \times n-p+1}$ a basis matrix of $\prod_{k=1}^{p-1} \mathbb{A}_{\sigma^k}$, then the condition recalled above is equivalent to

$$\dim \ker \left(\mathbb{A}_{\sigma^p} X_{p-1} \right) = 1 \tag{4.9}$$

for all $p \in \mathbb{N}_n$, which implies that

$$\prod_{k=1}^{n} \mathbb{A}_{\sigma^k} = 0 \tag{4.10}$$

and then (4.6) holds with $\gamma = (\sigma^1, \ldots, \sigma^n)$ and $|\gamma| = N$. It can be also proved that $|\sigma^1| = 1$, see [FM17; FM19].
The proposed approach consists in generating a set $\mathcal{A}$ of matrices for which sequences σ^p exist such that (4.9) holds, and then also (4.10). Moreover, by choosing the length $r^p \in \mathbb{N}$ of the subsequences, i.e. such that $|\sigma^p| = r^p$, for $p \geq 2$, the desired length N of the sequence γ can be fixed since $|\gamma| = 1 + \sum_{p=2}^{n} r^p$ (let us recall that $|\sigma^1| = 1$).

Example 4.1 *The following example is to illustrate the necessary and sufficient condition* (4.9) *on a set of matrices such that* (4.6) *holds. Let us consider a set of matrices* $\{A_1, A_2, A_3\}$. *Let* $n = 4$, $q = 3$, $1 \in I_s$ *and* $(2, 3) \in I_{ns}$. *We set the singular matrix to be* A_1 *and the nonsingular matrices to be* A_2 *and* A_3.

Table 4.1 shows how $\dim\ker(\mathbb{A}_\sigma)$ *increases by 1 (*$\dim\mathrm{im}(\mathbb{A}_\sigma)$ *decreases by 1) when*

$$\mathbb{A}_\gamma = A_1A_2A_3A_3A_2A_1A_2A_3A_1A_2A_1 = 0$$

σ_i^j	σ_5^4	σ_4^4	σ_3^4	σ_2^4	σ_1^4	σ_3^3	σ_2^3	σ_1^3	σ_2^2	σ_1^2	σ_1^1	
$A_{\sigma_i^j}$	A_1	A_2	A_3	A_3	A_2	A_1	A_2	A_3	A_1	A_2	A_1	$\mathbf{1}_4$
$\dim\mathrm{im}(\mathbb{A}_\sigma)$	0	1	1	1	1	1	2	2	2	3	3	4
$\dim\ker(\mathbb{A}_\sigma)$	4	3	3	3	3	3	2	2	2	1	1	0
$\mathbb{A}_\sigma$	$\mathbb{A}_\sigma^4$					$\mathbb{A}_\sigma^3$			$\mathbb{A}_\sigma^2$		$\mathbb{A}_\sigma^1$	

Table 4.1: Sequence γ, subsequences, Image and Kernel dimensions.

In particular, the first subsequence has a fixed length of 1, the second, the third and the fourth ones have length 2, 3, 5 respectively i.e. $r^1 = 1$, $r^2 = 2$, $r^3 = 3$ *and* $r^4 = 5$. *Each sequence begins with a singular matrix* A_1, *completed by nonsingular matrices. If* $\gamma \in I^N$ *s.t.* $\mathbb{A}_\gamma = 0 \Leftrightarrow \dim\mathrm{im}(\mathbb{A}_\gamma) = 0$ *and* $\dim\ker(\mathbb{A}_\gamma) = n$.

The algorithm presented in [FM19] is recalled and commented hereafter, more detailed explanations can be found in the cited paper.
The initialization (lines 4-13) consists in randomly computing the q_s singular matrices and $q_{ns} - n + 1$ nonsingular ones. In particular the nonsingular matrices are given by $A_{ns} = T_s$ and the singular ones by $A_s = T_s^{-1}\Lambda_s T_s$ with:

(i) Λ_s a diagonal matrix whose diagonal has 0 as first entry and the other ones are randomly generated but non-null.

(ii) T_s a randomly generated invertible matrix.

The first iteration of Algorithm 1 (line 15) consists in choosing the only matrix composing σ^1 among the singular ones. Then, the algorithm builds a sequence of subsequences σ^p (lines 16-24) such that (4.9) holds. To this end:

(a) compute the random sequences α_p and β_p of modes related to nonsingular matrices and such that $|\alpha_p| + |\beta_p| = r^p - 2$ (line 17). Note that $\mathbb{A}_{\alpha_p}$ and $\mathbb{A}_{\beta_p}$ are nonsingular;

(b) select $s_p \in \mathcal{I}_s$, then A_{s_p} singular (line 18);

(c) define (line 19)

$$\begin{aligned} B_p &= \mathbb{A}_{\alpha_p}^{-1} T_{s_p}^{-1} R_p C_p^{-1} \\ C_p &= \left[\mathbb{A}_{\beta_p} X_{p-1} \quad (\mathbb{A}_{\beta_p} X_{p-1})^\perp\right] \end{aligned} \tag{4.11}$$

Algorithm 1 Build a dead-beat stabilizable system.

Input: q_s and q_{ns} cardinalities of $\mathcal{I}$ and $\mathcal{I}_s$, subsequences lengths r^p.

1: $\mathcal{I}_s \leftarrow \emptyset$ ▷ Initialization
2: $\mathcal{I}_{ns} \leftarrow \emptyset$
3: $\mathcal{A} \leftarrow \emptyset$
4: **for** $s \in \mathbb{N}_{q_s}$ **do** ▷ Generate $\mathcal{I}_s$
5: generate A_s singular ▷ Item (i)
6: insert s in $\mathcal{I}_s$
7: insert A_s in $\mathcal{A}$
8: **end for**
9: **for** $j \in \mathbb{N}_{q_{ns}-n+1}$ **do** ▷ Generate a part of $\mathcal{I}_{ns}$
10: generate A_{q_s+j} nonsingular ▷ Item (ii)
11: insert $q_s + j$ in $\mathcal{I}_{ns}$
12: insert A_{q_s+j} in $\mathcal{A}$
13: **end for**
14: random selection of $s_1 \in \mathcal{I}_s$ ▷ First step
15: $\sigma^1 \leftarrow s_1$
16: **for** $p \in \{i \in \mathbb{N} : 2 \leq i \leq n\}$ **do** ▷ p-th step
17: random selection of α_p and β_p ▷ Item (a)
18: random selection of $s_p \in \mathcal{I}_s$ ▷ Item (b)
19: compute B_p ▷ Item (c)
20: $A_{i_p} \leftarrow B_p$ ▷ Item (d)
21: insert i_p in $\mathcal{I}_{ns}$
22: insert A_{i_p} in $\mathcal{A}$
23: $\sigma^p \leftarrow (\beta_p, i_p, \alpha_p, s_p)$
24: **end for**

Output: $\mathcal{A}$ ▷ Matrix set of the stabilizable system

with

$$R_p = \left[\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \bar{R}_p & \\ 0 & & & \end{array}\right]$$

where $\bar{R}_p \in \mathbb{R}_p^{n-1\times n-1}$ is an arbitrary nonsingular matrix and where $V^\perp$ is a basis of the subspace orthogonal to the one spanned by the columns of V, implying the nonsingularity of C_p;

(d) define the new mode i_p such that $A_{i_p} = B_p$, define $\sigma^p = (\beta_p, i_p, \alpha_p, s_p)$ and include A_{i_p} in the matrix set $\mathcal{A}$ (lines 20-23). It is proved in [FM17] that $\dim\ker(A_{s_p}\mathbb{A}_{\alpha_p}B_p\mathbb{A}_{\beta_p}X_{p-1}) = 1$ and then (4.9) is satisfied with $\sigma^p = (\beta_p, i_p, \alpha_p, s_p)$.

4.2.3.2 The pecularities of finite fields

Generally speaking, if we have a vector space E over a field $\mathbb{F}$ and a bilinear form $B : E \times E \to \mathbb{F}$, it is said that v, w in E are orthogonal if $B(v, w) = 0$. For the special case $E = \mathbb{F}^n$, letting $v = (a_1, ..., a_n)$ and $w = (b_1, ..., b_n)$, the bilinear form is defined by $a_1b_1 + ... + a_nb_n$. Orthogonality is central when examining the construction of matrix C_p (see Equation (4.11)). Indeed, given both matrices $\mathbb{A}_{\beta_p}$ and X_{p-1}, we must build a basis of the subspace orthogonal to $\mathbb{A}_{\beta_p}X_{p-1}$. And yet, regarding orthogonality, some pecularities due to the fact that we do not consider the field of real numbers, should be highlighted. For example, it is possible for an element of the vector space E over a field $\mathbb{F}$ to be "orthogonal to itself". It means that the bilinear form may vanish *i.e.* $B(v, v) = 0$ for $v \neq 0$. As an example, let us consider the field $\mathbb{F}_2$. The field $\mathbb{F}_2$ is the basic field with 2 elements $\{0, 1\}$ with the two laws: the addition which is the exclusive or XOR (*i.e.* the addition modulo 2) and the multiplication which is the logical AND denoted

$\wedge$. It turns out that the vector $(1,0,1)^T$ is orthogonal to itself. Another pecularity should be highlighted and encompass the previous one. Let $E^{'\perp}$ be the subspace defined by

$$E^{'\perp} = \{v \in E | B(v,w) = 0 \text{ for all } w \in E'\}$$

If $E' \subset E$ is a k-dimensional subspace, then $E^{'\perp}$ is of dimension $n-k$. However, unlike in the field of real numbers, $(E, E^{'\perp})$ is not necessary a basis because it may happen that $E \cap E^{'\perp} \neq \emptyset$. In other words, elements of $E^{'\perp}$ may be linearly dependent from the basis of E while they are orthogonal to the elements of E. The fact that a vector can be orthogonal to itself enters such a situation. In our context, the consequence is that Algorithm 1 may fail when computing B_p at line 19. Indeed, C_p may not be invertible. As as example, let us consider again the field $\mathbb{F}_2$ and the matrices

$$\mathbb{A}_{\beta_p} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, X_{p-1} = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}^T$$

Then, we have that

$$\mathbb{A}_{\beta_p} X_{p-1} = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}^T$$

The vectors which are orthogonal to $(1,0,1)^T$ are all the vectors $(b_1,b_2,b_3)^T$ that fulfill $1b_1+0b_2+1b_3 = 0$. It yields the conditions $b_1 = b_3$ and b_2 arbitrary. Hence, the set of vectors orthogonal to $(1,0,1)^T$ is the space generated by $(1,0,1)^T$ and $(0,1,0)^T$. Consequently,

$$C_p = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

which is not invertible.

As another example, let us consider the matrices defined on the field $\mathbb{F}_4$

$$\mathbb{A}_{\beta_p} = \begin{pmatrix} 1 & 1 & \alpha+1 & 1 \\ 1 & \alpha & \alpha & 1 \\ \alpha+1 & 0 & \alpha & \alpha+1 \\ 0 & 1 & 1 & \alpha+1 \end{pmatrix}, \; X_{p-1} = \begin{pmatrix} 1 \\ \alpha \\ 0 \\ \alpha \end{pmatrix}$$

Then, we have that

$$\mathbb{A}_{\beta_p} X_{p-1} = \begin{pmatrix} 1 \\ 0 \\ \alpha \\ \alpha+1 \end{pmatrix}$$

The set of vectors orthogonal to $\mathbb{A}_{\beta_p} X_{p-1}$ is the space generated by:

$$(\mathbb{A}_{\beta_p} X_{p-1})^{\perp} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \alpha & 0 & \alpha+1 \end{pmatrix}$$

and

$$C_p = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \alpha & 0 & 0 & 1 \\ \alpha+1 & \alpha & 0 & \alpha+1 \end{pmatrix}$$

which is not invertible. Indeed, $v_1 = \alpha \cdot v_4 + v_2$ where v_i are the column vectors of C_p $(= 1, \ldots, 4)$.

