Designing Boolean Functions and Digital Sequences for Cryptology and Communications

Miroslav Marinov Dimitrov

Supervisor: Prof. Tsonka Baicheva

Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Department "Mathematical Foundations of Informatics"

this dissertation is submitted
for the awarding of educational and scientific degree philosophiae doctor
in professional area 4.6
informatics and computer science

Sofia 2023

Acknowledgements

I would like to express my deepest appreciation to my supervisor prof. D.Sc. Tsonka Baicheva for the invaluable patience, feedback, and constructive criticism. I cannot begin to express my thanks for her editing help, late-night feedback sessions, and moral support. I would also like to extend my deepest gratitude to Ph.D. Nikolay Nikolov for the countless interesting conversations regarding the challenges addressed in this thesis and for bringing my attention to the PSL problem in the first place. I'm deeply indebted to prof. Bernhard Esslinger from the university of Siegen for the editing help, support, and trust he invested in me. I'm extremely grateful to the hardware parts he gratuitously sent me to build a mini GPU grid, which successfully solved some of the problems regarding the rotated binary sequences. I am also grateful to Ph.D. Georgi Ivanov who motivated me to start my Ph.D. journey through his recommendations to prof. D.Sc. Tsonka Baicheva. Special thanks to Ph.D. Violeta Andreeva for her unwavering support. Last but not least, I am thankful to my family, especially my parents, who sparked the love of mathematics in me, to my wife Geri, and our wonderful princess Mariah, for the inspiration and emotional support.

Table of contents

List of figures

List of tables

List of Algorithms

Preface

0.1 Scientific contributions

The main scientific contributions could be summarized as follows:

1. A rich collection of popular S-boxes is analyzed in great detail.

2. It is shown that the majority of chaos-based S-boxes are vulnerable to linear cryptanalysis. A simple and lightweight algorithm is proposed, which significantly outperforms all previously published chaos-based S-boxes, in those cryptographic terms, which they utilize for comparison.

3. By introducing some new definitions like couplings, coordinate decomposition, degree of descendibility, and CELAT, the S-box nonlinearity optimization problem is projected to a satisfiability problem, which could be attacked by using SAT solvers.

4. By applying the SAT solver it is shown that 8×8 bijective S-boxes with all eight coordinates having the maximal nonlinearity value of 116 do exist.

5. A strategy of analyzing various spectra channels to detect hidden patterns and anomalies in S-boxes is proposed.

6. A simple and efficient algorithm based on a heuristic search by shotgun hill climbing to construct binary sequences with small peak sidelobe levels (PSL) is proposed. The algorithm successfully revealed binary sequences of lengths between 106 and 300 with record-breaking PSL values.

7. By using some useful properties, the aforementioned algorithm time and memory complexities are reduced to $\mathcal{O}(n)$. This allowed us to reach record-breaking PSL values for less than a second. Moreover, the efficiency range of the algorithm is further extended to binary sequences of longer lengths.

8. A detailed comparison and fine-grain analysis of the proposed algorithms is performed. By using the insights of this analysis, a heuristic algorithm is proposed, which successfully reached all the optimal PSL values known in the literature, which was previously discovered by an exhaustive search. This was achieved by using a low-cost mid-range computer station, while the time required to reach the optimal PSL value for most of the lengths is less than a second.

9. A GPU efficient algorithm addressing the well-known computational problem of finding the lowest possible PSL among the set of a binary sequence B and all binary sequences generated by rotations of B is proposed. The problem is projected to a perfectly balanced parallelizable algorithm. By using the algorithm, the search space of all m-sequences with lengths $2^n - 1$, for $18 \leq n \leq 20$ is successfully exhausted. Furthermore, a complete list of all PSL-optimal Legendre sequences for lengths up to 432100 is revealed. A conjecture is made, that all PSL-optimal Legendre sequences, with or without rotations, and with lengths N greater than 235723, are strictly greater than $\sqrt{N}$.

10. Some useful mathematical properties related to the flip operation of the skew-symmetric binary sequences are discovered, which could be exploited to significantly reduce the memory complexity of state-of-the-art stochastic Merit Factor (MF) optimization algorithms from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$, without degrading their time complexity. As a proof of concept, a lightweight algorithm was constructed, which could optimize pseudo-randomly generated skew-symmetric binary sequences with long lengths (up to $10^5 + 1$) to skew-symmetric binary sequences with a MF greater than 5. This contradicts the Bernasconi conjecture, that a stochastic search procedure will not yield MF higher than 5 for long binary sequences (sequences with lengths greater than 200).

