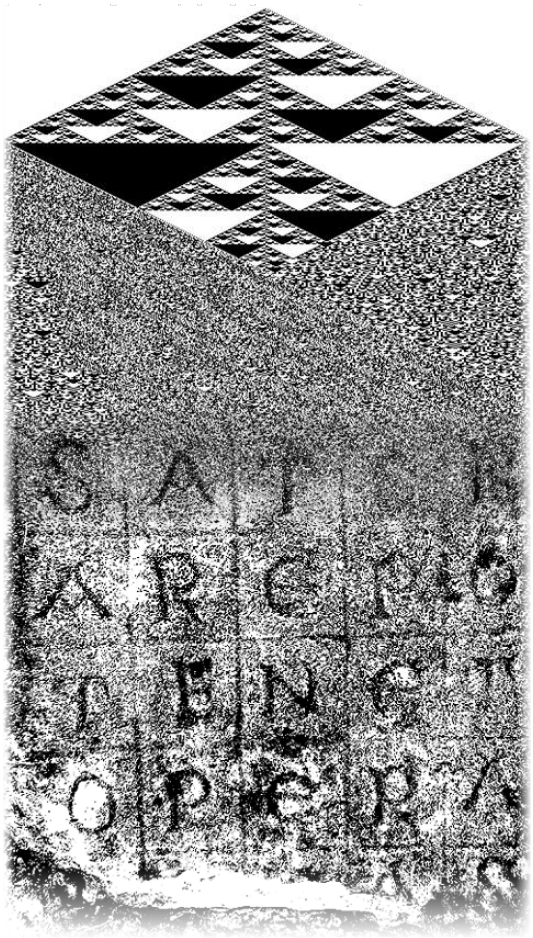
CELLULAR AUTOMATA, BOOLEAN FUNCTIONS AND COMBINATORIAL DESIGNS

LUCA MARIOT



Typeset with LaTeX, `classicthesis` template by André Miede.

S	A	T	O	R
A	R	E	P	O
T	E	N	E	T
O	P	E	R	A
R	O	T	A	S

Note on the titlepage: the cover art represents the dynamic evolution of CA rule 2377216590, morphed with a fragment of the SATOR square in Brusaporto, Bergamo, Italy.
https://en.wikipedia.org/wiki/Sator_Square
https://www.wolframalpha.com/input/?i=Cellular+automaton+2377216590

Scuola di Dottorato
Università degli Studi di Milano–Bicocca

Dipartimento di Informatica, Sistemistica e Comunicazione

PhD Program in Computer Science – XXX Cycle

CELLULAR AUTOMATA, BOOLEAN FUNCTIONS AND COMBINATORIAL DESIGNS

PhD Thesis

Luca Mariot
701962

Double PhD Program – Université Nice Sophia Antipolis

Supervisor: Prof. Alberto Leporati
Co-supervisor: Prof. Enrico Formenti

Tutor: Prof. Giuseppe Vizzari
Coordinator: Prof. Stefania Bandini

ACADEMIC YEAR 2016/2017

It may be surprising to see emotional sensibility invoked à propos of mathematical demonstrations which, it would seem, can interest only the intellect. This would be to forget the feeling of mathematical beauty, of the harmony of numbers and forms, of geometric elegance. This is a true esthetic feeling that all real mathematicians know, and surely it belongs to emotional sensibility.

—Henri Poincaré, *Reflections – Mathematical Creation*

ABSTRACT

The goal of this thesis is to investigate *Cellular Automata* (CA) from the perspective of *Boolean functions* and *combinatorial designs*. Besides the inherent theoretical interest, this research also bases its motivation in cryptography, since Boolean functions and combinatorial designs have several applications in the design of *Pseudorandom Number Generators* (PRNG) and *Secret Sharing Schemes* (SSS).

The contributions presented in this thesis are developed along three main research lines, organized as follows.

The first research line concerns the use of heuristic optimization algorithms for designing Boolean functions with good cryptographic properties, to be used as local rules in CA-based PRNG. The main motivation is to improve Wolfram's pseudorandom generator, which has been shown to be vulnerable to two cryptanalytic attacks due to the poor cryptographic properties of rule 30. In this research line, we first develop a discrete *Particle Swarm Optimizer* (PSO) which explores the space of truth tables of balanced Boolean functions having good nonlinearity, resiliency and propagation criteria. Next, we design a *Genetic Algorithm* (GA) which works on a different representation of Boolean functions, namely their *Walsh spectrum*.

The second research line deals with vectorial Boolean functions generated by CA *global rules*. The first contribution investigates the period of preimages of *spatially periodic configurations* under the action of surjective CA, a problem which is related to the maximum number of players in a CA-based SSS already published in the literature. The second contribution analyzes the cryptographic properties of CA global rules, focusing on their *algebraic degree*, *nonlinearity* and *differential uniformity*. We then adopt a heuristic approach based on *Genetic Programming* (GP) to evolve *S-boxes* defined by CA with nonlinearity and differential uniformity. As a last contribution in this research line, we focus on the *resiliency* criterion and introduce a new cryptographic property for CA-based S-boxes, namely *asynchrony immunity*.

The third research line deals with combinatorial designs generated by CA. We specifically focus on the case of *Orthogonal Latin Squares* (OLS), since they are equivalent to *perfect authentication codes* and *threshold secret sharing schemes*. To this end, our first contribution in this research line concerns the construction and the enumeration of OLS generated through linear CA, leveraging on results from the theory of finite fields. The second contribution, on the other hand, extends the investigation to OLS generated by nonlinear CA, using both a combinatorial approach for exhaustive enumeration and a heuristic approach based on GA and GP.

RIASSUNTO

Lo scopo di questa tesi è studiare gli *Automi Cellulari* (AC) dalla prospettiva delle *funzioni booleane* e dei *disegni combinatorici*. Oltre all'intrinseco interesse teorico, questa ricerca è motivata da applicazioni alla crittografia, dato che sia le funzioni booleane che i disegni combinatorici sono utilizzati nella progettazione di *generatori di numeri pseudocasuali* (*Pseudorandom Number Generators*, PRNG) e degli *schemi di condivisione di segreti* (*Secret Sharing Schemes*, SSS).

I contributi della tesi sono stati sviluppati lungo tre linee di ricerca, di seguito descritte.

La prima linea di ricerca riguarda l'utilizzo di algoritmi di ottimizzazione euristica per cercare funzioni booleane aventi buone proprietà crittografiche, da impiegare come regole locali nei PRNG basati su AC. La motivazione principale di questo studio è il miglioramento del generatore di Wolfram, che è stato dimostrato essere vulnerabile a due attacchi crittoanalitici, a causa delle cattive proprietà crittografiche della regola 30. In primo luogo, in questa linea di ricerca viene sviluppata una versione discreta di un *Particle Swarm Optimizer* (PSO), che esplora lo spazio delle tabelle di verità di funzioni booleane bilanciate aventi una buona combinazione di *nonlinearità*, *ordine di resilienza* e *criterio di propagazione*. In seguito, viene proposto un *Algoritmo Genetico* (*Genetic Algorithm*, GA) basato su una rappresentazione differente delle funzioni booleane, in particolare il loro *spettro di Walsh*.

La seconda linea di ricerca si occupa delle funzioni booleane vettoriali generate dalle regole globali degli AC. Il primo contributo in questa linea di ricerca considera il periodo delle preimmagini di *configurazioni spazialmente periodiche* sotto l'azione di un AC suriettivo, un problema che è collegato al numero di partecipanti di un SSS basato sugli AC pubblicato in letteratura. Il secondo contributo consiste nell'analisi delle proprietà crittografiche delle regole globali degli AC, con particolare riguardo al loro *grado algebrico*, *nonlinearità* e *uniformità differenziale*. Viene in seguito adottato un approccio euristico basato sulla *Programmazione Genetica* (*Genetic Programming*, GP) per ottimizzare le *S-box* definite da AC con una buona nonlinearità e uniformità differenziale. Infine, viene considerato l'*ordine di resilienza* e introdotta una nuova proprietà crittografica per le S-box generate da AC, l'*immunità all'asincronia*.

La terza linea di ricerca riguarda i disegni combinatorici generati tramite AC. Più precisamente, viene considerato il caso dei *Quadrati Latini Ortogonali* (*Orthogonal Latin Squares*, OLS), poiché sono equivalenti ai *codici di autenticazione perfetti* e agli *schemi di condivisione dei segreti a soglia*. Il primo contributo in questa linea di ricerca concerne

la costruzione e il conteggio degli OLS generati da AC lineari, basandosi su risultati della teoria dei campi finiti. Il secondo contributo, d'altro canto, generalizza questa analisi a OLS generati da AC non-lineari, sia attraverso un metodo combinatorico per l'enumerazione esaustiva, sia tramite un approccio euristico basato su GA e GP.

RÉSUMÉ

Le but de cette thèse est l'étude des *Automates Cellulaires* (AC) dans la perspective des *fonctions booléennes* et des *dessins combinatoires*. Au delà de son intérêt théorique, cette recherche est motivée par ses applications à la cryptographie, puisque les fonctions booléennes et les dessins combinatoires sont utilisés pour construir des *générateurs de nombres pseudoaléatoires* (*Pseudorandom Number Generators*, PRNG) et des *schémas de partage de secret* (*Secret Sharing Schemes*, SSS).

Les résultats présentés dans la thèse ont été développé sur trois lignes de recherche, organisées comme suit.

La première ligne de recherche concerne l'emploi des algorithmes d'optimisation heuristique pour chercher des fonctions booléennes ayant des bonnes propriétés cryptographiques, à utiliser comme des règles locales dans des PRNG basés sur les AC. La motivation principale est l'amélioration du générateur de Wolfram, qui a été montré être vulnérable vis à vis de deux attaques cryptanalytiques, à cause des mauvaises propriétés cryptographiques de la règle 30. D'abord, on développe une version discrète d'un *optimisateur par essaims particulaires* (*Particle Swarm Optimzer*, PSO) qui explore l'espace des tables de vérité des fonctions booléennes balancées ayant une bonne combinaison de *nonlinéarité*, ordre de *résilience* et *critère de propagation*. Ensuite, on propose un *Algorithme Génétique* (*Genetic Algorithm*, GA) qui travail sur une représentation différente des fonctions booléennes, c'est à dire leur *spectre de Walsh*.

La deuxième ligne de recherche s'occupe des fonctions booléennes vectorielles engendrées par les règles globales des AC. La première contribution considère la période des preimages des *configurations spatialement périodiques* sous l'action d'un AC surjectif, un problème qui est lié au nombre des joueurs qui peuvent participer dans un SSS basé sur les AC qui est publié dans la littérature. La deuxième contribution analyse les propriétés cryptographiques des règles globales des AC, en se concentrant sur le degré algébrique, la *nonlinéarité* et l'*uniformité differentielle*. Nous adoptons après un approche heuristique basé sur la *Programmation Génétique* (*Genetic Programming*, GP) pour optimiser des *S-boxes* définies par des AC avec une bonne nonlinéarité et uniformité differentielle. Enfin, on considère l'*ordre de résilience* et on introduit une nouvelle propriété cryptographique pour les S-boxes engendrées par les AC, l'*immunité à l'asynchronie*.

La troisième ligne de recherche se concentre sur les dessins combinatoires engendrés par les AC. Plus précisément, on considère les *Carrés Latins Orthogonaux* (*Orthogonal Latin Squares*, OLS), puisque ils sont équivalents aux *codes d'authentification cartésiens* et aux *schémas de*

partage de secret à seuil. Notre première contribution dans cette ligne de recherche regarde la construction et l'énumération des OLS engendrés par des AC linéaires, qui se base sur des résultats de la théorie des corps finis. La deuxième contribution, par contre, généralise cette investigation à des OLS engendrés par des AC nonlinéaires, soit avec un méthode combinatoire pour l'énumération exhaustive, soit avec une approche heuristique basé sur les GA et la GP.

PUBLICATIONS

This thesis is based on the following papers published by the author during the doctorate:

[1] L. Mariot and A. Leporati. A cryptographic and coding-theoretic perspective on the global rules of cellular automata. *Natural Computing*, *DOI:* 10.1007/s11047-017-9635-0, 2017.

[2] L. Mariot, A. Leporati, A. Dennunzio, and E. Formenti. Computing the periods of preimages in surjective cellular automata. *Natural Computing*, 16(3):367–381, 2017.

[3] L. Mariot, E. Formenti, and A. Leporati. Enumerating orthogonal latin squares generated by bipermutive cellular automata. In *Cellular Automata and Discrete Complex Systems - 23rd IFIP WG 1.5 International Workshop, AUTOMATA 2017, Milan, Italy, June 7-9, 2017, Proceedings*, pages 151–164, 2017.

[4] L. Mariot, S. Picek, D. Jakobovic, and A. Leporati. Evolutionary algorithms for the design of orthogonal latin squares based on cellular automata. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Berlin, Germany, July 15-19, 2017*, pages 306–313, 2017.

[5] S. Picek, L. Mariot, A. Leporati, and D. Jakobovic. Evolving s-boxes based on cellular automata with genetic programming. In *Genetic and Evolutionary Computation Conference, Berlin, Germany, July 15-19, 2017, Companion Material Proceedings*, pages 251–252, 2017.

[6] S. Picek, L. Mariot, B. Yang, D. Jakobovic, and N. Mentens. Design of s-boxes defined with cellular automata rules. In *Proceedings of the Computing Frontiers Conference, CF'17, Siena, Italy, May 15-17, 2017*, pages 409–414, 2017.

[7] L. Mariot. Asynchrony immune cellular automata. In *Cellular Automata - 12th International Conference on Cellular Automata for Research and Industry, ACRI 2016, Fez, Morocco, September 5-8, 2016. Proceedings*, pages 176–181, 2016.

[8] L. Mariot and A. Leporati. Resilient vectorial functions and cyclic codes arising from cellular automata. In *Cellular Automata - 12th International Conference on Cellular Automata for Research and Industry, ACRI 2016, Fez, Morocco, September 5-8, 2016. Proceedings*, pages 34–44, 2016.

[9] L. Mariot and A. Leporati. A genetic algorithm for evolving plateaued cryptographic boolean functions. In *Theory and Practice of Natural Computing - Fourth International Conference, TPNC 2015, Mieres, Spain, December 15-16, 2015. Proceedings*, pages 33–45, 2015.

[10] L. Mariot and A. Leporati. Heuristic search by particle swarm optimization of boolean functions for cryptographic applications. In *Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015, Companion Material Proceedings*, pages 1425–1426, 2015.

[11] L. Mariot and A. Leporati. On the periods of spatially periodic preimages in linear bipermutive cellular automata. In *Cellular Automata and Discrete Complex Systems - 21st IFIP WG 1.5 International Workshop, AUTOMATA 2015, Turku, Finland, June 8-10, 2015. Proceedings*, pages 181–195, 2015.

Additionally, parts of this thesis are based on the following preprints and postprints published by the author during the doctorate:

[12] L. Mariot, S. Picek, A. Leporati, and D. Jakobovic. Cellular automata based s-boxes. Cryptology ePrint Archive, Report 2017/1055, 2017. URL: https://eprint.iacr.org/2017/1055.

[13] L. Mariot, E. Formenti, and A. Leporati. Constructing orthogonal latin squares from linear cellular automata. *CoRR*, abs/1610.00139, 2016. URL: http://arxiv.org/abs/1610.00139.

Below is a list of the author's past publications:

[14] A. Leporati and L. Mariot. Cryptographic properties of bipermutive cellular automata rules. *J. Cellular Automata*, 9(5-6):437–475, 2014.

[15] L. Mariot and A. Leporati. Sharing secrets by computing preimages of bipermutive cellular automata. In *Cellular Automata - 11th International Conference on Cellular Automata for Research and Industry, ACRI 2014, Krakow, Poland, September 22-25, 2014. Proceedings*, pages 417–426, 2014.

[16] A. Leporati and L. Mariot. 1-resiliency of bipermutive cellular automata rules. In *Cellular Automata and Discrete Complex Systems - 19th International Workshop, AUTOMATA 2013, Gießen, Germany, September 17-19, 2013. Proceedings*, pages 110–123, 2013.

ACKNOWLEDGMENTS

A common saying goes that Science is a collective endeavor. Somewhat, this sentence has acquired a more personal meaning to me since I started my PhD studies. Thinking about all the people I met, worked or lived with during these last three years led me to conclude that what I accomplished in my PhD would not have been possible without them – or that it would have been very different.

Hence, this is an attempt to show my gratitude towards all these people that somehow helped and supported me.

First of all, I would like to thank my PhD supervisor and mentor, Alberto Leporati, for all the time he devoted to me during these three years, and for all the precious advices he gave me to improve my work and to further my career.

Thanks also to Enrico Formenti, my second supervisor, for welcoming me in his research group during my one-year period spent in Nice, and for always being available to hear and discuss my new ideas, even after a day full of lectures.

I am extremely grateful to several people that I met at the Department of Informatics, Systems and Communications in Milan. Among them, special thanks go to Marco Previtali, Simone Zaccaria, Stefano Beretta, Yuri Pirola, Riccardo Dondi, Enrico Porreca, Luca Manzoni, Luca Denti and Murray Patterson, with whom I shared not only the workplace, but also a lot of good and funny moments.

I would like to thank my tutor, Giuseppe Vizzari, who supervised all my PhD training activities. I am also grateful to the coordinator of my PhD program, Stefania Bandini, for encouraging me to undertake the double degree agreement with the University of Nice Sophia Antipolis. Other faculty members that I had the pleasure to know at the Department and that I want to thank here for various reasons are Claudio Ferretti, Claudio Zandron, Daniela Besozzi, Giancarlo Mauri, Gianluca Della Vedova and Paola Bonizzoni.

Thanks to all my coauthors – Stjepan Picek, Domagoj Jakobovic, Bohan Yang, Nele Mentens and Alberto Dennunzio, whose collaboration helped me to improve a lot my research presented in this thesis, and with whom I hope to collaborate again in the future.

My deepest feeling of gratitude goes to my family. Thanks to my parents, Ambra and Carlo, to my brother Davide, to my nephew Jordan, and to Valentina, for their unconditional love and for always encouraging me to pursue my dreams.

A final special thanks goes to Nicole, for her constant support that helped get me through these last few months before the end of my PhD, and for the wonderful friendship we share.

CONTENTS

1 INTRODUCTION 1
1.1 Heuristic Optimization of Boolean Functions 2
1.2 Cryptographic and Coding-Theoretic Analysis of CA 2
1.3 Combinatorial Designs and CA 3
1.4 Thesis Outline 5

I BACKGROUND NOTIONS 9
2 CELLULAR AUTOMATA 11
2.1 Infinite CA 12
2.2 Finite CA 15
2.3 Injectivity and Surjectivity in CA 18
3 LINEAR RECURRING SEQUENCES, COMBINATORIAL DESIGNS AND CODING THEORY 23
3.1 Linear Recurring Sequences and Linear Feedback Shift Registers 23
3.2 Combinatorial Designs, Latin Squares and Orthogonal Arrays 26
3.3 Error-Correcting Codes 29
4 CRYPTOGRAPHY 35
4.1 Symmetric Key Cryptography 36
4.2 Block and Stream Ciphers 40
4.3 Authentication Codes 45
4.4 Secret Sharing Schemes 47
5 BOOLEAN FUNCTIONS FOR CRYPTOGRAPHY 53
5.1 Basic Definitions and Representations of Boolean Functions 53
5.2 Cryptographic Properties of Boolean Functions 55
5.2.1 Balancedness 56
5.2.2 Algebraic Degree 56
5.2.3 Nonlinearity 57
5.2.4 Correlation Immunity and Resiliency 58
5.2.5 Strict Avalanche Criterion and Propagation Criterion 58
5.2.6 Theoretical Bounds 59
5.3 Affine Equivalence for Boolean Functions 60
5.4 Basic Definitions and Representations of S-boxes . . . 60
5.5 Cryptographic Properties of S-boxes 62
5.5.1 Balancedness 62
5.5.2 Algebraic degree 63
5.5.3 Nonlinearity 63
5.5.4 Differential Uniformity 63
5.5.5 Resiliency 64

5.6 Affine Equivalence for S-boxes 65
6 HEURISTIC OPTIMIZATION ALGORITHMS 67
6.1 Local Search Methods 67
6.1.1 Hill Climbing 68
6.1.2 Simulated Annealing 68
6.2 Evolutionary Computation Algorithms 68
6.2.1 Genetic Algorithms 68
6.2.2 Genetic Programming 70
6.3 Particle Swarm Optimization 71
7 SURVEY OF LITERATURE 73
7.1 CA-based PRNGs and Stream Ciphers 73
7.2 CA-based Block Ciphers 75
7.3 Rotation-Symmetric S-boxes 76
7.4 CA-based Secret Sharing Schemes 77
7.5 Heuristic Optimization of Boolean Functions and S-boxes 79

II HEURISTIC OPTIMIZATION OF BOOLEAN FUNCTIONS 81
8 HEURISTIC SEARCH OF BOOLEAN FUNCTIONS BY DISCRETE PSO 83
8.1 PSO Algorithm Description 85
8.1.1 Position Update for Balanced Functions . . . 85
8.1.2 Fitness Functions 87
8.1.3 Overall PSO Algorithm 88
8.2 Parameters Tuning 89
8.2.1 Problem Statement 89
8.2.2 Local Unimodal Sampling 89
8.2.3 Continuous Genetic Algorithms 90
8.2.4 Meta-Fitness Function 90
8.2.5 Meta-Optimization Results 91
8.3 PSO Experiments 92
8.3.1 Experimental Setting 92
8.3.2 Best Solutions Found 92
8.3.3 Comparison with other Heuristics 94
8.4 Conclusions 96
9 EVOLVING PLATEAUED BOOLEAN FUNCTIONS BY GENETIC ALGORITHMS 97
9.1 Genetic Algorithm Description 98
9.1.1 Chromosomes Encoding 98
9.1.2 Objective and Fitness Functions 100
9.1.3 Genetic Operators 100
9.1.4 Overall GA Procedure 102
9.2 GA Experiments 103
9.2.1 Experimental Setting 103
9.2.2 Results 104
9.3 Conclusions 105

III CRYPTOGRAPHIC AND CODING-THEORETIC ANALYSIS OF CELLULAR AUTOMATA 107
10 PREIMAGES PERIODS IN SURJECTIVE CA 109
10.1 Problems Statement and Basic Results 110
10.1.1 Periods of SPC Preimages in Surjective CA . . 111
10.1.2 Graph Characterization of Preimages 112
10.2 Linear CA and Linear Recurring Sequences 116
10.2.1 LBCA Preimages and Concatenated LRS . . . 116
10.2.2 Sum Decomposition of Concatenated LRS . . 117
10.2.3 Characteristic Polynomial of Concatenated LRS 118
10.3 Further Results 121
10.3.1 Computing the Period of a Single Preimage . 121
10.3.2 Periods Multiplicities 123
10.3.3 LBCA over Finite Rings Alphabets 124
10.4 Conclusions 126
11 S-BOXES BASED ON CELLULAR AUTOMATA 127
11.1 Cryptographic Properties of CA Global Rules 128
11.1.1 Algebraic Degree 128
11.1.2 Bounds on Nonlinearity and Differential Uniformity 129
11.2 Heuristic Design of CA-based S-boxes 135
11.2.1 Genetic Programming Approach 135
11.2.2 GP Results 137
11.2.3 Exhaustive Search Results 138
11.3 Conclusions 140
12 RESILIENT AND ASYNCHRONOUS IMMUNE CA 143
12.1 Resilient Functions and Cyclic Codes from CA 144
12.1.1 1–Resiliency of Bipermutive NBCA 144
12.1.2 Linear CA and Cyclic Codes 145
12.1.3 Cyclic Hamming Codes through Linear CA . 147
12.2 Asynchronous Immune CA 149
12.2.1 Basic Definition and Properties of Asynchrony Immunity 149
12.2.2 Search of AI Rules up to 4 Variables 151
12.3 Conclusions 152

IV COMBINATORIAL DESIGNS AND CELLULAR AUTOMATA 155
13 ORTHOGONAL LATIN SQUARES ARISING FROM LINEAR CA 157
13.1 Characterization Results 158
13.1.1 Latin Squares from Bipermutive CA 158
13.1.2 Orthogonal Latin Squares from Linear Bipermutive CA 160
13.1.3 Threshold Schemes and Authentication Codes from Linear CA 162
13.2 Counting coprime polynomial pairs 164
13.2.1 Problem Statement 164

13.2.2 Equivalence relation based on Euclid's algorithm . . . 165
13.2.3 Counting (non-) coprime pairs . . . 168
13.2.4 Counting $(1, 1)$ coprime pairs by recurrence . 170
13.3 MOLS based on Linear CA . . . 173
13.3.1 MOLS from Irreducible Polynomials . . . 174
13.3.2 Lower Bounds on Linear CA-based MOLS . . 175
13.4 Conclusions . . . 176
14 ORTHOGONAL LATIN SQUARES BASED ON NONLINEAR CA 179
14.1 Combinatorial Approach . . . 180
14.1.1 Invariance Under Reflection and Complement 181
14.1.2 Pairwise Balancedness . . . 182
14.2 Combinatorial Enumeration of Pairwise Balanced Bipermutive Rules . . . 184
14.2.1 Counting Pairwise Balanced Bipermutive Rules 184
14.2.2 Exhaustive Search Experiments . . . 186
14.3 Heuristic Optimization Approach . . . 188
14.3.1 Fitness Functions . . . 188
14.3.2 Single Bitstring Encoding . . . 189
14.3.3 Double Bitstring and Double Tree Encodings . 190
14.3.4 Balanced Quaternary String Encoding . . . 190
14.4 Experimentation . . . 192
14.4.1 Experimental Settings . . . 192
14.4.2 Results . . . 193
14.5 Conclusions . . . 195

V FINAL REMARKS 197
15 CONCLUSIONS AND OPEN PROBLEMS 199
15.1 Heuristic Optimization of Boolean Functions . . . 199
15.1.1 Discrete Particle Swarm Optimization . . . 199
15.1.2 Genetic Algorithms and Spectral Inversion . . 200
15.2 Cryptographic and Coding-Theoretic Analysis of CA 201
15.2.1 Preimages Period in Surjective CA . . . 201
15.2.2 CA-Based S-boxes . . . 202
15.2.3 Resiliency and Asynchrony Immunity . . . 205
15.3 Combinatorial Designs and Cellular Automata . . . 206
15.3.1 Orthogonal Latin Squares from Linear CA . . 206
15.3.2 Orthogonal Latin Squares from Nonlinear CA 208

BIBLIOGRAPHY 211

LIST OF FIGURES

Figure 1 Example of ECA with local rule 150. 14
Figure 2 NBCA 16
Figure 3 PBCA 16
Figure 4 Examples of NBCA and PBCA with local rule 150. 16
Figure 5 de Bruijn graph associated to the ECA F defined by rule 150. 19
Figure 6 Diagram of a linear feedback shift register. . . 24
Figure 7 Example of Latin square of order $N = 4$. 27
Figure 8 Orthogonal Latin squares of order $N = 4$, and their superposition. 28
Figure 9 Communication model studied in coding theory. 29
Figure 10 Covering of the message space by the codewords of C. 30
Figure 11 Basic communication scheme studied in cryptography. 36
Figure 12 Encryption 44
Figure 13 Decryption 44
Figure 15 Combiner model for the PRNG of a Vernam-like stream cipher. 45
Figure 16 Schematic description of a $(2,3)$ threshold SSS. 49
Figure 17 Example of GP tree encoding of a Boolean function. 71
Figure 18 Block scheme of Wolfram's stream cipher based on Rule 30. 74
Figure 19 Setup phase of the extended scheme proposed in [112]. 78
Figure 20 Example of preimage construction of an SPC $y = {}^{\omega}u^{\omega}$. The period of the sequence of blocks w_i is at most q^{d-1} 79
Figure 21 Example of application of swap-based position update. 86
Figure 22 Construction of a preimage by s-fusion of finite blocks. 112
Figure 23 $y = {}^{\omega}011^{\omega}$ 114
Figure 24 $y = {}^{\omega}1000^{\omega}$ 114
Figure 25 Examples of u-closure graphs for the elementary CA 106. 114
Figure 26 $y = {}^{\omega}011^{\omega}$ 115
Figure 27 $y = {}^{\omega}1000^{\omega}$ 115

Figure 28 Examples of u-closure graphs for the elementary CA 150. 115
Figure 29 Diagram of two concatenated LFSR. 117
Figure 30 Block $x_{[0,11]}$ which generates preimage $x \in F^{-1}(y)$ under rule 150. 123
Figure 31 PBCA S-Boxes 134
Figure 32 Generic (n, n)-functions 134
Figure 33 Best attainable nonlinearity for PBCA S-boxes and generic S-boxes up to $n = 7$ variables. . . 134
Figure 34 Concatenation of a LFSR with a $n - m$ zeros. . 146
Figure 35 Initialization 148
Figure 36 Complete codeword 148
Figure 37 Example of systematic encoding of using rule 1768527510. 148
Figure 38 Syndrome computation 149
Figure 39 Error correction 149
Figure 40 Example of error correction using rule 1768527510. The cell marked by $*$ indicates where the error occurred. 149
Figure 41 Truth table of F_{150} 160
Figure 42 Latin square $L_{F_{150}}$ 160
Figure 43 Example of Latin square of order 4 induced by rule 150. 160
Figure 44 Rule 150 161
Figure 45 Rule 90 161
Figure 46 Overlay 161
Figure 47 OLS generated by bipermutive CA with rule 150 and 90. 161
Figure 48 Transition graph for the finite state automaton realizing δ. 167

LIST OF TABLES

Table 1 Comparison of Best PSO Parameters 91
Table 2 CGA-Evolved PSO Parameters 92
Table 3 Best Boolean Functions Found, fit_1 93
Table 4 Best Boolean Functions Found, fit_2 93
Table 5 Best Boolean Functions Found, fit_3 93
Table 6 Maximum Nonlinearity Achieved by CI(1) Functions 94
Table 7 Maximum Nonlinearity Achieved by CI(1) and PC(1) functions 95
Table 8 Comparison of Nl and $cidev_2$ Values 95
Table 9 Comparison of Nl and AC_{max} Values 96
Table 10 Cryptographic profiles and spectral multiplicities for plateaued functions of $n = 6$ and $n = 7$ variables 103
Table 11 Statistics of the best solutions found by GA and SA 104
Table 12 Statistical results and comparison. 138
Table 13 Tree sizes, Equation (141) 139
Table 14 Exhaustive classification of CA-based S-boxes up to size 5×5 139
Table 15 Equivalence classes of bijective 3×3 CA S-boxes 140
Table 16 Equivalence classes of bijective 4×4 CA S-boxes 141
Table 17 List of $d = 4$ rules inducing (t, n)–AI CA with $t = 3$ and $n \leqslant 10$. 152
Table 18 Truth table for the transition function δ 167
Table 19 Sizes of the search spaces for the different types of sets of bipermutive functions pairs of up to $d = 7$ variables. 186
Table 20 Distribution of CA-based orthogonal Latin squares up to $d = 6$. 187
Table 21 Example of balanced quaternary string encoding. 191
Table 22 Best solutions found by GA and GP. 194

1 INTRODUCTION

Some of the deepest results in Computer Science share a common trend towards simple abstractions. Mentioning two notable examples, Turing [184] showed that any computable function can be calculated by a simple abstract machine whose behavior consists of either reading or writing 0s and 1s by moving on a bi-infinite tape, while Böhm and Jacopini [18] demonstrated that a programming language only needs to feature sequential composition, conditional statements and loop structures in order to be Turing-complete.

Cellular Automata (CA), originally introduced by Ulam [185] and von Neumann [188], are perhaps one of the computational models that better embody this simplicity principle. In fact, a CA consists of a collection of *cells*, either arranged on a line or a grid, where each cell applies a *local rule* over its neighbors in order to update its state. However, notwithstanding this simple structure, CA can exhibit very complex dynamical behaviors. This feature drew several researchers to consider CA in a wide variety of domains, be it in applicative contexts (such as the simulation of discrete complex systems) or in a more theoretical setting (such as the computational power of CA).

One of the most interesting applications of cellular automata concerns *cryptography*, whose general goal is to enable two abstract parties, a sender and a receiver, to securely communicate over a channel which is eavesdropped by an adversary. As a matter of fact, the dynamical complexity of CA can be exploited to realize cryptosystems which satisfy the principles of *confusion* and *diffusion* set forth by Shannon [164]. Moreover, the aforementioned structural simplicity of CA is also appealing in cryptography because it facilitates the security analysis of the resulting cryptosystem. Finally, due to their massive parallelism, CA can be efficiently implemented in hardware, which make them interesting for the design of cryptographic applications in environments with constrained computational resources.

The goal of this thesis is the study of *Boolean functions* and *combinatorial designs* generated by cellular automata. Indeed, by looking at the relevant literature [30, 31, 172, 173], one can see that *Boolean functions* and *combinatorial designs* underlie the design of several symmetric cryptographic primitives, such as *Pseudorandom Number Generators* (PRNG), *Substitution boxes* (S-boxes) and *Secret Sharing Schemes* (SSS).

For this reason, we structured our investigation along three research lines, which are based and improve on the current state of the art of CA-based cryptographic applications.

1.1 HEURISTIC OPTIMIZATION OF BOOLEAN FUNCTIONS

The first research line pertains the search of Boolean functions with good cryptographic properties to be used as local rules of CA for the design of PRNGs and *stream ciphers*. In this case, the first proposal of a CA-based PRNG for cryptographic purposes dates back to Wolfram [195], who proposed to use a CA equipped with rule 30 to generate pseudorandom keystreams to be used in a Vernam-like stream cipher. However, this design was shown to be vulnerable to cryptanalytic attacks by Meier and Staffelbach [118] and by Koc and Apohan [100], due to the poor cryptographic properties of the Boolean function representing rule 30. As a consequence, later research [114, 106, 63] focused on the search of local rules with better cryptographic properties, mainly using combinatorial methods.

Our contribution in this first research line is the use of *heuristic optimization algorithms* to evolve Boolean functions featuring a good combination of cryptographic properties. Heuristic techniques represent an interesting alternative to *algebraic constructions* [30] for finding Boolean functions which are useful for cryptographic purposes, as witnessed by several publications on the subject [123, 43, 2, 143]. Of course, the use of heuristic methods to optimize cryptographic properties is a research topic that transcends the use of the resulting Boolean functions as local rules of cellular automata: as we mentioned above, Boolean functions also play a vital role in the design of several cryptographic primitives that are not based on CA. Hence, the first research line addressed in this thesis is of independent interest.

More specifically, in this research line we first developed a discrete *Particle Swarm Optimizer* (PSO) which searches the space of truth tables of balanced Boolean functions having good nonlinearity, resiliency and propagation criteria. Next, we designed a *Genetic Algorithm* (GA) which works on a different representation of Boolean functions, namely their *Walsh spectrum*. The advantage is that several cryptographic criteria can be easily encoded on this representation, and the goal of GA is to find Walsh spectra with the desired properties that map to actual Boolean functions.

1.2 CRYPTOGRAPHIC AND CODING-THEORETIC ANALYSIS OF CA

The second research line considers CA as *vectorial Booolean functions* (or S-boxes) instead of focusing on their local rules. This approach is especially useful when considering CA for the design of block ciphers. However, one can see that there is a gap in the literature pertaining this aspect of CA-based cryptography. As a matter of fact, few works [74, 162, 176] have been published in the literature about the design of block ciphers based on CA, and most of them have not been subjected to a rigorous security analysis. A notable exception in

this respect is the CA equipped with rule χ proposed by Daemen et al. [52], whose cryptographic properties have been thoroughly investigated by the authors. To the best of our knowledge, rule χ is the only CA which is featured in a real-world cipher, namely KECCAK, which is now part of NIST SHA-3 standard [15].

However, beside the ad-hoc analysis which has already been carried out for χ, a general cryptographic analysis of CA is still missing in the literature. For this reason, we tackle this task in the second research line by subdividing it in three parts.

The first part investigates the period of preimages of *spatially periodic configurations* under the action of surjective CA. Although this problem may look unrelated to cryptography at a first glance, it is actually connected to determining the maximum number of players allowed in a CA-based Secret Sharing Scheme (SSS) proposed in [112]. In this context, we first show a general methodology based on *de Bruijn graphs* to compute the period of preimages in surjective CA. We then focus on the class of *linear CA*, for which we give a characterization of the preimages period problem in terms of *concatenated linear recurring sequences*.

In the second part of this research line, we begin analyzing the cryptographic properties of CA global rules that are useful for the design of S-boxes in block ciphers. In particular, we characterize the *algebraic degree* and we prove two upper bounds on the *nonlinearity* and *differential uniformity* achievable by S-boxes defined by CA, relating them to the corresponding properties of the underlying local rules. Then, we adopt a heuristic approach based on *Genetic Programming* to evolve CA-based S-boxes having a good trade-off of nonlinearity and differential uniformity.

In the last part, we focus our attention on another criterion of vectorial Boolean functions which is relevant in the context of pseudorandom generation and stream ciphers, namely *resiliency*. This allows us to extend our analysis by considering CA from the standpoint of *coding theory*. In particular, we show that linear CA are equivalent to cyclic codes, so that the CA resiliency order corresponds to the minimum distance of such codes. Additionally, the encoding process of these codes actually amounts to the computation of a CA preimage using the same tools developed for the first part, i. e. concatenated linear recurring sequences. Finally, this part also considers a new cryptographic criterion for the S-boxes of CA, namely *asynchrony immunity*, which could be of interest for the development of *side-channel countermeasures* in CA-based ciphers.

1.3 COMBINATORIAL DESIGNS AND CA

The third research line investigates the combinatorial designs generated by CA, specifically focusing on *Orthogonal Latin Squares* (OLS).

The reason is that OLS are equivalent to *perfect authentication codes* and to *threshold secret sharing schemes*. While the former have never been considered in the literature of CA-based cryptography, the latter have been investigated to some extent [153, 112]. However, all the proposed solutions feature a *sequential threshold* access structure, meaning that the shares in the SSS must satisfy an *adjacency constraint* to reconstruct the secret, beside reaching the specified threshold cardinality. On the other hand, a set of n *Mutually Orthogonal Latin Squares* (MOLS) is equivalent to a $(2, n)$–threshold scheme, meaning that each subset of 2 players out of n can recover the secret by pooling together the respective shares. Thus, a possible way to improve on the state of the art of CA-based SSS is to investigate how to construct sets of MOLS via CA. We remark that, even though threshold secret sharing is a problem which has already been solved without resorting to CA (such as in Shamir's scheme [163] or Blakley's scheme [17]), we deem interesting analyzing the OLS engendered by CA due also to the connections that these combinatorial designs have with coding theory [173].

In the first part of this research line we start by investigating a construction of OLS based on linear CA. In this case, we give an algebraic characterization based on invertible *Sylvester matrices*, which leads us to prove that two linear *bipermutive* CA generate orthogonal Latin squares if and only if the polynomials associated to their local rules are relatively prime. We then continue our analysis by counting all pairs of OLS generated by linear CA, or equivalently all pairs of coprime polynomials of degree n with nonzero constant term. Interestingly, to the best of our knowledge this specific variant of the counting problem of coprime polynomials seems to have received little attention in the literature of finite fields. We thus solve the problem through a recurrence equation, whose closed form turns out to be related to an integer sequence already known in the *Online Encyclopedia of Integer Sequences* (OEIS) for several other combinatorial and number-theoretic facts, not related to CA or OLS. Next, we provide a construction of MOLS generated by linear CA based on irreducible polynomials, and conjecture its optimality.

In the second part, we generalize our investigation to OLS generated by *nonlinear* bipermutive CA, motivated by the fact that these OLS have an application in the design of *cheater-immune* SSS [181]. To this end, we adopt both a combinatorial and a heuristic approach. In the combinatorial approach, we prove some necessary conditions that two nonlinear CA rules must satisfy in order to generate a pair of OLS, and we use such conditions to reduce the search space of all possible rule pairs. This allows us to classify all bipermutive CA equipped with nonlinear local rules of up to $d = 6$ variables that generate OLS. Next, in the heuristic approach we adopt both Genetic

Algorithms and Genetic Programming to evolve pairs of local rules of 7 and 8 variables which generate OLS.

1.4 THESIS OUTLINE

The rest of this thesis is structured in five parts.

Part i, *Background Notions*, covers all preliminary definitions and concepts which are needed to develop the results presented in this thesis. In particular, this part is composed of six chapters as follows:

- *Chapter* 2 recalls the basic notions regarding cellular automata. In particular, it begins by introducing the general model of infinite one-dimensional CA, and then it defines the finite models which will be mostly used in the thesis, namely *No Boundary CA* (NBCA) and *Periodic Boundary CA* (PBCA). The last section concludes with some basic remarks about injectivity and surjectivity properties in CA, which will be used in later chapters to present the original results of the thesis.

- *Chapter* 3 is divided in three sections. The first section covers the basic notions of the theory of *Linear Recurring Sequences* (LRS) and *Linear Feedback Shift Registers* (LFSR), upon which the results reported in Chapter 10 are based upon. The second section deals with *combinatorial designs*, focusing in particular on *Latin squares* and *orthogonal arrays*. Beside paving the way for the following introductory Chapters 4 and 5 respectively about cryptography and Boolean functions, this chapter is also used as a basis for the results proved in Part iv about the combinatorial designs generated by CA. Finally, the third section recalls some basic notions of *coding theory*, focusing in particular on linear error-correcting codes. This section will be used in Chapter 12 to prove the equivalence between linear CA and cyclic codes.

- *Chapter* 4 introduces all necessary background about cryptography. Starting from the generic definition of a cryptosystem, the chapter continues by discussing the various attack models and security definitions, focusing in particular on unconditional security for symmetric key cryptography. Next, the chapter discusses the main cryptographic primitives considered in this thesis, namely stream and block ciphers, authentication codes and secret sharing schemes.

- *Chapter* 5 collects all necessary information on *Boolean functions* for cryptography. The first section is devoted to the basic notions about single-output Boolean functions and their cryptographic properties, and it serves as a basis for the results presented in Chapters 8 and 9. The second section, on the other hand, generalizes the discussion to *vectorial Boolean functions*,

and describes their generalized cryptographic properties. Chapters 11 and 12 are based on the notions presented in this section.

- *Chapter* 6 features a brief introduction to heuristic optimization algorithms, focusing in particular on those which will be used in later chapters of the thesis. In particular, the sections about local search methods, Genetic Algorithms and Particle Swarm Optimization constitute the preliminary part for Chapters 8 and 9 on the heuristic optimization of Boolean functions with good cryptographic properties. The section about Genetic Programming, on the other hand, serves as a basis for the experimental results reported in Chapters 11 and 14 respectively on CA-based S-boxes and orthogonal Latin squares.

- *Chapter* 7 provides a survey of the literature regarding the applications of CA to cryptography, as well as a review of heuristic methods adopted for optimizing the cryptographic properties of Boolean functions.

Part ii, *Heuristic Optimization of Boolean Functions*, is devoted to the first research line investigated in this thesis, described in Section 1.1. It is composed of the following two chapters:

- *Chapter* 8 presents a discrete Particle Swarm Optimization algorithm for the search of balanced Boolean functions with a good combination of cryptographic properties, namely nonlinearity, resiliency and propagation criterion.

  This chapter is based on an extended version of the short paper *"Heuristic Search by Particle Swarm Optimization of Boolean Functions for Cryptographic Applications"*, presented at GECCO 2015.

- *Chapter* 9 discusses the heuristic search of plateaued Boolean functions, which are optimal with respect to nonlinearity, algebraic degree and resiliency order, through Genetic Algorithms and spectral inversion.

  This chapter is based on the conference paper *"A Genetic Algorithm for Evolving Plateaued Cryptographic Boolean Functions"*, presented at TPNC 2015.

Part iii, *Cryptographic and Coding-Theoretic Analysis of Cellular Automata*, focuses on the second research line discussed in Section 1.2, namely the analysis of the global rules of CA from the perspective of their cryptographic properties and applications to error-correcting codes. This part is composed of three chapters as follows:

- *Chapter* 10 investigates the periods of preimages of spatially periodic configurations under the action of surjective CA. This

research is motivated by the problem of determining the maximum number of players allowed in the CA-based secret sharing scheme proposed in [112]. The first part is devoted to characterizing the period of preimages under generic surjective CA, while the second one focuses on the specific case of linear CA by leveraging on the theory of linear recurring sequences presented in Chapter 3.

This chapter is based on the conference paper *"On the Periods of Spatially Periodic Preimages in Linear Bipermutive Cellular Automata"* presented at AUTOMATA 2015 and its extended journal version *"Computing the periods of preimages in surjective cellular automata"* published on Natural Computing.

- *Chapter 11* analyzes the cryptographic properties of S-boxes defined by finite CA. The theoretical part of the chapter characterizes the algebraic degree of CA and proves two upper bounds on the nonlinearity and differential uniformity of CA-based S-boxes by relating them to the corresponding properties of their local rules. Next, the experimental part employs a Genetic Programming algorithm to evolve S-boxes defined by CA with good cryptographic properties.

  This chapter is based on the preprint of tue paper *"Cellular Automata based S-boxes"* submitted to Cryptography and Communications, which is available on the `IACR eprint` archive, and it is a joint work with Stjepan Picek and Domagoj Jakobovic.

- *Chapter 12* concludes the investigation of the cryptographic properties of CA by addressing their resiliency order. Specifically, the chapter shows that the S-box defined by a bipermutive local rule is always at least 1–resilient. Additionally, we prove an equivalence between linear CA and cyclic codes, where the resiliency order of the former determines the minimum distance of the latter. Finally, we introduce a new property, called *Asynchrony Immunity*, which could be relevant in the design of CA-based ciphers as a countermeasure for clock-fault attacks.

  The first part of this chapter is based on the conference paper *"Resilient Vectorial Functions and Cyclic Codes Arising from Cellular Automata"* presented at ACRI 2016 and the corresponding journal version *"A cryptographic and coding-theoretic perspective on the global rules of cellular automata"* accepted in Natural Computing, and currently in press. Additionally, the part on asynchrony immunity is based on the conference paper *"Asynchrony Immune Cellular Automata"* presented at ACA 2016.

Part iv, *Combinatorial Designs and Cellular Automata*, discusses the results developed in the third research line presented in Section 1.3. It is composed of the following two chapters:

- *Chapter* 13 investigates Orthogonal Latin Squares (OLS) generated by linear cellular automata. In particular, it gives a characterization of such OLS which is based on the *Sylvester matrix* induced by two linear CA. The main theoretical result in this context is that two linear CA generate a pair of OLS if and only if the polynomials associated to their rules are relatively prime. Next, the chapter focuses on counting such pairs of coprime polynomials, uncovering a connection with an integer sequence already published in the OEIS. Finally, the chapter presents a construction of Mutual Orthogonal Latin Squares (MOLS) based on irreducible polynomials defining CA local rules, and conjectures its optimality as a closing remark.

  This chapter is based on an extended version of the exploratory paper *"Constructing Orthogonal Latin Squares from Linear Cellular Automata"* presented at AUTOMATA 2016, and available as an `arXiv` postprint.

- *Chapter* 14 generalizes the investigation of CA-based OLS by considering nonlinear bipermutive rules. In particular, the theoretical part of this chapter proves some necessary conditions on the local rules of two bipermutive CA that induce OLS, which allows to reduce the search space of candidate solutions and thus perform an exhaustive search up to diameter 6. The second part, on the other hand, employs Genetic Algorithms and Genetic Programming to evolve pairs of nonlinear bipermutive CA of diameter 7 and 8 which generate OLS.

  This chapter is based on the conference papers *"Enumerating Orthogonal Latin Squares Generated by Bipermutive Cellular Automata"* and *"Evolutionary algorithms for the design of orthogonal Latin squares based on cellular automata"*, respectively presented at AUTOMATA 2017 and GECCO 2017. The latter, in particular, is a joint work with Stjepan Picek and Domagoj Jakobovic.

Part v, *Final Remarks*, is composed of Chapter 15 which concludes the thesis by putting the presented results in perspective, and discusses several possible avenues for future research.

Part I

BACKGROUND NOTIONS

CELLULAR AUTOMATA

2

Cellular Automata (CA) are a particular class of parallel computational models. In its simplest form, a cellular automaton is composed of a grid of *cells*, where each cell applies a *local rule* to its *neighborhood* in order to compute its next state. The CA *global state* is the configuration of the states of all cells composing the grid at a given moment, and the dynamical behavior of the CA is determined by making all cells update in parallel in discrete time steps.

Despite the apparent simplicity of this model, CA are able to produce quite complex dynamic behaviors depending on the underlying local rule. As a matter of fact, by looking at the existing literature it is possible to remark that CA have been used to analyze and simulate a wide variety of discrete complex systems, including for instance physical phenomena [192], biological processes [170], ecosystems [6] and social dynamics [7].

Besides their applications in several scientific domains, CA have also been thoroughly investigated by mathematicians and computer scientists as a computational model and dynamical system *per se*. Originally, CA were introduced by Ulam [185] and von Neumann [188] respectively to model the growth of crystals and self-reproduction mechanisms in biology. Later, Hedlund [76] investigated CA from the point of view of dynamical systems theory, characterizing them as endomorphisms of the *full shift space*, the basic mathematical object studied in *symbolic dynamics*. Much of the interest in CA spawned however in the 70's when Gardner [66] popularized Conway's *Game of Life*, a two-dimensional CA where each cell becomes "dead" or "alive" depending on how many of its neighbors are alive. Successively, Wolfram [194] studied the computational properties of CA and classified them in four classes of increasing complexity depending on their dynamical behavior over finite grids. For a more complete survey about CA, we refer the reader to [83, 161, 87, 1, 34, 40].

In this chapter, we cover all the necessary background about CA used throughout the thesis. We begin in Section 2.1 by introducing the generic model of infinite one-dimensional CA based on symbolic and topological dynamics, reviewing the fundamental Curtis-Hedlund-Lyndon theorem that characterizes CA as endomorphisms of the full shift space. We then define in Section 2.2 the finite models of CA which we will diffusely investigate in the rest of this thesis from the cryptographic and combinatorial designs perspectives, namely *No Boundary CA* (NBCA) and *Periodic Boundary* CA (PBCA). Due to their importance in cryptographic applications, in Section 2.3 we finally

make some remarks about injectivity and surjectivity in CA, pointing out the relationships among the various CA models with respect to these properties.

2.1 INFINITE CA

Let Σ be a finite alphabet with $q \in \mathbb{N}$ symbols, and let Σ^n, Σ^* and $\Sigma^{\mathbb{Z}}$ respectively denote the set of all words over Σ of length $n \in \mathbb{N}$, the set of all finite words over Σ and the set consisting of all bi-infinite words over Σ. A *bi-infinite word* or *configuration* $x \in \Sigma^{\mathbb{Z}}$ can be considered as a function $x : \mathbb{Z} \to \Sigma$ assigning to each integer number a symbol from Σ. Given $x \in \Sigma^{\mathbb{Z}}$ and $i, j \in \mathbb{Z}$ such that $i \leqslant j$ and $j - i + 1 = n$, by $x_{[i,j]}$ we denote the finite block $(x_i, \cdots, x_j) \in \Sigma^n$. For $k \in \mathbb{N}$, the k-*left shift operator* $\sigma^k : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ shifts each symbol of a configuration $x \in \Sigma^{\mathbb{Z}}$ by k places to the left, i. e. for all $i \in \mathbb{Z}$ the i-th component of $\sigma^k(x)$ is defined as $\sigma^k(x)_i = x_{i+k}$. If $k = 1$, we simply write $\sigma(x)$, and refer to it as the *shift operator* in place of the 1-left shift operator. The *full shift space* is the pair $\langle \Sigma^{\mathbb{Z}}, \sigma \rangle$, that is, the set of all bi-infinite configurations over Σ together with the shift operator. Usually, this structure is considered as a metric space by endowing it with the *Cantor distance* $d_C : \Sigma^{\mathbb{Z}} \times \Sigma^{\mathbb{Z}} \to \mathbb{R}$, defined for all $x, y \in \Sigma^{\mathbb{Z}}$ as:

$$d_C(x,y) = \begin{cases} 0 & , \text{ if } x = y \\ 2^{-i} & , \text{ if } x \neq y,\ i = \max\{j \in \mathbb{N} : x_{-j} \neq y_{-j} \vee x_j \neq y_j\} \end{cases} . \tag{1}$$

Intuitively, under this metric two configurations are close to each other if they agree on a large block centered on the origin. Given $r \in \mathbb{R}^+$ the *open ball* $B_r(x)$ centered on $x \in \Sigma^{\mathbb{Z}}$ of radius r is the set of all configurations with distance less than r from x, that is

$$B_r(x) = \{y \in \Sigma^{\mathbb{Z}} : d_C(x,y) < r\} . \tag{2}$$

An *open set* $U \subseteq \Sigma^{\mathbb{Z}}$ is defined as a union of open balls. The family $\tau = \{U \subseteq \Sigma^{\mathbb{Z}}\}$ of all open sets in $\Sigma^{\mathbb{Z}}$ under the distance d_C satisfies the axioms of a *topology*, meaning that $\varnothing, \Sigma^{\mathbb{Z}} \in \tau$ and that τ is closed under finite intersections and (of course) arbitrary unions. This family is also known as the *Cantor topology* of the full shift space, since it is induced by the Cantor distance. Among the various properties of the Cantor topology, one of the most useful is that it is *compact*: every subfamily of open sets $\mathcal{U} \subseteq \tau$ whose union gives the whole space $\Sigma^{\mathbb{Z}}$ (i. e. an *open cover* of $\Sigma^{\mathbb{Z}}$) admits a finite subcover. Alternatively, one may define compactness in terms of limits by stating that every sequence of bi-infinite configurations admits a converging subsequence.

A function F from $\Sigma^{\mathbb{Z}}$ to itself is *continuous* if for all $x \in \Sigma^{\mathbb{Z}}$ and $\epsilon > 0$ there exists a $\delta > 0$ such that for all y with $d_C(x,y) < \delta$ it holds that $d_C(F(x), F(y)) < \epsilon$. By the *Heine-Cantor* theorem, a consequence of the compactness of $\Sigma^{\mathbb{Z}}$ under the Cantor topology is that

every continuous function is also *uniformly continuous*, which means that for all $\epsilon > 0$ there exists a $\delta > 0$ such that for each pair $x, y \in \Sigma^{\mathbb{Z}}$ with $d_C(x, y) < \delta$ it results that $d_C(F(x), F(y)) < \epsilon$. Remark the difference with the previous definition of continuity, which concerns the relationship between δ, ϵ and x: in a uniform continuous function, δ does not depend on the particular configuration x, but just on ϵ. For this reason, with a slight abuse of terminology we will use "continuous" instead of "uniformly continuous" throughout the rest of this thesis, unless ambiguities arise.

We are now ready to formally introduce the definition of infinite one-dimensional CA:

Definition 1. *An* infinite one-dimensional cellular automaton *(CA) is a quintuple* $A = \langle \Sigma, d, \omega, f, F \rangle$*, where:*

- Σ *is a finite set called the* state alphabet*.*
- $d \in \mathbb{N}$ *is the* diameter*.*
- $\omega \in \{0, \cdots, d-1\}$ *is the* offset*.*
- $f : \Sigma^d \to \Sigma$ *is the* local rule*.*
- $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ *is the* global rule *defined for all* $x \in \Sigma^{\mathbb{Z}}$ *and* $i \in \mathbb{Z}$ *as:*

$$F(x)_i = f(x_{[i-\omega, i-\omega+d-1]}) = f(x_{i-\omega}, x_{i-\omega+1}, \cdots, x_{i-\omega+d-1}) \ . \tag{3}$$

Hence, a CA can be seen as a bi-infinite array of cells (also called the *cellular array*), where each cell i updates its state by computing local rule f on the *neighborhood* $\nu = x_{[i-\omega, i-\omega+d-1]}$ formed by itself, the ω cells to its left and the $d - \omega - 1$ cells to its right.

With respect to the neighborhood, the CA literature usually considers the symmetric case where $d = 2r + 1$ and $\omega = r$, with $r \in \mathbb{N}$ being called the CA *radius*. Regarding the alphabet, the case which is most commonly investigated is the *Boolean* (or *binary*) one, where $\Sigma = \{0, 1\}$. *Elementary CA* (ECA) are the class of CA with Boolean alphabet and symmetric neighborhood with radius $r = 1$.

Since the local rule f of a CA is defined over the finite set Σ^d, it can be described by a *rule table* of q^d entries. Thus, once a particular ordering of the q^d elements of Σ^d is fixed, each row of the table corresponding to an input $x \in \Sigma^d$ can be filled with the respective value of $f(x)$. Unless otherwise specified, in this thesis we assume that the elements of Σ^d are always *lexicographically ordered*. Hence, given a total order $\leqslant_\Sigma$ on the alphabet symbols, for all $x, y \in \Sigma^d$ it holds $x \leqslant y$ if and only if $x_i \leqslant y_i$ for all $i \in \{1, \cdots, d\}$. As the value of f ranges over Σ, it means that one can fill a rule table in q^{q^d} different ways. This is also equivalent to the number of all possible local rules over Σ with diameter d. When Σ is the binary alphabet, the local rule is actually a *Boolean function*, and the rule table corresponds

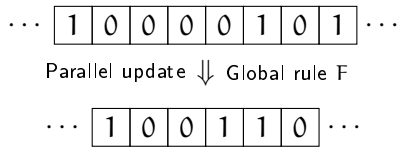


Figure 1: Example of ECA with local rule 150.

to the classic *truth table* representation (see Chapter 5). In this case, another common way used in the CA literature to identify local rules is by means of their *Wolfram code* [193], which basically amounts to the decimal encoding of the output column in the truth table. As an example, consider the CA in Figure 1, depicting the application of the global rule of an ECA on a portion of the cellular array. The local rule is defined as $f(x_{i-1}, x_i, x_{i+1}) = x_{i-1} \oplus x_i \oplus x_{i+1}$ for all $i \in \mathbb{Z}$, with $\oplus$ denoting the XOR operation. This corresponds to rule 150 using Wolfram's encoding.

Beside the symmetric case, in this thesis we will also consider the following neighborhoods:

- The *asymmetric* neighborhood, with $d = 2r$ and $\omega = r - 1$. In this case, each cell will look at itself, the $r - 1$ cells to its left and the r cells to its right to compute its next state. This is just a convention to accommodate the concept of radius also to the case of even diameters.
- The *one-sided* neighborhood, with $d \in \mathbb{N}$ and $\omega = 0$. Thus, each cell applies the local rule on itself and the $d - 1$ cells to its right.

Regarding the nature of the state alphabet, we will consider also the following situations beyond the binary case:

- $\Sigma = \mathbb{F}_q$, i. e. the alphabet is the *finite field* of order $q = p^\alpha$, where p is a prime number and $\alpha \in \mathbb{N}$.
- $\Sigma = \mathbb{Z}_m$, i. e. the alphabet is the *residue class ring* modulo $m \in \mathbb{N}$.

The Boolean alphabet can be seen as a particular case of both finite field and residue class ring alphabets, that is $\{0, 1\} = \mathbb{F}_2$ or $\{0, 1\} = \mathbb{Z}_2$. We will often focus on the former situation by endowing the binary alphabet with a field structure, due to the connections with Boolean functions (see Chapter 5).

The global rule of a CA can be seen as a function from the full shift space to itself. One of the seminal results in CA theory is the *Curtis-Hedlund-Lyndon* theorem[1], which gives a topological characterization of CA global rules:

Theorem 1 (Curtis-Hedlund-Lyndon). *Let $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ be a function from the full shift space to itself. Then, F is the global rule of a CA if and*

1 In the relevant literature, this result is often mentioned just as Hedlund's theorem. However, Hedlund [76] already credited Curtis and Lyndon as co-discoverers.

only if F *is continuous with respect to the Cantor distance and commutes with the shift operator, that is* $\sigma(F(x)) = F(\sigma(x))$ *for all* $x \in \Sigma^{\mathbb{Z}}$.

The property of commuting with the shift operator is also referred to as *shift-invariance*. Thus, if a function $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ is shift-invariant and continuous, then it is defined by a local rule f of a certain diameter d, and vice-versa. We do not delve into the topological consequences of the Curtis-Hedlund-Lyndon theorem, for which we refer the reader to [102, 88]. However, we will extensively use the shift-invariance property of CA to prove our results. Additionally, since a global rule uniquely defines the corresponding CA, in what follows we will often identify a CA only by its global rule instead of using the full quintuple specification of Definition 1.

We mentioned in the introduction of this section that CA are mostly studied as dynamical systems. Starting from an initial configuration of the cellular array, the *dynamical behavior* of a CA is determined by the iterated application of its global rule. Given a CA $A = \langle \Sigma, d, \omega, f, F \rangle$, an initial configuration $x \in \Sigma^{\mathbb{Z}}$ and a time $t \in \mathbb{N}$, by $x(t) = F^t(x)$ we denote the configuration of the array obtained after computing the t*–th iterate* of F, i. e. after applying the global rule F starting from x for t steps. Notice that $x(0) = x$. The *dynamical evolution* (also called the *orbit*) of A starting from x is the sequence of configurations $\{x(t)\}_{t\in\mathbb{N}}$. For all $i \in \mathbb{Z}$, the *trace* $\{x_i(t)\}_{t\in\mathbb{N}}$ is the sequence of states taken by cell x_i at time $t = 0, 1, 2, \cdots$.

2.2 FINITE CA

For practical applications, CA can obviously be implemented by using finite arrays only. This leads to the problem of updating the cells at the boundaries, since they do not have enough neighbors upon which the local rule can be applied. There are several ways to address this issue, two of which we consider in this thesis are *No Boundary CA* (NBCA) and *Periodic Boundary CA* (PBCA).

The NBCA approach is the simplest one to cope with boundary conditions: given an array of $n \geqslant d$ cells, where d is the diameter, one applies the local rule only to the $n - d + 1$ cells which have enough left and right neighbors, ignoring the remaining $d - 1$ cells at the boundaries. This means that the output of the global rule will be smaller than the input, so that the size of the cellular array "shrinks" by $d - 1$ cells. Since the resulting global rule is a finite vectorial function, one can assume without loss of generality that the offset is null, i. e. that the CA has a one-sided neighborhood with $\omega = 0$. This leads to the following definition:

Definition 2. *Let* Σ *be a finite alphabet and* $n, d \in \mathbb{N}$ *with* $n \geqslant d$. *Additionally, let* $f : \Sigma^d \to \Sigma$ *be a local rule of* d *variables. The* No Boundary

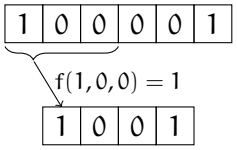


Figure 2: NBCA

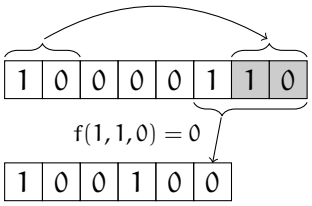


Figure 3: PBCA

Figure 4: Examples of NBCA and PBCA with local rule 150.

Cellular Automaton *(NBCA)* $F : \Sigma^n \to \Sigma^{n-d+1}$ *is the function defined for all* $x \in \Sigma^n$ *as*

$$F(x_1, \cdots, x_n) = (f(x_1, \cdots, x_d), f(x_2, \cdots, x_{d+1}), \cdots, f(x_{n-d+1}, \cdots, x_n)) \ . \tag{4}$$

On the other hand, the PBCA approach considers the finite cellular array as a ring, with the first cell following the last one. Hence, the ω last cells will use the $d - \omega - 1$ first ones as their right neighbors, while vice versa the $d - \omega - 1$ first cells will use the last ω as their left neighbors. Notice that in this case the size of the cellular array does not shrink by applying the global rule, since all cells are updated. As in the case of NBCA, one may assume $\omega = 0$ without loss of generality. We formalize the above discussion through this definition:

Definition 3. *Let* Σ *be a finite alphabet and* $n, d \in \mathbb{N}$. *Additionally, let* $f : \Sigma^d \to \Sigma$ *be a local rule of* d *variables. The* Periodic Boundary Cellular Automaton *(PBCA)* $F : \Sigma^n \to \Sigma^n$ *is the function defined for all* $x \in \Sigma^n$ *as*

$$F(x_1, \cdots, x_n) = (f(x_1, \cdots, x_d), f(x_2, \cdots, x_{d+1}), \cdots, f(x_n, \cdots, x_{d-1})) \ . \tag{5}$$

Figure 4 reports examples respectively of an NBCA and a PBCA with $n = 6$ cells and local rule 150.

Remark that in both Definitions 2 and 3 we only defined the global rules of the CA. This is because in this thesis we focus only on the *short-term behavior* of finite CA yielded by just one application of their global rules, instead of their asymptotical behavior induced by iterating the global rules for several time steps. This has a straightforward motivation in the no boundary approach, since as the size of the cellular array shrinks the global rule can be applied only for a finite number of steps. On the other hand, it is also easy to see that the asymptotical dynamic behavior of PBCA is trivially periodic, since for all sizes $n \in \mathbb{N}$ of the array the CA will return to its initial configuration after at most q^n steps, where $q = |\Sigma|$.

Considering the Curtis-Hedlund-Lyndon theorem, one can see that the shift-invariance property is sufficient to characterize the global rules of both NBCA and PBCA. In particular, it suffices to replace the

classic shift operator that acts on bi-infinite configurations with the *cyclic shift* σ_c, defined for all $x \in \Sigma^n$ as

$$\sigma_c(x) = \sigma_c(x_1, x_2, \cdots, x_{n-1}, x_n) = (x_2, x_3, \cdots, x_n, x_1) \ . \tag{6}$$

The continuity property, on the other hand, is always satisfied since any function between finite metric spaces is trivially continuous. For all $n \in \mathbb{N}$, both NBCA and PBCA thus coincide with the class of shift-invariant functions over Σ^n. This is due to the fact that, given a generic shift-invariant function over Σ^n, one can always define a local rule of diameter equal to the cellular array size, i. e. $d = n$. Similarly, in the infinite setting one needs continuity beside shift-invariance to characterize CA global rules, since a generic map $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ that commutes with the shift could be defined by a local rule whose output depends on arbitrarily far cells.

We finally remark the relation between PBCA and infinite CA restricted to the space of *spatially periodic configurations* (SPC), which we formally define as follows:

Definition 4. *A configuration* $x \in \Sigma^{\mathbb{Z}}$ *is* spatially periodic *if there exists* $p \in \mathbb{N}$, *with* $p \neq 0$, *such that* $\sigma^p(x) = x$. *In particular, such a* p *is called a* period *of* x. *The smallest integer* $p \in \mathbb{N}$ *among all periods of* x *is called the* least period *of* x.

Following the notation of [135], we denote by $x = {}^{\omega}u^{\omega}$ the SPC $x \in \Sigma^{\mathbb{Z}}$ obtained as the bi-infinite concatenation of block $u \in \Sigma^*$ with itself. Setting $p = |u|$ as the length of u, one clearly has that p is a period of x. If u is in turn aperiodic, then $p = p$ is the least period of x. Otherwise, the least period of x will be a divisor of p.

For all $n \in \mathbb{N}$, let $\Sigma^{\mathbb{Z}}_n$ denote the set of all SPC of least period n. The space $\Sigma^{\mathbb{Z}}_p$ of all spatially periodic configurations is defined as

$$\Sigma^{\mathbb{Z}}_p = \bigcup_{n \in \mathbb{N}} \Sigma^{\mathbb{Z}}_n \ . \tag{7}$$

The restriction of an infinite CA $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ to the set of SPC is denoted as F_p. For all $n \in \mathbb{N}$, let us now define the following set:

$$\Sigma^{\mathbb{Z}}_{d|n} = \bigcup_{d : d|n} \Sigma^{\mathbb{Z}}_d \ . \tag{8}$$

In other words, $\Sigma^{\mathbb{Z}}_{d|n}$ is the union of all spatially periodic configurations of least period d that divides n. By $F_{d|n}$ we denote the restriction of an infinite CA F defined by a local rule f to the set $\Sigma_{d|n}$. It is not difficult to see that $|\Sigma^{\mathbb{Z}}_{d|n}| = |\Sigma|^n$. In particular, one can observe that $F_{d|n}$ is actually equivalent to the bi-infinite concatenation of a PBCA $\tilde{F}$ of length n having the same local rule f that defines F. In particular, for all $u \in \Sigma^n$ and $x = {}^{\omega}u^{\omega} \in \Sigma^{\mathbb{Z}}_{d|n}$, the following relation holds:

$$F_{d|n}(x) = {}^{\omega}\tilde{F}(u)^{\omega} \ . \tag{9}$$

2.3 INJECTIVITY AND SURJECTIVITY IN CA

We now discuss the injectivity and surjectivity properties of CA, due to their importance for cryptographic applications. Basically, an infinite CA is injective (respectively, surjective) if and only if its global rule F is injective (respectively, surjective). This definition can be translated as is for PBCA, while of course in the case of NBCA there is only the notion of surjectivity, since the output space of the global rule is smaller than the input space.

One of the basic results in the theory of one-dimensional CA is that injectivity is equivalent to bijectivity, as shown in the following theorem proved by Hedlund [76]:

Theorem 2. *An injective infinite CA $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ is also surjective.*

On the other hand, surjectivity in infinite CA can be characterized by the *balancedness* property, meaning that for all $n \in \mathbb{N}$ greater than or equal to the CA diameter the sets of preimages of the corresponding NBCA with n cells all have the same cardinality [76]:

Lemma 1. *Let $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ be a CA defined by a local rule $f : \Sigma^d \to \Sigma$. Then, F is surjective if and only if the NBCA $F_n : \Sigma^n \to \Sigma^{n-d+1}$ based on local rule f is* balanced *for all $n \geqslant d$, that is, for all $u \in \Sigma^{n-d+1}$ it results that $|F_n^{-1}(u)| = q^{d-1}$, where $q = |\Sigma|$. Additionally, if F is surjective $|F^{-1}(y)| \leqslant q^{d-1}$ holds for all $y \in \Sigma^{\mathbb{Z}}$.*

A useful tool to study injectivity and surjectivity in CA is the *de Bruijn graph* representation. Given a finite alphabet Σ and $t \in \mathbb{N}$, the corresponding de Bruijn graph has vertex set Σ^t, and there exists a directed edge from $w_1 \in \Sigma^t$ to $w_2 \in \Sigma^t$ if and only if $w_1 = ax$ and $w_2 = xb$, where $a, b \in \Sigma$ and $x \in \Sigma^{t-1}$. In other words, two vertices are connected if and only if their respective words *overlap* respectively on the rightmost and the leftmost $t-1$ symbols. De Bruijn graphs have several applications beyond their use in the CA literature, such as designing *maximal length periodic sequences* in combinatorics [24, 183] and *de novo sequence assembly* in bioinformatics [198, 8, 9].

For the purposes of this thesis, we formally define de Bruijn graphs in terms of *fusion operators*, following the notation of Sutner [174]. Given $s \in \mathbb{N}$ and $u, v \in A^*$ such that $|u| \geqslant s$ and $|v| \geqslant s$, the *s-fusion operator* $\odot$ is defined as follows:

$$u \odot v = z \Leftrightarrow \exists x \in A^s,\ u_0, v_0 \in A^* : u = u_0 x,\ v = x v_0,\ z = u_0 x v_0$$

that is, z is obtained by overlapping the right part of u and the left part of v of length s.