It turns out that we can derive a necessary condition for Algorithm 1 to perform successfully at line 19. This condition is expressed as an inequality between the size of the field over which the SSSC operates and the dimension n of the system. It is based on a result borrowed from coding theory [MS77].
In coding theory, the purpose of an error-correcting code is to increase the ability to detect and correct transmission errors while not adding more overhead than necessary. A linear code takes a sequence of k symbols (the dimension of the code) and encodes it as a sequence of N symbols (the length of the code). These symbols come from an alphabet of size ℓ. Data are processed through a generator matrix G. Matrix G is a matrix whose rows form a basis for the linear code. It is a $k \times N$ matrix. Usually, error correcting codes are defined over finite fields such as $\mathbb{F}_{Q^m}$. The ability of a code to detect and correct errors is measured by its minimal distance d between code words. For the linear codes, the optimal minimal distance is reached by the codes called MDS (Maximum Distance Separable, see [MS77] for more details). For MDS codes, each extraction of the G matrix of square matrices is of full rank. This property is central for our purpose. Indeed, MDS codes could only be constructed on fields with a sufficient number of elements to ensure the full rank property [MS77]. This last remark guarantees an existence bound stated as a corollary given below.

Corollary 1 *The finite field $\mathbb{F}_{Q^m}$ over which the SSSC operates, considering that the matrix C_p is of size $k \times (N-k)$ with $N = 2k$, must fulfill $N - k \leq Q^m + 1$.*

Indeed, the standard form for a generator matrix G is $(I_k||H)$ where H is a $k \times (N-k)$ matrix ($||$ stands for the concatenation and it is recalled that I_k denotes the identity matrix of size k). Back to our context, consider that the matrix $(I_k||C_p)$ is a virtual generator matrix G. Since the size of G is $k \times N$ and the size of C_p is $n \times n$ (being n the dimension of the system), that amounts to setting $N = 2k$ and $k = n$. Hence, the inequality of Corollary 1 turns into

$$n \leq Q^m + 1 \tag{4.12}$$

It means that the size of the finite field $\mathbb{F}_{Q^m}$ must be chosen according to the dimension n of the system.

The SageMath code of Algorithm 1 is given in Appendix A. The inputs are the dimension n of the system, the number of singular and nonsingular matrices q_s and q_{ns} at line 40, α_p, β_p at lines 76-77. The outputs are the sequence γ at line 132 and the corresponding matrices at line 61.

4.2.3.3 Proof Of Concept Examples

We illustrate how to build an SSSC cipher and we show how it operates. All along this section, we will use the particular field $\mathbb{F}_4$ seen as the extension of degree 2 of the finite field $\mathbb{F}_2$. From this basic field, we construct $\mathbb{F}_4$ as a degree two extension, thus $\mathbb{F}_4$ is defined on $\mathbb{F}_2$ with the following polynomial of degree 2 with its coefficients over $\mathbb{F}_2$: $X^2 + X + 1$. Thus, we have $\mathbb{F}_4 = \mathbb{F}_2[X]/(X^2 + X + 1)$. An other way to describe $\mathbb{F}_4$ is to consider the root α of the polynomial $X^2 + X + 1$. Then, the elements of $\mathbb{F}_4$ are written with respect to α. More precisely, in this case, the elements of $\mathbb{F}_4$ are: (0,1, α, $\alpha+1$). All those elements have a degree strictly smaller than 2 as the definition polynomial is of degree 2.

Example 4.2 *First, we must build a switching linear automaton like* (4.4) *that acts as the cipher part of an SSSC. The dimension of the automaton is $n = 2$. With such a setting, the inequality* (4.12) *is verified since $n = 2$, $Q = 2$ and $m = 2$ and so $n = 2$ is less than $2^2 + 1 = 5$. Thus, invertible matrices C_p exists and Algorithm 1 should not fail at line 19.*

Next, we build a dead-beat stabilizable auxiliary system (4.5) *and so, a dead-beat stabilizable sequences of matrices A_i $(i = 1, \ldots, q)$. Then, we set $P_i = A_i$ $(i = 1, \ldots, q)$ for* (4.4)*. For simplicity, the matrices $R_{\sigma(t)}$ in* (4.4) *are all equal and set to the identity matrix.*
The switching rule σ is defined by a mere one-to-one correspondance between the cryptogram $c_t \in \mathbb{F}_4$ (four possible elements) and the mode $\sigma(t) \in \{1, 2, 3, 4\}$ $(q = 4)$.

Let us consider an instanciation of the automaton (4.4) *that reads*

$$\begin{cases} x_{t+1} = P_{\sigma(t)} x_t + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} c_t \\ z_t = x_t^1 + x_t^2 \end{cases} \tag{4.13}$$

The encryption function (2.12) *is defined as*

$$c_t = z_t + m_t \tag{4.14}$$

The equations of the decipher (2.21) *are exactly the same than* (4.13) *except the fact that the state vector* x_t *is replaced by the vector* $\hat{x}_t$*. The equations of the deciphering function* (2.13) *are also the same than* (4.14)*. Indeed, the addition and the substraction on the field* $\mathbb{F}_4$ *coincide one another.*
Let $\alpha_p = 1$*,* $\beta_p = 1$*,* $r^p = 4$ *for all* $p > 1$ *and* $r^1 = 1$ *by construction. The number of singular and nonsingular matrices are respectively* $q_s = 2$ *and* $q_{ns} = 2$*. The corresponding matrices are respectively* (A_0, A_1) *and* (A_2, A_3)*. As expected since inequality* (4.12) *is verified, Algorithm 1 performs successfully and gives*

$$A_0 = \begin{pmatrix} \alpha & \alpha \\ 1 & 1 \end{pmatrix}, A_1 = \begin{pmatrix} 0 & \alpha+1 \\ 0 & \alpha+1 \end{pmatrix}, A_2 = \begin{pmatrix} \alpha+1 & \alpha \\ \alpha & 1 \end{pmatrix}, A_3 = \begin{pmatrix} \alpha+1 & \alpha \\ 0 & 1 \end{pmatrix}$$

The sequence of modes γ *is* $\gamma = (\sigma^1, \sigma^2) = ((1), (2, 3, 2, 1))$*.*
A realization of a synchronization time plot is depicted on Figure 4.6, Figure 4.7 and Figure 4.8.

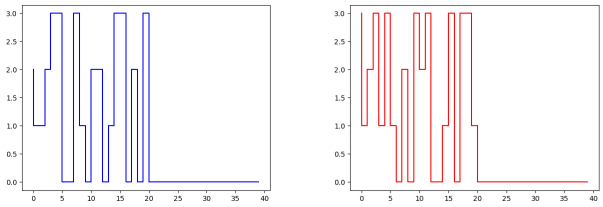


Figure 4.6: Time evolution of $x_t^1 - \hat{x}_t^1$ (left) and $x_t^2 - \hat{x}_t^2$ (right)

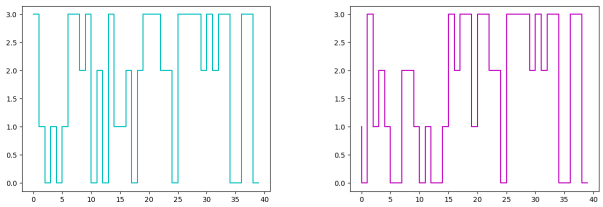


Figure 4.7: Time evolution of m_t^1 (left) and $\hat{m}_t^1$ (right)

From arbitrary initial conditions x_0 *and* $\hat{x}_0$*, a random sequence of plaintext symbols* $m_t \in \mathbb{F}_4^2$ *is generated. At* $t = 15$*, the dead-beat stabilizing sequence* γ *appears, causing the self-synchronization to succeed and so a proper decryption, after* $M = 20$*.*
For random messages and random initial conditions, 1000 *runs have been performed. The ratio between the number of successful resynchronizations after a time* t *and the total number of runs has been reported in Figure 4.9. As expected, the plot tends towards 1 as the time* t *before synchronization tends towards infinity. The stairs are explained by the fact that the ratio between the number of successful resynchronizations after a time* t *and the total number of runs has been not calculated for every* t *and is kept constant between two time windows.*

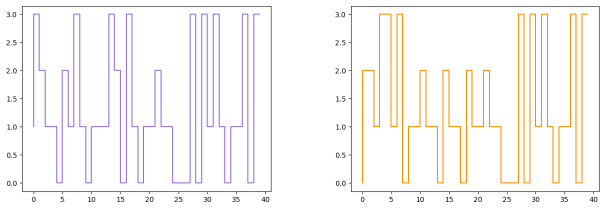


Figure 4.8: Time evolution of m_t^2 (left) and $\hat{m}_t^2$ (right)

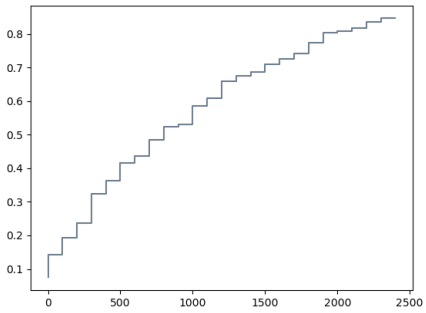


Figure 4.9: Experimental probability of successful resynchronizations after a time t.

Example 4.3 *Let us consider a second instanciation of the automaton* (4.4) *that reads*

$$\begin{cases} x_{t+1} = P_{\sigma(t)}\ x_t + \ I_4\ c_t \\ z_t = x_t^1 + x_t^2 \end{cases} \tag{4.15}$$

where I_4 is the square identity matrix of size 4. The encryption function (2.12) *is defined as*

$$c_t = z_t + m_t \tag{4.16}$$

The dimension of the automaton is $n = 4$. With such a setting, the inequality (4.12) *is verified since $n = 4$, $Q = 2$ and $m = 2$ and so $n = 4$ is less than $2^2 + 1 = 5$. Thus, invertible matrices C_p exists and Algorithm 1 should not fail at line 19.*

To build a mortal sequences of matrices, let $\alpha_p = 1$, $\beta_p = 2$, $r^p = 5$ for all $p > 1$ and $r^1 = 1$ by construction. The number of singular matrices is set to $q_s = 3$ and the corresponding matrices are (A_0, A_1, A_2). The number of nonsingular matrices is set to $q_{ns} = 5$ and the corresponding matrices are $(A_3, A_4, A_5, A_6, A_7)$. Algorithm 1 performs successfully and gives

$$A_0 = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & 1 & a+1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & a & a \end{pmatrix} A_1 = \begin{pmatrix} 1 & a+1 & a & a \\ 1 & a & a & a+1 \\ 0 & 0 & a+1 & a \\ a+1 & 0 & 1 & a+1 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} 0 & a+1 & a+1 & a+1 \\ a+1 & 0 & a & a \\ a+1 & 0 & 0 & 1 \\ 1 & 1 & 1 & a+1 \end{pmatrix} A_3 = \begin{pmatrix} a & a & 1 & 0 \\ a & 1 & a+1 & a+1 \\ 1 & a & a & 1 \\ 0 & 0 & a & a \end{pmatrix}$$

$$A_4 = \begin{pmatrix} a+1 & a & a & 0 \\ a+1 & 0 & 0 & a \\ 1 & 0 & a & 0 \\ a+1 & 0 & 1 & a+1 \end{pmatrix} A_5 = \begin{pmatrix} 1 & a+1 & 0 & a \\ a+1 & 0 & 1 & a \\ a & a+1 & a+1 & 0 \\ 0 & 0 & a+1 & 1 \end{pmatrix}$$

$$A_6 = \begin{pmatrix} 0 & a & a+1 & 1 \\ 0 & a+1 & a & 0 \\ a+1 & 1 & 1 & a \\ a & 0 & a & 1 \end{pmatrix} A_7 = \begin{pmatrix} 0 & 1 & a & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & a \\ a+1 & 1 & 1 & 1 \end{pmatrix}$$

The sequence of modes γ is $\gamma = (\sigma^1, \sigma^2, \sigma^3, \sigma^4) = ((2), (3, 3, 5, 3, 0), (5, 3, 6, 3, 0), (4, 6, 7, 4, 0))$.
A realization of a synchronization time plot is depicted on Figure 4.10 and Figure 4.11.