11. A new class of finite binary sequences with even lengths with alternate autocorrelation absolute values equal to 1, called pseudo skew-symmetric class, is found. It is shown that the MF values of the new class are closely related to the MF values of adjacent classes of Golay's skew-symmetric sequences.

12. Sub-classes of sequences based on the partition number problem, as well as the notion of potentials, measured by helper ternary sequences, are proposed. Binary sequences with MF records for binary sequences with many lengths less than 225, and all lengths greater than 225, are revealed. Two extremely hard search spaces of lengths 573 and 1009 are also attacked. It was estimated that a state-of-the-art stochastic solver requires

respectively 32 and 46774481153 years to reach MF values of 6.34, while the required time from the proposed algorithm to reach such MF values is just several hours.

13. Using aperiodic autocorrelation functions for the S-box reverse engineering problem is proposed.

0.2 Publications related with the thesis

1. Dimitrov, Miroslav M. "On the design of chaos-based S-boxes." IEEE Access 8 (2020): 117173-117181, **IF:3.367, Q2**.

2. Dimitrov, Miroslav, Tsonka Baitcheva, and Nikolay Nikolov. "Efficient generation of low autocorrelation binary sequences." IEEE Signal Processing Letters 27 (2020): 341-345, **IF:3.109, Q2**.

3. Dimitrov, Miroslav, Tsonka Baitcheva, and Nikolay Nikolov. "On the generation of long binary sequences with record-breaking PSL values." IEEE Signal Processing Letters 27 (2020): 1904-1908, **IF:3.109, Q2**.

4. Dimitrov, Miroslav. "On the aperiodic autocorrelations of rotated binary sequences." IEEE Communications Letters 25.5 (2020): 1427-1430, **IF:3.436, Q2**.

5. Dimitrov, Miroslav, Tsonka Baicheva, and Nikolay Nikolov. "Hybrid Constructions of Binary Sequences With Low Autocorrelation Sideobes." IEEE Access 9 (2021): 112400-112410, **IF:3.476, Q2**.

6. Dimitrov, Miroslav M. "A Framework for Fine-Grained Nonlinearity Optimization of Boolean and Vectorial Boolean Functions." IEEE Access 9 (2021): 124910-124920, **IF:3.476, Q2**.

7. Iliev, M., Nikolov, N., Dimitrov, M. and Bedzhev, B. "Genetic algorithm for synthesis of binary signals with optimal autocorrelation." 2020 International Conference on Information Technologies (InfoTech). IEEE, 2020.

8. Dimitrov, Miroslav. "On the Skew-Symmetric Binary Sequences and the Merit Factor Problem." arXiv preprint arXiv:2106.03377 (2021).

9. Dimitrov, Miroslav. "New Classes of Binary Sequences with High Merit Factor." arXiv preprint arXiv:2206.12070 (2022).

0.3 Talks

1. Tsonka Baicheva and Miroslav Dimitrov, "Cryptanalysis on short messages encrypted with M-138 cipher machine", Eighth International Workshop on Optimal Codes and Related Topics, Sofia, 2017.

2. Miroslav Dimitrov, Tsonka Baicheva and Georgi Ivanov, "Implementation of RSA Attack Using 2-Dimensional Lattices by Constructing Hypotheses of Keys With Low Hamming Weight", CRYPTACUS, Cryptanalysis in Ubiquitous Computing Systems, Azores, Portugal, 2018.

0.4 Citations (last updated on 20.11.2022)

On the design of chaos-based S-boxes.

1. Ahmed, Fawad, Muneeb Ur Rehman, Jawad Ahmad, Muhammad Shahbaz Khan, Wadii Boulila, Gautam Srivastava, Jerry Chun-Wei Lin, and William J. Buchanan. "A DNA Based Colour Image Encryption Scheme Using A Convolutional Autoencoder." ACM Transactions on Multimedia Computing, Communications and Applications (2022).

2. Ali, Asim, Muhammad Asif Khan, Ramesh Kumar Ayyasamy, and Muhammad Wasif. "A novel systematic byte substitution method to design strong bijective substitution box (S-box) using piece-wise-linear chaotic map." PeerJ Computer Science 8 (2022): e940.

3. Arun, S. "Intelligent Learning Algorithm for Counterfeiting Side-Channel Attack Using Adaptive Chaotic S-Boxes for AES."SRM Institute of Science and Technology, Department of Computer Science Engineering, PhD Thesis, 2022.