We then give the following definition of de Bruijn graph associated to a CA:

Definition 5. *Let $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ be a CA defined by a local rule $f : \Sigma^d \to \Sigma$ of diameter d. The* de Bruijn graph *associated to F is the directed labeled graph $G_{DB}(f) = (V, E, l)$ defined as follows:*

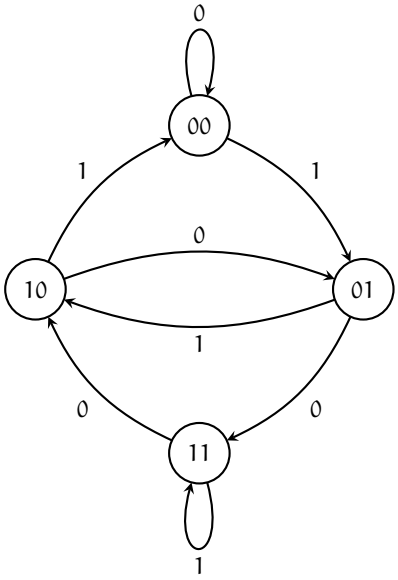


Figure 5: de Bruijn graph associated to the ECA F defined by rule 150.

- $V = \Sigma^{d-1}$
- *Given* $v_1, v_2 \in V$, $(v_1, v_2) \in E$ *if and only if there exists* $z \in \Sigma^d$ *such that* $z = v_1 \odot v_2$, *where* $\odot$ *denotes the* s*-fusion operator with* $s = d - 2$
- *For all* $(v_1, v_2) \in E$, *the label function* $l : E \to \Sigma$ *is defined as* $l(v_1, v_2) = f(v_1 \odot v_2)$

Figure 5 reports the de Bruijn graph associated to the ECA with local rule 150.

One can observe that a bi-infinite path $p_v = \{v_i : v_i \in V\}_{i \in \mathbb{Z}}$ on the *vertices* of the de Bruijn graph associated to an infinite CA F can be used to specify a configuration $x \in \Sigma^{\mathbb{Z}}$ by using the fusion operator:

$$x = \bigodot_{i \in \mathbb{Z}} v_i \tag{10}$$

Further, since the CA local rule is represented as a labeling of the edges, the bi-infinite path $p_e = \{l(v_i, v_{i+1}) : v_i, v_{i+1} \in p_v\}_{i \in \mathbb{Z}}$ on the labels of the edges of $G_{DB}(f)$ corresponds to the image of x under the global rule F, i. e.

$$y = p_e = F(x) \ . \tag{11}$$

This allows us to characterize injectivity and surjectivity of an infinite CA F by means of its de Bruijn graph $G_{DB}(f)$ as follows:

- F is injective if and only if for every distinct pair $p_{v_1} \neq p_{v_2}$ of bi-infinite paths on the vertices of $G_{DB}(f)$ the corresponding paths on the labelings of the edges p_{e_1} and p_{e_2} are distinct as well.

- F is surjective if and only if every configuration $y \in \Sigma^{\mathbb{Z}}$ there exists at least one bi-infinite path p_e on the edges of $G_{DB}(f)$ labeled by y.

Clearly, these characterizations can be straightforwardly translated also in the finite case for NBCA and PBCA.

One of the main classes of CA studied in this thesis consists of *bipermutive* CA, defined as follows:

Definition 6. *A CA $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ induced by a local rule $f : \Sigma^d \to \Sigma$ is called* left permutive *(respectively,* right permutive*) if, for all $z \in \Sigma^{d-1}$, the restriction $f_{R,z} : \Sigma \to \Sigma$ (respectively, $f_{L,z} : \Sigma \to \Sigma$) obtained by fixing the first (respectively, the last) $d-1$ coordinates of f to the values specified in z is a permutation on Σ. A CA which is both left and right permutive is said to be a* bipermutive *CA (BCA).*

Remark that when $\Sigma = \{0,1\}$ is the Boolean alphabet a local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$ is left permutive if and only if there exists a *generating function* $\varphi : \mathbb{F}_2^{d-1} \to \mathbb{F}_2$ such that

$$f(x_1, x_2, \cdots, x_d) = x_1 \oplus \varphi(x_2, \cdots, x_d) \ , \tag{12}$$

and symmetrically for right permutive rules. In fact, permutations over $\mathbb{F}_2$ can be constructed only through the XOR operator. Hence, one can see that the elementary rule 150 used up to now in our examples is bipermutive, since it is defined as the XOR of the three cells in the neighborhood.

Bipermutivity may also be expressed in terms of the de Bruijn graph representation, by interpreting $G_{DB}(f)$ as a *finite state automaton*. To this end, if $l(v_1, v_2) = x$, define the transition function as $\delta(v_1, x) = v_2$. Then, F is bipermutive if and only if for all $v_1, v_2 \in V$ with $v_1 \neq v_2$ and for all $x \in A$, it holds that $\delta(v_1, x) \neq \delta(v_2, x)$, i. e. the de Bruijn graph is a *permutation automaton*.

This observation leads us to deduce that infinite bipermutive CA are surjective. As a matter of fact, it is easy to see that for every configuration $y \in \Sigma^{\mathbb{Z}}$ one can always find a path on the edges labeled by y. Since the de Bruijn graph in this case is a permutation automaton, this means in particular that one can construct a preimage of y starting from any vertex $v \in \Sigma^{d-1}$, the reason being that the sets of labels on the outgoing and ingoing edges of v are permutations of Σ. Bipermutive infinite CA thus satisfy a stronger balancedness condition than the one stated in Lemma 1, since every configuration $y \in \Sigma^{\mathbb{Z}}$ has *exactly* q^{d-1} preimages.

Another interesting property of surjectivity in CA is its closure property when considering the restriction of the global rules to the set of spatially periodic preimages, as the next result proved in [57] shows:

Lemma 2. *Let $F : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ be a surjective CA. Then, given a SPC $y \in A^{\mathbb{Z}}$, each preimage $x \in F^{-1}(y)$ is also spatially periodic.*

As a matter of fact, surjective CA satisfy an even stronger condition than the closure property implied by the Lemma above. In particular, the global map $F : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ of a CA is surjective if and only if its restriction F_p to the set of SPC is surjective (see [57]). As a consequence, one can study certain properties of surjective CA by considering only their restriction to SPC. In the model-theoretic setting proposed by Sutner [175], this means that the set of SPC is an *elementary substructure* of the full shift space $A^{\mathbb{Z}}$ for surjective CA.

We conclude this section by mentioning the class of *reversible* CA (RCA), which are particularly interesting for cryptographic applications. Formally, an infinite CA $A = \langle \Sigma, d, \omega, f, F \rangle$ is reversible if its global rule $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ is bijective and the inverse $G = F^{-1}$ is again the global rule of an infinite CA, i. e. it is defined by a local rule $f' : \Sigma^{d'} \to \Sigma$ of a certain diameter d'. In practice, as the next theorem proved in [76, 154] shows, the reversibility of an infinite CA is characterized just by the bijectivity of its global rule:

Theorem 3. *Let* $A = \langle \Sigma, d, \omega, f, F \rangle$ *be an infinite CA with global rule* $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$. *Then,* F *is bijective if and only if its inverse function* $G = F^{-1}$ *is the global rule of a CA* $A' = \langle \Sigma, \delta', \omega', f', G \rangle$.

Notice that the above result does not say how to find the inverse global rule G. As a matter of fact, even characterizing the diameter of the inverse local rule is still an open problem [49].

Since an infinite RCA $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ is clearly reversible over the set of spatially periodic configurations $\Sigma^{\mathbb{Z}}_p$, it follows that for any $n \in \mathbb{N}$ the PBCA $F_n : \Sigma^n \to \Sigma^n$ is reversible as well. The converse is however not true: a local rule $f : \Sigma^d \to \Sigma$ may give rise to a PBCA whose global rule is invertible only for certain array lengths $n \in \mathbb{N}$, but the corresponding infinite CA is not reversible. As it will be discussed in Chapter 7, this is exactly the case for some CA used in real-world cryptographic applications.

LINEAR RECURRING SEQUENCES, COMBINATORIAL DESIGNS AND CODING THEORY

3

In this chapter, we recall the basic concepts about linear recurring sequences, combinatorial designs and coding theory that we will employ to prove the results in this thesis. Moreover, these notions will also be used as a basis in Chapters 4 and 5 respectively concerning cryptography and Boolean functions.

Section 3.1 discusses linear recurring sequences and linear feedback shift registers over finite fields, focusing on their periodicity properties. Section 3.2 describes the basic combinatorial designs considered in this thesis, namely Latin squares and orthogonal arrays, and describes the connections between them. Finally, Section 3.3 gives a brief overview of coding theory, introducing among other things linear cyclic codes and MDS codes.

3.1 LINEAR RECURRING SEQUENCES AND LINEAR FEEDBACK SHIFT REGISTERS

Sequences generated by recurrence relations play an important role in several applications, e. g. in the design of error-correcting codes in coding theory and pseudorandom number generators in cryptography. When the underlying set is the finite field $\mathbb{F}_q$ and the terms in the recurrence equation are related by a linear expression, the resulting sequence is also called a *linear recurring sequence* (LRS). Several properties of LRS (such as number of cycles of a certain periods, fixed points, etc.) are characterizable by means of linear algebraic methods.

We now cover the fundamental definitions and results about LRS that we will use in the rest of this work. All the proofs of the facts and the theorems mentioned in this section may be found in [107].

Definition 7. *Given* $k \in \mathbb{N}$ *and* $a,\ a_0,\ a_1,\ \cdots,\ a_{k-1} \in \mathbb{F}_q$*, a* linear recurring sequence *(LRS) of order* k *is a sequence* $s = s_0, s_1, \cdots$ *of elements in* $\mathbb{F}_q$ *which satisfies the following relation:*

$$s_{n+k} = a + a_0 s_n + a_1 s_{n+1} + \cdots + a_{k-1} s_{n+k-1} \quad \forall n \in \mathbb{N} \ . \tag{13}$$

The terms $s_0,\ s_1,\ \cdots,\ s_{k-1}$ which uniquely determine the rest of the LRS are called the *initial values* of the sequence. If $a = 0$ the sequence is called *homogeneous*, otherwise it is called *inhomogeneous*. In what follows, we will only deal with homogeneous LRS.

A linear recurring sequence can be generated by a device called *linear feedback shift register* (LFSR), depicted in Figure 6. Basically, a LFSR of order k is composed of k *delayed flip-flops* $D_0,\ D_1,\ \cdots,\ D_{k-1}$,

each containing an element of $\mathbb{F}_q$. At each step $n \in \mathbb{N}$, the elements $s_n,\ s_{n+1},\ \cdots,\ s_{n+k-1}$ in the flip-flops are shifted one place to the left, and D_{k-1} is updated with the linear combination defined in Equation (13). This kind of linear feedback shift registers are also called *Fibonacci* LFSR, as opposed to *Galois* LFSR where the adders are placed between one flip-flop and the other.

Notice that the output of a LFSR (that is, the LRS $s = s_0, s_1, \cdots$) is *ultimately periodic*, i. e. there exist $p, n_0 \in \mathbb{N}$ such that for all $n \geqslant n_0$, $s_{n+p} = s_n$. In fact, for all $n \in \mathbb{N}$ the state of the LFSR is completely described by the vector $(s_n, s_{n+1}, \cdots, s_{n+k-1})$. Since all the components of such a vector take values in $\mathbb{F}_q$, which is a finite set of q elements, after at most q^k shifts the initial value of the vector will be repeated. In particular, in [107] it is proved that if $a_0 \neq 0$, then the sequence produced by the LFSR (or, equivalently, the corresponding LRS) is *periodic*, in the sense of Definition 4.

The *characteristic polynomial* $a(x) \in \mathbb{F}_q[x]$ of a k-th order homogeneous LRS $s = s_0, s_1, \cdots$ is defined as:

$$a(x) = x^k - a_{k-1}x^{k-1} - a_{k-2}x^{k-2} - \cdots - a_0 \ . \tag{14}$$

The *multiplicative order* of the characteristic polynomial, denoted by $\mathrm{ord}(a(x))$, is the least integer e such that $a(x)$ divides $x^e - 1$, and it can be used to characterize the period of s. In fact, in [107] it is shown that if $a(x)$ is irreducible over $\mathbb{F}_q$ and $a(0) \neq 0$, then the period p of s equals $\mathrm{ord}(a(x))$, while in the general case where $a(x)$ is reducible $\mathrm{ord}(a(x))$ divides p.

A common way to represent a LRS $s = s_0, s_1, \cdots$ is through its *generating function* $G(x)$, which is the formal power series defined as:

$$G(x) = s_0 + s_1 x + s_2 x^2 + \cdots = \sum_{n=0}^{\infty} s_n x^n \tag{15}$$

In this case, the terms $s_0, s_1, \cdots$ are called the *coefficients* of $G(x)$. The set of all generating functions over $\mathbb{F}_q$ can be endowed with a ring structure in which sum and product are respectively pointwise addition and convolution of coefficients. The *fundamental identity of formal power series* states that the generating function $G(x)$ of a k-th order homogeneous LRS s can be expressed as a rational function:

$$G(x) = \frac{g(x)}{a^*(x)} = \frac{-\sum_{j=0}^{k-1} \sum_{i=0}^{j} a_{i+k-j} s_i x^j}{x^k a(1/x)} \ . \tag{16}$$

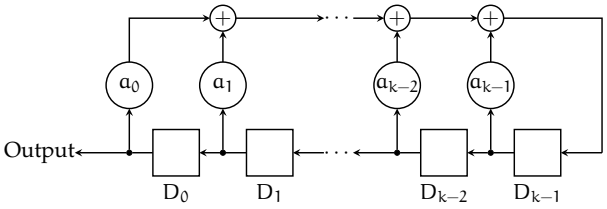


Figure 6: Diagram of a linear feedback shift register.

where $g(x)$ is the *initialization polynomial*, which depends on the k initial terms of sequence s (where $a_k = -1$), while $a^*(x) = x^k a(1/x)$ denotes the *reciprocal characteristic polynomial* of s.

A given LRS $s = s_0, s_1, \cdots$ over $\mathbb{F}_q$ satisfies several linear recurrence equations. Hence, several characteristic polynomials can be associated to s, one for each recurrence equation which s satisfies. The *minimal polynomial* $m(x)$ associated to s is the characteristic polynomial which divides all other characteristic polynomials of s, and it can be computed as follows:

$$m(x) = \frac{a(x)}{\gcd(a(x), h(x))} \ , \tag{17}$$

where $a(x)$ is a characteristic polynomial of s and $h(x) = -g^*(x)$ is the reciprocal of the initialization polynomial $g(x)$ appearing in Equation (16), with the sign changed. In [107] it is proved that the period of s equals the order of its minimal polynomial $m(x)$.

In order to study the periods of preimages of LBCA, we also need some results about families of linear recurring sequences. Denote by $S(f(x))$ the set of LRS having $f(x)$ as characteristic polynomial. Given $s = s_0, s_1 \cdots \in S(f(x))$ and $t = t_0, t_1, \cdots \in S(f(x))$ define the sum of LRS $\sigma = s + t$ as $\sigma_n = s_n + t_n$ for all $n \in \mathbb{N}$, and for $c \in \mathbb{F}_q$ define the scalar multiplication $\mu = c \cdot s$ as $\mu_n = c \cdot s_n$ for all $n \in \mathbb{N}$. Under these two operations, the set $S(f(x))$ is a vector space over $\mathbb{F}_q$. The following theorem shows what is the characteristic polynomial of the direct sum of two families of LRS:

Theorem 4. *Let $f_1(x), f_2(x) \in \mathbb{F}_q$ be non-constant monic polynomials, and let $S(f_1(x))$ and $S(f_2(x))$ be the families of LRS whose characteristic polynomials are respectively $f_1(x)$ and $f_2(x)$. Denoting by $S(f_1(x)) + S(f_2(x))$ the family of all LRS $\sigma + \tau$ where $\sigma \in S(f_1(x))$ and $\tau \in S(f_2(x))$, it follows that $S(f_1(x)) + S(f_2(x)) = S(c(x))$, where $c(x)$ is the least common multiple of $f_1(x)$ and $f_2(x)$.*

From Theorem 4, the following result states how to compute the least periods of the sum of two LRS in the special case when their characteristic polynomials are coprime:

Theorem 5. *Let σ_1 and σ_2 be two homogeneous LRS having minimal polynomials $m_1(x), m_2(x) \in \mathbb{F}_q[x]$ and periods $p_1, p_2 \in \mathbb{N}$, respectively. If $m_1(x)$ and $m_2(x)$ are relatively prime, then the minimal polynomial $m(x) \in \mathbb{F}_q[x]$ of the sum $\sigma = s + t$ is equal to $m_1(x) \cdot m_2(x)$, while the least period of σ is the least common multiple of p_1 and p_2.*

Finally, the following theorem characterizes the multiplicities of the least periods in $S(f(x))$ when $f(x)$ is the power of an irreducible polynomial:

Theorem 6. *Let $f(x) = g(x)^t$ with $g(x)$ monic and irreducible over $\mathbb{F}_q$ and such that $g(0) \neq 0$, $\deg(g(x)) = k$, $\mathrm{ord}(g(x)) = e$, and $t \in \mathbb{N}$ a positive*

integer. Let $s \in \mathbb{N}$ *be the smallest integer such that* $\rho^s \geqslant t$*, where* ρ *is the characteristic of* $\mathbb{F}_q$*. If* $t = 1$ *the family of LRS* $S(f(x))$ *is composed of the following numbers of sequences with the following least periods:*

- *one sequence of least period* 1
- $q^k - 1$ *sequences of least period* e

For $t \geqslant 2$*,* $S(f(x))$ *additionally contains the following numbers of sequences with the following least periods:*

- *for* $j \in \{1, \cdots, s-1\}$*,* $q^{k\rho^j} - q^{k\rho^{j-1}}$ *sequences of least period* $e\rho^j$
- $q^{kt} - q^{k\rho^s}$ *sequences of least period* $e\rho^s$

3.2 COMBINATORIAL DESIGNS, LATIN SQUARES AND ORTHOGONAL ARRAYS

Combinatorial design theory is a branch of combinatorics that studies sets and arrangements of objects satisfying certain *balancing* properties. Even though the first investigations in this discipline can be traced back at least to Euler [62], it was only the 20th century that combinatorial design theory developed as an independent field of research. The reason lies in the fact that the objects studied by combinatorial design theory have a wide variety of applications in several domains, ranging from the design of experiments in statistics to the development of error-correcting codes, whose interest mainly bloomed in the past few decades.

Combinatorial design theory is a vast field, and a complete treatment of the topic is clearly outside the scope of this thesis. In this section, we touch upon only the two combinatorial designs which will interest us in later chapters, namely *Latin squares* and *orthogonal arrays*. For further information on the subject, we refer the reader to [173, 46, 92]. In particular, in the rest of this section we adopt Stinson's notation [173].

In what follows, for all $N \in \mathbb{N}$ by $[N]$ we denote the set $\{1, \cdots, N\}$. We begin with the following definition:

Definition 8. *Let* $N \in \mathbb{N}$*. A* Latin square *of order* N *is a* $N \times N$ *square matrix* L *with entries from* $[N]$ *such that each row and each column is a permutation of* $[N]$*. Thus, for all* $i, j, k \in [N]$ *it holds that* $L(i, j) \neq L(i, k)$ *and* $L(j, i) \neq L(k, i)$*.*

Latin squares exist for all orders $n \in \mathbb{N}$. As a matter of fact, a simple construction is to take the vector $v = (1, 2, \cdots, N)$ as the first row of the matrix and then apply the cyclic shift σ_c for all subsequent rows. It easy to check that the resulting matrix is a Latin square. Figure 7 depicts the Latin square of order $N = 4$ constructed in this way.

1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

Figure 7: Example of Latin square of order $N = 4$.

Latin squares are related to algebraic structures called *quasigroups*, which we formally define below:

Definition 9. *A* quasigroup *of order* $N \in \mathbb{N}$ *is a pair* $\langle X, \circ \rangle$ *where* X *is a finite set of* N *elements, while* $\circ$ *is a binary operation over* X *such that for all* $x, y \in X$ *the two equations* $x \circ z = y$ *and* $z \circ x = y$ *admit a unique solution for all* $z \in X$.

As shown in [173], an algebraic structure $\langle X, \circ \rangle$ is a quasigroup if and only if its *Cayley table* is a Latin square.

Counting the number of Latin squares is an open problem for generic orders $n \in \mathbb{N}$ [46]. Several bounds and constructions are known, about which the reader can find further information in [92]. The numbers of Latin squares have been computed by exhaustive search up to order $N = 11$ [85]. A typical way to restrict the search space of possible Latin squares of a given order is to define equivalence relations on them, and then count or enumerate the equivalence classes in the resulting quotient space. One of the most common equivalence relations studied in Latin squares is *isotopy*: namely, two Latin squares L_1, L_2 of order N are isotopic if and only if there exist three permutations α, β, γ over $[N]$ respectively on the rows, the columns and the elements of $[N]$ which map L_1 to L_2. It is easy to check that this is indeed an equivalence relation. The equivalence classes in this case are also called *isotopy classes*.

We now turn to the *orthogonality property* of Latin squares, formally introduced below:

Definition 10. *Two matrices* L_1 *and* L_2 *of order* N *are called* orthogonal Latin squares *(OLS) if*

$$(L_1(i_1, j_1), L_2(i_1, j_1)) \neq (L_1(i_2, j_2), L_2(i_2, j_2)) \tag{18}$$

for all distinct pairs of coordinates $(i_1, j_1), (i_2, j_2) \in [N] \times [N]$. *Equivalently,* L_1 *and* L_2 *are orthogonal if their* superposition *yields all the ordered pairs in the Cartesian product* $[N] \times [N]$.

A set of k Latin squares which are pairwise orthogonal is denoted as a k-MOLS (the acronym standing for Mutually Orthogonal Latin Squares). Figure 8 reports an example of a pair of orthogonal Latin squares of order 4. Unlike in the case of Latin squares, OLS Latin

1	3	4	2
4	2	1	3
2	4	3	1
3	1	2	4

1	4	2	3
3	2	4	1
4	1	3	2
2	3	4	1

1,1	3,4	4,2	2,3
4,3	2,2	1,4	3,1
2,4	4,1	3,3	1,2
3,2	1,3	2,1	4,4

Figure 8: Orthogonal Latin squares of order $N = 4$, and their superposition.

squares do not exist for every possible order. In particular, Euler [62] conjectured that pairs of OLS do not exist for $N = 6$ and for all orders $N \equiv 2 \mod 4$. This conjecture was first proved for $N = 6$ by Tarry [178] at the beginning of the 20th century, basically using an exhaustive search method[1]. Several years later, Euler's general conjecture was disproved by Bose, Shrikande and Parker [19], who showed a construction of OLS for all orders $N \neq 6, 10$.

Given $N \in \mathbb{N}$, the maximum number for a set of mutually orthogonal Latin squares is $N - 1$. In this case, the set is called a *complete* set of MOLS. Complete sets of MOLS are equivalent to other geometrical objects, in particular *projective planes* and *affine planes* [46].

Another type of combinatorial designs which are closely related with orthogonal Latin squares are *orthogonal arrays*, which we define as follows:

Definition 11. *Let X be a finite set of v elements, and let t, k and λ be positive integers such that $2 \leqslant t \leqslant k$. A t–(v, k, λ)* orthogonal array *(t–(v, k, λ)–OA, for short) is a $\lambda v^t \times k$ rectangular matrix with entries from X such that, for any subset of t columns, every t–uple $(x_1, \cdots, x_t) \in X^t$ occurs in exactly λ rows.*

When $t = 2$ and $\lambda = 1$, the resulting orthogonal array is a $v^2 \times k$ matrix in which every pair of columns contains all ordered pairs of symbols from X. In this case, the orthogonal array is simply denoted as $OA(k, v)$, and it is equivalent to a set of $(k-2)$–MOLS. As a matter of fact, suppose that $L_1, \cdots, L_{k-2}$ are $(k-2)$–MOLS of order v. Without loss of generality, we can assume that $X = \{1, \cdots, v\}$. Then, consider a matrix A of size $v^2 \times k$ defined as follows:

- Columns 1 and 2 are filled with all ordered pairs $(i, j) \in X \times X$ arranged in lexicographic order.
- For $1 \leqslant i \leqslant v^2$ and $3 \leqslant h \leqslant k$, the entry (i, h) of A is defined as

$$A(i, h) = L_{h-2}(A(i, 1), A(i, 2)) \ . \tag{19}$$

1 The case of order $N = 6$ was also referred by Euler as the 36 *officers problem*: namely, given 36 officers coming from all possible combinations of 6 different ranks and 6 different regiments, arrange them on a 6×6 square such that no rank and no regiment is repeated on the same row or the same column.

In other words, column h is filled by reading the elements of the Latin square L_{h-2} from the top left down to the bottom right.

The resulting array is a $OA(k, v)$: indeed, let h_1, h_2 be two of its columns. If $h_1 = 1$ and $h_2 = 2$ one gets all the ordered pairs of symbols from X in lexicographic order. If $h_1 = 1$ (respectively, $h_1 = 2$) and $h_2 \geqslant 3$, one obtains all pairs because the h_1-th row (respectively, column) of L_{h_2-2} is a permutation over X. Finally, for $h_1 \geqslant 3$ and $h_2 \geqslant 3$ one still gets all ordered pairs since the Latin squares L_{h_1-2} and L_{h_2-2} are orthogonal. The reader can find further details about the inverse direction of the construction in [173].

3.3 ERROR-CORRECTING CODES

We now introduce the basic definitions and results related to error-correcting codes, focusing in particular on linear and cyclic codes. For a thorough treatment of the subject, the reader can refer to [117].

The theory of error-correcting codes studies the methods to encode messages transmitted by a sender over a noisy channel, so that (under specific assumptions) the receiver can correct eventual errors and retrieve the original message.

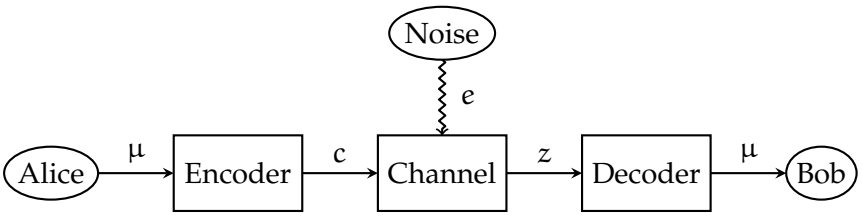


Figure 9: Communication model studied in coding theory.

Figure 9 represents the communication model usually considered in error-correcting code theory. Alice, the sender, wishes to send to Bob, the receiver, a message $\mu \in \Sigma^m$ of length m defined over a finite alphabet Σ. Since the channel is disturbed by some random noise, she uses and *encoder* to obtain the encoded message $c \in \Sigma^{m+r}$, which adds r *redundancy* symbols to protect it from eventual errors. After transmitting c over the channel, Bob receives a message $z \in \Sigma^{m+r}$, which could be the result of the noise over the channel that modified some symbols in c with a random *error pattern* $e \in \Sigma^{m+r}$. Bob thus feeds the received message z into a *decoder*, and if certain conditions are met (usually, that the number of errors occurred during transmission are below a specific threshold t) then the decoder output the original message μ.

In order to verify how many errors have occurred, one needs a way to measure how different are two distinct messages. For this reason, in coding theory the space of messages Σ^m is endowed with the *Hamming distance*, which counts how many positions in two messages are

different. Formally, given $x, y \in \Sigma^m$, the Hamming distance of x and y is defined as

$$d_H(x, y) = |\{i : x_i \neq y_i\}| . \tag{20}$$

This gives Σ^m the structure of a metric space. Clearly, since Σ^m is finite, many of the topological properties which we discussed in Chapter 2 about the full shift space such as compactness are trivially satisfied.

We can now give the formal definition of linear code. In what follows, we assume that the message space is a vector space over a finite field.

Definition 12. *Let* $n, m, d \in \mathbb{N}$ *such that* $n \geqslant m$*, and let* $q = \rho^\alpha$ *be the power of a prime number* ρ*. A* (n, m, d) linear code C *is a* m*-dimensional subspace of the vector space* $\mathbb{F}_q^n$*, such that the Hamming distance between any two vectors* $c_1, c_2 \in C$ *(called* codewords*) is at least* d*. The parameters* n*,* m *and* d *are respectively called the* length*, the* dimension *and the* minimum distance *of* C*.*

Since a (n, m, d) linear code C is a subspace of dimension m of $\mathbb{F}_q^n$, it is possible to specify it using a $m \times n$ matrix G whose rows form a set of m linearly independent codewords of C. Such a matrix G is called a *generator matrix* for code C. The encoding process simply amounts to multiplying a *message vector* $\mu \in \mathbb{F}_q^m$ by matrix G, thus obtaining the codeword $c = \mu G$. Another matrix related to a linear code C is its *parity check matrix* H of dimension $(n - m) \times n$. The parity check matrix is such that $Hx^\top = \underline{0}$ if and only if $x \in C$. In general, the vector $s = Hx^\top$ is called the *syndrome* of $x \in \mathbb{F}_q^n$.

Remark that, since every two distinct codewords in C are at Hamming distance at least d from each other, each vector $x \in \mathbb{F}_q^n$ will always belong to the ball of radius $t = \lfloor d - 1 \rfloor / 2$ of a unique codeword $c \in C$. This means that the union of all balls of radius t centered on the codewords of C do not intersect among themselves and cover the entire vector space $\mathbb{F}_q^n$, as depicted in Figure 10. For this reason, the parameter t is also called the *covering radius* of the code. Consequently, under the assumption that at most t errors occur during transmission (i. e. at most t symbols of the codeword are modified

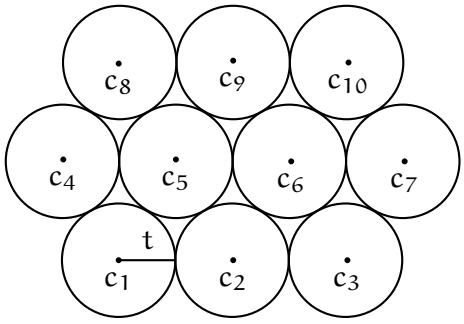


Figure 10: Covering of the message space by the codewords of C.

by noise), a straightforward error-correction procedure for Bob is to determine to which ball the received word z belongs to, and then output the center of the ball as the codeword transmitted by Alice. This method, which is called *Sphere Shrinking* (SS), has the drawback that the decoding procedure can take exponential time in the size of the alphabet, since Bob has to compute the Hamming distance between z and all the valid codewords in C in order to determine the minimum one.

A more efficient error-correcting algorithm is *Syndrome Decoding*, which works as follows. Let $c \in C$ be a codeword and $e \in \mathbb{F}_q^n$ be an *error pattern* introduced by the channel, having *Hamming weight* at most t, i. e. the number of nonzero coordinates in e is less than or equal to t. A received word can thus be expressed as $z = c + e$. Given a parity check matrix H of C, the syndrome of z is $s = Hz^\top = H(c+e)^\top = Hc^\top + He^\top = He^\top$. In order to retrieve c, it thus suffices to determine the error pattern e corresponding to s, and output $c = z + e$. This task can be performed by storing in a table the set of all error patterns of weight at most t together with their syndromes.

Let $x, y \in \mathbb{F}_q^n$. Then, the *scalar product* of x and y is defined as:

$$x \cdot y = \bigoplus_{i=1}^{n} x_i \cdot y_i = x_1 \cdot y_1 \oplus x_2 \cdot y_2 \oplus \cdots \oplus x_n \cdot y_n \ . \tag{21}$$

In particular, two vectors are called *orthogonal* if their scalar product is zero.

The *dual code* of a (n, m, d) linear code C is the set

$$C^\perp = \{x \in \mathbb{F}_2^n : \ x \cdot y = 0, \forall y \in C\} \ , \tag{22}$$

that is, the set of all vectors in $\mathbb{F}_q^n$ which are orthogonal to the codewords in C. The parity check matrix H of C is a generator matrix for $C^\perp$, and vice versa the generator matrix G of C is a parity check matrix for $C^\perp$.

The parameters of a code C induce some bounds on its size. In particular, *Singleton's bound* states that any (n, m, d) linear code C over $\mathbb{F}_q$ satisfies the following inequality:

$$|C| \leqslant q^{n-d+1} \ . \tag{23}$$

Codes that satisfy this bound with equality are called *Maximum Distance Separable* (MDS) codes. As discussed in Section 4, MDS codes play an important role in cryptography, especially in the design of linear diffusion layers for block ciphers. MDS codes can also be characterized in terms of orthogonal arrays, as the next result shows. A proof of this theorem can be found in [173].

Theorem 7. *A (n, m, d) linear code C over $\mathbb{F}_q^n$ is MDS if and only if its codewords are the rows of a $t-(q, n, 1)-$OA, where $t = n - d + 1$.*

We now introduce the class of linear *cyclic codes*.

Definition 13. *A* (n, m, d) *linear code* $C \subseteq \mathbb{F}_q^n$ *is called* cyclic *if it is closed under* cyclic shifts, *i. e. for all* $c = (c_1, c_2 \cdots, c_n) \in C$, *it holds that* $c' = \sigma(c) = (c_2, \cdots, c_n, c_1) \in C$.

A cyclic code is described by its *generator polynomial*, which is defined as $g(x) = g_0 + g_1 x + \cdots + g_{n-m} x^{n-m}$, where $g_i \in \mathbb{F}_q$ for all $i \in [n-m]$. If the m-bit message $\mu = (\mu_0, \cdots, \mu_{m-1})$ is represented by the polynomial $\mu(x) = \mu_0 + \mu_1 x + \cdots + \mu_{m-1} x^{m-1}$, then the polynomial corresponding to the codeword c is $c(x) = \mu(x)g(x)$. There exists a one-to-one correspondence between cyclic codes of length n and divisors of $x^n - 1$. In particular, a (n, m, d) code C is cyclic if and only if its generator polynomial $g(x)$ divides $x^n - 1$.

Given a (n, m, d) cyclic code C with generator polynomial $g(x)$ of degree $n - m$, the polynomial $h(x) = (x^n - 1)/g(x)$ of degree m is called the *parity check polynomial* of C. Analogously to the parity check matrix, $h(x)$ satisfies the property that the codeword associated to a polynomial $d(x)$ belongs to C if and only if $d(x)h(x) = 0$. The relationship between the generator/parity check polynomials of a cyclic code C and its generator/parity check matrices is given by the following result:

Theorem 8. *Let* $C \subseteq \mathbb{F}_q^n$ *be a* (n, m, d) *cyclic linear code with generator polynomial* $g(x) = g_0 + g_1 x + \cdots + g_{n-m} x^{n-m}$ *and parity check polynomial* $h(x) = h_0 + h_1 x + \cdots + h_m x^m$. *Then the following are respectively a generator and a parity check matrix for* C:

$$G = \begin{pmatrix} g_0 & \cdots & g_{n-m} & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & g_0 & \cdots & g_{n-m} & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & g_0 & \cdots & g_{n-m} \end{pmatrix}, \tag{24}$$

$$H = \begin{pmatrix} h_m & \cdots & h_0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & h_m & \cdots & h_0 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & h_m & \cdots & h_0 \end{pmatrix}. \tag{25}$$

As a consequence of Theorem 8, the dual code $C^\top$ of a cyclic code is again a cyclic code of length n and dimension $n - m$.

One of the main advantages of cyclic codes is that they can be easily implemented using linear feedback shift registers, as shown in [117, pp. 193–195]. In particular, if the parity check polynomial $h(x)$ of a (n, m, d) cyclic code is such that $h_0 \neq 0$, the codeword of a message $\mu \in \mathbb{F}_2^m$ can be generated by a LFSR of length m whose tap polynomial is the reciprocal $\tilde{h}(x) = h_m + h_{m-1} x + \cdots + x^m$ of $h(x)$. The registers are initialized to the values $\mu_0, \cdots, \mu_{m-1}$ of μ, and the LFSR

is evolved for n steps. The output of length n produced by the LFSR is the codeword corresponding to μ. Notice that the first m output bits are exactly the original message μ, while the remaining $n - m$ are the parity check bits. This encoding procedure is called *systematic*, since the bits of the message appear unaltered in the corresponding codeword. If no errors are introduced by the channel, the decoding process is immediate since it just consists of truncating the codeword to its first m bits.

CRYPTOGRAPHY

4

Cryptography is a discipline at the intersection of computer science and mathematics that studies the methods enabling two or more parties to securely communicate among each other in the presence of adversaries. As such, cryptography aims at developing protocols that enforces two fundamental properties of secure systems [136]: *confidentiality*, meaning that only authorized users can access protected data, and *integrity*, i. e. protected data cannot be tampered with or modified by non-authorized users.

The communication scheme studied in cryptography is illustrated in Figure 11. Similarly to the communication model of coding theory discussed in Section 3.3, the basic scenario of cryptography contemplates a *sender* and a *receiver*, respectively named Alice and Bob. Alice's goal is to send a *plaintext message* PT to Bob over an insecure communication channel which is wiretapped by an *opponent*, Oscar. Hence, Oscar can read and modify any message transiting on the channel. In order to thwart any interception attempt by Oscar, Alice feeds the plaintext PT to an *encryption function*, which depends also on an *encryption key* K_E. The result of the encryption function is the *ciphertext* CT, which is the message actually transmitted over the channel and eventually intercepted by Oscar. On the other side of the channel, Bob applies on CT a *decryption function*, that analogously depends on a *decryption key* K_D. The output of the decryption function is the original plaintext message PT which was originally meant to Bob by Alice. The various components of this transmission protocol are such that Oscar cannot recover the plaintext PT from the observed ciphertext CT if he does not know the decryption key K_D.

We formalize the above discussion in the following definition of a cryptosystem, adopting Stinson's notation [172]:

Definition 14. *A* cryptosystem *is a sextuple* $\mathsf{CS} = \langle \mathcal{P}, \mathcal{C}, \mathcal{K}_\mathsf{E}, \mathcal{K}_\mathsf{D}, \mathcal{E}, \mathcal{D} \rangle$ *where:*

- $\mathcal{P}$ *is a finite set of* plaintexts
- $\mathcal{C}$ *is a finite set of* ciphertexts
- $\mathcal{K}_\mathsf{E}$ *and* $\mathcal{K}_\mathsf{D}$ *are finite sets of* encryption *and* decryption keys, *respectively*
- $\mathcal{E} = \{\mathsf{e} : \mathcal{P} \to \mathcal{C}\}$ *and* $\mathcal{D} = \{\mathsf{d} : \mathcal{C} \to \mathcal{P}\}$ *are finite families respectively of* encryption *and* decryption functions

In particular, there exists a bijective mapping $\mathcal{M} : \mathcal{K}_\mathsf{E} \to \mathcal{K}_\mathsf{D}$ *between encryption and decryption keys. Moreover, for each* $\mathsf{K}_\mathsf{E} \in \mathcal{K}_\mathsf{E}$ *there exists*

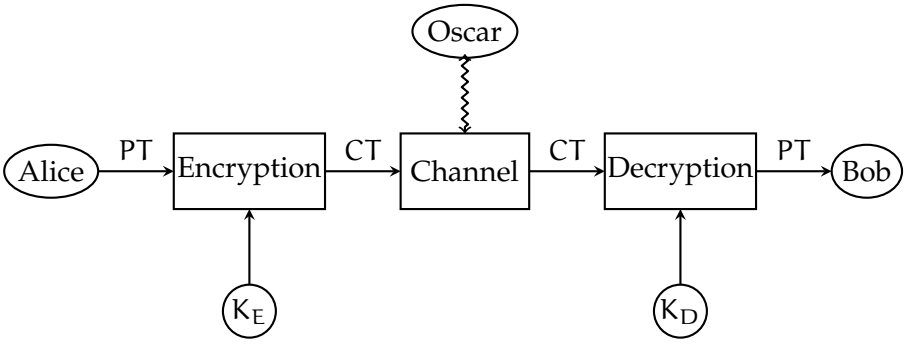


Figure 11: Basic communication scheme studied in cryptography.

an encryption function $e_{K_E} \in \mathcal{E}$ *and a decryption function* $d_{K_D} \in \mathcal{D}$ *where* $K_D = \mathcal{M}(K_E)$*, and such that for all* $PT \in \mathcal{P}$ *it holds that*

$$d_{K_D}(e_{K_E}(PT)) = PT \ . \tag{26}$$

The property expressed in Equation 26 is the most important one of a cryptosystem, since it states that encryption and decryption must be *idempotent* operations. Thus, given a plaintext message, applying to it in sequence an encryption function and the corresponding decryption function results in the original plaintext.

A broad classification of cryptosystems is based on the nature of the encryption and decryption keys. In particular, one may speak of a *symmetric* (or *secret*) key cryptosystem when $K_E = K_D$ and $\mathcal{M} = \mathrm{Id}$ is the identity function, i. e. Alice and Bob use the same key both for encryption and decryption. On the other side, if K_E and K_D are distinct or $\mathcal{M}$ is not the identity function, then one is dealing with an *asymmetric* (or *public key*) cryptosystem. In this thesis, we mainly focus on the symmetric key setting. Thus, in what follows we omit the subscript $_E$ and $_D$ when talking about a key K or the keyspace $\mathcal{K}$ to which it belongs.

In the remainder of this chapter, we focus on symmetric key cryptosystems in Section 4.1 by first modeling Oscar's attack capabilities and then by discussing three different security definitions a cryptosystem can satisfy. Subsequently, in Section 4.2 we further classify symmetric key cryptosystems by introducing *stream* and *block* ciphers. We conclude the chapter by presenting in Sections 4.3 and 4.4 two other cryptographic protocols, namely *authentication codes* and *secret sharing schemes*.

4.1 SYMMETRIC KEY CRYPTOGRAPHY

Historically, Kerckhoff [95, 96] was the first to observe that the security of a cryptosystem should not lie in the secrecy of the particular families $\mathcal{E}$ and $\mathcal{D}$ (an approach commonly referred to as *security through obscurity*), but rather on the secrecy of the key K used to select

an encryption and a decryption function respectively from $\mathcal{E}$ and $\mathcal{D}$. Since we are considering symmetric key cryptosystems, this means that Alice and Bob must agree on a common key before beginning transmission. Additionally, they must do so in a secure way, therefore not using the communication channel which is intercepted by Oscar. This posits the problem of *secure key agreement*, which obviously raises several practical questions. The most common remark is usually the following one: if Alice and Bob have access to a secure channel to share the key, then why do not use it directly to communicate the plaintext messages in the first place?

We will not delve into the details about how Alice and Bob can securely share a key. Rather, as most texts on symmetric cryptography do, we take secure key agreement as an axiom, and assume that Alice and Bob already figured out a way to agree on a common secret key[1].

When considering the security of a cryptosystem, the first thing the designer has to bear in mind are the capabilities of the attacker. In particular, depending on the type of attack Oscar can perform on a cryptosystem, one can isolate the following four *attack models*:

- *Ciphertext only attack*: Oscar has only the ciphertext CT transmitted over the channel
- *Known plaintext attack*: Oscar has a plaintext PT and the corresponding ciphertext CT
- *Chosen plaintext attack*: Oscar has access to the encryption function e_K, and can thus generate any plaintext message PT and obtain the corresponding ciphertext CT
- *Chosen ciphertext attack*: Oscar has access to the decryption function d_K, and can thus recover the original plaintext PT corresponding to any ciphertext CT he intercepts over the channel

Remark that the opponent's goal is not to decrypt ciphertexts in an impromptu manner. Rather, what Oscar aims for is the value of the secret key K, which would give him complete control over any message transmitted over the channel. This is why it makes sense to consider also the chosen ciphertext attack model.

The next step consists in addressing the security of a cryptosystem with respect to the computational resources that Oscar has at its disposal to perform one of the above attacks. In this sense, as summarized in [172], there are three common definitions adopted in the literature:

- *Computational security*: a cryptosystem is said to be computationally secure if the best attack that an opponent can apply on it

1 Usually, this problem is solved by means of public key cryptography (see for example the *Diffie-Hellmann key exchange protocol* [56]).

requires at least N operations, with N being a very large number. The problem with this approach is that no cryptosystem can be proved to be secure under this definition. In particular, computational security is studied only with respect to specific cryptanalytic attacks. Of course, this is not an absolute guarantee, since if a cryptosystem is secure against a particular attack, it can well be the case that it is vulnerable to other attacks.

- *Provable security*: this approach consists of reducing the security of a cryptosystem to solving a problem which is believed to be hard from the computational complexity point of view. In this case, Oscar's computational resources are bounded to the class of *probabilistic polynomial time* (PPT) algorithms. Thus, given a provable secure cryptosystem CS, finding a PPT algorithm that solves the problem CS has been reduced to would be equivalent to breaking CS. Most of *complexity-based* cryptography rests upon this security definition, and the hardness assumptions of certain computational problems (such as, for example, factoring integer numbers). However, we will not use provable security in this thesis. The interested reader may refer to [89, 72, 73] for further information on the subject.

- *Unconditional security*: in this security definition, the computational resources of the opponents are not bounded. Thus, a cryptosystem is unconditionally secure if Oscar cannot successfully attack it even with infinite computational resources.

Since we discuss more in detail in Chapter 5 the notion of computational security with respect to the framework of Boolean functions, let us now have a few additional words on unconditional security. In particular, we consider unconditional security under the weakest attack model for Oscar, namely ciphertext only attacks.

Let $\langle \mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ be a symmetric key cryptosystem. Suppose we have defined two *discrete random variables* $\chi : \mathcal{P} \to \mathbb{R}$ and $\kappa : \mathcal{K} \to \mathbb{R}$ respectively over the plaintext set $\mathcal{P}$ and the keyspace $\mathcal{K}$. Additionally, given $K \in \mathcal{K}$, let us define $C(K) = \{e_K(x) : x \in \mathcal{P}\}$, i.e. the set of all ciphertexts generated by computing e_K over all possible plaintexts. Then, the random variable $\gamma : \mathcal{C} \to \mathbb{R}$ over the set of ciphertexts has an associated probability distribution that equals

$$Pr[\gamma = y] = \sum_{K \in \mathcal{K} : y \in C(K)} Pr[\kappa = K] \cdot Pr[x = d_K(y)] \ . \tag{27}$$

On the other hand, the conditional probability distribution that a ciphertext $y \in \mathcal{C}$ will be generated by a plaintext $x \in \mathcal{P}$ is given by

$$Pr[\gamma = y | \chi = x] = \sum_{K \in \mathcal{K} : x = d_K(y)} Pr[\kappa = K] \ . \tag{28}$$

As a consequence, one can combine Equations (27) and (28) to determine through *Bayes' theorem* the probability that a particular observed ciphertext $y \in \mathcal{C}$ has been generated by a plaintext $x \in \mathcal{P}$ as follows:

$$\Pr[\chi = x | \gamma = y] = \frac{\Pr[\chi = x] \cdot \Pr[\gamma = y | \chi = x]}{\Pr[\gamma = y]} \ . \tag{29}$$

We can now introduce the property of *perfect secrecy*, which is the standard definition of unconditional security for symmetric key cryptosystems under ciphertext only attacks:

Definition 15. *Let* $\mathrm{CS} = \langle \mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ *be a symmetric key cryptosystem with two discrete random variables* $\chi : \mathcal{P} \to \mathbb{R}$ *and* $\kappa : \mathcal{K} \to \mathbb{R}$ *respectively defined over* $\mathcal{P}$ *and* $\mathcal{K}$*. Then,* CS *has* perfect secrecy *if and only if for all* $x \in \mathcal{P}$ *and* $y \in \mathcal{C}$ *it holds that*

$$\Pr[x|y] = \Pr[x] \ . \tag{30}$$

In other words, Definition 15 states that a cryptosystem is perfectly secure if Oscar does not gain any information on the plaintext x just by observing the ciphertext y.

Recalling our previous discussion of Latin squares in Section 3.2, let us turn our attention to a concrete example of symmetric cryptosystem satisfying perfect secrecy, the *one-time pad*:

Definition 16. *Let* Σ *be a finite alphabet of* q *symbols and* $n \in \mathbb{N}$*. Additionally, let* L *be a Latin square of order* q *with entries from* Σ*. The* one-time pad *(OTP) of length* n *over* Σ *is the symmetric cryptosystem* $\mathrm{OTP}_n = \langle \mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ *where:*

- $\mathcal{P} = \mathcal{C} = \mathcal{K} = \Sigma^n$
- *For each* $x = (x_1, \cdots, x_n), K = (K_1, \cdots, K_n) \in \Sigma^n$*, the encryption function* $e_K(x)$ *is defined as*

$$e_K(x) = (e_{K_1}(x_1), \cdots, e_{K_n}(x_n)) = (L(x_1, K_1), \cdots, L(x_n, K_n)) \ . \tag{31}$$

- *Symmetrically, given a ciphertext* $y = (y_1, \cdots, y_n) \in \Sigma^n$ *and a key* $K \in \Sigma^n$*, the decryption function* $d_K(y)$ *is defined as*

$$d_K(y) = (d_{K_1}(y_1), \cdots, d_{K_n}(y_n)) = (L(K_1, y_1), \cdots, L(K_n, y_n)) \ . \tag{32}$$

Hence, for each component $i \in [n]$ of the plaintext message x and the key K, the one-time pad encryption function computes the i-th component of the ciphertext y_i as the entry of L whose row and column are respectively indexed by x and K. Analogously, the i-th component of the plaintext is recovered by the decryption function returning the entry of L at the coordinates indexed by K and y.

Given that each row and each column of the Latin square L is a permutation of Σ, it is easy to see that the OTP encryption and decryption functions are idempotent operators. Moreover, assuming that a uniform probability distribution is defined over the keyspace $\mathcal{K}$ (i. e. , each symbol of a key is independently sampled with uniform probability), one can also see that the OTP satisfies the perfect secrecy property. As a matter of fact, suppose that Oscar observed a particular ciphertext symbol y_i, and that he has access to the Latin square L used by Alice for encryption. Given that Oscar does not know the symbol of the key K_i, he can determine the column of the Latin square which Bob will use to recover x_i, but not the corresponding row. Since the column indexed by y_i is a permutation of Σ and that Alice randomly chose K_i with uniform probability, this means that the conditional probability $\Pr[x_i|y_i]$ equals the apriori probability $\Pr[x_i]$. Consequently, the OTP meets the definition of perfect secrecy.

Shannon [164] was the first to observe this property of OTP. In fact, he went way beyond than that: up to isomorphism (i. e. up to the particular Latin square used for encryption and decryption), Shannon proved that the one-time pad is the only kind of symmetric cryptosystem achieving perfect secrecy. This can also be translated in the following three conditions, which must be simultaneously satisfied:

(a) The key must have the same length as the plaintext message.

(b) There must be a one-to-one correspondence between plaintext messages and keys.

(c) The key must be *random*, i. e. sampled with uniform probability from $\mathcal{K}$.

In particular, condition (b) means that a key cannot be used to encrypt more than one message, otherwise the scheme would be vulnerable to a known plaintext attack. This is the reason of the "one-time" adjective in OTP.

4.2 BLOCK AND STREAM CIPHERS

As one may think, the three conditions that characterize perfect secrecy greatly hampers the use of OTP in several practical scenarios. In fact, Alice and Bob should share over a secure channel a key which is exactly as long as the plaintext. This immediately leads to the question about why Alice does not directly use this channel to transmit the plaintext. Nevertheless, the OTP is still finds some applications in the design of certain cryptosystems (e. g. in *quantum cryptography* [5]). More generally, the notion of unconditional security is also used in cryptographic protocols other than encryption schemes, such as *authentication codes* and *secret sharing schemes* on which we will elaborate more in the next two sections.

Let us now return to the security of cryptosystems. From now on, we will assume that the key K is shorter than the plaintext message, and thus we have to fall back to the computational security setting.

After observing the shortcomings of the OTP scheme in [164], Shannon described in the same paper two general principles that every cryptosystem should satisfy in order to frustrate statistical attacks, namely *diffusion* and *confusion*. The diffusion principle states that each symbol in the plaintext should depend on several symbols of the ciphertext. In other terms, the statistical structure of the plaintext should be sufficiently "spread" over the ciphertext. On the other hand, the confusion principle says that the symbols of the plaintext should depend in a complicated way on the symbols of the key.

Although diffusion and confusion are not rigorously defined concepts, they are often the two main guidelines that cryptographers use to design symmetric cryptosystems achieving computational security.

One may classify symmetric key cryptosystems as *block ciphers* and *stream ciphers*. In a block cipher, the plaintext is usually encrypted by dividing it in chunks of symbols called *blocks* that are subsequently combined with several *round keys* derived from the initial secret key through a *scheduling algorithm*. In a stream cipher, on the contrary, each symbol of the plaintext is combined with the corresponding symbol of a *keystream*, computed from the initial secret key through a keystream generator algorithm [2].

Let us have a closer look at block ciphers, starting by formally defining them. In what follows, we assume that the underlying alphabet for the plaintext, the ciphertext and the key is $\Sigma = \{0, 1\}$.

Definition 17. *Let* $r, n \in \mathbb{N}$. *A* block cipher *with* block length n *and* r rounds *is a cryptosystem* $BC = \langle \mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ *where:*

- $\mathcal{P} = \mathcal{C} = \mathcal{K} = \{0, 1\}^n$, *with* $n \in \mathbb{N}$.
- *The family of encryption functions* $\mathcal{E}$ *is defined by a* key scheduling function $KS : \mathcal{K} \to \mathcal{K}$ *and by a* round function $g : \mathcal{P} \times \mathcal{K}$ *for all key* $K \in \mathcal{K}$ *and plaintext* $x \in \mathcal{P}$ *as follows:*

$$e_K(x) = g^r(x, K) \ , \tag{33}$$

where for all $i \in \{1, \cdots, r\}$ *the function* g^i *is inductively defined as:*

$$g^i(x, K) = \begin{cases} g(x, KS(K)) & , \text{ if } i{=}1 \\ g(g^{i-1}(x, K), KS(KS^{i-1}(K))) & , \text{ if } i{>}1 \end{cases} \tag{34}$$

with $KS^0 \equiv Id$ *denoting the identity function.*

2 Remark that the difference between block and stream ciphers is often a fuzzy one: depending on the definition of a "symbol", a stream cipher could also be interpreted as a block cipher. This is the case, for instance, of the RC4 stream cipher [157], where each symbol is actually an 8-bit block. On the other hand, block ciphers can be used under certain modes of operation as stream ciphers to encrypt one bit at a time.

- *Denoted as* $h = g^{-1}$ *the inverse round function, the family of decryption functions* $\mathcal{D}$ *is defined symmetrically for all key* $K \in \mathcal{K}$ *and ciphertext* $y \in \mathcal{C}$ *as follows:*

$$d_K(y) = h^r(y, K) \ , \tag{35}$$

with h^i *being defined for all* $i \in \{1, \cdots, r\}$ *as:*

$$h^i(y, K) = \begin{cases} h(y, KS^r(K)) & , \text{ if } i{=}1 \\ h(h^{i-1}(y, K), KS^{r-i+1}(K))) & , \text{ if } i{>}1 \end{cases} \tag{36}$$

As a consequence, a block cipher is completely specified by its round function g (together with its inverse h) and the key scheduling algorithm KS. In particular, the only secret information that must be shared by Alice and Bob is the initial key K, since they can independently derive all the round keys they need by applying KS.

Concerning the round function, one of the most common design approach is the *Substitution-Permutation Network* (SPN). Suppose we want to encrypt a block of plaintext bits $x \in \{0,1\}^n$ of length $n = lm$. A SPN cipher is usually composed of the following parts:

- A *Permutation box* (P-box) $\pi : \{0,1\}^n \to \{0,1\}^n$ which permutes blocks of n-bits.
- A set of m *Substitution boxes* (S-boxes) $\sigma_i : \{0,1\}^l \to \{0,1\}^l$, each of which takes as input an l-bit block and output another l-bit block. Each S-box is invertible (i. e. , a permutation).

Let $x, K \in \{0,1\}^n$, with $u = x \oplus K$ denoting their bitwise XOR and $u_{[i,j]}$ the block included between $i < j$ of u. Then, the round function g is defined as

$$g(x, K) = \pi(\sigma_1(u_{[1,l]}), \sigma_2(u_{[l+1,2l]}), \cdots, \sigma_m(u_{[m(l-1)+1,lm]}) \ . \tag{37}$$

Following Definition 17, a plaintext $x \in \{0,1\}^n$ is transformed in the ciphertext $y \in \{0,1\}^n$ by applying for each round $i \in \{1, \cdots, r\}$ the following procedure:

1. *Initialization:* Let x_{i-1} be the result of the previous encryption round, with x_0 being the original plaintext block
2. *Key combination phase*: combine x_{i-1} with the current round key K_i by computing a bitwise XOR.
3. *Substitution phase*: Let u be the result of the key combination phase. Then, split u in m blocks $b_1, \cdots, b_m \in \{0,1\}^l$ of l bits each, and for each $i \in \{1, \cdots, m\}$ apply the S-box σ_i to b_i, obtaining $c_i = \sigma_i(b_i)$

4. *Permutation phase*: Let $C = c_1 \| c_2 \| \cdots \| c_m$ be the concatenation of the blocks computed by the S-boxes at the previous step. Then, apply the permutation π to C, obtaining the block $x_i = \pi(C)$. In the final round r, the resulting block x_r is the ciphertext y.

The decryption process of a SPN cipher consists simply in the above procedure applied in reverse order. This is the reason why all operators involved in encryption (S-boxes and P-box) must be invertible.

The permutation π is also called the *diffusion layer* of the cipher. As a matter of fact, one can see that the P-box acts on the whole block of length n, performing a transposition of the bits inside it. Hence, the permutation phase realizes the diffusion principle. Usually, this phase is implemented in real-world block ciphers through a linear invertible transformation. In many cases, this transformation is derived from the generator matrix of a linear MDS code, since it provides optimal diffusion. For instance, in the *Rijndael* cipher, which won the NIST competition for the *Advanced Encryption Standard* (AES), the MixColumns operation used in the permutation phase is defined by a circulant MDS matrix [156].

On the other side, the S-boxes used in the substitution phase are the elements devoted to confusion. As we will discuss in detail in Chapter 5, S-boxes can be regarded as vectorial Boolean functions $F : \mathbb{F}_2^l \to \mathbb{F}_2^l$, and they must satisfy certain properties (namely, high nonlinearity) in order to provide good confusion.

A different approach for the construction of block ciphers is the *Feistel Network* (FN), from the name of its inventor who first proposed it in the design of the IBM cipher *Lucifer*, upon which the *Data Encryption Standard* (DES) encryption algorithm is based [171].

In a Feistel network, the plaintext $x \in \{0,1\}^n$ of length $n = 2m$ is represented by a left block $L \in \{0,1\}^m$ and a right block $R \in \{0,1\}^m$ of equal length m, i.e. $x = (L, R)$. At each round $i \in \{1, \cdots, r\}$, let $L_{i-1}, R_{i-1} \in \{0,1\}^m$ be respectively the left and right block of the previous step (with $(L_0, R_0) = (L, R) = x$), and let $K_i \in \{0,1\}^k$ be the round key for step i. Then, the round function g is defined as

$$g(L_{i-1}, R_{i-1}, K_i) = (L_i, R_i) \ . \tag{38}$$

where the blocks L_i and R_i are computed as follows:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i) \ , \end{aligned} \tag{39}$$

where $f : \{0,1\}^m \times \{0,1\}^k \to \{0,1\}^m$ is a S-box mapping the previous right block R_{i-1} and the round key K_i to an m-bit block, which is subsequently XORed with the previous left block L_{i-1}. As in the case of SPN ciphers, decryption is achieved by performing the same round operations in reverse order. Differently from SPN ciphers, however, the S-box used in a FN does not need to be bijective, since invertibility is ensured by the XOR operation of Equation (38).

A block cipher can be used to encrypt a plaintext message of arbitrary length $x \in \{0,1\}^*$ by dividing it in blocks of length n and then by applying on each of them the round function and the key scheduling algorithm. Depending on the *operation mode*, each block can be encrypted independently (as in *Electronic Code Book* mode, ECB) or by combining them in a specified way (as in *Cipher Block Chaining* mode, CBC). The interested reader can refer to [172] for further information about modes of operation of block ciphers.

We now turn to stream ciphers. In particular, we focus on *Vernam-like synchronous stream ciphers*, defined as follows:

Definition 18. *A* Vernam-like synchronous stream cipher *is a cryptosystem* $SC = \langle \mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ *where:*

- $\mathcal{P} = \mathcal{C} = \mathcal{K} = \{0,1\}^*$
- *The family of encryption functions* $\mathcal{E}$ *is defined by a* keystream generator function $g : \mathcal{K} \to \{0,1\}^*$ *which, given an initial secret key* $K \in \mathcal{K}$ *of length* $l \in \mathbb{N}$ *and the length of the plaintext* $n \geqslant l$*, produces a binary string* z *of length* n *called the* keystream*. The encryption function* $e_K \in \mathcal{D}$ *is thus defined as*

$$e_K(x) = x \oplus z = x \oplus g(K) \ , \tag{40}$$

  i. e. , the bitwise XOR between the plaintext x *and the keystream* z

- $\mathcal{D} = \mathcal{E}$*. In particular, each encryption function* e_K *is an involution: for all ciphertexts* $y \in \{0,1\}^*$*, it holds*

$$d_K(y) = e_K(y) = y \oplus z = y \oplus g(K) \ . \tag{41}$$

It is easy to see that $d_K(e_K(x)) = x$ for all $x \in \{0,1\}^*$, since the XOR is a idempotent operation. Figure 14 depicts the encryption and decryption block schemes for a Vernam-like stream cipher.

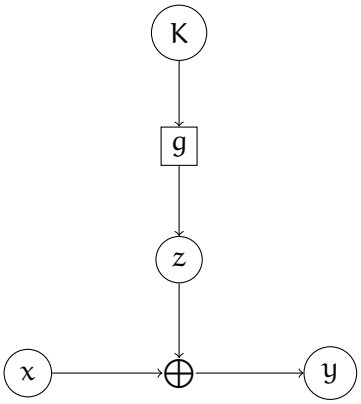


Figure 12: Encryption

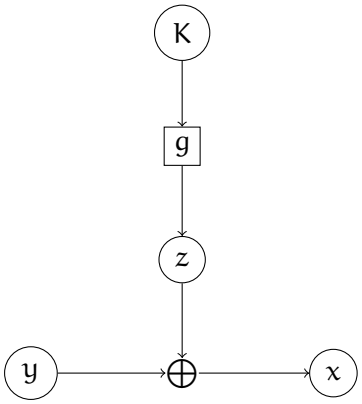


Figure 13: Decryption

Figure 14: Encryption and decryption in a Vernam-like stream cipher.

One may notice a close resemblance between a Vernam-like stream cipher and the OTP. As a matter of fact, the Cayley table of the XOR operation corresponds to one of the two possible Latin squares of order 2 (the other being the complement of the XOR). Hence, by generating a random keystream which is as long as the plaintext, one obtains the OTP over the binary alphabet[3]. In practice, however, one usually adopts a deterministic *Pseudorandom Number Generator* (PRNG) as the keystream generator g, to stretch a short secret key $K \in \mathcal{K}$ shared by Alice and Bob (which as been chosen randomly) to an arbitrarily long pseudorandom keystream. As a consequence, a Vernam-like stream cipher does not satisfy perfect secrecy, and its computational security entirely rests upon the properties of the underlying PRNG. Remark that the generated keystream will repeat after at most 2^n steps if the state of the PRNG is described by a register of n bits.

A common method for designing a PRNG for a stream cipher is the *combiner model* [30], represented in Figure 15.

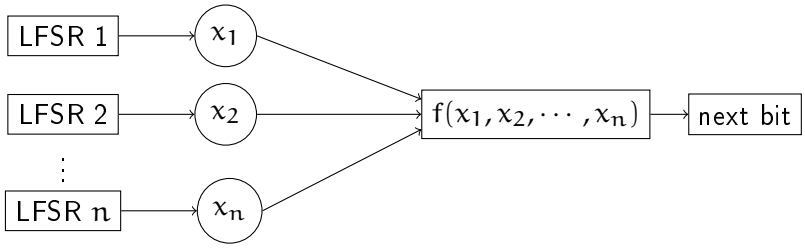


Figure 15: Combiner model for the PRNG of a Vernam-like stream cipher.

In this model, the outputs of n LFSRs are combined by a Boolean function $f : \{0,1\}^n \to \{0,1\}$. A PRNG based on the combiner model can generate a sequence whose maximal length is upper bounded by the LCM of the n LFSRs composing it. In order to resist to specific attacks, the Boolean function used in the combiner model must satisfy several properties, which are discussed in Chapter 5.

4.3 AUTHENTICATION CODES

Recall that the goal of cryptography is to develop secure communication systems meeting the requirements of *confidentiality* and *integrity*. However, up to now we only analyzed cryptosystems, which provide confidentiality but do not not address integrity. In fact, Oscar is able not only to eavesdrop the messages passing through the channel, but also to modify them. Hence, he could tamper with the ciphertext message transmitted by Alice to Bob, or even pose as Bob (in the so-

3 Historically, this encryption scheme was known as the *Vernam cipher*, since it was patented by Gilbert Vernam in 1919 [187]. This is also the reason why we called the cryptosystem of Definition 18 as Vernam-like. The perfect secrecy of the Vernam cipher, however, was proved in 1949 by Shannon [164].

called *man in the middle* attack). The tools developed in cryptography to solve this problem are *authentication codes* (AC).

Generally speaking, an AC is composed of a *signing function* and a *verification function*. Given a message that Alice wishes to send to Bob, she computes the signing function on it, thus obtaining the *signature* of the message. Then, Alice transmits to Bob both the message and its signature. Finally, Bob applies to the received message the verification function, and checks whether the resulting signature matches with the received one. If not, Bob knows that someone has modified the message during transmission. A typical method for constructing authentication codes relies on the use of public-key cryptosystems, where signing and verification respectively depend on private and public keys [172].

In what follows we focus on *symmetric* authentication codes, where the signing and verification functions coincide. Hence, from now on by AC we always refer to the symmetric case, unless otherwise specified. We begin with the following definition taken from [173]:

Definition 19. *An* Authentication Code *is a quadruple*

$$\mathrm{AC} = \langle \mathcal{M}, \mathcal{A}, \mathcal{K}, \mathcal{E} \rangle \ ,$$

where:

- $\mathcal{M}$ *is a finite set of* messages
- $\mathcal{A}$ *is a finite set of* authenticators
- $\mathcal{K}$ *is a finite set of* keys
- $\mathcal{E}$ *is a set of* authentication rules*, where for each key* $K \in \mathcal{K}$ *there exists a rule* $e_K \in \mathcal{E}$ *with* $e_K : \mathcal{M} \to \mathcal{A}$

Similarly to the case of symmetric cryptosystems, in an AC Alice and Bob agree ahead of time on a secret key $K \in \mathcal{K}$. When Alice wants to transmit a message $m \in \mathcal{M}$, she computes $a = e_K(m)$, and sends to Bob the pair $c = (m, a)$. On the other end of the communication channel, Bob computes again the authentication rule on the received message, and verifies whether $a = e_K(m)$. If $a \neq e_K(m)$, Bob can conclude that the message has been modified in transit.

Oscar can perform two types of attacks on an AC:

- *Impersonation attack*: Oscar injects a new pair $c' = (m', a')$ into the channel.
- *Substitution attack*: Oscar observes a message pair $c = (m, a)$ sent by Alice, and replaces it with a new pair $c' = (m', a')$.

In both attacks, the only information which is unknown to Oscar is the secret key K chosen by Alice and Bob, while all remaining components of the AC (in particular, the family of authentication rules

$\mathcal{E}$) are supposed to be public. The goal of Oscar is to deceive Bob by making him accept his fake message c' as legitimately sent by Alice. Let us denote by $\Pr[\mathrm{Imp}]$ and $\Pr[\mathrm{Sub}]$ the *deception probabilities* that Oscar successfully performs an impersonation attack and a substitution attack, respectively. Ideally, these probabilities should be as low as possible.

An orthogonal array with $t = 2$ and k columns (or equivalently, a set of $(k-2)$-MOLS as illustrated in Section 3.2) can be used to construct an authentication code with optimal deception probabilities. In particular, let A be an $\mathrm{OA}(k, v)$, and define $\mathcal{M} = [k]$, $\mathcal{A} = [v]$ and $\mathcal{K} = [v^2]$. Given $K \in \mathcal{K}$, the authentication rule e_K is defined for all possible messages $m \in \mathcal{M}$ as

$$e_K(m) = B(K, m) \ , \tag{42}$$

that is, the authenticator corresponding to m is the entry in B whose row and column are respectively indexed by K and m.

Let us first analyze the deception probability $\Pr[\mathrm{Imp}]$ of this AC. Supposing that Oscar has injected a message pair $c' = (m', a')$, he will succeed in his impersonation attack only if the authenticator a' equals $e_K(m')$. Let us define $\mathcal{R}(m', a') = \{R \in [v^2] : B(R, m') = a'\}$ as the set of all rows in B which have entry a' in the column indexed by m'. Since B is a $\mathrm{OA}(k, v)$, it follows that $|\mathcal{R}(a')| = v$. Now, Oscar will succeed in the attack only if the key K' which he uses to determine the authentication rule belongs to the set $\mathcal{R}(m', a')$. Considering that $|\mathcal{K}| = v^2$, this means that the deception probability $\Pr[\mathrm{Imp}]$ equals

$$\Pr[\mathrm{Imp}] = \frac{|\mathcal{R}(m', a')|}{|\mathcal{K}|} = \frac{v}{v^2} = \frac{1}{v} \ . \tag{43}$$

Concerning the deception probability $\Pr[\mathrm{Sub}]$ of substitution, suppose now that Oscar observed the message $c = (m, a)$ and replaced it with $c' = (m', a')$. Since B is assumed to be public, Oscar knows that the key K used by Alice and Bob is a row of B belonging to the set $\mathcal{R}(m, a)$. As a consequence, Oscar's substitution will succeed only if $K \in \mathcal{R}(m, a) \cap \mathcal{R}(m', a')$. Remark that $|\mathcal{R}(m, a) \cap \mathcal{R}(m', a')| = 1$, because B is a $\mathrm{OA}(k, v)$. Thus, the deception probability is equal to

$$\Pr[\mathrm{Sub}] = \frac{|\mathcal{R}(m, a) \cap \mathcal{R}(m', a')|}{|\mathcal{R}(m, a)|} = \frac{1}{v} \ . \tag{44}$$

As a consequence, both deception probabilities $\Pr[\mathrm{Imp}]$ and $\Pr[\mathrm{Sub}]$ equals $\frac{1}{v}$, with v being the number of possible authenticators. Authentication codes reaching this lower bound on the deception probabilities are also called *perfect* (or *Cartesian* [169]), in analogy with the perfect secrecy property of the OTP discussed in Section 4.1.

4.4 SECRET SHARING SCHEMES

Secret sharing schemes (SSS) are a cryptographic primitive underlying several protocols such as *secure multiparty computation* [38] and *gener-*

alized oblivious transfer [179]. The basic scenario addressed by a SSS considers a *dealer* who wants to share a *secret* K (for instance, a cryptographic key) among a set of *players*, so that only certain authorized subsets of players may reconstruct K.