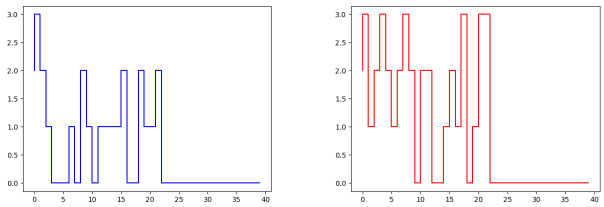


Figure 4.10: Time evolution of $x_t^1 - \hat{x}_t^1$ (left) and $x_t^2 - \hat{x}_t^2$ (right)

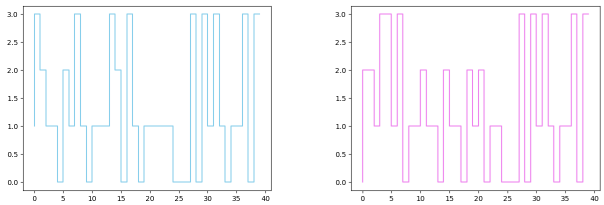


Figure 4.11: Time evolution of m_t (left) and $\hat{m}_t$ (right)

From arbitrary initial conditions x_0 and $\hat{x}_0$, a random sequence of plaintext symbols $m_t \in \mathbb{F}_4$ is generated. At $t = 17$, the dead-beat stabilizing sequence γ appears, causing the self-synchronization to succeed and so a proper decryption, after $M = 22$.
For random messages and random initial conditions, 1000 *runs have been performed. The ratio between the number of successful resynchronizations after a time t and the total number of runs has been reported in Figure 4.12. As expected, the plot tends towards 1 as the time t before synchronization tends towards infinity. The stairs are explained by the fact that the ratio between the number of successful resynchronizations after a time t and the total number of runs has been not calculated for every t and is kept constant between two time windows.*

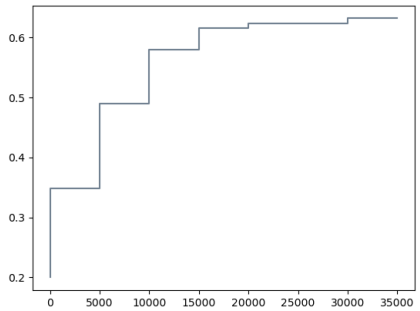


Figure 4.12: Experimental probability of successful resynchronizations after a time t.

4.3 Conclusion

In this chapter, we have first proposed a new architecture of statistical self-synchronizing stream ciphers. It involves an hybrid architecture with two modes of operations, each one governed by an LPV model exhibiting a nonlinear dynamics. It has been explained how the design problem can be recast as a switched system design under security constraints. It has been shown that control-theoretic concepts like flatness and structural analysis are relevant to this end. We have focused on a specific security criteria, that is diffusion. On the other hand, it has been shown how the concept of dead-beat stabilizability can yield an approach for the design of self-synchronizing stream ciphers involving switched linear systems. This Proof-Of-Concept clearly deserves further investigation to give a complete cipher. As most of the remaining issues enter the area of pure cryptography, they are not addressed here but the main lines are mentioned in this conclusion. For a real-world application, automata with higher dimension and larger sequences should be used. The design approach remains unchanged and still work. However, the more the length of the dead-beat stabilizing sequences, the longer the time before synchronization. Hence, the setting of the parameters should obey a trade-off. Besides, as usual in cryptography, security analysis must be performed. The way how to incorporate the secret key is also an important issue.

Part III

Evaluation of Feistel Structures and its Applications

Chapter 5

Introducing the FBCT: A Cryptanalysis Tool for Feistel constructions

Boomerang attacks date back to 1999, when David Wagner introduced them at FSE to break COCONUT98 [Wag99b]. When presented, this variant of differential attacks [BS91a] shook up the conventional thinking that consisted in believing that a cipher with only small probability differentials is secure. At Eurocrypt 2018, [Cid+18] introduced the Boomerang Connectivity Table (BCT), a tool to compute the probability of the middle round of a boomerang distinguisher from the description of the cipher's Sbox(es). Their new table and the following works led to a refined understanding of boomerangs, and resulted in a series of improved attacks. Still, these works only addressed the case of Substitution Permutation Networks, and completely left out the case of ciphers following a Feistel construction. In this chapter, we address this lack by introducing the `FBCT`, the Feistel counterpart of the BCT.

5.1 Cryptanalysis of block ciphers

Cryptanalysis of symmetric primitives has undergone considerable development as two powerful cryptanalytic techniques were published, in particular differential cryptanalysis by Biham and Shamir [BS91b] and linear cryptanalysis [TG91], [Mat93] by Matsui, followed by many variants. Differential and linear cryptanalysis are both statistical attacks. They exploit undesirable probabilistic properties of block ciphers. Meaning that they exploit statistical relations between the plaintexts and ciphertexts.

- In linear cryptanalysis, the bias is induced by linear approximations of the nonlinear round function involving bits of the plaintext, the ciphertext and the key that are verified with a probability that is significantly higher than for a random permutation.
- Differential cryptanalysis studies the propagation of differences in a cipher, exploiting the statistical bias observed in the output difference distribution of the cipher when the difference between two plaintexts is fixed. The variants of differential include: truncated differential cryptanalysis [Knu94], Related-key differential cryptanalysis [Bih94], impossible differential cryptanalysis [Knu98; BBS99], higher-order differential cryptanalysis [Knu94; Lai94] and Boomerang cryptanalysis [Wag99a]. Boomerang attack against Feistel networks is studied in Section 5.2.

5.1.1 Linear attacks

Linear attacks is a known plaintext attack on block ciphers introduced by Gilbert, Chassé and Tardy-Cordfir [GC90; TG91]. Firstly, it was successfully applied to the FEAL cipher by Matsui and a year later to DES [Mat93].

Linear attacks is based on finding affine approximations of the cipher, which hold with relatively high probability. The main idea is to find affine relations between the plaintext, the ciphertext and the secret key such that we can deduce information about some key bits.

$$\alpha \cdot m \oplus \beta \cdot e_K(m) = 0$$

where α and β are two n-tuples, also called masks. The pair $(\alpha \to \beta)$ is the linear approximation of the cipher.

Definition 5.1 *(Correlation of a linear approximation). Let $F : \mathbb{F}^{2n} \to \mathbb{F}^{2n}$ be a vectorial boolean function. Assume that the masks for input m and output $F(m)$ are α and β, respectively. The correlation of the linear approximation is defined as*

$$C(\alpha, \beta) = 2 \times Pr[\alpha \cdot m + \beta \cdot F(m) = 0] - 1$$

If F is a random permutation, i.e. an ideal cipher, we expect such relation $\alpha \cdot m \oplus \beta \cdot F(m) = 0$ to hold with probability $\frac{1}{2}$ for any pair of nonzero elements (α, β). For a keyed permutation, however, the bias of the linear approximation, defined by,

$$\epsilon(\alpha, \beta) = C(\alpha, \beta)/2$$

can be very high. Let $f = f_{n_r-1} \circ \cdots \circ f_1 \circ f_0$ be an iterated function. Linear approximations $(\gamma_i \to \gamma_{i+1})$ of a single round f_i can be concatenated into a linear trail $(\gamma_0 \to \gamma_1 \to \cdots \to \gamma_{n_r})$ whose correlation is estimated using the piling-up lemma.

Lemma 5.1 *(Piling-up lemma). Let $(\gamma_0 \to \gamma_1 \to \ldots \gamma_{n_r})$ be a linear trail of an iterated permutation. Then, the correlation of the linear trail can be computed as*

$$C(\gamma_0, \gamma_{n_r}) = \prod_{i=0}^{n_r-1} C(\gamma_i, \gamma_{i+1})$$

As in differential case, it is possible to combine several linear trails with masks α and β into a linear hull [Nyb] $(\alpha \to \beta)$. Extensions of linear cryptanalysis include the zero-correlation attack [BR14], which is based on linear approximations with a bias equal to zero.

5.1.2 Differential cryptanalysis and Variants

Differential cryptanalysis is a chosen plaintext attack. It was first introduced by Biham and Shamir [BS91b] who were the first to publish a differential attack against DES [BS92]. The attack was faster than exhaustive search.
In differential cryptanalysis, the main task is to analyze the propagation of differences that evolve through the various rounds of a cipher and investigate for the specific differences which propagate with high probability. The pairs of input-output differences can be used to obtain some bits of the secret key. The input-output difference pair is called a *differential*.

Let E be a block cipher operating on blocks of n bits using a key K of k bits and n_r rounds of a round function f. m and m' are two plaintext blocks, and X_i and X'_i their intermediate ciphertexts throughout the encryption process under K respectively. We denote them by $m = X_0, X_1, \ldots, X_{n_r-1}, X_{n_r} = E_K(m) = c$ and $m' = X'_0, X'_1, \ldots, X'_{n_r-1}, X'_{n_r} = E_K(m) = c'$.

Definition 5.2 *(Differential). A differential on t rounds of an iterated cipher E, with $t \leq n_r$, is a pair $(\delta_0, \delta_t) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ such that δ_0 is the input difference and δ_t is the output difference. Let E^t be a block cipher of t rounds.*

We recall that differential cryptanalysis aims at finding differentials which appear with high probability. The probability of a differential is given below.

Definition 5.3 *(Probability of a differential). The probability of a differential* $(\delta_0 \rightarrow \delta_t)$ *denoted by* $Pr[\delta_0 \rightarrow \delta_t]$*, is defined by:*

$$Pr[\delta_0 \rightarrow \delta_t] = Pr_{m,K}\Big(E_K^t(m) \oplus E_K^t(m \oplus \delta_0) = \delta_t\Big)$$

where $Pr_{m,K}$ *is the probability computed on all possible input plaintexts* $m \in \mathbb{F}_2^n$ *and all possible keys* $K \in \mathbb{F}_2^n$*.*

Computing the probability of a differential on t rounds needs to compute the probability on average over all keys and plaintexts. A simpler approach consists in studying *differential characteristics*, this can be evaluated by computing probabilities of differences for each round.

Definition 5.4 *(Differential characteristic). A differential characteristic on* t *rounds of a cipher* E *is a sequence* $(\delta_0, \delta_1, \ldots, \delta_t) \in (\mathbb{F}_2^n)^{t+1}$ *that specifies the input-output difference for each round.*

The probability of a differential characteristic is defined as follows.

Definition 5.5 *(Probability of a differential characteristic). The probability of a differential characteristic* $(\delta_0, \delta_1, \ldots, \delta_t) \in (\mathbb{F}_2^n)^{t+1}$ *is given by:*

$$Pr[\delta_0, \delta_1, \ldots, \delta_t] = Pr_{m,K}\Big(E_K^i(m) \oplus E_K^i(m \oplus \delta_i) = \delta_t\Big), \quad \forall i \in [1, t]$$

In order to study in a formal way differential cryptanalysis, we need a good probabilistic model of the block cipher. We assume that the cipher is a Markov one [LMM91], which implies that the probability of any characteristic is roughly the same for the large majority of keys.

Definition 5.6 *(Markov cipher). An iterated cipher with round function* f *is a Markov cipher if there is a group operation* $\otimes$ *for defining differences such that, for all choices of* $\alpha \neq 0$ *and* $\beta \neq 0$*,*

$$Pr[Y_0 \otimes Y_1 = \beta \mid m_0 \otimes m_1 = \alpha, m_0 = \gamma]$$

is independent of γ *when the subkey* k *is uniformly random.*

In an iterated Markov cipher, the differential probability of a differential characteristic is independent of the actual input of a given round, and the round subkeys are independent and uniformly distributed, the individual round probabilities are independent and may be computed as:

$$Pr[\delta_0, \delta_1, \ldots, \delta_t] = \prod_{i=0}^{t-1} p_i$$

where $p_i = Pr[\delta_i \rightarrow \delta_{i+1}]$ for $i \in \{0, \ldots, t-1\}$.
Conceptually, no real block cipher is a Markov cipher, since most modern design use a key-schedule algorithm to generate round subkeys out of the key. Thus, round subkeys are virtually always statistically dependent.

In an iterative block cipher, confusion is provided by a nonlinear layer such as S-box applications. An S-box is said to be active if the input difference is nonzero. In order to compute the probability of a differential through an S-box, the difference distribution table is used.
The difference distribution table (DDT) of an S-box is a table that lists the number of pairs that fulfill each possible input and output differences.

Definition 5.7 *(Difference Distribution Table). The difference distribution table of an* $n \times n$*-bit S-box is a matrix of size* $2^n \times 2^n$ *such that*

$$DDT[\alpha, \beta] = \#\{m \in \mathbb{F}_2^n | S(m \oplus \alpha) \oplus S(m) = \beta\}$$

The coefficient at row α, column β of this table is the number of inputs $m \in \mathbb{F}_2^n$ for which the difference α transitions to the difference β through the S-box. The corresponding probability is obtained by dividing this value by the total number of possible inputs m:

$$Pr[\alpha \xrightarrow[S]{} \beta] = \frac{DDT[\alpha, \beta]}{2^n}$$

When there is no difference between the two inputs of an S-box, this S-box is said to be passive and the differential transition occurs with probability 1. Therefore, in order to obtain a high-probability characteristic, an attacker has to minimize the number of active S-boxes with nonzero input differences and ensure that their probabilities of transition are as high as possible. This maximum probability of transition is derived from the differential uniformity of an S-box.