4. Aslam, Mazzamal, Saira Beg, Adeel Anjum, Zakria Qadir, Shawal Khan, Saif Ur Rehman Malik, and MA Parvez Mahmud. "A strong construction of S-box using Mandelbrot set an image encryption scheme." PeerJ Computer Science 8 (2022): e892.

5. Freyre-Echevarría, Alejandro, Ismel Martínez-Díaz, Carlos Miguel Legón Pérez, Guillermo Sosa-Gómez, and Omar Rojas. "Evolving nonlinear S-boxes with improved theoretical resilience to power attacks." IEEE Access 8 (2020): 202728-202737.

6. Freyre-Echevarrıa, Alejandro. "On the Generation of Cryptographically Strong Substitution Boxes from Small Ones and Heuristic Search." In 10 th Workshop on Current Trends in Cryptology (CTCrypt 2021), p. 112.

7. Gonzalez, Francisco, Ricardo Soto, and Broderick Crawford. "Stochastic Fractal Search Algorithm Improved with Opposition-Based Learning for Solving the Substitution Box Design Problem." Mathematics 10, no. 13 (2022): 2172.

8. Grassi, Giuseppe. "Chaos in the Real World: Recent Applications to Communications, Computing, Distributed Sensing, Robotic Motion, Bio-Impedance Modelling and Encryption Systems." Symmetry 13, no. 11 (2021): 2151.

9. Hongjun Liu, Xingyuan Wang, Mengdi Zhao and Yujun Niu, "Constructing Strong S-Box by 2D Chaotic Map with Application to Irreversible Parallel Key Expansion", International Journal of Bifurcation and Chaos, Vol. 32, No. 11, 2250163 (2022)

10. Ibrahim, Saleh, and Alaa M. Abbas. "Efficient key-dependent dynamic S-boxes based on permutated elliptic curves." Information Sciences 558 (2021): 246-264.

11. Ibrahim, Saleh, and Alaa M. Abbas. "A novel optimization method for constructing cryptographically strong dynamic S-boxes." IEEE Access 8 (2020): 225004-225017.

12. Kang, Man, and Mingsheng Wang. "New Genetic Operators for Developing S-Boxes With Low Boomerang Uniformity." IEEE Access 10 (2022): 10898-10906.

13. Lambić, Dragan. "Comments on “On the Design of Chaos-Based S-Boxes”." IEEE Access 9 (2021): 49354-49354.

14. Lawnik, Marcin, Lazaros Moysis, and Christos Volos. "Chaos-Based Cryptography: Text Encryption Using Image Algorithms." Electronics 11.19 (2022): 3156.

15. Manzoor, Atif, Amjad Hussain Zahid, and Malik Tahir Hassan. "A New Dynamic Substitution Box for Data Security Using an Innovative Chaotic Map." IEEE Access 10 (2022): 74164-74174.

16. Musheer Ahmad;Reem Alkanhel;Walid El-Shafai;Abeer D. Algarni;Fathi E. Abd El-Samie;Naglaa F. Soliman, "Multi-Objective Evolution of Strong S-Boxes Using Non-dominated Sorting Genetic Algorithm-II and Chaos for Secure Telemedicine", IEEE Access 10 (2022): 112757 - 112775.

17. Prinetto, Paolo Ernesto, and Samuele Yves Cerini. "Empirical Evaluation of the Resilience of Novel S-Box Implementations Against Power Side-Channel Attacks.", Master's Degree Thesis.

18. Si, Yuanyuan, Hongjun Liu, and Yuehui Chen. "Constructing keyed strong S-Box using an enhanced quadratic map." International Journal of Bifurcation and Chaos 31, no. 10 (2021): 2150146.

19. Soto, Ricardo, Broderick Crawford, Francisco González Molina, and Rodrigo Olivares. "Human behaviour based optimization supported with self-organizing maps for solving the S-box design Problem." IEEE Access 9 (2021): 84605-84618.

20. Xu, Ying, Mengdi Zhao, and Hongjun Liu. "Design an irreversible key expansion algorithm based on 4D memristor chaotic system." The European Physical Journal Special Topics (2022): 1-9.

21. Zahid, Amjad Hussain, Musheer Ahmad, Ahmed Alkhayyat, Malik Tahir Hassan, Atif Manzoor, and Alaa Kadhim Farhan. "Efficient dynamic S-box generation using linear trigonometric transformation for security applications." IEEE Access 9 (2021): 98460-98475.