We give the following formal definition of a generic SSS:

Definition 20. *A* Secret Sharing Scheme *(SSS) is a sextuple*

$$S = \langle \mathcal{K}, \mathcal{B}, \mathcal{P}, \Gamma, \mathcal{D}, \mathcal{C} \rangle$$

where:

- $\mathcal{K}$ *is a finite set of secrets.*
- $\mathcal{B}$ *is a finite set of shares.*
- $\mathcal{P}$ *is a finite set of players.*
- $\Gamma \subseteq 2^{\mathcal{P}}$ *is a family of subsets of players called the* access structure *of* S*. The elements of* Γ *are the* authorized subsets *of* S*.*
- $\mathcal{D}$ *is a family of* distribution rules*: for each secret* $K \in \mathcal{K}$ *there exists* $d_K : \mathcal{P} \to \mathcal{B}$ *with* $d_K \in \mathcal{D}$*.*
- $\mathcal{C}$ *is a family of* combination rules*: for each secret* $K \in \mathcal{K}$*, denote by* $d_K(\mathcal{P})$ *the* image *of the distribution rule* $d_K \in \mathcal{D}$*. Then, there exists a combination rule* $c_K : 2^{d_K(\mathcal{P})} \to \mathcal{K}$ *with* $c_K \in \mathcal{C}$ *such that for all authorized subset* $A \in \Gamma$ *it holds that* $c_K(A) = K$*.*

Suppose that $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ is the set of players. The protocol of a SSS S can be summarized in two steps, namely the *setup phase* and the *reconstruction phase*. During the setup phase, the dealer chooses a secret $K \in \mathcal{K}$ and sends the share $B_i = d_K(P_i)$ to player P_i for all $i \in \{1, \cdots, n\}$. In the reconstruction phase, the players of an authorized subset $A = \{P_{i_1}, P_{i_2}, \cdots, P_{i_k}\} \in \Gamma$ pool together their shares $B_{i_1}, B_{i_2}, \cdots, B_{i_k}$ received from the dealer and compute the combination rule $c_K(A)$. The result is the value of the secret $K \in \mathcal{K}$.

Usually, the access structure Γ is required to be *monotone*, that is, if $A_1 \in \Gamma$ and $A_1 \subseteq A_2$ then $A_2 \in \Gamma$. This is a reasonable property, since if an authorized subset includes more shares than what are actually needed to compute the combination rule, then it is intuitive to think that the players can still recover the secret by discarding the extra shares. In particular, an authorized subset $M \in \Gamma$ is called *minimal* if $N \notin \Gamma$ for all $N \subset M$, that is, if all shares in M must be used to recover the secret. A monotone access structure Γ can thus be defined as the union-closure of the *basis* Γ_0, which is the family of all minimal authorized subsets.

One of the most studied models of SSS are *threshold schemes*, originally introduced by Shamir [163] and Blakley [17], which we define as follows:

Definition 21. *Let* $t, n \in \mathbb{N}$ *with* $t \leqslant n$. *A* (t, n)–threshold scheme *is a SSS* $S = \langle \mathcal{K}, \mathcal{B}, \mathcal{P}, \Gamma, \mathcal{D}, \mathcal{C} \rangle$ *where* $\mathcal{P}$ *is a set of* n *players and the access structure is defined as* $\Gamma = \{A \subseteq 2^{\mathcal{P}} : |A| \geqslant t\}$.

In other words, in a (t, n)–threshold scheme all subsets having at least t players can reconstruct the secret. It is easy to see that the access structure in this scheme is monotone, since Γ is defined by the following basis:

$$\Gamma_0 = \{A \subseteq 2^{\mathcal{P}} : |A| = t\} \ , \tag{45}$$

i. e. the minimal authorized subsets of S are all those subsets of exactly t players. Figure 16 represents an example of $(2, 3)$–threshold scheme, where each combination of 2 players can recover the secret.

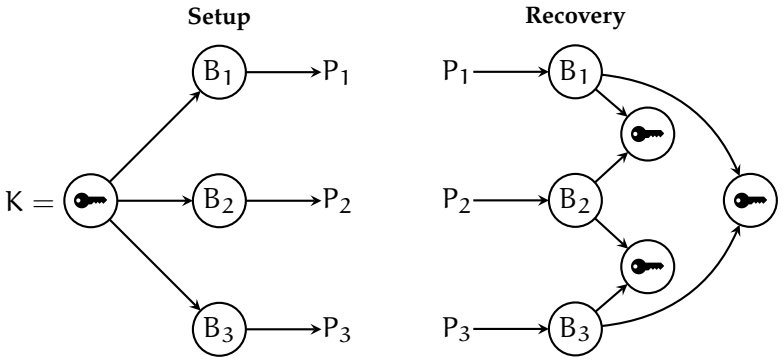


Figure 16: Schematic description of a $(2, 3)$ threshold SSS.

The security model adopted for the study of secret sharing schemes considers the information an attacker can obtain about the secret K by having the shares of a generic unauthorized subset. In particular, schemes which do not leak any information on K by knowing the shares of any unauthorized subset $U \notin \Gamma$ are called *perfect*, analogously to the perfect secrecy property of the OTP cryptosystem which we discussed in Section 4.1. To formalize this notion in a probabilistic framework, we follow the approach laid out by Stinson [172].

Let $\mathcal{K}$ be the space of secrets and $\mathcal{B}$ the space of possible shares. Given a secret $K \in \mathcal{K}$, the set $\mathcal{D}_S$ denotes the family of all distribution rules induced by K. The dealer selects both the secret and the corresponding distribution rule according to two probability distributions, which we respectively denote by $\Pr[K]$ and $\Pr[d]$. These probability distributions and the family of distribution rules $\mathcal{D} = \bigcup_{K \in \mathcal{K}} \mathcal{D}_K$ are assumed to be public, hence known to an attacker. Considering a generic subset of players $G \subseteq \mathcal{P}$, a *shares distribution* δ_G is a possible assignment of shares to the members of G. Given a distribution rule d, the corresponding shares distribution δ_G is thus the image of the restriction $d|_G$. By $\mathcal{B}_G(G) = \{d|_G : \varphi \in \mathcal{D}_K\}$ we denote the set of all possible shares distributions to G induced by the secret K. The probability distribution on all possible values of δ_G is obtained as follows:

$$\Pr[\delta_G] = \sum_{K \in \mathcal{K}} \left(\Pr[K] \cdot \sum_{d \in \mathcal{B}_G(S)} \Pr[\varphi] \right) \quad . \tag{46}$$

We can now give the formal definition of *perfect* secret sharing scheme.

Definition 22. *A secret sharing scheme* $S = \langle \mathcal{K}, \mathcal{B}, \mathcal{P}, \Gamma, \mathcal{D}, \mathcal{C} \rangle$ *with access structure* $\Gamma \subseteq 2^{\mathcal{P}}$ *and family of distribution rules* $\mathcal{D} = \bigcup_{K \in \mathcal{K}} \mathcal{D}_K$ *is* perfect *if for all unauthorized subsets* $U \notin \Gamma$ *and for all shares distributions* δ_U *it results that* $\Pr[K|\delta_U] = \Pr[S]$.

Looking at the definition above, one may ask whether there is a relation between the size of the secret and the size of the shares in a perfect SSS. Let us assume that $\mathcal{K} = \Sigma^n$ and $\mathcal{B} = \Sigma^m$ are both defined over an alphabet Σ. Then, the inequality $m \geqslant n$ is a necessary condition for a SSS to be perfect, i. e. the size of the shares must be at least equal to the size of the secret. This requirement is similar to the property of perfect secrecy where the secret key must be at least as long as the plaintext message. Perfect secret sharing scheme where the size of the shares equals the size of the secret are also called *ideal*.

We conclude this section by describing how a t–$(\nu, n+1, 1)$ orthogonal array A can be used to implement a (t, n)–*threshold scheme* with n players $P_1, \cdots, P_n$.

In the setup phase, the dealer chooses with uniform probability the secret K from the support set X and a row $A(i, \cdot)$ in the OA such that the last component equals K. Next, for all $j \in \{1, \cdots, n\}$ the dealer distributes to player P_j the share $B_j = A(i, j)$. Thus, we have that

$$\Pr[K] = \Pr[d] = \frac{1}{|X|} = \frac{1}{\nu} \quad . \tag{47}$$

In particular, notice that the probability of selecting the distribution rule d equals $\frac{1}{\nu}$ since there are exactly ν rows having K as their last component, due to the fact that the orthogonal array has $\lambda = 1$.

In the reconstruction phase, any subset of t players $P_{j_1}, \cdots, P_{j_t}$ can recover the secret, the reason being that the shares $(B_{j_1}, \cdots, B_{j_t})$ form a t–uple which uniquely identifies row $A(i, \cdot)$. Conversely, suppose that a set of $t-1$ players $U = \{P_{i_1}, \cdots, P_{i_{t-1}}\}$ attempts to determine the secret. Then, the $(t-1)$–uple $\delta_U = (B_{j_1}, \cdots, B_{j_{t-1}})$ representing the shares distribution to the players in U occurs in the columns $j_1, \cdots, j_{t-1}$ in ν rows of the array. By considering also the last column, one obtains a t–uple $(B_{j_1}, \cdots, B_{j_{t-1}}, A(i_h, n+1))$ for all $1 \leqslant h \leqslant \nu$. Since $\lambda = 1$, it must be the case that all these t–uples are distinct, and thus they must differ in the last component. Hence, the ν rows containing the $(t-1)$–uple $(B_{j_1}, \cdots, B_{j_{t-1}})$ determine a permutation on the last column, which means that the conditional probability of K given the shares distribution δ_U equals

$$\Pr[K|\delta_U] = \Pr[K] = \frac{1}{\nu} \quad . \tag{48}$$

Since this result is independent from the particular value of the key K or the shares distribution δ_U, it follows that this threshold scheme is perfect. Remark also that both $\mathcal{K}$ and $\mathcal{B}$ correspond to the support set X of the OA, and thus they have the same cardinality. Assuming that $X = \Sigma^m$ (i.e. , X is described by a set of strings of length m over the alphabet Σ), this means that the threshold scheme based on the OA is also ideal. We summarize these facts in the following result:

Theorem 9. *Let A be a t–$(v, n+1, 1)$–OA. Then, there exists a perfect and ideal (t, n)–threshold scheme.*

BOOLEAN FUNCTIONS FOR CRYPTOGRAPHY

5

Boolean functions play a fundamental role in several areas of computer science, ranging from complexity theory to machine learning [47]. In cryptography, the computational security of a symmetric cipher often relies on the properties of the particular Boolean functions that underlie its design.

In this chapter, we cover all necessary background notions about the cryptographic properties of Boolean functions and their vectorial counterparts, S-boxes. Throughout this chapter, we follow Carlet [30, 31] as a reference for our discussion respectively on Boolean functions and S-boxes. Other reference works on Boolean functions for cryptography include [48, 109].

The rest of this chapter is structured as follows. In Section 5.1 we give the basic definitions and presents four representation methods of Boolean functions. We then overview in Section 5.2 the main cryptographic properties of Boolean functions considered in the next chapters of this thesis, reporting the various trade-offs among them and introducing the relation of affine equivalence. Next, Section 5.4 focuses on the basic definitions and representations of S-boxes, while Section 5.5 gives an overview of their cryptographic criteria. Finally, Section 5.6 defines affine equivalence for vectorial Boolean functions, and mentions two other more general equivalence relations, namely EA and CCZ equivalence.

5.1 BASIC DEFINITIONS AND REPRESENTATIONS OF BOOLEAN FUNCTIONS

A *Boolean function* of $n \in \mathbb{N}$ variables is a mapping $f : \mathbb{F}_2^n \to \mathbb{F}_2$. The basic way to represent a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is by means of its *truth table*, which specifies for each of the possible 2^n input vectors of $\mathbb{F}_2^n$ the corresponding output value of f. Hence, for any $n \in \mathbb{N}$ the set $\mathcal{F}_n$ of Boolean functions of n variables is composed of 2^{2^n} functions. Once an ordering of the input variables $x_1, \cdots, x_n$ has been established, a truth table can be compactly described just by the 2^n-bit string $\Omega(f) \in \mathbb{F}_2^{2^n}$ representing the output values of the corresponding function.

Another common representation of Boolean functions is the *Algebraic Normal Form* (ANF). In particular, the ANF of $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is defined by the following multivariate polynomial:

$$P_f(x) = \bigoplus_{I \in 2^{[n]}} a_I \left(\prod_{i \in I} x_i \right) , \tag{49}$$

Hence, the ANF represents a Boolean function as a sum of products over $\mathbb{F}_2$. Remark that this representation is unique, since the mapping which associates a Boolean function to its ANF is a bijection from $\mathcal{F}_n$ to the quotient ring $\mathbb{F}_2[x_1, \cdots, x_n]/(x_1^2 \oplus x_1, \cdots, x_n^2 \oplus x_n)$. The *algebraic degree* of a Boolean function f is the cardinality of the largest subset $I \in 2^{[n]}$ in the ANF of f such that $a_I \neq 0$. Boolean functions having degree $d = 1$ are called *affine* functions. In particular, an affine function is *linear* if $a_\varnothing = 0$, i. e. if the constant term in its ANF is null. Given $\omega \in \mathbb{F}_2^n$, the linear Boolean function $L_\omega : \mathbb{F}_2^n \to \mathbb{F}_2$ is defined for all $x \in \mathbb{F}_2^n$ as $L_\omega(x) = \omega \cdot x$.

The relationship between the ANF coefficients and the truth table of f is given by the *Möbius transform*, defined for all $x \in \mathbb{F}_2^n$ as:

$$f(x) = \bigoplus_{I \subseteq \mathrm{supp}(x)} a_I , \tag{50}$$

where $\mathrm{supp}(x) = \{i : x_i \neq 0\}$ is the *support* of x.

A third representation which is useful to characterize several cryptographic properties of Boolean functions is the *Walsh transform*.

Definition 23. *The* Walsh transform *(also the* Walsh-Hadamard transform*) of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is the function $W_f : \mathbb{F}_2^n \to \mathbb{R}$ defined for all $\omega \in \mathbb{F}_2^n$ as*

$$W_f(\omega) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus \omega \cdot x} , \tag{51}$$

In particular, the coefficient $W_f(\omega)$ measures the *correlation* between f and the linear function L_ω selected by ω. The multiset $\mathcal{S}(f)$ of all Walsh coefficients of f is also called the *Walsh spectrum* of f, while the maximum coefficient in absolute value is called the *spectral radius* or the *linearity* of f.

A property satisfied by all Boolean functions is *Parseval's relation*, which states that the sum of all squared Walsh coefficients is constant. Formally, for all $f : \mathbb{F}_2^n \to \mathbb{F}_2$ it holds

$$\sum_{\omega \in \mathbb{F}_2^n} [W_f(\omega)]^2 = 2^{2n} . \tag{52}$$

The *inverse Walsh transform* is used to recover the truth table of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ starting from its Walsh coefficients. In

particular, the *polar value* $\hat{f}(x) = (-1)^{f(x)}$ corresponding to $x \in \mathbb{F}_2^n$ equals

$$\hat{f}(x) = 2^{-n} \sum_{\omega \in \mathbb{F}_2^n} \hat{F}(\omega) \cdot (-1)^{\omega \cdot x} \quad . \tag{53}$$

A straightforward method to compute the Walsh transform would be to simply perform the sum in Equation (54) for each vector $\omega \in \mathbb{F}_2^n$. Since there are 2^n vectors $\omega \in \mathbb{F}_2^n$ and that each coefficient is determined by a sum of 2^n products, it follows that this naive algorithm would require $\mathcal{O}(2^{2n})$ steps to compute the Walsh transform of a Boolean function with n variables. However, there exists a divide-and-conquer recursive algorithm that allows one to compute the Walsh transform in $\mathcal{O}(n \cdot 2^n)$ steps. Further information on this *Fast Walsh Transform* (FWT) algorithm can be found in [30]. Notice that, since Equation (53) still describes a Walsh transform up to normalization by a constant, the same FWT algorithm can be adapted to compute the inverse Walsh transform as well.

Finally, a fourth representation of Boolean functions which we will use in this thesis is the *autocorrelation function*:

Definition 24. *The* autocorrelation function *of* $f : \mathbb{F}_2^n \to \mathbb{F}_2$ *is the function* $\hat{r}_f : \mathbb{F}_2^n \to \mathbb{R}$ *defined for all* $s \in \mathbb{F}_2^n$ *as*

$$\hat{r}_f(s) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus f(x \oplus s)} \quad . \tag{54}$$

The maximum absolute value AC_{max} for $s \in \mathbb{F}_2^n \setminus \{0\}$ of the autocorrelation function is called the *absolute indicator* of f. The *Wiener-Khinchin-Einstein theorem*[1] states a relationship between the autocorrelation function and the Walsh transform.

Theorem 10 (Wiener-Khinchin-Einstein)**.** *Let* $f : \mathbb{F}_2^n \to \mathbb{F}_2$ *be a Boolean function. The following equality holds for all* $\omega \in \mathbb{F}_2^n$*:*

$$[W_f(\omega)]^2 = \sum_{s \in \mathbb{F}_2^n} \hat{r}_f(s) \cdot (-1)^{\omega \cdot s} \quad . \tag{55}$$

The above theorem implies that the autocorrelation function can also be computed through the FWT algorithm. In particular, it suffices to compute the Walsh transform of f, square all its coefficient and finally apply the inverse Walsh transform in order to obtain the autocorrelation function.

5.2 CRYPTOGRAPHIC PROPERTIES OF BOOLEAN FUNCTIONS

The notion of computational security defined in Section 4.1 is usually declined with respect to specific types of cryptanalysis that a symmetric key cryptosystem must be resistant to. This rationale translates to

1 Usually, this theorem is referred in the literature just as the *Wiener-Khinchin* theorem. However, in [67] it is shown that Einstein already proved a similar result twenty years before the independent discoveries by Wiener and Khinchin.

the fact that the Boolean functions involved in the design of block and stream ciphers must satisfy certain criteria in order to resist to various kinds of attacks. In this section we consider five cryptographic properties of Boolean functions, namely *balancedness*, *algebraic degree*, *nonlinearity*, *resiliency* and *propagation criterion*, and discuss their relevance with respect to the computational security of stream ciphers. Next, we discuss some theoretical bounds that these properties induce among themselves, and conclude by introducing the notion of *affine equivalence* of Boolean functions.

5.2.1 *Balancedness*

A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is *balanced* if

$$|f^{-1}(0)| = |f^{-1}(1)| = 2^{n-1} \ . \tag{56}$$

This means that the truth table of such functions is a string composed by an equal number of zeros and ones. Balancedness is easily characterized by the Walsh transform. In particular, f is balanced if and only if the Walsh transform over the null vector $\underline{o}$ is null:

$$W_f(\underline{o}) = 0 \ . \tag{57}$$

Balancedness is a fundamental criterion that every Boolean function used in symmetric ciphers should satisfy. In fact, unbalanced functions present a statistical bias that can be exploited for linear and differential cryptanalysis.

5.2.2 *Algebraic Degree*

In Section 5.1 we defined the algebraic degree $\deg(f)$ of a Boolean function f as the cardinality of the largest subset of variables with nonzero ANF coefficient. In the case of stream ciphers and cryptographic PRNGs, the algebraic degree of the involved Boolean functions is related to the notion of *linear complexity*: given a sequence produced by a PRNG, its linear complexity is the length of the shortest LFSR which generates it.

PRNGs which produce sequences having low linear complexity can be attacked using the *Berlekamp-Massey algorithm* [115]. In [158] it is shown that PRNGs based on Boolean functions having high algebraic degree also have a high linear complexity, making the Berlekamp-Massey algorithm computationally unfeasible. Thus, the algebraic degree of Boolean functions used in the design of stream ciphers and cryptographic PRNGs should be as high as possible.

5.2.3 *Nonlinearity*

Considering the truth table representation, cryptographic Boolean functions should lie at a high Hamming distance from all affine functions. Formally, the *nonlinearity* of a Boolean function f is defined as follows.

Definition 25. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function, and let $\mathcal{A}$ be the set of all affine functions in n variables. The* nonlinearity *of f is defined as*

$$N_f = d_H(f, \mathcal{A}) = \min\{d_H(\Omega(f), \Omega(a)) : \ a \in \mathcal{A}\} \ .$$

When used in stream ciphers, Boolean functions having low nonlinearity may expose to *fast-correlation attacks* (for details, see for example [39]). For this reason, the nonlinearity should be as high as possible, to provide better confusion.

The following result gives a simple formula to compute the nonlinearity of a Boolean function by means of its Walsh transform.

Lemma 3. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function. Denoting by $\mathcal{L}(f)$ the spectral radius of f, the nonlinearity of f is equal to*

$$N_f = 2^{n-1} - \frac{1}{2}\mathcal{L}(f) \ . \tag{58}$$

In order to have high nonlinearity, the spectral radius must be low. An interesting problem is to determine what is the maximum nonlinearity obtainable by a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$. Recalling Parseval's relation (52), we know that for each Boolean function in n variables the sum of all its squared Walsh coefficients must equal 2^{2n}. Hence, to reach the minimum spectral radius the squared spectrum of the function must be uniformly divided among all 2^m vectors, which means that $\hat{F}^2(\omega) = 2^n$ for all $\omega \in \mathbb{F}_2^n$. Thus, the Walsh coefficients must all have the same absolute value $2^{\frac{n}{2}}$, giving the following upper bound on nonlinearity:

$$N_f \leqslant 2^{n-1} - 2^{\frac{n}{2}-1} \ . \tag{59}$$

Clearly, equality in (59) can occur only if n is even, since the Walsh coefficients of a Boolean function must be integer numbers. The Boolean functions achieving this bound are called *bent* functions, and by Equation (57) they are not balanced, since $W_f(\underline{0}) = \pm 2^{\frac{n}{2}} \neq 0$. Relation (59) is also called the *covering radius bound*, since if n is even it coincides with the maximum covering radius of the *Reed-Muller code* $R(1, n)$ [117]. When n is odd, we can uniformly divide the squared Walsh spectrum over a subset $S \subseteq \mathbb{F}_2^n$ of 2^{n-1} vectors. Since by Parseval's relation the remaining 2^{n-1} vectors must all have null Walsh coefficients, it follows that $W_f(\omega) = \pm 2^{\frac{n+1}{2}}$ for all $\omega \in S$. The bound on nonlinearity for this kind of functions then becomes

$$N_f \leqslant 2^{n-1} - 2^{\frac{n+1}{2}-1} \ . \tag{60}$$

Equality in 60 can be obtained by *quadratic* functions (that is, Boolean functions which have algebraic degree $d = 2$). For this reason, this inequality is also called the *quadratic bound*. For $n = 1, 3, 5, 7$ it is known that the maximum nonlinearity coincides with the quadratic bound. For $n > 7$ odd, however, this is still an open problem. In [91] it was proved that the maximum nonlinearity strictly exceeds the quadratic bound if $n > 7$, but a definitive upper bound for the general case, included between the quadratic bound and the covering radius bound, is yet to be discovered.

5.2.4 *Correlation Immunity and Resiliency*

Another essential cryptographic criterion for Boolean functions, introduced by Siegenthaler in [167], is *correlation immunity*. In what follows, we denote by $w_H(x)$ the Hamming weight of vector x, i. e. the number its nonzero coordinates.

Definition 26. *A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is t–th order correlation immune (with $1 \leqslant t \leqslant n$) if the truth tables of the restrictions of f obtained by fixing at most t input coordinates all have the same Hamming weight.*

We denote by $CI(t)$ a t–th order correlation immune function. A Boolean function f which is both balanced and t–th order correlation immune is also called t-*resilient*. This means that each restriction of f where at most t input variables are fixed is balanced.

Siegenthaler showed in [168] that if the Boolean functions used in a stream cipher are not t–resilient then it is possible to apply an efficient *correlation attack* using t LFSRs.

A Walsh characterization of correlation immunity has been proved by Xiao and Massey in [196].

Theorem 11. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function. Then, f is t–th order correlation immune if and only if $W_f(\omega) = 0$ for all vectors $\omega \in \mathbb{F}_2^n$ such that $w_H(\omega) \leqslant t$.*

Hence, by Equation (57) and Theorem 11, one can check if a given Boolean function is t–resilient by verifying that its Walsh transform vanishes for all vectors ω having Hamming weight at most t.

5.2.5 *Strict Avalanche Criterion and Propagation Criterion*

Webster and Tavares [190] defined the *Strict Avalanche Criterion* (SAC) as a more stringent property than the *avalanche effect*. If a Boolean function f satisfies the SAC, then the probability that the output of f changes whenever a single input bit is complemented is $\frac{1}{2}$. The practical consequence in using Boolean functions satisfying the SAC in cryptosystems is that plaintexts "close" to each other (that is, separated by a small Hamming distance) produce ciphertexts which are

completely different. Hence, this property is related to the diffusion principle of symmetric ciphers.

A generalization of the SAC, described in [150], is the *propagation criterion* of order l, which takes into account the complementation of at most l input bits.

Definition 27. *A Boolean function* $f : \mathbb{F}_2^n \to \mathbb{F}_2$ *satisfies the* propagation criterion of order l *(with* $1 \leqslant l \leqslant n$*) if the function* $f(x) \oplus f(x \oplus s)$ *is balanced for all nonzero vectors* $s \in \mathbb{F}_2^n$ *having Hamming weight at most* l.

Functions satisfying the propagation criterion of order l are also called $PC(l)$ functions. Similarly to resiliency, in [150] a characterization of the propagation criterion based on the zeros of the autocorrelation function was proved.

Theorem 12. *A Boolean function* $f : \mathbb{F}_2^m \to \mathbb{F}_2$ *is* $PC(l)$ *if and only if* $\hat{r}_f(s) = 0$ *for all* $s \in \mathbb{F}_2^n$ *such that* $1 \leqslant w_H(s) \leqslant l$.

5.2.6 *Theoretical Bounds*

Generally, the cryptographic properties described in the previous sections cannot be satisfied simultaneously by a Boolean function, since they induce several trade-offs among them. A complete survey regarding these trade-offs can be found in [30]: in what follows, we report only the most useful ones for our results.

The first result pertains to resiliency and algebraic degree. In [167] Siegenthaler proved the following bound.

Theorem 13 (Siegenthaler's bound)**.** *Let* $f : \mathbb{F}_2^n \to \mathbb{F}_2$ *be a* t*–resilient Boolean function with algebraic degree* d*. Then, it holds that*

$$d \leqslant n - t - 1 \ . \tag{61}$$

The second result concerns nonlinearity and order of resiliency. In particular, Tarannikov [177] showed an upper bound on the maximum nonlinearity obtainable by a t–resilient function.

Theorem 14 (Tarannikov's bound)**.** *Let* $f : \mathbb{F}_2^n \to \mathbb{F}_2$ *be a* t*–resilient Boolean. Then, the following relation holds:*

$$N_f \leqslant 2^{n-1} - 2^{t+1} \ . \tag{62}$$

In what follows, by (n, t, d, nl) we denote the *profile* of a balanced boolean function of n variables having resiliency order t, algebraic degree d and nonlinearity nl.

An interesting class of Boolean functions is the set of *plateaued functions*, originally introduced by Zhang and Zheng [200]. Formally, a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ with spectral radius $\mathcal{L}(f)$ is plateaued if $W_f(\omega) \in \{-\mathcal{L}(f),\ 0,\ +\mathcal{L}(f)\}$ for all $\omega \in \mathbb{F}_2^n$. Thus, Walsh spectra of plateaued functions take at most three values. Plateaued functions are especially interesting for cryptography, since they satisfy

with equality both Siegenthaler's and Tarannikov's bounds, a feature which makes them optimal with respect to algebraic degree, nonlinearity and resiliency. The profile of a plateaued boolean function is of the form $(n, r-2, n-r-3, 2^{n-1}-2^{r-1})$, where $r \geqslant \frac{n}{2}$, from which it follows that $\mathcal{L}(f) = 2^r$. Notice that if n is even and $r = \frac{n}{2}$, then a plateaued function is bent.

We conclude by mentioning a bound concerning t–resilient and $PC(l)$ Boolean functions, which was proved in [36].

Theorem 15. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a $PC(l)$ boolean function. Then, the order t of resiliency of f is upper bounded by*

$$t \leqslant n - l - 1 \ . \tag{63}$$

5.3 AFFINE EQUIVALENCE FOR BOOLEAN FUNCTIONS

As we saw in Section 5.1, the number of Boolean functions of n variables is 2^{2^n}. This makes the search of Boolean functions with good cryptographic properties very difficult, since the corresponding space $\mathcal{F}_n$ is too huge to be exhaustively explored even for small values of n. The notion of *affine equivalence* among Boolean functions has been introduced to address this enumeration problem:

Definition 28. *Let $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$ be two Boolean functions of n variables. Then, f and g are* affine equivalent *if there exists an affine permutation $P : \mathbb{F}_2^n \to \mathbb{F}_2$ such that*

$$g(x) = f(A(x)) \tag{64}$$

for all $x \in \mathbb{F}_2^n$.

One can easily check that affine equivalence is actually an equivalence relation. Interestingly, the cryptographic properties of balancedness, algebraic degree and nonlinearity are all preserved under affine equivalence [30]. Thus, one can study these three properties just by considering the equivalence classes induced by these relation, which are considerably less than the total number of Boolean functions of n variables. For example, in the case of $n = 5$ variables there are $2^{2^5} = 4294967296$ functions in total, but the quotient space induced by the affine equivalence relation is composed of only 48 classes. Berlekamp and Welch classified these 48 classes with respect to their cryptographic properties in [13].

5.4 BASIC DEFINITIONS AND REPRESENTATIONS OF S-BOXES

We now turn to vectorial Boolean functions. Let $n, m \in \mathbb{N}$. A *vectorial Boolean function* (also called a (n, m)–function or a *S-box* in the cryptographic context) is a mapping $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ with n input variables

and m outputs. By $f_1, \cdots, f_m : \mathbb{F}_2^n \to \mathbb{F}_2$ we denote the *coordinate functions* of F, that is, the m Boolean functions which specify the value of each output bit of F. More precisely, $F(x)$ is defined for all $x \in \mathbb{F}_2^n$ as:

$$F(x_1, \cdots, x_n) = (f_1(x_1, \cdots, x_n), \cdots, f_m(x_1, \cdots, x_n)) \ . \tag{65}$$

The *component functions* of F are defined as $v \cdot F$ for all $v \in \mathbb{F}_2^{m^*}$, with $\mathbb{F}_2^{m^*} = \mathbb{F}_2^m \setminus \{0\}$. Since

$$v \cdot F = v_1 f_1(x_1, \cdots, x_n) \oplus \cdots \oplus v_m f_m(x_1, \cdots, x_n) \ ,$$

it follows that the component functions are the (non-trivial) linear combinations of the coordinate functions of F.

Remark 1. *Recall that in Section 2.2 we defined the global rule of a finite CA in terms of its local rule $f : \Sigma^d \to \Sigma$ as follows:*

- *No Boundary CA: $F : \Sigma^n \to \Sigma^{n-d+1}$ is defined for all $x \in \Sigma^n$ as:*

$$F(x_1, x_2, \cdots, x_n) = (f(x_1, \cdots, x_d), \cdots, f(x_{n-d+1}, \cdots, x_n)) \ . \tag{66}$$

- *Periodic Boundary CA: $F : \Sigma^n \to \Sigma^n$ is defined for all $x \in \Sigma^n$ as:*

$$F(x_1, x_2, \cdots, x_n) = (f(x_1, \cdots, x_d), \cdots, f(x_n, \cdots, x_{d-1})) \ . \tag{67}$$

Hence, when the state alphabet is $\Sigma = \mathbb{F}_2$, the global rule of a finite CA is a vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ whose coordinate functions correspond to the local rule f applied to the neighborhood $(x_i, \cdots, x_{i+d-1})$. In the NBCA case we have $m = n - d + 1$ while for PBCA it holds $m = n$.

The truth table representation can be easily extended to the vectorial case: given $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$, one can simply concatenate the truth tables of the m coordinate functions of F:

$$\Omega(F) = (\Omega(f_1), \Omega(f_2), \cdots, \Omega(f_m)) \ . \tag{68}$$

It follows that the truth table of F has size $m \cdot 2^n$. Thus, the cardinality of the set $\mathcal{F}_{n,m}$ of all (n, m)–functions is $2^{m \cdot 2^n}$.

The algebraic normal form and the Walsh transform are also defined similarly to their single-output counterparts which we introduced in Section 5.1. In particular, given $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $x \in \mathbb{F}_2^n$ the ANF of $F(x)$ equals

$$P_f(x) = \bigoplus_{I \in 2^{[n]}} a_I \left(\prod_{i \in I} x_i \right) \ , \tag{69}$$

the only difference being that the coefficient a_I is an element of $\mathbb{F}_2^m$ instead of $\mathbb{F}_2$ as for the ANF of a single-output Boolean function.

On the other hand, the Walsh transform of F is defined in terms of its components functions. Thus, we have

$$W_{v\cdot F}(\omega) = \sum_{x\in\mathbb{F}_2^n} (-1)^{v\cdot F(x)\oplus\omega\cdot x} \tag{70}$$

for all $\omega \in F_2^n$ and $v \in \mathbb{F}_2^{m^*}$.

The autocorrelation function of F is defined in analogous way, but we will not use it in this thesis.

5.5 CRYPTOGRAPHIC PROPERTIES OF S-BOXES

In this section we show how the vectorial counterparts of the cryptographic properties of Boolean functions are characterized in terms of either the coordinates or the component functions of S-boxes. We frame each property with respect to the cryptanalytic attacks that can be performed on the S-boxes of block ciphers.

5.5.1 *Balancedness*

A vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is balanced if for all output vectors $y \in \mathbb{F}_2^m$ it results that $|F^{-1}(y)| = 2^{n-m}$. Equivalently, F is balanced if and only if all its component functions are balanced, i. e. if the following relation

$$W_{v\cdot F}(\underline{o}) = 0 \tag{71}$$

holds for all $v \in \mathbb{F}_2^{m^*}$. Remark that balancedness of the coordinate functions is a necessary but not sufficient condition for the balancedness of F: it can be the case that certain linear combinations of balanced coordinate functions yield unbalanced component functions.

Balanced vectorial Boolean functions are critical components in the design of confusion layers for block ciphers, since they ensure that each output vector $y \in \mathbb{F}_2^m$ has a uniform probability of occurring, given that the input vectors are randomly sampled. Hence, balancedness constitutes a first security measure to stand statistical cryptanalysis by the attacker. Moreover, (n, n)–functions actually correspond to bijective S-boxes, which are fundamental to ensure decryption in SPN block ciphers, as we mentioned in Section 4.2. It is easy to determine the number of bijective S-boxes of n variables: each n-bit vector can be indexed by a natural number $i \in \{0, \cdots, 2^n - 1\}$. Consequently, a bijective (n, n)–function can be seen as a permutation of 2^n natural numbers. This means that there are $(2^n)!$ bijective n-bit S-boxes, i. e. the size of the symmetric group over a set of 2^n objects.

5.5.2 *Algebraic degree*

The algebraic degree of a vectorial function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is by definition the degree of its ANF, and it corresponds to the maximal degree of the coordinate functions of F.

S-boxes with a low algebraic degree are vulnerable to *higher-order differential attacks* [98], which are a generalization of the differential attack introduced by Biham and Shamir [16] based on the formal derivative of Boolean functions.

5.5.3 *Nonlinearity*

Similarly to the single-output case, the nonlinearity of a vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2$ corresponds to its Hamming distance from all affine functions. This property can be characterized as the *minimal* nonlinearity among all component functions of F, that is

$$N_F = \min_{v \in \mathbb{F}_2^{n^*}} \left\{ 2^{n-1} - \frac{1}{2} \mathcal{L}(v \cdot F) \right\} , \tag{72}$$

where $L(v \cdot F) = \max_{\omega \in \mathbb{F}_2^n} \{|W_{v \cdot F}(\omega)|\}$ is the linearity of the component function $v \cdot F$.

S-boxes used in the design of block ciphers (both under the SPN and FN approaches) must have high nonlinearity in order to resist *linear cryptanalysis* attacks [116].

Since Boolean functions are a particular case of vectorial functions, the *covering radius bound* also holds for any (n, m)–function:

$$N_F \leqslant 2^{n-1} - 2^{\frac{n}{2}-1}. \tag{73}$$

As in the single-output case, vectorial functions satisfying the above bound are called *bent*. Bent (n, m)–functions exist only for even values of n and $m \leqslant n/2$ [31].

When $m = n$, a better bound exists. The *Sidelnikov-Chabaud-Vaudenay bound* [35] states that the nonlinearity of any (n, n)–function F satisfies the following inequality:

$$N_F \leqslant 2^{n-1} - 2^{\frac{n-1}{2}}. \tag{74}$$

Functions satisfying with equality bound (74) are called *Almost Bent* (AB) functions. As one may notice, these functions can exist only if n is odd. AB functions provide an optimal resistance to linear cryptanalysis attacks.

5.5.4 *Differential Uniformity*

Let F be a (n, m)–function, with $a \in \mathbb{F}_2^{n^*}$ and $b \in \mathbb{F}_2^m$. We define the *difference distribution set* of F with respect to a and b as:

$$D_F(a, b) = \{x \in \mathbb{F}_2^n : F(x) \oplus F(x \oplus a) = b\}. \tag{75}$$

The *difference table* of F is a $2^n - 1 \times 2^m$ table where for all $a \in \mathbb{F}_2^{n^*}$ and $b \in \mathbb{F}_2^m$ the entry at position (a, b) corresponds to the cardinality of the delta difference set $D_F(a, b)$, and it is denoted as $\delta_F(a, b)$. The *differential uniformity* δ_F is then defined as [128]:

$$\delta_F = \max_{\substack{a \in \mathbb{F}_2^{n^*} \\ b \in \mathbb{F}_2^m}} \delta_f(a, b). \tag{76}$$

Ideally, the differential uniformity of the S-boxes used in a block cipher should be as low as possible in order to provide better resistance to differential cryptanalysis attacks [16].

When $m < n$, it can be proved that the minimum value achievable for differential uniformity is 2^{n-m}. Functions satisfying this bound are called *Perfect Nonlinear* (PN), and it can be shown that they coincide with the class of bent vectorial functions [128]. On the other hand, for $m = n$ it follows that the minimum value for differential uniformity is 2, due to the fact that if $x \in \mathbb{F}_2^n$ is a solution to the equation $F(x) \oplus F(x \oplus a) = b$, then $x \oplus a$ is also a solution. S-boxes reaching this lower bound are called *Almost Perfect Nonlinear* (APN) functions [130].

Every AB function is also APN, but the converse does not hold in general [32]. Contrary to AB functions, APN functions also exist for even number of variables. A common method to search for APN functions is to use *power functions*. In this case, one represents a vectorial Boolean function as a univariate polynomial, by identifying the vector space $\mathbb{F}_2^n$ as the finite field $\mathbb{F}_{2^n}$. Then, power functions are monomials of the form $F(x) = x^d$. In this case, one can show that it suffices to check the APN property only for $a = 1$ [31]. A result reported in [31] and proved by Dobbertin shows that if n is odd then APN power functions are also permutations (i. e. bijective S-boxes). On the other hand, APN power functions are not permutations for any even n.

The existence of APN permutation with even number of variables remained open until 2009, when Dillon showed an example of an APN permutation of 6 variables [23]. However, for all n even and greater than 6 the problem is still open.

5.5.5 *Resiliency*

Resiliency for vectorial functions is defined analogously to the single-output case. In particular, $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is t–resilient if, by fixing any t input variables $x_{i_1}, \cdots, x_{i_t}$, the resulting restriction $\tilde{F} : \mathbb{F}_2^{n-t} \to \mathbb{F}_2^m$ is a balanced vectorial function, i. e. $|\tilde{F}^{-1}(y)| = 2^{n-t-m}$ for all $y \in \mathbb{F}_2^m$. Note that t–resilient functions can exist only if $m < n - t$.

Similarly to nonlinearity and balancedness, the resiliency of a vectorial function can also be characterized by the resiliency of its component functions, as the next result [31] shows:

Lemma 4. *Let* $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ *be a vectorial Boolean function in* n *variables and* m *outputs. Then,* F *is* t*–resilient if and only if for all* $v \in (\mathbb{F}_2^m)^*$ *the component function* $v \cdot F$ *is* t*-resilient.*

Resilient (n, m)–functions are useful in the design of PRNG for stream ciphers, and also for achieving *key renewal*: suppose that Oscar managed to discover at most t bits of a key $K \in \mathbb{F}_2^n$ used by Alice and Bob in a symmetric cryptosystem. Then, Alice and Bob can apply a t–resilient function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ to get a new key $K_1 = F(K)$ about which Oscar does not know anything, for the restriction of F to the t bits he found is balanced. Clearly, the downside of this solution is that the new key is shorter than the original one.

As the next result reported in [173] shows, linear resilient vectorial functions are equivalent to binary linear codes.

Theorem 16. *There exists a* (n, m, d) *binary linear code* C *if and only if there exists a* t*–resilient linear* (n, m)*–function such that* $t = d - 1$.

Resilient vectorial functions are also related to a particular type of combinatorial designs called *Large sets of Orthogonal Arrays*. (LOA). Formally, a *large set of* $t - (v, k, \lambda)$ *orthogonal arrays* ($t - (v, k, \lambda)$–LOA), is defined a set of v^{k-t}/λ simple $t - (v, k, 1)$–OA such that each k-tuple of symbols occurs in exactly one of the orthogonal arrays of the set. The following result [173] illustrates the relation between resilient functions and LOAs:

Theorem 17. *A* $t - (2, n, 2^{n-m-t})$*–LOA exists if and only if there exists a* t*–resilient* (n, m)*–function.*

5.6 AFFINE EQUIVALENCE FOR S-BOXES

Equivalence relations are useful also in the context of S-boxes, in order to reduce the search space among all vectorial Boolean functions with interesting cryptographic properties. In this thesis, we consider only the case of equivalence relations over (n, n)–functions, but all definitions reported in this section can be straightforwardly adapted to (n, m)–functions as well.

The notion of affine equivalence is generalized to the vectorial case as follows:

Definition 29. *Two* (n, n)*–functions* $F, G : \mathbb{F}_2^n \to \mathbb{F}_2^n$ *are called* affine equivalent *(AE) if there exist two affine permutations* $A_1, A_2 : \mathbb{F}_2^n \to \mathbb{F}_2^n$ *such that*

$$G(x) = A_2(F(A_1(x))) \tag{77}$$

for all input vectors $x \in \mathbb{F}_2^n$.

All the cryptographic properties discussed in this section for vectorial Boolean functions are invariant under affine equivalence.

In the literature, there are also other more general equivalence relations for S-boxes. We mention them here for the sake of completeness, even though in this thesis we will focus mainly on AE. The *Extended Affine* (EA) equivalence [31] is similar to the AE notion reported in the definition above, but in this case the function G is also summed (i. e. , XORed) with a third affine permutation $A_3 : \mathbb{F}_2^n \to \mathbb{F}_2^n$. In the *Carlet-Charpin-Zinoviev* (CCZ) equivalence, on the other hand, two (n, n)–functions F and G are equivalent if and only if one can obtain the *graph* Γ_G of G by composing the graph Γ_F of F with an affine permutation over $\mathbb{F}_2^n$ [32].

AE is a special case of EA equivalence, which is in turn a special case of CCZ equivalence. However, not all cryptographic properties mentioned above are invariant under these equivalence relations. For example, the algebraic degree of an (n, n)–function is not preserved under CCZ equivalence [31].

HEURISTIC OPTIMIZATION ALGORITHMS

6

A *combinatorial optimization problem* is defined as a function $\mathcal{P} : \mathcal{I} \to \mathcal{S}$ which maps a set $\mathcal{I}$ of *problem instances* to a family $\mathcal{S}$ of *solution spaces*. For all $I \in \mathcal{I}$, the solution space $S = \mathcal{P}(I)$ is a finite set equipped with an *objective function* $f : S \to \mathbb{R}$ assigning to each *candidate solution* $x \in S$ a measure of how good x is in solving that particular problem instance. The goal of combinatorial optimization is to find an *optimal solution* x^* that maximizes f, that is

$$x^* = \mathrm{argmax}_{x \in S}\{f(x)\} \ . \tag{78}$$

In most combinatorial optimization problems of practical importance, the solution space S is usually too huge to be explored in an exhaustive manner. Hence, one has to resort to *heuristic algorithms* in order to find a (sub)optimal solution in a reasonable amount of time.

In this chapter, we recall the basic concepts of the heuristic algorithms which we will use in the rest of this thesis. We begin in Section 6.1 by describing the two *local search methods* of *Hill Climbing* and *Simulated Annealing*, which starts from an initial candidate solutions and gradually improves it by slight modifications. We then introduce in Section 6.2 *Genetic Algorithms* and *Genetic Programming*, which on the other hand evolve a *population* of candidate solutions by leveraging on principles based on biological evolution. We finally conclude the chapter in Section 6.3 by discussing *Particle Swarm Optimization*, another population-based heuristic which is inspired by the movements of swarms.

6.1 LOCAL SEARCH METHODS

Let $d_S : S \times S \to \mathbb{R}$ be a distance defined over the solution space S. Similarly to the situation arising in error-correcting codes that we encountered in Section 3.3, we assume that there exists a *minimum distance* d_m such that $d_S(x, x') \geqslant d_m$ holds for each pair of distinct solutions $x, x' \in S$. We thus define the *neighborhood* $N(x)$ of a solution $x \in S$ as the set of all solutions $y \neq x$ with minimal distance from x:

$$N(x) = \{y \in S : \ \forall z \in S \ d_S(z, x) \geqslant d_S(y, x)\} \ . \tag{79}$$

A *local search method* (or local optimization algorithm) usually works by randomly generating an initial candidate solution $x \in S$, evaluating its objective function $f(x)$ and then by replacing it with a new solution $y \in N(x)$ from its neighborhood. The selection method for the new solution y depends on the underlying algorithm.

6.1.1 *Hill Climbing*

The *Hill Climbing* (HC) optimization algorithm always selects the best solution in the neighborhood (i. e. , the one maximizing $f(x)$). Formally, the current solution x is replaced by y, where

$$y = \mathrm{argmax}_{z \in N(x)}\{f(z)\} \ . \tag{80}$$

This process is iterated until a solution x is produced whose neighbors all have an equal or smaller objective function value than that of x. Clearly, if the optimization problem features several local optima, HC is very likely to get stuck in one of them.

6.1.2 *Simulated Annealing*

Simulated Annealing (SA) is a more refined local search algorithm, which is inspired by the process of metal annealing [97]. During a SA iteration, only one solution y in the neighborhood of x is generated, usually in a random way. If the objective function computed over y is higher than the objective function value of x, then x is replaced by y. Otherwise, x is replaced by y with a certain *acceptance probability* P_a. This acceptance probability does not remain constant during the execution of the algorithm: as a matter of fact, P_a is computed through a *temperature parameter* T, which is decreased at each SA iteration by a *cooling schedule*. Hence, the probability of accepting a solution which is worse than the current one decreases over time.

Formally, for all solutions $x \in S$ and $y \in N(x)$ generated by the SA algorithm the acceptance probability P_a is defined as follows:

$$P_a = \begin{cases} 1 & , \quad \text{if } f(x) < f(y) \\ e^{-\left(\frac{|f(y)-f(x)|}{T}\right)} & , \quad \text{if } f(x) \geqslant f(y) \end{cases} \tag{81}$$

while at each iteration the temperature parameter is updated as:

$$T \leftarrow \alpha T \ , \tag{82}$$

where $\alpha \in (0, 1)$.

In this way, the SA algorithm favors *exploration* of the search space during the first optimization steps, and then stabilize towards *exploitation* of the current solution in later steps. This results in an heuristic which is able to escape local optima more easily than Hill Climbing.

6.2 EVOLUTIONARY COMPUTATION ALGORITHMS

6.2.1 *Genetic Algorithms*

Genetic Algorithms (GAs) are an heuristic search method loosely based on the principles of evolution theory, in particular *natural selection*.

They belong to a broader class of nature-inspired optimization methods called *Evolutionary Computation* (EC) algorithms. Holland [79] originally introduced GAs to pursue two objectives: the abstraction of the adaptive processes in natural systems, and the design of artificial systems which would take advantage of the adaptive processes abstracted from natural systems. The second goal led Holland to apply GAs to combinatorial optimization problems.

The main idea behind GAs is to represent a *population* of candidate solutions, or *individuals*, as strings of bits (usually called *chromosomes*), and evolve them by means of *genetic operators*. The evolved solutions are successively evaluated against a *fitness function*, which generally represents the objective function to be optimized, and the solutions having higher fitness value are selected for the successive generation. The process is repeated until a certain number of generations is reached or a sufficiently fit solution is found.

Goldberg [70] points out four characteristics featured by GAs which make them a more robust search method than usual optimization techniques, such as hill climbing:

- GAs do not work directly on the parameters of the problems, but on a *coding* of the parameters (which is, essentially, the fixed-length bitstring representation).
- Instead of iteratively optimizing a single candidate solution, GAs evolve in parallel a population of solutions.
- GAs use only the value of the fitness function to optimize the solutions. No additional information on the underlying search space (such as derivatives) is required.
- GAs employ *probabilistic operators* to evolve their solutions, rather than *deterministic operators* such as those used by hill climbing.

The traditional genetic operators used by a GA are the following:

- *Reproduction operator*. This operator is used to select the individuals in the current population which will reproduce in the next generation. Usually, the value of the fitness function drives the selection process. In the *roulette wheel* reproduction operator, for example, individuals are selected with probability proportional to their fitness.
- *Mutation operator*. The aim of the mutation operator is to introduce random changes in the chromosome of an individual. Typically, a mutation operator flips the value of the bits in a chromosome with low probability.
- *Crossover operator*. Crossover recombines the chromosomes of two or more individuals in order to produce a new chromosome (the offspring). There are several methods to implement

crossover operators, among which one of the most used is *one-point crossover*. Given two parents p_1 and p_2 of length n, a *crossover site* $s \in \{1, \cdots, n\}$ is randomly selected. Then, a child string c_1 is produced by copying the bits of p_1 up to position s, and the bits of p_2 from position $s+1$ to n. Symmetrically, a second child string c_2 is built by swapping the order of the copying operations: from position 1 to s the bits are taken from p_2, and from position $s+1$ to n they are copied from p_1.

Generally speaking, these three genetic operators are combined by a GA using the following procedure:

1. Initialize the population with a random set of candidate solutions represented as bitstrings, and compute their fitness values.
2. Using the reproduction operator, choose a subset of individuals from the current population which will reproduce in the next generation.
3. Generate the new offspring from the parents selected in the previous step by applying mutation and crossover operators.
4. Compute the fitness values of the individuals in the offspring.
5. Create the new population by copying the individuals in the offspring. Optionally, it is possible to copy a few of the most fit individuals from the old population, by using *elitist* or *steady state* strategies. This ensures that the maximum fitness value present in the population is a non-decreasing monotone function of the number of generations.
6. Until a termination condition is met (e. g. a specific number of generations has been reached), return to point 2.

Several variations of the basic GA heuristic have been developed, ranging from more sophisticated solution encodings (based for example on permutations [191]) to variation operators preserving particular properties of the solutions, such as their balancedness [123].

6.2.2 *Genetic Programming*

Genetic Programming (GP) is an EC algorithm originally proposed by Koza [101]. The main difference with GA is that in GP the evolved individuals are not candidate solutions, but rather *computer programs* represented as *trees*. In particular, the leaf nodes of the tree represent the input of the program. For example, if the computer program corresponding to a candidate solution is a Boolean function, then the leaves of the tree are the input variables $x_1, \cdots, x_n$ of the function. The internal nodes, on the other hand, are operators (such as AND,

OR and XOR in the case of Boolean functions) combining the results coming from the children nodes. The output is finally determined by the root node.

The genetic operators of GP follow the same philosophy of those used in GA, but they are obviously adjusted to accommodate for this new representation. In particular, *subtree crossover* is used to swap two random subtrees of two individuals, while in *subtree mutation* a subtree of a solution is replaced by another random tree. Figure 17 represents an example of tree encoding of a Boolean function $f : \mathbb{F}_2^4 \to \mathbb{F}_2$ in 4 variables. In this case, the input variables x_1 and x_2 are com-

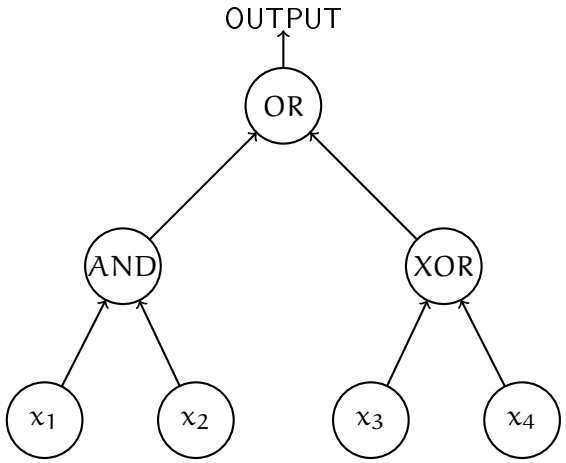


Figure 17: Example of GP tree encoding of a Boolean function.

bined by an AND operator, while x_3 and x_4 are XORed. The outputs of these two operations are finally combined in the root node by an OR operator, thus giving the following algebraic expression:

$$f(x_1, x_2, x_3, x_4) = (x_1 \text{ AND } x2) \text{ OR } (x_3 \text{ XOR } x_4) \ . \tag{83}$$

As in the case of GA, several variations of GP have been proposed in the literature, including for example *Semantic GP* [127, 186], where constraints on the semantics of the program output are enforced, and *Cartesian GP* [124], in which the solutions are represented as graphs instead of trees.

6.3 PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is a stochastic optimization heuristic originally introduced by Kennedy and Eberhart [93]. PSO basically works by representing a set of candidate solutions of an optimization problem as a *swarm of particles* which move in a coordinated manner through the search space, usually a subset of $\mathbb{R}^m$. At each time step $t \in \mathbb{N}$, the current *position* of the i-th particle $x_i^{(t)} \in \mathbb{R}^m$ is updated using the recurrence equation

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t)} \ ,$$

where $v_i^{(t)} \in \mathbb{R}^m$ is the *velocity vector* of the i-th particle at time t. The candidate solution represented by the new position is then evaluated by a fitness function, which is usually the function to be optimized, as in the case of GA. Each coordinate $j \in \{1, \cdots, m\}$ of the i-th particle velocity is in turn stochastically updated as follows:

$$v_{ij}^{(t+1)} = w \cdot v_{ij}^{(t)} + R_{ij} \cdot \varphi \cdot (g_j - x_{ij}^{(t)}) + R_{ij} \cdot \psi \cdot (b_{ij} - x_{ij}^{(t)}) \ ,$$

where the current velocity of the particle $v_{ij}^{(t)}$ is weighted by the *inertia* parameter $w \in \mathbb{R}$, the value $R_{ij} \in [0, 1]$ is a random number sampled with uniform probability, and φ and ψ are constants which respectively determine the influence of the *global best* solution $g \in \mathbb{R}^m$ found so far by the *neighborhood* of the i-th particle and the influence of the *local best* solution $b_i \in \mathbb{R}^m$ found so far by the i-th particle. In order to control the velocity of the particle, each component $v_{ij}^{(t+1)}$ is also limited in absolute value by a global parameter v_{max}. Various topologies can be used to define a neighborhood for the particles, such as the *Von Neumann* topology and the *ring* topology. In this work, however, we focus only on the *fully informed particle* paradigm [119], in which the global best solution g is simply the best solution discovered so far by the whole swarm.

The PSO heuristic has been successfully applied to several continuous optimization problems (see for example [147] for a survey). However, there are no obvious ways to apply it to discrete search spaces.

Kennedy and Eberhart proposed in [94] a variant of their original PSO algorithm in order to solve binary optimization problems. The solutions are represented as vectors of m bits, and the search space is geometrically interpreted as the m-dimensional hypercube $\mathbb{F}_2^m$. Consequently, the particles move through the vertices of this hypercube. The velocity vector becomes a *probability vector*: given the i-th particle in the swarm, for each coordinate $j \in \{1, \cdots, m\}$ the position x_i with respect to dimension j is updated by sampling a Bernoullian random variable with parameter p_{ij}. In particular, if a sampled random number $r \in [0, 1]$ is less than p_{ij} then the value of the j-th coordinate of x_i is updated to 1, otherwise it is updated to 0.

The advantage of this discrete version is that it is possible to use the same velocity equation defined for the basic PSO procedure to update the probability vectors of the particles, provided that their components are normalized on the interval $[0, 1]$. To this end, Kennedy and Eberhart adopted in [94] the *logistic function*, defined for all $x \in \mathbb{R}$ as:

$$S(x) = \frac{1}{1 + \exp(-x)} \ .$$

SURVEY OF LITERATURE

7

In this chapter, we analyze the state of the art concerning the applications of CA to cryptography, as well as the literature pertaining to the heuristic optimization of Boolean functions and S-boxes.

We begin our survey in Sections 7.1 and 7.2 by analyzing the works related to the design of stream and block ciphers based on cellular automata. We then give a brief overview in Section 7.3 of the field of *rotation-symmetric* S-boxes, which are basically periodic boundary CA under disguise where the diameter of the local rule equals the size of the cellular array. Next, in Section 7.4 we move to secret sharing schemes based on CA, remarking that the few solutions proposed in the relevant literature all feature a *sequential threshold* access structure, where the shares must satisfy an adjacency constraint. Finally, we conclude in Section 7.5 by describing the research line devoted to the optimization of Boolean functions and S-boxes through heuristic techniques.

7.1 CA-BASED PRNGS AND STREAM CIPHERS

Most applications of CA to cryptography published in the literature deal with the generation of *pseudorandom sequences* for *stream ciphers*. The first work in this line of research dates back to Wolfram [195], who proposed to exploit the chaotic behaviour of a ECA equipped with rule 30 to build a pseudorandom number generator (PRNG). The idea was to represent the random seed of the PRNG as the initial configuration of the cellular array and iterate the CA with periodic boundary conditions using rule 30 for a certain number of steps. Then, the trace of the central cell of the CA was sampled as a keystream for a Vernam-like cipher. Figure 18 displays a diagram for the overall cipher proposed by Wolfram. The initial configuration of the CA which acts as the PRNG seed is framed in blue, while the sampled trace that constitutes the keystream in red. Wolfram extensively analyzed the dynamic evolution of this rule over several initial configurations, using statistical tests and methods from dynamical systems theory, such as the computation of *Lyapunov's exponent* which is a measure of the stability of a dynamical system. The obtained results seemed to indicate that this rule could produce good pseudorandom sequences, potentially useful for the implementation of a stream cipher. From a practical point of view, Wolfram recommended to use a CA with at least $n = 127$ cells, using various sampling strategies

in order to destroy the correlations of the bits in the sequence (for example, sampling in alternating time steps).

Unfortunately, Wolfram's PRNG turned out to be very weak from a cryptographic standpoint: Meier and Staffelbach [118] proved that it is vulnerable to a known plaintext attack, unless the CA is initialized with a seed of at least 1000 bits. The attack exploits the right permutivity of rule 30, which effectively halves the keyspace of initial configuration from 2^n to $2^{\frac{n}{2}}$ possible seeds. Later, Koc and Apohan [100] showed another attack on Wolfram's PRNG based on a linear approximation of rule 30.

These two attacks are a consequence of the fact that rule 30 does not have good cryptographic properties. In particular, this rule has nonlinearity $N_f = 2$, which makes it vulnerable to linear approximation like in Koc and Apohan's attack. Moreover, rule 30 is not 1–resilient, which is the reason why Meier and Staffelbach's attack proved to be so efficient: as a matter of fact, rule 30 features some statistical correlations allowing one to reduce even further the keyspace, which is already been halved to $2^{\frac{n}{2}}$ by the right permutivity of the rule.

More generally, Martin [114] showed by an exhaustive search that, among the 256 elementary rules, only the 8 *linear* rules are 1–resilient. This fact can be interpreted as a corollary of Tarannikov's bound, stated in Theorem 14: if the local rule is defined over $d = 3$ variables, then the maximum nonlinearity for 1–resilient Boolean functions is $2^{3-1} - 2^{1+1} = 0$.

The consequence is that elementary CA rules are not adequate for building a cryptographic PRNG or a stream cipher, so it is necessary to explore the spaces of rules having higher radii. This research line has been pursued by Leporati and Mariot in [106], where the authors showed that bipermutive local rules are 1–resilient. Since the space of bipermutive rules of d variables is $2^{2^{d-2}}$, the authors performed an exhaustive search of all bipermutive rules of radius $r = 2, 3$ with the goal of finding those having the highest nonlinearity values and resiliency orders. Formenti et al. [63] also performed a similar exhaustive search over the set of local rules of radius $r = 2$, leveraging on the classification of Boolean functions of 5 variables up to affine equivalence carried out by Berlekamp and Welch in [13].

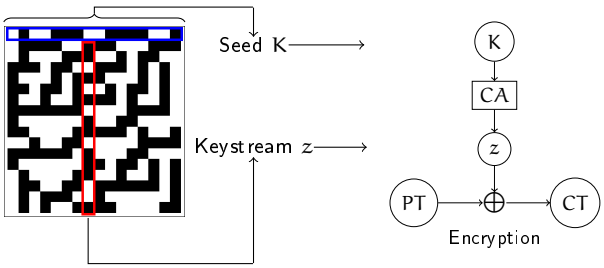


Figure 18: Block scheme of Wolfram's stream cipher based on Rule 30.

7.2 CA-BASED BLOCK CIPHERS

The first block cipher based on cellular automata was proposed by Gutowitz [74]; it was designed using both irreversible and reversible CA. In particular, for the irreversible part, the author employed permutive local rules. Given a configuration $c \in \mathbb{F}_2^n$, permutive rules allow one to determine a preimage $p \in \mathbb{F}_2^{n+d-1}$ by constructing a path over the de Bruijn graph, as discussed in Section 2.3. Gutowitz proposed to iterate this preimage computation process for the diffusion phase of the cipher. Additionally, reversible *block CA* were used in the substitution phase to ensure the invertibility of the resulting S-box. In a block CA, the local rule does not determine the next state of a single cell, but rather the state of a block of multiple cells. At the beginning, the cellular array is divided in blocks of m cells each, and then a permutation $f : \mathbb{F}_2^m \to \mathbb{F}_2^m$ is applied to each block in parallel. Next, the local rule is applied by shifting the blocks one place to the right using periodic boundary conditions.

A second type of reversible CA used for block ciphers are *second-order CA*. Given a local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$, the next state at time $t+1$ of the i-th cell $c_i(t+1)$ in a second-order CA is determined as follows:

$$c_i(t+1) = c_i(t-1) \oplus f(c_i(t), \cdots, \cdots, c_{i+d-1}(t)) \ . \tag{84}$$

As a consequence, in order to compute the next state of a second-order CA, one has to know both the value of the current configuration and its value at the previous time step. Seredynsky et al. [162] investigated second-order CA as S-boxes, studying the *avalanche properties* of CA of length $n = 32$ and $n = 64$ equipped with local rules of radius $r = 2$ and $r = 3$.

A third approach for the construction of CA-based block ciphers was set forth by Szaban et al. in [176]. Instead of considering classes of CA already known to be reversible, the authors first considered the set of all 256 local rules of radius $r = 1$ and selected those which had the best nonlinearity and autocorrelation values by evolving the CA for a certain number of steps over a cellular array of size $n = 8$. This selection resulted in six local rules, which were subsequently checked for bijectivity over longer array lengths.

Notice that the solutions described up to now always focus on the *iterated behaviour* of the CA. Another perspective is to consider the S-box corresponding only to one CA iteration (that is, the CA global rule). This approach has been mainly investigated by Daemen et al. [52, 51], focusing in particular on the class of *complementing landscapes cellular automata* (CLCA). A complementing landscape is a string s over the alphabet $\{0, 1, *\}$, where the symbol $*$ indicates a *don't care* that can be either 0 or 1. Thus, each landscape defines a set of possible binary strings. Given a collection of complementing landscapes S_{cl}, each of length $d-1$, the local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$ induced by

S_{cl} flips the state of the i-th cell if the string $(x_{i-\omega}, \cdots, x_{i-\omega+\delta-1})$ appears in one of the landscapes of S_{cl}, and leaves it unchanged otherwise. In [52], CLCA for block cipher design were studied distinguishing between *locally* and *globally* invertible rules. In particular, all locally invertible CLCA turn out to be *involutions*, while globally invertible CLCA are invertible only over certain sets of periodic configurations. This implies that a globally invertible CLCA is not reversible in general, and the inverse global rule is typically specified using a sequential algorithm. In this case, one can prove that a specific CLCA F is globally invertible over $\Sigma_n^{\mathbb{Z}}$ by first providing a *seed* for every possible periodic configuration $y \in \Sigma_n^{\mathbb{Z}}$, i. e. a value $x \in \Sigma$ such that there exists $i \in \{0, \cdots, n-1\}$ with $F^{-1}(y)_i = x$. The second step consists of providing a *leap*: starting from the seed x at position i, the value of the preimage $F^{-1}(y)$ at position $i-k$ can be determined, for a certain $k \in \mathbb{N}$. Next, the value at position $i-k$ becomes a new seed, and the leap procedure is repeated until either the preimage of y is completed or no additional bits can be determined, in which case one has to provide a new seed for the remaining positions. The size k of the leap depends on the diameter of the local rule, and thus the number of independent seeds required for reconstructing a preimage depends on the period n of the configurations. As an example, consider the case where $k = 2$ and n is odd: then, a single seed in any position i is sufficient to determine the whole preimage of a periodic configuration $y \in \mathcal{F}_n$ under a globally invertible CLCA. In particular, the CLCA χ introduced by Daemen et al. [51] with diameter $d = 3$ and offset $\omega = 0$ defined by the single landscape $l = 01$ is globally invertible over all periodic configurations of odd length. The CLCA χ also has a simple description in terms of correlation and propagation characteristics, which makes it interesting for cryptographic purposes. Indeed, this CA is the only nonlinear component used in Keccak [15], where it is applied to a periodic array of length $n = 5$. In this case, the resulting S-box has nonlinearity and differential uniformity both equal to 8. Earlier hash functions such as Panama [50] and RadioGatún [14] also employed the one's complement of χ, called γ.

7.3 ROTATION-SYMMETRIC S-BOXES

Rotation symmetry has been extensively studied in the case of Boolean functions, in particular with respect to the nonlinearity property. To the best of our knowledge, the first who generalized this concept to S-boxes were Rijmen et al. [155]. Formally, A (n, n)-function F is *rotation-symmetric* if there exists a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ such that, for all $x \in \mathbb{F}_2^n$,

$$F(x) = (f(x), f(\sigma_c(x)), f(\sigma_c^2(x)), \cdots, f(\sigma_c^{n-1}(x))) \ , \tag{85}$$

where $\sigma_c^i(x)$ denotes the cyclic shift σ_c applied i times to vector x. Thus, rotation-symmetric S-boxes have a simple algebraic description since they are defined by a single coordinate function f, and this feature also yields an advantage at the implementation level.

Given the definition above, it is not difficult to see that a S-box F is rotation-symmetric if and only if it commutes with the cyclic shift, i.e. $F(\sigma_c(x)) = \sigma_c(F(x))$ for all $x \in \mathbb{F}_2^n$. From Hedlund's theorem, stated in Section 2.1, it follows that rotation-symmetric S-boxes are actually finite CA with periodic boundary conditions. In particular, the local rule of the CA is the coordinate function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ of the S-box, meaning that the diameter d equals the length n of the CA. Additionally, from Equation (85) one can see that the offset of the CA is $\omega = 0$, hence each cell looks at its state and the states of its $n-1$ right neighbors with periodic boundary conditions to compute its next state.

Rijmen et al. [155] showed that bijective S-boxes generated using power maps or exponentiations over finite fields are linearly equivalent to rotation-symmetric S-boxes. This result is particularly interesting, since, as remarked by the authors of [155], almost all S-boxes used in practical cryptographic applications are generated through power maps and exponentiations.

Kavut [90] enumerated all 6×6 bijective rotation-symmetric S-boxes with maximum nonlinearity 24, showing that up to affine equivalence there are only 4 functions with differential uniformity 4 and algebraic degree 5. Finally, more recently, Liu et al. [108] analyzed the iterated behaviour of (n, n) functions and provided a construction of rotation-symmetric S-boxes satisfying *perfect diffusion*.

7.4 CA-BASED SECRET SHARING SCHEMES

During the last few years some SSS based on *cellular automata* (CA) have been proposed in the literature, the first of which can be traced back to del Rey, Mateus and Sánchez [153]. Specifically, this scheme exploits the reversibility of higher-order CA, which we already encountered in Section 7.2. The secret is represented as one of the k initial conditions in a k-th order CA which is then evolved for n iterations. Each player then receives one of the n resulting CA configurations as a share. The access structure generated by this scheme can be defined as a (k, n) *sequential threshold*, since at least k *consecutive* shares are required in order to evolve backwards the LMCA and recover the secret, meaning that there are in total $n - k + 1$ minimal authorized subsets. Most of the later CA-based SSS [111, 60, 61] use the same LMCA principle of del Rey, Mateus and Sánchez's scheme, and thus feature similar access structures.

More recently, a CA-based SSS designed on a different principle has been proposed by Mariot and Leporati in [112]. In particular, this

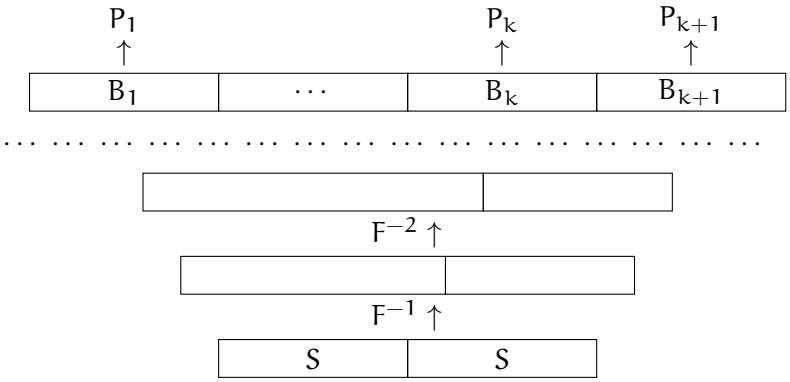


Figure 19: Setup phase of the extended scheme proposed in [112].