Definition 5.8 *(Differential uniformity). The differential uniformity of an S-box is the highest coefficient of its DDT, with the first row and the first column excluded.*

If an adversary performs an attack by finding a high probability differential then a distinguisher can be construct.

Definition 5.9 *(Distinguisher). A distinguisher is an algorithm that can decide with a probability higher than* $\frac{1}{2}$ *whether a set of plaintext-ciphertext pairs was generated by a random permutation or by a cipher.*

For a random permutation π of $\mathbb{F}_2^n$ and for any differences δ, $\Delta \in \mathbb{F}_2^n$

$$Pr[\pi(m + \delta) + \pi(m) = \Delta] = \frac{1}{2^n}$$

If one proposes a differential through t rounds (δ_0, δ_t) of e appearing with probability p, significantly greater than 2^{-n}, one will be able to distinguish the cipher from a random permutation.

Attacks on the last round

Differential cryptanalysis needs a $n_r - 1$ rounds differential characteristic of the cipher E denoted (δ, Δ) (see Figure 5.1) with a high probability p. Let K_{n_r} denote the key involved in the last round.

$$E_K = f_{K_{n_r-1}} \circ f_{K_{n_r-2}} \circ \cdots \circ f_{K_0}$$

where each round function f_{K_i} depends on a subkey K_i derived from the master key K. This distinguisher can be turned into an attack covering the full cipher as follows:

1. *data collection phase:* ask for the corresponding ciphertexts $(c_{0,i}, c_{1,i})$ of pairs of q messages $(m_{0,i}, m_{1,i})$ whose difference equals δ, for $i \in \{0, \ldots, q-1\}$;

2. *data analysis phase:*

   (a) guess the bits involved in the differential of the last round key K'_{n_r-1};

   (b) using the guessed subkey, decipher the last round and observe the resulting differences $f^{-1}_{K'_{n_r-1}}(c_{0,i}) \oplus f^{-1}_{K'_{n_r-1}}(c_{1,i})$

   (c) if the number of occurrences of Δ matches the theoretical bias (about $p \times q$), then the guess on K'_{n_r-1} is likely to be correct i.e. $K'_{n_r-1} = K_{n_r-1}$, else go back to step 2a.

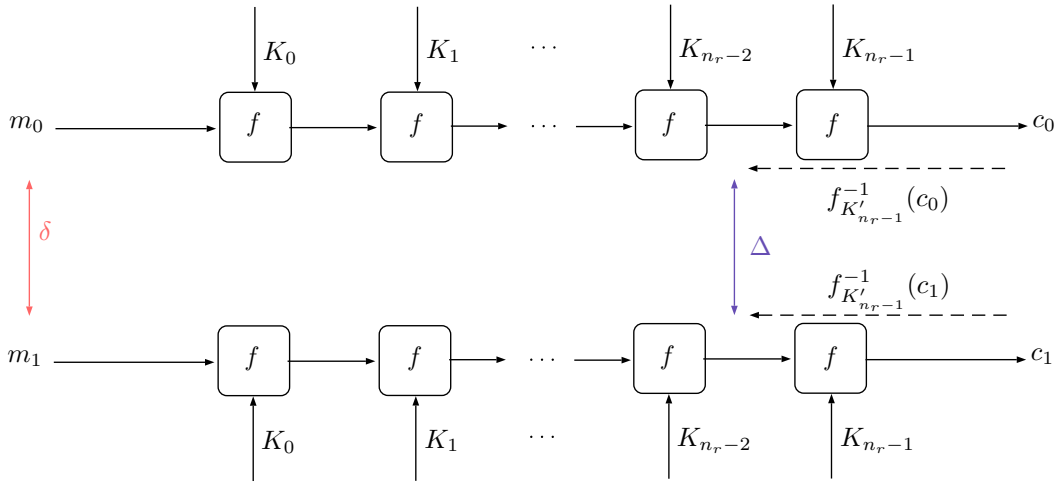


Figure 5.1: A differential attack on the last round of an iterated block cipher.

Truncated differentials

Introduced by Knudsen in [Knu94], truncated differential cryptanalysis is an extension of differential cryptanalysis in which not all the bits of the difference are fixed. It tends to be used on ciphers that appear to be resistant to classical differential cryptanalysis. The basic idea is to construct a partially known input/output differences where after a certain number of rounds only a part of the output difference is known.
Typically, these differentials predict that some parts of the output difference is zero, while other parts are nonzero (without more precision). Truncated differentials are efficient mainly against ciphers for which all the layers operate on well aligned blocks, for example transformations operating on bytes rather than on individual bits. Using binary permutations (that are optimal in hardware) is the adequate tool to provide resistance against it.

Related-key differential cryptanalysis

In 1993, Biham proposed to consider related-key differential attacks [Bih94], in which the attacker can not only know the encryption of two plaintexts m_0 and m_1 under the original key K but also under some derived keys K'. The relations between K and K' are known —typically $K' = K \oplus c$, for some constant c that is known to the attacker. Although Figure 5.2 shows the related-key setting for differential attack, this model can be applied to multiple cryptanalysis techniques.

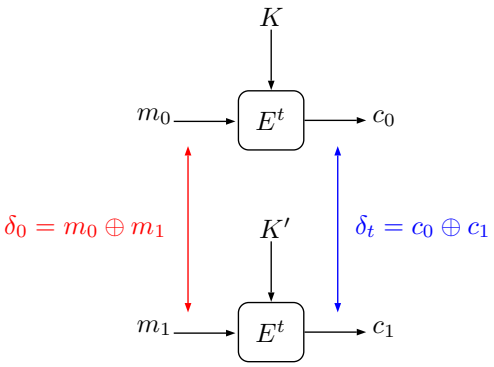


Figure 5.2: A related-key differential on t rounds of a cipher E.

Impossible differential cryptanalysis.

Impossible differential cryptanalysis was first introduced by Knudsen in [Knu98; BBS99]. Instead of approaching differentials that have high differential probability, impossible differentials bank on zero differential probabilities. The approach combines two differentials with probability 1 for the cipher at the top and at the bottom and uses a resulting conflict (miss-in-the-middle) when both directions are concatenated together.

Higher-order differential cryptanalysis.

Higher order differential cryptanalysis is considered to be a generalization to higher degrees [Knu94; Lai94] of differential cryptanalysis. Higher order differential cryptanalysis is usually used against ciphers that claims security against differential attack. Their aim is to establish derivatives on the differences considered on the round function to reduce the algebraic degree of f making it easy to analyze.

5.2 Boomerang attacks and introduction of the FBCT

Boomerang attack is an extension of differential cryptanalysis and is an adaptive chosen plaintext and ciphertext attack. In a basic version of Boomerang attack, the attacker does not try to cover the whole cipher with a unique highly-probable differential characteristic. Indeed, boomerang attacks make use of two small differentials covering half of the attacked rounds each, and can beat differential cryptanalysis when no high probability differential exists for the whole cipher.

In the basic form of the distinguisher, (represented on the left in Figure 5.3), the attacker has access to the encryption (E) and decryption (E^{-1}) oracles, and studies particular quartets of messages. First, she chooses m_0 and constructs $m_1 = m_0 \oplus \alpha$; using E, she obtains the corresponding ciphertexts c_0 and c_1 from which she deduces two additional ciphertexts by computing: $c_2 = c_0 \oplus \delta$ and $c_3 = c_1 \oplus \delta$. By calling the decryption oracle she retrieves the corresponding plaintexts m_2 and m_3 and then checks if $m_2 \oplus m_3 = \alpha$. A boomerang distinguisher is obtained if the probability that $m_2 \oplus m_3 = \alpha$ is higher for the cipher than for a random permutation.

In summary, a boomerang distinguisher is based on a couple of plaintext and ciphertext differences (α, δ) for which the following property among quartets of messages has a high probability:

$$E^{-1}(E(m_0) \oplus \delta) \oplus E^{-1}(E(m_0 \oplus \alpha) \oplus \delta) = \alpha.$$

In the original approach, the attacked cipher E is written as the composition of two sub-ciphers E_0 and E_1: $E = E_1 \circ E_0$. If for the sub-cipher E_0 the input difference α leads to the output difference β with probability p (and similarly γ leads to δ with probability q over E_1) the previous event was thought to have a probability of p^2q^2.

Following this breakthrough some variants were proposed including a related-key version [Kim+04; BDK05] and an impossible-differential one (see [CY09]). Improvements were also proposed on top of this, like a version that does not require access to the decryption oracle (named *amplified boomerang attack* [KKS01]) that was further developed into the so-called *rectangle attack* [BDK01].

The validity of boomerang attacks and in particular of the p^2q^2 formula were later questioned by Murphy [Mur11] with an example of distinguisher that seemed valid but was in fact of probability zero. The opposite phenomenon, that is distinguishers that happen with probability higher than what is expected, was also presented by Biryukov and Khovratovich in [BK09], and some particular cases (termed *Ladder Switch*, *Sbox Switch* and *Feistel Switch*) were explained.

All these observations were later formalized in a framework called *sandwich attack* [DKS10] for which the cipher is divided in 3 parts instead of 2, as represented on the right of Figure 5.3: a middle part E_m (termed *boomerang switch*) is introduced between E_0 and E_1. Dunkelman et al. applied this framework to KASUMI.

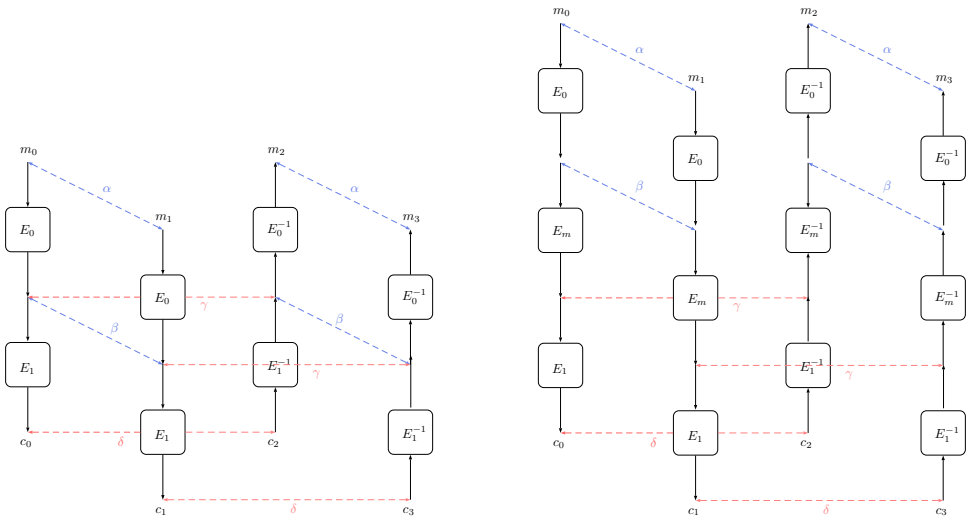


Figure 5.3: Configuration of the basic boomerang attack (left) and of the sandwich attack (right). Circled numbers correspond to a numbering that helps referencing states in the following discussions.

In [Cid+18] it has been introduced a tool called the *Boomerang Connectivity Table* that allows to easily evaluate the probability of the middle part E_m in the case where it covers one round and when the studied cipher follows an SPN construction. Their technique reduces the problem of computing the probability of the boomerang switch over one round function to the one of computing it over one Sbox only.

Equally as an Sbox with a Difference Distribution Table with small coefficients provides resistance against differential attacks, an Sbox with a Boomerang Connectivity Table (BCT) with small coefficients prevents an attacker from building efficient boomerang-style attacks. A study of some important families of Sboxes has been made in [BC18], [BPT19] and [Li+19], just to cite a few. Another interesting line of work that followed the paper of [Cid+18]. is the determination of the probability of a boomerang switch E_m that covers more than one round, and that was addressed for SPN ciphers in [WP19; SQH19].

Still, to the best of our knowledge a similar analysis has not been provided yet for Feistel constructions [Fei74], while it cannot be denied that it is an equally important type of block cipher design, instantiated for instance by the widely used 3-DES and by CLEFIA [Shi+07] (ISO/IEC 29192-2).

Our Contributions. We address this lack and investigate what can be said on a boomerang switch when the studied cipher follows a Feistel construction. In case the Feistel round function contains some affine layers and a single Sbox layer we introduce the `FBCT`, the Feistel counterpart of the Boomerang Connectivity Table and show that it is related to the second order derivative of the Sbox at play. Our model elucidates the last switch that is not explained by the BCT by showing that the Feistel Switch corresponds to the diagonal of our table.