22. Zhou, Peizhao, Junxiao Du, Kai Zhou, and Shengfei Wei. "2D mixed pseudo-random coupling PS map lattice and its application in S-box generation." Nonlinear Dynamics 103, no. 1 (2021): 1151-1166.

Efficient generation of low autocorrelation binary sequences

23. Anjana, K., and Sayed Abdulhayan. "Analysis of Precoding with Kasami sequence for LTE MIMO." Journal of Pharmaceutical Negative Results (2022): 1228-1234.

24. Bošković, Borko, and Janez Brest. "Computational search of long skew-symmetric binary sequences with high merit factors." MENDEL. Vol. 28. No. 2, 2022.

25. Boukerma, Sabrina, Khaled Rouabah, SalahEddine Mezaache, and Salim Atia. "Efficient method for constructing optimized long binary spreading sequences." International Journal of Communication Systems 34, no. 4 (2021): e4719.

26. Brest, J., and Boskovic, B. (2021). Low Autocorrelation Binary Sequences: Best-Known Peak Sidelobe Level Values. IEEE Access, 9, 67713-67723.

27. Kuznetsov, Alexandr, Luca Romeo, Nikolay Poluyanenko, Sergey Kandiy, and Kateryna Kuznetsova. "Optimizing Hill Climbing Algorithm Parameters for Generation of Cryptographically Strong S-Boxes." (2022).
28. Miskiv, V. M. V., I. N. Prudyus, S. Ye Fabirovskyy, and M. Pashchuk Yu. "Optimal search for binary skew-symmetric sequences with minimal levels of side lobes." Mathematical Modeling and Computing 7, no. 2 (2020): 410-419.
29. Rosli, Siti Julia, Hasliza A. Rahim, Khairul Najmy Abdul Rani, Ruzelita Ngadiran, Wan Azani Mustafa, Muzammil Jusoh, Mohd Najib Mohd Yasin et al. "A Hybrid Modified Sine Cosine Algorithm Using Inverse Filtering and Clipping Methods for Low Autocorrelation Binary Sequences." CMC-Computers, Materials & Continua 71, no. 2 (2022): 3533-3556.
30. Sabrina Boukerma, "Etude et Optimisation des Codes Pseudo-Aleatoires pour les Applications a Acces Multiple", Doctorat Science thesis, 2021.
31. Vuk, Miha. "Algoritem za iskanje binarnih sekvenc z nizko avtokorelacijo." PhD diss., 2021.
32. Xiuping PENG, Hongxiao LI, Shide WANG, Hongbin LIN. Balanced optimal almost binary sequence pairs of period $N \equiv 1$ (mod4)[J]. Journal on Communications, 2021, 42(12): 163-171.
33. Zhou, Zheng, Yue Wu, Xiaofei Yang, and Yicong Zhou. "Neural Style Transfer With Adaptive Auto-Correlation Alignment Loss." IEEE Signal Processing Letters 29 (2022): 1027-1031.

On the generation of long binary sequences with record-breaking PSL values

34. Bošković, Borko, and Janez Brest. "Computational search of long skew-symmetric binary sequences with high merit factors." MENDEL. Vol. 28. No. 2, 2022.
35. Brest, Janez, and Borko Bošković. "Low Autocorrelation Binary Sequences: Best-Known Peak Sidelobe Level Values." IEEE Access 9 (2021): 67713-67723.
36. Brest, Janez, Borko Boškovic. "Neperiodicna binarna zaporedja z dobrimi avtokorelacijskimi lastnostmi: nizke vrednosti stranskih reznjev." Društvo Slovenska sekcija IEEE. 31-th International Electrotechnical and Computer Science Conference: 355-358.

On the aperiodic autocorrelations of rotated binary sequences

37. Bošković, Borko, and Janez Brest. "Computational search of long skew-symmetric binary sequences with high merit factors." MENDEL. Vol. 28. No. 2, 2022.

38. Brest, Janez, and Borko Bošković. "Low Autocorrelation Binary Sequences: Best-Known Peak Sidelobe Level Values." IEEE Access 9 (2021): 67713-67723.

39. Brest, Janez, Borko Boškovic. "Neperiodicna binarna zaporedja z dobrimi avtokorelacijskimi lastnostmi: nizke vrednosti stranskih reznjev." Društvo Slovenska sekcija IEEE. 31-th International Electrotechnical and Computer Science Conference: 355-358.