SSS is based on the preimage construction algorithm of bipermutive CA used by Gutowitz [74] for the design of his CA block cipher mentioned in Section 7.2. The secret S is represented as a finite configuration of length m of a NBCA equipped with a bipermutive local rule. The dealer runs the preimage construction algorithm until a preimage of length $k \cdot m$ is obtained. This preimage is then split in k blocks of size m each, which the dealer successively distributes to the players. In order to recover the secret, the players must combine in the correct order their blocks to reconstruct the preimage, and then evolve the CA forward until they obtain the secret. A simple modification allows to extend this (k, k) threshold scheme to $k+1$ players by appending a copy of the secret to its right, in order to obtain a final preimage of $k+1$ blocks of size m in which the two sets of players $P_1, \cdots, P_k$ and $P_2, \cdots, P_{k+1}$ can recover the secret using the same procedure. As a matter of fact, by combining the respective shares the dynamic evolution of the resulting pieces of preimages collapse on one of the two copies of the secret. Figure 19 displays the setup phase of this extended scheme. Clearly, this procedure can be generalized to get a (k, n)-threshold scheme by concatenating k copies of the secret. Hence, this scheme features a sequential threshold access structure as well, since all minimal authorized subsets are of the form $\{P_i, \cdots, P_{i+k-1}\}$. The difference with the approach set forth in [153] is that in the latter the shares must satisfy a *temporal adjacency* constraint (the shares being successive configurations of a LMCA), while in this scheme the shares must be *spatially adjacent*, since they are block of a NBCA preimage. Further, by Lemma 2 we know that the preimages of spatially periodic configurations in surjective CA are spatially periodic as well. This means that at a certain point the shares in the scheme proposed in [112] will begin to repeat themselves, thus yielding a *cyclic threshold* access structure. More precisely, recall from Section 2.3 that one can construct a preimage of a configuration $y \in \Sigma^{\mathbb{Z}}$ under a surjective CA F by building a path on the vertices of its de Bruijn graph. This is done by reading the corresponding symbols of y over the edges labels. In particular, the preimage is determined by

fusing all vertices labels which one visits throughout the path. Since the configuration whose we are computing a preimage is spatially periodic, the labels read on the edges will begin to repeat themselves at a certain point. Hence, if the vertex label equals the one where we started our path at that repetition point of y, the preimage repeats itself as well. This happens after at most q^{d-1} repetitions of the block defining the SPC y, since there are q^{d-1} vertices in the de Bruijn graph of F. Figure 20 exemplifies this preimage computation process in a bipermutive CA. As a consequence, determining the

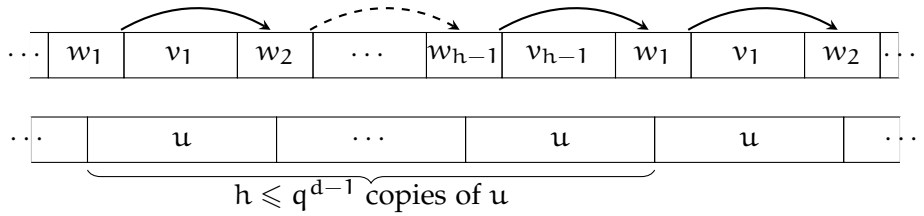


Figure 20: Example of preimage construction of an SPC $y = {}^{\omega}u^{\omega}$. The period of the sequence of blocks w_i is at most q^{d-1}

maximum number of players in the SSS proposed in [112] depends on the characterization of the periods of preimages of spatially periodic configurations in bipermutive CA.

7.5 HEURISTIC OPTIMIZATION OF BOOLEAN FUNCTIONS AND S-BOXES

The body of literature devoted to the optimization of the cryptographic properties of Boolean functions and S-boxes is rather large. In this section, we only mention the works which are relevant for the subsequent chapters of this thesis. For further information on the subject, the reader is referred to [137].

The first attempt at designing a GA to optimize the cryptographic properties of Boolean functions dates back to Millan, Clark and Dawson [122]. There, the authors used the classic bitstring representation of GA to encode the truth table of Boolean functions, with the goal of maximizing their nonlinearity. Later, the same authors presented an improvement in [123] which confined the GA to balanced Boolean functions, combining it also with an additional hill climbing optimization step. The fitness function, in this case, took into account also the resiliency order and the propagation criterion of the evolved function, besides their nonlinearity.

Clark and Jacob [44] proposed a two-stage method for optimizing the cryptographic properties of Boolean functions, using simulated annealing in the first stage and hill climbing in the second one. Successively, the same authors together with Maitra and Stanica set forth a different approach to this optimization problem in [42]. In partic-

ular, they proposed to use a *spectral inversion method*, where the candidate solutions are not the truth tables of Boolean functions, but rather Walsh spectra already satisfying certain cryptographic properties. Since applying the inverse Walsh transform to any Walsh spectrum does not yield a Boolean function in general, the objective function of the optimization problem becomes minimizing the deviation of the resulting *pseudo-Boolean function* $f : \mathbb{F}_2^n \to \mathbb{R}$ from being a true Boolean function. The authors of [42] thus applied a simulated annealing algorithm on this new problem representation.

Picek, Jakobovic and Golub [142] were the first to apply GP to the evolution of Boolean functions with good cryptographic properties, comparing its performance with GA. Successively, Picek, Marchiori, Batina and Jakobovic [144] experimented with several EC algorithms by combining them with algebraic constructions, in order to investigate the maximum nonlinearity achievable by balanced Boolean functions of 8 variables. In fact, for these functions it is still an open problem to determine whether the upper bound on nonlinearity is 118 or 116. Hrbacek and Dvorak [81] applied Cartesian GP to evolve bent functions up to 16 variables. On the other hand, the cryptographic property of correlation immunity has been investigated by Picek et al. in [140, 138]. More recently Picek et al. [143] compared the performance of four evolutionary algorithms (namely GA, GP, Cartesian GP and evolutionary strategy) under three different fitness functions which take into account several cryptographic properties.

The first work adopting EC algorithms for optimizing the cryptographic properties of S-boxes dates back to Millan et al. [121]. There, the authors designed a GA to evolve S-boxes with high nonlinearity and low autocorrelation. Burnett et al. [26] used an heuristic method to generate S-boxes with the same structure as those featured in the MARS block cipher, one of the AES finalists [28].

Fuller, Millan and Dawson [65] proposed a multi-objective optimization approach for the heuristic construction of cryptographically strong S-boxes, showing that power functions can be evolved to obtain solutions with the best possible trade-off between nonlinearity and autocorrelation.

Picek et al. [146] applied GP and Cartesian GP to the evolution of S-boxes, devising a method to adapt these two heuristic to the permutation encoding. Finally, Picek et al. proposed a new cost function in [139] to evolve S-boxes with higher nonlinearity values.

Part II

HEURISTIC OPTIMIZATION OF BOOLEAN FUNCTIONS

HEURISTIC SEARCH OF BOOLEAN FUNCTIONS BY DISCRETE PARTICLE SWARM OPTIMIZATION

8

In Chapter 7 we saw that Wolfram's pseudorandom generator for stream ciphers is vulnerable to two cryptanalytic attacks, due to the fact that rule 30 is not 1–resilient and does not have a high nonlinearity. Moreover, we observed by Tarannikov's bound that none of the $2^{2^3} = 256$ local rules of diameter $d = 3$ are 1–resilient and nonlinear at the same time, thus limiting the usefulness of elementary CA in the design of PRNGs for cryptographic purposes.

On the other hand, we also mentioned that the space of binary local rules of diameter d, or equivalently the set of Boolean functions of d variables, is composed of 2^{2^d} elements, which makes exhaustive search unfeasible for any $d > 5$. The search of Boolean functions with good cryptographic properties is thus a combinatorial optimization problem which is interesting both for CA-based cryptography and for the field of cryptographic Boolean functions in general.

In order to cope with the combinatorial explosion resulting by increasing the number of variables, one can rely on two different approaches. The first is the use of *algebraic constructions*, which allows one to determine infinite classes of Boolean functions with good cryptographic properties from scratch (*primary constructions*) or to obtain new functions by modifying known ones (*secondary constructions*) [30]. On the other hand, the second approach is based on the use of *heuristic techniques*, about which we gave an overview in Chapter 7. In this chapter and the next one, we focus on the latter approach.

In particular, the aim of this chapter is to investigate the application of Particle Swarm Optimization (PSO) to search balanced Boolean functions with good cryptographic properties. As far as we know, this is the first time that PSO is applied to this optimization problem. More precisely, Saber et al. showed the existence of Boolean functions of 9 variables 3–resilient functions with nonlinearity 240 and degree 5 using a modified version of the PSO algorithm [159]. However, the details of the modified algorithm are not available in that correspondence, and moreover the authors state that their PSO variant is based on Clark et al.'s spectral inversion method mentioned in Section 7.5 [42]. On the other hand, in this chapter we apply PSO to the classic truth table representation of Boolean functions, an approach which, to the best of our knowledge, has never been attempted before in the relevant literature.

To this end, we propose a modified version of the discrete PSO algorithm for permutation problems designed by Hu, Eberhart and

Shi [82] by adapting it to the case of $\binom{2^n}{2^{n-1}}$ *combinations*, thus limiting the search space to *balanced* Boolean functions. This modification is implemented by a new update method for the positions of the particles, which preserves the Hamming weights of the truth tables. To further improve the nonlinearity and the deviation from correlation immunity of the candidate solutions, the modified PSO algorithm is also integrated with the Hill Climbing procedure by Millan, Clark and Dawson [123]. We address the problem of finding optimal values for the social and cognitive constants, inertia and maximum velocity parameters in the PSO velocity equation by using two meta-optimization techniques: Local Unimodal Sampling (LUS) and Continuous Genetic Algorithms (CGA, also known as Real-Coded GA). While the former has already been adopted in the literature to tune the parameters of PSO [134], to our knowledge CGA have never been applied to this meta-optimization task. We compare the performances of LUS and CGA in tuning the PSO parameters for the case of Boolean functions of $n = 7$ variables with three underlying fitness functions (each targeting a different set of cryptographic properties), and observe that CGA achieve better results. We finally employ the parameters optimized through CGA to run the PSO algorithm on the spaces of Boolean functions of up to $n = 12$ variables. The results of the experiments are reported and compared with those achieved by other heuristic methods published in the literature, focusing only on the best solutions found. We observe that our PSO algorithm is able to find Boolean functions with similar or better combinations of nonlinearity, correlation immunity and propagation criterion than the ones produced by other methods, especially when the number of variables is less than 10. It is also found that the properties (especially the nonlinearity) get worse as the number of variables increases, suggesting that further parameters tuning is required in this case.

The remainder of this chapter is structured as follows. Section 8.1 describes the modified PSO algorithm, focusing on the update method for the particle positions which preserves their Hamming weights, and defines the fitness functions employed in the experiments. Section 8.2 deals with the parameter tuning problem, briefly introducing the two meta-optimization techniques used (LUS and CGA) and then reporting their results. Section 8.3 describes the experiments performed on the PSO algorithm with the CGA-evolved parameters and reports the results, comparing them with those obtained by other optimization methods. Finally, Section 8.4 summarizes the results presented in this chapter.

8.1 PSO ALGORITHM DESCRIPTION

8.1.1 *Position Update for Balanced Functions*

Using the truth table representation, the discrete PSO heuristic described by Kennedy and Eberhart [94] mentioned in Section 6.3 can be straightforwardly applied to the optimization problem of finding good cryptographic properties of Boolean functions of n variables. In this case, the particles would move in the space of $m = 2^n$ binary vectors. However, the method proposed in [94] to update the positions of the particles does not give any control over their Hamming weights, since each component in the probability vector is sampled independently from the others. Hence, there are no guarantees that the generated truth tables will be balanced, a fundamental property for cryptographic Boolean functions. A possible solution to this drawback is to add an *unbalancedness* penalty in the fitness function, an approach which has been followed in [142] for Genetic Algorithms and Genetic Programming. Our preliminary experiments however showed that this method is not satisfactory with PSO, since the proportion of generated balanced functions is really low. Thus, it is necessary to use an update operator which limits the search space to the set of balanced Boolean functions.

Hu, Eberhart and Shi [82] adapted the discrete PSO algorithm in order to apply it to *permutation problems*. Their update method works by stochastically *swapping* the values in the permutation vector which represents the position. In particular, the component x_{ij} of the i-th particle is changed with probability p_{ij} by swapping it with x_{ik}, where k is such that $x_{ik} = g_j$. As a consequence, the permutation represented by vector x_i is adjusted by making it more similar to the global best solution g.

From a combinatorial point of view, the set of balanced Boolean functions of n variables is isomorphic to the set of $\binom{2^n}{2^{n-1}}$ combinations. In fact, a subset of 2^{n-1} out of 2^n objects can be represented by its *characteristic function*, which is basically a balanced binary vector $x \in \mathbb{F}_2^m$, where $m = 2^n$. Starting from this observation, we generalised the update operator proposed by Hu, Eberhart and Shi to the case of balanced combinations. Given the balanced binary vector $x_i \in \mathbb{F}_2^m$ and the corresponding probability vector $p_i \in [0,1]^m$, for each coordinate $j \in \{1, \cdots, m\}$ a random number $r \in [0,1]$ is sampled with uniform probability, and if r is less than p_{ij} then a swap is performed as follows. First, the value of x_{ij} is compared with that of the global best in the same index, g_j. If the two values are equal, then no action is taken. Otherwise, the bit in x_{ij} is swapped with another bit x_{ik}, where $k \neq j$ is such that $x_{ik} \neq g_k$ and $x_{ik} \neq x_{ij}$. These two conditions ensure that, while the Hamming weight of the vector is preserved, its Hamming distance from the global best solution is de-

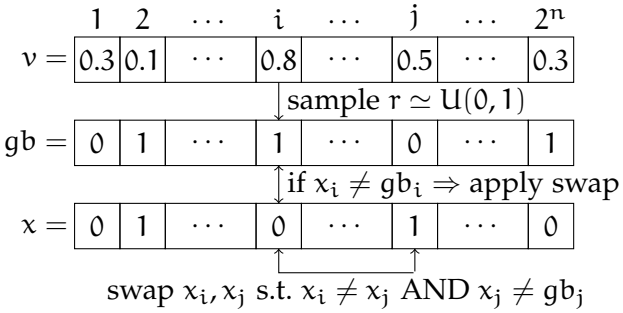


Figure 21: Example of application of swap-based position update.

creased by 2. In fact, if only $x_{ik} \neq g_k$ is verified, swapping the values of x_{ij} and x_{ik} yields the same Hamming distance between x_i and g. Since there can be more than one index k which satisfies these two conditions, our update operator randomly selects one of them. Figure 21 depicts an example of application of our update operator over a binary string.

The whole update process is then repeated using the local best b_i of the i-th particle instead of the global best g. In this way, x_i is changed by considering both the social attraction of the whole swarm and the cognitive attraction of the particle. Moreover, if the current position of the particle is equal to g, a random pair of bits in x_i is swapped in order to avoid premature convergence, a solution similar to the one proposed in [82].

Algorithm 1 reports the general pseudocode implementing our position update operator. The input parameters x_i and y are balanced binary vectors which respectively represent the position of the i-th particle in the swarm and either the position of the global best g or local best b_i. We assume that $x_i \neq y$. Vector p_i is the probability vector associated to the i-th particle and $m = 2^n$ is the length of x_i. The procedure RAND-UNIF() samples a random number $r \in [0, 1]$ with uniform probability, which is used to determine whether a swap is required by comparing it to p_{ij}. The subroutine FIND-CAND-SWAP(), whose details are omitted, performs the search of a suitable index for the swap, returning 0 if it cannot find one.

Algorithm 1 UPDATE-BAL-POS(x_i, y, p_i, m)

```
for j := 1 to m do
  r := RAND-UNIF()
  if (r < p_ij AND x_ij ≠ y_j) then
    k := FIND-CAND-SWAP(x_i, j)
    if (k ≠ 0) then
      Swap x_ij with x_ik
    end if
  end if
end for
```

8.1.2 *Fitness Functions*

We tested our Particle Swarm Optimizer with three fitness functions, all of which have to be maximized. The three cryptographic properties which we considered for optimization are nonlinearity, correlation immunity and Strict Avalanche Criterion (SAC, i. e. the propagation criterion $\mathrm{PC}(1)$).

While we already have a numerical formula to compute the nonlinearity of a Boolean function (see Equation (58) in Section 5.2), for correlation immunity and propagation criteria orders we only gave binary definitions (i. e. either a function is t–th order correlation immune or $\mathrm{PC}(l)$, or it is not). For these reason, we adopted two *deviation measures* originally introduced in [123] in order to better drive the optimization process of PSO through the fitness function. We formalize these measures in the following definition.

Definition 30. *The* deviation from t-th order correlation immunity and propagation criterion $\mathrm{PC}(l)$ *of a Boolean function* $f : \mathbb{F}_2^n \to \mathbb{F}_2$ *are respectively defined as*

$$\mathrm{cidev}_t(f) = \max\{|W_f(\omega)| : \omega \in \mathbb{F}_2^n,\ 1 \leqslant w_H(\omega) \leqslant t\} \tag{86}$$

$$\mathrm{pcdev}_l(f) = \max\{|\hat{r}(s)| : s \in \mathbb{F}_2^n,\ 1 \leqslant w_h(s) \leqslant l\}\ . \tag{87}$$

In other words, $\mathrm{cidev}_t(f)$ and $\mathrm{pcdev}_l(f)$ corresponds to the maximum absolute values of the Walsh transform and the autocorrelation function restricted only to the vectors respectively having Hamming weight at most t and l.

The first fitness function fit_1 considers the three properties of nonlinearity, deviation from first order correlation immunity and deviation from the SAC:

$$\mathrm{fit}_1(f) = \mathrm{Nl}(f) - \frac{\mathrm{cidev}_1(f)}{4} - \frac{\mathrm{pcdev}_1(f)}{8}\ .$$

Since the values of the Walsh and autocorrelation spectra of a balanced Boolean function are respectively multiples of 4 and 8, the two deviations in fit_1 are normalized by these two factors. This fitness function closely resembles those defined in [123] for Genetic Algorithms, where it is proposed either to minimize the *normalized deviation* of the Boolean function, defined as the maximum value between $\mathrm{cidev}_t(f)/4$ and $\mathrm{pcdev}_l(f)/8$, or to maximize the difference between nonlinearity and $\mathrm{cidev}_t(f)$. We adopted the latter as our second fitness function, with $t = 2$:

$$\mathrm{fit}_2(f) = \mathrm{Nl}(f) - \mathrm{cidev}_2(f)\ .$$

Finally, our third fitness function targets the nonlinearity and the absolute indicator of Boolean functions, two criteria which have been

optimized together by several heuristic methods proposed in the literature [43, 2, 142]:

$$\mathrm{fit}_3(f) = \mathrm{Nl}(f) - \mathrm{AC}_{max}(f) \ .$$

It can be observed that none of the above fitness functions takes into account the algebraic degree, as opposed for example to the ones employed in [142]. The motivation for this choice is twofold. First, algebraic degree is a property which is easier to optimize than nonlinearity or correlation immunity, the reason being that as $n \to \infty$, the algebraic degree of a random Boolean function of n variables is almost surely $n-1$ [30]. Hence, heuristic methods using algebraic degree in their fitness functions are likely to find Boolean functions having maximum degree but which are not $\mathrm{CI}(t)$ or $\mathrm{PC}(l)$. Second, as we show in Section 8.3.2, our PSO algorithm is able to discover Boolean functions reaching Siegenthaler's bound, despite the fact that our fitness functions do not consider the algebraic degree.

8.1.3 *Overall PSO Algorithm*

To further improve the performance of our Particle Swarm Optimizer, we combined it with the Hill Climbing (HC) algorithm designed by Millan, Clark and Dawson [123]. This technique works by swapping a pair of bits in the truth table of a balanced Boolean function in order to increase its nonlinearity and decrease its deviation from $\mathrm{CI}(k)$. In what follows, we denote by NL-CI(κ)-HC the HC procedure which increases nonlinearity while decreasing $\mathrm{cidev}_k(f)$, while NL-HC stands for the HC targeted only at increasing nonlinearity. The reader is referred to [123] for further details about the general HC method.

The type of Hill Climbing performed by our PSO algorithm depends on the underlying fitness function: in the case of fit_1 and fit_2 respectively NL-CI(1)-HC and NL-CI(2)-HC are applied, while for fit_3 NL-HC is used.

We now summarise the overall procedure of our discrete Particle Swarm Optimizer:

1. Given a swarm of size N, for all $i \in \{1, \cdots, N\}$ initialize the i-th particle by randomly creating a balanced binary vector $x_i \in \mathbb{F}_2^m$ and a probability vector $p_i \in [0,1]^m$, where $m = 2^n$ and n is the number of variables of the Boolean functions.

2. Given $k \in \{1,2,3\}$, for all $i \in N$ compute the fitness value fit_k of solution x_i found by particle i.

3. Update the global best solution g and the local best solutions b_i, for all $i \in \{1, \cdots, N\}$.

4. For all $i \in \{1, \cdots, N\}$, update the probability vector v_i using the PSO velocity recurrence, and then normalize each coordinate through the logistic function.

5. For all $i \in \{1, \cdots, N\}$, update the position vector x_i. If $x_i = g$ or $x_i = b_i$, swap a random pair of bits in x_i, otherwise invoke UPDATE-BAL-POS(x_i, g, p_i, m) and then apply procedure UPDATE-BAL-POS(x_i, b_i, p_i, m).

6. Depending on the fitness function, apply to all particles the hill climbing optimization step NL-CI(K)-HC or NL-HC described in [123].

7. If the maximum number of iterations has been reached, output the global best solution g, otherwise return to step 2.

8.2 PARAMETERS TUNING

8.2.1 *Problem Statement*

It has been widely shown in the literature that the choice of the velocity parameters greatly influences the performance of PSO [165, 182]. Instead of searching by trial-and-error a good combination of parameters for our PSO algorithm, we tackled the problem in a systematic way using a *meta-optimization* approach.

The main idea behind meta-optimization is to consider the selection of the parameters governing an optimizer O as an optimization problem itself. An *overlaying* meta-optimizer M is then applied to explore the parameters space, using a meta-fitness function to assess the performance of O under a given combination of parameters.

A candidate solution for the meta-optimization problem of our discrete PSO is thus a vector $(w, \varphi, \psi, v_{max}) \in \mathbb{R}^4$ which specifies the four parameters to be used in the velocity equation. Considering the observations reported in [94], we chose to limit the value of each parameter in the interval $[0, 10]$. For the choice of the overlaying meta-optimizer, we decided to test *Local Unimodal Sampling* (LUS) and *Continuous Genetic Algorithms* (CGA).

8.2.2 *Local Unimodal Sampling*

LUS is a local search technique which iteratively improves the current solution x by sampling with uniform probability a point y in its neighborhood $N(x)$. Considering a maximization problem, if the fitness value of y is higher than that of x, the current solution is set to y. Otherwise, the size of $N(x)$ is decreased by a discount factor β, the rationale being that by sampling with a constant-size neighborhood the algorithm is not guaranteed to converge to a local optimum. The sampling process is then repeated until a termination criterion is met, which is usually a minimum threshold τ for the size of the neighborhood. Pedersen and Chipperfield [134] employed LUS to tune the

velocity parameters of a Particle Swarm Optimizer aimed at training the weights of artificial neural networks.

8.2.3 *Continuous Genetic Algorithms*

CGA are a generalization of Genetic Algorithms to continuous optimization problems, which represent the chromosome of a candidate solution using a vector of real numbers in place of a binary string. To our knowledge, CGA have never been applied to tune PSO parameters. In the context of our meta-optimization problem, we employed the *flat operator* introduced by Radcliffe [151] as the crossover method of our CGA, while for the mutation procedure we relied on the simple *random operator* proposed by Michalewicz [120]. The reproduction operator is implemented using the *roulette wheel method*, which stochastically selects an individual with a probability proportional to its fitness. Specifically, given a population of P chromosomes, the next generation is created as follows. Using the roulette wheel method, $P/2$ pairs of chromosomes are formed, and for each pair (x, y) an offspring of two chromosomes (c_1, c_2) is created by applying with probability p_c the flat crossover operator (if it is not applied, the chromosome pair (x, y) is simply reproduced unaltered). The random mutation operator is then employed with probability p_m to each locus of the chromosomes in the offspring. We also used an *elitist strategy* to ensure that the best individual is preserved in the next generation.

8.2.4 *Meta-Fitness Function*

The *meta-fitness function*, used to drive the search for a good combination of PSO parameters, is clearly the most intensive step from a computational point of view. In fact, given a vector $x \in \mathbb{R}^4$ several runs of our discrete PSO algorithm must be performed, in order to have a statistically significant measure of its performance under the parameters specified by x. In particular, we chose to test the case of balanced Boolean functions defined on $n = 7$ variables, using a swarm of $N = 50$ particles evolved for $I = 100$ iterations. The PSO algorithm is executed for $R = 30$ independent runs, and at each run the fitness of the global best solution g at the last iteration is recorded. According to Pedersen and Chipperfield [134], the average fitness value μ_g of the global best over all R optimization runs should be used as the meta-fitness function. Since from a cryptographic point of view we are interested in Boolean functions satisfying the best possible properties, we also considered the maximum fitness value max_g achieved by the global best over R runs. It is known that for Boolean functions of 7 variables the maximum value of nonlinearity is $Nl_{max} = 56$ [132], which corresponds to the quadratic bound mentioned in Section 5.2.3.

Hence, the maximum value achievable by max_g is 56 with respect to fitness functions fit_1 and fit_2. No function of 7 variables having absolute indicator $AC_{max} < 16$ has ever been reported in the literature, and it has been conjectured that 16 is the best lower bound [199]. As a consequence, under the current state of knowledge the maximum value reachable by max_g with respect to fit_3 is $56 - 16 = 40$.

Given a parameter vector $x \in \mathbb{R}^4$, our meta-fitness function can thus be defined as

$$mfit_k(x) = \mu_g + max_g \ ,$$

where $k \in \{1, 2, 3\}$ indicates the fitness function fit_k which is being optimized by the PSO algorithm.

8.2.5 *Meta-Optimization Results*

Following the methodology described in [134], for each underlying fitness function fit_k we performed $M = 6$ runs to assess the performances of both LUS and CGA, thus carrying out a total of 36 meta-optimization experiments. In the case of LUS we adopted the value $\beta = 0.33$ for the discount factor and $\tau = 0.001$ for the minimum threshold of the neighborhood size. On the other hand, for the CGA meta-optimizer we used a population of $P = 20$ individuals evolved for $G = 100$ generations, setting the crossover and mutation probability respectively to $p_c = 0.95$ and $p_m = 0.05$.

Table 1 compares the best parameters combinations found by LUS and CGA over the 6 meta-optimization runs for each fitness function.

Table 1: Comparison of Best PSO Parameters

fit_k	Method	μ_g	max_g	$mfit_k(f)$
fit_1	LUS	52.7	56	108.7
	CGA	53	56	109
fit_2	LUS	46	52	98
	CGA	46.27	56	102.27
fit_3	LUS	30.87	40	70.86
	CGA	38.4	40	78.4

It can be observed that CGA outperforms LUS with respect to all three fitness functions. While in the case of fit_1 there is only a slight difference concerning the average fitness values μ_g, for fit_2 the best parameter combination found by LUS did not allow the Particle Swarm Optimizer to reach the maximum fitness value of 56, whereas for fit_3 the mean fitness μ_g of LUS is remarkably lower than that achieved by CGA. However, the higher performance of CGA

is associated with a higher computational cost, since Genetic Algorithms are a population-based heuristic. As a matter of fact, in our experimental setting a single CGA meta-optimization run required $P \cdot G \cdot R \cdot N \cdot I = 3.0 \cdot 10^8$ fitness evaluations, which took almost 17 hours to complete on a 64-bit Linux machine, with a Core i5 architecture and a CPU running at 2.8 GHz. On the other hand, with the selected β and τ parameters LUS performed an average of 4971 fitness evaluations per single meta-optimization run before reaching the minimum threshold, roughly corresponding to 1.3 hours of CPU time on the same machine.

8.3 PSO EXPERIMENTS

8.3.1 *Experimental Setting*

We now describe the experiments performed with our Particle Swarm Optimizer. Regarding the velocity parameters, we adopted the best combination evolved by the CGA meta-optimizer, since it achieved an higher meta-fitness value with respect to the ones obtained by LUS. The values of the selected parameters for each fitness function are reported in Table 2.

Table 2: CGA-Evolved PSO Parameters

fit_k	w	φ	ψ	v_{max}
fit_1	0.5067	2.8751	1.3587	3.5008
fit_2	0.7614	2.0073	2.0273	2.7183
fit_3	0.2828	2.1824	0.8951	4.2639

We applied our PSO algorithm on the spaces of balanced Boolean functions from $n = 7$ to $n = 12$ variables. The number of particles and iterations were set to $P = 200$ and $I = 400$ respectively. Finally, for each value of n and fitness function fit_k, we carried out $R = 100$ PSO runs.

8.3.2 *Best Solutions Found*

Tables 3 to 5 show for each fitness function the cryptographic properties of the best balanced Boolean functions discovered by PSO, that is, the properties of the global best solution g which scored the highest fitness value among all the $R = 100$ optimization runs. We reported the algebraic degree as well, even if we did not adopt this criterion in any of the three fitness functions. As a general observation, one can notice in Tables 3 and 4 that the Boolean functions discovered by PSO satisfying $\mathrm{CI}(k)$ always have an algebraic degree of $n - 1 - k$, which is

Table 3: Best Boolean Functions Found, fit_1

Property	7	8	9	10	11	12
Nl	56	112	236	480	972	1972
deg	5	6	7	8	9	10
cidev_1	0	0	0	0	0	0
pcdev_1	0	0	8	8	8	8

Table 4: Best Boolean Functions Found, fit_2

Property	7	8	9	10	11	12
Nl	56	112	232	476	972	1972
deg	4	6	7	8	9	10
cidev_1	0	8	8	8	8	16
cidev_2	0	8	8	8	8	16

the maximum allowed by Siegenthaler's bound. Hence, these results empirically confirm that it is not necessary to consider the algebraic degree in the definition of the PSO fitness functions, as we mentioned in Section 8.1.2. Looking in particular at Table 3, we can see that our

Table 5: Best Boolean Functions Found, fit_3

Property	7	8	9	10	11	12
Nl	56	116	236	480	976	1972
deg	5	6	7	9	10	11
AC_{max}	16	32	48	80	128	208

PSO algorithm scales fairly well to higher numbers of variables with respect to the optimization of cidev_1, even if the CGA parameters were evolved only for the case $n = 7$. As a matter of fact, all the best Boolean functions found by PSO with fit_1 are first order correlation immune (and thus 1-resilient, since they are also balanced). Moreover, for $n = 7$ and $n = 8$ they also satisfy the Strict Avalanche Criterion $\mathrm{PC}(1)$, while for higher values of n they reach the minimum deviation $\mathrm{pcdev}_1 = 8$. Nevertheless, our Particle Swarm Optimizer is able to find Boolean functions of up to $n = 11$ variables which satisfy both $\mathrm{CI}(1)$ and $\mathrm{PC}(1)$, even if their nonlinearity is lower (for a detailed comparison with other heuristic methods, see Section 8.3.3).

On the other hand, Table 4 shows that by using fitness function fit_2 the Particle Swarm Optimizer does not perform well when the number of variables is higher than 7. In fact, 2-resilient functions are found only for $n = 7$, while in all other cases the deviation from $\mathrm{CI}(2)$ is at

least 8. However, it is worth noticing that the best solution of 7 variables, besides satisfying with equality Siegenthaler's bound, achieves Tarannikov's bound on nonlinearity as well, since $56 = 2^{7-1} - 2^{2+1}$. Hence, as mentioned in Section 5.2.6, this is a $(7, 2, 4, 56)$ *plateaued* Boolean function.

Finally, another different behaviour of the PSO algorithm can be observed using fitness function $\mathtt{fit}_3$. Indeed, one can see from Table 5 that as the number of variables grows the absolute indicator of the best solution gets worse. Nonetheless, for $n = 8$ and $n = 11$ the nonlinearity values achieved with $\mathtt{fit}_3$ are greater than those obtained using $\mathtt{fit}_1$, while they are equal in all other cases.

8.3.3 *Comparison with other Heuristics*

We now compare the results of our Particle Swarm Optimizer with those obtained by other heuristic methods. Due to the great heterogeneity in the experimental settings and the parameters adopted in the relevant literature, a comprehensive comparison is not possible. For this reason, in Tables 6 to 9 we summarise the results separately for each class of cryptographically significant balanced Boolean functions discovered by the PSO algorithm. A dash symbol in the tables indicates that the corresponding data is not available, either because the heuristic failed to discover Boolean functions with those cryptographic properties or because that specific case was not considered.

Table 6 reports the maximum nonlinearity achieved by $\mathrm{CI}(1)$ functions. In this case, we used Genetic Algorithms (GA) [123], Directed Search Algorithm (DSA) [131] and Simulated Annealing (SA) [43] for the comparison. It can be seen that for $n = 7$ variables our PSO algorithm manages to find 1-resilient functions having maximum nonlinearity 56, while SA stops at 52. For $8 \leqslant n \leqslant 12$, the results achieved by PSO are globally similar to those of the other optimization methods, except in the case of $n = 11$ variables where it reaches a maximum nonlinearity of 972 instead of 976. In particular, our PSO outperforms both Genetic Algorithms and Simulated Annealing for $n = 9$ and $n = 10$ variables.

Table 6: Maximum Nonlinearity Achieved by $\mathrm{CI}(1)$ Functions

Method	7	8	9	10	11	12
GA [123]	–	112	232	476	976	1972
DSA [131]	–	112	236	480	976	–
SA [43]	52	112	232	476	–	–
PSO	56	112	236	480	972	1972

In Table 7 the maximum nonlinearity of balanced Boolean functions which satisfy both CI(1) and PC(1) is considered. By comparing the results achieved by PSO and SA, we can see that also in this case the former reaches a higher value of nonlinearity for $n = 7$ variables, while for $n = 8$ it is equal to SA. To our knowledge, no heuristic method has ever been applied to discover functions satisfying both CI(1) and PC(1) of $n > 8$ variables. However, our PSO algorithm managed to find this kind of functions for up to $n = 11$ variables, even though for $n > 8$ they were not the best solutions among all the optimization runs with respect to fitness function fit_1. The nonlinearity of these functions is reported in Table 7 for completeness.

Table 7: Maximum Nonlinearity Achieved by CI(1) and PC(1) functions

Method	7	8	9	10	11	12
SA [43]	52	112	—	—	—	—
PSO	56	112	232	476	968	—

Table 8 reports the maximum nonlinearity achieved by Boolean functions with minimal deviation from second order correlation immunity. In particular, the performances of PSO and GA are compared, since in this case we used the same fitness function defined in [123]. As we already discussed in Section 8.3.2, we can observe that our PSO algorithm does not generalize well to higher numbers of variables. As a matter of fact, PSO manages to reach the same results achieved by GA only for $n = 8$ variables, while in all other cases either the nonlinearity or the deviation from CI(2) is worse. We remark however that for $n = 7$ the 2-resilient functions found by PSO have the same value of nonlinearity as the ones discovered by SA in [43].

Table 8: Comparison of Nl and $cidev_2$ Values

Method		7	8	9	10	11	12
GA [123]	Nl	—	112	232	480	976	1972
	$cidev_2$	—	4	8	8	8	8
PSO	Nl	56	112	232	476	972	1972
	$cidev_2$	0	8	8	8	8	16

Similar considerations can be made for the comparisons in Table 9, which reports the maximum nonlinearity reached by Boolean functions having minimal absolute indicator. The benchmark heuristics in this case are Multi-Objective Random Bit Climber (RBC) [2], Genetic Programming (GP) [142] and again SA. It can be observed that for $n = 7$ variables PSO obtained the same results as RBC and SA, while for $n = 8$ it discovered the same combination of Nl and AC_{max} fea-

Table 9: Comparison of Nl and AC_{max} Values

Method		7	8	9	10	11	12
RBC [2]	Nl	56	116	–	–	–	–
	AC_{max}	16	24	–	–	–	–
GP [142]	Nl	–	116	–	–	–	–
	AC_{max}	–	32	–	–	–	–
SA [43]	Nl	56	116	238	484	982	1986
	AC_{max}	16	24	40	56	88	128
PSO	Nl	56	116	236	480	976	1972
	AC_{max}	16	32	48	80	128	208

tured by GP. However, for $n > 8$ our PSO scored worse values than SA with respect to both nonlinearity and absolute indicator.

8.4 CONCLUSIONS

In this chapter, we applied a new PSO algorithm to search for balanced Boolean functions from $n = 7$ to $n = 12$ variables with good cryptographic properties. The performed experiments lead us to conclude that our PSO is able to generate Boolean functions having similar or better combinations of nonlinearity, first order correlation immunity and Strict Avalanche Criterion than those obtained by other optimization methods, while it does not perform well when it minimizes deviation from $\mathrm{CI}(2)$ or the absolute indicator. The only notable exception is that the best solution found by PSO under fit_2 for 7 variables is a $(7, 2, 4, 56)$ plateaued Boolean function, which thus satisfies with equality both Siegenthaler's and Tarannikov's bounds.

The reason of the poor performances of PSO with the second and third fitness functions could lie in the fact that the velocity parameters were evolved only for the case of $n = 7$ variables. This suggests that further parameters tuning is required for $n \geqslant 8$. Considering the high computational cost of our meta-fitness function, it may be preferable to use the LUS meta-optimizer for this task instead of CGA.

EVOLVING PLATEAUED BOOLEAN FUNCTIONS BY GENETIC ALGORITHMS

9

An interesting approach for designing Boolean functions with good cryptographic properties is *spectral inversion*, which we overviewed in Chapter 7. Originally introduced by Clark et al. [42], this method uses a dual representation of the optimization problem where the candidate solutions are Walsh spectra instead of truth tables encoded by bitstrings. The advantage of this representation is that several interesting cryptographic properties of Boolean functions can be easily characterized over the Walsh spectrum, such as resiliency and nonlinearity. The problem is that applying the inverse Walsh transform to a generic spectrum that satisfies these properties does not yield a Boolean function in general, but rather a *pseudoboolean function* from $\mathbb{F}_2^n$ to $\mathbb{R}$. Thence, the objective of this dual optimization problem is to minimize the deviation of the candidate pseudoboolean functions. The optimal solutions are thus Boolean functions $f : \mathbb{F}_2^n \to \mathbb{F}_2$ which already satisfy by design the desired cryptographic properties encoded by their Walsh spectra.

The goal of this chapter is to investigate the application of *permutation based Genetic Algorithms* for evolving cryptographic Boolean functions by spectral inversion, a method which was conjectured to be more efficient than Simulated Annealing in [42].

In particular, we design a GA in which the chromosomes of the evolved solutions are Walsh spectra of *plateaued pseudoboolean functions*. The motivation for this choice is twofold. First, the spectra of plateaued pseudoboolean functions are three-valued, hence they have an easy combinatorial characterization. Moreover, plateaued Boolean functions are optimal with respect to both Siegenthaler's and Tarannikov's bounds on the maximum achievable algebraic degree and nonlinearity for a given resiliency order. Since our GA manipulates permutations of *repeated* values, we propose a crossover and a mutation operator which ensure that the modified genes in the offspring correspond to different values in the Walsh spectrum.

Let us recall that, as the number of Boolean functions of n variables is 2^{2^n}, exhaustively searching for plateaued Boolean functions (or, more in general, cryptographically relevant Boolean functions) becomes unfeasible for $n > 5$. For this reason, we assess the performance of our GA in generating plateaued Boolean functions of $n = 6$ and $n = 7$ variables. The results show that our GA outperforms Simulated Annealing in finding plateaued Boolean functions of $n = 6$ variables, while for $n = 7$ SA still yields better average fitness val-

ues, even if neither technique is able to generate a plateaued Boolean function in this case.

The remainder of this chapter is organized as follows. Section 9.1 describes our permutation-based Genetic Algorithm, defining the solution encoding, the fitness function and the adopted genetic operators. Section 9.2 presents the results obtained by our GA on the optimization of pseudoboolean plateaued functions for $n = 6$ and $n = 7$ variables, and compares them with the results achieved by the SA algorithm described in [42]. Finally, Section 9.3 recaps the contributions of this chapter.

9.1 GENETIC ALGORITHM DESCRIPTION

9.1.1 *Chromosomes Encoding*

The main idea underlying the chromosome encoding of our GA is to represent a candidate solution as a *permutation* of a Walsh spectrum $\mathcal{S} \in \mathbb{R}^{2^n}$. This *spectral inversion* approach to heuristic design of cryptographic Boolean functions was originally introduced by Clark, Jacob, Maitra and Stanica in [42].

As a first observation, notice that representing the chromosome as a permutation of the spectrum *positions* would allow us to employ classic permutation-based GA, such as those designed for the Traveling Salesman Problem [71]. However, the Walsh spectrum is generally composed of *repeated* values. This means that a position-based encoding would make the GA search into a space which is much bigger than what is actually needed, since several swaps performed by permutation-based genetic operators would map to the same values in the Walsh spectrum. Hence, we represent our candidate solution directly by its Walsh spectrum *values*, which is equivalent to performing permutations over a *multiset* $\mathcal{M}$.

Recall from Section 5.1 that, by Parseval's identity, the sum of the squared Walsh coefficients of any n-variable Boolean function equals 2^{2n}. Moreover, the values occurring in Equation (54) of the Walsh transform are all integers, hence we can start to model a candidate solution as a vector of 2^n integers which sum to 2^{2n}. Additionally, we are interested only in plateaued Boolean functions, so that each Walsh coefficient can only take its value in the set $V = \{-\mathcal{L}(f), 0, +\mathcal{L}(f)\}$, where the spectral radius is defined as $\mathcal{L}(f) = 2^r$, with $\lceil \frac{n}{2} \rceil \leqslant r \leqslant n$. We thus need to determine the *multiplicities* of the elements of V in order to characterize the multiset $\mathcal{M}$ required to build the spectrum. Using the approach sketched in [42], these multiplicities can be derived from the following observations:

(1) Since a plateaued Boolean function is t–resilient with $t = r - 2$, all positions which correspond to input vectors having at most t nonzero coordinates must be set to zero. Therefore, in order

to meet t–resiliency there must be *at least* $\#0_{res} = \sum_{i=0}^{t} \binom{n}{i}$ zero-valued positions in the spectrum.

(2) Each *nonzero position* in the spectrum contributes by a term of $(\pm 2^r)^2 = 2^{2r}$ in Parseval's identity. Thus, the total number of nonzero positions in the spectrum is given by $\# \pm 2^r = \frac{2^{2n}}{2^{2r}}$.

(3) From (1) and (2) we deduce that the number of additional positions set to zero other than the ones for satisfying t–resiliency is $\#0_{add} = 2^n - ((\# \pm 2^r) + (\#0_{res}))$

(4) Setting $\hat{f}(\underline{0}) = 1$ yields that $\sum_{\omega \in \mathbb{F}_2^n} W_f(\omega) = 2^n$. Notice that this is an arbitrary assumption, since we are considering only those functions mapping the null vector to 0. However, this does not bias the final search space, since by setting $\hat{f}(\underline{0}) = -1$ one would always get plateaued functions having the same profile.

(5) By combining observations (2) and (4), we finally obtain the number of positions to be set to -2^r and $+2^r$ by solving the following system:

$$\begin{cases} (\# + 2^r) + (\# - 2^r) & = \frac{2^{2n}}{2^{2r}} \\ (\# + 2^r) - (\# - 2^r) & = 2^n \end{cases}$$

which gives

$$\begin{cases} \# + 2^r & = 2^{n-1}(2^{n-2r} + 1) \\ \# - 2^r & = 2^{n-1}(2^{n-2r} - 1) \end{cases}$$

In what follows, we denote by $x[i]$ the element at position i of vector x. Since there are $\#0_{res}$ positions in the spectrum which are set to zero for the resiliency constraint, we can restrict our representation only to those positions whose binary expansions have more than t nonzero coordinates. Hence, let us consider the *restricted ordered spectrum* defined as $\mathcal{S}_{ro} = (z_1, \cdots, z_l)$ having length $l = 2^n - \#0_{res}$ and whose first $\#0_{add}$ positions are set to zero, the next $\# - 2^r$ are set to -2^r and the final $\# + 2^r$ are set to $+2^r$. Additionally, let us denote by $P_r = \{j_1, \cdots, j_l\}$ the set of positions j_i such that $w_H(\mathrm{bin}(j_i)) > t$, where $w_H(\cdot)$ is the Hamming weight of the binary string passed as argument. Clearly, by permuting the components in $\mathcal{S}_{ro}$ the resulting spectrum maintains the desired cryptographic properties, since the multiplicities $\#0_{add}$, $\# - 2^r$ and $\# + 2^r$ are permutation invariant. However, we are interested only in those permutations which swap different values in the restricted spectrum. To address this problem, we employ the following equivalence relation $\sim_p$ on the symmetric group S_l: given two permutations $\pi_1, \pi_2 \in S_l$, define $\pi_1 \sim_p \pi_2$ if and only if $z_{\pi_1(i)} = z_{\pi_2(i)}$ for all $i \in \{1, \cdots, l\}$, where $z_{\pi_1(i)}, z_{\pi_2(i)}$ are components of $\mathcal{S}_{ro}$. We can thus characterize the permutations which

map different values in the restricted spectrum as the representatives of the equivalence classes in the quotient set $S_l/\sim_p$. With a little abuse of notation, in what follows we write $\pi \in S_l/\sim_p$ to directly denote the representative permutation π instead of the equivalence class $[\pi]_{\sim_p}$.

The *chromosome* which encodes a candidate solution evolved by our GA is a permutation $c = (z_{\pi(1)}, \cdots, z_{\pi(l)})$ of the restricted ordered spectrum $\mathcal{S}_{ro}$, where $\pi \in S_l/\sim_p$. The *decoding* of chromosome c which yields the corresponding pseudoboolean function $f : \mathbb{F}_2^n \to \mathbb{R}$, denoted by $dec(c)$, is carried out using the following procedure:

1. Initialize the Walsh spectrum $\mathcal{S}_f$ to the null vector $\underline{0} \in \mathbb{R}^{2^n}$.
2. For all $i \in \{1, \cdots, l\}$ set $\mathcal{S}_f[j_i] = c[i]$, where $j_i = P_r[i]$.
3. Perform *spectral inversion*: apply to $\mathcal{S}_f$ the inverse Walsh transform defined in Equation (53) in order to obtain the polar form $\hat{f}$ of function f.

9.1.2 *Objective and Fitness Functions*

In order to measure how good a pseudoboolean function is, the authors of [42] proposed an objective function based on the distance from the *nearest Boolean function*. Formally, given the polar form $\hat{f}$ of $f : \mathbb{F}_2^n \to \mathbb{R}$, the polar truth table of the nearest Boolean function $\hat{b} : \mathbb{F}_2^n \to \{-1, +1\}$ is obtained for all $x \in \mathbb{F}_2^n$ as follows:

$$\hat{b}(x) = \begin{cases} +1 & , \text{ if } \hat{f}(x) > 0 \\ -1 & , \text{ if } \hat{f}(x) < 0 \\ +1 \text{ or } -1 \text{ (chosen randomly)} & , \text{ if } \hat{f}(x) = 0 \end{cases} \tag{88}$$

Given a chromosome c and the corresponding pseudoboolean function $f = dec(c)$, the *objective function* to be minimized proposed in [42] is defined as:

$$obj(f) = \sum_{x \in \mathbb{F}_2^n} (\hat{f}(x) - \hat{b}(x))^2 \ . \tag{89}$$

This objective function measures the *deviation* of f from being a true Boolean function. Hence, an optimal solution to our problem is encoded by a chromosome c such that $obj(dec(c)) = 0$. Given how we designed the Walsh spectrum, such a solution corresponds to a plateaued Boolean function.

The *fitness function* $fit(\cdot)$ maximized by our GA is simply defined as the opposite of the objective function (89), that is, $fit(f) = -obj(f)$.

9.1.3 *Genetic Operators*

Considering the chromosome encoding adopted for the candidate solutions, an appropriate crossover operator for our GA has to preserve

the multiplicities $\#0_{add}$, $\#-2^r$ and $\#+2^r$ of the restricted spectrum, so that Parseval's identity and the other properties of plateaued functions are maintained. To this end, we designed a crossover operator loosely inspired by the one proposed in [123].

The main idea is to work at the loci level, and to use *counters* in order to keep track of the multiplicities of the three values 0, -2^r and $+2^r$ inserted in the offspring during the crossover phase. More precisely, given two parent chromosomes c_1 and c_2, our crossover operator builds an offspring chromosome o as follows:

1. Initialize to zero the counters cnt_z, cnt_n and cnt_p respectively associated to the spectral values 0, -2^r and $+2^r$.
2. For all $i \in \{1, \cdots, l\}$ such that $c_1[i] = c_2[i]$, copy either $c_1[i]$ or $c_2[i]$ in $o[i]$. Depending on the copied value, update the relevant counter.
3. For all $i \in \{1, \cdots, l\}$ such that $c_1[i] \neq c_2[i]$, determine the value to be copied in $o[i]$ as follows:
   a) If all three counters are below their maximum values (that is, $cnt\_z < \#0_{add}$, $cnt\_n < \#-2^r$ and $cnt\_p < \#+2^r$), randomly select $c_1[i]$ or $c_2[i]$ with probability $1/2$, and copy it in $o[i]$. Depending on the copied value, update the relevant counter.
   b) If one of the three counters reached its maximum value, check if either $c_1[i]$ or $c_2[i]$ is equal to the value associated to that counter. If so, copy the gene of the other parent in $o[i]$. Otherwise, randomly select $c_1[i]$ or $c_2[i]$ with probability $1/2$, and copy it in $o[i]$. In both cases, depending on the copied value, update the relevant counter.
   c) If two out of three counters reached their respective maximum values, copy the value associated to the remaining counter in $o[i]$.
4. Return the offspring chromosome o.

Concerning the mutation operator, we adopted a simple swap procedure which checks that the swapped values are different. In particular, let us assume that c is a chromosome of length l and that pos_0, pos_{-2^r} and pos_{+2^r} are the vectors specifying the positions of the zeros, -2^rs and $+2^r$s in c, respectively. Then, our mutation operator is applied to each locus $i \in \{1, \cdots, l\}$ of c with probability $p_\mu \in [0, 1]$, and it performs the following steps:

1. Setting $v = c[i]$, randomly select with probability $1/2$ one of the two positions vectors pos_t or pos_u, where $t \neq v$ and $u \neq v$.
2. Denoting by pos_s the selected positions vector, randomly draw with uniform probability an index j of pos_s.

3. Swap the values $c[i]$ and $c[pos_s[j]]$.
4. Swap the occurrence of i in pos_v with $pos_s[j]$.

Finally, for the selection operators we tested both *roulette wheel selection* and *deterministic tournament selection*. In roulette wheel selection, each candidate is selected with probability proportional to its fitness. Tournament selections, on the other hand, chooses the best individual from a set of T individuals randomly drawn from the population.

9.1.4 *Overall GA Procedure*

We can now summarize the overall procedure of our genetic algorithm. The input parameters are the number of variables n and the index $r \geqslant \frac{n}{2}$ of the target plateaued functions, the size of the population N (where N is even), the number of generations G to be performed, the crossover and mutation probabilities p_χ and p_μ, and the selection operator S.

1. *Initialization*: Set the profile $(n, r-2, n-r-3, 2^{n-1}-2^{r-1})$ of the target functions and the multiplicities $\#0_{res}$, $\#0_{add}$, $\#-2^r$ and $\#+2^r$ of the Walsh spectrum.
2. *Create Population*: For $i \in \{1, \cdots, N\}$, create a chromosome $c = (z_{\pi(1)}, \cdots, z_{\pi(l)})$ of length $l = 2^n - \#0_{res}$, where π is a random permutation of $S_l/\sim_p$, and add it to the current population $\mathcal{P}$.
3. *Initial Fitness Evaluation*: For each chromosome $c \in \mathcal{P}$, decode its respective pseudoboolean function $f = dec(c)$ and compute the fitness value $fit(f) = -obj(f)$, where $obj(\cdot)$ is defined as in Equation (89). Set the best solution B to the individual scoring the highest fitness value.
4. *Selection Phase*: Apply N times the selection operator S on the current population $\mathcal{P}$, thus creating a candidate population $\mathcal{C}$ of (eventually repeated) N chromosomes which will produce the next generation.
5. *Crossover Phase*: For all $i \in \{1, 3, \cdots, N-1\}$, sample a random number $r \in [0, 1]$. If $r < p_\chi$, apply the crossover operator *twice* to the pair $c_i, c_{i+1} \in \mathcal{C}$, and copy the two offspring chromosomes (o_i, o_{i+1}) in the new population $\mathcal{N}$. Otherwise, set $o_i = c_i$ and $o_{i+1} = c_{i+1}$, and copy them in $\mathcal{N}$.
6. *Mutation Phase*: For each chromosome $o \in \mathcal{N}$ and $j \in \{1, \cdots, l\}$, sample a random number $r \in [0, 1]$. If $r < p_\mu$, apply the mutation operator to $o[j]$.
7. *Fitness Evaluation*: For each chromosome $o \in \mathcal{N}$, compute the fitness value of $f = dec(o)$, and find the current best individual B_c having the highest fitness value in $\mathcal{N}$.

8. *Elitism*: If $\mathrm{fit}(B_c) \leqslant \mathrm{fit}(B)$, replace a random individual in $\mathcal{N}$ with B. Otherwise, update the best solution found so far by setting $B = B_c$.
9. *Population Update*: Set the current population $\mathcal{P}$ equal to $\mathcal{N}$.
10. *Termination Condition*: If the best solution found is optimal (i. e. $\mathrm{obj}(B) = 0$) or the maximum number of generations G has been reached, output the best solution B found by the GA. Otherwise, return to Step 4.

9.2 GA EXPERIMENTS

9.2.1 *Experimental Setting*

We tested our GA on the spaces of pseudoboolean functions of $n = 6$ and $n = 7$ variables, adopting in both cases index $r = 4$. This is the smallest integer value, yielding maximum nonlinearity, such that the resulting functions are not bent for $n = 6$. Table 10 reports the profiles and the multiplicities of the spectrum values for the corresponding plateaued Boolean functions. We limited our experimen-

Table 10: Cryptographic profiles and spectral multiplicities for plateaued functions of $n = 6$ and $n = 7$ variables

(n, m, d, nl)	$\#0_{res}$	$\#0_{add}$	$\# - 2^r$	$\# + 2^r$
$(6, 2, 3, 24)$	22	26	6	10
$(7, 2, 4, 56)$	29	35	28	36

tation to these two problem instances in order to compare our GA with Simulated Annealing. As a matter of fact, the basic SA algorithm described in [42] was able to find only 5 plateaued functions with profile $(7, 2, 4, 56)$ out of 500 optimization runs, and a *change of basis* procedure [43] had to be applied in order to convert some generated sub-optimal solutions into actual Boolean functions. Further, for $n = 6$ only bent functions were considered, but not generic plateaued functions. On the other hand, for higher number of variables the basic version of SA always failed to generate Boolean functions, hence the authors of [42] restricted their search space to the family of *rotation symmetric Boolean functions*, which we did not consider in this chapter.

For each value of n and selection operator considered, we performed $R = 500$ independent runs of our GA, using a population of $N = 30$ chromosomes evolved for $G = 500000$ generations. Thus, each GA run consisted of $F = 1.5 \cdot 10^7$ fitness evaluations. The crossover and mutation probabilities were respectively set to $p_\chi = 0.95$ and $p_\mu = 0.05$, while in the case where tournament selection was used we adopted a tournament size of $k = 3$.

Table 11: Statistics of the best solutions found by GA and SA

n	Stat	GA(RWS)	GA(DTS)	SA(T_1, α_1)	SA(T_2, α_2)
6	avg_o	14.08	13.02	19.01	19.03
	min_o	0	0	0	0
	max_o	16	16	28	28
	std_o	5.21	6.23	4.89	4.81
	#opt	60	93	11	10
	avg_t	83.3	79.2	79.1	79.4
7	avg_o	53.44	52.6	45.09	44.85
	min_o	47	44	32	27
	max_o	58	59	63	57
	std_o	2.40	2.77	4.39	4.18
	#opt	0	0	0	0
	avg_t	204.2	204.5	180.3	180.2

Concerning the comparison with Simulated Annealing, we implemented the algorithm described in [42] and tested it for $n = 6$ and $n = 7$ by setting the number of inner loops MaxIL and moves within an inner loop MIL respectively to MaxIL $= 5000$ and MIL $= 3000$, thus yielding the same number $F = 1.5 \cdot 10^7$ of fitness evaluations performed by our GA. Since the authors of [42] did not mention the initial temperature which they adopted for their experiments, we tested the values $T_1 = 100$ and $T_2 = 1000$ with cooling parameter respectively set to $\alpha_1 = 0.95$ and $\alpha_2 = 0.99$. As in the case of our GA, for each combination of parameters (n, T_0, α) we performed 500 runs of the SA algorithm.

9.2.2 *Results*

We performed all our experiments on a 64-bit Linux machine with a Core i5 architecture and a CPU running at 2.8 GHz. For $n = 6$, a set of 500 runs of GA or SA took approximately 11.5 hours to complete, while for $n = 7$ it took about 28.3 hours and 25 hours for GA and SA, respectively. Table 11 reports the results of the experiments. By GA(RWS) and GA(DTS) we denote our GA respectively with roulette wheel selection and deterministic tournament selection, while SA(T_i, α_i) stands for the SA algorithm run with initial temperature T_i and cooling parameter α_i, for $i \in \{1, 2\}$. For each parameters combination, Table 11 reports the average (avg_o), minimum (min_o), maximum (max_o) and standard deviation (std_o) values of the objective function $obj(\cdot)$ computed on the best solutions found, along with

the numbers of optimal solutions generated (#opt) and the average time per run in seconds (avg_t).

For $n = 6$ it can be observed that both versions of our GA outperforms SA with respect to the ratio of generated $(6, 2, 3, 24)$ functions versus the total number of optimization runs. In particular, the adoption of tournament selection produces better results than roulette wheel selection, with 93 plateaued functions achieved using the former operator against the 60 obtained using the latter one. On the other hand, changing the initial temperature and the cooling parameter α does not seem to influence the SA performances, with only 11 plateaued functions generated by $SA(T_1, \alpha_1)$ and 10 functions generated by $SA(T_2, \alpha_2)$. Notice also that the computational overhead introduced by our GA is not very high: for example, using roulette wheel selection the average time per run of our GA is 83.3 seconds, while with tournament selection a single run takes 79.2 seconds on average, which is in the same range as that employed by SA.

In the case of $n = 7$ variables, neither version of our GA nor SA is able to generate a plateaued Boolean function of profile $(7, 2, 4, 56)$. However, it can be seen that SA outperforms both versions of our GA. In particular, the GA obtains slightly better results using tournament selection than roulette wheel selection, but SA scores lower average objective function values than GA. The same difference can also be observed by comparing the minimum objective function values.

9.3 CONCLUSIONS

In this chapter, we proposed a genetic algorithm to evolve plateaued Boolean functions which satisfy good cryptographic properties. Instead of searching the space of Boolean functions (as it is usually done in the existing literature), we adopted the *spectral inversion* approach set forth by Clark, Jacob, Maitra and Stanica in [42], which represents a candidate solution as a Walsh spectrum already satisfying the desired cryptographic properties. The search space thus becomes the set of all plateaued pseudoboolean functions, and the objective function to be minimized is the distance of the candidate solution from the nearest Boolean function. The representation adopted for the chromosomes of our GA consists in a permutation of a restricted Walsh spectrum, in which the positions related to t–resiliency are not considered, being constantly set to zero. Since the coefficients in the spectrum of a plateaued Boolean function can take only three values, the chromosome actually encodes a permutation on a multiset. The decoding process first maps the loci of the chromosome to the positions in the Walsh spectrum having Hamming weight higher than t, and then the inverse Walsh transform is applied to obtain the associated pseudoboolean function. We designed a specialized crossover operator which employs counters in order to preserve the multiplicities of

the three values characterizing the spectrum of plateaued functions, while for mutation we adopted a simple swap-based operator that only exchanges those positions in the chromosome corresponding to different spectral values.

The performed experiments show that in the case of $n = 6$ variables our GA achieves better results than the Simulated Annealing algorithm proposed in [42] with respect to the ratio of generated $(6, 2, 3, 24)$ Boolean functions per number of optimization runs. In particular, our GA performs better when adopting deterministic tournament selection instead of basic roulette wheel selection, while modifying the initial temperature and the cooling parameter does not significantly change the SA performances. On the other hand, for $n = 7$ no heuristic technique is able to generate a $(7, 2, 4, 56)$ plateaued Boolean function, but SA scores on average lower objective function values than GA.

Extending the comparison to other *direct* heuristic methods (that is, heuristics which directly explore the space of Boolean functions) is not a straightforward task. The reason for this difficulty is twofold. First, there are no obvious ways to compare the sub-optimal solutions found, due to the different representations adopted. In particular, in our GA a sub-optimal solution is a pseudoboolean function which already satisfies the desired cryptographic properties, while in direct methods it is a Boolean function which do not satisfy these criteria. Second, to our knowledge only two direct heuristic methods have been reported in the literature to generate $(6, 2, 3, 24)$ plateaued functions [43, 27], but no information on the ratio of optimal solutions found per number of optimization runs are available. Nonetheless, these methods were also able to locate $(7, 2, 4, 56)$ functions.

Part III

CRYPTOGRAPHIC AND CODING-THEORETIC ANALYSIS OF CELLULAR AUTOMATA

PREIMAGES PERIODS IN SURJECTIVE CA

10

In Chapter 7 we overviewed the secret sharing scheme based on bipermutive CA proposed in [112]. In particular, we remarked that determining the maximum number of players in the cyclic access structure of this scheme is equivalent to characterizing the periods of preimages of spatially periodic configurations in bipermutive CA (BCA).

However, much of the literature on the periodic behavior of CA concerns their *temporal periodicity* [20, 21, 55]: namely, characterizing those integers $t \in \mathbb{N}$ such that, starting from a given configuration x of the cells, the orbit of the CA returns to x after t applications of its global rule F, i. e. $F^t(x) = x$.

Spatial periodicity, on the other hand, is a much less researched topic in the CA literature. As we reviewed in Chapter 2, one of the basic results in this case is that if F is a surjective infinite CA and $y \in \Sigma^{\mathbb{Z}}$ is a SPC then every preimage $x \in F^{-1}(y)$ is spatially periodic as well. This is a direct consequence of the *balancing property* of surjective CA, which implies that every configuration can only have a finite number of preimages (see Lemmas 2 and 1 in Section 2.3).

To our knowledge, there are no works in the literature that address the problem of actually characterizing the periods of SPC preimages. The aim of this chapter is to fill this gap by investigating the relation between the periods of SPC and the periods of their preimages under the action of several classes of surjective CA.

Besides being interesting from the perspective of CA theory and for CA-based secret sharing schemes, this research also has applications in bioinformatics. In fact, the theory of *concatenated linear recurring sequences*, which is a key tool used in the present chapter to characterize the periods of preimages in *linear* bipermutive CA, turns out to be useful also for studying the dynamics of *additive flowers*, a particular class of genetic regulatory networks introduced in [64].

A summary of the main contributions of the chapter follows. Given a SPC $y \in \Sigma^{\mathbb{Z}}$ of least period $p \in \mathbb{N}$, we observe that in generic surjective CA the least period of a preimage $x \in F^{-1}(y)$ is a multiple of p, where the multiplier h ranges in $\{1, \cdots, q^{d-1}\}$, with q being the size of the alphabet and d the diameter of the CA (Lemma 5 and 6). From this result, we also determine a first lower bound on the *multiplicity* of the least period of x, that is, how many other preimages of y have the same least period of x (Lemma 7). Successively, using the *de Bruijn graph representation* of CA, we introduce the notion of u-*closure graph* of a SPC y, whose cycles lengths turn out to be equivalent to the least periods of the preimages of y (Lemma 8). We thus

describe an algorithm to build the u-closure graph starting from any surjective CA F and SPC y. The complexity of this procedure turns out to be exponential in the least period of y and in the diameter of the CA. Remarking that the u-closure graph of a SPC under a BCA is composed only of disjoint cycles (Lemma 9), we narrow our attention to the special case of *linear bipermutive* CA (LBCA) defined over the *finite field* $\mathbb{F}_q$. In particular, we show that a preimage $x \in F^{-1}(y)$ is equivalent to a *concatenated linear recurring sequence* (CLRS), whose characteristic polynomial is the product of the characteristic polynomials respectively induced by the CA local rule and by configuration y (Theorem 19). Additionally, we present a procedure that given a $(d-1)$–cell block of a preimage $x \in F^{-1}(y)$ as input determines the least period of x. Moreover, we characterize the multiplicities of the least periods under the t-*th iterate* $F^{-t}(y)$ when the characteristic polynomial of the local rule is irreducible and does not divide the characteristic polynomial of y (Theorem 21). Finally, these results are generalized to LBCA defined over the *finite ring* $\mathbb{Z}_m$ (Theorem 22), using the *product CA conjugacy* described in [33].

The rest of this chapter is organized as follows. Section 10.1 shows that the least periods of SPC preimages are multiples of the periods of their respective images, and introduces the notion of u-closure graph of a SPC along with the algorithm to compute the multiplicities of the least periods in surjective CA. Section 10.2 characterizes preimages of LBCA as concatenated linear recurring sequences and derives a characteristic polynomial for the latter. Section 10.3 presents an algorithm to compute the least period of a single LBCA preimage, characterizes the multiplicities of the least periods in the particular case where the characteristic polynomial of the local rule is irreducible and generalizes the previous results to LBCA defined over finite rings as alphabets. Finally, Section 10.4 summarizes the results of this chapter.

10.1 PROBLEMS STATEMENT AND BASIC RESULTS

In this section, we present some basic results concerning the periods of preimages of spatially periodic configurations in surjective CA. To this end, we begin by formally stating the first main problem analyzed in this chapter, generalized to the t-th iterate case:

Problem 1. *Let* $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ *be a surjective CA defined by a local rule* $f : \Sigma^d \to \Sigma$, *let* $y \in \Sigma^{\mathbb{Z}}$ *be a SPC of least period* $p \in \mathbb{N}$ *and* $x \in F^{-t}(y)$ *be a* t*-th ancestor of* y, *for* $t \in \mathbb{N}$. *What is the least period of* x*?*

Besides computing the period of a single preimage, we are also interested in counting the *multiplicities* of all least periods appearing in the set of preimages of a spatially periodic configuration:

Problem 2. *Let* $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ *be a surjective CA defined by a local rule* $f : \Sigma^d \to \Sigma$, *with* $|\Sigma| = q$, *and* $y \in \Sigma^{\mathbb{Z}}$ *be a SPC of least period* $p \in \mathbb{N}$.

For all multipliers $h \in \{1, \cdots, q^{d-1}\}$*, what is the number of preimages* $N_h(y, F)$ *of* y *under* F *having least period* hp*?*

10.1.1 *Periods of SPC Preimages in Surjective CA*

We begin our analysis of Problem 1 by considering the general case where the CA is only surjective. To this end, we first show that if $y \in \Sigma^{\mathbb{Z}}$ is a SPC having least period $p \in \mathbb{N}$, then the least periods of its preimages are multiples of p.

Lemma 5. *Let* $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ *be a surjective CA,* $y \in \Sigma^{\mathbb{Z}}$ *be a spatially periodic configuration of least period* $p \in \mathbb{N}$ *and* $x \in F^{-1}(y)$ *be a preimage of* y*. Then, the least period* $k \in \mathbb{N}$ *of* x *is a multiple of* p*.*

Proof. Suppose that k is not a multiple of p, and let $k = jp + r$ with $j = \lfloor k/p \rfloor$ and $0 < r < p$. Since x is spatially periodic of least period x, it follows that $\sigma^k(x) = x$. Moreover, by Hedlund's theorem it holds that $F(\sigma^t(x)) = \sigma^t(F(x))$ for all $t \in \mathbb{Z}$. Hence,

$$\begin{aligned} y = F(x) = F(\sigma^k(x)) = \sigma^k(F(x)) = \sigma^k(y) = \\ = \sigma^{jp+r}(y) = \sigma^r(\sigma^{jp}(y)) = \sigma^r(y) \neq y \end{aligned}$$

where the last inequality follows from the fact that $r < p$. Having obtained a contradiction, k is a multiple of p. □

By employing the *balancing* condition of surjective CA, the following result gives an upper bound on the value of the least period multiplier:

Lemma 6. *Let* $|\Sigma| = q$ *and let* $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ *be a surjective CA defined by a local rule* f *of diameter* d*. Further, let* $y \in \Sigma^{\mathbb{Z}}$ *be a SPC of least period* $p \in \mathbb{N}$ *and* $x \in F^{-1}(y)$ *be a preimage of* y *having least period* $k = hp$*. Then,* $h \in \{1, \cdots, q^{d-1}\}$*.*

Proof. The proof of Lemma 5 already implies that $h \geqslant 1$, so it suffices to show that $h \leqslant q^{d-1}$. In what follows, by $F_m : \Sigma^{m+d-1} \to \Sigma^m$ we denote the NBCA of length $m + d - 1$ induced by F.

Let $u \in \Sigma^p$ be a block of length p taken from y (hence $y = {}^{\omega}u^{\omega}$), and let $s = d - 1$ and $Q = q^s$. By Lemma 2, we have that $|F_m^{-1}(u)| = Q$, where $m = p + s$. Hence, there are Q blocks $x_1, x_2, \cdots, x_Q \in \Sigma^m$ such that $F_m(x_i) = u$ for all $i \in \{1, \cdots, Q\}$. Since $x \in F^{-1}(y)$ is a SPC of least period hp, there exists $v \in \Sigma^{hp+s}$ with $v = w_1 z w_1$ and $w_1 \in \Sigma^s$ such that $x = {}^{\odot}v^{\odot}$ and $F_{hp+s}(v) = u^h$ (see Figure 22). As a consequence, block v is obtained by "gluing" together h blocks of $F_m^{-1}(u)$ using the s-fusion operator. Formally, this means that $v = \bigodot_{x_j \in S} x_j$, where $S \subseteq F_m^{-1}(u)$ and $|S| = h$. Recalling that $|F_m^{-1}(u)| = Q = |\Sigma|^s = |\Sigma|^{d-1}$, it follows that $h \leqslant q^{d-1}$. □

The following Corollary straightforwardly generalizes Lemma 6 to preimages under the t-th iterate of F:

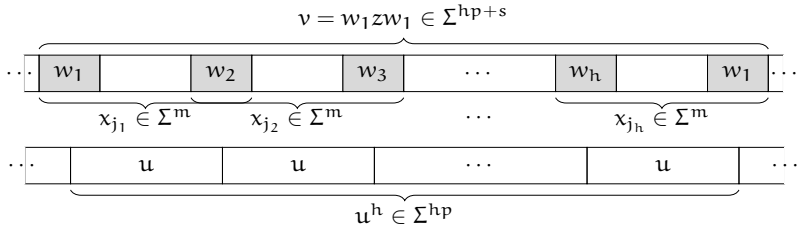


Figure 22: Construction of a preimage by s-fusion of finite blocks.

Corollary 1. *Let* $|\Sigma| = q$, *and let* $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ *be a surjective CA defined by a local rule of diameter* r. *Further, let* $y \in \Sigma^{\mathbb{Z}}$ *be a SPC of least period* $p \in \mathbb{N}$ *and* $x \in F^{-t}(y)$ *be a* t*-th ancestor of* y. *Then, the least period of* x *equals* $k = \left(\prod_{i=1}^{t} h_i\right) \cdot p$, *where* $h_i \in \{1, \cdots, q^{d-1}\}$ *for all* $i \in \{1, \cdots, t\}$.

Proof. We prove the result by induction on $t \in \mathbb{N}$. First, remark that the base case $t = 1$ corresponds to Lemma 6. For the induction step, assume that the condition holds up to $t-1$, and consider a t-th ancestor $x \in F^{-t}(y)$. Clearly, x can be expressed as a preimage of a preimage $x_{t-1} \in F^{-(t-1)}(y)$ under the $(t-1)$-th iterate of F, i. e. $x \in F^{-1}(x_{t-1})$. By Lemma 6 we know that the least period of x is $k = h_t k_{t-1}$, where $h_t \in \{1, \cdots, q^{d-1}\}$ and k_{t-1} is the least period of k_{t-1}. Further, by induction hypothesis we have $k_{t-1} = \left(\prod_{i=1}^{t-1} h_i\right) \cdot p$, where $h_i \in \{1, \cdots, q^{d-1}\}$ for all $i \in \{1, \cdots, t-1\}$. Hence, it follows that $k = h_t \cdot \left(\prod_{i=1}^{t-1} h_i\right) \cdot p = \left(\prod_{i=1}^{t} h_i\right) \cdot p$. □

The following lemma gives a first lower bound on $N_h(y, F)$:

Lemma 7. *Let* $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ *be a surjective CA such that* $|\Sigma| = q$, $y \in \Sigma^{\mathbb{Z}}$ *a SPC of least period* $p \in \mathbb{N}$, *and* $x \in F^{-1}(y)$ *a preimage of* y *having least period* hp, *with* $h \in \{1, \cdots, q^{d-1}\}$. *Then,* $N_h(y, F) \geqslant h$.