We study the properties of the `FBCT` for some categories of cryptographic Sboxes (in particular APN Sboxes and Sboxes based on the inverse mapping) and investigate if the maximum in the `FBCT` is invariant for Sboxes that are in the same equivalence classes for an equivalence that is affine, extended-affine and CCZ [CCZ98].

In a bottom-up approach, we start from this notion of `FBCT` (that covers switches of one round) and then introduce the `FBDT` to deal with a 2-round switch and finally propose the `FBET` that treats the case of an arbitrary number of rounds. We explain the relation between all these new notions and give examples of their application.

Finally, we illustrate our approach by applying it to the cipher `LBlock-s` (used in the CAESAR candidate `LAC`), and provide a 16-round distinguisher which probability is evaluated to be higher than $2^{-56.14}$.

5.3 Conclusion

We presented the appropriate notion of `FBCT` and gave its main properties and relations with other established cryptographic tables. We provided a rather simple expression of the probability of a boomerang switch occurring over two rounds and a more complex expression for probabilities over multiple rounds. The motivations for the contribution and further details can be found in [Bou+20].

Conclusion

This thesis aimed at bringing a connection between encryption purposes and control theory. It has been motivated by the unprecedented need for ensuring security of data and systems like Cyber Physical Systems. Likely to the timely investigation of encrypted control in the literature, this thesis highlighted the interest of using automatic control concepts to tackle other issues borrowed from cryptography. A framework for the design of dedicated Self-Synchronizing Stream Ciphers has been proposed. The design is recast as control theoretical issues. It is based on left inversion, flatness, automata admitting a matrix representation like LPV models, and structural analysis. As a main contribution from an automatic control point of view, it has been proposed a systematic methodology based on digraph considerations to construct flat LPV automata. From a cryptographic point of view, the interest of such a framework lies in that a broader class of self-synchronizing automata can be proposed compared to the ones obtained from the fully-fledged Maurer's principle that is restricted to T-functions. Furthermore, the hybrid architecture, not included in the Maurer's framework, has been motivated by a trade-off between computational complexity and security. A large portion of these works are described in Part II.

Summary of the content of the chapters

Chapter 1 and Chapter 2 are mainly devoted to some necessary recalls form control theory and cryptography respectively. They facilitate the understanding of the contributions detailed in Chapter 3, Chapter 4 and Chapter 5.

Chapter 1. Background on LPV dynamical systems along with illustrations have been recalled. As a particular realization of a structured LTI system, it has been shown how an LPV system can be described in terms of graphs. Then, graphical conditions given in Theorem 1.3 and Theorem 1.4 have been recalled to characterize with polynomial complexity the flat outputs of an LPV SISO or MIMO LPV system. Those conditions are convenient, as seen in Part II, to construct digraphs that yield flat LPV systems and hence, to design Self-synchronizing Stream Ciphers (Chapter 3 and Chapter 4).

Chapter 2. In this chapter, generalities on symmetric cryptography mainly on stream ciphers have been recalled. The different architecture related to Self-Synchronizing Stream Ciphers was detailed. The cryptanalysis devoted to the security evaluation of cryptographic primitives was discussed. The general principles of the known attacks like algebraic attacks, differential attacks and Linear cryptanalysis have been given. This chapter allowed to explain how the construction of dynamical systems, having some specific property, can be used to construct a specific class of SSSC.

Chapter 3. We have provided a new methodology borrowed from graph and control theory to construct a family of deterministic Self-Synchronizing Stream Ciphers that possibly yield non-triangular state transition functions. The SSSCs involve Multiple Input Multiple Output LPV automata to accelerate the encryption process. The issue of a trade-off between synchronization, complexity and security have been analyzed. If the computational load is neglected, security regarding the consideration on the T function is guaranteed. Indeed, with a dynamical system admitting an inherent delay greater than 1, it has been shown that non triangular state transitions function can be obtained. However, reducing the computation to its lowest cost (inherent delay equal to 1) necessarily leads to state transition functions conjugate with T-functions and is a limitation as far as security is concerned. In all cases, the diffusion is shown bad

but is inherent to a deterministic SSSC. To deal with the problem, a hybrid architecture that leads to the concept of statistical SSSC has been proposed in next chapter.

Chapter 4. We have first proposed a new architecture of statistical self-synchronizing stream ciphers. It involves an hybrid architecture with two modes of operations, each one governed by an LPV model exhibiting a nonlinear dynamics. It has been explained how the design problem can be recast as a switched system design under security constraints. It has been shown that control-theoretic concepts like flatness and structural analysis are still relevant to this end. We have focused on the specific security criteria, that is diffusion. It turns out that with such an architecture, the diffusion has been clearly improved while the computational complexity is kept reduced. On the other hand, it has been shown how the concept of dead-beat stabilizability can yield an approach for the design of self-synchronizing stream ciphers involving switched linear systems. Security is left to future work. In particular, the way how to incorporate the secret key is also an important issue.

Chapter 5. This chapter studied the computation of the probability of a boomerang switch for Feistel ciphers, a problem that had not been addressed so far in spite of the importance of such constructions. The counterpart of the BCT, namely the FBCT was introduced. The main properties of FBCT and its relations with other cryptographic tables have been established. We also showed how to extend a boomerang switch to an arbitrary number of rounds and provided a generic formula to compute the corresponding probability. However, applying this formula over many rounds quickly requires too much computational power. The FBCT is still a useful tool in finding S-boxes with the best properties or that are the most likely to lead to incompatibilities for a Feistel cipher. This chapter presented the work done for the master's program, with some finalizations done at the start of the thesis. This is an overview; further details are in [Bou+20]. Indeed, dilation in a digraph determines the structural controllability in the corresponding structural system.

Further research directions

Future work i) The property of Controllability of the LPV system must be investigated. Controllability is a fundamental concept in dynamical systems [Lue79]. It corresponds to the ability to steer a state to an arbitrary final state in finite time from an arbitrary initial one. The system is not controllable means that there is a part of the state space that will not be visited. The property of controllability of LTI systems can be determined by the rank of its controllability matrix (usual Kalman criterion). However, in our context, controllability must be fulfilled from a structural point of view with the same motivation as flatness. That is, the property must hold for any parametrization of the LTI system in terms of the secret key and so requires the structural approach. It comes down to verifying whether the proposed construction (see Chapter 3) that yieds LPV flat automata and so, self-Synchronzing Stream Ciphers involve the so-called dilations in the corresponding digraph [Chi74].

Definition 5.10 *(Dilation). Let Σ_Λ be a structured linear system and $\mathcal{G}(\Sigma_\Lambda)$ its associated digraph. We consider S a set of k_S vertices in the vertex set of the digraph not containing the input. Denote by $T(S)$ the set of vertices $\mathbf{v^i}$ for $i = 1, 2, \ldots, k_T$ such that there is an oriented edge from $\mathbf{v^i}$ to $\mathbf{v^j}$ where $\mathbf{v^j} \in S$. S is a dilation if:*

$$k_S - k_T = d_S > 0$$

d_S is called the dilation of S.

Dilation represents a set of vertices that is driven directly by another set smaller in number of elements. In this case, the system is not controllable.

Example 5.1 *Consider here the set $S = \{\mathbf{v^1}, \mathbf{v^2}, \mathbf{v^3}\}$ with $k_S = 3$. Determine the set $T(S)$ containing all the $\mathbf{v^j}$ with the property that there is an oriented edge going from $\mathbf{v^j}$ to S. Clearly, $T(S) = \{\mathbf{v^2}, \mathbf{v^4}\}$ with $k_S = 2$.*

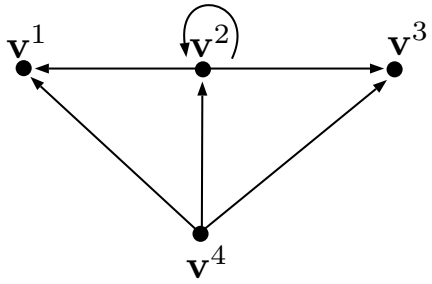


Figure 4: Example of digraph containing dilation

Since $k_S - k_T = 1 > 0$, it means that the digraph contains a dilation.

Future work ii) An instantiation of a dedicated Self-Synchronizing Stream Cipher (SSSC) involving an automaton with finite input memory using non-triangular state transition functions must be proposed. The instantiation admits a matrix representation deduced from a general and systematic methodology proposed in Chapter 3. To get a complete cipher, the Key Schedule, the design rationale and the security analysis must be provided. The choices of the field on which the cryptosystem operates, the dimension n of the internal state, the inherent delay r and the structure of the matrices A and P must be justified. Indeed, it is necessary to choose the number of non-zero elements of the matrices A and P. Beyond the number of non-zero elements, it is also necessary to choose their location (row and column). It must be decided whether a non-zero entry will be 1 or will correspond to an S-box. All these questions must consider several criteria regarding security, in particular good resistance to classical attacks like Time-memory-data trade-off attacks, Guess-and-Determine attacks, Differential / Linear Cryptanalysis, Algebraic Attacks, Cube Attacks given in Chapter 2 and good diffusion delay, while satisfying a trade-off regarding computational complexity for implementation concern. Finally, hardware implementation results must be performed and compared with some state-of-the art competitors. All these analyses must be done following the same procedure as in Stanislas [Fra+20] that was only dedicated to SISO LPV systems.

Appendix A

Algorithm to build a dead-beat stabilizable system

```
T_inverse = []
#Generate the singular matrice
def random_singular_matrix(field, size):

    test = True
    while(test == True):
        A = random_matrix(field, size)
        v_p = []
        vp = A.eigenspaces_left(format='galois')
        for i in range(len(vp)):
            v_p.append(vp[i][0])
        if A.rank() == size:
            test = 0 in v_p


    B = matrix(field, 1, 1, [0])
    A = block_matrix([ [B, 0], [0, A] ], subdivide = False)


    T = random_matrix(field , size + 1 , size + 1)

    while T.rank() != size + 1:
        T = random_matrix(field , size + 1 , size + 1)

    A = T^(-1) * A * T

    T_inverse.append(T^(-1))
    return A

#Generate the nonsingular matrice
def random_nonsingular_matrix(field, size):
     test = True
     while(test == True):
         A = random_matrix(field, size)
         test = A.is_singular()

     return A


def stabilizable_system(field, size, q_s, q_ns):

    I_s  = []
    I_ns = []
    A_s  = []
    A_ns = []
```

```
46
47      sigma_p = []
48
49      #Generate I_s
50      for s in range(q_s):
51          A = random_singular_matrix(field, size - 1)
52          A_s.append(A)
53          I_s.append(s)
54
55      #Generate a part of I_ns
56      for j in range(q_ns - size + 1):
57          A = random_nonsingular_matrix(field, size)
58          A_ns.append(A)
59          I_ns.append(j + q_s)
60
61      A = A_s + A_ns
62
63      #The first subsequence, of one element, corresponds to a singular matrix
64      s0 = choice(I_s)
65
66      sigma_p = []
67      sigma_p.append([s0])
68
69      stable = [A[s0]]
70
71      x_im = [A[s0]]
72
73      for k in range(1, size):
74
75          #Compute the random sequences a_p and b_p
76          ap = [choice(list(I_ns)) for i in range(1)]
77          bp = [choice(list(I_ns)) for i in range(1)]
78
79          #select s_p in I_s
80          sp = choice(I_s)
81
82          #Compute A_ap
83          A_ap = ones_matrix(field, size)
84          A_ap = A[ap[-1]]
85
86          for i in reversed(ap[:-1]):
87              A_ap = A_ap * A[i]
88
89          #Compute A_bp
90          A_bp = ones_matrix(field, size)
91          A_bp = A[bp[-1]]
92
93          for i in reversed(bp[:-1]):
94              A_bp = A_bp * A[i]
95
96          #Compute R_p
97          R = random_nonsingular_matrix(field, size - 1)
98          B = matrix(field, 1, 1, [1])
99          R_p = block_matrix([ [B, 0], [0, R] ], subdivide = False)
100
101         #Basis matrix
102         if (k == 1):
103             A_p = A[s0]
104             s = A_p.column_space()
105             X = s.basis_matrix()
106
107         if (k >= 2):
108             A_p = A[sigma_p[-1][-1]]
109             for i in reversed((sigma_p[-1][:-1])):
110                 A_p =  A_p * A[i]
111
112             x_im.append(A_p)
```

```
113             for i in reversed(x_im[:-1]):
114                 A_p =  A_p * i
115
116             v = A_p.image()
117             s = A_p.column_space()
118             X = s.basis_matrix()
119
120
121         #Compute C_p
122         Cp = matrix(field, 0, size)
123         Cp = block_matrix([[A_bp * X.transpose(), ((A_bp * X.transpose()).kernel()).
    matrix().transpose()]], subdivide = False)
124
125         #Compute B_p
126         B_p = A_ap^(-1) * T_inverse[sp] * R_p * Cp^(-1)
127         A.append(B_p)
128
129         I_ns.append(1 + I_ns[-1])
130
131         #Compute sigma_p
132         sigma_p.append(bp + [I_ns[-1]] + ap + [sp])
```

Appendix B

Illustrative examples

An instantiation of a dedicated deterministic Self-Synchronizing Stream Cipher (SSSC), introduced in [BMM22], involving an LPV automaton MIMO has been implemented in a platform that involves Arduino MEGA cards as shown by Picture B.1. Information to be encrypted can be symbols from a text captured by a touchscreen or digitalized signals delivered by a temperature sensor. Each card can encrypt or decrypt data according to the source of information that is used and the channel throughout encrypted data is conveyed. The Arduino MEGA cards involve an ATMEGA 2560 processor. It has a 16 MHz Clock Speed and a 256 KB Flash Memory. Self-synchronization has been highlighted in this hardware environment.