40. Chen, Zhixiong, Zhihua Niu, Yuqi Sang, and Chenhuang Wu. "Arithmetic autocorrelation of binary m-sequences." Cryptologia (2022): 1-10.

Genetic algorithm for synthesis of binary signals with optimal autocorrelation

41. Li, Ning, Mengdao Xing, Yaxin Hou, Shengwei Zhou, and Guang-Cai Sun. "Ship Focusing and Positioning Based on 2-D Ambiguity Resolving for Single-Channel SAR Mounted on High-Speed Maneuvering Platforms With Small Aperture." IEEE Transactions on Geoscience and Remote Sensing 60 (2022): 1-13.

Hybrid Constructions of Binary Sequences With Low Autocorrelation Sideobes

42. Bošković, Borko, and Janez Brest. "Computational search of long skew-symmetric binary sequences with high merit factors." MENDEL. Vol. 28. No. 2, 2022.

43. Brest, Janez, Borko Boškovic. "Neperiodicna binarna zaporedja z dobrimi avtokorelacijskimi lastnostmi: nizke vrednosti stranskih reznjev." Društvo Slovenska sekcija IEEE. 31-th International Electrotechnical and Computer Science Conference: 355-358.

On the skew-symmetric binary sequences and the merit factor problem

44. Bošković, Borko, and Janez Brest. "Computational search of long skew-symmetric binary sequences with high merit factors." MENDEL. Vol. 28. No. 2, 2022.

New Classes of Binary Sequences with High Merit Factor

45. Brest, Janez, Borko Boškovic. "Neperiodicna binarna zaporedja z dobrimi avtokorelacijskimi lastnostmi: nizke vrednosti stranskih reznjev." Društvo Slovenska sekcija IEEE. 31-th International Electrotechnical and Computer Science Conference: 355-358.

Chapter 1

Introduction

Boolean functions, vector Boolean functions, or S-boxes, and digital sequences are widely used in various practical fields such as telecommunications, radar technology, navigation, cryptography, measurement sciences, biology, or industry.

S-boxes are one of the most important primitives to be found in modern block ciphers. A weak S-box, in a cryptographic perspective, can be exploited by various attacks like linear cryptanalysis [17], differential cryptanalysis [18] , boomerang attack [147], algebraic attacks [34] or others like in [59]. Arguably, one of the most important properties of a given S-box is its nonlinearity. An S-box with high nonlinearity can be achieved by using the finite field inversion method [113]. However, such S-box is closely related to various algebraic structures. As a proactive countermeasure to future algebraic attacks, new ways of generation or optimization of pseudo-random S-boxes are proposed. Some examples of the aforementioned algorithms are published in [32], [85], [107], [108], and [145]. However, heuristically optimization of a given S-boxes could be a resource-consuming task.

Given their significance and importance, the design principles of an S-box construction, especially when implemented in a widely used and critical cryptosystem, should be publicly available and reproducible. However, in some cases, a given S-box generation method is not announced, or worse, misleadingly announced as a pseudo-randomly generated one. The reasons for obfuscating the design of a given S-box are manifold. For example, the initial S-boxes used in the Data Encryption Standard (DES) [55] were originally modified by NSA. The reasons for applying those modifications were not known. However, in [33], D. Coppersmith announces the motivation behind the S-box modifications. It appears that the agency knew about the existence of differential attacks about 20 years before the academic world.

Hiding a given S-box design could be related to some hidden construction, the knowledge of which could be exploited to gain a significant advantage in terms of hardware implementation. For example, as discovered in [21], the S-boxes used in the hash function Streebog and the 128-bit block cipher Kuznyechik, standardized by the Russian Federation, are designed with such a hidden structure. A user knowing this decomposition could implement the given S-box with a significantly smaller hardware footprint, allowing him to reach an up to 8 times faster S-box look-up.

A practical reason for hiding the design of a given S-box could be related to an encapsulated trapdoor as discussed in [128]. Even though the aforementioned trapdoor can be easily detected, as shown in [151], the motivation for finding other trapdoor S-box techniques should not be underestimated. Moreover, the designers of a given S-box could unintentionally create it with a flaw, which further rises the academic attention to the S-box reverse engineering problem.

Finding binary sequences whose aperiodic autocorrelation characteristics are collectively small according to some pre-defined criteria is a famous and well-studied problem. Two such measures are the Peak Sidelobe Level (PSL) and the Merit Factor (MF) value, which was first introduced by Golay in 1972 [60]. However, before Golay's definition, Littlewood [98] studied the norms of polynomials with ± 1 coefficients on the unit circle of the complex plane.