Proof. Since x is a preimage of y, we have to show that there are at least $h-1$ other preimages of y having least period hp. Given that $\sigma^p(y) = y$, by Hedlund's theorem we know that the following identity stands for all $i \in \mathbb{Z}$:

$$F(\sigma^{ip}(x)) = \sigma^{ip}(F(x)) = \sigma^{ip}(y) = y \ ,$$

In particular, if $i \in \{1, \cdots, h-1\}$ then $\sigma^{ip}(x) \neq x$ (otherwise, this would contradict the hypothesis that x has least period hp). Thus, we can construct $h-1$ distinct preimages of y by simply shifting x of ip coordinates, for $i \in \{1, \cdots, h-1\}$. All these preimages have least period hp, hence it follows that $N_h(y, F) \geqslant 1 + h - 1 = h$. □

10.1.2 *Graph Characterization of Preimages*

We now introduce a graph-based method to study the periods of preimages in surjective CA. Given a CA $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ defined on

an alphabet A such that $|\Sigma| = q$, and a configuration $y \in \Sigma^{\mathbb{Z}}$, a preimage $x \in F^{-1}(y)$ can be viewed as a *bi-infinite path* π labeled by y on the associated de Bruijn graph $G_{DB}(f)$, i.e. $\pi = \{v_i\}_{i\in\mathbb{Z}}$ such that $l(v_i, v_{i+1}) = y_i$ for all $i \in \mathbb{Z}$. In particular, by setting $s = d-2$, preimage x can be defined as the bi-infinite s-fusion of the vertices visited by π, that is, $x = \bigodot_{v_i\in\pi} v_i$. If F is surjective, for all configurations $y \in \Sigma^{\mathbb{Z}}$ we can always find at least one bi-infinite path on $G_{DB}(f)$ labeled by y.

We now define a second graph which will be used to determine the least periods of the preimages and their multiplicities:

Definition 31. *Let $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ be a surjective CA and let $G_{DB}(f)$ be its de Bruijn graph. Additionally, let $y \in \Sigma^{\mathbb{Z}}$ be spatially periodic of least period $p \in \mathbb{N}$, and let $u \in \Sigma^p$ be a block of length p of y, i.e. $y = {}^{\omega}u^{\omega}$. The* u-closure *of $G_{DB}(f)$ (also called the* unfolding *of $G_{DB}(f)$ along u) is the graph $\overline{G^u_{DB}}(f) = (V, E)$, where:*

- $V = \Sigma^{d-1}$
- *Given $v_1, v_2 \in V$, $(v_1, v_2) \in E$ if and only if there exists a finite path $\pi = v_1, \cdots, v_2$ labeled by u on the de Bruijn graph $G_{DB}(f)$*

As the next Lemma shows, the cycle structure of the u-closure graph is directly related to the least periods of the preimages of $y = {}^{\omega}u^{\omega}$ and their multiplicities.

Lemma 8. *Let $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ be a CA defined by local rule $f : \Sigma^d \to \Sigma$, and let $y = {}^{\omega}u^{\omega} \in \Sigma^{\mathbb{Z}}$ for $u \in \Sigma^p$ be a SPC of least period $p \in \mathbb{N}$. Given $h \in \{1, \cdots, q^{d-1}\}$, denote by $\mathcal{C}^u_h(f)$ the (possibly empty) set of distinct cycles of length h in the u-closure graph $\overline{G^u_{DB}}(f)$. Then, the number of preimages $x \in F^{-1}(y)$ of least period hp equals $N_h(y, F) = h \cdot |\mathcal{C}^u_h(f)|$.*

Proof. Remark that an edge (w_1, w_2) of $\overline{G^u_{DB}}(f)$ represents the first and the last $(d-1)$–cell blocks of a finite preimage $v \in F^{-1}_{p+d-1}(u)$. Considering Figure 22, this means that the blocks $w_1, \cdots, w_h, w_1$ occurring in $x \in F^{-1}(y)$ between the end and the beginning of a copy of u correspond to a cycle $c \in \mathcal{C}^u_h(f)$ of length h in $\overline{G^u_{DB}}(f)$. Thus, by Lemma 7 a single cycle $c \in \mathcal{C}^u_h(f)$ identifies h possible preimages of least period hp, depending from which vertex the path starts. Therefore, the number of preimages of least period hp is given by the number of distinct cycles of length h multiplied by h. □

In order to build the u-closure graph, it is possible to use a variation of *depth-first search* (DFS) in which the de Bruijn graph is explored up to depth p following only the paths labeled by u, without checking if a node has already been visited or not. In order to assess the time complexity of this procedure, observe first that the out-degree of each vertex $v \in \Sigma^{d-1}$ in $G_{DB}(f)$ is $|\Sigma| = q$, and thus v can have at most q outgoing edges labeled by the same symbol $s \in A$. Consequently,

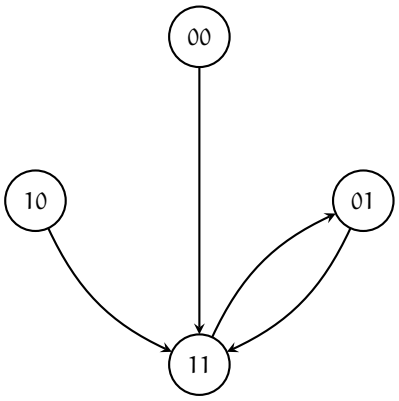


Figure 23: $y = {}^{\omega}011^{\omega}$

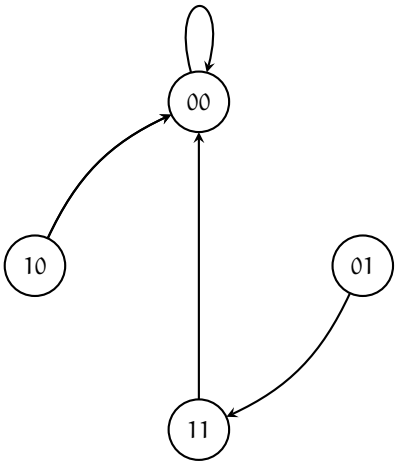


Figure 24: $y = {}^{\omega}1000^{\omega}$

Figure 25: Examples of u-closure graphs for the elementary CA 106.

starting from $v \in \Sigma^{d-1}$ the DFS can visit at most the following number of vertices:

$$1 + q + q^2 + \cdots + q^p = \sum_{i=0}^{p} q^i = \frac{q^{p+1} - 1}{q - 1} = O(q^p) \ .$$

In particular, the worst case occurs when for each symbol u_i of u each node in the i-th level of the DFS tree has q outgoing edges labeled by u_i. Since the DFS must be called for all $v \in \Sigma^{d-1}$, the time complexity for building the u-closure graph is thus $O(q^{d-1} \cdot q^p) = O(q^{p+d-1})$.

The u-closure graph contains at most q^{d-1} edges, since u has exactly q^{d-1} preimages under $F^{-1}_{p+d-1}(u)$ and there can be at most a one-to-one correspondence between the prefixes and the suffixes of length $d-1$ of these preimages. Thus, once the u-closure graph is built, a DFS visit can be employed to determine its cycles and their respective lengths in $O(q^{d-1})$ steps. Starting from the de Bruijn graph of a surjective CA as input, this means that the overall procedure to compute the least periods of the preimages of y and their cardinalities takes $O(q^{p+d-1} + q^{d-1})$ steps.

Notice that, in general, the u-closure of $G_{DB}(f)$ is not composed of disjoint cycles. Figures 23 and 24 report two examples of u-closure graphs for the CA F based on rule 106, the former corresponding to the configuration $y = {}^{\omega}011^{\omega}$ and the latter for $y = {}^{\omega}1000^{\omega}$.

In both cases, the resulting u-closure graphs have cycles with preperiods, and all $(d-1)$–cell blocks in the preperiods cannot appear in any preimage of y (otherwise, the preimages containing them would not be spatially periodic, contradicting Lemma 2).

We now show that if the local rule is *bipermutive* then $\overline{G^{u}_{DB}(f)}$ is composed only of disjoint cycles:

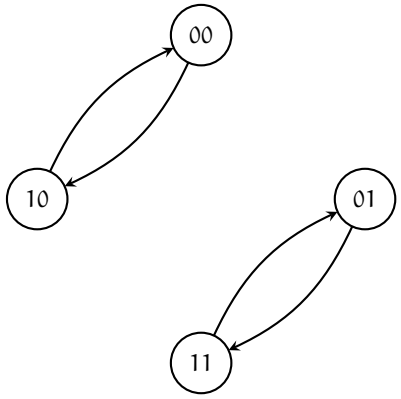


Figure 26: $y = {}^{\omega}011^{\omega}$

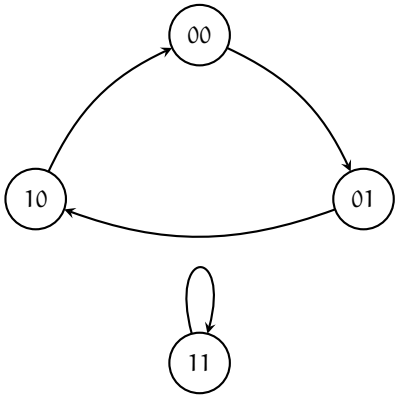


Figure 27: $y = {}^{\omega}1000^{\omega}$

Figure 28: Examples of u-closure graphs for the elementary CA 150.

Lemma 9. *Let $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ be a surjective CA defined by a bipermutive local rule $f : \Sigma^d \to A$. Then, for all SPC $y = {}^{\omega}u^{\omega}$ of least period $p \in \mathbb{N}$ with $u \in \Sigma^p$, the u-closure graph $\overline{G^u_{DB}}(f)$ is composed only of disjoint cycles.*

Proof. Let $v \in \Sigma^{d-1}$ be a vertex of the de Bruijn graph $G_{DB}(f)$. Since f is right permutive, the set of labelings $l(v, w_i)$ of the outgoing edges of v is a permutation on Σ. Hence, there exists exactly one path starting from v and labeled by u on the de Bruijn graph, which means that v has exactly one outgoing edge in the u-closure graph $G^u_{DB}(f)$. Analogously, since f is also left permutive, the set of labelings $l(w_i, v)$ of the incoming edges of v is a permutation on Σ as well. As a consequence, there is exactly one path ending in v and labeled by u on $G_{DB}(f)$, meaning that v has exactly one incoming edge $\overline{G^u_{DB}}(f)$. Since each vertex of the u-closure graph $\overline{G^u_{DB}}(f)$ has both in-degree and out-degree equal to 1, the thesis follows. □

A consequence of Lemma 9 is that the construction of the u-closure graph takes $\Theta(q^{d-1} \cdot p)$ steps for bipermutive CA, since each DFS call on the de Bruijn graph returns only one path labeled by u. Figures 26 and 27 depict the u-closure graphs for $y = {}^{\omega}011^{\omega}$ and $y = {}^{\omega}1000^{\omega}$ under the elementary bipermutive rule 150, defined as $f_{150}(x_{i-1}, x_i, x_{i+1}) = x_{i-1} \oplus x_i \oplus x_{i+1}$.

As a concluding remark for this section, observe that the construction of the u-closure graph, as well as Lemma 8, can be generalized by induction to the t-th iterate F^t. Of course, in this case both the construction of the graph and its visit become exponential in t, thus yielding a total complexity of $O(q^{p+(d-1)t} + q^{(d-1)t})$ for determining the multiplicities of the least periods in $F^{-t}(y)$. On the other hand, once the u-closure graph of $F^{-t}(y)$ has been built, it is not difficult to see that the number of t-th ancestors $x \in F^{-t}(y)$ having least period hp are $h \cdot |\mathcal{C}^u_h(f)|$, where by Corollary 1 $h = \prod_{i=1}^{t} h_i$, with $h_i \in \{1, \cdots, q^{d-1}\}$ for all $i \in \{1, \cdots, t\}$.

10.2 LINEAR CA AND LINEAR RECURRING SEQUENCES

Lemma 9 suggests that both Problems 1 and 2 are easier to analyze in the bipermutive context, since BCA do not feature paths with preperiods in the u-closure graph. In this section, we narrow our attention to the class of LBCA, showing that in this case further information about the periods of preimages can be obtained. In particular, we characterize the preimages of LBCA as a particular kind of *concatenated linear recurring sequences*, and determine the corresponding *characteristic polynomials*.

10.2.1 *LBCA Preimages and Concatenated LRS*

Let $F : \mathbb{F}_q^{\mathbb{Z}} \to \mathbb{F}_q^{\mathbb{Z}}$ be a LBCA of diameter d, offset $\omega = 0$ and local rule $f : \mathbb{F}_q^d \to \mathbb{F}_q$ defined by a vector $(c_0, \cdots, c_{d-1}) \in \mathbb{F}_q^d$, where $c_0 \neq 0$ and $c_{d-1} \neq 0$. Given $x \in \mathbb{F}_q^d$ and $y = f(x)$, it holds that:

$$\begin{aligned} y &= c_0x_0 + c_1x_1 + \cdots + c_{d-2}x_{d-2} + c_{d-1}x_{d-1} \\ x_{d-1} &= c_{d-1}^{-1}(-c_0x_0 - c_1x_1 - \cdots - c_{d-2}x_{d-2} + y) \ . \end{aligned}$$

Setting $e = c_{d-1}^{-1}$ and $a_i = -e \cdot c_i$ for all $i \in \{0, \cdots, d-2\}$, we obtain

$$x_{d-1} = a_0x_0 + a_1x_1 + \cdots + a_{d-2}x_{d-2} + ey \ . \tag{90}$$

Equation (90) defines the inverse permutation $f_{R,z}^{-1}$ of $f_{R,z} : \mathbb{F}_q \to \mathbb{F}_q$, obtained by fixing the first $d-1$ coordinates of f to the values of $z = (x_0, \cdots, x_{d-2})$. Hence, given a configuration $y \in \mathbb{F}_q^{\mathbb{Z}}$ and the $(d-1)$–cell block $x_{[0,d-2]} \in \mathbb{F}_q^{d-1}$ in a preimage $x \in F^{-1}(y)$, for all $n > d-1$ it results that:

$$x_n = a_0x_{n-(d-1)} + a_1x_{n-d} + \cdots + a_{d-2}x_{n-1} + ey_n \ , \tag{91}$$

and by setting $k = d-1$ and $v_n = y_n$ for all $n \in \mathbb{N}$, Equation (91) can be rewritten as

$$x_{n+k} = a_0x_n + a_1x_{n+1} + \cdots + a_{k-1}x_{n+k-1} + ev_n \ . \tag{92}$$

Equation (92) reminds the definition of a linear recurring sequence of order $k = d-1$, with the exception of term ev_n. However, if y is a spatially periodic configuration of period p then it is possible to describe the sequence $v = v_0, v_1, \cdots$ as a linear recurring sequence of order $l \leqslant p$ defined by

$$v_{n+l} = b_0v_n + b_1v_{n+1} + \cdots + b_{l-1}v_{n+l-1} \ , \tag{93}$$

where $b_i \in \mathbb{F}_q$ for all $i \in \{0, \cdots, l-1\}$, and the initial terms of the sequence are $v_0 = y_0$, $v_1 = y_1$, $\cdots$, $v_{l-1} = y_{l-1}$. In the worst case, the LRS v will have order $l = p$, and it will be generated by the trivial LFSR which cyclically shifts a word of length p.

As a consequence, preimage $x \in F^{-1}(y)$ is a linear recurring sequence of a special kind, where x_{n+k} is determined not only by the previous $k = d-1$ terms, but it is also "disturbed" by the LRS v. In particular, we define x as the *concatenation* of sequences s and v, which we denote by $x = s \leftsquigarrow v$, where $s = s_0, s_1, \cdots$ is the k-th order LRS satisfying the recurrence

$$s_{n+k} = a_0 s_n + a_1 s_{n+1} + \cdots + a_{k-1} s_{n+k-1} \ , \tag{94}$$

and whose initial values are $s_0 = x_0$, $s_1 = x_1$, $\cdots$, $s_{k-1} = x_{k-1}$.

Equivalently, a preimage $x \in F^{-1}(y)$ is generated by a LFSR of order $k = d-1$ where the feedback is summed with the output of an LFSR of order l multiplied by $e = c_{d-1}^{-1}$, which produces the sequence v. Similarly to concatenated LRS, we call this system a *concatenation* of LFSR. Figure 29 depicts the block diagram of this concatenation.

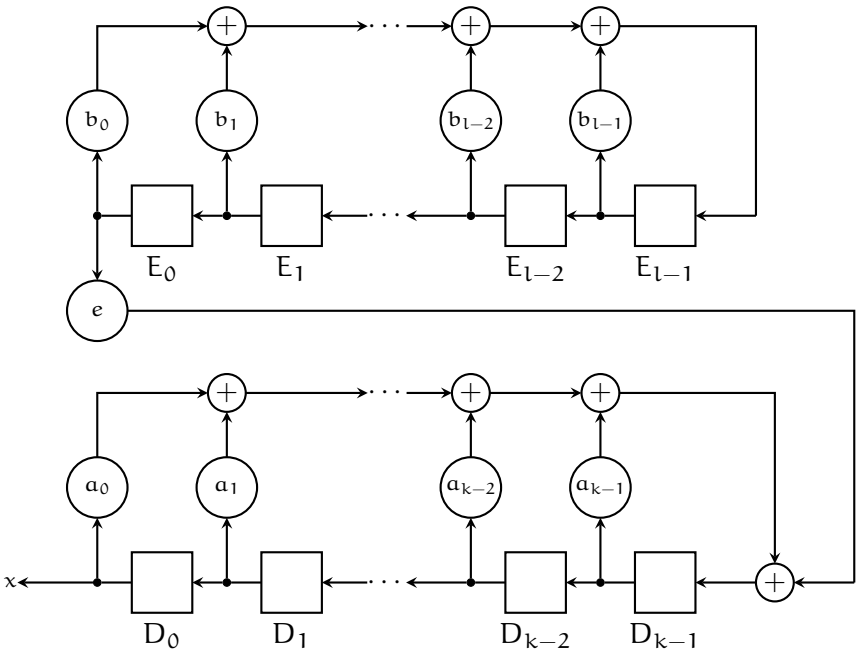


Figure 29: Diagram of two concatenated LFSR.

In conclusion, we have shown that the periods of the preimages $x \in F^{-1}(y)$ are equivalent to the periods of the concatenated LRS generated by the LFSR in Figure 29, where the disturbing LFSR is initialized with the values $y_0, \cdots, y_{l-1}$. In particular, since multiplying the terms of a LRS by a constant does not change its period, in what follows we will assume $e = 1$.

10.2.2 *Sum Decomposition of Concatenated LRS*

In order to study the period of the concatenated linear recurring sequence $s \leftsquigarrow v$ giving rise to preimage $x \in F^{-1}(y)$, we first prove that it can be decomposed into the *sum* of two LRS: namely, sequence s

and the 0-*concatenation* $u = s \leftsquigarrow_0 v$ satisfying the same recurrence Equation (92), but whose k initial terms $u_0, \cdots, u_{k-1}$ are set to 0.

Theorem 18. *Let* $s = s_0, s_1, \cdots$ *and* $v = v_0, v_1, \cdots$ *be the LRS respectively satisfying Equations* (94) *and* (93)*, with* $s_0 = x_0, \cdots, s_{k-1} = x_{k-1}$ *and* $v_0 = y_0, \cdots, v_{l-1} = y_{l-1}$*. Further, let* $x = s \rightsquigarrow v$ *be the concatenation of* s *and* v *defined by Equation* (92)*, where* $e = 1$*, and let* $u = s \leftsquigarrow_0 v$ *be the* 0*-concatenation of sequences* s *and* v*, where* $u_0 = u_1 = \cdots = u_{k-1} = 0$*. Then,* $x_n = s_n + u_n$ *for all* $n \in \mathbb{N}$*.*

Proof. Since $u_0 = \cdots = u_{k-1} = 0$, for all $n \in \{0, \cdots, k-1\}$ it holds

$$s_n + u_n = s_n + 0 = x_n \ .$$

Therefore, it remains to prove $x_n = s_n + u_n$ for all $n \geqslant k$. We proceed by induction on n. For $n = k$, we have

$$\begin{aligned} s_k + u_k &= a_0 s_0 + \cdots + a_{k-1} s_{k-1} + \\ &\quad + a_0 u_0 + \cdots + a_{k-1} u_{k-1} + v_0 \\ &= a_0 x_0 + \cdots + a_{k-1} x_{k-1} + v_0 = x_k \ . \end{aligned}$$

For the induction step we assume $s_n + u_n = x_n$ for all n in the range $\{k, \cdots, m\}$. For $n = m+1$, the sum $s_{m+1} + u_{m+1}$ is equal to:

$$\begin{aligned} s_{m+1} + u_{m+1} &= a_0 s_{m-k+1} + \cdots + a_{k-1} s_m + \\ &\quad + a_0 u_{m-k+1} + \cdots + a_{k-1} u_m + v_{m-k+1} \\ &= a_0 (s_{m-k+1} + u_{m-k+1}) + \cdots + \\ &\quad + a_{k-1} (s_m + u_m) + v_{m-k+1} \ . \end{aligned} \tag{95}$$

By induction hypothesis, $s_{m-k+i} + u_{m-k+i} = x_{m-k+i}$ for each index $i \in \{1, \cdots, k\}$. Hence, Equation (95) can be rewritten as

$$s_{m+1} + u_{m+1} = a_0 x_{m-k+1} + \cdots + a_{k-1} x_m + v_{m-k+1} = x_{m+1} \ .$$

□

10.2.3 *Characteristic Polynomial of Concatenated LRS*

Theorem 18 tells us that a preimage $x \in F^{-1}(y)$ can be generated by the sum of two LRS: the LRS generated by the concatenated LFSR of Figure 29, where the disturbed LFSR is initialized to zero, and the LRS produced by the *non-disturbed* LFSR, that is, the leftmost LFSR in Figure 29 initialized to the values $x_0, \cdots, x_{k-1}$ without the feedback from the rightmost LFSR.

We now show that this sum decomposition allows one to determine a characteristic polynomial of the concatenated sequence $x = s \rightsquigarrow v$. To this end, we first need a result proved by Chassé[37], which concerns the generating function of the 0-concatenation $u = s \leftsquigarrow_0 v$. The

proof stands on the observation that for all $n \in \mathbb{N}$, the n-th term of u is given by the linear combination $\sum_{i=0}^{n-1} A_n^{(i)} \cdot v_i$, where the terms $A_n^{(i)}$ depend only on the coefficients a_j which define Equation (94). In particular, we will need the values of $A_n^{(0)}$ for $n \geqslant 0$, which can be computed by the following recurrence equation:

$$A_n^{(0)} = \begin{cases} \sum_{j=0}^{k-1} a_j A_{n-k+j}^{(0)} \ , \text{ if } n > 1 \\ 1 \ , \text{ if } n = 1 \\ 0 \ , \text{ if } n = 0 \end{cases} \tag{96}$$

where $k = d - 1$ and $A_{n-k+j}^{(0)} = 0$ if $n - k + j < 0$. Using our notation and terminology, Chassé's result can thus be stated as follows:

Lemma 10. *Let $u = s \leftsquigarrow_0 v$ be the 0-concatenation of the LRS s and v defined in Theorem 18, and let $V(x)$ be the generating function of v. Denoting by $\mathcal{A}(x)$ the generating function of the sequence $A = \{A_{n+1}^{(0)}\}_{n\in\mathbb{N}}$, the generating function of u is equal to*

$$U(x) = x \cdot \mathcal{A}(x) \cdot V(x) \ . \tag{97}$$

Moreover, if $a(x) \in \mathbb{F}_q[x]$ is the characteristic polynomial of the sequence s associated to the recurrence equation (94), then $a(x)$ is also a characteristic polynomial of A.

We now prove that the characteristic polynomial of $s \leftsquigarrow v$ is the product of the characteristic polynomials of s and v.

Theorem 19. *Let $s \leftsquigarrow v$ be the concatenation of LRS s and v defined by Equation (92) with $e = 1$, and let $a(x), b(x) \in \mathbb{F}_q[x]$ be the characteristic polynomials of s and v, respectively associated to the linear recurring sequences respectively defined by Equation (94) and (93). Then, $a(x) \cdot b(x)$ is a characteristic polynomial of $s \leftsquigarrow v$.*

Proof. By Theorem 18 the concatenation of LRS s and v can be written as $s \leftsquigarrow v = s + u$, where $u = s \leftsquigarrow_0 v$ is the 0-concatenation associated to $s \leftsquigarrow v$. By applying the fundamental identity of formal power series (Equation (16)) and Lemma 10, the following equalities hold:

$$S(x) = \frac{g_s(x)}{a^*(x)} \tag{98}$$

$$U(x) = \frac{x \cdot g_A(x) \cdot g_v(x)}{a^*(x) \cdot b^*(x)} \ , \tag{99}$$

where $g_s(x)$, $g_A(x)$ and $g_v(x)$ are polynomials whose coefficients are computed according to the numerator in the right hand side of Equation (16). Hence, the generating function of $s \leftsquigarrow v$ is:

$$G(x) = S(x) + U(x) = \frac{g_s(x) \cdot b^*(x) + x \cdot g_A(x) \cdot g_v(x)}{a^*(x) \cdot b^*(x)} \ . \tag{100}$$

By applying again the fundamental identity of formal power series to Equation (100), we deduce that the reciprocal of $c(x) = a^*(x) \cdot b^*(x)$ is a characteristic polynomial of $s \rightsquigarrow v$. Denoting by k and l the degrees of $a(x)$ and $b(x)$ respectively, it follows that $c(x) = x^{k+l} \cdot a(1/x) \cdot b(1/x)$, and thus the reciprocal of $c(x)$ is

$$c^*(x) = x^{k+l} \cdot \frac{1}{x^{k+l}} \cdot a(x) \cdot b(x) = a(x) \cdot b(x) \ . \tag{101}$$

Therefore, $a(x) \cdot b(x)$ is a characteristic polynomial of $s \rightsquigarrow v$. □

Theorem (19) thus gives a characteristic polynomial for all preimages $x \in F^{-1}(y)$ of a spatially periodic configuration $y \in \mathbb{F}_q^{\mathbb{Z}}$. As a matter of fact, the polynomials $a(x)$ and $b(x)$ do not depend on the particular value of the block $x_{[0,d-2]}$, but only on the local rule f and on configuration y, respectively. From the LFSR point of view, this means that a preimage $x \in F^{-1}(y)$ can be generated by a single LFSR implementing the $(k+l)$-th order recurrence equation

$$\sigma_{n+k+l} = c_0\sigma_n + c_1\sigma_{n+1} + \cdots + c_{k+l-1}\sigma_{n+k+l-1} \ , \tag{102}$$

where for all $\mu \in \{0, \cdots, k+l-1\}$ the term c_μ is the μ-th convolution coefficient in the product $a(x) \cdot b(x)$ given by

$$c_\mu = \sum_{i+j=\mu} a_i b_j, \text{ for } i \in \{0, \cdots, k\} \text{ and } j \in \{0, \cdots, l\} \ . \tag{103}$$

Additionally, the first $k = d-1$ initial terms $\sigma_0, \cdots, \sigma_{k-1}$ in Equation (102) are initialized to the values in $x_{[0,d-2]}$, while the remaining l ones are obtained using the recurrence equation (92). Hence, by applying the fundamental identity of formal power series, the numerator of Equation (100) can also be expressed as:

$$g(x) = -\sum_{j=0}^{k-1} \sum_{i=0}^{j} c_{i+k-j}\sigma_i x^j \ . \tag{104}$$

As in the case of Lemmas 6 and 8, Theorem 19 can be easily extended to the t-th iterate F^t for any $t > 1$. In this case, a t-th ancestor $x \in F^{-t}(y)$ can be expressed by the following sequence of concatenated LRS:

$$x = s(t) \rightsquigarrow s(t-1) \rightsquigarrow \cdots \rightsquigarrow s(1) \rightsquigarrow v \ , \tag{105}$$

where $s(i)$ belongs to the family of LRS $S(a(x))$ for all $i \in \{1, \cdots, t\}$. In other words, the t-th ancestor $x \in F^{-t}(y)$ is obtained by concatenating t sequences generated by the characteristic polynomial $a(x)$ of the CA local rule, which are in turn concatenated with configuration y. Notice that preimage computation, in this case, can be carried out by a *cascade* of concatenated LFSR, where the leftmost t ones all

have the same characteristic polynomial but possibly different initialization values, while the rightmost one generates v.

Consequently, by iteratively applying Theorem 19, we obtain that the characteristic polynomial of $x \in F^{-t}(y)$ is

$$c(x) = a(x)^t \cdot b(x) \ . \tag{106}$$

10.3 APPLICATIONS TO PERIODS COMPUTATION, MULTIPLICITIES COUNT AND FINITE RINGS ALPHABETS

To summarize the results discussed so far, in this section we explore the applications of the equivalence between LBCA preimages and CLRS presented in Section 10.2, starting from the most specific one and then generalizing. Specifically, in Section 10.3.1 we describe an algorithm which, given as inputs a SPC y of a LBCA over $\mathbb{F}_q$ and a $(d-1)$–cell block of one of its preimages $x \in F^{-1}(y)$, computes the least period of x. On the other hand, Section 10.3.2 characterizes the multiplicities of the preimages of a SPC y in the particular case where the characteristic polynomial of the local rule is irreducible and does not factorize the polynomial of y. Finally, Section 10.3.3 generalizes the results presented in Section 10.2 to the case where the CA alphabet is a *finite ring*.

10.3.1 *Computing the Period of a Single Preimage*

We now present a high-level procedure to compute the period of a single preimage. Given a LBCA $F : \mathbb{F}_q^{\mathbb{Z}} \to \mathbb{F}_q^{\mathbb{Z}}$ defined by a local rule $f : \mathbb{F}_q^d \to \mathbb{F}_q$ of diameter d and offset $\omega = 0$, a spatially periodic configuration $y \in \mathbb{F}_q^{\mathbb{Z}}$ and a $(d-1)$–cell block $x_{[0,d-2]} \in \mathbb{F}_q^{d-1}$ of a preimage $x \in F^{-1}(y)$, the procedure can be described as follows:

1. Compute the minimal polynomial $b(x)$ of the linear recurring sequence v, where $v_n = y_n$ for all $n \in \mathbb{N}$.
2. Set the characteristic polynomial $a(x)$ associated to the inverse permutation $f_{R,z}^{-1}$ to $a(x) = x^k - a_{k-1}x^{k-1} - \cdots - a_0$, where $k = d-1$ and the coefficients a_i are those appearing in the recurrence equation (94).
3. Compute the polynomial $g(x)$ given by Equation (104), and set $h(x) = -g^*(x)$.
4. Determine the minimal polynomial of x by computing
$$m(x) = \frac{a(x) \cdot b(x)}{\gcd(a(x) \cdot b(x), h(x))} \ . \tag{107}$$
5. Compute the order of $m(x)$, and output it as the least period of preimage x.

For step 1, the minimal polynomial of v can be found using the *Berlekamp-Massey algorithm* [115], by giving as input to it the string composed by the first $2p$ elements of v, where p is the period of y (and hence the period of v as well). The time complexity of this algorithm is $O(p^2)$. Step 4 requires the computation of a greatest common divisor, which can be performed using the Euclidean division algorithm in $O(n^2)$ steps, where $n = \max\{\deg(a(x)b(x)), \deg(h(x))\}$. Finally, the order of $m(x)$ in step 5 can be determined by first factorizing the polynomial, for example by using *Berlekamp's algorithm* [12], which has a time complexity of $O(D^3)$ where D is the degree of $m(x)$, if the characteristic ρ of $\mathbb{F}_q$ is sufficiently small. Once the factorization of $m(x)$ is known, $\mathrm{ord}(m(x))$ can be computed using the following theorem proved in [107]:

Theorem 20. *Let $m(x) \in \mathbb{F}_q[x]$ be a polynomial of positive degree such that $m(0) \neq 0$. Let $m(x) = a \cdot \prod_{i=0}^{n} f_i(x)^{b_i}$ be the canonical factorization of $m(x)$, where $a \in \mathbb{F}_q$, $b_1, \cdots, b_n \in \mathbb{N}$ and $f_1(x), \cdots, f_n(x) \in \mathbb{F}_q[x]$ are distinct monic irreducible polynomials. Then $\mathrm{ord}(m(x)) = e\rho^t$, where ρ is the characteristic of $\mathbb{F}_q$, $e = \mathrm{lcm}(\mathrm{ord}(f_1(x)), \cdots, \mathrm{ord}(f_n(x)))$ and t is the smallest integer such that $\rho^t \geqslant \max\{b_1, \cdots, b_n\}$.*

Notice that Theorem 20 depends on the orders of the irreducible polynomials involved in the factorization of $m(x)$. A method to find the order of an irreducible polynomial that relies on the factorization of $q^D - 1$ is reported in [107]. There exist several factorization tables for numbers in this form, especially for small values of q (see [189]).

We now present a practical application of the procedure described above. The computations in the following example have been carried out with the computer algebra system MAGMA.

Example 1. *Let $F : \mathbb{F}_2^{\mathbb{Z}} \to \mathbb{F}_2^{\mathbb{Z}}$ be the LBCA with offset $\omega = 1$ and local rule 150, defined as $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$ for all $x \in \mathbb{F}_2^3$. Let $y \in \mathbb{F}_2^{\mathbb{Z}}$ be a spatially periodic configuration of least period $p = 4$ generated by the block $y_{[0,3]} = (0, 0, 1, 1)$, and let $x_{[0,1]} = (1, 0)$ be the initial 2-cell block of a preimage $x \in F^{-1}(y)$. Since $\omega = 1$, we have to shift y one place to the left, thus sequence v is generated by block $v_{[1,3]} = (0, 1, 1, 0)$. Feeding the string $(0, 1, 1, 0, 0, 1, 1, 0)$ to the Berlekamp-Massey algorithm yields the minimal polynomial $b(x) = x^3 + x^2 + x + 1$, while the characteristic polynomial associated to rule 150 is $a(x) = x^2 + x + 1$. Hence, it follows that $c(x) = a(x) \cdot b(x) = x^5 + x^3 + x^2 + 1$ is a characteristic polynomial of the preimage. Since the first 5 elements of preimage x are $1, 0, 1, 0, 0$, the initialization polynomial of Equation (104) is $g(x) = x^4 + x^3 + 1$, from which we deduce that $h(x) = x^4 + x + 1$. Considering that $h(x)$ is irreducible, the greatest common divisor of $c(x)$ and $f(x)$ is 1, and thus by Equation (107) $c(x)$ is also the minimal polynomial of the preimage. The factorization of $c(x)$ is $(x+1)^3(x^2+x+1)$, and the orders of $x+1$ and x^2+x+1 are respectively 1 and 3, from which it follows that the least common multiple e is 3. Finally, the smallest integer t such that $2^t \geqslant 3$ is $t = 2$. Therefore, by applying*

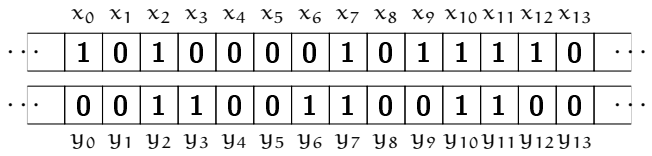


Figure 30: Block $x_{[0,11]}$ which generates preimage $x \in F^{-1}(y)$ under rule 150.

Theorem 20 the least period of preimage x *is* $\epsilon 2^t = 12$. *Figure 30 shows the actual value of the block* $x_{[0,11]}$ *which generates preimage* x.

The above procedure can be adapted to the case of t-th ancestors $x \in F^{-t}(y)$ by setting the characteristic polynomial in step 2 to $a(x)^t$, according to Equation (106). Clearly, at step 3 the computation of polynomial $g(x)$ defined in Equation (104) becomes more expensive, since the sequence σ of Equation (102) is now a $(kt + l)$-order LRS. Additionally, the complexity of step 5 grows exponentially in the degree D of the minimal polynomial $m(x)$ computed at step 4, since it depends on the factorization of $q^D - 1$.

10.3.2 *Periods Multiplicities*

As a further application of Theorem 19, we characterize the least periods of preimages with respect to the t-th iterate of LBCA in the special case where $a(x)$ is irreducible and relatively prime to $b(x)$.

Our characterization result, which is analogous to Theorem 6, is the following:

Theorem 21. *Let* $F : \mathbb{F}_q^{\mathbb{Z}} \to \mathbb{F}_q^{\mathbb{Z}}$ *be a LBCA having local rule* $f : \mathbb{F}_q^d \to \mathbb{F}_q$, *and let* $a(x) = x^k - a_{k-1}x^{k-1} - \cdots - a_0$ *be the characteristic polynomial associated to* f, *where* $k = d - 1$, *and* $\mathrm{ord}(a(x)) = \epsilon$. *For* $t \in \mathbb{N}$, *let* $s \in \mathbb{N}$ *be the smallest integer such that* $\rho^s \geqslant t$, *where* ρ *is the characteristic of* $\mathbb{F}_q$. *Further, let* $y \in \mathbb{F}_q^{\mathbb{Z}}$ *be a SPC of least period* $p \in \mathbb{N}$, *and let* $b(x)$ *be the minimal polynomial of sequence* ν *defined as* $\nu_n = y_{n+r}$ *for all* $n \in \mathbb{N}$. *If* $a(x)$ *is irreducible and does not divide* $b(x)$, *then:*

- *If* $t = 1$, $F^{-t}(y)$ *is composed of one sequence with least period* p *and* $q^k - 1$ *sequences with least period* $\mathrm{lcm}(\epsilon, p)$.
- *If* $t \geqslant 2$, $F^{-t}(y)$ *also contains* $q^{k\rho^j} - q^{k\rho^{j-1}}$ *sequences with least period* $\mathrm{lcm}(\epsilon\rho^j, p)$ *for* $j \in \{1, \cdots, s-1\}$, *and* $q^{kt} - q^{k\rho^s}$ *sequences with least period* $\mathrm{lcm}(\epsilon\rho^s, p)$.

Proof. By Equation (106), recall that $a(x)^t \cdot b(x)$ is a characteristic polynomial for all $x \in F^{-t}(y)$, which means that

$$F^{-t}(y) \subseteq S(a(x)^t \cdot b(x)) \ . \tag{108}$$

Since $a(x)$ and $b(x)$ are coprime, it holds that

$$\mathrm{lcm}(a(x)^t, b(x)) = a(x)^t \cdot b(x) \ . \tag{109}$$

Consequently, on account of Theorem 4 and Equation (109), the following equality holds:

$$S(a(x)^t \cdot b(x)) = S(a(x)^t) + S(b(x)) \ . \tag{110}$$

Thus, by (108) and (110) we conclude that $F^{-t}(y) = S(a(x)) + v$, i. e. the set of preimages of y under F^{-t} is a coset of the vector space $S(a(x)^t) + S(b(x))$. In particular, $F^{-t}(y)$ is obtained by forming all possible sums $u + v$ for $u \in S(a(x))$. Since $a(x)$ and $b(x)$ are coprime, Theorem 5 states that the least period of $u + v$ is $\mathrm{lcm}(l, p)$, where l is the least period of u. Finally, since $a(x)$ is irreducible, Theorem 6 characterizes the possible values of l and the corresponding numbers of sequences in $S(a(x)^t)$ attaining those values of l as least period, thus concluding the proof. □

10.3.3 *LBCA over Finite Rings Alphabets*

In this section, we assume that $\Sigma = \mathbb{Z}_m$, where $\mathbb{Z}_m$ is the finite ring of residue classes modulo $m \in \mathbb{N}$. A CA $F : \mathbb{Z}_m^{\mathbb{Z}} \to \mathbb{Z}_m^{\mathbb{Z}}$ is linear and bipermutive if and only if the coefficients c_0 and c_{d-1} of its local rule are invertible over $\mathbb{Z}_m$, i. e. $\gcd(c_0, m) = \gcd(c_{d-1}, m) = 1$.

Let us first consider the case where $m = q_1 q_2$ with q_1 and q_2 coprime. In [33], the authors showed that a LBCA $F : \mathbb{Z}_m^{\mathbb{Z}} \to \mathbb{Z}_m^{\mathbb{Z}}$ is *conjugated* to the function $G : \mathbb{Z}_{q_1}^{\mathbb{Z}} \times \mathbb{Z}_{q_2}^{\mathbb{Z}} \to \mathbb{Z}_{q_1}^{\mathbb{Z}} \times \mathbb{Z}_{q_2}^{\mathbb{Z}}$, which is defined for all $(x_1, x_2) \in \mathbb{Z}_{q_1}^{\mathbb{Z}} \times \mathbb{Z}_{q_2}^{\mathbb{Z}}$ as

$$G(x_1, x_2) = (F_{q_1}(x_1), F_{q_2}(x_2)) \ , \tag{111}$$

where F_{q_1} and F_{q_2} denote the application of rule F respectively reduced modulo q_1 and q_2. The homomorphism which maps a configuration $x \in \mathbb{Z}_m^{\mathbb{Z}}$ to its pair of *factor configurations* $(x_1, x_2) \in \mathbb{Z}_{q_1}^{\mathbb{Z}} \times \mathbb{Z}_{q_2}^{\mathbb{Z}}$ is defined as

$$\psi(x) = ([x]_{q_1}, [x]_{q_2}) \ , \tag{112}$$

where $[x]_{q_1}$ and $[x]_{q_2}$ respectively denote componentwise reduction modulo q_1 and q_2 of configuration x. The inverse homomorphism which recomposes a pair of configurations $(x_1, x_2) \in \mathbb{Z}_{q_1}^{\mathbb{Z}} \times \mathbb{Z}_{q_2}^{\mathbb{Z}}$ into a configuration $x \in \mathbb{Z}_m^{\mathbb{Z}}$ is defined as

$$\psi^{-1}(x_1, x_2) = x_2 + q_2[(x_1 - x_2)\hat{q}_2]_{q_1} \ , \tag{113}$$

where addition and subtraction are performed componentwise, and $\hat{q}_2$ is the multiplicative inverse of q_2 over $\mathbb{Z}_{q_1}$. Notice that $\hat{q}_2$ exists since $\gcd(q_1, q_2) = 1$.

The conjugacy can be extended to any $m \in \mathbb{N}$ as follows. First, let $m = \prod_{i=1}^{s} p_i^{\alpha_i}$ be the prime power factorization of m, and let $q_i = p_i^{\alpha_i}$ for all $i \in \{1, \cdots, s\}$. It follows that $\gcd(q_i, q_j) = 1$ for all $i \neq j$, since p_i and p_j are distinct prime numbers. The homomorphism $\psi_s : \mathbb{Z}_m^{\mathbb{Z}} \to \mathbb{Z}_{q_1}^{\mathbb{Z}} \times \cdots \times \mathbb{Z}_{q_s}^{\mathbb{Z}}$ is defined for all $x \in \mathbb{Z}_m^{\mathbb{Z}}$ as:

$$\psi_s(x) = ([x]_{q_1}, \cdots, [x]_{q_s}) \ . \tag{114}$$

For the inverse homomorphism, observe that $q_1, \cdots, q_s$ induce two sequences of rings $\{R_2, \cdots, R_s\}$ and $\{Q_2, \cdots, Q_s\}$, where R_j and Q_j are defined for $j \in \{2, \cdots, s\}$ as:

$$R_j = \mathbb{Z}_{q_1}^{\mathbb{Z}} \times \cdots \times \mathbb{Z}_{q_j}^{\mathbb{Z}} \ , \tag{115}$$

$$Q_j = \mathbb{Z}_{m_j}^{\mathbb{Z}}, \ \ m_j = \prod_{i=1}^{j} q_i \ . \tag{116}$$

Likewise, $q_1, \cdots, q_s$ induce a sequence of mappings $\{\psi_2^{-1}, \cdots, \psi_s^{-1}\}$ where for $j \in \{2, \cdots, s\}$ the inverse homomorphism $\psi_j^{-1} : R_j \to Q_j$ is defined for all $(x_1, \cdots, x_j) \in R_j$ as follows:

$$\psi_j^{-1}(x_1, \cdots, x_j) = \begin{cases} \psi^{-1}(x_1, x_2), & \text{if } j = 2 \\ \psi^{-1}(\psi_{j-1}^{-1}(x_1, \cdots, x_{j-1}), x_j), & \text{if } j > 2 \end{cases} \tag{117}$$

The following theorem shows how to compute the least periods of the preimages of a spatially periodic configuration under a linear and bipermutive CA $F : \mathbb{Z}_m^{\mathbb{Z}} \to \mathbb{Z}_m^{\mathbb{Z}}$.

Theorem 22. *Let* $m = \prod_{i=1}^{s} q_i$ *be a positive integer where* $q_i = \rho_i^{\alpha_i}$ *with* ρ_i *prime and* $\alpha_i \geqslant 1$ *for all* $i \in \{1, \cdots, s\}$. *Let* $F : \mathbb{Z}_m^{\mathbb{Z}} \to \mathbb{Z}_m^{\mathbb{Z}}$ *be a linear bipermutive CA, and let* $y \in \mathbb{Z}_m^{\mathbb{Z}}$ *be a spatially periodic configuration having least period* $p \in \mathbb{N}$, *with* $p_1, \cdots, p_s \in \mathbb{N}$ *respectively being the least periods of the factor configurations* $y_1 = [y]_{q_1}, \cdots, y_s = [y]_{q_s}$. *Then, given a preimage* $x \in F^{-1}(y)$, *the least period of* x *is* $k = \operatorname{lcm}(k_1, \cdots, k_s)$, *where* $k_i = h_i p_i$ *and* $h_i \in \{1, \cdots, q_i^{d-1}\}$ *for all* $i \in \{1, \cdots, s\}$.

Proof. We prove only the case $m = q_1 q_2$, the general case following by induction on the values q_i. Since F is linear and bipermutive, it follows that F is conjugated to the product CA G of Equation (111) through the isomorphism defined in Equations (112) and (113). Thus,

$$F^{-1}(y) = F^{-1}(\psi^{-1}(y_1, y_2)) = \psi^{-1}(G^{-1}(y_1, y_2)) \ .$$

As a consequence, the least period of $x \in F^{-1}(y)$ equals the least period of $\psi(x) = (x_1, x_2) \in G^{-1}(y_1, y_2)$.

Remark that $P \in \mathbb{N}$ is a period of (x_1, x_2) if and only if P is a period of both x_1 and x_2. By Lemma 6, x_1 and x_2 have least period $k_1 = h_1 p_1$ and $k_2 = h_2 p_2$ respectively, with $h_1 \in \{1, \cdots, q_1^{d-1}\}$ and $h_2 \in \{1, \cdots, q_2^{d-1}\}$. Since $k = \operatorname{lcm}(k_1, k_2)$ is a common multiple of k_1 and k_2, it follows that k is a period of both x_1 and x_2, and thus it is a period of (x_1, x_2). Let us now suppose that k is not the least period of (x_1, x_2), i. e. there exists $k' < k$ such that $\sigma^{k'}(x_1, x_2) = (x_1, x_2)$. From the discussion above, it follows that k' is a period of both x_1 and x_2 as well, and thus k' is a common multiple of k_1 and k_2, contradicting the fact that $k = \operatorname{lcm}(x_1, x_2)$. □

As a final remark, observe that if ρ is prime then the ring of residue classes $\mathbb{Z}_\rho$ is a finite field. Consequently, if m has a square-free factorization $m = \prod_{i=1}^{s} \rho_i^{\alpha_i}$ with $\alpha_i = 1$ for all $i \in \{1, \cdots, s\}$, and $F : \mathbb{Z}_m^{\mathbb{Z}} \to \mathbb{Z}_m^{\mathbb{Z}}$ is a LBCA over $\mathbb{Z}_m$, the least periods of the t-th ancestors $x \in F^{-t}(y)$ can be characterized by first finding the least periods of the factor preimages $[x]_{\rho_1}, \cdots, [x]_{\rho_s}$ using Theorem 21, and then by computing their least common multiple according to Theorem 22.

10.4 CONCLUSIONS

In this work, we studied the relation between the periods of spatially periodic configurations of surjective CA and the periods of their preimages. In the generic surjective case the periods of preimages are multiples of the periods of their respective images. Starting from this fact, we introduced a graph-theoretic method based on the de Bruijn representation of CA that allows one to compute the least periods of preimages and their multiplicities. Successively, by focusing on the linear and bipermutive case, we showed that every LBCA preimage can be characterized as a concatenated LRS, whose characteristic polynomial is the product of the characteristic polynomials which are associated to the component sequences. From this result, we derived an algorithm to compute the least period of a single LBCA preimage and we characterized the periods of all preimages along with their multiplicities, in the case where the characteristic polynomial of the local rule is irreducible. We finally showed how to generalize these results to LBCA defined over the finite ring $\mathbb{Z}_m$ as state alphabet.

S-BOXES BASED ON CELLULAR AUTOMATA

11

As we remarked in Chapter 7, one can see that most of the literature pertaining the applications of CA in cryptography is centered on the design of stream ciphers and pseudorandom number generators. This can be explained by observing that even though Wolfram's PRNG was shown to be insecure (due to the weaknesses of rule 30), its overall structure is quite simple. This lead some researchers (see for instance [114, 106, 63]) to focus on the search of CA local rules having good cryptographic profiles in order to thwart the attacks discovered by Meier and Staffelbach [118] and Koc and Apohan [100], but retaining Wolfram's overall design of CA pseudorandom generator.

On the other hand, the design of S-boxes based on CA for block ciphers is a research topic which has received relatively little attention in the literature. This could be the reason why, at least as far as our knowledge goes, there are comparatively fewer works concerning the cryptographic properties of CA global rules as opposed to CA local rules. As we noted in Section 7.2, one remarkable exception in this regard is [52], where the authors analyzed the propagation and correlation characteristics of a CA equipped with rule 210 in Wolfram's numbering convention, denoted as χ. Interestingly, rule χ is an example of S-box based on CA which is employed in real-world applications, since it is the only nonlinear component of the KECCAK sponge construction [15]. Moreover, the rotation symmetric S-boxes (see Section 7.3) can be considered as CA-based S-boxes, since they are basically defined by a CA whose diameter equals the cellular array size. Even in this case, however, little work has been carried out to analyze their cryptographic properties in a systematic way.

The aim of this chapter is thus to undertake an investigation of the cryptographic properties of CA global rules by considering them as vectorial Boolean functions, and to relate them to the properties of the underlying local rules. To this end, we consider criteria that are relevant for the design of S-boxes in block ciphers, i. e. balancedness, algebraic degree, nonlinearity and differential uniformity, leaving resiliency and its connection with linear codes for the next chapter.

We carry out our analysis of CA-based S-boxes both from a theoretical and an experimental point of view. For the theoretical part, we first observe that the algebraic degree of the global rule of a CA equals the algebraic degree of its local rule, leveraging on the fact that the coordinate functions of a CA correspond to its local rule applied to different neighborhoods. We then prove two upper bounds on the

nonlinearity and differential uniformity of S-boxes based on CA, both for no boundary and periodic boundary conditions. Next, we compare our nonlinearity bound with the Sidelnikov-Chabaud-Vaudenay bound for generic S-boxes reported in Section 5.5.3, observing that rule χ is optimal with respect to the former.

For the experimental part we employ *Genetic Programming* (GP) to construct CA-based S-boxes with good cryptographic properties. In particular, we evolve Boolean functions in the form of GP trees, which are then used as CA local rules where the diameter equals the length of the cellular array. As already discussed, this setting corresponds to the case of rotation symmetric S-boxes, and finds its motivation in the nonlinearity bound proved in the theoretical part. The problem instances considered in our experiments ranges from 4×4 to 8×8 sizes, and the results show that our GP algorithm manages to find optimal S-boxes up to size 7×7. We then report an exhaustive classification of all CA-based S-boxes from sizes 3×3 to 5×5 with respect to their bijectivity, nonlinearity and differential uniformity. Finally, we give a classification up to affine equivalence for the 3×3 and 4×4 cases, since for 5×5 the resulting number of equivalence classes is too large to be exhaustively checked.

The rest of the chapter is organized as follows. Section 11.1 is devoted to the analysis of the global rules of CA, focusing on their algebraic degree, nonlinearity and differential uniformity. Section 11.2 describes the structure of our GP algorithm and the experimental results obtained by evolving CA-based S-boxes of sizes up to 8×8, as well as an exhaustive classification up to size 5×5. Finally, Section 11.3 summarizes the contributions of the chapter.

11.1 CRYPTOGRAPHIC PROPERTIES OF CA GLOBAL RULES

In the rest of this chapter, we will consider finite CA defined over the binary alphabet $\Sigma = \mathbb{F}_2$, both with no boundary and periodic boundary conditions. In this section we investigate the cryptographic properties of such CA, starting from their algebraic degree. We then prove two upper bounds on the nonlinearity and differential uniformity achievable by these CA.

11.1.1 *Algebraic Degree*

The results reported in this section are for no boundary CA, but they can easily adapted to the PBCA case.

Recall from Remark 1 that, given a NBCA $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ with $m = n - d + 1$ and defined by a local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$, the coordinate function $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$ corresponds to local rule f applied on the neighborhood $\{i, \cdots, i + d - 1\}$ for all $i \in \{1, \cdots, n - d + 1\}$.

Since the algebraic degree of a (n, m)–function equals the maximal degree of its coordinate functions, we obtain the following result:

Lemma 11. *Let* $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ *be a NBCA with* $m = n - d + 1$ *defined by a local rule* $f : \mathbb{F}_2^d \to \mathbb{F}_2$. *Then, the degree of* F *equals the degree of* f.

Proof. For $k \in \{1, \cdots, m\}$, define $N_k = \{k, \cdots, k + d - 1\}$ and let us denote by $\mathcal{P}(N_k)$ the power set of N_k. Notice that $N_1 = N$, where N is the index set for the ANF of f. For all $I = \{I_1, \cdots, I_j\} \in \mathcal{P}(N)$, let us define the *shifted* subset of I as $\sigma_k(I) = \{I_1 + k - 1, \cdots, I_j + k - 1\}$, which ranges in the power set $\mathcal{P}(N_k)$. On the other hand, given an index set $L \in \mathcal{P}(N_k)$ one can recover the original subset $I \in \mathcal{P}(N)$ by computing $I = \sigma_{-k}(L) = \{L_1 - k + 1, \cdots, L_j - k + 1\}$. Then, by Equation (49) we have that

$$P_{f_k}(x) = \bigoplus_{L \in \mathcal{P}(N_k)} a_L \left(\prod_{l \in L} x_l \right) . \tag{118}$$

Since for every $L \in \mathcal{P}(N_k)$ there exists $I \in \mathcal{P}(N)$ such that $I = \sigma_{-k}(L)$, by Remark 1 it also follows that $a_L = a_I$, so we can rewrite (118) as:

$$P_{f_k}(x) = \bigoplus_{L \in \mathcal{P}(N_k)} a_I \left(\prod_{l \in L} x_l \right) \text{ , where } I = \sigma_{-k}(L). \tag{119}$$

Since the shifting operation does not change the cardinality of subsets, it follows that

$$max_{I \in \mathcal{P}(N)}\{|I| : a_I \neq 0\} = max_{L \in \mathcal{P}(N_k)}\{|L| : a_I \neq 0\} , \tag{120}$$

from which one obtains that $deg(f_k) = deg(f_1) = deg(f)$. □

11.1.2 *Bounds on Nonlinearity and Differential Uniformity*

We now show two upper bounds on the nonlinearity and differential uniformity of S-boxes defined by CA, relating them to the corresponding properties of the underlying local rules. To prove our results, we make use of the following theorem by Nyberg [129], concerning how the nonlinearity and the differential uniformity of an S-box are affected by adding a coordinate function while maintaining fixed the number of input variables.

Theorem 23. *Let* $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ *be a* (n, m)*–function defined by* m *coordinate functions* $f_1, \cdots, f_m : \mathbb{F}_2^n \to \mathbb{F}_2$. *Additionally, let* $g : \mathbb{F}_2^n \to \mathbb{F}_2$, *and define* $\tilde{F} : \mathbb{F}_2^n \to \mathbb{F}_2^{m+1}$ *as follows:*

$$\tilde{F}(x_1, \cdots, x_n) = (f_1(x_1, \cdots, x_n), \cdots, f_m(x_1, \cdots, x_n), g(x_1, \cdots, x_n)) . \tag{121}$$

Then, the following upper bounds hold:

$$N_F(\tilde{F}) \leqslant \min\{N_F(F), N_F(g)\} . \tag{122}$$

$$\frac{1}{2}\delta_F \leqslant \delta_{\tilde{F}} \leqslant \delta_F . \tag{123}$$

Consider now a CA (either with no boundary or periodic boundary conditions) with n cells and local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$. How do the nonlinearity and differential uniformity of F change by adding a new cell, thus obtaining a new CA $\tilde{F}$ of $n+1$ cells? Observe that Theorem 23 cannot be directly applied here, because we need to address the case where both a coordinate function *and* an input variable are added to the original CA. We first address this situation for generic S-boxes (i. e. not necessarily defined by a CA rule) in the following result:

Theorem 24. *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be an S-box defined by m coordinate functions $f_1, \cdots, f_m : \mathbb{F}_2^m \to \mathbb{F}_2$, and let $g : \mathbb{F}_2^{n+1} \to \mathbb{F}_2$ be a Boolean function defined on $n+1$ variables. Define $\tilde{F} : \mathbb{F}_2^{n+1} \to \mathbb{F}_2^{m+1}$ as follows:*

$$\tilde{F}(x_1, \cdots, x_n) = (f_1(x_1, \cdots, x_n), \cdots, f_m(x_1, \cdots, x_n), g(x_1, \cdots, x_{n+1})) \ . \tag{124}$$

Then, $\tilde{F}$ satisfies the following bounds:

$$N_{\tilde{F}} \leqslant \min\{2 \cdot N_F, N_g\} \ , \tag{125}$$

$$\delta_F \leqslant \min\{2 \cdot \delta_F, \delta_g\} \ . \tag{126}$$

Proof. We begin by addressing the bound on nonlinearity. We are going to analyze the Walsh-Hadamard transform of $\tilde{F}$ by classifying its component functions as follows:

(i) The $2^m - 1$ component functions that do not select the new coordinate g, i. e. those described by the vectors $\tilde{v} = (v, 0) \in \mathbb{F}_2^{m+1}$, where $v \in \mathbb{F}_2^{m*}$.

(ii) The single component function that just selects g, defined by the vector $(\underline{0}, 1)$ where $\underline{0} \in \mathbb{F}_2^m$.

(iii) Finally, the $2^m - 1$ component functions that select g and whose first m coordinates are not all zeros, which are defined by the vectors $\tilde{v} = (v, 1) \in \mathbb{F}_2^{m+1}$, where $v \in \mathbb{F}_2^{m*}$.

Consider the component functions of type (i). Let $\tilde{v} = (v, 0) \in \mathbb{F}_2^{m+1}$, where $v \in \mathbb{F}_2^{m*}$. Then, the Walsh-Hadamard transform of $\tilde{v} \cdot \tilde{F}$ computed on $\omega \in \mathbb{F}_2^{n+1}$ equals

$$\begin{aligned} W_{\tilde{v}\cdot\tilde{F}}(\tilde{\omega}) &= \sum_{\tilde{x} \in \mathbb{F}_2^{n+1}} (-1)^{\tilde{v}\cdot\tilde{F}(\tilde{x}) \,\oplus\, \tilde{\omega}\cdot\tilde{x}} = \\ &= \sum_{(x, x_{n+1}) \in \mathbb{F}_2^{n+1}} (-1)^{(v,0)\cdot(F(x), g(x_{n+1})) \,\oplus\, (\omega, \omega_{n+1})\cdot(x, x_{n+1})} = \\ &= \sum_{(x, x_{n+1}) \in \mathbb{F}_2^{n+1}} (-1)^{v\cdot F(\tilde{x}) \,\oplus\, \omega\cdot x} \cdot (-1)^{\omega_{n+1}\cdot x_{n+1}} \ . \end{aligned} \tag{127}$$

Let us rewrite the right hand side of Equation (127) by dividing the sum with respect to the value of x_{n+1}:

$$W_{\tilde{v}\cdot\tilde{F}}(\tilde{\omega}) = \sum_{(x,0)\in\mathbb{F}_2^{n+1}} (-1)^{v\cdot F(v)\oplus\omega\cdot x} + \sum_{(x,1)\in\mathbb{F}_2^{n+1}} (-1)^{v\cdot F(x)\oplus\omega\cdot x\oplus\omega_{n+1}} = \\ = \sum_{x\in\mathbb{F}_2^n} (-1)^{v\cdot F(x)\oplus\omega\cdot x} + (-1)^{\omega_{n+1}} \cdot \sum_{x\in\mathbb{F}_2^n} (-1)^{v\cdot F(x)\oplus\omega\cdot x} \ . \tag{128}$$

Notice that the two sums in Equation (128) correspond to the Walsh-Hadamard coefficient $W_{v\cdot F}(\omega)$. Thus, it holds that

$$W_{\tilde{v}\cdot\tilde{F}}(\tilde{\omega}) = \begin{cases} 0 & \text{, if } \omega_{n+1} = 0 \\ 2\cdot W_{v\cdot F}(\omega) & \text{, if } \omega_{n+1} = 1 \end{cases} \tag{129}$$

Hence, by Equation (129) we have that the linearity of $\tilde{F}$ will be at least twice the linearity of F, from which it follows that

$$N_{\tilde{F}} \leqslant 2\cdot N_F \ . \tag{130}$$

Let us now consider the component of type (ii), i. e. the one defined by $\tilde{v} = (0, 1)$. In this case, it is easy to see that $N_{\tilde{v}\cdot\tilde{F}} = N_g$, which yields

$$N_{\tilde{F}} \leqslant N_g \ . \tag{131}$$

Since the nonlinearity of $\tilde{F}$ is defined as the minimum nonlinearity among all its component functions, by combining Equations (130) and (131) we get

$$N_{\tilde{F}} \leqslant \min\{2\cdot N_F, N_g\} \ .$$

We finally address the differential uniformity bound. Given the vectors $\tilde{a} = (a, a_{n+1}) \in \mathbb{F}_2^{n+1}$ and $\tilde{b} = (b, b_{m+1}) \in \mathbb{F}_2^{m+1}$, the delta difference set of $\tilde{F}$ with respect to $\tilde{a}$ and $\tilde{b}$ is

$$D_{\tilde{F}}(a, b) = \{\tilde{x} = (x, x_{n+1}) \in \mathbb{F}_2^{n+1} : \tilde{F}(\tilde{x}\oplus\tilde{a})\oplus\tilde{F}(\tilde{x}) = \tilde{b}\} = \\ = \{\tilde{x} \in \mathbb{F}_2^{n+1} : [F(x\oplus a), g(\tilde{x}\oplus\tilde{a})]\oplus[F(x), g(\tilde{x})] = (b, b_{m+1})\} \ , \tag{132}$$

Where we can further rewrite the right hand side of (132) as:

$$\{\tilde{x} \in \mathbb{F}_2^{n+1} : [F(x\oplus a)\oplus F(x) = b] \wedge [g(\tilde{x}\oplus\tilde{a})\oplus g(\tilde{x}) = b_{m+1}]\} = \\ = \{\tilde{x} \in \mathbb{F}_2^{n+1} : [x \in D_F(a, b)] \wedge [\tilde{x} \in D_g(\tilde{a}, \tilde{b})]\} = \\ = \{\tilde{x} \in \mathbb{F}_2^{n+1} : x \in D_F(a, b)\} \cap \{(x, x_{n+1}) \in \mathbb{F}_2^{n+1} : \tilde{x} \in D_g(\tilde{a}, \tilde{b})\} = \\ = A \cap B \ . \tag{133}$$

From Equation (133) we have that $B = D_g(\tilde{a}, \tilde{b})$, and thus its cardinality is $|B| = \delta_g(\tilde{a}, \tilde{b})$. On the other hand, for set A we obtain

$|A| = 2 \cdot |D_F(a,b)| = 2 \cdot \delta_F(a,b)$, since the vectors $\tilde{x}$ in A are constructed by taking all vectors x belonging to $D_F(a,b)$ and by appending to their right a 0 and a 1. Consequently, it holds that

$$\delta_{\tilde{F}}(\tilde{a},\tilde{b}) = |A \cap B| \leqslant \min\{2 \cdot \delta_F(a,b), \delta_g(\tilde{a},\tilde{b})\} \ . \tag{134}$$

Finally, observe that one can construct the delta difference tables of maximum cardinality of $\tilde{F}$ by taking all possible intersections between the delta difference sets of maximum cardinality of F and g. Hence, the differential uniformity of $\tilde{F}$ satisfies

$$\delta_{\tilde{F}} \leqslant \min\{2 \cdot \delta_F, \delta_g\} \ . \tag{135}$$

□

Of course, the upper bounds given in Equation (125) are not tight. In fact, the component functions of type (iii) could yield a lower nonlinearity and differential uniformity than those featured by the components of types (i) and (ii) considered in the proof of Theorem 24.

Before turning our attention to the CA case, we still need one more preliminary result about how the nonlinearity and differential uniformity of a Boolean function change by adding dummy variables:

Lemma 12. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function with nonlinearity N_f and differential uniformity δ_f. Given $t \in \mathbb{N}$, define $\tilde{f} : \mathbb{F}_2^{n+t} \to \mathbb{F}_2$ for all $x \in \mathbb{F}_2^{n+t}$ as follows:*

$$\tilde{f}(x_1, \cdots, x_n, x_{n+1}, \cdots, x_{n+t}) = f(x_1, \cdots, x_n) \ . \tag{136}$$

Then, the following equalities hold:

$$N_{\tilde{f}} = 2^t \cdot N_f \ , \delta_{\tilde{f}} = 2^t \cdot \delta_f \ . \tag{137}$$

Proof. We proceed by induction on t.

For $t = 1$, one can easily see that $\tilde{f}$ is a special case of the vectorial function $\tilde{F}$ considered in Theorem 24 with $m = 1$, with the difference that no new output coordinates are added. Hence, the Walsh-Hadamard transform of $\tilde{f}$ is described by Equation (129), which yields $N_{\tilde{f}} = 2 \cdot N_f$. On the other hand, for $\tilde{a} = (a, a_{n+1}) \in \mathbb{F}_2^{n+1}$ and $b \in \mathbb{F}_2$, the delta difference set of $\tilde{f}$ is

$$D_{\tilde{f}}(\tilde{a}, b) = \{(x, x_{n+1}) \in \mathbb{F}_2^{n+1} : f(x \oplus a) \oplus f(x) = b\} \ ,$$

from which it follows that $\delta_{\tilde{f}}(\tilde{a}, b) = 2 \cdot \delta_f(a, b)$, and thus $\delta_{\tilde{f}} = 2 \cdot \delta_f$.

Next, assume $t > 1$ and consider the case $t + 1$, with $f' : \mathbb{F}_2^{n+t} \to \mathbb{F}_2$ indicating the function truncated at $n + t$ variables. Then, by induction hypothesis the following equalities are satisfied:

$$N_F(f') = 2^t \cdot N_F(f) \ ,$$
$$\delta_{f'} = 2^t \cdot \delta_f \ .$$

Similarly to the case $t = 1$, the Walsh coefficients of $\tilde{f} : \mathbb{F}_2^{n+t+1} \to \mathbb{F}_2$ are as in Equation (129), from which one obtains

$$N_{\tilde{f}} = 2 \cdot N_{f'} = 2 \cdot 2^t \cdot N_f = 2^{t+1} \cdot N_f \ . \tag{138}$$

Finally, the delta difference set $D_{\tilde{f}}(\tilde{a}, b)$ is again constructed by appending a 0 and a 1 to all vectors in $D_{f'}(a, b)$. Hence, the equality $\delta_{\tilde{f}}(\tilde{a}, b) = 2 \cdot \delta_{f'}(a, b)$ holds for all $\tilde{a} = (a, a_{n+t+1}) \in \mathbb{F}_2^{n+t+1}$, from which it finally follows that

$$\delta_{\tilde{f}} = 2 \cdot \delta_{f'} = 2 \cdot 2^t \delta_f = 2^{t+1} \cdot \delta_f \ .$$

□

Leveraging on the above results, we can now prove upper bounds on the nonlinearity and differential uniformity of S-boxes defined by CA, both in the no boundary and the periodic settings:

Theorem 25. *Let $f : \mathbb{F}_2^d \to \mathbb{F}_2$ and $n \geqslant d$. Then, the NBCA and PBCA $\tilde{F}$ with n input cells and local rule f satisfy the following bounds:*

$$N_{\tilde{F}} \leqslant 2^{n-d} \cdot N_f \tag{139}$$

$$\delta_{\tilde{F}} \leqslant 2^{n-d} \cdot \delta_f \ . \tag{140}$$

Proof. We first address the no boundary case. Let $\tilde{F} : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ be a NBCA with local rule f. We proceed by induction on $m = n - d + 1$.

For $m = 2$, we can apply Theorem 24 by setting function $F = f$ and $g : \mathbb{F}_2^{d+1} \to \mathbb{F}_2$ defined as follows:

$$g(x_1, x_2, \cdots, x_{d+1}) = f(x_2, \cdots, x_{d+1}) \ .$$

Thus, Theorem 24 yields that

$$N_{\tilde{F}} \leqslant \min\{2 \cdot N_f, N_g\} \ ,$$
$$\delta_{\tilde{f}} \leqslant \min\{2 \cdot \delta_f, \delta_g\} \ .$$

Additionally, by Lemma 12 we know that

$$N_g = 2 \cdot N_f \ ,$$
$$\delta_g = 2 \cdot \delta_f \ .$$

Since $m - 1 = n - d + 1 - 1 = 1$, the three bounds are satisfied in the base case.

Next, let us assume that $m > 2$ and consider the case $m + 1$, with $\tilde{F} : \mathbb{F}_2^{n+1} \to \mathbb{F}_2^{m+1}$ being the NBCA with $n + 1$ cells. In particular, define $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ as the NBCA with n cells, and $g : \mathbb{F}_2^{n+1} \to \mathbb{F}_2$ as $g(x_1, \cdots, x_{n+1}) = f(x_{n-d}, \cdots, x_{n+1})$. Again, Theorem 24 yields

$$N_{\tilde{F}} \leqslant \min\{2 \cdot N_F, N_g\} \ ,$$
$$\delta_{\tilde{F}} \leqslant \min\{2 \cdot \delta_F, \delta_g\} \ .$$

Figure 31: PBCA S-Boxes

CA size n \ Rule size d	3	4	5	6	7
3	2	–	–	–	–
4	4	4	–	–	–
5	8	8	12	–	–
6	16	16	24	24	–
7	32	32	48	48	56

Figure 32: Generic (n, n)-functions

$n \times n$	N_F
3×3	2
4×4	4
5×5	12
6×6	24
7×7	56

Figure 33: Best attainable nonlinearity for PBCA S-boxes and generic S-boxes up to $n = 7$ variables.

while by Lemma 12 we obtain

$$N_g = 2^m \cdot N_f \ ,$$
$$\delta_g = 2^m \cdot \delta_f \ .$$

Remarking that $m + 1 = n - d + 1$, by induction hypothesis we get

$$N_{\tilde{F}} \leqslant 2^m \cdot N_f = 2^{n-d} \cdot N_f \ ,$$
$$\delta_{\tilde{F}} \leqslant 2^m \cdot \delta_f = 2^{n-d} \cdot \delta_f \ ,$$

which concludes the proof for the NBCA case. Finally, for the periodic case it just suffices to observe that the PBCA is constructed by adding $n - d$ coordinate functions to the NBCA $\tilde{F}$ without extending the number of input variables, where the new coordinates always coincide with the local rule f applied on the rightmost and leftmost $d - 1$ cells. Hence, Theorem 23 can be applied here, from which one deduces that the same bounds for nonlinearity and differential uniformity also hold for the PBCA case. □

Tables 31 and 32 report the best nonlinearity values respectively reachable by PBCA as given by Theorem 25, for various values of d and n, and by generic bijective (n, n)-functions. Table 31 is lower triangular because the bound of Theorem 25 is meaningful only if $n \geqslant d$. For the maximum nonlinearity of N_f of the local rule we considered the quadratic bound, since it is known to be optimal for balanced Boolean functions of sizes up to $d = 7$ variables [30]. By comparing Tables 31 and 32 one can see that the only case where CA are able to reach the same best values as generic (n, n)-functions is when $d = n$, i. e. the rotation symmetric case which corresponds to the diagonal of Table 31. This also explains from a theoretical point of view why the nonlinearity of the CA χ used in Keccak is suboptimal with respect to the Sidelnikov-Chabaud-Vaudenay bound, since the neighborhood size of the rule is $d = 3$ while $n = 5$. Rule χ is, however, optimal with respect to the nonlinearity bound given in Theorem 25.