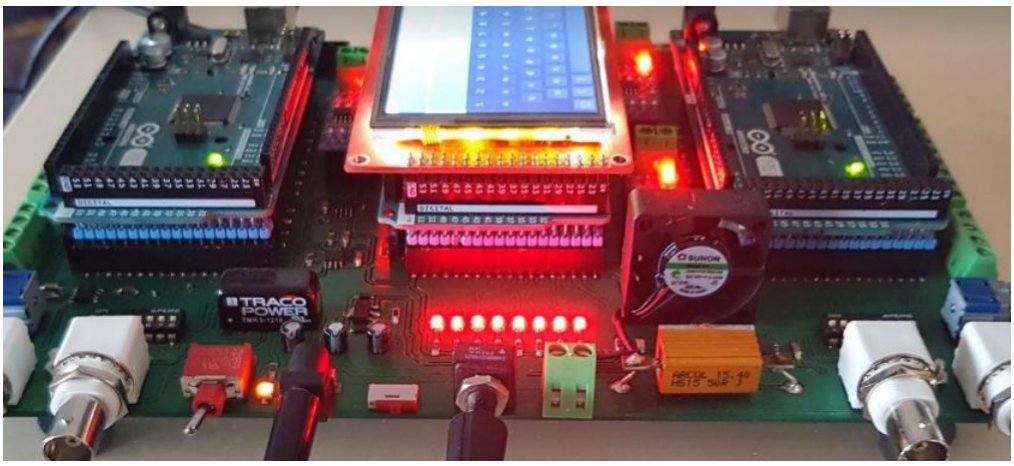


Figure B.1: Secure data exchange between Arduino MEGA cards

Figure B.2 shows the local supervisor of the setup connecting the cards together. Card 1 is connected to temperature sensors gathering the data in plaintext data. The data is then encrypted by the card 1 and sent over to card 2, which is responsible for decrypting 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 desynchronization.

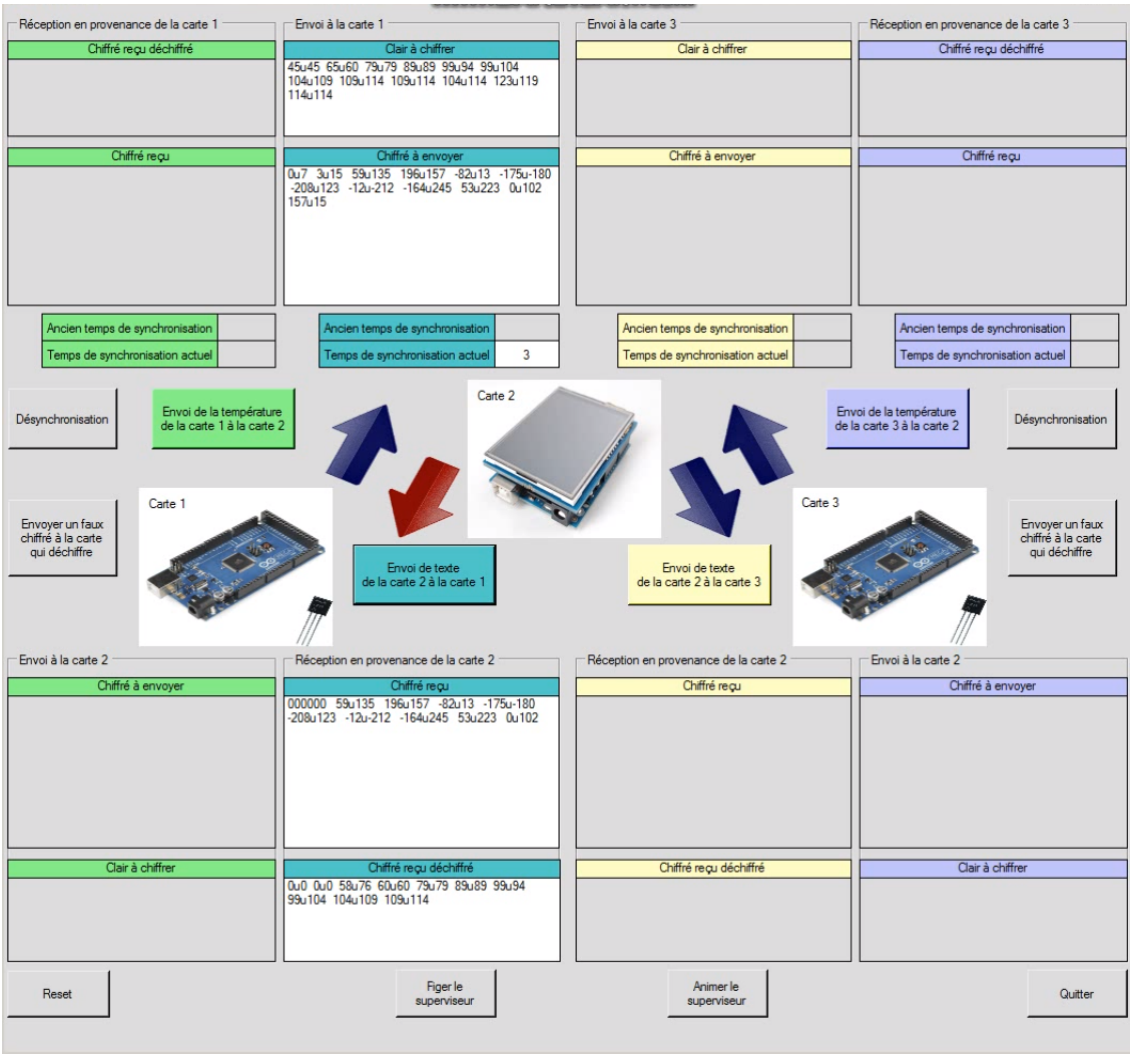


Figure B.2: Supervisor

Bibliography

[Sha49] Claude E Shannon. "Communication theory of secrecy systems". In: *The Bell system technical journal* 28.4 (1949), pp. 656–715.

[SM69] M. K. Sain and J. L. Massey. "Invertibility of Linear Time-Invariant Dynamical Systems". In: *IEEE Trans. Automatic Control* 14 (1969), pp. 141–149.

[Sil69] L. M. Silverman. "Inversion of Multivariable Linear Systems". In: *IEEE Trans. Automatic Control* 14.3 (June 1969), pp. 270–276.

[Chi74] Ching-Tai Lin. "Structural controllability". In: *IEEE Transactions on Automatic Control* 19.3 (1974), pp. 201–208.

[Fei74] Horst Feistel. *Block cipher cryptographic system*. US Patent 3,798,359. Mar. 1974.

[Hén76] Michel Hénon. "A two-dimensional mapping with a strange attractor". In: *The theory of chaotic attractors*. Springer, 1976, pp. 94–102.

[WM76] Diffie W. and Hellman M. "New directions in cryptography". In: *IEEE Trans. on Information Theory* 22.6 (1976), pp. 644–654.

[MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error correcting codes*. Vol. 16. Elsevier, 1977.

[Sta77] Data Encryption Standard. "National Bureau of Standards, NBS FIPS PUB 46". In: *US Department of Commerce* (1977).

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[Lue79] David G Luenberger. *Introduction to dynamic systems; theory, models, and applications*. Tech. rep. 1979.

[Hel80] Martin Hellman. "A cryptanalytic time-memory trade-off". In: *IEEE transactions on Information Theory* 26.4 (1980), pp. 401–406.

[Sin81] S. N. Singh. "A modified algorithm for invertibility in nonlinear systems". In: *IEEE Trans. Automat. Control* 26 (1981), pp. 595–598.

[CW82] Don Coppersmith and Shmuel Winograd. "On the asymptotic complexity of matrix multiplication". In: *SIAM Journal on Computing* 11.3 (1982), pp. 472–492.

[CW87] Don Coppersmith and Shmuel Winograd. "Matrix multiplication via arithmetic progressions". In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. 1987, pp. 1–6.

[GC90] Henri Gilbert and Guy Chassé. "A statistical attack of the Feal-8 cryptosystem". In: *Conference on the Theory and Application of Cryptography*. Springer. 1990, pp. 22–33.

[SB90] S.M. Shahruz and S. Behtash. "Design of controllers for linear parameter-varying systems by the gain scheduling technique". In: *29th IEEE Conference on Decision and Control*. 1990, 2490–2491 vol.4. DOI: 10.1109/CDC.1990.204074.

[BS91a] Eli Biham and Adi Shamir. "Differential Cryptanalysis of DES-like Cryptosystems". In: *CRYPTO'90*. Ed. by Alfred J. Menezes and Scott A. Vanstone. Vol. 537. LNCS. Springer, Heidelberg, Aug. 1991, pp. 2–21. DOI: 10.1007/3-540-38424-3_1.

[BS91b] Eli Biham and Adi Shamir. "Differential cryptanalysis of DES-like cryptosystems". In: *Journal of CRYPTOLOGY* 4.1 (1991), pp. 3–72.

[LMM91] Xuejia Lai, James L Massey, and Sean Murphy. "Markov ciphers and differential cryptanalysis". In: *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer. 1991, pp. 17–38.

[Mau91] U. M. Maurer. "New approaches to the design of self-synchronizing stream cipher". In: *Advance in Cryptography, In Proc. Eurocrypt '91, Lecture Notes in Computer Science* (1991), pp. 548–471.

[TG91] Anne Tardy-Corfdir and Henri Gilbert. "A known plaintext attack of FEAL-4 and FEAL-6". In: *Annual International Cryptology Conference*. Springer. 1991, pp. 172–182.

[BS92] Eli Biham and Adi Shamir. "Differential cryptanalysis of the full 16-round DES". In: *Annual international cryptology conference*. Springer. 1992, pp. 487–496.

[DGV92] 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*. Vol. 1. Singapore, Nov. 1992, pp. 279–283.

[Mat93] Mitsuru Matsui. "Linear cryptanalysis method for DES cipher". In: *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer. 1993, pp. 386–397.

[Bih94] Eli Biham. "New Types of Cryptanalytic Attacks Using related Keys (Extended Abstract)". In: *EUROCRYPT'93*. Ed. by Tor Helleseth. Vol. 765. LNCS. Springer, Heidelberg, May 1994, pp. 398–409. DOI: 10.1007/3-540-48285-7_34.

[Gol94] Jovan Dj Golić. "Linear cryptanalysis of stream ciphers". In: *International Workshop on Fast Software Encryption*. Springer. 1994, pp. 154–169.

[Knu94] Lars R Knudsen. "Truncated and higher order differentials". In: *International Workshop on Fast Software Encryption*. Springer. 1994, pp. 196–211.

[Lai94] Xuejia Lai. "Higher order derivatives and differential cryptanalysis". In: *Communications and cryptography*. Springer, 1994, pp. 227–233.

[Fli+95] Michel Fliess et al. "Flatness and defect of non-linear systems: introductory theory and examples". In: *International journal of control* 61.6 (1995), pp. 1327–1361.

[MOV96] A. J. Menezes, P. C. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Oct. 1996.

[SK96] Bruce Schneier and John Kelsey. "Unbalanced Feistel networks and block cipher design". In: *International Workshop on Fast Software Encryption*. Springer. 1996, pp. 121–144.