11.2 HEURISTIC DESIGN OF CA-BASED S-BOXES

The number of S-boxes induced by CA is much smaller than the total number of S-boxes. In fact, as a CA is defined by a single Boolean function $f : \mathbb{F}_2^d \to \mathbb{F}_2$, the number of CA-based S-boxes is 2^{2^d} for any size $n \in \mathbb{N}$ of the cellular array. On the other hand, generic S-boxes of size $n \times n$ can be exhaustively searched only up to $n = 4$, since their total number is $n \cdot 2^n$. This has been done by Leander and Poschmann in [104], where the authors classified the nonlinearity and differential uniformity of all 4×4 S-boxes up to affine equivalence.

In the case of cellular automata, the truth table approach would allow to exhaustively search the set of all $n \times n$ S-boxes up to diameter $d = 5$, since this corresponds to the set of Boolean functions of 5 variables. As we have seen in the previous section, the truth table approach yields a lot of information about the cryptographic properties of the resulting S-boxes; on the other hand, it does not give many insights about their *implementation properties* such as area, latency and power, which are equally important in the design of real-world block ciphers. As argued in [145], using a *tree* representation of a CA local rule gives a better approximation of the implementation cost of the resulting S-box. The downside of this solution is that the space of Boolean trees is too huge to be exhaustively searched even for small sizes such as 4×4.

To address this problem, in this section we design a *Genetic Programming* algorithm to heuristically search for CA-based S-boxes with good cryptographic properties. The experimental results show that GP is able to evolve S-boxes with optimal cryptographic properties for sizes up to 7×7. Finally, we perform an exhaustive search of all CA-based S-boxes of sizes 3×3 and 4×4 using the truth table approach, and then classify them up to affine equivalence with respect to their nonlinearity and differential uniformity.

11.2.1 *Genetic Programming Approach*

We saw in Section 6.2.2 that Genetic Programming (GP) is a heuristic optimization algorithm especially suited for solutions that can be encoded as trees. In our quest of good S-boxes based on CA, we employed GP to evolve trees representing Boolean functions of n variables used as CA local rules. In particular, we assumed the following: the state of the CA is represented with a periodic array of size n. The elements of the binary array are used as Boolean variables in a GP tree (GP leaves), where the variable c_0 denotes the value that is being updated. The variables $c_1, \cdots, c_{n-1}$ denote the cells to the right of the current cell. Additionally, the neighborhood of a cell is formed by the cell itself and the $n - 1$ cells to its right, so each value in the current state can be used in a local update rule, which corresponds to the

case of rotation symmetric S-boxes (i. e. $d = n$). As discussed in the previous section, this is motivated by the fact that the CA nonlinearity bound given by Theorem 25 coincides with the Sidelnikov-Chabaud-Vaudenay bound in this case.

For the function set, we adopted the following operators: NOT, which inverts its single argument, XOR, NAND, NOR, each of which takes two input arguments. Additionally, we used the function IF, which takes three arguments and returns the second one if the first one evaluates to *true*, and the third one otherwise. This function corresponds to the multiplexer gate (MUX).

Our GP algorithm evaluates a candidate solution in the following manner: all the possible 2^n input states are considered, and for each state the same local rule is applied in parallel to each of the variables to determine the next state. The obtained global rule represents a candidate S-box that is then evaluated according to the desired cryptographic criteria.

For the fitness function, we required that the evolved S-boxes are balanced (i. e. bijective), with high nonlinearity and low differential uniformity. We structured our fitness function in two stages as follows. First, the balancedness is verified, and if an S-box is balanced, we assign it a value of 0, otherwise the value equals -1; this is denoted with the label BAL. We then calculate the nonlinearity and differential uniformity only if the S-box is balanced. In particular, the differential uniformity is subtracted from the value 2^n, since the objective is to minimize this property.

The fitness function maximized by our GP algorithm is thus the following:

$$\mathrm{fitness} = \mathrm{BAL} + \Delta_{\mathrm{BAL},0}\left(N_F + \left(1 - \frac{\mathrm{nMinN}_F}{2^n}\right) + (2^n - \delta_F)\right) \quad , \tag{141}$$

where $\Delta_{\mathrm{BAL},0}$ represents the *Kronecker delta function* that equals one when the function is balanced (i. e. $\mathrm{BAL} = 0$) and zero otherwise and nMinN_F represents the number of occurrences of the current value of N_F in the population. Since the difference in neighboring levels of nonlinearity is always at least 2 (due to the fact that we considered only bijective S-boxes), the part of the expression including nMinN_F acts as a secondary criterion which is effectively being minimized and assumes values in the range $[0, 1]$.

For the selection operator, we used a modified version of tournament selection with $t = 3$, where the worst of three randomly selected individuals is eliminated and then crossover is applied to the remaining two individuals from the tournament. The rationale behind this variant is that it does not require specifying a crossover probability. The new individual is then mutated with a probability of 0.5. We note that we used the mutation probability to select whether an individual would be mutated or not, and the mutation operator is executed

only once on a given individual; e. g. if the mutation probability is 0.5, then on average 5 out of every 10 new individuals will be mutated and one mutation will be performed on each of those 5 individuals.

As variation operators we adopted simple tree crossover, uniform crossover, size fair, one-point, and context preserving crossover (selected at random), and subtree mutation [148]. All our experiments suggested that having a maximum tree depth equal to the size n of the S-box is sufficient. The initial population is created at random and every experiment is repeated 50 times.

Algorithm 2 reports the pseudocode of our overall GP heuristic.

Algorithm 2 Genetic Programming for evolving CA-based S-boxes

repeat
  randomly select 3 individuals;
  remove the worst of 3 individuals;
  child = crossover (remaining two individuals);
  perform mutation on child, with given individual mutation probability;
  generate CA-based S-box using child Boolean function
  evaluate S-box
  assign fitness to child
  insert child into population;
until stopping criterion reached

In order to examine the influence of the GP parameters, we carried out a tuning phase for the stopping criterion and the population size. The starting set of parameters was tested on S-boxes of size $n = 6$, with population 500, 1000, and 2000, for which 30 runs were executed. Although there were no significant differences, the best results in terms of average fitness value and number of optimal solutions were obtained with a population size of 2000, which we used in the subsequent experiments. Finally, we set the stopping criterion to 2000000 fitness evaluations, since no change of the best solution was detected afterwards.

11.2.2 *GP Results*

Table 12 reports the statistical results of our heuristic search, averaged over the best obtained values for each run. Column T_max denotes the theoretical maximal value of our fitness function, determined by the Sidelnikov-Chabaud-Vaudenay bound for nonlinearity and the APN bound for differential uniformity. This column serves as an indicator of the performance of our search technique.

By observing the table, one can see that the problem is easy for S-boxes of sizes 4×4 and 5×5. Indeed, in the former case all experiments finished with the optimal value, while in the latter most

of the runs reached an optimal solution. For the 6×6 case, we observe a larger standard deviation, but the obtained value is still high. Subsequently, one can remark that the problem becomes difficult for S-boxes of size 7×7, although there are a few runs reaching the optimal value. Finally, for the 8×8 size the results do not even come close to the optimal value.

Observe that for 4×4, 5×5, and 7×7 sizes we obtained S-boxes with the best possible values of nonlinearity and differential uniformity, i. e. Almost Perfect Nonlinear (APN) permutations. In the 6×6 case, our best solution is just slightly suboptimal with respect to the differential uniformity property. As a matter of fact, the permutation discovered by Dillon [23] is APN, and thus its differential uniformity equals 2, while our solution has differential uniformity equal to 4.

Table 12: Statistical results and comparison.

S-box size	T_max	GP			N_F	δ_F
		Max	Avg	Std dev		
4×4	16	16	16	0	4	4
5×5	42	42	41.73	1.01	12	2
6×6	86	84	80.47	4.72	24	4
7×7	182	182	155.07	8.86	56	2
8×8	364	318	281.87	13.86	82	20

In Table 13 we display statistics for the tree sizes for every S-box dimension. Note that the values are averaged over all runs and not only over those that resulted in S-boxes with optimal values. The two most interesting cases seem to be 4×4 and 8×8. In the former case, it appears to be easy to obtain optimal values (note that in all runs we obtained optimal solutions as given in Table 12), and therefore GP easily found even longer rules that result in optimal S-boxes. In the latter case, we see that even finding solutions that result in bijective S-boxes with suboptimal properties requires on average long trees. We consider this to be one of the reasons for relative lack of success in the 8×8 case. In other terms, one would need much larger trees and consequently much longer evolution time to obtain better solutions.

11.2.3 *Exhaustive Search Results*

In this section, we concentrate on S-boxes up to 5×5 size, i. e. those that can be exhaustively checked under the truth table representation. First, in Table 14 we classify CA-based S-boxes of sizes 3×3, 4×4, and 5×5. Column $\mathcal{S}^n_{CA}$ represents the number of S-boxes defined by PBCA with diameter equal to the array length ($d = n$, i. e. the number of Boolean functions of n variables). Furthermore, columns $\mathcal{B}^n_{CA}$ and

Table 13: Tree sizes, Equation (141)

S-box size	Min	Max	Avg	Std dev
4×4	**8**	103	48.77	25.03
5×5	**6**	67	26.27	11.93
6×6	**9**	82	35.13	18.71
7×7	**6**	64	30.63	14.77
8×8	**15**	119	68.7	28.93

$\mathrm{Opt}(\mathcal{B}^n_{CA})$ stand for the number of bijective CA-based S-boxes of size n and the number of optimal bijective CA-based S-boxes, respectively. Here, by optimal we consider having the largest possible nonlinearity and the smallest possible differential uniformity.

Table 14: Exhaustive classification of CA-based S-boxes up to size 5×5

$n \times n$	$\mathcal{S}^n_{CA}$		$\mathrm{Opt}(\mathcal{B}^n_{CA})$
3×3	256	36	12
4×4	65536	1536	512
5×5	4294967296	22500002	2880

As one can see from the table, only a small part of these CA-based S-boxes is bijective. Moreover, the number of optimal S-boxes which are also bijective is even smaller. We emphasize that $\mathrm{Opt}(\mathcal{B}^n_{CA})$ is the number of optimal CA-based S-boxes, since other generic S-boxes which are affine equivalent to these cannot be defined by a CA.

For sizes larger than 5×5, an exhaustive search is not possible. Still, a simple estimation can be made. The total number of CA-based S-boxes equals the number of Boolean functions of the corresponding size, i. e. 2^{2^n}. Next, the number of balanced Boolean functions of size n equals $\binom{2^n}{2^{n-1}}$, which also represents a trivial upper bound on the number of bijective CA-based S-boxes, due to the *balancing property* of surjective CA (see Section 2.3). As an example, for size 5×5, the number of bijective S-boxes defined by CA represents only the 26.7% of the possible balanced Boolean functions of size 5.

We now give a classification of bijective CA-based S-boxes up to affine equivalence for $n = 3, 4$, since in the 5×5 case the resulting quotient space is too large to be exhaustively searched. Tables 15 and 16 report the results of this classification. Each row of these tables specifies the number of the class, the truth table of its representative in hexadecimal format, the number of CA S-boxes inside it, and whether this class is optimal or not with respect to nonlinearity and differential uniformity.

Table 15: Equivalence classes of bijective 3×3 CA S-boxes

Class	Representative	#S-boxes	Optimal
0	0, 1, 2, 3, 4, 5, 6, 7	6	No
1	0, 1, 2, 3, 4, 5, 7, 6	6	No
2	0, 1, 2, 3, 4, 6, 7, 5	12	No
3	0, 1, 2, 4, 3, 6, 7, 5	12	Yes

The optimal trade-off of nonlinearity and differential uniformity for 3×3 S-boxes is reached when both of them equal 2. On the other hand, for the 4×4 size an optimal S-box has both nonlinearity and differential uniformity equal to 4. There are in total 16 equivalence classes of 4×4 generic S-boxes with such properties, which are denoted as $G_0, \ldots, G_{15}$ by Leander and Poschmann [104].

As it can be observed from Tables15 and 16, in the 3×3 case one of the classes is optimal, while for 4×4 size there are 4 optimal classes out of 18.

11.3 CONCLUSIONS

In this chapter we considered the S-boxes arising from cellular automata. Specifically, we first showed upper bounds for the nonlinearity and differential uniformity achievable by CA, both in the no boundary and periodic boundary settings. Seeing that the nonlinearity bound equals the Sidelnikov-Chabaud-Vaudenay bound only when the local rule diameter equals the CA length, we then applied a heuristic approach based on Genetic Programming to evolve CA rules correspnding to rotation symmetric S-boxes with good cryptographic properties. The experimental results suggest that our method has a great potential, since it is able to produce optimal S-boxes up to dimension 7×7.

On the basis of the presented results one can observe that for dimensions from 4×4 up to 7×7 it is possible to find CA rules that result in S-boxes with very good cryptographic properties. Here, only the 6×6 case remains slightly suboptimal when compared to the APN permutation found by Dillon [23].

When considering the 8×8 case, the obtained CA rules resulted in suboptimal S-boxes which are far from the results achieved by other heuristics techniques (see e. g. [139]). We also observed that the average tree sizes of 8×8 rules are significantly larger when compared with the other considered dimensions, which is likely an indicator of a more complicated evolution process.

Naturally, since not all S-boxes can be represented with a cellular automaton rule, our technique cannot be used to design all optimal S-boxes of the corresponding size. Nevertheless, we are confident that

Table 16: Equivalence classes of bijective 4×4 CA S-boxes

Class	Representative	#S-boxes	Optimal
0	F, D, B, 9, 7, 5, 3, 1, E, C, A, 8, 6, 4, 2, 0	16	No
1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, F, E	32	No
3	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, D, E, F, C	32	No
4	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, D, C, F, E	16	No
6	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, C, B, D, F, E	32	No
9	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, C, D, E, B, F	64	No
41	0, 1, 2, 3, 4, 5, 7, 6, 8, A, 9, C, B, F, E, D	128	No
193	0, 1, 2, 3, 4, 5, 8, A, 6, C, 7, F, D, B, 9, E	128	No
270	0, 1, 2, 3, 4, 6, 8, B, 5, C, 9, D, E, A, 7, F	128	Yes (G_4)
272	0, 1, 2, 3, 4, 6, 8, B, 5, C, D, 7, 9, F, A, E	128	Yes (G_6)
273	0, 1, 2, 3, 4, 5, 8, A, 6, C, 7, F, E, B, 9, D	128	No
278	0, 1, 2, 3, 4, 6, 8, B, 5, C, D, 7, A, F, 9, E	128	Yes (G_5)
279	0, 1, 2, 3, 4, 5, 8, A, 6, B, C, 7, D, F, E, 9	128	No
281	0, 1, 2, 3, 4, 5, 7, 8, 6, 9, A, C, F, B, D, E	128	No
282	0, 1, 2, 3, 4, 6, 8, B, 5, C, D, 7, F, 9, E, A	128	Yes (G_3)
288	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, C, E, F, B, D, A	32	No
289	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, C, E, B, F, D, A	64	No
291	0, 1, 2, 3, 4, 5, 7, 6, 8, A, 9, B, C, F, E, D	64	No
294	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, B, A, E, F, D, C	32	No

the corpus of obtainable functions is still large enough to give a sufficient diversity in the design of future block ciphers. As a closing remark, we note that the GP approach also offers easy handling of the resulting S-box latency and area when implemented in hardware, a property which makes our methodology even more usable.

RESILIENT AND ASYNCHRONOUS IMMUNE CA

12

This chapter is divided in two parts. In the first one, we conclude our analysis of the cryptographic properties of CA global rules by addressing their resiliency. As we saw in Chapter 3, the resiliency criterion of vectorial Boolean functions is relevant in the context of pseudorandom generation for stream ciphers. Beside that, we also remarked that the resiliency order of vectorial functions is linked to the minimum distance of linear codes. Our motivation for studying error-correcting codes induced by CA is always of cryptographic nature: as we observed in Chapter 4, MDS codes play an important role in the diffusion layers of block ciphers.

We first show that the global rules of *bipermutive* NBCA are always at least 1–resilient, thus generalizing the result in [106] about bipermutive local rules. We then prove an equivalence between *linear CA* and linear *cyclic codes*. Leveraging on our discussion in Chapter 10, we show how the systematic encoding of cyclic codes actually corresponds to the preimage computation process of the all-zeros configuration in linear CA, the latter being equivalent to the concatenation of a LFSR disturbed by the null linear recurring sequence. On the other hand, syndrome computation is equivalent to the application of the NBCA global rule. To sum up these results, we show how the $(7, 4, 3)$ cyclic Hamming code can be implemented using a NBCA of length $n = 7$ and diameter $d = 5$.

In the second part of this chapter, we introduce a new cryptographic criterion related to *asynchronous cellular automata*, a generalized CA model where some cells do not update their state. The property that we consider is *asynchrony immunity* (AI), which could be of interest in the context of *side-channel attacks*, where the targeted vulnerability does not involve the mathematical structure of the cipher, but rather its implementation. In particular, the side-channel attack model motivating our investigation of AI is the following. Suppose that a CA of length n is used as an S-box in a block cipher, and that an attacker is able to inject *clock faults* by making t cells not updating. If the CA is not (t, n)–AI, then the attacker could gain some information on the internal state of the cipher by analyzing the differences of the output distributions in the original CA and the asynchronous CA. We remark that, as far as we know, this kind of side-channel attacks have also been applied in the literature to stream ciphers based on *clock-controlled LFSRs* [78]. Hence, this could represent another motivation for investigating asynchrony immunity in CA, given the equivalence

between concatenated LFSRs and preimage computation of linear CA uncovered in Chapter 10.

The rest of this chapter is structured as follows. In Section 12.1, we prove that the global rules of bipermutive CA are always at least 1–resilient. We then show the connection between linear cyclic codes and linear CA, and exemplify the presented results by showing how to simulate the $(7,4,3)$ cyclic Hamming codes using a linear NBCA. Next, in Section 12.2 we define the considered model of asynchronous CA and formally introduce the definition of asynchrony immunity, giving some basic theoretical results regarding this property. Subsequently, we perform a computer search of $(3,10)$–asynchrony immune CA up to diameter $d = 4$, classifying them with respect to their nonlinearity. We finally summarize the results of this chapter in Section 12.3.

12.1 RESILIENT FUNCTIONS AND CYCLIC CODES FROM CA

12.1.1 1–*Resiliency of Bipermutive NBCA*

We now show that bipermutive no boundary CA are always at least 1–resilient when considered as vectorial Boolean functions. To this end, recall that a rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$ of diameter $d \in \mathbb{N}$ is *bipermutive* if it is defined as

$$f(x_1, x_2, \cdots, x_{d-1}, x_d) = x_1 \oplus g(x_2, \cdots, x_{d-1}) \oplus x_d \tag{142}$$

for all $x = (x_1, x_2, \cdots, x_{d-1}, x_d) \in \mathbb{F}_2^d$, where $g : \mathbb{F}_2^{d-2} \to \mathbb{F}_2$.

In what follows, we will make use of the following secondary construction of resilient Boolean functions, originally proved by Siegenthaler [167]:

Lemma 13. *Let* $I = \{i_1, \cdots, i_{t+1}\} \subseteq [n]$ *and* $J = \{j_1, \cdots, j_{n-t-1}\} = [n] \setminus I$ *be complementary sets of indices. Additionally, let* $f : \mathbb{F}_2^n \to \mathbb{F}_2$ *be a Boolean function of* n *variables defined as*

$$f(x_1, \cdots, x_n) = g(x_{j_1}, \cdots, x_{j_{n-t-1}}) \oplus x_{i_1} \oplus \cdots \oplus x_{i_{t+1}} \ ,$$

where $g : \mathbb{F}_2^{n-t-1} \to \mathbb{F}_2$ *is a Boolean function of* $n - t - 1$ *variables. Then,* f *is* t*–resilient.*

Hence, XORing a new variable increases by 1 the resiliency order of f. Clearly, this means that any bipermutive local rule is also a 1–resilient Boolean function[1].

The following result characterizes the component functions of a cellular automaton based on a bipermutive rule:

1 This fact was independently rediscovered much later in the context of CA by Leporati and Mariot [105], without using Siegenthaler's construction.

Lemma 14. *Let* $n, d \in \mathbb{N}$ *and let* $F : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ *be a CA of length* n *defined by a bipermutive local rule* $f : \mathbb{F}_2^d \to \mathbb{F}_2$*. Given* $m = n - d + 1$*, the component function* $v \cdot F$ *is bipermutive for all* $v \in \mathbb{F}_2^m \setminus \{\underline{0}\}$ *as well.*

Proof. Let f be defined as in Equation (142). Given $v \in \mathbb{F}_2^m \setminus \{\underline{0}\}$, recall that the *support* of v is $supp(v) = \{i : v_i \neq 0\}$. Then, the component function $v \cdot F$ can be expressed as:

$$\begin{aligned} v \cdot F = x_{i_1} \oplus g(x_{i_1+1}, \cdots, x_{i_1+1+d-2}) \oplus \\ \oplus x_{i_1+d-1} \oplus \cdots \oplus x_{i_k} \oplus \\ \oplus g(x_{i_k+1}, \cdots, x_{i_k+d-2}) \oplus x_{i_k+d-1} \ . \end{aligned} \tag{143}$$

Notice that the leftmost and rightmost variables x_{i_1} and x_{i_k+d-1} appear exactly once in Equation (143), thus they are never canceled. Let G be the Boolean function defined as:

$$\begin{aligned} G(x_{i_1+1}, \cdots, x_{i_k+d-2}) = g(x_{i_1+1}, \cdots, x_{i_1+d-2}) \oplus \\ \oplus x_{i_1+d-1} \oplus \cdots \oplus x_{i_k} \oplus \\ \oplus g(x_{i_k+1}, \cdots, x_{i_k+d-2}) \ . \end{aligned} \tag{144}$$

Hence, the component function $v \cdot F$ has the form:

$$v \cdot F = x_{i_1} \oplus G(x_{i_1+1}, \cdots, x_{i_k+d-2}) \oplus x_{i_k+d-1} \ . \tag{145}$$

As a consequence, $v \cdot F$ is bipermutive. □

By combining Lemmas 13 and 14, we get the following result:

Theorem 26. *Let* $n, d \in \mathbb{N}$ *with* $n \geqslant d$ *and let* $F : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ *be a NBCA of length* n *defined by a bipermutive local rule* $f : \mathbb{F}_2^d \to \mathbb{F}_2$*. Then,* F *is at least* 1*–resilient.*

12.1.2 *Linear CA and Cyclic Codes*

Recall that a NBCA $F : \mathbb{F}_q^n \to \mathbb{F}_q^{n-d+1}$ is called *linear* over the finite field $\mathbb{F}_q$ if its local rule is defined as $f(x_1, \cdots x_d) = a_1 x_1 \oplus \cdots \oplus a_d x_d$, with $a_i \in \mathbb{F}_q$ for all $i \in \{1, \cdots, d\}$. In this case, we can define the *polynomial associated to* f as follows:

$$p_f(x) = a_1 + a_2 x + \cdots + a_{d-1} x^{d-2} + a_d x^{d-1} \ , 0 \tag{146}$$

i.e. $p_f(x) \in \mathbb{F}_2[x]$ is a polynomial of degree $d-1$ with coefficients over $\mathbb{F}_q$. The global rule of F is described by a $(n-d+1) \times n$ *transition matrix* M_F of the following form:

$$M_F = \begin{pmatrix} a_1 & \cdots & a_d & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & a_1 & \cdots & a_d & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & a_1 & \cdots & a_d \end{pmatrix} \ . \tag{147}$$

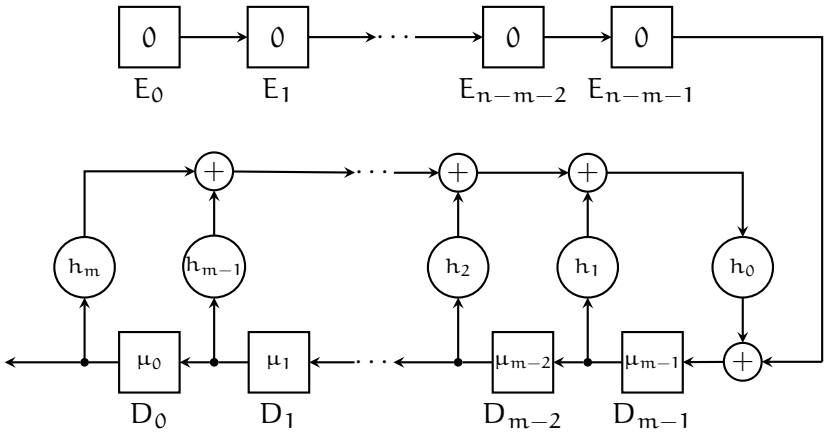


Figure 34: Concatenation of a LFSR with a $n - m$ zeros.

In particular, when the CA is both bipermutive and linear we have $a_1 \neq 0$ and $a_d \neq 0$. The application of the CA global rule F to a configuration $x \in \mathbb{F}_q^n$ corresponds to the multiplication $y = M_F x^\top$. In what follows, we cosider only the case of binary linear CA, i. e. $q = 2$.

One can notice that the generator and parity check matrices of Equation (24) in Theorem 8 have the same form of the linear CA matrix in Equation (147). In particular, the systematic encoding for cyclic codes described above can be simulated through cellular automata. As observed in Chapter 10, computing a preimage of a spatially periodic configuration in a linear bipermutive CA is equivalent to a concatenation of LFSR, where the LFSR associated to the local rule is disturbed by the LFSR which generates the spatially periodic configuration. In our case, we are only interested in a preimage of a finite configuration. Thus, the general scheme consists of the LFSR associated to the rule where the feedback is additively disturbed by the bits of the configuration. If one takes the all-zeros configuration $\underline{0}$, it can be observed that the resulting concatenated LFSR of Figure 34 is equivalent to the LFSR used for the systematic encoding of a cyclic code (see e. g. [117, 75]). As a matter of fact, adding a sequence of zeros to the feedback of a LFSR does not change its dynamics. In the context of cellular automata, the system represented in Figure 34 is equivalent to the computation of a preimage of $\underline{0} \in \mathbb{F}_2^{n-m}$, in particular the preimage determined by the m-bit block μ.

We have thus proved the following result:

Theorem 27. *Let $F : \mathbb{F}_2^{m+\rho} \to \mathbb{F}_2^m$ be a linear cellular automaton defined by a local rule $f(x) = a_1 x_1 \oplus \cdots \oplus a_d x_d$ of diameter $d = \rho + 1$ with $\rho \in \mathbb{N}$, and let $g(x) = a_1 + a_2 x + \cdots + a_d x^\rho$ be the polynomial associated to f. If $g(x)$ divides $x^n - 1$ where $n = m + \rho$, then F is equivalent to a cyclic code C of length n and dimension m. The generator matrix of C is the CA matrix M_F associated to F, while $g(x)$ is the generator polynomial of C. Additionally, let $\tilde{h}(x) = h_m + h_{m-1} x + \cdots + h_0 x^m$ be the reciprocal of the parity check polynomial $h(x) = (x^n - 1)/g(x)$, and let*

$\tilde{f}(x) = h_m x_1 \oplus \cdots \oplus h_0 x_{m+1}$ *be the corresponding local rule. Then, the matrix* $M_{\tilde{F}}$ *associated to the linear CA* $\tilde{F} : \mathbb{F}_2^{m+\rho} \to \mathbb{F}_2^{\rho}$ *induced by rule* $\tilde{f}$ *is a parity check matrix for* C*, and* $C = \tilde{F}^{-1}(\underline{o})$*.*

In other words, by Theorem 27 we can implement a linear cyclic code of length n and dimension m with a no boundary cellular automaton as follows:

1. Given m and $n = m + \rho$ with $\rho \in \mathbb{N}$, determine a local rule f of diameter $d = \rho + 1$ such that the associated polynomial $g(x)$ divides $x^n - 1$.

2. Compute the reciprocal $\tilde{h}(x)$ of the parity check polynomial $h(x) = (x^n - 1)/g(x)$, and determine the corresponding local rule $\tilde{f}$ of diameter $m + 1$.

3. *Systematic encoding*: Let $\tilde{F} : \mathbb{F}_2^{m+\rho} \to \mathbb{F}_2^{\rho}$ be the linear CA of length n induced by $\tilde{f}$. A message $\mu \in \mathbb{F}_2^m$ is encoded by computing the preimage $x \in \tilde{F}^{-1}(\underline{o})$ whose leftmost m-bit block equals μ. This preimage is computed by the LFSR in Figure 34.

4. *Syndrome computation*: given $x \in \mathbb{F}_2^{m+\rho}$, the syndrome of vector x is $s = \tilde{F}(x)$. If the syndrome s equals $\underline{o} \in \mathbb{F}_2^{\rho}$ then x is a codeword of C. Otherwise, one can apply the syndrome decoding procedure to retrieve the original codeword.

Notice that up to now we did not consider the minimum distance of the cyclic codes generated through linear CA, which is necessary in order to assess their error-correction capability. This is where the CA resiliency order comes into play. We already know from Section 12.1.1 that all bipermutive CA are always at least 1–resilient, thus a linear and bipermutive CA which satisfies the hypotheses of Theorem 27 is equivalent to a linear cyclic code with minimum distance at least 2. More in general, we can refine Theorem 27 on account of Theorem 16 reported in Chapter 3 as follows:

Theorem 28. *Let* $F : \mathbb{F}_2^{m+\rho} \to \mathbb{F}_2^m$ *be a linear CA satisfying the hypotheses of Theorem 27. If* F *is* $(d-1)$*–resilient, then the cyclic code associated to* F *has minimum distance* d*.*

12.1.3 *Cyclic Hamming Codes through Linear CA*

To sum up the results presented in the previous section, we show an example of cyclic code generated by a linear CA. In particular we focus on *cyclic Hamming codes*, which have minimum distance 3 and can thus correct up to 1 error [75]. The main reason for this choice is the simplicity of syndrome decoding. As a matter of fact, the position of the column that includes the value of the syndrome in the parity check matrix H of a Hamming code is the position where the error occurred.

Example 2 (The $(7,4,3)$ cyclic Hamming code). *Let $F : \mathbb{F}_2^7 \to \mathbb{F}_2^4$ be the linear CA induced by the local rule $f : \mathbb{F}_2^4 \to \mathbb{F}_2$ defined for all $x \in \mathbb{F}_2^4$ as $f(x) = x_1 \oplus x_2 \oplus x_4$. The associated polynomial is $g(x) = 1 + x + x^3$, while the CA matrix is:*

$$M_F = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} . \tag{148}$$

The polynomial $g(x)$ divides $x^7 - 1$, and we have $h(x) = (x^7 - 1)/g(x) = 1 + x + x^2 + x^4$. Further, we can deduce from matrix M_F that F is 2–resilient. As a matter of fact, it is not difficult to see by exhaustive enumeration that each nonzero vector v results in a sum of rows which always have at least 3 ones. Hence, by Theorem 28 the code C associated to F is the $(7,4,3)$ cyclic Hamming code. Remark that $\tilde{h}(x) = 1 + x^2 + x^3 + x^4$ is the reciprocal of the parity check polynomial $h(x)$. The local rule $\tilde{f}$ associated to the polynomial $\tilde{h}(x)$ is $\tilde{f}(x) = x_1 \oplus x_3 \oplus x_4 \oplus x_5$, and thus it has diameter $d = 2$. The Wolfram code representing the truth table of $\tilde{f}$ is 1768527510, while the transition matrix of the linear CA $\tilde{F} : \mathbb{F}_2^7 \to \mathbb{F}_2^3$ induced by $\tilde{f}$ is:

$$M_{\tilde{F}} = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} . \tag{149}$$

Let $\mu = (0,1,1,0) \in \mathbb{F}_2^4$ be a 4-bit message. The systematic encoding of μ under the Hamming code $(7,4,3)$ can be accomplished by computing the preimage x of $(0,0,0)$ under the action of $\tilde{F}$, with the leftmost 4 bits of x initialized to μ. This process is depicted in Figure 37. Hence, the codeword corresponding to μ is $x = (0,1,1,0,1,0,0)$.

Let us now assume that x is transmitted through a noisy channel and the fourth bit of x is flipped, thus yielding the word $\tilde{x} = (0,1,1,1,1,0,0)$. The receiver applies to $\tilde{x}$ the CA $\tilde{F}$ defined by rule 1768527510, thus obtaining the syndrome $s = F(x) = (1,1,0)$, as shown in Figure 40(a). To correct the error, the receiver looks at the CA matrix $M_{\tilde{F}}$ and finds that the syndrome appears in the fourth column. Thus, the receiver knows that a transmission error has occurred in the fourth position of $\tilde{x}$, and the original codeword can be recovered as $\tilde{x} \oplus (0,0,0,1,0,0,0) = x$.

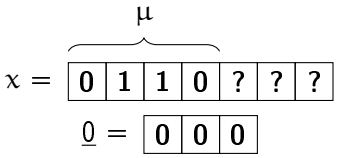


Figure 35: Initialization

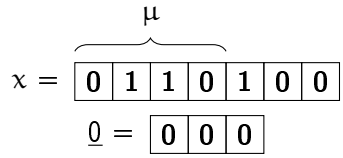


Figure 36: Complete codeword

Figure 37: Example of systematic encoding of using rule 1768527510.

$$x = \overset{*}{\boxed{0}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{0}}$$
$$s = \boxed{1}\boxed{1}\boxed{0}$$

Figure 38: Syndrome computation

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$
$$\uparrow$$

Figure 39: Error correction

Figure 40: Example of error correction using rule 1768527510. The cell marked by $*$ indicates where the error occurred.

12.2 ASYNCHRONOUS IMMUNE CA

12.2.1 *Basic Definition and Properties of Asynchrony Immunity*

Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ be a no boundary CA of length n and local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$. Additionally, let $I = \{i_1, \cdots, i_t\} \subseteq [m]$ be a subset of indices of the output variables of the CA, where $m = n - d + 1$. The t–*asynchronous CA* (t–ACA) $\tilde{F}_I$ induced by I on F is obtained by preventing the input variables $x_{i_1}, \cdots, x_{i_t}$ to update. In particular, for all indices $i_k \in I$ the coordinate function f_{i_k} equals the identity, while f_j still corresponds to the local rule f applied to the neighborhood $\{j, \cdots, j + d - 1\}$ for all remaining indices $j \in J = [m] \setminus I$.

The property of asynchrony immunity can be described by a three-move game between a user and an adversary. Let $d, m \in \mathbb{N}$, with $m = n - d + 1$ and $t \leqslant m$. The game works as follows:

1. The user chooses a local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$ of diameter d
2. The adversary chooses $j \leqslant t$ cells in the range $[m]$.
3. The user evaluates the output distribution D of the no boundary CA $F : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ and the distribution $\tilde{D}$ of the asynchronous CA $\tilde{F} : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ where the t cells selected by the adversary are not updated
4. *Outcome*: if both D and $\tilde{D}$ equal the uniform distribution, the user wins. Otherwise, the adversary wins

Recall that a CA $F : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ is *balanced* if for all $y \in \mathbb{F}_2^m$ it holds that $|F^{-1}(y)| = 2^{d-1}$. Then, F is called (t, n)–asynchrony immune if, for all $j \leqslant t$, the asynchronous CA $\tilde{F} : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ resulting from not updating any subset of j cells of F is balanced as well. Thus, asynchrony immune CA rules represent the *winning strategies* of the user in the above game.

Notice the difference between the asynchrony immunity game and the t–*resilient functions game* introduced in [41]: in the latter, generic vectorial Boolean functions $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ are considered instead of cellular automata, and the adversary selects both values and positions of the t input variables. As a matter of fact, the winning strategies

of that game actually correspond to the usual definition of resilient (n, m)–functions.

We formally define asynchrony immunity in CA as follows:

Definition 32. *Let* $m, n, d, t \in \mathbb{N}$ *with* $m = n - d + 1$ *and* $t \leqslant m$*, and let* $f : \mathbb{F}_2^d \to \mathbb{F}_2$ *be a local rule of diameter* d*. The NBCA* $F : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ *defined by rule* f *is* (t, n)–asynchrony immune *((t, n)–AI) if, for all sets* $I \subseteq \{0\} \cup [m]$ *with* $|I| \leqslant t$*, the* t*–ACA* $\tilde{F}_I : \mathbb{F}_2^n \to \mathbb{F}_2^m$ *is balanced, i. e.* $|\tilde{F}_I^{-1}(y)| = 2^{d-1}$ *holds for all* $y \in \mathbb{F}_2^m$*.*

Remark 2. *The definition of* (t, n)*–asynchrony immunity implies in particular that the CA itself is balanced, since if* $|I| = 0$ *then we get the synchronous global rule* F*. This also means that the local rule must be a balanced Boolean function as well.*

Among all possible 2^{2^d} rules of diameter d, we are interested in finding all local rules inducing asynchrony immune CA that satisfy additional useful cryptographic properties, such as high nonlinearity. As a consequence, proving necessary conditions for (t, n)–AI helps one to prune the search space for possible candidates.

We begin by showing that asynchrony immunity is invariant under reflection and complement. To this end, recall that the *reverse* of a vector $x = (x_1, \cdots, x_n)$ is the same vector arranged in reverse order, i. e. $x^R = (x_n, \cdots, x_1)$, while the *complement* of x is defined as the vector $x^C = (1 \oplus x_0, \cdots, 1 \oplus x_n)$. Given $f : \mathbb{F}_2^d \to \mathbb{F}_2$, the *reflected* and *complemented* rules f^R and f^C are respectively defined as $f^R(x) = f(x^R)$ and $f^C(x) = 1 \oplus f(x)$, for all $x \in \mathbb{F}_2^d$. For all $n > d$, the reflected and complemented NBCA $F^R : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ and $F^C : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ are respectively defined for all $x \in \mathbb{F}_2^n$ as follows:

$$F^R(x) = F(x^R)^R = (f(x_d, \cdots, x_1), \cdots, f(x_n, \cdots, x_{n-d+1})) \ , \tag{150}$$

$$F^C(x) = \underline{1} \oplus F(x) = (1 \oplus f(x_1, \cdots, x_d), \cdots, 1 \oplus f(x_{n-d+1}, \cdots, x_n)) \ . \tag{151}$$

The following result shows that asynchrony immunity is preserved under reflection and complement.

Lemma 15. *Let* $f : \mathbb{F}_2^d \to \mathbb{F}_2$ *be a local rule inducing a* (t, n)*–AI NBCA* $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$*, with* $m = n - d + 1$ *and* $t \leqslant m$*. Then, the reflected and complemented rules* f^R *and* f^C *are* (t, n)*–AI as well.*

Proof. Let $I = \{i_1, \cdots, i_l\} \subseteq [m]$, with $l \leqslant t$. For the reflected rule f^R, we know by (150) that $F^R(x) = F(x^R)^R$. It follows that the reflection of the l-ACA $\tilde{F}_I$ is defined as:

$$\tilde{F}_I^R(x) = \tilde{F}_J(x^R)^R = (f(x_d, \cdots, x_1), \cdots, x_{j_1}, \cdots, x_{j_l}, \cdots, f(x_n, \cdots, x_m)) \ , \tag{152}$$

where $J = \{j_1, \cdots, j_l\}$ and $j_s = m - i_s$ for all $1 \leqslant s \leqslant l$. Rule f induces a (t, n)–AI NBCA and J is still a set of $l \leqslant t$ indices, thus

$|F^{-1}(y)| = |\tilde{F}_J^{-1}(y)| = 2^{d-1}$ for all $y \in \mathbb{F}_2^m$. Since the reverse operator is a bijection over both $\mathbb{F}_2^n$ and $\mathbb{F}_2^m$, by Equation (152) it results that $|(F^R)^{-1}(y)| = |F^{-1}(y)|$ and $|(\tilde{F}_I^R)^{-1}(y)| = |\tilde{F}_J^{-1}(y)|$. Thus, the reflected CA F^R is (t, n)–AI as well.

Analogously, with rule f^C the l-ACA $\tilde{F}_I$ is defined as:

$$\tilde{F}_I^C(x) = (1 \oplus f(x_1, \cdots, x_d), \cdots, x_{i_1}, \cdots, x_{i_l}, \cdots, 1 \oplus f(x_m, \cdots, x_n)) \ . \tag{153}$$

Hence we can compute $\tilde{F}_I^C$ by XORing $\tilde{F}_I$ with a bitmask composed of all 1s excepts in the positions $i_1, \cdots, i_l$. Since this operation is again a bijection over $\mathbb{F}_2^m$ and F is (t, n)–asynchrony immune, it means that $|(F^C)^{-1}(y)| = |F^{-1}(y)| = 2^{d-1}$ and $|(\tilde{F}_I^C)^{-1}(y)| = |\tilde{F}_I^{-1}(y)| = 2^{d-1}$ for all $y \in \mathbb{F}_2^m$. Thus, F^C is also (t, n)–AI. □

12.2.2 *Search of AI Rules up to 4 Variables*

In order to search for asynchrony immune NBCA having additional cryptographic properties, by Remark 2 and Lemma 15 we only need to explore balanced rules under the equivalence classes induced by reflection and complement. We performed an exhaustive search among all elementary rules of diameter $d = 3$ in order to find those inducing (t, n)–asynchrony immune CA with $t = 3$ and $n < 10$. The reason why we limited our analysis to these particular values is twofold. First, checking for asynchrony immunity is a computationally cumbersome task, since it requires to determine the output distribution of the t-ACA for all possible choices of at most t blocked cells. Second, the S-boxes employed as nonlinear components in several real-world cryptographic primitives, such as Keccak [15], are not large: usually, their size do not exceed 8.

In our quest for asynchrony immune CA we also took into account the *nonlinearity* property of the underlying local rules. Up to reflection and complement, and neglecting the identity rule that trivially satisfies AI for every length n and order t, we found that only rule 60 generates asynchrony immune CA for all lengths $n < 10$. Since rule 60 is linear, however, it is not interesting from the cryptographic standpoint. We thus extended the search by considering all local rules of diameter $d = 4$. The search returned a total of 18 rules that generate (t, n)–asynchrony immune CA with $t = 3$ and $n \leqslant 10$, among which several of them are nonlinear. Table 1 reports the Wolfram codes of the discovered rules, along with their nonlinearity values and algebraic normal form (ANF). One can notice from the ANF column in Table 17 that all discovered rules depend on the input variable x_1 in a linear way. Hence, each rule can be written as $f(x_1, x_2, x_3, x_4) = x_2 \oplus g(x_1, x_3, x_4)$, where $g : \mathbb{F}_2^3 \to \mathbb{F}_2$. This means that the discovered rules are all *center permutive*, i. e. by fixing the values of all variables except x_2 the resulting restrictions of the functions are permutations over $\mathbb{F}_2$. Remark that the elementary rule 60

Rule	$\mathrm{Nl}(f)$	$f(x_0, x_1, x_2, x_3)$
13107	0	$1 \oplus x_1$
13116	4	$x_1 \oplus x_2 \oplus x_3 \oplus x_2x_3$
13155	2	$1 \oplus x_1 \oplus x_2 \oplus x_0x_2 \oplus x_2x_3 \oplus x_0x_2x_3$
13164	2	$x_1 \oplus x_0x_2 \oplus x_3 \oplus x_0x_2x_3$
13203	2	$1 \oplus x_1 \oplus x_0x_2 \oplus x_0x_2x_3$
13212	2	$x_1 \oplus x_2 \oplus x_0x_2 \oplus x_3 \oplus x_2x_3 \oplus x_0x_2x_3$
13251	4	$1 \oplus x_1 \oplus x_2 \oplus x_2x_3$
13260	0	$x_1 \oplus x_3$
13875	2	$1 \oplus x_1 \oplus x_3 \oplus x_0x_3 \oplus x_2x_3 \oplus x_0x_2x_3$
14028	2	$x_1 \oplus x_0x_3 \oplus x_2x_3 \oplus x_0x_2x_3$
14643	2	$1 \oplus x_1 \oplus x_0x_3 \oplus x_0x_2x_3$
14796	2	$x_1 \oplus x_3 \oplus x_0x_3 \oplus x_0x_2x_3$
15411	4	$1 \oplus x_1 \oplus x_3 \oplus x_2x_3$
15420	0	$x_1 \oplus x_2$
15555	0	$1 \oplus x_1 \oplus x_2 \oplus x_3$
15564	4	$x_1 \oplus x_2x_3$
26214	0	$x_0 \oplus x_1$
26265	0	$1 \oplus x_0 \oplus x_1 \oplus x_3$

Table 17: List of $d = 4$ rules inducing (t, n)–AI CA with $t = 3$ and $n \leqslant 10$.

is center permutive as well, being defined as $f(x_1, x_2, x_3) = x_2 \oplus x_3$. This seems to suggest that center permutivity is a necessary condition for asynchrony immunity, a property that would greatly reduce the search space of possible AI candidates with interesting cryptographic properties. We formalize this remark in the following conjecture:

Conjecture 1. *Let* $f : \mathbb{F}_2^d \to \mathbb{F}_2$ *be a local rule of* d *variables inducing a* (t, n)*–asynchrony immune CA. Then, rule* f *is center-permutive.*

We strongly suspect that this conjecture is true, since Manzoni [110] showed that center permutivity is related to the surjectivity (and thus to the balancing property) of asynchronous infinite CA.

12.3 CONCLUSIONS

In this chapter, we investigated the resiliency property of bipermutive CA and then introduced the new criterion of *asynchrony immunity*, motivating it as a possible countermeasure for clock-fault attacks in CA-based ciphers.

In particular, we proved that the global rule of a bipermutive NBCA F is always at least 1–resilient, since each component of F is still a bipermutive Boolean function. We then presented an equivalence between linear cyclic codes and linear NBCA, showing that syndrome computation in the former is equivalent to applying the global rule to the received word in the latter. The resiliency order of a linear and bipermutive CA can thus be used to determine the minimum distance of the corresponding cyclic code. We then applied these results by showing how the $(7, 4, 3)$ cyclic Hamming code can be implemented using a linear NBCA of diameter $r = 2$.

Next, we formally introduced the property of asynchrony immunity in NBCA, proving some basic theoretical results about invariance under reflection and complement operations. We finally performed a computer search of all (t, n)–asynchrony immune CA with $t = 3$ and $n \leqslant 10$ defined by local rules of diameter up to $d = 4$, classifying them with respect to their nonlinearity.

Part IV

COMBINATORIAL DESIGNS AND CELLULAR AUTOMATA

ORTHOGONAL LATIN SQUARES ARISING FROM LINEAR CA

13

In Chapter 10 we analyzed the periods of preimages of spatially periodic configurations under the action of surjective CA. This problem turned out to be related to the maximum number of players allowed in the secret sharing scheme based on bipermutive CA described in [112], which induces a cyclic access structure. In particular, beside reaching a minimum threshold, the shares of the players must satisfy an adjacency constraint, since they are blocks of a CA preimage.

As we discussed in Section 4.4, (t, n)–threshold schemes are equivalent to $t-(v, n+1, 1)$ orthogonal arrays. Hence, a possible direction to design a CA-based threshold scheme without adjacency constraints is to investigate under which conditions CA are able to generate orthogonal arrays. This would improve on the state of the art: as remarked in Section 7.4, the current SSS based on CA proposed in the literature all feature a sequential threshold access structure. Remark also that, even though the problem of (t, n) threshold secret sharing has already been solved with solutions not based on CA (see e. g. Shamir's scheme [163]), we deem interesting to investigate the orthogonal arrays engendered by CA, due also to their connections with coding theory and Boolean functions.

To make our research question more concrete, we consider orthogonal arrays of the form $OA(k, n)$, i. e. with $t = 2$. The reason for this choice is twofold. First, as we observed in Section 3.2 $OA(k, n)$ corresponds to sets of mutually orthogonal Latin squares (MOLS), which have a simpler combinatorial description. Second, beside being equivalent to a $(2, n)$–threshold schemes, a $OA(k, n)$ can also be used to generate a perfect authentication code, as explained in Section 4.3.

As a consequence, in this chapter and the next one we undertake an investigation of orthogonal Latin squares generated by cellular automata. More precisely, in this chapter we focus on the characterization and enumeration of orthogonal Latin squares generated by linear CA, while in the next one we generalize the problem to nonlinear CA.

In particular, in the first part of this chapter we start by showing that every bipermutive cellular automaton of diameter d and length $n = 2(d-1)$ induces a Latin square of order q^{d-1}, where q is the cardinality of the CA state alphabet. We then prove that two bipermutive CA with linear local rules generate a pair of orthogonal Latin squares if and only if their associated *Sylvester matrix* is invertible, i. e. if and only if the polynomials corresponding to their local rules are relatively prime. Leveraging on this characterization result, we then

describe a $(2, n)$–threshold scheme and a perfect authentication code based on linear CA.

In the second part of this chapter we focus on counting pairs of linear CA generating OLS. This problem turns out to be equivalent to the enumeration of coprime pairs of polynomials with degree n and nonzero constant term. We first show an approach to count such pair by defining an equivalence relation based on Euclid's algorithm. Then, we completely solve the counting problem for $q = 2$ through a recurrence equation, remarking that the resulting integer sequence is already known in the OEIS for several other facts. Finally, we present a construction for sets of MOLS based on linear CA, conjecturing its optimality as a closing remark.

The remainder of this chapter is organized as follows. Section 13.1 presents the proof that a pair of linear CA induce orthogonal Latin squares if and only if the associated polynomials are coprime, and describes a $(2, n)$-threshold scheme and a perfect authentication code which are based on this characterization result. Section 13.2 addresses the enumeration of coprime polynomial pairs. Section 13.3 describes a construction for sets of linear CA-based MOLS. Finally, Section 13.4 briefly summarizes the content of this chapter.

13.1 CHARACTERIZATION RESULTS

In this section, we first observe that any bipermutive CA can be used to generate a Latin square. We then prove a necessary and sufficient condition which characterizes when the Latin squares associated to two linear bipermutive CA are orthogonal. Finally, we use these findings to describe a $(2, n)$–threshold scheme and an authentication code based on linear CA.

13.1.1 *Latin Squares from Bipermutive CA*

We begin by showing that any bipermutive NBCA of diameter d and length $n = 2m$ generates a Latin square of order $N = q^m$, where $m = d - 1$. To this end, we first need some additional notation and definitions.

Given an alphabet Σ of q symbols, in what follows we assume that a total order $\leqslant$ is defined over Σ^m, and that $\phi : \Sigma^m \to [N]$ is a monotone one-to-one mapping between Σ^m and $[N] = \{1, \cdots, q^m\}$, where $[N]$ is endowed with the usual order of natural numbers. We denote by ψ the inverse mapping of ϕ.

We now formally define the notion of square associated to a CA:

Definition 33. *Let* $f : \Sigma^d \to \Sigma$ *be a local rule of diameter* d *over an alphabet* Σ *of* q *symbols. The* square *associated to the NBCA* $F : \Sigma^{2m} \to A^m$ *with*

$m = d-1$ *defined by rule* $f : \Sigma^d \to \Sigma$ *is the square matrix* $\mathcal{S}_F$ *of size* $q^m \times q^m$ *with entries from* Σ^m *defined for all* $1 \leqslant i, j \leqslant q^m$ *as*

$$\mathcal{S}_F(i,j) = \phi(F(\psi(i)||\psi(j))) \ , \tag{154}$$

where $\psi(i)||\psi(j) \in \Sigma^{2m}$ *denotes the concatenation of* $\psi(i), \psi(j) \in \Sigma^m$.

Hence, the square $\mathcal{S}_F$ is defined by encoding the first half of the CA configuration as the row coordinate i, the second half as the column coordinate j and the output $F(\psi(i)||\psi(j))$ as the entry at (i,j).

We remark that this representation has been adopted in several works in the CA literature, even though under a different guise. Indeed, one can consider the square associated to a CA as the *Cayley table* of an algebraic structure $(A, \circ)$, where A is a set of size 2^{n-1} isomorphic to Σ^{d-1}, and $\circ$ is a binary operation over A. The two operands $x, y \in A$ are represented by the vectors respectively composed of the leftmost and rightmost $d-1$ input cells of the CA, while the $d-1$ output cells represent the result $z = x \circ y$. To the best of our knowledge, the first researchers who employed this algebraic characterization of cellular automata were Pedersen [133] and Eloranta [59], respectively for investigating their periodicity and partial reversibility properties. Other works in this line of research include Moore and Drisko [126], which studied the algebraic properties of the square representation of CA, and Moore [125], which considered the computational complexity of predicting CA whose local rules define solvable and nilpotent groups.

The next Lemma proved in [112] states that fixing at least $d-1$ adjacent cells in the global rule of a bipermutive CA yields a permutation between the remaining variables and the output:

Lemma 16. *Let* $F : \Sigma^n \to \Sigma^{n-d+1)}$ *be a NBCA defined by a bipermutive local rule* $f : \Sigma^d \to \Sigma$. *Then, by fixing at least* $b \geqslant 2(d-1)$ *adjacent coordinates of* $x \in \Sigma^n$ *to* $\tilde{x} \in \Sigma^b$, *the restriction* $F|_{\tilde{x}} : \Sigma^{n-b} \to \Sigma^{n-b}$ *of the global rule is a permutation over* Σ^{n-b}.

On account of Lemma 16, it is now straightforward to prove that the squares associated to bipermutive CA are indeed Latin squares:

Lemma 17. *Let* $f : \Sigma^d \to \Sigma$ *be a bipermutive local rule defined over* Σ *with* $|\Sigma| = q$, *and let* $m = 2(d-1)$. *Then, the square* L_F *of the bipermutive NBCA* $F : \Sigma^{2m} \to \Sigma^m$ *is a Latin square of order* $N = q^m$ *over* $X = [N]$.

Proof. Let $i \in [N]$ be a row of L_F, and let $\psi(i) = (x_0, \cdots, x_{m-1}) \in \Sigma^m$ be the vector associated to i with respect to the total order $\leqslant$ on Σ^m. Consider now the set $C = \{c \in \Sigma^{2m} : (c_1, \cdots, c_m) = \psi(i)$, i. e. the set of configurations of length $2m$ whose first m coordinates coincide with $\psi(i)$, and let $F|_C : \Sigma^m \to \Sigma^m$ be the restriction of global rule F determined by C. Then, $F|_C$ is a permutation over Σ^m by Lemma 16 .As a consequence, the i-th row of L_F is a permutation of $X = [N]$.

A symmetric argument holds when considering a column j of L_F with $1 \leqslant j \leqslant N$, which fixes the rightmost m variables of F to $\psi(j)$. Hence, every column of L_F is also a permutation of X, and thus L_F is a Latin square of order N. □

As an example, for $\Sigma = \mathbb{F}_2$ and diameter $d = 3$, Figure 43 reports the Latin square L_F associated to the NBCA $F_{150} : \mathbb{F}_2^4 \to \mathbb{F}_2^2$ defined by rule 150, i. e. $f_{150}(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$. The mapping ϕ is defined as $\phi(00) \mapsto 1$, $\phi(10) \mapsto 2$, $\phi(01) \mapsto 3$ and $\phi(11) \mapsto 4$.

13.1.2 *Orthogonal Latin Squares from Linear Bipermutive CA*

We now aim at characterizing pairs of CA which generate orthogonal Latin squares. For alphabet $\Sigma = \mathbb{F}_2$ and diameter $d = 3$ there exist only two bipermutive rules up to reflection and complement, i. e. rule 150 and rule 90, the latter defined as $f_{90}(x_1, x_2, x_3) = x_1 \oplus x_3$. Both rules are linear and their associated Latin squares of order $N = 4$ are orthogonal, as shown in Figure 47.

In what follows, we narrow our investigation to linear rules over the finite field $\mathbb{F}_q$. Recall from Section 12.1.2 that we can associate a polynomial $p_f(x)$ of degree $d-1$ defined as in Equation (146) to any linear rule f of diameter d over $\mathbb{F}_q$. Additionally, the global rule of a linear NBCA of length n and diameter d is defined by a transition matrix of size $(n-d+1) \times n$ defined as in Equation (147), where each row corresponds to a shift of the local rule coefficients.

The following result gives a necessary and sufficient condition on the transition matrices of linear bipermutive NBCA that generate orthogonal Latin squares:

Lemma 18. *Let* $F : \mathbb{F}_q^{2(d-1)} \to \mathbb{F}_q^{d-1}$ *and* $G : \mathbb{F}_q^{2(d-1)} \to \mathbb{F}_q^{d-1}$ *be linear CA of diameter* d *with linear rules* $f(x_1, \cdots, x_d) = a_1 x_1 + \cdots a_d x_d$ *and* $g(x_1, \cdots, x_d) = b_1 x_1 + \cdots b_d x_d$ *respectively, where* $a_1, b_1, a_d, b_d \neq 0$. *Additionally, let* M_F *and* M_G *be the* $d-1 \times 2(d-1)$ *matrices associated*

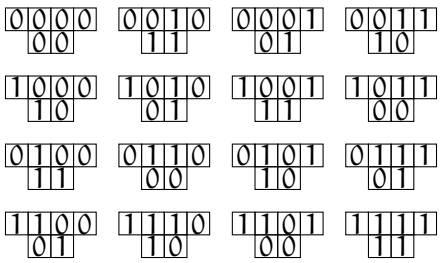


Figure 41: Truth table of F_{150}

1	4	3	2
2	3	4	1
4	1	2	3
3	2	1	4

Figure 42: Latin square $L_{F_{150}}$

Figure 43: Example of Latin square of order 4 induced by rule 150.

1	4	3	2
2	3	4	1
4	1	2	3
3	2	1	4

Figure 44: Rule 150

1	2	3	4
2	1	4	3
3	4	1	2
4	3	2	1

Figure 45: Rule 90

1,1	4,2	3,3	2,4
2,2	3,1	4,4	1,3
4,3	1,4	2,1	3,2
3,4	2,3	1,2	4,1

Figure 46: Overlay

Figure 47: OLS generated by bipermutive CA with rule 150 and 90.

to the global rules F *and* G *respectively, and define the* $2(d-1) \times 2(d-1)$ *matrix* M *as*

$$M = \begin{pmatrix} a_1 & \cdots & a_d & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & a_1 & \cdots & a_d & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & a_1 & \cdots & a_d \\ b_1 & \cdots & b_d & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & b_1 & \cdots & b_d & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & b_1 & \cdots & b_d \end{pmatrix}, \tag{155}$$

i.e. , M *is obtained by superposing the transition matrices* M_F *and* M_G*. Then, the Latin squares* L_F *and* L_G *generated by* F *and* G *are orthogonal if and only if the determinant of* M *over* $\mathbb{F}_q$ *is not null.*

Proof. Denote by $z = x \| y$ the concatenation of vectors x and y. We show that the function $\mathcal{H} : \mathbb{F}_q^{2(d-1)} \times \mathbb{F}_q^{2(d-1)} \to \mathbb{F}_q^{2(d-1)} \times \mathbb{F}_q^{2(d-1)}$, defined for all $(x, y) \in \mathbb{F}_q^{2(d-1)} \times \mathbb{F}_q^{2(d-1)}$ as

$$\mathcal{H}(x, y) = (F(z), \mathcal{G}(z)) = (\tilde{x}, \tilde{y}) \tag{156}$$

is bijective. Let us rewrite Equation (156) as a system of two equations:

$$\begin{cases} F(z) = M_F z^\top = \tilde{x} \\ \mathcal{G}(z) = M_G z^\top = \tilde{y} \end{cases}. \tag{157}$$

As M consists of the superposition of M_F and M_G, Equation (157) defines a linear system in $2(d-1)$ equations and $2(d-1)$ unknowns with associated matrix M. Thus, we have that $\mathcal{H}(x, y) = Mz^\top$, and $\mathcal{H}$ is bijective if and only if the determinant of M is not null. □

Remark that matrix M in Equation (155) is a *Sylvester matrix*, and its determinant is the *resultant* of the two polynomials $p_f(x)$ and $p_g(x)$ associated to f and g respectively. The resultant of two polynomials is nonzero if and only if they are relatively prime (see [107]). We thus have the following result:

Theorem 29. *Let* $f, g : \mathbb{F}_q^d \to \mathbb{F}_q$ *be linear bipermutive rules of diameter* $d \in \mathbb{N}$*, and let* $m = d - 1$*. Then, the squares* L_F *and* L_G *of order* q^m *respectively associated to the NBCA* $F : \mathbb{F}_q^{2m} \to \mathbb{F}_q^m$ *and* $G : \mathbb{F}_q^{2m} \to \mathbb{F}_q^m$ *are orthogonal if and only if the polynomials* $p_f(x)$ *and* $p_g(x)$ *are relatively prime, i.e. if and only if* $\gcd(p_f(x), p_g(x)) = 1$.

Hence, combining Theorem 29 with the construction of orthogonal arrays based on MOLS which we described in Section 3.2, we obtain the following characterization theorem:

Theorem 30. *Let* $f_1, \cdots f_n$ *be linear bipermutive local rules over* $\mathbb{F}_q$ *of diameter* d *whose associated polynomials are pairwise coprime. Then, given* $m = d - 1$*, there exists a set of* n *MOLS of order* $N = q^m$*, or equivalently an* $OA(n + 2, N)$.

13.1.3 *Threshold Schemes and Authentication Codes from Linear CA*

Given the equivalence between MOLS and OA, Theorem 30 gives some additional insights on how to design a CA-based secret sharing scheme with threshold $t = 2$. The same goes also for perfect authentication codes, since they are equivalent to orthogonal arrays of the form $OA(k, n)$, as shown in Section 4.3.

Let the secret S be a vector of $\mathbb{F}_q^m$ where $m = d - 1$, and assume that there are n players $P_1, \cdots, P_n$. Then, the setup phase of the secret sharing scheme is as follows:

SETUP PHASE

INITIALIZATION:

1. Pick n relatively prime polynomials $p_{f_1}(x), \cdots, p_{f_n}(x)$ over $\mathbb{F}_q$ with degree $d-1$ and nonzero constant term, and build the corresponding linear rules $f_1, \cdots, f_n$ of diameter d
2. Concatenate secret S with a random vector $R \in \mathbb{F}_q^m$, thus obtaining a configuration $C \in \mathbb{F}_q^{2m}$ of length $2(d-1)$

LOOP: For all $i \in \{1, \cdots n\}$ do:

1. Given $F_i : \mathbb{F}_q^{2m} \to \mathbb{F}_2^m$ the NBCA defined by rule f_i, compute $B_i = F_i(C)$
2. Send share B_i to player P_i

Adopting the point of view of OA, The second initialization step corresponds to the phase where the dealer chooses one of the rows of the array whose first component is the secret. Additionally, the dealer determines the remaining entries in the selected row by applying the n CA $F_1, \cdots, F_n$ over the initial configuration C. As explained in Section 4.4, these entries are the shares to be distributed to the players.

For the recovery phase, suppose that two players P_i and P_j want to determine the secret. Let B_i and B_j respectively denote the share of P_i and P_j. Since the orthogonal array is public, both P_i and P_j know the CA linear rules f_i and f_j used by the dealer to compute their shares. Hence, they adopt the following procedure to recover S:

RECOVERY PHASE

INITIALIZATION:

1. Compute the Sylvester matrix M by superposing the transition matrices of the CA F_i and F_j
2. Determine the inverse matrix M^{-1}

RECONSTRUCTION:

1. Determine the vector $y = (B_i \| B_j)$ by concatenating the two shares B_i and B_j
2. Reconstruct the secret by computing $C = (S \| R) = M^{-1} y^{\top}$

OUTPUT: Return the first half of C as the secret S

For a perfect authentication code, a possible protocol based on linear bipermutive CA that exploits Theorem 30 is the following:

AUTHENTICATION CODE

INITIALIZATION :

1. Alice and Bob establish a set of linear bipermutive rules $f_1, \cdots f_n$ over $\mathbb{F}_q$ of diameter d such that the corresponding polynomials are pairwise coprime
2. Alice and Bob agree on a key $K \in \mathbb{F}_q^{2m}$, where $m = d - 1$, which they keep secret.

SIGNING PHASE : Given a message $i \in \{1, \cdots, n\}$, Alice computes the authenticator $a = F_i(K)$, where $F_i : \mathbb{F}_q^{2m} \to \mathbb{F}_2^{2m}$ is the NBCA defined by rule f_i, and sends $c = (i, a)$ to Bob

VERIFICATION PHASE : Bob checks whether $F_i(K) = a$. If so, he accepts the message as legitimately sent by Alice. Otherwise, he rejects the message as being altered by an adversary.

As a concluding remark, observe that both the $(2, n)$–threshold scheme and the authentication code described above require the determination of a set of n pairwise coprime polynomials over $\mathbb{F}_q$ with degree $d - 1$ and nonzero constant term. For practical purposes, one could settle for a set of n irreducible polynomials of degree $d - 1$,

since they are clearly pairwise coprime and all of them have nonzero constant term (see [107]). The advantage is that there are several efficient algorithms described in the literature for generating irreducible polynomials (see for instance [166]).

13.2 COUNTING COPRIME POLYNOMIAL PAIRS

By Theorem 30, one can generate a set of n MOLS of order q^m through linear CA of diameter d by finding n pairwise relatively prime polynomials of degree $d-1$. The problem of counting the number of pairs of relatively prime polynomials over finite fields has been investigated in several papers (see [152, 10, 80]). However, notice that determining the number of pairs of linear CA inducing orthogonal Latin squares entails counting only specific pairs of polynomials, namely those whose constant term is not null. This is due to the requirement that the CA local rules must be bipermutive.

To the best of our knowledge, this particular counting problem has not been considered in the literature, for which reason we address it in this section. For simplicity, we limit our investigation only to polynomials over $\mathbb{F}_2$, since this represents the most useful case for practical applications. Hence, a linear bipermutive rule f of diameter $d = n$ will correspond to a monic polynomial $P_f(x)$ of degree n and constant term equal to 1.

In the rest of this section, we first formally state the counting problem of our interest. We then present two possible approach two solve it: the first is based on an equivalence relation on polynomial pairs induced by Euclid's division algorithm. However, this method proves itself to be inconclusive for achieving our goal, since it relies on a conjecture about the proportion of coprime and non coprime polynomial pairs over $\mathbb{F}_2$ which still seems to be open in the literature. We thus shift to a different method, based on a recurrence equation, through which we finally solve the counting problem and, in turn, the conjecture induced by the equivalence relation approach. The closed form of this recurrence equation turns out to generate an integer sequence which is known in the OEIS for several other facts not related to polynomials, Latin squares or cellular automata. Finally, we present a construction for a set of k-MOLS based on linear CA (i. e. , a construction for sets of pairwise coprime polynomials), and conclude by conjecturing its maximality.

13.2.1 *Problem Statement*

Let $n \in \mathbb{N}$ be a positive integer, and define $f, g \in \mathbb{F}_2[x]$ as follows:

$$f(x) = 1 + a_1 x + \cdots + a_{n-1} x^{n-1} + x^n \ , \tag{158}$$
$$g(x) = 1 + b_1 x + \cdots + b_{n-1} x^{n-1} + x^n \ , \tag{159}$$

where $a_i, b_i \in \mathbb{F}_2$ for all $i \in [n-1]$. In other words, f and g are polynomials with coefficients over the finite field $\mathbb{F}_2$, both of degree n and with nonzero constant term. Denote by $P_n^{1,1}$ the set of all pairs (f, g) of such polynomials. We are interested in the following problem:

Problem 3. *Define* $C_n^{1,1}$ *and* $NC_n^{1,1}$ *as the sets*

$$C_n^{1,1} = \{(f, g) \in P_n^{1,1} : \gcd(f, g) = 1\} \ , \tag{160}$$
$$NC_n^{1,1} = \{(f, g) \in P_n^{1,1} : \gcd(f, g) \neq 1\} \ . \tag{161}$$

What are the cardinalities of $C_n^{1,1}$ *and* $NC_n^{1,1}$*?*

Stated otherwise, we want to count the number of coprime and non-coprime pairs of polynomials in $P_n^{1,1}$.

In what follows, we will also make use of the following notation:

- $P_n^{1,0}$: set of pairs of polynomials of degree n where the first polynomial has nonzero constant term while the second not.
- $P_n^{0,1}$: set of pairs of polynomials of degree n where the first polynomial does not have constant term while the second does.
- $C_n^{1,0}, NC_n^{1,0}$: respectively the set of coprime pairs and the set of non-coprime pairs in $P_n^{1,0}$
- $C_n^{0,1}, NC_n^{0,1}$: respectively the set of coprime pairs and the set of non-coprime pairs in $P_n^{0,1}$

Additionally, in order to avoid burdening the notation, we will refer to a set and its cardinality with the same symbol.

13.2.2 *Equivalence relation based on Euclid's algorithm*

Notice that Problem 3 has already been solved for the general case where there are no constraints on the constant terms, i.e. f and g are just two polynomials of degree n (see [152, 10]). Denoting by C_n and NC_n respectively the sets of coprime and non-coprime pairs of polynomials of degree n with any constant term, it holds that

$$C_n = NC_n = 2^{2n-1} \ , \tag{162}$$

i.e. there are exactly as many coprime pairs as non-coprime pairs of degree n. The idea behind the proof reported in [10] (which can be generalized to any finite field $\mathbb{F}_q$) is that for each non-coprime pair $(f, g) \in NC_n$ one can construct a coprime pair $(f', g') \in C_n$ in the following way:

1. Apply Euclid's algorithm to the pair (f, g). Since f and g are non-coprime, the last remainder will be 0.
2. Replace the last remainder with 1, and reverse Euclid's algorithm using the same sequence of quotients computed for (f, g).
3. By construction, the pair (f', g') obtained at the end of the reverse algorithm will be coprime.

Observe that, if we apply this procedure to a non-coprime pair $(f, g) \in \mathrm{NC}_n^{1,1}$, the polynomials f' and g' in the new coprime pair will not have nonzero constant terms in general, even though not both of them will have null constant term (otherwise they would have a factor x in common).

We thus need to analyze more in detail Euclid's algorithm, in order to see how changing the last remainder affects the constant terms of the intermediate remainders and, in turn, those of f' and g'.

Suppose that r_i and r_{i+1} are two intermediate remainders produced by Euclid's algorithm at step i. In particular, we assume that $r_0 = f$ and $r_1 = g$ are the initial polynomials which form the original pair. Then, at step $i+1$ it holds that

$$r_i(x) = q_{i+1}(x) r_{i+1}(x) + r_{i+2}(x) \ . \tag{163}$$

Remark that if both r_i and r_{i+1} have null constant terms then r_{i+2} will have null constant term as well, independently of the quotient q_{i+1}. Since Euclid's algorithm consists of iteratively applying Equation (163) at each step, it follows that if we start from a pair (f, g) where both f and g have null constant terms then all intermediate remainders in the algorithm will also have null constant terms. Conversely, if we start from a pair (f, g) where at least one of the two polynomials have a nonzero constant term, then both constant terms of all subsequent adjacent remainders pairs r_i, r_{i+1} will not be null at the same time.

More formally, for all steps i in Euclid's algorithm we can interpret the presence/absence of the constant terms in r_i, r_{i+1} as the *state* (c_i, c_{i+1}) of a *finite discrete dynamical system*. In particular, c_i and c_{i+1} respectively denote the constant terms of r_i and r_{i+1}. Since we are interested in pairs where both polynomials have nonzero constant term, we can rule out the possibility that $(c_i, c_{i+1}) = (0, 0)$. Hence, we have that $(c_i, c_{i+1}) \in (\mathbb{F}_2^2)^* = \{(1, 1), (1, 0), (0, 1)\}$ for each step i.

Denoting by $X_{i+1} \in \mathbb{F}_2$ the constant term in q_{i+1}, we can derive the transition function $\delta : (\mathbb{F}_2^2)^* \times \mathbb{F}_2 \to (\mathbb{F}_2^2)^*$ which maps a pair (c_i, c_{i+1}) to the next (c_{i+1}, c_{i+2}) using Equation (163), to assess the presence/absence of the constant term in c_{i+2}. Table 18 reports the transition function δ for all possible $(2^2 - 1) \cdot 2 = 6$ inputs in $(\mathbb{F}_2^2)^* \times \mathbb{F}_2$. The whole dynamical system can also be considered as a *finite state automaton* (FSA), whose transition graph is depicted in

(c_i, c_{i+1})	X_{i+1}	$\delta((c_i, c_{i+1}), X_{i+1})$
$(1,1)$	0	$(1,1)$
$(1,1)$	1	$(1,0)$
$(1,0)$	0	$(0,1)$
$(1,0)$	1	$(0,1)$
$(0,1)$	0	$(1,0)$
$(0,1)$	1	$(1,1)$

Table 18: Truth table for the transition function δ

Figure 48. Consider now a pair of polynomials $(f, g) \in P_n^{1,1}$. The sequence of quotients $q_1, q_2, \cdots$ yielded by Euclid's algorithm induces a path on the FSA graph starting from state $(1,1)$, which is labeled by the constant terms of the quotients. The final state at the end of the path will be either $(1,1)$, $(1,0)$ or $(0,1)$.

What happens if we change the final state to one of the remaining two states, and invert the process with the same sequence of constant terms, reading them in reverse order? Observe from Table 18 that the FSA is a *permutation automaton*, meaning that if we take two distinct states and read the same quotient constant term X_{i+1}, then the two output states after applying δ will be distinct as well. Formally, for all distinct pairs $(c_i, c_{i+1}) \neq (c'_i, c'_{i+1})$ and X_{i+1}, it holds that

$$\delta((c_i, c_{i+1}), X_{i+1}) \neq \delta((c'_i, c'_{i+1}), X_{i+1}) \ . \tag{164}$$

A simple induction argument shows that this permutation property stands also for sequences of constant terms. Thus, if we start from two different initial states and apply the same sequence of constant terms, the final states will be different as well. Clearly, this fact also holds for the inverse automaton, i. e. the FSA with the same transition diagram as that of Figure 48 but with inverted arrows.

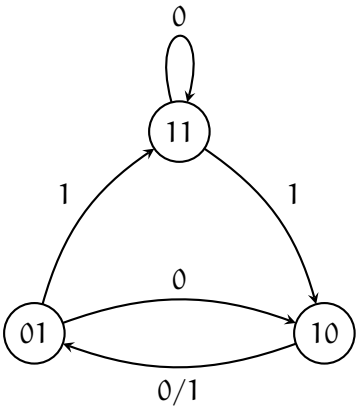


Figure 48: Transition graph for the finite state automaton realizing δ.

As a consequence, from a polynomial pair $(f, g) \in P_n^{1,1}$ we can build two pairs $(f', g') \in P_n^{0,1}$ and $(f'', g'') \in P_n^{1,0}$ as follows: the former having f' without constant term and g' with constant term, and the latter having f'' with constant term and g'' without constant term. These pairs are constructed by changing the constant terms of the last remainders of (f, g) and then by inverting Euclid's algorithm with the same sequence of quotients in reverse order.

In particular, assuming that $c_{k-1}, c_k \in \mathbb{F}_2$ are the constant terms of the last remainders pair (r_{k-1}, r_k) of (f, g) in Euclid's algorithm, the other two pairs (r'_{k-1}, r'_k) and (r''_{k-1}, r''_k) respectively of (f', g') and (f'', g'') can be computed as follows:

$$(r'_{k-1}, r'_k) = \begin{cases} (1 + r_{k-1}, r_k) & \text{, if } (c_{k-1}, c_k) = (1,1) \\ (r_{k-1}, 1 + r_k) & \text{, if } (c_{k-1}, c_k) = (1,0) \\ (1 + r_{k-1}, r_k) & \text{, if } (c_{k-1}, c_k) = (0,1) \end{cases}, \tag{165}$$

$$(r''_{k-1}, r''_k) = \begin{cases} (r_{k-1}, 1 + r_k) & \text{, if } (c_{k-1}, c_k) = (1,1) \\ (1 + r_{k-1}, 1 + r_k) & \text{, if } (c_{k-1}, c_k) = (1,0) \\ (1 + r_{k-1}, 1 + r_k) & \text{, if } (c_{k-1}, c_k) = (0,1) \end{cases}. \tag{166}$$

Consequently, the set of quotients sequences together with the last pairs of remainders without constant terms induce an *equivalence relation*, where each equivalence class is of size $2^2 - 1 = 3$ and contains respectively a pair $(f, g) \in P_n^{1,1}$, a pair $(f', g') \in P_n^{0,1}$ and a pair $(f'', g'') \in P_n^{1,0}$. Thus, this equivalence relation partitions the set $P = P_n^{1,1} \cup P_n^{0,1} \cup P_n^{1,0}$ containing polynomial pairs of degree n where at least one of the two polynomials has a nonzero constant term. Moreover, we can conclude that each equivalence class of size 3 contains *exactly one* non-coprime pair, since only one pair can have a path ending in $(1, 0)$, due to the permutation property of the FSA.

13.2.3 *Counting (non-) coprime pairs*

The problem with the equivalence relation described in the previous section is that, unfortunately, it does not tell us which of the three pairs of an equivalence class is coprime. However, we can remark the following facts:

- The cardinality of each set $P_n^{1,1}$, $P_n^{1,0}$ and $P_n^{0,1}$ is $2^{2(n-1)}$, since it just consists of all pairs of vector coefficients $(a_1, \cdots, a_{n-1})$ and $(b_1, \cdots, b_{n-1})$ which describes the two polynomials composing the pair.

- Consequently, the cardinality of the support set P of the equivalence relation is $3 \cdot 2^{2(n-1)}$, since the three sets $P_n^{1,1}$, $P_n^{1,0}$ and $P_n^{0,1}$ are clearly disjoint.
- From the remark above, we thus have the following equation:

$$NC_n^{1,1} + C_n^{1,1} = NC_n^{0,1} + C_n^{0,1} = NC_n^{1,0} + C_n^{1,0} = 2^{2(n-1)} \tag{167}$$

- Additionally, we can derive the following system from the equivalence relation:

$$\begin{cases} C_n^{1,1} = NC_n^{0,1} + NC_n^{1,0} \\ C_n^{1,0} = NC_n^{1,1} + NC_n^{0,1} \\ C_n^{0,1} = NC_n^{1,1} + NC_n^{1,0} \end{cases} \tag{168}$$

- Finally, if the order of the polynomials in a (non-)coprime pair $(f', g') \in P_n^{1,0}$ is swapped, then $(g', f') \in P_n^{0,1}$. As a consequence, we also have that

$$\begin{cases} NC_n^{1,0} = NC_n^{0,1} \\ C_n^{1,0} = C_n^{0,1} \end{cases} \tag{169}$$

- Thus, putting together (168) and (169) we get

$$C_n^{1,1} = 2 \cdot NC_n^{1,0} \tag{170}$$

Therefore, we need to determine the size of the set $NC_n^{1,0}$ of non-coprime pairs of degree n where the first polynomial has a nonzero constant term while the second not. A possible way is to express $NC_n^{1,0}$ in terms of the size of $NC_n^{1,1}$. In particular, by conducting some experiments with the MAGMA computer algebra system, the following conjecture seems to hold for any $n > 2$:

Conjecture 2. *Let $NC_n^{1,0}$ denote the number of non-coprime pairs of polynomials (f, g) over $\mathbb{F}_2$ of degree $n > 2$ where only f has nonzero constant term, and let $NC_n^{1,1}$ be the number of non-coprime pairs of polynomials of the same form except that both f and g have nonzero constant term. Then,*

$$NC_n^{1,0} = NC_n^{1,1} - 1 \ . \tag{171}$$

As far as we know, there are no results in the literature that prove Equation (171). However, since Conjecture 2 seems to be confirmed by our computer experiments, let us take it for granted for the time being. In the next section, we will formally prove it using a different method. Combining Equations (167), (170) and (171), one finally gets the following system:

$$\begin{cases} C_n^{1,1} = 2 \cdot (NC_n^{1,1} - 1) \\ NC_n^{1,1} = 2^{2(n-1)} - C_n^{1,1} \end{cases} \tag{172}$$

which yields

$$C_n^{1,1} = 2 \cdot \frac{2^{2(n-1)} - 1}{3} \ , \tag{173}$$

$$NC_n^{1,1} = 2 \cdot \frac{2^{2n-3} + 1}{3} \ . \tag{174}$$

Notice that Equation (173) counts all *ordered* coprime pairs of polynomials in $P_n^{1,1}$. To get the number of *distinct* coprime pairs $DC_n^{1,1}$ one simply needs to divide it by 2, thus obtaining

$$DC_n^{1,1} = \frac{2^{2(n-1)} - 1}{3} = \frac{4^{n-1} - 1}{3} \ . \tag{175}$$

In the next section, we prove the same formula in Equation (175) using a different method, which does not rely on Conjecture 2.

13.2.4 *Counting* $(1,1)$ *coprime pairs by recurrence*

We now solve Problem 3 using a recurrence equation.

First of all, observe that the number of pairs in $P_n^{1,1}$ can be expressed as follows:

$$P_n^{1,1} = C_n^{1,1} + NC_n^{1,1} = C_n^{1,1} + S_n^{1,1} + DNC_n^{1,1} \ , \tag{176}$$

where $S_n^{1,1}$ and $DNC_n^{1,1}$ respectively denote the number of symmetric pairs (f, f) and the number of distinct non coprime pairs in $P_n^{1,1}$.

Clearly, it results that $S_n^{1,1} = 2^{n-1}$, since we just need to count the central coefficients $a_1, \cdots, a_{n-1} \in \mathbb{F}_2$ of f. On the other hand, if $\gcd(f, g) \neq 1$, then the greatest common divisor of f and g has form

$$h(x) = 1 + h_1 x + \cdots + h_{d-1} + x^d \ , \tag{177}$$

where $d \in \{1, \cdots, n-1\}$. This means that

$$f(x) = h(x) \cdot p(x) \tag{178}$$

$$g(x) = h(x) \cdot q(x) \tag{179}$$

where $(p, q) \in C_{n-d}^{1,1}$, that is, p and q are coprime polynomials of degree $n - d$ both having nonzero constant term. Since there are 2^{d-1} greatest common divisors for each degree $d \in \{1, \cdots, n-1\}$, it follows that the number of distinct non-coprime pairs $DNC_n^{1,1}$ of degree n is determined by the sum of all coprime pairs $C_{n-d}^{1,1}$ having degree $n - d \in \{1, \cdots, n-1\}$, where each term is multiplied by 2^{d-1}. In other words, it results that

$$DNC_n^{1,1} = \sum_{d=1}^{n-1} 2^{d-1} \cdot C_{n-d}^{1,1} \ . \tag{180}$$

Since $P_n^{1,1} = 2^{2(n-1)}$, Equation (176) can be rewritten as:

$$C_n^{1,1} = 2^{2(n-1)} - 2^{n-1} - \sum_{d=1}^{n-1} 2^{d-1} \cdot C_{n-d}^{1,1} \ . \tag{181}$$

By Equation (181), the difference between $C_n^{1,1}$ and $C_{n-1}^{1,1}$ equals

$$\begin{aligned} C_n^{1,1} - C_{n-1}^{1,1} = {} & 2^{2(n-1)} - 2^{2^{n-2}} - 2^{n-1} + 2^{n-2} - \\ & - \sum_{d=1}^{n-1} 2^{d-1} \cdot C_{n-d}^{1,1} + \sum_{d=1}^{n-2} 2^{d-1} \cdot C_{n-1-d}^{1,1} \ . \end{aligned} \tag{182}$$

Extracting the first term from the first sum in (182) yields

$$\begin{aligned} \cancel{C_n^{1,1} - C_{n-1}^{1,1}} = {} & 2^{2(n-1)} - 2^{2^{n-2}} - 2^{n-1} + 2^{n-2} \cancel{- 2^0 \cdot C_{n-1}^{1,1}} - \\ & - \sum_{d=2}^{n-1} 2^{d-1} \cdot C_{n-d}^{1,1} + \sum_{d=1}^{n-2} 2^{d-1} \cdot C_{n-1-d}^{1,1} \ . \end{aligned} \tag{183}$$

Then, by reorganizing the terms of the two sums in (183) one obtains

$$\begin{aligned} & - \sum_{d=2}^{n-1} 2^{d-1} \cdot C_{n-d}^{1,1} + \sum_{d=1}^{n-2} 2^{d-1} \cdot C_{n-1-d}^{1,1} \quad = \\ & - 2^1 \cdot C_{n-2}^{1,1} - 2^2 \cdot C_{n-3}^{1,1} - \cdots - 2^{n-2} \cdot C_1^{1,1} \ + \\ & + 2^0 \cdot C_{n-2}^{1,1} + 2^1 \cdot C_{n-3}^{1,1} + \cdots + 2^{n-3} \cdot C_1^{1,1} \ = \\ = {} & - \sum_{d=1}^{n-2} 2^{d-1} \cdot C_{n-1-d}^{1,1} \ . \end{aligned} \tag{184}$$

which means that Equation (183) can be rewritten as

$$C_n^{1,1} = 2^{2(n-1)} - 2^{2^{n-2}} - 2^{n-1} + 2^{n-2} - \sum_{d=1}^{n-2} 2^{d-1} \cdot C_{n-1-d}^{1,1} \ . \tag{185}$$

If one iterates the above procedure by subtracting $C_{n-2}, C_{n-3}, \cdots, C_1^{1,1}$ from $C_n^{1,1}$, the following result is finally obtained:

$$C_n^{1,1} = 2^{2(n-1)} - \sum_{i=0}^{n-2} 2^{2i} - 2^{n-1} + \sum_{i=0}^{n-2} 2^i \ . \tag{186}$$

The two sums in Equation (186) evaluate to the following expressions:

$$\sum_{i=0}^{n-2} 2^{2i} = \frac{4^{n-1} - 1}{3} \tag{187}$$

$$\sum_{i=0}^{n-2} 2^i = 2^{n-1} - 1 \tag{188}$$

$$\tag{189}$$

Hence, the number of coprime pairs $C_n^{1,1}$ corresponds to:

$$C_n^{1,1} = 4^{n-1} - \frac{4^{n-1}-1}{3} - 2^{n-1} + 2^{n-1} - 1 = \\ = 4^{n-1} - 1 - \frac{4^{n-1}-1}{3} = 2 \cdot \frac{4^{n-1}-1}{3} \ . \tag{190}$$

Dividing by 2, one finally obtains the number of distinct coprime polynomial pairs:

$$DC_n^{1,1} = \frac{4^{n-1}-1}{3} \ , \tag{191}$$

We summarize the above discussion in the following theorem:

Theorem 31. *The number of distinct coprime pairs of polynomials* (f, g) *over* $\mathbb{F}_2$ *of degree* n *where both* f *and* g *have nonzero constant term, or equivalently the number of pairs of orthogonal Latin squares having order* 2^{n-1} *that are generated by linear bipermutive CA of diameter* $n+1$ *is:*

$$DC_n^{1,1} = \frac{4^{n-1}-1}{3} \ . \tag{192}$$

Let $a(n)$ be the integer sequence defined by Equation (192). Then, the first terms of this sequence for $n \geqslant 1$ are:

$$a(n) = 0, 1, 5, 21, 85, 341, 1365, \cdots \tag{193}$$

which is a shifted version of OEIS sequence A002450 [84], defined by

$$b(n) = \frac{4^n - 1}{3} \ . \tag{194}$$

In particular, it is easily seen that $b(n) = C_{n+1}^{1,1}$, i. e. $b(n)$ corresponds to the number of coprime pairs of polynomials of degree $n+1$ over $\mathbb{F}_2$ where both polynomials have nonzero constant term. Sequence A002450 is known for several other facts that are not related to polynomials or orthogonal Latin squares arising from linear CA, such as:

- The *Collatz function* [103] applied to one of the terms of $b(n)$ reaches the periodic point 1 after $2n$ iterations.
- $b(n+1)$ is the number of steps performed when generating all n-step walks starting from the origin of the square lattice $\mathbb{Z}^2$, where the possible directions are $\uparrow, \downarrow, \leftarrow, \rightarrow$.
- $b(n)$ corresponds to the *Lucas sequence* $U_n(5, 4)$ [11].

For the number of distinct non-coprime pairs $DNC_n^{1,1}$, one has first to remove the symmetric pairs (f, f) from Equation (191) and then divide the result by 2, which yields

$$DNC_n^{1,1} = \frac{1}{2}\left(\frac{4^{n-1}-1}{3} + 1 - 2^{n-1}\right) = \\ = \frac{2^{2n-3} - 3 \cdot 2^{n-2} + 1}{3} = \frac{(2^{n-1}-1)(2^{n-2}-1)}{3} \ . \tag{195}$$

Notice that Equation (195) corresponds to the *Gaussian binomial coefficient* $\binom{n-1}{2}_2$ [149]. In the general case, the integer sequence corresponding to $\binom{n}{2}_2$ for $n \in \mathbb{N}$ equals

$$c(n) = \binom{n}{2}_2 = \frac{(2^n - 1)(2^{n-1} - 1)}{3} \ , \tag{196}$$

which corresponds to OEIS sequence A006095 [86]. Comparing with Equation (195), it follows that $c(n) = \mathrm{DNC}^{1,1}_{n+1}$ for all $n > 0$, i. e. $c(n)$ is the number of non-coprime pairs of polynomials of degree $n+1$ over $\mathbb{F}_2$ where both polynomials have nonzero constant term.

Finally, observe that one can also compute $C^{1,0}_n$ using Theorem (31). The only difference is that there are no symmetric pairs for the case $(1, 0)$ (hence the term 2^{n-1} is not present in the recurrence equation). In particular, this yields

$$C^{1,0}_n = 4^{n-1} - \frac{4^{n-1} - 1}{3} \ , \tag{197}$$

from which one can deduce that

$$C^{1,0}_n - C^{1,1}_n = 1 \ , \tag{198}$$

or equivalently,

$$\mathrm{NC}^{1,0}_n = \mathrm{NC}^{1,1}_n - 1 \ , \tag{199}$$

which also proves Conjecture 2 reported in the previous section.

13.3 MOLS BASED ON LINEAR CA

In the literature related to Latin squares, a natural question is to determine the maximum number of MOLS for a given order. In this section, we tackle this question for MOLS generated by linear CA over $\mathbb{F}_q$, thus generalizing the counting problem addressed in the previous section for $q = 2$. Given $n \in \mathbb{N}$ we consider in particular the following two problems:

Problem 4. *What is the maximum number $N_{n,q}$ of linear bipermutive CA over $\mathbb{F}_q$ of diameter $n+1$ whose Latin squares are mutually orthogonal? As shown in Section 13.1, this actually amounts to compute the maximum number of monic pairwise coprime polynomials of degree n and nonzero constant term over $\mathbb{F}_q$.*

Problem 5. *How many maximal sets of $N_{n,q}$ MOLS generated by linear CA do there exist?*

In the remainder of this section, we present a construction for sets of MOLS based on linear CA defined by pairwise coprime polynomials over $\mathbb{F}_q$, conjecturing its optimality by empirical observations.

13.3.1 *MOLS from Irreducible Polynomials*

To formalize our discussion, let

$$S_{n,q} = \{a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + x^n : a_i \in \mathbb{F}_q, a_0 \neq 0\} \ , \quad (200)$$

i. e. $S_{n,q}$ is the set of all degree n monic polynomials $f \in \mathbb{F}_q[x]$ with nonzero constant term a_0. Moreover, let

$$\mathcal{M}_{n,q} = \{\mathcal{A}_{n,q} \subseteq S_{n,q} : \forall f \neq g \in \mathcal{A}_{n,q}, \gcd(f,g) = 1\} \ . \quad (201)$$

In other words, $\mathcal{M}_{n,q}$ is the family of subsets of $S_{n,q}$ of pairwise coprime polynomials. In order to solve Problem 4, we have to determine the maximal cardinality of the subsets in $\mathcal{M}_{n,q}$, that is

$$N_{n,q} = \max_{\mathcal{A}_{n,q} \in \mathcal{M}_{n,q}} \{|\mathcal{A}_{n,q}|\} \ . \quad (202)$$

On the other hand, for Problem 5 we want to count how many sets in $\mathcal{M}_{n,q}$ have cardinality $N_{n,q}$:

$$T_{n,q} = |\{\mathcal{A}_{n,q} \in \mathcal{M}_{n,q} : |\mathcal{A}_{n,q}| = N_{n,q}\}| \quad (203)$$

We begin by considering the set $\mathcal{I}_{n,q}$ of irreducible polynomials of degree n over $\mathbb{F}_q$, all of which are trivially pairwise coprime. Hence, $\mathcal{I}_{n,q}$ is included in all subsets having maximum cardinality $N_{n,q}$. *Gauss' formula* [68] can be used to count the number of irreducible polynomials of degree n:

$$I_{n,q} = |\mathcal{I}_{n,q}| = \frac{1}{n} \sum_{d|n} \mu(d) \cdot q^{\frac{n}{d}} \ , \quad (204)$$

where μ denotes the *Möbius function*. Let $d = \rho_1^{\alpha_1} \rho_2^{\alpha_2} \cdots \rho_k^{\alpha_k}$ be the prime factorization of $d \in \mathbb{N}$. Then, d is called *square-free* (s.f.) if $\alpha_i = 1$ for all $i \in \{1, \cdots, k\}$, i. e. if d is not divisible by any prime power with exponent higher than 1. The Möbius function of d is defined as:

$$\mu(d) = \begin{cases} 1 & \text{, if } d \text{ is s.f. and has an even number of prime factors} \\ -1 & \text{, if } d \text{ is s.f. and has an odd number of prime factors} \\ 0 & \text{, if } d \text{ is not s.f.} \end{cases} \quad (205)$$

We thus have that

$$N_{n,q} \geqslant I_{n,q} \ . \quad (206)$$

In order to refine this lower bound, we have to determine how many other (reducible) polynomials of degree n one can add to $\mathcal{I}_{n,q}$ so that the resulting set only includes pairwise coprime polynomials. Consider the following construction:

CONSTRUCTION-IRREDUCIBLE(n, q)

INITIALIZATION: Initialize set $\mathcal{P}_{n,q}$ to $\mathcal{I}_{n,q}$

LOOP: For all $1 \leqslant k \leqslant \left\lfloor \frac{n}{2} \right\rfloor$ do:

1. Build set $\mathcal{P}'_{k,q}$ by multiplying each polynomial in $\mathcal{I}_{k,q}$ with a distinct polynomial in $\mathcal{I}_{n-k,q}$
2. Add set $\mathcal{P}'_{k,q}$ to $\mathcal{P}_{n,q}$

OUTPUT: return $\mathcal{P}_{n,q}$

Hence, set $\mathcal{P}$ is constructed by first adding all irreducible polynomials of degree n, then by adding the set of all irreducible polynomials of degree 1 multiplied by as many irreducible polynomials of degree $n-1$, the set of all irreducible polynomials of degree 2 multiplied by as many irreducible polynomials of degree $n-2$, and so on. One can see that all polynomials added to $\mathcal{P}$ in this way are pairwise coprime, since they all have distinct irreducible factors.

13.3.2 *Lower Bounds on Linear CA-based MOLS*

Remark that the procedure CONSTRUCTION-IRREDUCIBLE can be iterated only up to $k \leqslant \left\lfloor \frac{n}{2} \right\rfloor$, because by symmetry the irreducible polynomials of degree $n-k$ with $k > \left\lfloor \frac{n}{2} \right\rfloor$ correspond to those of degree $k \leqslant \left\lfloor \frac{n}{2} \right\rfloor$. Notice also that, when n is even, the last step of the procedure consists of squaring all irreducible polynomials of degree $\frac{n}{2}$.

Hence, we have shown that the set $\mathcal{P}$ which is generated by procedure CONSTRUCTION-IRREDUCIBLE is indeed a member of the family $\mathcal{M}_{n,q}$. The cardinality of such set is given by

$$C_{n,q} = |\mathcal{P}_{n,q}| = I_{n,q} + \sum_{k=1}^{\left\lfloor \frac{n}{2} \right\rfloor} I_{k,q} \ . \tag{207}$$

As a matter of fact, beside the initial step when one adds all irreducible polynomials of degree n to $\mathcal{P}$, in each iteration k of the loop the number of polynomials that one can obtain by multiplying two irreducible factors is bounded by the number of irreducible polynomials of degree k, which is $I_{k,q}$. We have thus obtained the following result, which gives a more precise lower bound on $N_{n,q}$:

Theorem 32. *The maximum number of pairwise coprime polynomials of degree n over $\mathbb{F}_q$ with nonzero constant term is at least $C_{n,q}$:*

$$N_{n,q} \geqslant C_{n,q} \ . \tag{208}$$

A natural question arising from Theorem 32 is whether the above construction is optimal, i. e. if the maximum number of pairwise coprime polynomials $N_{n,q}$ is actually equal to $C_{n,q}$. Experimentally, we verified this hypothesis by exhaustively generating all maximal sets of pairwise coprime polynomials for $q = 2$ up to degree $n = 6$. Hence, this leads to the following conjecture:

Conjecture 3. *The maximum number of MOLS generated by bipermutive linear CA with local rule of diameter $n+1$, or equivalently the maximum number of pairwise coprime polynomials of degree n over $\mathbb{F}_q$ with nonzero constant term is $C_{n,q}$, that is,*

$$N_{n,q} = C_{n,q} \ . \tag{209}$$

We now consider how many sets of pairwise coprime polynomials of degree n one can obtain through the above construction. As observed before, the set $\mathcal{I}_{n,q}$ of irreducible polynomials of degree n is included in each $A_{n,q} \in \mathcal{M}_{n,q}$ having maximal cardinality. Hence, this set does not yield any choice from the combinatorial point of view. Additionally, remark that for all steps $k \in \left\{1, \cdots, \left\lfloor \frac{n}{2} \right\rfloor\right\}$ the set $\mathcal{P}'$ is constructed by multiplying each irreducible polynomial $f(x)$ in $\mathcal{I}_{k,q}$ with a distinct irreducible polynomial $g(x)$ in $\mathcal{I}_{n-k,q}$. Since the sequence $\{I_{k,q}\}_{n\in\mathbb{N}}$ is monotonically non-decreasing [107], this means that $I_{n-k,q} \geqslant I_{k,q}$ for all $n \in \mathbb{N}$ and $1 \leqslant k \leqslant \left\lfloor \frac{n}{2} \right\rfloor$. Consequently, the number of possible choices for the subsets of irreducible polynomials of degree $n-k$ to be multiplied with the irreducible polynomials of degree k is

$$P'_{k,q} = \binom{I_{n-k,q}}{I_{k,q}} \tag{210}$$

Observe that for distinct degrees $k_1, k_2 \in \left\{1, \cdots, \left\lfloor \frac{n}{2} \right\rfloor\right\}$ the choices for building the sets $\mathcal{P}'_{k_1,q}$ and $\mathcal{P}'_{k_1,q}$ are independent. This means that each $\mathcal{P}'_{k_1,q}$ can be combined with each $\mathcal{P}'_{k_1,q}$, so the number of possible choices in this case is

$$R'_{k_1,k_2} = P'_{k_1,q} \cdot P'_{k_2,q} \ . \tag{211}$$

Combining Equations (210) and (211), one finally obtains that the number of maximal families of pairwise coprime polynomials of degree n over $\mathbb{F}_q$ with nonzero constant term is:

$$D_{n,q} = \prod_{k=1}^{\left\lfloor \frac{n}{2} \right\rfloor} P'_{k,q} = \prod_{k=1}^{\left\lfloor \frac{n}{2} \right\rfloor} \binom{I_{n-k,q}}{I_{k,q}} \ . \tag{212}$$

In conclusion, one also obtains the following result for $T_{n,q}$:

Theorem 33. *The number of maximal families of pairwise coprime polynomials of degree n over $\mathbb{F}_q$ with nonzero constant term is at least $D_{n,q}$, i. e.*

$$T_{n,q} \geqslant D_{n,q} \ . \tag{213}$$

13.4 CONCLUSIONS

In this chapter, we undertook an investigation of orthogonal Latin squares generated through CA, motivated by the design of threshold secret sharing schemes and perfect authentication codes.

First, we proved that the global rule of any bipermutive CA of diameter d and length $2(d-1)$ can be used to generate a Latin square of order $N = q^{d-1}$, with q being the size of the state alphabet. We then focused on orthogonal Latin squares generated by linear bipermutive CA, showing a characterization result based on the Sylvester matrix induced by two linear local rules. In particular, we proved that two linear bipermutive CA generate orthogonal Latin squares if and only if the polynomials associated to their local rules are relatively prime. Next, we described a $(2, n)$–threshold scheme and a perfect authentication code based on this characterization results.

In the second part of the chapter, we addressed the problem of counting the number of linear CA pairs over $\mathbb{F}_2$ generating orthogonal Latin squares, i. e. the number of coprime polynomial pairs (f, g) of degree n over $\mathbb{F}_2$ where both f and g have nonzero constant term. We presented two approaches to solve this problem, namely an equivalence relation based on Euclid's algorithm and a recurrence equation. In particular, the former approach relies on a conjecture about the number of non-coprime polynomial pairs which we finally proved with the recurrence equation approach. Moreover, we remarked that the integer sequence generated by the closed-form formula of the recurrence equation correspond to A002450, a sequence which is already known in the OEIS for several other facts not related to polynomials or orthogonal Latin squares.

In the last part of the chapter, we presented a construction based on irreducible polynomials which gives a lower bound on the number of MOLS generated by linear CA over $\mathbb{F}_q$. In particular, computer search experiments for $q = 2$ seem to confirm that this lower bound is always satisfied with equality, for which reason we conjectured its optimality. Finally, we also derived a lower bound on the number of maximal families of MOLS induced by the proposed construction.

ORTHOGONAL LATIN SQUARES BASED ON NONLINEAR CA

14

In Chapter 13 we investigated the conditions under which linear CA generate orthogonal Latin squares, motivated by the goal of designing a $(2, n)$–threshold scheme. However, remark that secret sharing schemes based on *linear constructions* (i. e. where the secret is determined by a linear combination of the shares) are vulnerable to *cheaters*. In particular, suppose that a dishonest player submits a fake share during the recovery phase. Clearly, all the other players in the protocol will obtain an incorrect value of the secret. On the other hand, if the scheme is linear then the cheater can recover the real secret for himself by using the incorrect value obtained during the recovery phase and the original share received from the dealer. Secret sharing schemes which are resilient toward this kind of attack are also called *cheater-immune* [181]. In the case of a $(2, n)$–threshold access structure, such schemes are equivalent to orthogonal Latin squares based on *nonlinear constructions*.

For this reason, in this chapter we investigate orthogonal Latin squares induced by *nonlinear* bipermutive CA. In particular, we adopt both a combinatorial methodology to enumerate all pairs of bipermutive local rules which generate OLS up to diameter $d = 6$ and classify them with respect to their nonlinearity, and a heuristic approach to evolve such pairs of local rules with diameter $d = 7, 8$ through Genetic Algorithms (GA) and Genetic Programming (GP).

Beside the cryptographic motivation of cheater-immune SSS, the goal of this research is twofold: first, we aim at understanding the mathematical structure of OLS generated by nonlinear bipermutive CA, and in particular determine which kind of conditions characterize them. This could have possible applications also in coding theory, especially in the design of nonlinear MDS codes. Second, as far as the authors are aware, there have been no attempts in the literature to apply evolutionary computation (EC) algorithms for the design of OLS. The closest example one can find is a work by Safadi et al. [160] where GA were used to evolve. Ashlock [4] also used GA to generate other kinds of combinatorial designs, but not OLS. The reason for this gap in the literature could lie in the difficulty of designing a suitable encoding for the feasible solutions handled by EC algorithms. Indeed, it is not simple to optimize the orthogonality of two Latin squares while simultaneously preserving their row-column permutation property using stochastic operators like crossover and mutation. As a consequence, evolving OLS based on nonlinear bipermutive CA

could also be an interesting benchmark problem for heuristic optimization algorithms like GA and GP.

In particular, for the optimization problem of evolving CA-based OLS we leverage on the fact that bipermutive rules of d variables are defined by *generating functions* of $d-2$ variables. Hence, the genotype of each individual in the population represents a pair of generating functions, thus ensuring that the corresponding phenotype of the candidate solution is a pair of bipermutive CA producing two Latin squares. Consequently, we can focus the optimization effort of GA and GP on the orthogonality and nonlinearity properties of the solutions, without checking the row-column permutation constraint.

The rest of this chapter is organized as follows. In Section 14.1 we first prove that the basic reflection and complement operations on local rules preserve the orthogonality relation of the resulting Latin squares. Then, we show that two bipermutive local rules giving rise to orthogonal Latin squares must be *pairwise balanced*, which basically means that the four pairs $(0,0)$, $(1,0)$, $(0,1)$ and $(1,1)$ must occur an equal number of times in the superposition of their truth tables. Additionally, we prove that pairwise balancedness is a property preserved from the generating functions to the corresponding bipermutive rules, but not vice versa. In Section 14.2 we derive a formula for the number of pairwise balanced bipermutive rules, and apply a combinatorial algorithm to enumerate all such pairs. Next, we classify the pairs resulting in orthogonal Latin squares with respect to their nonlinearity values. Section 14.3 presents the four encodings for the genotype of the candidate solutions used in GA and GP, and describes the ad-hoc genetic operators for pairwise balanced generating functions. Section 14.4 describes the experimental setting adopted for our GA and GP algorithms and discusses the obtained results. Finally, in Section 14.5 we sum up the contributions of this chapter.

14.1 COMBINATORIAL APPROACH

Since the problem of characterizing pairs of bipermutive CA which generate orthogonal Latin squares has already been solved in Chapter 13 when the underlying local rules are linear, we now consider the more general case of nonlinear bipermutive CA. In order to tackle this problem, in this section we prove some results that allow us to reduce the search space of all bipermutive functions pairs. Then, we use these results to enumerate all pairs of bipermutive CA generating orthogonal Latin squares up to $d = 6$ variables and to evolve them through GA and GP for $d = 7, 8$.

14.1.1 *Invariance Under Reflection and Complement*

Let $\mathcal{B}_d$ be the set of all pairs of bipermutive Boolean functions of d variables. As bipermutive functions are defined by their generating functions of $d-2$ variables (see Section 2.3), for all $d \geqslant 2$ it follows that $|\mathcal{B}_d| = |\mathcal{G}_d|$, where $\mathcal{G}_d = \{(\varphi, \gamma) \in \mathcal{F}_{d-2} \times \mathcal{F}_{d-2}\}$. Since the cardinality of $\mathcal{F}_{d-2}$ is $2^{2^{d-2}}$, the size of $\mathcal{G}_d$ is $2^{2^{d-2}} \cdot 2^{2^{d-2}} = 2^{2^{d-1}}$, which means that the set $\mathcal{G}_d$ is isomorphic to $\mathcal{F}_{d-1}$, i.e. the space of Boolean functions of $d-1$ variables.

Clearly, if two bipermutive CA induced by a pair of local rules (f, g) give rise to orthogonal Latin squares, then the CA defined by the swapped pair (g, f) will generate the same orthogonal Latin squares in reverse order. We now show that the basic transformations of reflection and complement mentioned in Section 12.2 preserve the orthogonality relation as well:

Lemma 19. *Let $F, G : \mathbb{F}_2^{2(n-1)} \to \mathbb{F}_2^{d-1}$ be two bipermutive CA respectively defined by local rules $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$ of d variables, and let S_F, S_G be the associated Latin squares of order 2^{d-1}. Additionally, let F_R, G_R and F_C, G_C be the CA respectively defined by the reverses f_R, g_R and the complements f_C, g_C of f, g, and let S_{F_R}, S_{G_R} and S_{F_C}, S_{G_C} be the corresponding Latin squares. Then, the following hold:*

- *S_F and S_G are orthogonal if and only if S_{F_R}, S_{G_R} are orthogonal.*
- *S_F and S_G are orthogonal if and only if S_{F_C}, S_{G_C} are orthogonal.*

Proof. Since both reflection and complement are idempotent transformations, it suffices to show only one direction of the implications, i.e. assuming that S_F and S_G are orthogonal. This means that

$$(F(x\|y), G(x\|y)) \neq (F(x'\|y'), G(x'\|y'))$$

for all distinct pairs $(x, y), (x', y') \in \mathbb{F}_2^{d-1} \times \mathbb{F}_2^{d-1}$, since the mapping ϕ which associates binary vectors of length $d-1$ to positive integers in the range $\{1, \cdots, 2^{d-1}\}$ is bijective.

Let us now consider the CA F_R induced by the reflected local rule f_R. Then, for all $(x, y) \in \mathbb{F}_2^{d-1} \times \mathbb{F}_2^{d-1}$ with $x = (x_1, \cdots, x_{d-1})$ and $y = (y_1, \cdots, y_{d-1})$, it follows that

$$\begin{aligned} F_R(x\|y) &= (f_R(x_1, \cdots, x_{d-1}, y_1), \cdots, f_R(x_{d-1}, y_1, \cdots, y_{d-1})) = \\ &= (f(y_1, x_{d-1}, \cdots, x_1), \cdots, f(y_{d-1} \cdots, y_1, x_{d-1})) = \\ &= F(y_R\|x_R)_R \ , \end{aligned}$$

i.e., the output value of the reflected CA F_R is obtained by computing the reflected output of F evaluated on the reflected input $y_R\|x_R$. Analogously, the same fact holds for G_R with respect to G. Since for all $(x, y), (x', y') \in \mathbb{F}_2^{d-1} \times \mathbb{F}_2^{d-1}$ such that $(x, y) \neq (x', y')$ one has that $(y_R, x_R) \neq (y'_R, x'_R)$, it follows that

$$(F(y_R\|x_R)_R, G(y_R\|x_R)_R) \neq (F(y'_R\|x'_R)_R, G(y'_R\|x'_R)_R) \ ,$$

which means that S_{F_R} and S_{G_R} are orthogonal Latin squares.

Next, let us consider the CA F_C induced by the complemented local rule f_C. The output value of F_C over $x\|y$ is

$$\begin{aligned} F_C(x\|y) &= (f_c(x_1,\cdots,x_{d-1},y_1),\cdots,f_c(x_{d-1},y_1,\cdots,y_{d-1})) = \\ &= (1\oplus f(x_1,\cdots,x_{d-1},y_1),\cdots,1\oplus f(x_{d-1},y_1,\cdots,y_{d-1})) = \\ &= \underline{1}\oplus F(x\|y)\ , \end{aligned}$$

where $\underline{1} = (1,\cdots,1) \in \mathbb{F}_2^{d-1}$. Similarly for G_C, one has $G_C(x\|y) = \underline{1}\oplus G(x\|y)$. Given two pairs $(x,y),(x',y') \in \mathbb{F}_2^{d-1}\times\mathbb{F}_2^{d-1}$ such that $(x,y)\neq(x',y')$, it clearly holds that $(\underline{1}\oplus x,\underline{1}\oplus y)\neq(\underline{1}\oplus x',\underline{1}\oplus y')$, from which it follows

$$(\underline{1}\oplus F(x\|y),\underline{1}\oplus G(x\|y))\neq(\underline{1}\oplus F(x'\|y'),\underline{1}\oplus G(x'\|y'))\ .$$

As a consequence, the Latin squares S_{F_C} and S_{G_C} are orthogonal. □

14.1.2 *Pairwise Balancedness*

We now turn to analyze the truth tables of bipermutive rules whose CA generate orthogonal Latin squares. As an example, consider the pair of functions $f,g:\mathbb{F}_2^3\to\mathbb{F}_2$ defined as $f(x_1,x_2,x_3)=x_1\oplus x_3$ and $g(x_1,x_2,x_3)=x_1\oplus x_2\oplus x_3$, namely rules 90 and 150 using Wolfram's numbering convention. The Latin squares of order $d=4$ induced by the corresponding bipermutive CA $F,G:\mathbb{F}_2^4\to\mathbb{F}_2^2$ are orthogonal, since by Theorem 29 f and g are linear and their associated polynomials $P_f(x)=1+x^2$ and $P_g(X)=1+x+x^2$ are coprime. The truth tables $\Omega(f),\Omega(g)\in\mathbb{F}_2^8$ are the following:

$$\Omega(f)=(0,1,0,1,1,0,1,0)\ , \tag{214}$$

$$\Omega(g)=(0,1,1,0,1,0,0,1)\ . \tag{215}$$

Placing side by side these truth tables, one can see that there are $2^{3-2}=2$ occurrences of each of the four pairs $(0,0)$, $(1,0)$, $(0,1)$ and $(1,1)$. We define this property as *pairwise balancedness*:

Definition 34. *Two Boolean functions $f,g:\mathbb{F}_2^n\to\mathbb{F}_2$ of d variables are* pairwise balanced *if the $(n,2)$-function $(f,g):\mathbb{F}_2^n\to\mathbb{F}_2^2$ defined as $(f,g)(x)=(f(x),g(x))$ is balanced, that is $|(f,g)^{-1}(y_1,y_2)|=2^{d-2}$ for all $(y_1,y_2)\in\mathbb{F}_2^2$.*

We now prove that pairwise balancedness is a necessary condition for a pair of bipermutive local rules whose CA generate orthogonal Latin squares:

Lemma 20. *Let $F,G:\mathbb{F}_2^{2(n-1)}\to\mathbb{F}_2^{d-1}$ be bipermutive CA respectively induced by local rules $f,g:\mathbb{F}_2^n\to\mathbb{F}_2$, and suppose that the associated Latin squares S_F, S_G are orthogonal. Then, f and g are pairwise balanced.*

Proof. Let $H : \mathbb{F}_2^{d-1} \times \mathbb{F}_2^{d-1} \to \mathbb{F}_2^{d-1} \times \mathbb{F}_2^{d-1}$ be the function defined as $H(x, y) = (F(x\|y), G(x\|y))$ for all $(x, y) \in \mathbb{F}_2^{d-1} \times \mathbb{F}_2^{d-1}$. Since S_F and S_G are orthogonal, it follows that H is bijective.

Consider two vectors $c, d \in \mathbb{F}_2^{d-1}$ and suppose that their first components, namely c_1 and d_1, are fixed. We want to compute the number of preimages $(x_1, \cdots, x_{d-1}, y_1) \in \mathbb{F}_2^n$ which map to (c_1, d_1) under (f, g). In order to do so, we evaluate the ratio d/M, where:

- d is the number of input pairs $(x, y) \in \mathbb{F}_2^{d-1} \times \mathbb{F}_2^{d-1}$ such that the first components of the respective output pairs $H(x, y)$ equal (c_1, d_1).
- M is the number of input pairs $(x, y) \in \mathbb{F}_2^{d-1} \times \mathbb{F}_2^{d-1}$ where x and the first component of y are fixed.

In this way, we count the total number of preimages of H which map to (c_1, d_1) and normalize it by the number of preimages where the first d components of H are fixed, thus determining the number of preimages of (c_1, d_1) under (f, g).

As H is bijective, d corresponds to the number of pairs of binary vectors of length $d-1$ where the first components are fixed, which are $2^{d-2} \cdot 2^{d-2} = 2^{2(n-2)}$. On the other hand $M = 2^{d-2}$, since we only have $d-2$ free variables in the input configuration of the CA. Hence, it follows that $|(f, g)^{-1}(y_1, y_2)| = N/M = 2^{2(n-2)}/2^{d-2} = 2^{d-2}$. □

In the next Lemma, we show that pairwise balanced generating functions induce pairwise balanced bipermutive CA:

Lemma 21. *Let $\varphi, \gamma : \mathbb{F}_2^{d-2} \to \mathbb{F}_2$ be pairwise balanced generating functions of $d-2$ variables, with $d > 2$. Then, the corresponding bipermutive rules $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$ induced by φ and γ are pairwise balanced.*

Proof. Let $(y_1, y_2) \in \mathbb{F}_2^2$. One has that $|(\varphi, \gamma)^{-1}(y_1, y_2)| = 2^{d-4}$, since the generating functions φ and γ are pairwise balanced. Additionally, for all $\tilde{x} = (x_2, \cdots, x_{d-1}) \in (\varphi, \gamma)^{-1}(y_1, y_2)$, let $(x_1, \tilde{x}, x_d)$ denote the vector $(x_1, x_2, \cdots, x_{d-1}, x_d)$. Then, by Equation (12) it follows that $(0, \tilde{x}, 0) \in (f, g)^{-1}(y_1, y_2)$ and $(1, \tilde{x}, 1) \in (f, g)^{-1}(y_1, y_2)$. Similarly, for all vectors $\tilde{x} \in (\varphi, \gamma)^{-1}(\bar{y}_1, \bar{y}_2)$ where $\bar{y}_1 = 1 \oplus y_1$ and $\bar{y}_2 = 1 \oplus y_2$, it holds that $(1, \tilde{x}, 0) \in (f, g)^{-1}(y_1, y_2)$ and $(0, \tilde{x}, 1) \in (f, g)^{-1}(y_1, y_2)$. Since the fiber of (y_1, y_2) under (f, g) is given by

$$\begin{aligned}(f, g)^{-1}(y_1, y_2) = &\{(0, \tilde{x}, 0) : \tilde{x} \in (\varphi, \gamma)^{-1}(y_1, y_2)\} \cup \\ &\cup \{(1, \tilde{x}, 0) : \tilde{x} \in (\varphi, \gamma)^{-1}(\bar{y}_1, \bar{y}_2)\} \cup \\ &\cup \{(0, \tilde{x}, 1) : \tilde{x} \in (\varphi, \gamma)^{-1}(\bar{y}_1, \bar{y}_2)\} \cup \\ &\cup \{(1, \tilde{x}, 1) : \tilde{x} \in (\varphi, \gamma)^{-1}(y_1, y_2)\} \qquad (216)\end{aligned}$$

and since the four sets in Equation (216) are disjoint and have the same cardinality of $(\varphi, \gamma)^{-1}(y_1, y_2)$, we can finally conclude that

$$|(f, g)^{-1}(y_1, y_2)| = 4 \cdot |(\varphi, \gamma)^{-1}(y_1, y_2)| = 4 \cdot 2^{d-4} = 2^{d-2} \ . \qquad (217)$$

□

Remark that the converse of Lemma 21 does not hold. As a matter of fact, for $d = 4$ variables there already exist several instances of bipermutive functions pairs which produce orthogonal Latin squares (and hence are pairwise balanced) but whose generating functions are not pairwise balanced. An example is given by the two following linear rules:

$$f(x_1, x_2, x_3, x_4) = 1 \oplus x_1 \oplus x_3 \oplus x_4 \ ,$$
$$g(x_1, x_2, x_3, x_4) = x_1 \oplus x_4 \ .$$

The generating function of g in this case is the constant function defined as $\gamma(x) = 0$ for all $x \in \mathbb{F}_2^2$. Hence, the pairs $(0,1)$ and $(1,1)$ never occur when superimposing the truth tables of the two generating functions of f and g.

14.2 COMBINATORIAL ENUMERATION OF PAIRWISE BALANCED BIPERMUTIVE RULES

In this section, we enumerate all bipermutive rules pairs generating orthogonal Latin squares up to $d = 6$ variables and we classify them according to their nonlinearity.

14.2.1 *Counting Pairwise Balanced Bipermutive Rules*

The space of pairs of pairwise balanced generating functions is easily characterizable from the combinatorial point of view. In fact, for $d > 2$, each pairwise balanced pair $\varphi, \gamma : \mathbb{F}_2^{d-2} \to \mathbb{F}_2$ can be represented by a string s of length 2^{d-2} over the alphabet $A = \{1, 2, 3, 4\}$, where each symbol in s corresponds to the decimal encoding of one of the possible four pairs $(0,0)$, $(1,0)$, $(0,1)$ and $(1,1)$ occurring in the superposition of the truth tables. Since φ and γ are pairwise balanced, the string s must be balanced as well, meaning that the number of occurrences of each of the four symbols of A must be 2^{d-4}. Hence, the number of pairwise balanced pairs of generating functions of $d-2$ variables equals

$$\#\mathrm{Bal}\mathcal{G}_d = \binom{2^{d-2}}{2^{d-4}} \cdot \binom{3 \cdot 2^{d-4}}{2^{d-4}} \cdot \binom{2^{d-3}}{2^{d-4}} \ . \tag{218}$$

As a matter of fact, to construct a balanced quaternary string of length 2^{d-2} one has first to select the positions of the 2^{d-4} occurrences of the first symbol, which can be chosen in $\binom{2^{d-2}}{2^{d-4}}$ different ways. Next, the 2^{d-4} occurrences of the second symbol must be chosen among the $2^{d-2} - 2^{d-4} = 3 \cdot 2^{d-4}$ remaining positions, which can be done in $\binom{3 \cdot 2^{d-4}}{2^{d-4}}$ different ways. Finally, for the 2^{d-4} occurrences of the third

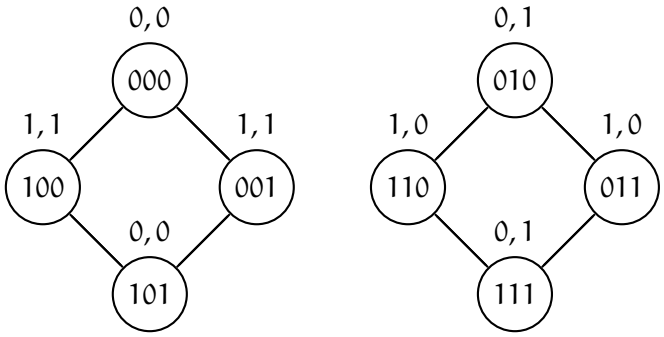


Figure 49: Graph representation of the pairwise balanced bipermutive rules 90 and 150.

symbol one has to choose among $2^{d-2} - 2 \cdot 2^{d-4} = 2^{d-3}$ remaining positions, corresponding to $\binom{2^{d-3}}{2^{d-4}}$ possible choices. At this point, the occurrences of the fourth symbols are fixed.

However, we saw at the end of Section 14.1 that pairwise balancedness is not a necessary condition on the generating functions to obtain pairwise balanced bipermutive rules. Consequently, by enumerating all balanced quaternary strings of length 2^{d-2} one only explores a subset of the space of pairwise balanced bipermutive rules of d variables, and thus in turn a subset of the space of bipermutive CA pairs generating orthogonal Latin squares of order 2^{d-1}.

We thus have to resort to a combinatorial characterization of pairwise balanced bipermutive functions. To this end, we adopt the *graph representation* of bipermutive rules, originally introduced in [105]. Given $d \in \mathbb{N}$, consider an undirected graph $G = (V, E)$ where $V = \mathbb{F}_2^n$. Two nodes $v_1, v_2 \in V$ are connected by an edge if and only if they differ either in their leftmost or rightmost coordinates, while they agree on the remaining ones. Thus, G is composed of 2^{d-2} connected components, and each connected component is composed of 4 nodes all having degree 2. A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ can be represented as a labeling function $l_f : V \to \{0, 1\}$ on the nodes of G. If f is bipermutive, then the labels of adjacent nodes must differ, while the labels of two nodes separated by a path of length 2 must be equal.

Given a pair of bipermutive functions $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$, we can still represent them on the graph as a labeling function $l_{f,g} : V \to \{0, 1\}^2$ on the nodes, where the labels are pairs specifying the outputs of the two functions. Assume that f and g are pairwise balanced: then, each pair $(y_1, y_2) \in \mathbb{F}_2^2$ occurs 2^{d-2} times as a label on G. As an example, Figure 49 depicts the graph representation of rule 90 and 150, which are pairwise balanced. Additionally, due to the property of different labels on adjacent nodes, it follows that exactly half of the connected components contain all $(0, 0)$ and $(1, 1)$ labels, while the remaining half contain all $(1, 0)$ and $(0, 1)$ labels. Since there are only two types of connected components with respect to the labels ($(0, 0)/(1, 1)$ and

$(1,0)/(0,1)$), it means that we can choose them in $\binom{2^{d-2}}{2^{d-3}}$ different ways. Moreover, let $C = \{v_1, v_2, v_3, v_4\}$ be a connected component where $(v_1, v_2), (v_1, v_3), (v_4, v_2), (v_4, v_3) \in E$, and assume that the labels on the nodes are either $(0,0)$ or $(1,1)$. Then, the labels can be arranged in two different ways, namely $(l_{f,g}(v_1), l_{f,g}(v_4)) = (0,0)$ and $(l_{f,g}(v_2, v_3)) = (1,1)$ or otherwise $(l_{f,g}(v_1), l_{f,g}(v_4)) = (1,1)$ and $(l_{f,g}(v_2), l_{f,g}(v_3)) = (0,0)$. In the same way, the labels on the nodes of a connected component of the type $(1,0)/(0,1)$ can be placed in two different ways. As a consequence, each of the $\binom{2^{d-2}}{2^{d-3}}$ ways for choosing the connected components with labels $(0,0)/(1,1)$ and $(1,0)/(0,1)$ gives rise to $2^{2^{d-3}} \cdot 2^{2^{d-3}} = 2^{2^{d-2}}$ pairwise balanced bipermutive functions. We have thus proved the following result:

Lemma 22. *The number of pairwise balanced pairs of bipermutive Boolean functions* $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$ *of* d *variables is:*

$$\#\mathrm{BalB}_d = \binom{2^{d-2}}{2^{d-3}} \cdot 2^{2^{d-2}} \quad . \tag{219}$$

14.2.2 *Exhaustive Search Experiments*

We now use the results discussed in the previous section to perform an exhaustive enumeration of all pairwise bipermutive rules pairs up to diameter 6 that orthogonal Latin squares. Table 19 reports the sizes of the search spaces for the sets of all pairs of bipermutive functions, the set of pairwise balanced generating functions and the set of pairwise balanced bipermutive functions of up to $d = 7$ variables.

One can notice that for $d \geqslant 7$ the resulting search space is too large to be exhaustively searched, even by focusing on the subsets of pairwise balanced generating functions. For this reason, we enumerated the set of pairwise balanced bipermutive functions BalB_d only up to $d = 6$ variables. To this end, we implemented an algorithm by Knuth [99] to generate all balanced binary strings of length 2^{d-2}, where the positions set to 0 and 1 respectively correspond to the $(0,0)/(1,1)$ and $(1,0)/(0,1)$ connected components. Then, for each

Table 19: Sizes of the search spaces for the different types of sets of bipermutive functions pairs of up to $d = 7$ variables.

d	$\#\mathcal{B}_d$	$\#\mathrm{BalG}_d$	$\#\mathrm{BalB}_d$
3	16	0	8
4	256	24	96
5	65536	2520	17920
6	4294967296	63006300	843448320
7	$\approx 1.84 \cdot 10^{19}$	$\approx 9.96 \cdot 10^{15}$	$\approx 2.58 \cdot 10^{18}$

Table 20: Distribution of CA-based orthogonal Latin squares up to $d = 6$.

d	LS_size	#total	#lin	#nonlin	nl_dist
3	4×4	1	1	0	–
4	8×8	9	5	4	$(4,4,4)$
5	16×16	213	21	192	$(4,4,96)$, $(8,8,96)$
6	32×32	66685	85	66600	$(4,4,512)$, $(8,8,4020)$, $(12,12,17992)$, $(16,16,28388)$, $(20,20,14384)$, $(4,12,8)$, $(8,16,160)$, $(12,20,128)$, $(16,24,88)$

balanced combination of connected components we generated all possible $2^{2^{d-2}}$ arrangements of the labels, constructed the resulting pairs of bipermutive functions, and computed their respective nonlinearity values. Finally, we generated the associated Latin squares of order $d = 2^{d-1}$, and checked for their orthogonality.

We remark that the enumeration of $\mathrm{Bal}\mathcal{B}_6$ is a computationally intensive task, since it took approximately 22 hours to complete under our Java implementation on a 64-bit Linux machine with 40 Intel Xeon cores running at 2.4 GHz.

Table 20 reports the distribution of linear and nonlinear pairs of orthogonal Latin squares. For each value of d, the corresponding size of the Latin squares is reported, along with the number of linear and nonlinear pairs of bipermutive functions generating orthogonal Latin squares. Additionally, in the last column we report the distribution of nonlinearity values in triplets $(\mathrm{nl}(f), \mathrm{nl}(g), \#\mathrm{num})$ where $\mathrm{nl}(f)$ and $\mathrm{nl}(g)$ respectively denote the nonlinearity values of f and g, while $\#\mathrm{num}$ is the number of pairs generating orthogonal Latin squares that achieve those values. Notice that all numbers are divided by 8, since we have to consider the pairs with swapped order, which halve the resulting sets, and the reflection and complement transformations, which by Lemma 19 additionally reduce them to a quarter.

As a qualitative remark on the distributions reported in Table 20, one may observe that linear pairs become more sparse as the number of variables d increases, while the majority of the pairs are nonlinear. Moreover, one can see that for $d = 6$ there are pairs with functions of different nonlinearities. This finding falsified our initial belief that

two bipermutive functions inducing orthogonal Latin squares must have the same value of nonlinearity, an empirical observation which held up to $d = 5$ variables.

14.3 HEURISTIC OPTIMIZATION APPROACH

It has been remarked in Section 14.1 that finding nonlinear bipermutive CA rules generating OLS can be reduced to the search of the corresponding pairs of nonlinear generating functions φ, γ. In fact, we already know by Lemma 17 that the bipermutive functions f, g corresponding to φ, γ give rise to a pair of Latin squares when used as local rules of two CA. Therefore, by representing the genotype of the candidate solutions as pairs of generating functions, we can focus the optimization efforts of GA and GP only on the nonlinearity and orthogonality properties, without having to consider the row-column permutation constraints of the squares generated by the CA.

In this section, we first describe the two fitness functions that we used to evaluate the phenotype corresponding to a pair of generating functions (φ, γ), i.e. the Latin squares generated by the CA F, G with bipermutive local rules f, g defined by (φ, γ). Then, we proceed by describing the GA and GP encodings that we adopted in our experiments. Finally, we discuss the experimental settings and the results obtained by GA and GP on this optimization problem.

14.3.1 *Fitness Functions*

Since we are interested in obtaining pairs of OLS, the fitness functions used by GA and GP must in the first place measure the deviation of two Latin squares generated by a pair of generating functions from being orthogonal. The most natural approach is to count the number of *repeated ordered pairs* in the superposition of two Latin squares. The optimization task is thus to minimize such quantity, since having zero repeated pairs in the superposition means that each pair of symbols occurs exactly once (i.e. the two Latin squares are orthogonal). Since the exhaustive search results presented in Section 14.2.2 showed that most of the OLS are generated by pairs of nonlinear CA for $d > 4$, we decided not to check the nonlinearity property in our first fitness function, which is formally defined below.

Let $\varphi, \gamma : \mathbb{F}_2^{d-2} \to \mathbb{F}_2$ be a pair of generating functions of $d-2$ variables, and let $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$ be the corresponding bipermutive functions of d variables. Denote by L_F and L_G the Latin squares of order $[N] = 2^{d-1}$ induced by the CA $F, G : \mathbb{F}_2^{2(n-1)} \to \mathbb{F}_2^{d-1}$ with local rules f and g. Let $\mathrm{Sup}_{L_F, L_G} : [N] \times [N] \to [N] \times [N]$ be the *superposition* function of L_F and L_G defined as

$$\mathrm{Sup}_{L_F, L_G}(i, j) = (L_F(i, j), L_G(i, j)) \tag{220}$$

for all $(i,j) \in [N] \times [N]$. Then, the value of the fitness function fit_1 evaluated on the individual (φ,γ) is given by:

$$\mathrm{fit}_1(\varphi,\gamma) = |\mathrm{rep}(L_F, L_G)| \ , \tag{221}$$

where $\mathrm{rep}(L_F, L_G)$ is the set defined as:

$$\mathrm{rep}(L_F, L_G) = \left\{(x,y) \in [N] \times [N] : \left|\mathrm{Sup}^{-1}_{L_F,L_G}(x,y)\right| > 1\right\} \ . \tag{222}$$

Remark that the range of fit_1 is $\{0,\cdots,2^{2(n-1)}\}$, since the two Latin squares L_F and L_G have order 2^{d-1}. In particular, an optimal solution has fitness value 0, hence the objective is to minimize fit_1.

In addition to fit_1, we tested a second fitness function that also takes into account the nonlinearity of the generating functions, to ensure that an optimal solution corresponds to a pair of OLS which is generated by nonlinear bipermutive CA. In particular, given two generating functions $\varphi,\gamma : \mathbb{F}_2^{d-2} \to \mathbb{F}_2$ the value of the second fitness function fit_2 computed over (φ,γ) is:

$$\mathrm{fit}_2(\varphi,\gamma) = \mathrm{fit}_1(\varphi,\gamma) + \mathrm{NlPen}(\varphi,\gamma) \cdot N^2 \tag{223}$$

where $d = 2^{d-1}$ and $\mathrm{NlPen}(\cdot,\cdot)$ is a penalty factor defined as follows:

$$\mathrm{NlPen}(\varphi,\gamma) = \begin{cases} 0 \ , & \text{if } \mathrm{Nl}(\varphi) > 0 \ \text{ AND } \ \mathrm{Nl}(\gamma) > 0 \\ 1 \ , & \text{if } \mathrm{Nl}(\varphi) = 0 \ \text{ XOR } \ \mathrm{Nl}(\gamma) = 0 \\ 2 \ , & \text{if } \mathrm{Nl}(\varphi) = 0 \ \text{ AND } \ \mathrm{Nl}(\gamma) = 0 \end{cases} \tag{224}$$

In other words, the penalty factor is 0 if both generating functions are nonlinear, 1 if only one of them is linear, and 2 if both functions are linear. Considering what we said about the range of fit_1, from (223) and (224) it follows that the range of fit_2 is $\{0,\cdots,3\cdot 2^{2(n-1)}\}$. As for fit_1, the optimization objective in this case is to minimize fit_2.

14.3.2 *Single Bitstring Encoding*

The first representation for the genotype of GA solutions encodes a pair of generating functions as a single bitstring. Given two generating functions $\varphi,\gamma : \mathbb{F}_2^{d-2} \to \mathbb{F}_2$ respectively with truth tables $\Omega(f), \Omega(g) \in \mathbb{F}_2^{2^{d-2}}$, the chromosome which represents the pair (φ,γ) is defined as $c(\varphi,\gamma) = \Omega(f)\|\Omega(g)$, where $\|$ denotes the concatenation of two strings. Hence, the chromosome is a bitstring of length 2^{d-1} whose first half corresponds to the truth table of φ, while the second half is the truth table of γ.

Under this encoding, we apply the standard variation operators used in GA, namely one-point crossover and bit-flip mutation. Remark that one-point crossover always produces offspring whose left or right half is inherited from one of the two parents, except when

the crossover point happens to be exactly in the middle of the two parents chromosomes (in which case the first child inherits the left half from the first parent and the second half from the second parent, and vice versa for the second child). From the point of view of the phenotype, this means that both children inherit one of the four Latin squares from their parents, or two if the crossover point is in the middle of the parents.

14.3.3 *Double Bitstring and Double Tree Encodings*

In the second encoding that we used in our experiments, we considered each individual as composed of two independent parts that represent a pair of generating functions. In particular, in the case of GA each chromosome consists of the two bitstrings representing the truth tables of length 2^{d-2} of the generating functions. Formally, given $\varphi, \gamma : \mathbb{F}_2^{d-2} \to \mathbb{F}_2$, the associated GA chromosome is defined as $c(\varphi, \gamma) = (\Omega(\varphi), \Omega(\gamma)) \in \mathbb{F}_2^{2^{d-2}} \times \mathbb{F}_2^{2^{d-2}}$. Then, one-point crossover and bit-flip mutation are applied independently on the two components of the chromosomes. Notice that in the case of crossover, contrary to the single bitstring encoding, the Latin squares of the offspring always differ from those of the parents, since the two generating functions are recombined independently.

We used a similar double encoding for the candidate solutions evolved by GP. In particular, each chromosome in this case is represented by two Boolean trees which encode the algebraic expressions of the generating functions. Analogously to the GA case, under this encoding we apply the standard tree crossover and mutation operators of GP independently on the two Boolean trees of the generating functions.

14.3.4 *Balanced Quaternary String Encoding*

We exploited the pairwise balancedness property mentioned in Section 14.1.2 to devise a third encoding for the candidate solutions of GA. In particular, given two generating functions $\varphi, \gamma : \mathbb{F}_2^{d-2} \to \mathbb{F}_2$ of $d-2$ variables, in this encoding the chromosome represents the superposition of the truth tables of φ and γ as a *quaternary string* of length 2^{d-2} over the set $Q = \{1, 2, 3, 4\}$, by associating the four pairs of $\mathbb{F}_2^2$ to the elements of Q as follows:

$$(0,0) \to 1; \;\; (1,0) \to 2; \;\; (0,1) \to 3; \;\; (1,1) \to 4 \;\; .$$

Under this encoding, the pairwise balancedness constraint is equivalent to require that each number from 1 to 4 occurs 2^{d-4} times in the string. Consider again the example described in Section 14.1.2 of the two generating functions $\varphi, \gamma : \mathbb{F}_2^3 \to \mathbb{F}_2$ respectively defined by the truth tables $\Omega(\varphi) = 01011010$ and $\Omega(\gamma) = 01101001$. Table 21

Table 21: Example of balanced quaternary string encoding.

x	000	100	010	110	001	101	011	111
$\varphi(x)$	0	1	0	1	1	0	1	0
$\gamma(x)$	0	1	1	0	1	0	0	1
$c(\varphi,\gamma)$	1	4	3	2	4	1	2	3

reports the superimposed truth tables of φ and γ along with the corresponding quaternary chromosome $c(\varphi,\gamma)$. From the example, one can observe that each number from 1 to 4 in the chromosome column appears $2^{3-1} = 2$ times.

Clearly, applying classic one-point crossover and mutation operators to balanced quaternary chromosomes does not guarantee that the produced offspring will be balanced as well. Therefore, we designed ad-hoc operators in order to preserve the pairwise balancedness property so that GA searches only the constrained space instead of the whole set of pairs of generating functions.

Our crossover operator is loosely inspired from the operator described in [123] for balanced Boolean functions and the operator proposed in Chapter 9 for the Walsh spectra of plateaued functions. More precisely, our operator employs four counters to keep track of the multiplicities of the four values in the child chromosome. Given two quaternary chromosomes p_1, p_2 of length d, a child chromosome c is generated as follows:

1. Set the four counters cnt_1, cnt_2, cnt_3 and cnt_4 to 0.
2. If the number of positions where p_1 and p_2 have equal values is greater than $d/2$, then randomly choose p_1 or p_2 and apply the following permutation to each of its loci:
$$1 \to 3;\ \ 2 \to 4;\ \ 3 \to 1;\ \ 4 \to 2\ .$$
3. Determine the positions where p_1 and p_2 have equal values and copy them in the child c.
4. Pick a random position $i \in \{1, \cdots, N\}$ among those which have not already been selected and such that $p_1[i]$ and $p_2[i]$ have different values. Then, the value of $c[i]$ is determined by one of the following cases, depending on the values of the counters $cnt_{p_1[i]}$ and $cnt_{p_2[i]}$:
   a) If the values of $cnt_{p_1[i]}$ and $cnt_{p_2[i]}$ are both below the threshold 2^{d-4}, randomly copy $p_1[i]$ or $p_2[i]$ in $c[i]$, and increase the corresponding counter.
   b) If only one of the two counters has reached 2^{d-4}, then copy the value corresponding to the other counter in $c[i]$, and increase such counter.

c) If both counters reached 2^{d-4}, copy one of the two remaining values v_1, v_2 in $c[i]$ by applying again cases (a) and (b) to cnt_{v_1} and cnt_{v_2}.

5. Return to step 4 until all positions in the child have been filled.

It can easily be seen from the above procedure that if both parents are balanced quaternary strings, then the generated offspring will be balanced as well. Since this crossover operator produces only one child, we apply it twice for each pair of parents. Notice also that step (2) is performed in order to avoid producing offspring which is too similar to the parents (a similar strategy was also adopted in [123]).

On the other hand, for mutation we adopted a simple operator where each value in a locus to be mutated is swapped with the value in another locus, chosen in a random way. Thus, the balancedness property is preserved since swaps do not change the number of occurrences of the symbols in a string.

14.4 EXPERIMENTATION

We tested our GA and GP on the sets of bipermutive functions pairs of $d = 7$ and $d = 8$ variables, which are the smallest instances of this optimization problem which are not amenable to exhaustive search. In fact, the corresponding search spaces of generating functions pairs of $d - 2 = 5$ and $d - 2 = 6$ variables have sizes 2^{64} and 2^{128}, respectively. From the point of view of the phenotype, $d = 7$ corresponds to Latin squares of size 64×64, while $d = 8$ to Latin squares of size 128×128.

In the remainder of this section, we describe the experimental settings adopted for GA and GP, and we discuss the obtained results.

14.4.1 *Experimental Settings*

As mentioned in Section 14.3.3, the GP encoding uses elementary Boolean functions to build a tree representing each of the two generating functions, whereas the corresponding Boolean variables are used as terminals. The function set in our experiments comprise functions AND, OR, XOR, XNOR, which all take two arguments, and function NOT which takes a single argument. Additionally, we included the function IF, which takes three arguments and returns the second one if the first one evaluates to true, and the third one otherwise. Finally, we set the maximum tree depth to 5. A lower bound on the size of the tree space with such parameters can be estimated using the method described in [58]. In particular, considering only the four binary operators, a tree can be composed of at most 15 internal nodes and 16 leaves. Since each internal node can take 4 different values while the terminal on the leaves are Boolean values, Table 1 of [58] reports a

total of $1.82 \cdot 10^{14}$ possible trees, a quantity which is not amenable to exhaustive search. Moreover, as mentioned above, this is a lower bound since we are ignoring the ternary IF operator.

Regarding the population size, in the case of GP we set it to to 500. On the other hand, for GA we set the population size to 30 individuals. In fact, the preliminary experiments that we performed for parameter tuning showed that bigger populations do not produce better results with GA.