[Mat97] Mitsuru Matsui. "New block encryption algorithm MISTY". In: *International Workshop on Fast Software Encryption*. Springer. 1997, pp. 54–68.

[CCZ98] Claude Carlet, Pascale Charpin, and Victor Zinoviev. "Codes, bent functions and permutations suitable for DES-like cryptosystems". In: *Designs, Codes and Cryptography* 15 (1998), pp. 125–156.

[Knu98] Lars Knudsen. "DEAL-a 128-bit block cipher". In: *complexity* 258.2 (1998), p. 216.

[BBS99] Eli Biham, Alex Biryukov, and Adi Shamir. "Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 1999, pp. 12–23.

[DR99] Joan Daemen and Vincent Rijmen. "AES proposal: Rijndael". In: (1999).

[JR99] Oliver Jung and Christoph Ruland. "Encryption with Statistical Self-Synchronization in Synchronous Broadband Networks". In: *Cryptographic Hardware and Embedded Systems*. Ed. by Çetin K. Koç and Christof Paar. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 340–352.

[LL99] DJ Leith and WE Leithead. "Comments on the prevalence of linear parameter varying systems". In: *Technical report, Dep. of EE Enginering, University of Strathclyde, Glasgow UK* (1999).

[Wag99a] David Wagner. "The boomerang attack". In: *International Workshop on Fast Software Encryption*. Springer. 1999, pp. 156–170.

[Wag99b] David A. Wagner. "The Boomerang Attack". In: *Fast Software Encryption, 6th International Workshop, FSE'99, Rome, Italy, March 24-26, 1999, Proceedings*. Ed. by Lars R. Knudsen. Vol. 1636. Lecture Notes in Computer Science. Springer, 1999, pp. 156–170.

[Dae+00] Joan Daemen et al. "Nessie proposal: the block cipher Noekeon". In: *Nessie submission* (2000).

[FM00] Michel Fliess and Richard Marquez. "Towards a module theoretic approach to discrete time linear predictive control". In: *International Journal of Control* 73 (2000), pp. 606–623.

[LL00] Douglas Leith and W.E. Leithead. "Survey of gain-scheduling analysis and design". In: *Int. J. Control* 73 (Jan. 2000), pp. 1001–1025. DOI: 10.1080/002071700411304.

[RR00] H. Radjavi and P. Rosenthal. *Simultaneous Triangularization*. Springer, 2000.

[RS00] Wilson J. Rugh and Jeff S. Shamma. "Research on gain scheduling". In: *Automatica* 36.10 (2000), pp. 1401–1425. ISSN: 0005-1098. DOI: https://doi.org/10.1016/S0005-1098(00)00058-3. URL: http://www.sciencedirect.com/science/article/pii/S0005109800000583.

[BDK01] Eli Biham, Orr Dunkelman, and Nathan Keller. "The Rectangle Attack - Rectangling the Serpent". In: *EUROCRYPT 2001*. Ed. by Birgit Pfitzmann. Vol. 2045. LNCS. Springer, Heidelberg, May 2001, pp. 340–357. DOI: 10.1007/3-540-44987-6_21.

[Dwo01] Morris Dworkin. "NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation-Methods and Techniques". In: *National Institute of Standards and Technology/US Department of Commerce* (2001). URL: https://csrc.nist.gov/publications/detail/sp/800-38a/final.

[KKS01] John Kelsey, Tadayoshi Kohno, and Bruce Schneier. "Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent". In: *FSE 2000*. Ed. by Bruce Schneier. Vol. 1978. LNCS. Springer, Heidelberg, Apr. 2001, pp. 75–93. DOI: 10.1007/3-540-44706-7_6.

[Sch01] C.W. Scherer. "LPV control and full block multipliers". In: *Automatica* 37.3 (2001), pp. 361–375. ISSN: 0005-1098. DOI: https://doi.org/10.1016/S0005-1098(00)00176-X. URL: http://www.sciencedirect.com/science/article/pii/S000510980000176X.

[CHJ02] Don Coppersmith, Shai Halevi, and Charanjit Jutla. "Cryptanalysis of stream ciphers with linear masking". In: *Annual International Cryptology Conference*. Springer. 2002, pp. 515–532.

[HR02] Philip Hawkes and Gregory G Rose. "Guess-and-determine attacks on SNOW". In: *International Workshop on Selected Areas in Cryptography*. Springer. 2002, pp. 37–46.

[Cou03] Nicolas T Courtois. "Fast algebraic attacks on stream ciphers with linear feedback". In: *Annual International Cryptology Conference*. Springer. 2003, pp. 176–194.

[CM03] Nicolas T Courtois and Willi Meier. "Algebraic attacks on stream ciphers with linear feedback". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2003, pp. 345–359.

[FA03] Jean-Charles Faugere and Gwénolé Ars. "An algebraic cryptanalysis of nonlinear filter generators using Gröbner bases". PhD thesis. INRIA, 2003.

[JM03] 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*. 2003, pp. 87–99.

[Sar03] Palash Sarkar. "Hiji-bij-bij: A New Stream Cipher with a Self-Synchronizing Mode of Operation". In: *IACR Cryptology ePrint Archive* 2003 (2003), p. 14.

[Arm04] Frederik Armknecht. "Improving fast algebraic attacks". In: *International Workshop on Fast Software Encryption*. Springer. 2004, pp. 65–82.

[DGV04] J. Daemen, R. Govaerts, and J. Vandewalle. *A practical approach to the design of high speed self-synchronizing stream ciphers*. Tech. rep. e-Stream Project, 2004.

[Haw+04] P. Hawkes et al. *Primitive Specification for SSS*. Tech. rep. Available at: `http://www.ecrypt.eu.org/stream/ciphers/sss/sss.pdf`. e-Stream Project, 2004.

[Kas+04] E. Kasper et al. *Correlated Keystreams in MOUSTIQUE*. Tech. rep. ESTREAM Project, 2004.

[Kim+04] Jongsung Kim et al. "The Related-Key Rectangle Attack - Application to SHACAL-1". In: *ACISP 04*. Ed. by Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan. Vol. 3108. LNCS. Springer, Heidelberg, July 2004, pp. 123–136. DOI: `10.1007/978-3-540-27800-9_11`.

[KS04] Alexander Klimov and Adi Shamir. "New cryptographic primitives based on multiword T-functions". In: *Fast Software Encryption: 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004. Revised Papers 11*. Springer. 2004, pp. 1–15.

[SA04] Hebertt Sira-Ramirez and Sunil K Agrawal. *Differentially flat systems*. Crc Press, 2004.

[BDK05] Eli Biham, Orr Dunkelman, and Nathan Keller. "Related-Key Boomerang and Rectangle Attacks". In: *EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. LNCS. Springer, Heidelberg, May 2005, pp. 507–525. DOI: `10.1007/11426639_30`.

[JM05] Antoine Joux and Frédéric Muller. "Two Attacks Against the HBB Stream Cipher". In: *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*. 2005, pp. 330–341.

[Klí05] Vlastimil Klíma. "Cryptanalysis of Hiji-bij-bij (HBB)". In: *IACR Cryptology ePrint Archive* 2005 (2005), p. 3.

[Can06] Christophe De Cannière. "Trivium: A stream cipher construction inspired by block cipher design principles". In: *International Conference on Information Security*. Springer. 2006, pp. 171–186.

[DP06] J. Daemen and K. Paris. *The self-Synchronzing Stream Cipher Moustique*. Tech. rep. Available at: http://www.ecrypt.eu.org/stream/p3ciphers/mosquito/mosquito_p3.pdf. e-Stream Project, 2006.

[FB06] T. Floquet and JP. Barbot. "State and unknown input estimation for linear discrete-time systems". In: *Automatica* 42.11 (2006), pp. 1883–1889.

[JM06a] A. Joux and F. Muller. "Chosen-ciphertext attack against Mosquito". In: *Lecture Note in Computer Science* (2006).

[JM06b] A. Joux and F. Muller. "Chosen-Ciphertext Attacks Against MOSQUITO". In: *Fast Software Encryption - FSE 2006*. Vol. 4047. Lecture Notes in Computer Science. Springer, 2006, pp. 390–404.

[Sat06] Masayuki Sato. "Filter design for LPV systems using quadratically parameter-dependent Lyapunov functions". In: *Automatica* 42.11 (2006), pp. 2017–2023. ISSN: 0005-1098. DOI: `https://doi.org/10.1016/j.automatica.2006.07.001`. URL: `http://www.sciencedirect.com/science/article/pii/S0005109806002755`.

[SH06] S. Sundaram and C. Hadjicostis. "Designing Stable Inverters and State Observers for Switched Linear Systems with Unknown Inputs". In: *Proc. of the 45th IEEE Conference on Decision and Control*. San Diego, CA, USA, Dec. 2006.

[BD07] Eli Biham and Orr Dunkelman. *Differential cryptanalysis of stream ciphers*. Tech. rep. Computer Science Department, Technion, 2007.

[Bog+07] Andrey Bogdanov et al. "PRESENT: An ultra-lightweight block cipher". In: *International workshop on cryptographic hardware and embedded systems*. Springer. 2007, pp. 450–466.

[HJM07] Martin Hell, Thomas Johansson, and Willi Meier. "Grain: a stream cipher for constrained environments". In: *International journal of wireless and mobile computing* 2.1 (2007), pp. 86–93.

[LJ07] S.M. Lee and Hyunseok Ju. "Output feedback model predictive control for LPV systems using parameter-dependent Lyapunov function". In: *Applied Mathematics and Computation* 190 (July 2007), pp. 671–676. DOI: 10.1016/j.amc.2007.01.061.

[Shi+07] Taizo Shirai et al. "The 128-Bit Blockcipher CLEFIA (Extended Abstract)". In: *FSE 2007*. Ed. by Alex Biryukov. Vol. 4593. LNCS. Springer, Heidelberg, Mar. 2007, pp. 181–195. DOI: 10.1007/978-3-540-74619-5_12.

[Tót+07] R. Tóth et al. "Discrete time LPV I/O and state space representations, differences of behavior and pitfalls of interpolation". In: *2007 European Control Conference (ECC)*. 2007, pp. 5418–5425.

[Ber+08] Côme Berbain et al. "Sosemanuk, a fast software-oriented stream cipher". In: *New stream cipher designs*. Springer, 2008, pp. 98–118.

[FD08] Mazen Farhood and Geir E. Dullerud. "Control of nonstationary LPV systems". In: *Automatica* 44.8 (2008), pp. 2108–2119. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2007.12.016. URL: http://www.sciencedirect.com/science/article/pii/S0005109808000290.

[Käs+08] Emilia Käsper et al. "Correlated Keystreams in Moustique". In: *Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings*. 2008, pp. 246–257.

[Arn+09] François Arnault et al. "A New Approach for FCSRs". In: *Selected Areas in Cryptography*. Ed. by Michael J. Jacobson, Vincent Rijmen, and Reihaneh Safavi-Naini. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 433–448. ISBN: 978-3-642-05445-7.

[Ber+09] Guido Bertoni et al. "Keccak sponge function family main document". In: *Submission to NIST (Round 2)* 3.30 (2009), pp. 320–337.

[BK09] Alex Biryukov and Dmitry Khovratovich. "Related-Key Cryptanalysis of the Full AES-192 and AES-256". In: *ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. LNCS. Springer, Heidelberg, Dec. 2009, pp. 1–18. DOI: 10.1007/978-3-642-10366-7_1.

[CY09] Jiali Choy and Huihui Yap. "Impossible Boomerang Attack for Block Cipher Structures". In: *IWSEC 09*. Ed. by Tsuyoshi Takagi and Masahiro Mambo. Vol. 5824. LNCS. Springer, Heidelberg, Oct. 2009, pp. 22–37.

[De +09] JA De Doná et al. "A flatness-based iterative method for reference trajectory generation in constrained NMPC". In: *Nonlinear Model Predictive Control*. Springer, 2009, pp. 325–333.

[DS09] Itai Dinur and Adi Shamir. "Cube attacks on tweakable black box polynomials". In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2009, pp. 278–299.

[Joc09] Danko R. Jocić. "Interpolation norms between row and column spaces and the norm problem for elementary operators". In: *Linear Algebra and its Applications* 430.11 (2009), pp. 2961–2974. ISSN: 0024-3795. DOI: https://doi.org/10.1016/j.laa.2009.01.011. URL: https://www.sciencedirect.com/science/article/pii/S0024379509000305.

[MD09] Gilles Millérioux and Jamal Daafouz. "Flatness of switched linear discrete-time systems". In: *IEEE Transactions on Automatic Control* 54.3 (2009), pp. 615–619.