For the selection process, we employ a steady-state selection with a 3-tournament operator for both GA and GP, that in each iteration randomly selects three individuals for the tournament and eliminates the worst one. A new individual is created immediately by crossing over the remaining two from the tournament, which then in GP undergoes mutation with probability of 0.5. The variation operators used for GP are simple tree crossover, uniform crossover, size fair, one-point, and context preserving crossover [148] (selected at random) and subtree mutation. For GA, the variation operators are one-point crossover and bit-flip mutation in the case of single and double bitstring encodings, while we adopt the balanced crossover and swap mutation operators described in Section 14.3.4 for the quaternary string encoding. In the GA experiments, after an initial phase of parameter tuning we observed that setting the crossover and mutation probabilities respectively to 0.95 and 0.2 yielded the best results.

Common parameters for all the experiments include the termination condition of 300000 fitness evaluations. We chose this particular bound because our preliminary tests showed that optimal solutions are mostly found before reaching this amount of evaluations, both for GA and GP. Finally, each experiment is repeated 50 times.

14.4.2 *Results*

For GA, we performed a total of 6 experiments, given by the combinations of 3 encodings and 2 problem instances. In particular, with GA we used only the first fitness function $\mathtt{fit}_1$ which counts the number of repeated pairs, since we observed that adding the nonlinearity constraint did not modify the performances in a significant way. On the other hand, with GP we performed a total of 4 experiments, given by the combinations of 2 fitness functions and 2 problem instances. In what follows, we compactly denote a GA experiment as $(\mathtt{GA}, n, \mathtt{enc}_i)$, where d is the number of variables of the bipermutive functions (which thus can be either 7 or 8), while $\mathtt{enc}_i$ represents the encoding adopted. In particular, $\mathtt{enc}_1$ stands for single bitstring, $\mathtt{enc}_2$ for double bitstring and $\mathtt{enc}_3$ for balanced quaternary strings. Likewise, we denote a GP experiment as $(\mathtt{GP}, n, \mathtt{fit}_i)$, where d still stands for the number of variables, while $\mathtt{fit}_i$ denotes the fitness function.

Table 22: Best solutions found by GA and GP.

Exp.	avg	std	#opt	#lin	#nlin
$(GA, 7, enc_1)$	520.32	360.16	12/50	0	12
$(GA, 7, enc_2)$	565.44	389.03	15/50	0	15
$(GA, 7, enc_3)$	392.64	328.47	18/50	0	18
$(GA, 8, enc_1)$	4165.44	604	1/50	0	1
$(GA, 8, enc_2)$	4222.16	125.03	0/50	0	0
$(GA, 8, enc_3)$	4696.48	135.51	0/50	0	0
$(GP, 7, fit_1)$	0	0	50/50	50	0
$(GP, 7, fit_2)$	0	0	50/50	0	50
$(GP, 8, fit_1)$	0	0	50/50	47	3
$(GP, 8, fit_2)$	0	0	50/50	0	50

Table 22 reports the results obtained in each experiment. In particular, for each combination we show the average fitness and the standard deviation computed on the best solutions generated over all 50 runs, along with the number of optimal solutions found and their distribution as linear/nonlinear pairs. In general, one can observe that GP has a clear advantage over GA, since it always converges to an optimal solution in each experimental run for both $d = 7$ and $d = 8$ variables. On the other hand, GA only manages to generate OLS for $d = 7$ variables (except for a single optimal solution of $d = 8$ variables found under the single bitstring encoding). Moreover, even in the case of $d = 7$ variables it can be seen that the success rate of GA in generating OLS is remarkably lower than that achieved by GP. One can additionally remark that the balanced quaternary encoding gives a slight advantage to GA over single and double bitstrings, with respect to the number of optimal solutions found.

Interestingly, one can notice that the optimal solutions produced by GP under fitness function fit_1 are mostly given by linear pairs. More precisely, under fit_1 GP managed to find only 3 nonlinear pairs of $d = 8$ variables which generated OLS. The addition of the nonlinearity penalty factor with fit_2 however solved the issue, since in this case all optimal solutions found are given by pairs of nonlinear generating functions. On the other hand, it can be observed that all optimal solutions found by GA with fitness functions fit_1 are nonlinear. This difference could be explained by the fact that the set of operators used for GP trees include also the XOR, which is a linear operator. Hence, when the optimization criterion is just the minimization of the repeated pairs, it could be easier for GP to find pairs of generating functions whose trees are composed only of XOR, which correspond to linear solutions. Since the number of linear functions

is much smaller than the total number of Boolean functions, it could be that GP finds very quickly an orthogonal solution by sticking to linear pairs. On the other hand, there is no clear relationship between the truth-table based encodings used by GA and the nonlinearity of the generating functions, which could explain why GA always find nonlinear optimal solutions, even if with much more difficulty.

Considering the nonlinear optimal solutions found by GA and GP, we can additionally remark an interesting fact. First, all the optimal bipermutive rules found using the single and double bitstring representations with GA and the double tree encoding with GP satisfy the pairwise balancedness property introduced in Section 14.1.2. Since these encodings do not enforce pairwise balancedness as a constraint (like with quaternary strings on the generating functions), this finding seems to support the conjecture that all bipermutive rules pairs inducing OLS must be pairwise balanced, a fact that we experimentally assessed by exhaustive search up to $d = 6$ variables.

14.5 CONCLUSIONS

In this chapter, we investigated orthogonal Latin squares generated by nonlinear bipermutive CA. We first proved that all pairs of bipermutive rules inducing orthogonal Latin squares must be pairwise balanced, meaning that the superposition of their truth tables must yield an equal number of occurrences of the four pairs $(0,0)$, $(1,0)$, $(0,1)$ and $(1,1)$. We then used a combinatorial algorithm to enumerate all pairwise balanced Boolean functions of up to $d = 6$ variables, finding those which generate orthogonal Latin squares and classifying them with respect to their nonlinearity values. The results of our computer search showed that, as the number of variables of the local rules increases, most of the orthogonal pairs are nonlinear.

Next, we addressed the problem of designing nonlinear CA-based orthogonal Latin squares using GA and GP. Specifically, we formulated the optimization problem as the search of pairs of nonlinear generating functions inducing OLS. We experimented three different encodings for the candidate solutions of GA, namely single bitstring, double bitstring and balanced quaternary string, introducing ad-hoc crossover and mutation operators for the last one. On the other hand, with GP we adopted a double tree representation. We tested the two metaheuristics on the problem instances of $d = 7, 8$ variables, remarking that GP is always able to converge to an optimal solution in both problem instances, while GA manages to generate OLS with a lower success rate only for $d = 7$. On the other hand, we also observed that GP mostly finds linear solutions when the fitness function counts only the number of repeated pairs, while the solutions found by GA are always nonlinear.

Part V

FINAL REMARKS

CONCLUSIONS AND OPEN PROBLEMS

15

In this final chapter, we summarize the research lines investigated throughout the thesis and discuss several directions for future developments on the subject of Boolean functions and combinatorial designs generated by cellular automata.

In particular, for each contribution we first summarize the results presented in the corresponding chapter, and then we describe a series of open problems and possible ideas for further research.

15.1 HEURISTIC OPTIMIZATION OF BOOLEAN FUNCTIONS

15.1.1 *Discrete Particle Swarm Optimization*

In Chapter 8 we designed a discrete version of Particle Swarm Optimization which explores the set of balanced Boolean functions, using the truth table representation. The main feature of this algorithm is the update operator for the particle positions, which swaps different values in the truth table of a candidate solution in order to maintain its Hamming weight. We tested our PSO on 6 problem instances, namely the sets of balanced Boolean functions from $n = 7$ to $n = 12$ variables, by adopting three fitness functions. Each of the three fitness functions, in particular, optimized a different combination of nonlinearity, correlation immunity and propagation criterion.

The experimental result showed that under the first fitness function our PSO algorithm is able to produce solutions with similar or better combinations of cryptographic properties than other heuristic methods already published in the literature. On the other hand, for the other two fitness functions we observed that the performances of PSO do not scale beyond $n = 7$ variables, which suggests that further parameter tuning is needed for the remaining problem instances. Nonetheless, in the instance with $n = 7$ variables our algorithm was able to find a plateaued Boolean function with profile $(7, 2, 4, 56)$, which is the best possible trade-off among nonlinearity, algebraic degree and resiliency order which is allowed respectively by Siegenthaler's and Tarannikov's bounds.

There are several venues for future developments on the subject. One possibility is to test our PSO with other fitness functions, such as the one adopted in [43] for Simulated Annealing, which measures how flat the Walsh spectrum of a Boolean function is. Another interesting direction of research would be to modify the UPDATE-BAL-POS() procedure in such a way that only the swaps which increase nonlin-

earity or decrease the deviation from k-th correlation immunity are performed. This could be accomplished, for instance, by integrating the Hill Climbing step inside the update procedure, instead of performing it after the particles positions have been modified.

15.1.2 *Genetic Algorithms and Spectral Inversion*

In Chapter 9 we investigated the performances of a Genetic Algorithm at optimizing the cryptographic properties of Boolean functions through the spectral inversion approach set forth in [42]. The main idea is to represent a candidate solution as a permutation of a Walsh spectrum already satisfying the desired cryptographic properties, such as high nonlinearity and resiliency order. Then, the optimization task becomes minimizing the deviation of the resulting pseudoboolean function that results by applying the inverse Walsh transform to such a spectrum. We designed a specific crossover and a mutation operator that preserve the constraint on the Walsh spectrum, and adopted the same fitness function defined in [42] which measures the deviation from the nearest Boolean function.

We tested our GA over the spaces of pseudoboolean functions of $n = 6$ and $n = 7$ variables, with the optimization goal of finding plateaued Boolean functions whose profile is defined by the minimal index that does not result in bent functions (the latter being unbalanced, and thus not directly usable for cryptographic purposes). We then compared the obtained results with those achieved by the Simulated Annealing algorithm proposed in [42]. The results show that for $n = 6$ our GA outperforms SA with respect to the number of optimal solutions found per number of optimization runs. On the other hand, for the instance with $n = 7$ variables neither technique was able to produce an optimal solution, even if SA scored an average better value for the fitness function.

The obtained results suggest that our GA does not scale well for $n \geqslant 7$, the likely reason being that it gets stuck in local optima. A possible way to overcome this drawback is to combine the global search capabilities of GA with a local search technique. A straightforward method to investigate this idea could be the integration of our GA inside the SA algorithm of [42], using for example the *Genetic Annealing* framework [197]. The obvious downside to this solution, however, would be the significantly higher amount of computational resources required to carry out a single optimization run.

An alternative solution could be to add a Hill Climbing optimization step in our GA, similarly to the strategy proposed in [123] which we adopted for our PSO algorithm in Chapter 8. In the context of our GA, a Hill Climbing optimization step would require characterizing the pairs of Walsh coefficients which, if swapped, would decrease the deviation of the resulting pseudoboolean function.

An additional direction for further research would be to consider different cryptographic properties other than nonlinearity, algebraic degree and resiliency. For example, the heuristic search of Boolean functions satisfying only propagation criterion $\mathrm{PC}(l)$ could be done using the same basic spectral inversion method of [42]: in this case, it would suffice to evolve through our GA the autocorrelation spectrum instead of the Walsh spectrum of the candidate solutions. However, remark that considering properties related to both the autocorrelation and Walsh spectrum at the same time would require modifying the representation of the candidate solutions. This is due to the fact that a valid swap on the Walsh spectrum could induce an invalid swap on the autocorrelation function (and vice versa), because of the Wiener-Khintchine theorem.

15.2 CRYPTOGRAPHIC AND CODING-THEORETIC ANALYSIS OF CA

15.2.1 *Preimages Period in Surjective CA*

In Chapter 10 we addressed the problem of computing the period of preimages of spatially periodic configurations (SPC) under the action of surjective CA. As a matter of fact, it is well known (see [57]) that in surjective CA every preimage of a SPC is spatially periodic as well. However, little or no work has been carried out in the literature to actually characterize the periods of such preimages. Beside its theoretical interest, this problem also turns out to be equivalent to determining the maximum number of players allowed in the CA-based secret sharing scheme proposed in [112].

In the first part of the chapter we considered the periods of preimages in generic surjective CA. In particular, we develop an algorithm based on the *de Bruijn graph representation* which computes the periods of all preimages of a SPC surjective CA, along with their multiplicities. Next, we narrowed our focus on the specific class of linear bipermutive CA over finite fields. In this case, the problem can be reduced to the study of concatenated linear recurring sequences (LRS), for which we provided a complete characterizations of their periods and multiplicities. Finally, we further extended this analysis by considering also linear CA defined over a finite ring as the state alphabet.

There are several directions along which this contribution can be extended and improved. As a matter of fact, Chapter 10 addressed two extreme cases of the preimages periods problem: the most generic one dealing with surjective CA, for which some facts and bounds can be derived, and the case of linear and bipermutive CA over finite fields, about which every major question can be settled by leveraging on the theory of LRS.

Still, one can consider several intermediate classes between surjective CA and LBCA, one of the most interesting being bipermutive

CA equipped with *nonlinear* local rules. Notice that the *affine* case can be still solved using the tools of concatenated LRS. Specifically, let $F : \mathbb{F}_q^{\mathbb{Z}} \to \mathbb{F}_q^{\mathbb{Z}}$ be a bipermutive CA with affine local rule $f : \mathbb{F}_q^d \to \mathbb{F}_q$ of radius d, i.e. f is a linear combination of the neighborhood cells plus a constant $a \in \mathbb{F}_q$, meaning that the k–th order LRS associated to the inverse permutation $f_{R,z}^{-1}$ is *inhomogeneous*. In particular, a k–th order inhomogeneous LRS can be expressed as a $(k+1)$-th order homogeneous LRS, which allows one to apply all the results proved in Chapter 10 about concatenated LRS to the affine case as well. From the CA point of view, this means that an affine local rule of diameter d can be seen as a linear rule defined on a larger neighborhood, namely $\{i, \cdots, i+d\}$.

Clearly, the above procedure cannot be applied to nonlinear rules, where the preimages are generated by a *Nonlinear Feedback Shift Register* (NFSR) disturbed by the LFSR which generates the spatially periodic configuration. We note that this case is interesting also for another cryptographic application other than CA-based secret sharing schemes, since the concatenation of a NFSR and a LFSR is the main primitive upon which the stream cipher Grain is based [77]. Hence, developing a method to study the periods of preimages in nonlinear bipermutive CA could also be useful for cryptanalyzing this cipher.

Successively, one could also consider classes of surjective CA more general than bipermutive CA. The *openness* property could be an interesting starting point to investigate, since configurations of open CA have a constant number of preimages, which can be viewed as a weaker condition than bipermutivity [102]. Hence, the openness property could induce some regularities on the structure of the u-closure graph that could simplify the analysis.

Concerning generic surjective CA, we also remark that the upper bound about the time complexity for the construction of the u-closure graph via DFS given in Chapter 10 is not tight. As a matter of fact, the worst case mentioned in Section 10.1 cannot occur in surjective CA due to their balancing property, which implies that the DFS tree associated to a vertex can be balanced only up to a certain depth. Taking into account this fact, one could derive a better upper bound on the time complexity of the graph construction procedure.

15.2.2 *CA-Based S-boxes*

In Chapter 11 we undertook an investigation of the cryptographic properties of S-boxes defined by finite CA, both in the no boundary and periodic setting. We started our theoretical analysis by considering the algebraic degree of such S-boxes, showing that it corresponds to the algebraic degree of the CA local rule. This is due to the fact that the degree of a S-box is defined as the maximal degree among all its coordinate functions, and each coordinate function corresponds to the

CA local rule applied to the relevant neighborhood. Next, we proved two upper bounds respectively on the nonlinearity and differential uniformity achievable by CA-based S-boxes, always relating them to the corresponding properties of the underlying local rules. In particular, for the nonlinearity of periodic boundary CA we observed that our upper bound coincides with the Sidelnikov-Chabaud-Vaudenay bound only when the local rule diameter equals the CA length, which corresponds to the case of rotation symmetric S-boxes.

In the experimental part of this chapter, we applied a Genetic Programming algorithm to evolve rotation symmetric S-boxes featuring an optimal combination of nonlinearity and differential uniformity. More specifically, we considered the problem instances of 4×4 up to 8×8 S-boxes sizes, since under our GP tree encoding even the 4×4 case is not amenable to an exhaustive search. The experimental results show that our GP is always able to produce optimal solutions up to size 7×7, with the exception of the 6×6 case where our evolved S-boxes remain slightly suboptimal with respect to differential uniformity. In the 8×8 problem instance, on the other hand, our GP algorithm does not come even close to the theoretical optimal values, and its performances are consistently worse than those achieved by algebraic constructions or other heuristic methods.

Finally, we used the truth table representation to perform an exhaustive search of all bijective S-boxes defined by CA rules up to size 5×5, observing that only a small fractions are optimal with respect to their cryptographic properties. Next, we carried out a classification of all CA-based S-boxes of sizes 3×3 and 4×4 up to affine equivalence. In particular, we observed that there exist 4 equivalence classes of such S-boxes for sizes 3×3, among which one of them is optimal with respect to nonlinearity and differential uniformity. On the other hand, for the 4×4 case we found 19 equivalence classes, four of which are optimal as per the classification presented in [104].

There are several options to consider for further research on the subject. From the theoretical side, a possible direction is to investigate lower bounds on the nonlinearity and differential uniformity of CA S-boxes based on specific subclasses of local rules, such as plateaued Boolean functions. We note that this question has already been investigated in Mariot et al. [113] for permutive local rules. In particular, computer searches performed on small input sizes suggest that permutive rules always satisfy with equality the bound on nonlinearity given in Theorem 25, an example of which is the rule χ used in the Keccak S-box. However, the authors of [113] later observed a mistake in the proof of this fact, and they are currently investigating either how to fix it or to disprove it.

Another interesting option would be to study more in detail the inverses of bijective S-boxes defined by CA. As we remarked in Chapter 2, if a periodic boundary CA is invertible for certain array sizes,

then the corresponding infinite CA is not reversible in general. This is the case, for instance, with the χ local rule used in KECCAK, which results in a permutation for every odd size of the cellular array, but not for even sizes. The consequence is that the inverse of such a CA cannot be described by a local rule as well, even though the global rule is shift-invariant. Hence, an interesting problem would be to characterize subclasses of CA rules which result in permutations for every size of the cellular array. We note that the reversibility problem has been thoroughly investigated in the CA literature, due to its many connections with reversibility of physical systems (see for instance [180, 49]). Consequently, a possible idea could be to consider some classes of reversible CA already known in the literature (such as *marker* CA [45], which are a generalization of complementing-landscape CA to which rule χ belongs to), and analyze them with respect to their cryptographic properties.

Regarding our experimental results with GP, the most obvious direction for further research is to focus on the 8×8 size and try to improve the values of cryptographic properties. Of course, in Chapter 11 we concentrated only on a small set of cryptographic properties, so one could include other relevant properties like algebraic degree in the fitness function. An interesting open problem in this context is the design of APN functions of maximal algebraic degree. As noted in [25], the upper bound on the algebraic degree of (n, n) APN functions is n. However, up to now there are no known examples of such S-boxes. Notice that, by Lemma 11, the bound on the algebraic degree of (n, n) bijective S-boxes defined by CA is still n, since it corresponds to the case of rotation-symmetric S-boxes. Hence, a possible direction for future research is to use GP to see if it is possible to evolve n-bit APN rotation symmetric S-boxes with algebraic degree n.

As it can be seen in Table 31, for the 4×4 and 6×6 sizes the best obtainable nonlinearity still equals the quadratic bound respectively when $d = 3$ and $d = 5$. Hence, it would be interesting to investigate whether in these cases our heuristic approach based on GP is still able to evolve S-boxes with optimal nonlinearity with $d = n - 1$, i. e. where the local rule depends on all input variables except one.

For our classification results presented in the last part of the chapter, a first direction for further improvement is to classify 3×3 and 4×4 S-boxes defined by CA under more general equivalence relations, like Extended Affine (EA) equivalence and Carlet-Charpin-Zinoviev (CCZ) equivalence. As we remarked in Chapter 5, in fact, S-boxes which are affine equivalent may not be EA or CCZ equivalent. Hence, the goal here is to determine whether S-boxes defined by CA correspond to other vectorial functions already known in the literature under these equivalence relations, or if they are new. Finally, a further direction for future research would be to extend the affine equivalence classification of CA-based S-boxes for size 5×5.

Here, the number of equivalence classes is too huge to be exhaustively searched. A possible idea to overcome this obstacle would be to limit the classification to specific classes of S-boxes, as done for example in [22] for quadratic permutations.

15.2.3 *Resiliency and Asynchrony Immunity*

In Chapter 12, we continued our investigation of the cryptographic properties of CA by considering the resiliency criterion, which is relevant in the design of Pseudorandom Number Generators (PRNG) for stream ciphers. Specifically, we proved that the global rule of every bipermutive CA is always at least 1–resilient, thus lifting the result proved in [105] for bipermutive local rules. Then, we uncovered an interesting equivalence between linear CA and linear cyclic codes, remarking that the transition matrix of the former corresponds to the generator/parity check matrix of the latter. In particular, we showed how the encoding and decoding processes of a linear cyclic code correspond respectively to the computation of a preimage of the null vector $\underline{0}$ and the application of the global rule of a linear CA. Further, we observed that the resiliency order of a linear CA determines the minimum distance of the associated code. We then summarized this discussion by showing how the $(7, 4, 3)$ cyclic Hamming code can be implemented by a linear CA with diameter $d = 5$.

In the second part of this chapter, we introduced a new cryptographic property for CA which could be of relevance in the design of side-channel countermeasures, namely Asynchrony Immunity. After formally defining the property, we proved that the basic operations of reflection and complement preserve asynchrony immunity. We finally performed a computer search of all $(3, 10)$–asynchrony immune CA by classifying them with respect to their nonlinearity, remarking that all their local rules are center permutive.

Some possible future directions of research on the topic are the following. First, about the coding-theoretic part of our investigation, cyclic codes form a broad class including for example *BCH* and *Reed-Solomon* codes [117]. Hence, it could be interesting to investigate how to implement these codes through CA by elaborating on the method presented in Chapter 12.

As we mentioned in Chapter 4, *MDS codes* are also employed to design the *diffusion layers* of block ciphers, such as the MixColumns operation of Rijndael, the encryption algorithm which constitutes the AES standard (see [53]). Thus, another direction of research worth exploring is to consider the design of MDS codes by means of linear CA for lightweight implementations of diffusion linear layers.

Regarding asynchrony immunity, the fact that all CA rules found by our computer search are center permutive suggests that center-permutivity is a necessary condition for asynchrony immunity, a prop-

erty that would greatly reduce the search space for possible AI candidates with interesting cryptographic properties. For future research, we thus plan to investigate Conjecture 1.

Another possible direction to explore is related to the maximum nonlinearity achievable by local rules of asynchrony immune CA. For instance, an interesting question could be to verify if it is possible to design an infinite family of asynchrony immune CA whose local rules are bent or plateaued. A preliminary step to accomplish this task could be to see whether it is possible to characterize asynchrony immunity by using the Walsh spectrum of the CA global rules. We suspect that this characterization can indeed be obtained, due to the fact that resiliency and asynchrony immunity have very similar definitions.

Finally, a fact which could be useful for computer search of AI cellular automata is that an infinite CA is surjective if and only if its finite counterpart is balanced for all lengths $n \in \mathbb{N}$, as remarked in Chapter 2. Thus, it would make sense to limit the search only to surjective CA, by adapting for instance Amoroso and Patt's algorithm [3].

15.3 COMBINATORIAL DESIGNS AND CELLULAR AUTOMATA

15.3.1 *Orthogonal Latin Squares from Linear CA*

In Chapter 13 we began investigating the connections between combinatorial designs and cellular automata, namely focusing on the generation of Orthogonal Latin Squares (OLS) using linear CA. In particular, we showed that any bipermutive CA with diameter d and length $n = 2(d-1)$ can generate a Latin square of order q^{d-1}, with q being the size of the CA state alphabet. Successively, we characterized the pairs of linear bipermutive CA generating OLS as those whose associated Sylvester matrix is invertible. This led us to conclude that two linear bipermutive CA generate a pair of OLS if and only if the polynomials associated to their local rules are relatively prime. We then used this result to describe a $(2, n)$–threshold secret sharing scheme and a perfect authentication code based on linear CA.

In the subsequent part of the chapter we focused on counting pairs of coprime polynomials, or equivalently counting pairs of OLS induced by linear CA. The peculiarity of this problem, which makes it different from all other counting results regarding coprime polynomials already published in the literature, is that both polynomials composing the pair must have a nonzero constant term, in order to ensure the bipermutivity property of the resulting CA local rules.

We presented two approaches to solve this counting problem over the finite field $\mathbb{F}_2$. In particular, the first approach is based on an equivalence relation over polynomial pairs induced by Euclid's algorithm. However, in order to prove the counting formula for the num-

ber of coprime polynomials with this method, we relied on a conjecture about the proportion of non-coprime polynomial pairs which seemed to hold from the experimental point of view. Then, in the second approach, we tackled the counting problem by solving a recurrence equation, whose closed-form formula determines the number of coprime polynomial pairs of our interest as well as it proves the conjecture raised in the first approach. Interestingly, the integer sequence associated with this recurrence equation turned out to be already published in the OEIS as sequence A002450, which is known for several other combinatorial and number-theoretic facts that are not related to polynomials or OLS.

Finally, in the last part of the chapter we presented a construction for sets of linear CA-based Mutually Orthogonal Latin Squares (MOLS) based on irreducible polynomials. In particular, the computer search which we performed up to degree 6 made us conjecture that this construction is optimal.

There are several opportunities for further improvements on the results presented in this chapter.

A first direction for future research would be to extend our investigation to orthogonal arrays of the form $t-(v,k,1)$–OA generated by linear CA, with $t > 2$. A characterization result for such kind of OA would allow one to design a general (t,n)-threshold secret sharing scheme based on CA, or equivalently to design linear MDS codes through CA. A possible idea to achieve this result would be to first characterize which subclass of bipermutive CA generate *Latin hypercubes*, i. e. the generalization of Latin squares to higher dimensions. From there, the next step would be to characterize sets of linear CA inducing Orthogonal Latin Hypercubes (OLH), which are equivalent to orthogonal arrays [92]. We suspect that such kind of k–dimensional orthogonal hypercubes can always be obtained by sets of n linear CA whose associated polynomials are k-wise relatively prime, i. e. every subset of k polynomials out of n does not have any common factor. However, we note that there are no straightforward ways to generalize the concept of resultant to more than two polynomials [69]. As a matter of fact, some of the existing generalizations involve matrices which do not correspond to those related to hypercubes generated by CA. To the best of our knowledge, the only resultant matrix for several polynomials that most resemble the CA hypercube case has been defined in [54], which could thus represent a starting point for future work on the subject.

A natural open problem regarding the count of coprime polynomials is to generalize our formula for polynomials over a generic finite field $\mathbb{F}_q$, with $q > 2$. Then, it would be interesting to verify if the resulting integer sequences are already reported in the OEIS, like in the $q = 2$ case with sequence A002450. Additionally, another idea is to further develop the connection between the number of non-coprime

pairs in $P_n^{1,1}$ and the Gaussian binomial coefficient. In particular, since $\binom{n-1}{2}_2$ corresponds to the number of subspaces of dimension 2 of $\mathbb{F}_2^{n-1}$ (see [29]), an interesting question is whether it is possible to describe a bijection between these subspaces and the non-coprime polynomial pairs in $P_n^{1,1}$.

Finally, two additional open problems concern either proving or disproving Conjecture 3 about the optimality of our construction for sets of MOLS based on linear CA, as well as deriving a better lower bound for the number $T_{n,q}$ of such families.

15.3.2 *Orthogonal Latin Squares from Nonlinear CA*

In Chapter 14 we generalized our investigation of CA-based orthogonal Latin squares by considering nonlinear bipermutive CA, motivated by the design of *cheater-immune secret sharing schemes* (SSS). In the context of threshold acces structures, in particular, cheater-immune SSS are equivalent to orthogonal arrays arising from nonlinear constructions.

We tackled this investigation by adopting both a combinatorial and a heuristic approach. In the combinatorial approach, we first showed some basic invariance properties of orthogonal Latin squares generated by CA. In particular, we showed that if two bipermutive CA induce orthogonal Latin squares, then the Latin squares of the reflected and complemented CA are orthogonal as well. Successively, we introduced the definition of *pairwise balancedness*, proving that it is a necessary condition satisfied by every pair of bipermutive CA generating OLS. Additionally, we showed that pairwise balancedness is a property preserved from the generating functions to the corresponding bipermutive rules, but not vice versa. We then focused on determining the number of pairwise balanced generating functions and the number of pairwise balanced bipermutive rules. In the latter case, we used a graph-based representation of bipermutive functions which allowed us to encode a pair of pairwise rule as a special labeling on the nodes of this graph. Finally, we employed a combinatorial algorithm to explore the space of all pairwise bipermutive rules up to $d = 6$ variables, retaining only those that generate OLS, and we classified them with respect to their nonlinearity.

In the heuristic approach, we adopted Genetic Algorithms and Genetic Programming to evolve pairs of OLS generated by nonlinear bipermutive CA equipped with rules of diameter $d = 7$ and $d = 8$, which are the smallest problem instances not amenable to exhaustive search. More specifically, we experimented with three different encodings for GA and with a single encoding for GP. The third GA encoding, in particular, is based on a characterization of pairwise balanced generating functions as quaternary strings. We then defined two fitness functions, the former taking into account only the orthogonality

property of the candidate solutions and the latter also considering their nonlinearity. Our experiments showed that GP is always able to find optimal solutions, but under the first fitness function all resulting OLS arise from linear CA. On the other hand, using the second fitness function all optimal solutions found by GP are nonlinear. GA, on the contrary, always manages to find OLS generated by nonlinear CA, even under the first fitness function. However, GA performs consistently worse than GP in the $d = 8$ problem instance.

Concerning the future directions of research for the combinatorial approach, we plan to investigate sufficient conditions that two nonlinear bipermutive CA must satisfy in order to generate orthogonal Latin squares. Another direction worth investigating is to analyze the pairs of nonlinear rules found in this chapter by exhaustive search from the perspective of *pseudorandom number generation*, and compare them with others stemming from different classifications, like those presented in [63, 106].

Regarding the heuristic approach, it would be interesting to compare the performance of GA and GP with other optimization algorithms. Since the objects we are dealing with in this optimization problem are Boolean functions used as CA rules, one could leverage on the research about the heuristic optimization of Boolean functions with good cryptographic properties, a problem which we also addressed in this thesis. A possible idea would be to explore both population-based approaches like the discrete PSO discussed in Chapter 8 and Cartesian GP [142, 143], as well as local search methods such as Simulated Annealing [43]. A different comparison perspective worth exploring would also be to adapt algebraic constructions of Boolean functions evolved through GP [141] in order to generate orthogonal Latin squares.

Another interesting experimental direction to investigate would be to increase the number of variables of the generating functions, to assess up to which dimension of the problem GP is able to produce optimal solutions. Finally, one could also consider the natural extension of evolving k *Mutually Orthogonal Latin Squares* (MOLS) based on CA. In this case, the encoding is a straightforward extension of the double tree representation, since it suffices to represent a candidate solution with k independent trees. The fitness function can also be easily modified by summing the number of repeated pairs in each superposition of the k Latin squares.

BIBLIOGRAPHY

[1] A. Adamatzky, "Mathematical basis of cellular automata, introduction to," in *Encyclopedia of Complexity and Systems Science*, 2009, pp. 5438–5440 (cit. on p. 11).

[2] H. E. Aguirre, H. Okazaki, and Y. Fuwa, "An evolutionary multiobjective approach to design highly non-linear boolean functions," in *Genetic and Evolutionary Computation Conference, GECCO 2007, Proceedings, London, England, UK, July 7-11, 2007*, 2007, pp. 749–756 (cit. on pp. 2, 88, 95, 96).

[3] S. Amoroso and Y. N. Patt, "Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures," *J. Comput. Syst. Sci.*, vol. 6, no. 5, pp. 448–464, 1972 (cit. on p. 206).

[4] D. Ashlock, "Finding designs with genetic algorithms," in *Computational and Constructive Design Theory*, Springer, 1996, pp. 49–65 (cit. on p. 179).

[5] G. V. Assche, *Quantum cryptography and secret-key distillation*. Cambridge University Press, 2006 (cit. on p. 40).

[6] H. Balzter, P. W. Braun, and W. Köhler, "Cellular automata models for vegetation dynamics," *Ecological modelling*, vol. 107, no. 2, pp. 113–125, 1998 (cit. on p. 11).

[7] S. Bandini, F. Rubagotti, G. Vizzari, and K. Shimura, "A cellular automata based model for pedestrian and group dynamics: motivations and first experiments," in *Parallel Computing Technologies - 11th International Conference, PaCT 2011, Kazan, Russia, September 19-23, 2011. Proceedings*, 2011, pp. 125–139 (cit. on p. 11).

[8] D. Belazzougui, T. Gagie, V. Mäkinen, and M. Previtali, "Fully dynamic de Bruijn graphs," in *String Processing and Information Retrieval - 23rd International Symposium, SPIRE 2016, Beppu, Japan, October 18-20, 2016, Proceedings*, 2016, pp. 145–152 (cit. on p. 18).

[9] D. Belazzougui, T. Gagie, V. Mäkinen, M. Previtali, and S. J. Puglisi, "Bidirectional variable-order de Bruijn graphs," in *LATIN 2016: Theoretical Informatics - 12th Latin American Symposium, Ensenada, Mexico, April 11-15, 2016, Proceedings*, 2016, pp. 164–178 (cit. on p. 18).

[10] A. T. Benjamin and C. D. Bennett, "The probability of relatively prime polynomials," *Mathematics Magazine*, vol. 80, no. 3, pp. 196–202, 2007 (cit. on pp. 164, 165).

[11] A. T. Benjamin and J. J. Quinn, *Proofs that really count: the art of combinatorial proof*, 27. MAA, 2003 (cit. on p. 172).

[12] E. R. Berlekamp, "Factoring polynomials over finite fields," *Bell System Technical Journal*, vol. 46, no. 8, pp. 1853–1859, 1967 (cit. on p. 122).

[13] E. R. Berlekamp and L. R. Welch, "Weight distributions of the cosets of the (32, 6) Reed-Muller code," *IEEE Trans. Information Theory*, vol. 18, no. 1, pp. 203–207, 1972 (cit. on pp. 60, 74).

[14] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "Radiogatún, a belt-and-mill hash function," *IACR Cryptology ePrint Archive*, vol. 2006, p. 369, 2006 (cit. on p. 76).

[15] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, *The* Keccak *reference*, 2011. [Online]. Available: http://keccak.noekeon.org/ (cit. on pp. 3, 76, 127, 151).

[16] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *J. Cryptology*, vol. 4, no. 1, pp. 3–72, 1991 (cit. on pp. 63, 64).

[17] G. R. Blakley *et al.*, "Safeguarding cryptographic keys," in *Proceedings of the national computer conference*, vol. 48, 1979, pp. 313–317 (cit. on pp. 4, 48).

[18] C. Böhm and G. Jacopini, "Flow diagrams, Turing machines and languages with only two formation rules," *Commun. ACM*, vol. 9, no. 5, pp. 366–371, 1966 (cit. on p. 1).

[19] R. C. Bose and S. S. Shrikhande, "On the construction of sets of mutually orthogonal latin squares and the falsity of a conjecture of Euler," *Transactions of the American Mathematical Society*, vol. 95, no. 2, pp. 191–209, 1960 (cit. on p. 28).

[20] M. Boyle and B. Kitchens, "Periodic points for onto cellular automata," *Indagationes Mathematicae*, vol. 10, no. 4, pp. 483–493, 1999 (cit. on p. 109).

[21] M. Boyle and B. Lee, "Jointly periodic points in cellular automata: computer explorations and conjectures," *Experimental Mathematics*, vol. 16, no. 3, pp. 293–302, 2007 (cit. on p. 109).

[22] D. Bozilov, B. Bilgin, and H. A. Sahin, "A note on 5-bit quadratic permutations' classification," *IACR Trans. Symmetric Cryptol.*, vol. 2017, no. 1, pp. 398–404, 2017 (cit. on p. 205).

[23] K. Browning, J. Dillon, M. McQuistan, and A. Wolfe, "An APN permutation in dimension six," *Finite Fields: theory and applications*, vol. 518, pp. 33–42, 2010 (cit. on pp. 64, 138, 140).

[24] N. G. de Bruijn, *Acknowledgement of priority to C. Flye Sainte-Marie on the counting of circular arrangements of 2n zeros and ones that show each n-letter word exactly once*. Department of Mathematics, Technological University, 1975 (cit. on p. 18).

[25] L. Budaghyan, C. Carlet, T. Helleseth, N. Li, and B. Sun, "On upper bounds for algebraic degrees of APN functions," *IEEE Transactions on Information Theory*, 2017, ISSN: 0018-9448. DOI: 10.1109/TIT.2017.2757938 (cit. on p. 204).

[26] L. Burnett, G. Carter, E. Dawson, and W. Millan, "Efficient methods for generating MARS-like s-boxes," in *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, 2000, pp. 300–314 (cit. on p. 80).

[27] L. Burnett, W. Millan, E. Dawson, and A. J. Clark, "Simpler methods for generating better boolean functions with good cryptographic properties," *Australasian J. Combinatorics*, vol. 29, pp. 231–248, 2004 (cit. on p. 106).

[28] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. Matyas Jr, L. O'Connor, M. Peyravian, D. Safford, *et al.*, "MARS-a candidate cipher for AES," *NIST AES Proposal*, vol. 268, 1998 (cit. on p. 80).

[29] P. J. Cameron, *Notes on Counting: An Introduction to Enumerative Combinatorics*. Cambridge University Press, 2017, vol. 26 (cit. on p. 208).

[30] C. Carlet, "Boolean functions for cryptography and error correcting codes," *Boolean models and methods in mathematics, computer science, and engineering*, vol. 2, pp. 257–397, 2010 (cit. on pp. 1, 2, 45, 53, 55, 59, 60, 83, 88, 134).

[31] ——, "Vectorial boolean functions for cryptography," *Boolean models and methods in mathematics, computer science, and engineering*, vol. 134, pp. 398–469, 2010 (cit. on pp. 1, 53, 63, 64, 66).

[32] C. Carlet, P. Charpin, and V. A. Zinoviev, "Codes, bent functions and permutations suitable for des-like cryptosystems," *Des. Codes Cryptography*, vol. 15, no. 2, pp. 125–156, 1998 (cit. on pp. 64, 66).

[33] G. Cattaneo, A. Dennunzio, and L. Margara, "Solution of some conjectures about topological properties of linear cellular automata," *Theor. Comput. Sci.*, vol. 325, no. 2, pp. 249–271, 2004 (cit. on pp. 110, 124).

[34] J. Cervelle, A. Dennunzio, and E. Formenti, "Chaotic behavior of cellular automata," in *Encyclopedia of Complexity and Systems Science*, 2009, pp. 978–989 (cit. on p. 11).

[35] F. Chabaud and S. Vaudenay, "Links between differential and linear cryptanalysis," in *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, 1994, pp. 356–365 (cit. on p. 63).

[36] P. Charpin and E. Pasalic, "On propagation characteristics of resilient functions," in *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, 2002, pp. 175–195 (cit. on p. 60).

[37] G. Chassé, "Some remarks on a LFSR "disturbed" by other sequences," in *EUROCODE '90, International Symposium on Coding Theory and Applications, Udine, Italy, November 5-9, 1990, Proceedings*, 1990, pp. 215–221 (cit. on p. 118).

[38] D. Chaum, C. Crépeau, and I. Damgård, "Multiparty unconditionally secure protocols (extended abstract)," in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, 1988, pp. 11–19 (cit. on p. 47).

[39] V. V. Chepyzhov and B. J. M. Smeets, "On A fast correlation attack on certain stream ciphers," in *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, 1991, pp. 176–185 (cit. on p. 57).

[40] B. Chopard, "Cellular automata modeling of physical systems," in *Encyclopedia of Complexity and Systems Science*, 2009, pp. 865–892 (cit. on p. 11).

[41] B. Chor, O. Goldreich, J. Håstad, J. Friedman, S. Rudich, and R. Smolensky, "The bit extraction problem of t-resilient functions (preliminary version)," in *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, 1985, pp. 396–407 (cit. on p. 149).

[42] J. A. Clark, J. L. Jacob, S. Maitra, and P. Stanica, "Almost boolean functions: the design of boolean functions by spectral inversion," *Computational Intelligence*, vol. 20, no. 3, pp. 450–462, 2004 (cit. on pp. 79, 80, 83, 97, 98, 100, 103–106, 200, 201).

[43] J. A. Clark, J. L. Jacob, S. Stepney, S. Maitra, and W. Millan, "Evolving boolean functions satisfying multiple criteria," in *Progress in Cryptology - INDOCRYPT 2002, Third International Conference on Cryptology in India, Hyderabad, India, December 16-18, 2002*, 2002, pp. 246–259 (cit. on pp. 2, 88, 94–96, 103, 106, 199, 209).

[44] J. A. Clark and J. Jacob, "Two-stage optimisation in the design of boolean functions," in *Information Security and Privacy, 5th Australasian Conference, ACISP 2000, Brisbane, Australia, July 10-12, 2000, Proceedings*, 2000, pp. 242–254 (cit. on p. 79).

[45] A. Clarridge and K. Salomaa, "A cryptosystem based on the composition of reversible cellular automata," in *Language and Automata Theory and Applications, Third International Conference,*

LATA 2009, Tarragona, Spain, April 2-8, 2009. Proceedings, 2009, pp. 314–325 (cit. on p. 204).

[46] C. J. Colbourn and J. H. Dinitz, "Combinatorial designs," in *Handbook of Discrete and Combinatorial Mathematics*. 1999 (cit. on pp. 26–28).

[47] Y. Crama and P. L. Hammer, *Boolean models and methods in mathematics, computer science, and engineering*. Cambridge University Press, 2010, vol. 2 (cit. on p. 53).

[48] T. W. Cusick and P. Stanica, *Cryptographic Boolean functions and applications*. Academic Press, 2017 (cit. on p. 53).

[49] E. Czeizler and J. Kari, "A tight linear bound on the neighborhood of inverse cellular automata," in *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, 2005, pp. 410–420 (cit. on pp. 21, 204).

[50] J. Daemen and C. S. K. Clapp, "Fast Hashing and Stream Encryption with PANAMA," in *Fast Software Encryption, 5th International Workshop, FSE '98, Paris, France, March 23-25, 1998, Proceedings*, 1998, pp. 60–74 (cit. on p. 76).

[51] J. Daemen, R. Govaerts, and J. Vandewalle, "An efficient nonlinear shift-invariant transformation," in *Proceedings of the 15th Symposium on Information Theory in the Benelux, B. Macq, Ed., Werkgemeenschap voor Informatie - En Communicatietheorie*, 1994, pp. 108–115 (cit. on pp. 75, 76).

[52] J. Daemen, R. Govaerts, and J. Vandewalle, "Invertible shift-invariant transformations on binary arrays," *Applied Mathematics and Computation*, vol. 62, no. 2, pp. 259 –277, 1994 (cit. on pp. 3, 75, 76, 127).

[53] J. Daemen and V. Rijmen, *The Design of Rijndael*. Springer-Verlag New York, Inc., 2002 (cit. on p. 205).

[54] J. Deißler, "A resultant for Hensel's lemma," *arXiv preprint arXiv:1301.4073*, 2013 (cit. on p. 207).

[55] A. Dennunzio, P. D. Lena, E. Formenti, and L. Margara, "Periodic orbits and dynamical complexity in cellular automata," *Fundam. Inform.*, vol. 126, no. 2-3, pp. 183–199, 2013 (cit. on p. 109).

[56] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Information Theory*, vol. 22, no. 6, pp. 644–654, 1976 (cit. on p. 37).

[57] B. Durand, "Global properties of cellular automata," in *Cellular Automata and Complex Systems*, Springer, 1999, pp. 1–22 (cit. on pp. 20, 21, 201).

[58] M. Ebner, "On the search space of genetic programming and its relation to nature's search space," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 2, 1999, pp. 1357–1361 (cit. on p. 192).

[59] K. Eloranta, "Partially permutive cellular automata," *Nonlinearity*, vol. 6, no. 6, pp. 1009–1023, 1993 (cit. on p. 159).

[60] Z. Eslami, S. H. Razzaghi, and J. Zarepour-Ahmadabadi, "Secret image sharing based on cellular automata and steganography," *Pattern Recognition*, vol. 43, no. 1, pp. 397–404, 2010 (cit. on p. 77).

[61] Z. Eslami and J. Zarepour-Ahmadabadi, "A verifiable multi-secret sharing scheme based on cellular automata," *Inf. Sci.*, vol. 180, no. 15, pp. 2889–2894, 2010 (cit. on p. 77).

[62] L. Euler, *Recherches sur une nouvelle espece de quarres magiques*. Zeeuwsch Genootschao, 1782 (cit. on pp. 26, 28).

[63] E. Formenti, K. Imai, B. Martin, and J. Yunès, "Advances on random sequence generation by uniform cellular automata," in *Computing with New Resources - Essays Dedicated to Jozef Gruska on the Occasion of His 80th Birthday*, C. Calude, R. Freivalds, and I. Kazuo, Eds., 2014, pp. 56–70 (cit. on pp. 2, 74, 127, 209).

[64] E. Formenti, C. Papazian, and P.-A. Scribot, "Additive flowers," in *CIBB 2014*, 2014 (cit. on p. 109).

[65] J. Fuller, W. Millan, and E. Dawson, "Multi-objective optimisation of bijective s-boxes," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2004, 19-23 June 2004, Portland, OR, USA*, 2004, pp. 1525–1532 (cit. on p. 80).

[66] M. Gardner, "Mathematical Games: The fantastic combinations of John Conway's new solitaire game "Life"," *Scientific American*, vol. 223, pp. 120–123, 1970 (cit. on p. 11).

[67] W. Gardner, "Introduction to einstein's contribution to time-series analysis," *IEEE ASSP Magazine*, vol. 4, no. 4, pp. 4–5, 1987 (cit. on p. 55).

[68] C. F. Gauß, *Disquisitiones arithmeticae*. Humboldt-Universität zu Berlin, 1801 (cit. on p. 174).

[69] I. M. Gelfand, M. Kapranov, and A. Zelevinsky, *Discriminants, resultants, and multidimensional determinants*. Springer Science & Business Media, 2008 (cit. on p. 207).

[70] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1989 (cit. on p. 69).

[71] D. E. Goldberg, R. Lingle, *et al.*, "Alleles, loci, and the traveling salesman problem," in *Proceedings of an international conference on genetic algorithms and their applications*, vol. 154, 1985, pp. 154–159 (cit. on p. 98).

[72] O. Goldreich, *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001 (cit. on p. 38).

[73] ——, *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004 (cit. on p. 38).

[74] H. Gutowitz, "Cryptography with dynamical systems," in *Cellular Automata and Cooperative Systems*, Springer, 1993, pp. 237–274 (cit. on pp. 2, 75, 78).

[75] R. W. Hamming, *Coding and information theory (2. ed.)* Prentice Hall, 1986 (cit. on pp. 146, 147).

[76] G. A. Hedlund, "Endomorphisms and automorphisms of the shift dynamical systems," *Mathematical Systems Theory*, vol. 3, no. 4, pp. 320–375, 1969 (cit. on pp. 11, 14, 18, 21).

[77] M. Hell, T. Johansson, A. Maximov, and W. Meier, "The grain family of stream ciphers," in *New Stream Cipher Designs - The eSTREAM Finalists*, 2008, pp. 179–190 (cit. on p. 202).

[78] J. J. Hoch and A. Shamir, "Fault analysis of stream ciphers," in *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, 2004, pp. 240–253 (cit. on p. 143).

[79] J. H. Holland, "Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.," 1975 (cit. on p. 69).

[80] X. Hou and G. L. Mullen, "Number of irreducible polynomials and pairs of relatively prime polynomials in several variables over finite fields," *Finite Fields and Their Applications*, vol. 15, no. 3, pp. 304–331, 2009 (cit. on p. 164).

[81] R. Hrbacek and V. Dvorak, "Bent function synthesis by means of cartesian genetic programming," in *Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*, 2014, pp. 414–423 (cit. on p. 80).

[82] X. Hu, R. C. Eberhart, and Y. Shi, "Swarm intelligence for permutation optimization: a case study of n-queens problem," in *2003 IEEE Swarm Intelligence Symposium, SIS 2003, Indianapolis, IN, USA, April 24-26, 2003*, 2003, pp. 243–246 (cit. on pp. 84–86).

[83] A. Ilachinski, *Cellular automata: a discrete universe*. World Scientific Publishing Co Inc, 2001 (cit. on p. 11).

[84] T. O. E. of Integer Sequences (OEIS), *Sequence* A002450, accessed 29 september 2016. [Online]. Available: http://oeis.org/A002450 (cit. on p. 172).

[85] ——, *Sequence* A002860, accessed 5 november 2017. [Online]. Available: http://oeis.org/A002860 (cit. on p. 27).

[86] ——, *Sequence* A006095, accessed 29 september 2016. [Online]. Available: http://oeis.org/A006095 (cit. on p. 173).

[87] J. Kari, "Theory of cellular automata: A survey," *Theor. Comput. Sci.*, vol. 334, no. 1-3, pp. 3–33, 2005 (cit. on p. 11).

[88] ——, "Basic concepts of cellular automata," in *Handbook of Natural Computing*, 2012, pp. 3–24 (cit. on p. 15).

[89] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014 (cit. on p. 38).

[90] S. Kavut, "Results on rotation-symmetric S-boxes," *Inf. Sci.*, vol. 201, pp. 93–113, 2012 (cit. on p. 77).

[91] S. Kavut, S. Maitra, and M. D. Yücel, "Search for boolean functions with excellent profiles in the rotation symmetric class," *IEEE Trans. Information Theory*, vol. 53, no. 5, pp. 1743–1751, 2007 (cit. on p. 58).

[92] A. D. Keedwell and J. Dénes, *Latin squares and their applications*. Elsevier, 2015 (cit. on pp. 26, 27, 207).

[93] J Kennedy and R Eberhart, "Particle swarm optimization," in *proceedings of IEEE International Conference on neural networks (ICNN'95)*, 1995 (cit. on p. 71).

[94] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 5, 1997, pp. 4104–4108 (cit. on pp. 72, 85, 89).

[95] A. Kerckhoff, "La cryptographie militaire," *Journal des sciences militaires*, vol. IX, no. 5–38, Janvier 1883 (cit. on p. 36).

[96] ——, "La cryptographie militaire," *Journal des sciences militaires*, vol. IX, no. 161-191, Février 1883 (cit. on p. 36).

[97] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," in *Spin Glass Theory and Beyond: An Introduction to the Replica Method and Its Applications*, World Scientific, 1987, pp. 339–348 (cit. on p. 68).

[98] L. R. Knudsen, "Truncated and higher order differentials," in *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, 1994, pp. 196–211 (cit. on p. 63).

[99] D Knuth, *The art of computer programming, vol. 4, pre-fascicle 3a*, 2011 (cit. on p. 186).

[100] C. Koc and A. Apohan, "Inversion of cellular automata iterations," *IEE Proceedings-Computers and Digital Techniques*, vol. 144, no. 5, pp. 279–284, 1997 (cit. on pp. 2, 74, 127).

[101] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992, vol. 1 (cit. on p. 70).

[102] P. Kurka, "Topological dynamics of cellular automata," in *Encyclopedia of Complexity and Systems Science*, 2009, pp. 9246–9268 (cit. on pp. 15, 202).

[103] J. C. Lagarias, *The ultimate challenge: The 3x+ 1 problem*. American Mathematical Soc., 2010 (cit. on p. 172).

[104] G. Leander and A. Poschmann, "On the Classification of 4-Bit S-Boxes," in *Arithmetic of Finite Fields*, ser. Lecture Notes in Computer Science, C. Carlet and B. Sunar, Eds., vol. 4547, Springer Berlin Heidelberg, 2007, pp. 159–176 (cit. on pp. 135, 140, 203).

[105] A. Leporati and L. Mariot, "1-Resiliency of bipermutive cellular automata rules," in *Cellular Automata and Discrete Complex Systems - 19th International Workshop, AUTOMATA 2013, Gießen, Germany, September 17-19, 2013. Proceedings*, 2013, pp. 110–123 (cit. on pp. 144, 185, 205).

[106] ——, "Cryptographic properties of bipermutive cellular automata rules," *J. Cellular Automata*, vol. 9, no. 5-6, pp. 437–475, 2014 (cit. on pp. 2, 74, 127, 143, 209).

[107] R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications*. Cambridge university press, 1994 (cit. on pp. 23–25, 122, 161, 164, 176).

[108] J. Liu, S. Mesnager, and L. Chen, "On the diffusion property of iterated functions," in *Cryptography and Coding - 15th IMA International Conference, IMACC 2015, Oxford, UK, December 15-17, 2015. Proceedings*, 2015, pp. 239–253 (cit. on p. 77).

[109] O. A. Logachev, A. A. Sal_nikov, and V. I_A_shchenko, *Boolean functions in coding theory and cryptography*. American Mathematical Soc., 2012, vol. 241 (cit. on p. 53).

[110] L. Manzoni, "Asynchronous cellular automata and dynamical properties," *Natural Computing*, vol. 11, no. 2, pp. 269–276, 2012 (cit. on p. 152).

[111] G. Á. Marañón, L. H. Encinas, and Á. M. del Rey, "A multi-secret sharing scheme for color images based on cellular automata," *Inf. Sci.*, vol. 178, no. 22, pp. 4382–4395, 2008 (cit. on p. 77).

[112] L. Mariot and A. Leporati, "Sharing secrets by computing preimages of bipermutive cellular automata," in *Cellular Automata - 11th International Conference on Cellular Automata for Research and Industry, ACRI 2014, Krakow, Poland, September 22-25, 2014. Proceedings*, 2014, pp. 417–426 (cit. on pp. 3, 4, 7, 77–79, 109, 157, 159, 201).

[113] ——, "A cryptographic and coding-theoretic perspective on the global rules of cellular automata," *Natural Computing*, 2017, ISSN: 1572-9796. DOI: 10.1007/s11047-017-9635-0 (cit. on p. 203).

[114] B. Martin, "A Walsh exploration of elementary CA rules," *J. Cellular Automata*, vol. 3, no. 2, pp. 145–156, 2008 (cit. on pp. 2, 74, 127).

[115] J. L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Information Theory*, vol. 15, no. 1, pp. 122–127, 1969 (cit. on pp. 56, 122).

[116] M. Matsui, "Linear cryptanalysis method for DES cipher," in *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, 1993, pp. 386–397 (cit. on p. 63).

[117] R. McEliece, *The theory of information and coding*. Cambridge University Press, 2002 (cit. on pp. 29, 32, 57, 146, 205).

[118] W. Meier and O. Staffelbach, "Analysis of pseudo random sequence generated by cellular automata," in *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, 1991, pp. 186–199 (cit. on pp. 2, 74, 127).

[119] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004 (cit. on p. 72).

[120] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs*, ser. Artificial intelligence. Springer, 1992 (cit. on p. 90).

[121] W. Millan, L. Burnett, G. Carter, A. J. Clark, and E. Dawson, "Evolutionary heuristics for finding cryptographically strong s-boxes," in *Information and Communication Security, Second International Conference, ICICS'99, Sydney, Australia, November 9-11, 1999, Proceedings*, 1999, pp. 263–274 (cit. on p. 80).

[122] W. Millan, A. J. Clark, and E. Dawson, "An effective genetic algorithm for finding highly nonlinear boolean functions," in *Information and Communication Security, First International Conference, ICICS'97, Beijing, China, November 11-14, 1997, Proceedings*, 1997, pp. 149–158 (cit. on p. 79).

[123] ——, "Heuristic design of cryptographically strong balanced boolean functions," in *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, 1998, pp. 489–499 (cit. on pp. 2, 70, 79, 84, 87–89, 94, 95, 101, 191, 192, 200).

[124] J. F. Miller and P. Thomson, "Cartesian genetic programming," in *Genetic Programming, European Conference, Edinburgh, Scotland, UK, April 15-16, 2000, Proceedings*, 2000, pp. 121–132 (cit. on p. 71).

[125] C. Moore, "Predicting nonlinear cellular automata quickly by decomposing them into linear ones," *Physica D: Nonlinear Phenomena*, vol. 111, no. 1-4, pp. 27–41, 1998 (cit. on p. 159).

[126] C. Moore, A. A. Drisko, *et al.*, "Algebraic properties of the block transformation on cellular automata," *Complex Systems*, vol. 10, no. 3, pp. 185–194, 1996 (cit. on p. 159).

[127] A. Moraglio, K. Krawiec, and C. G. Johnson, "Geometric semantic genetic programming," in *Parallel Problem Solving from Nature - PPSN XII - 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part I*, 2012, pp. 21–31 (cit. on p. 71).

[128] K. Nyberg, "Perfect Nonlinear S-Boxes," in *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, ser. Lecture Notes in Computer Science, vol. 547, Springer, 1991, pp. 378–386 (cit. on p. 64).

[129] ——, "S-boxes and round functions with controllable linearity and differential uniformity," in *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, 1994, pp. 111–130 (cit. on p. 129).

[130] K. Nyberg and L. R. Knudsen, "Provable security against differential cryptanalysis," in *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, USA, August 16-20, 1992, Proceedings*, 1992, pp. 566–574 (cit. on p. 64).

[131] E. Pasalic and T. Johansson, "Further results on the relation between nonlinearity and resiliency for boolean functions," in *Cryptography and Coding, 7th IMA International Conference, Cirencester, UK, December 20-22, 1999, Proceedings*, 1999, pp. 35–44 (cit. on p. 94).

[132] N. J. Patterson and D. H. Wiedemann, "The covering radius of the $(2^{15}, 16)$ reed-muller code is at least 16276," *IEEE Trans. Information Theory*, vol. 29, no. 3, pp. 354–355, 1983 (cit. on p. 90).

[133] J. Pedersen, "Cellular automata as algebraic systems," *Complex Systems*, vol. 6, no. 3, pp. 237–250, 1992 (cit. on p. 159).

[134] M. E. H. Pedersen and A. J. Chipperfield, "Simplifying particle swarm optimization," *Appl. Soft Comput.*, vol. 10, no. 2, pp. 618–628, 2010 (cit. on pp. 84, 89–91).

[135] D. Perrin and J.-É. Pin, *Infinite words: automata, semigroups, logic and games*. Academic Press, 2004, vol. 141 (cit. on p. 17).

[136] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing, 4th Edition*. Prentice Hall, 2012 (cit. on p. 35).

[137] S. Picek, *Applications of evolutionary computation to cryptology. PhD thesis*. 2015. [Online]. Available: `\url{http://www.win.tue.nl/ipa/?event=applications-of-evolutionary-computation-to-cryptology}` (cit. on p. 79).

[138] S. Picek, C. Carlet, D. Jakobovic, J. F. Miller, and L. Batina, "Correlation immunity of boolean functions: an evolutionary algorithms perspective," in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015*, 2015, pp. 1095–1102 (cit. on p. 80).

[139] S. Picek, M. Cupic, and L. Rotim, "A new cost function for evolution of s-boxes," *Evolutionary Computation*, vol. 24, no. 4, pp. 695–718, 2016 (cit. on pp. 80, 140).

[140] S. Picek, S. Guilley, C. Carlet, D. Jakobovic, and J. F. Miller, "Evolutionary approach for finding correlation immune boolean functions of order t with minimal Hamming weight," in *Theory and Practice of Natural Computing - Fourth International Conference, TPNC 2015, Mieres, Spain, December 15-16, 2015. Proceedings*, 2015, pp. 71–82 (cit. on p. 80).

[141] S. Picek and D. Jakobovic, "Evolving algebraic constructions for designing bent boolean functions," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, ACM, 2016, pp. 781–788 (cit. on p. 209).

[142] S. Picek, D. Jakobovic, and M. Golub, "Evolving cryptographically sound boolean functions," in *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013, Companion Material Proceedings*, 2013, pp. 191–192 (cit. on pp. 80, 85, 88, 95, 96, 209).

[143] S. Picek, D. Jakobovic, J. F. Miller, L. Batina, and M. Cupic, "Cryptographic boolean functions: one output, many design criteria," *Appl. Soft Comput.*, vol. 40, pp. 635–653, 2016 (cit. on pp. 2, 80, 209).

[144] S. Picek, E. Marchiori, L. Batina, and D. Jakobovic, "Combining evolutionary computation and algebraic constructions to find cryptography-relevant boolean functions," in *Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*, 2014, pp. 822–831 (cit. on p. 80).

[145] S. Picek, L. Mariot, B. Yang, D. Jakobovic, and N. Mentens, "Design of S-boxes defined with cellular automata rules," in *Proceedings of the Computing Frontiers Conference, CF'17, Siena, Italy, May 15-17, 2017*, 2017, pp. 409–414 (cit. on p. 135).

[146] S. Picek, J. F. Miller, D. Jakobovic, and L. Batina, "Cartesian genetic programming approach for generating substitution boxes of different sizes," in *Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015, Companion Material Proceedings*, 2015, pp. 1457–1458 (cit. on p. 80).

[147] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation," *Journal of Artificial Evolution and Applications*, vol. 2008, 2008 (cit. on p. 72).

[148] R. Poli, W. B. Langdon, and N. F. McPhee, *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008, (With contributions by J. R. Koza) (cit. on pp. 137, 193).

[149] G. Pólya and G. Alexanderson, "Gaussian binomial coefficients.," *Elemente der Mathematik*, vol. 26, pp. 102–109, 1971 (cit. on p. 173).

[150] B. Preneel, W. V. Leekwijck, L. V. Linden, R. Govaerts, and J. Vandewalle, "Propagation characteristics of boolean functions," in *Advances in Cryptology - EUROCRYPT '90, Workshop on the Theory and Application of Cryptographic Techniques, Aarhus, Denmark, May 21-24, 1990, Proceedings*, 1990, pp. 161–173 (cit. on p. 59).

[151] N. J. Radcliffe, "Equivalence class analysis of genetic algorithms," *Complex Systems*, vol. 5, no. 2, 1991 (cit. on p. 90).

[152] A. Reifegerste, "On an involution concerning pairs of polynomials over $\mathbb{F}_2$," *J. Comb. Theory, Ser. A*, vol. 90, no. 1, pp. 216–220, 2000 (cit. on pp. 164, 165).

[153] Á. M. del Rey, J. P. Mateus, and G. R. Sánchez, "A secret sharing scheme based on cellular automata," *Applied Mathematics and Computation*, vol. 170, no. 2, pp. 1356–1364, 2005 (cit. on pp. 4, 77, 78).

[154] D. Richardson, "Tessellations with local transformations," *J. Comput. Syst. Sci.*, vol. 6, no. 5, pp. 373–388, 1972 (cit. on p. 21).

[155] V. Rijmen, P. S. L. M. Barreto, and D. L. G. Filho, "Rotation symmetry in algebraically generated cryptographic substitution tables," *Inf. Process. Lett.*, vol. 106, no. 6, pp. 246–250, 2008 (cit. on pp. 76, 77).

[156] V. Rijmen and J. Daemen, "Advanced encryption standard," *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, pp. 19–22, 2001 (cit. on p. 43).

[157] R Rivest, *Rivest cipher 4 (rc4)*, 1987 (cit. on p. 41).

[158] R. A. Rueppel and O. Staffelbach, "Products of linear recurring sequences with maximum complexity," *IEEE Trans. Information Theory*, vol. 33, no. 1, pp. 124–131, 1987 (cit. on p. 56).

[159] Z. Saber, M. F. Uddin, and A. M. Youssef, "On the existence of (9, 3, 5, 240) resilient functions," *IEEE Trans. Information Theory*, vol. 52, no. 5, pp. 2269–2270, 2006 (cit. on p. 83).

[160] R. Safadi and R. Wang, "The use of genetic algorithms in the construction of mixed multilevel orthogonal arrays," DTIC Document, Tech. Rep., 1992 (cit. on p. 179).

[161] J. L. Schiff, *Cellular automata: a discrete view of the world*. John Wiley & Sons, 2011, vol. 45 (cit. on p. 11).

[162] M. Seredynski and P. Bouvry, "Block encryption using reversible cellular automata," in *Cellular Automata, 6th International Conference on Cellular Automata for Research and Industry, ACRI 2004, Amsterdam, The Netherlands, October 25-28, 2004, Proceedings*, 2004, pp. 785–792 (cit. on pp. 2, 75).

[163] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979 (cit. on pp. 4, 48, 157).

[164] C. E. Shannon, "Communication theory of secrecy systems," *Bell Labs Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949 (cit. on pp. 1, 40, 41, 45).

[165] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII, 7th International Conference, EP98, San Diego, CA, USA, March 25-27, 1998, Proceedings*, 1998, pp. 591–600 (cit. on p. 89).

[166] V. Shoup, "Fast construction of irreducible polynomials over finite fields," *J. Symb. Comput.*, vol. 17, no. 5, pp. 371–391, 1994 (cit. on p. 164).

[167] T. Siegenthaler, "Correlation-immunity of nonlinear combining functions for cryptographic applications," *IEEE Trans. Information Theory*, vol. 30, no. 5, pp. 776–780, 1984 (cit. on pp. 58, 59, 144).

[168] ——, "Decrypting a class of stream ciphers using ciphertext only," *IEEE Trans. Computers*, vol. 34, no. 1, pp. 81–85, 1985 (cit. on p. 58).

[169] G. J. Simmons, "A cartesian product construction for unconditionally secure authentication codes that permit arbitration," *J. Cryptology*, vol. 2, no. 2, pp. 77–104, 1990 (cit. on p. 47).

[170] G. C. Sirakoulis, I. Karafyllidis, C. Mizas, V. A. Mardiris, A. Thanailakis, and P. Tsalides, "A cellular automaton model for the study of DNA sequence evolution," *Comp. in Bio. and Med.*, vol. 33, no. 5, pp. 439–453, 2003 (cit. on p. 11).

[171] D. E. Standard, "Data encryption standard," *Federal Information Processing Standards Publication*, 1999 (cit. on p. 43).

[172] D. R. Stinson, *Cryptography - theory and practice*. CRC Press, 1995 (cit. on pp. 1, 35, 37, 44, 46, 49).

[173] ——, *Combinatorial designs - constructions and analysis*. Springer, 2004 (cit. on pp. 1, 4, 26, 27, 29, 31, 46, 65).

[174] K. Sutner, "De Bruijn graphs and linear cellular automata," *Complex Systems*, vol. 5, no. 1, pp. 19–30, 1991 (cit. on p. 18).

[175] ——, "Cellular automata, decidability and phasespace," *Fundam. Inform.*, vol. 104, no. 1-2, pp. 141–160, 2010 (cit. on p. 21).

[176] M. Szaban and F. Seredynski, "Cryptographically strong s-boxes based on cellular automata," in *Cellular Automata, 8th International Conference on Cellular Automata for Research and Industry, ACRI 2008, Yokohama, Japan, September 23-26, 2008. Proceedings*, 2008, pp. 478–485 (cit. on pp. 2, 75).

[177] Y. Tarannikov, "On resilient boolean functions with maximal possible nonlinearity," in *Progress in Cryptology - INDOCRYPT 2000, First International Conference in Cryptology in India, Calcutta, India, December 10-13, 2000, Proceedings*, 2000, pp. 19–30 (cit. on p. 59).

[178] M. Tarry, *Le Problème des 36 officiers...: Congrès de Paris 1900*. Secrétariat de l'Association française pour l'avancement des sciences, 1900 (cit. on p. 28).

[179] T. Tassa, "Generalized oblivious transfer by secret sharing," *Des. Codes Cryptography*, vol. 58, no. 1, pp. 11–21, 2011 (cit. on p. 48).

[180] T. Toffoli and N. H. Margolus, "Invertible cellular automata: a review," *Physica D: Nonlinear Phenomena*, vol. 45, no. 1-3, pp. 229–253, 1990 (cit. on p. 204).

[181] M. Tompa and H. Woll, "How to share a secret with cheaters," *J. Cryptology*, vol. 1, no. 2, pp. 133–138, 1988 (cit. on pp. 4, 179).

[182] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Inf. Process. Lett.*, vol. 85, no. 6, pp. 317–325, 2003 (cit. on p. 89).

[183] J. Tuliani, "De Bruijn sequences with efficient decoding algorithms," *Discrete Mathematics*, vol. 226, no. 1-3, pp. 313–336, 2001 (cit. on p. 18).

[184] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem," *Proceedings of the London mathematical society*, vol. 2, no. 1, pp. 230–265, 1937 (cit. on p. 1).

[185] S. Ulam, "Random processes and transformations," in *Proceedings of the International Congress on Mathematics*, vol. 2, 1952, pp. 264–275 (cit. on pp. 1, 11).

[186] L. Vanneschi, M. Castelli, L. Manzoni, and S. Silva, "A new implementation of geometric semantic GP and its application to problems in pharmacokinetics," in *Genetic Programming - 16th European Conference, EuroGP 2013, Vienna, Austria, April 3-5, 2013. Proceedings*, 2013, pp. 205–216 (cit. on p. 71).

[187] G. Vernam, *Secret signaling system*, US Patent 1,310,719, 1919 (cit. on p. 45).

[188] J. Von Neumann, *Theory of self-reproducing automata*. Edited by Burks, Arthur W. University of Illinois Press, 1966 (cit. on pp. 1, 11).

[189] S. Wagstaff, *Cunningham project*, `http://homes.cerias.purdue.edu/~ssw/cun/index.html`, Accessed 22 July 2016, 2002 (cit. on p. 122).

[190] A. F. Webster and S. E. Tavares, "On the design of s-boxes," in *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, 1985, pp. 523–534 (cit. on p. 58).

[191] D. Whitley and A. M. Sutton, "Genetic algorithms - A survey of models and methods," in *Handbook of Natural Computing*, 2012, pp. 637–671 (cit. on p. 70).

[192] D. A. Wolf-Gladrow, *Lattice-gas cellular automata and lattice Boltzmann models: an introduction*. Springer, 2004 (cit. on p. 11).

[193] S. Wolfram, "Statistical mechanics of cellular automata," *Reviews of modern physics*, vol. 55, no. 3, p. 601, 1983 (cit. on p. 14).

[194] ——, "Universality and complexity in cellular automata," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1-2, pp. 1–35, 1984 (cit. on p. 11).

[195] ——, "Cryptography with cellular automata," in *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, 1985, pp. 429–432 (cit. on pp. 2, 73).

[196] G. Xiao and J. L. Massey, "A spectral characterization of correlation-immune combining functions," *IEEE Trans. Information Theory*, vol. 34, no. 3, pp. 569–571, 1988 (cit. on p. 58).

[197] X. Yao, "Optimization by genetic annealing," in *Proceedings of the Second Australian Conference on Neural Networks*, Sydney Univ. Electr. Eng., 1991, pp. 94–97 (cit. on p. 200).

[198] D. R. Zerbino and E. Birney, "Velvet: algorithms for de novo short read assembly using de Bruijn graphs," *Genome research*, vol. 18, no. 5, pp. 821–829, 2008 (cit. on p. 18).

[199] K. Zhang and Y. Zheng, "GAC - the criterion for global avalanche characteristics of cryptographic functions," *J. UCS*, vol. 1, no. 5, pp. 320–337, 1995 (cit. on p. 91).

[200] Y. Zheng and X. Zhang, "Plateaued functions," in *Third International Conference on Information and Communications Security, ICICS'99*, V. Varadharajan and Y. Mu, Eds., ser. LNCS, vol. 1726, Springer, 1999, pp. 284–300 (cit. on p. 59).

Milano, Italia
Nice, France

—

December MMXVII