[Car10] C. Carlet. "Boolean Models and Methods in Mathematics, Computer Science, and Engineering". In: Cambridge Press, 2010. Chap. Vectorial Boolean Functions for Cryptography.

[DKS10] Orr Dunkelman, Nathan Keller, and Adi Shamir. "A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony". In: *CRYPTO 2010*. Ed. by Tal Rabin. Vol. 6223. LNCS. Springer, Heidelberg, Aug. 2010, pp. 393–410. DOI: 10.1007/978-3-642-14623-7_21.

[HDM10] W. P. M. H. Heemels, J. Daafouz, and G. Millérioux. "Observer-Based Control of Discrete-Time LPV Systems With Uncertain Parameters ". In: *IEEE Transactions on Automatic Control* 55.9 (2010), pp. 2130–2135.

[Arn+11a] F. Arnault et al. "Revisiting LFSRs for Cryptographic Applications". In: *IEEE Transactions on Information Theory* 57.12 (2011), pp. 8095–8113.

[Arn+11b] François Arnault et al. "Revisiting LFSRs for Cryptographic Applications". In: *IEEE Transactions on Information Theory* 57.12 (2011), pp. 8095–8113.

[GNL11] Zheng Gong, Svetla Nikova, and Yee Wei Law. "KLEIN: a new family of lightweight block ciphers". In: *International workshop on radio frequency identification: security and privacy issues*. Springer. 2011, pp. 1–18.

[Meu11] Thomas Meurer. "Flatness-based trajectory planning for diffusion–reaction systems in a parallelepipedon—A spectral approach". In: *Automatica* 47.5 (2011), pp. 935–949.

[Mur11] Sean Murphy. "The Return of the Cryptographic Boomerang". In: *IEEE Trans. Information Theory* 57.4 (2011), pp. 2517–2521. DOI: 10.1109/TIT.2011.2111091. URL: https://doi.org/10.1109/TIT.2011.2111091.

[Shi+11] Kyoji Shibutani et al. "Piccolo: An Ultra-Lightweight Blockcipher". In: *CHES*. Vol. 6917. Lecture Notes in Computer Science. Springer, 2011, pp. 342–357. DOI: 10.1007/978-3-642-23951-9_23. URL: https://www.iacr.org/archive/ches2011/69170343/69170343.pdf.

[WZ11] Wenling Wu and Lei Zhang. "LBlock: a lightweight block cipher". In: *International conference on applied cryptography and network security*. Springer. 2011, pp. 327–344.

[Cha+12] Abbas Chamseddine et al. "Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle". In: *IEEE Transactions on Aerospace and Electronic Systems* 48.4 (2012), pp. 2832–2848.

[Kan+12a] Christoph Kandler et al. "A differential flatness based model predictive control approach". In: Oct. 2012, pp. 1411–1416. ISBN: 978-1-4673-4503-3. DOI: 10.1109/CCA.2012.6402435.

[Kan+12b] Christoph Kandler et al. "A differential flatness based model predictive control approach". In: *2012 IEEE International Conference on Control Applications*. IEEE. 2012, pp. 1411–1416.

[Gér+13] Benoît Gérard et al. "Block ciphers that are easier to mask: How far can we go?" In: *International Conference on Cryptographic Hardware and Embedded Systems*. Springer. 2013, pp. 383–399.

[PM13] J. Parriaux and G. Millérioux. "Nilpotent semigroups for the characterization of flat outputs of switched linear and LPV discrete-time systems". In: *Systems and Control Letters* 62.8 (2013), pp. 679–685.

[SGB13] Olivier Sename, Péter Gáspár, and József Bokor. *Robust Control and Linear Parameter Varying Approaches: Application to Vehicle Dynamics*. Feb. 2013. ISBN: 9783642361104. DOI: 10.1007/978-3-642-36110-4.

[BR14] Andrey Bogdanov and Vincent Rijmen. "Linear hulls with correlation zero and linear cryptanalysis of block ciphers". In: *Designs, codes and cryptography* 70.3 (2014), pp. 369–383.

[Gro+14] Vincent Grosso et al. "LS-designs: Bitslice encryption for efficient masked software implementations". In: *International Workshop on fast software encryption*. Springer. 2014, pp. 18–37.

[Ban+15] Subhadeep Banik et al. "Midori: A block cipher for low energy". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2015, pp. 411–436.

[Bea+15] Ray Beaulieu et al. "The SIMON and SPECK lightweight block ciphers". In: *Proceedings of the 52nd Annual Design Automation Conference*. 2015, pp. 1–6.

[Ber+15] Thierry P Berger et al. "Extended generalized Feistel networks using matrix representation to propose a new lightweight block cipher: Lilliput". In: *IEEE Transactions on Computers* 65.7 (2015), pp. 2074–2089.

[MB15] G. Millérioux and T. Boukhobza. "Characterization of flat outputs for LPV discrete-time systems: a graph-oriented approach". In: *54th Conference on Decision and Control (CDC 2015)*. Osaka, Japan, Dec. 2015.

[Pfi+15] Harald Pfifer et al. "Linear Parameter Varying Techniques Applied to Aeroservoelastic Aircraft: In Memory of Gary Balas". In: *IFAC-PapersOnLine* 48.26 (2015). 1st IFAC Workshop on Linear Parameter Varying Systems LPVS 2015, pp. 103–108. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.11.121. URL: http://www.sciencedirect.com/science/article/pii/S2405896315023794.

[Sho+15] Y. Shoukry et al. "Secure state reconstruction in differentially flat systems under sensor attacks using satisfiability modulo theory solving". In: *2015 54th IEEE Conference on Decision and Control (CDC)*. 2015.

[Bei+16] Christof Beierle et al. "The SKINNY family of block ciphers and its low-latency variant MANTIS". In: *Annual International Cryptology Conference*. Springer. 2016, pp. 123–153.

[Dob+16] Christoph Dobraunig et al. "Ascon v1. 2". In: *Submission to the CAESAR Competition* 5.6 (2016), p. 7.

[Dra17] Brandon Dravie. "Synchronization and dynamical systems : application to cryptography. (Synchronisation et systèmes dynamiques : application à la cryptographie)". PhD thesis. University of Lorraine, Nancy, France, 2017. URL: https://tel.archives-ouvertes.fr/tel-01591554.

[DGM17] Brandon Dravie, Philippe Guillot, and Gilles Millérioux. "Security proof of the canonical form of self-synchronizing stream ciphers". In: *Designs, Codes and Cryptography* 82.1 (2017), pp. 377–388.

[FM17] M. Fiacchini and G. Millérioux. "Dead-beat stabilizability of autonomous switched linear discrete-time systems". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 4576–4581.

[BC18] Christina Boura and Anne Canteaut. "On the Boomerang Uniformity of Cryptographic Sboxes". In: *IACR Trans. Symm. Cryptol.* 2018.3 (2018), pp. 290–310. ISSN: 2519-173X. DOI: 10.13154/tosc.v2018.i3.290-310.

[Cid+18] Carlos Cid et al. "Boomerang Connectivity Table: A New Cryptanalysis Tool". In: *EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. LNCS. Springer, Heidelberg, Apr. 2018, pp. 683–714. DOI: 10.1007/978-3-319-78375-8_22.

[BPT19] Christina Boura, Léo Perrin, and Shizhu Tian. "Boomerang Uniformity of Popular S-box Constructions". In: *Proceedings of The Eleventh International Workshop on Coding and Cryptograph (WCC)*. 2019.

[FM19] M. Fiacchini and G. Millérioux. "Dead-beat Stabilizability of discrete-time switched linear systems: algorithms and applications". In: *IEEE Trans. on Automatic Control* 64 (9 2019), pp. 3839–3845.

[Li+19] Kangquan Li et al. "New Results About the Boomerang Uniformity of Permutation Polynomials". In: *IEEE Trans. Information Theory* 65.11 (2019), pp. 7542–7553. DOI: 10.1109/TIT.2019.2918531. URL: https://doi.org/10.1109/TIT.2019.2918531.

[Sán+19] Helem S. Sánchez et al. "Bibliographical review on cyber attacks from a control oriented perspective". In: *Annual Reviews in Control* 48 (2019), pp. 103–128.

[SQH19] Ling Song, Xianrui Qin, and Lei Hu. "Boomerang Connectivity Table Revisited". In: *IACR Trans. Symm. Cryptol.* 2019.1 (2019), pp. 118–141. ISSN: 2519-173X. DOI: 10.13154/tosc.v2019.i1.118-141.

[WP19] Haoyang Wang and Thomas Peyrin. "Boomerang Switch in Multiple Rounds". In: *IACR Trans. Symm. Cryptol.* 2019.1 (2019), pp. 142–169. ISSN: 2519-173X. DOI: 10.13154/tosc.v2019.i1.142-169.

[Bou+20] Hamid Boukerrou et al. "On the Feistel Counterpart of the Boomerang Connectivity Table". In: *IACR Transactions on Symmetric Cryptology* 2020.1 (May 2020), pp. 331–362. DOI: 10.13154/tosc.v2020.i1.331-362. URL: https://hal.inria.fr/hal-02945065.

[Fra+20] J. Francq et al. "Non-Triangular Self-Synchronizing Stream Ciphers". In: *IEEE Transactions on Computers* (2020). Early Access.

[ALY21] A. Abdelwahab, W. Lucia, and A. Youssef. "Covert Channels in Cyber-Physical Systems". In: *IEEE Control Systems Letters* 5.4 (2021), pp. 1273–1278.

[FZ21] Moritz Fauser and Ping Zhang. "Resilient Homomorphic Encryption Scheme for Cyber-Physical Systems". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE. 2021, pp. 5634–5639.

[BMM22] Hamid Boukerrou, Gilles Millerioux, and Marine Minier. "Towards a new design of ciphers to secure CPS: the role of control theory". In: *2022 26th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE. 2022, pp. 13–18.

[AG] Pierre Apkarian and Pascal Gahinet. *A Convex Characterization of Gain-Scheduled H∞ Controllers*.

[Nyb] Kaisa Nyberg. "Linear approximation of block ciphers (rump session)". In: *EUROCRYPT*. Vol. 94, pp. 9–12.

Résumé

Cette thèse vise à établir un lien entre la cryptographie et la théorie du contrôle. Elle se place dans le contexte de la sécurité des données et des systèmes, en particulier les systèmes cyber-physiques. A l'image de la question actuelle du contrôle chiffré dans les systèmes en réseau, cette thèse a mis en évidence l'intérêt d'utiliser les concepts de l'automatique pour aborder des questions en lien avec la cryptographie. Un cadre pour la conception de chiffreurs à flot auto-synchronisants, une classe particulière de chifreurs symétriques, a été proposé. La conception est reformulée sous forme de questions de la théorie de contrôle telles que l'inversion à gauche, la platitude, le recours à des automates admettant une représentation matricielle comme les modèles LPV et l'analyse structurelle. La principale contribution du point de vue de l'automatique est la proposition d'une méthodologie pour construire des automates LPV plats. D'un point de vue cryptographique, l'intérêt d'un tel cadre réside dans le fait qu'une classe plus large de chiffreurs à flot auto-synchronisants peut être proposée au regard des algorithmes et des principes existants. Plusieurs architectures, y compris des automates hybrides, ont été proposés et justifiées par des compromis performances dynamiques/sécurité.

Mots-clés: Analyse structurelle, Platitude, Systèmes LPV, Synchronisation, Cryptographie, Cryptanalyse

Abstract

This thesis aimed at bringing a connection between encryption purposes and control theory. It has been motivated by the unprecedented need for ensuring security of data and systems like Cyber Physical Systems. Likely to the timely investigation of encrypted control in the literature, this thesis highlighted the interest of using automatic control concepts to tackle other issues borrowed from cryptography. A framework for the design of dedicated Self-Synchronizing Stream Ciphers has been proposed. The design is recast as control theoretical issues. It is based on left inversion, flatness, automata admitting a matrix representation like LPV models, and structural analysis. As a main contribution from an automatic control point of view, it has been proposed a systematic methodology based on digraph considerations to construct flat LPV automata. From a cryptographic point of view, the interest of such a framework lies in that a broader class of self-synchronizing automata can be proposed compared to the ones obtained from the fully-fledged Maurer's principle that is restricted to T-functions. Furthermore, the hybrid architecture, not included in the Maurer's framework, has been motivated by a trade-off between computational complexity and security.

Keywords: Structural analysis, Flatness, LPV systems, Synchronization, Cryptography, Cryptanalysis