One of the desirable characteristics a given binary sequence should possess is a low peak sidelobe level. Some well-known constructions of such sequences includes the Barker codes [9], Rudin-Shapiro sequences [129][136], m-sequences [67], Gold codes [66], Kasami codes [84], Weil sequences [130], Legendre sequences [124]. Nevertheless, none of the aforementioned constructions guarantees that the generated binary sequence will possess the lowest possible (optimal) PSL value. Thus, currently, initiating an exhaustive search is the only way to reveal an optimal PSL value for binary sequences of some fixed length. The PSL-optimal values of binary sequences with lengths n greater than 84 are still unknown. This is not surprising, since the cardinality of the search space comprised of all binary sequences with some fixed length rises exponentially.

Golay's publications reveal a dedication to the merit factor problem for nearly twenty years (surveyed in [80]). Since then, a significant number of possible constructions of binary sequences with high merit factors were published. Near-optimal and optimal candidates are found by using heuristic search methods for longer lengths or a more direct approach, like the exhaustive search method, for smaller problem spaces. In [65], the merit factor problem was referenced by Golay as ***...challenging and charming***.

The problem of minimizing the merit factor is also known as the "low autocorrelated binary string problem", or the LABS problem. It has been well studied in theoretical physics and chemistry. For example, the LABS problem is correlated with the quantum models of magnetism. Bernasconi predicted that [14] ***... stochastic search procedures will not yield merit factors higher than about 5 for long sequences***. By long sequences, Bernasconi was referring to binary sequences with lengths greater than 200. Furthermore, in [41] the problem was described as ***... amongst the most difficult optimization problems***. Since the merit factor problem has resisted more than 50 years of theoretical attacks, a significant number of computational pieces of evidence were collected.

In this thesis, several design strategies for constructing and analyzing Boolean functions, S-boxes, and digital sequences are proposed. In Chapter 2 the preliminaries are provided. In Sections 2.1 and 2.2 some important definitions regarding Boolean and vector Boolean functions are given. Then, in Section 2.3 a rich collection of popular S-boxes is thoroughly analyzed. In general, the S-box construction methods could be divided into four categories as shown in Section 2.4. Then, S-boxes generated by using chaotic functions (CF) are analyzed to measure their actual resistance to linear cryptanalysis. The majority of the published papers using CFs emphasize the average nonlinearity of the S-box coordinates only, ignoring the rest of the S-box components in the process. Thus, integrating such S-boxes in a given cryptosystem should be done with considerable caution. Furthermore, it appears that in the context of the nonlinearity optimization problem the profit of using chaos structures is negligible. During our experiments, by using two heuristic methods and starting from pseudo-random S-boxes, we repeatedly reached S-boxes, which significantly outperform all previously published CF-based S-boxes, in those cryptographic terms, which the aforementioned papers utilize for comparison. Then, in Section 2.5, we project the S-box nonlinearity optimization problem to a satisfiability problem, which could be solved by using SAT solvers. This is achieved by introducing some new definitions like couplings, coordinate decomposition, degree of descendibility, S-box coordinate extended linear approximation table (CELAT), as well as some useful properties and inner connections. The SAT projection revealed that we could successfully construct bijective 8×8 S-boxes from 8 Boolean functions as components, each of which possesses the maximum nonlinearity value of 116. The provided toolbox could serve in cases, where the designer's goal is to increase (or intentionally decrease) the nonlinearity of a given S-box by applying as minimum changes as possible. For example, we demonstrate how the Skipjack S-box, developed by the U.S. National Security Agency (NSA), and the Kuznyechik S-box, developed by the Russian Federation's standardization agency, could be optimized to a higher nonlinearity by tweaking just 4 and 12 bits, respectively (out of 2048).

In Chapter 3, a strategy of analyzing various spectra channels to detect hidden patterns and anomalies in popular S-boxes is discussed. It could serve as a more fine-grained extension to the methods discussed in [119]. More specifically, by applying spectral analysis on various

S-box characteristics, as a linear approximation, difference distribution, and auto-correlation tables, we can detect visual symmetries or anomalies, which could not only serve as proof that the S-box was not generated pseudo-randomly but additionally provides some further information about the inner structure of the S-box, making the complete reverse-engineering of the hidden construction possible [1].

Chapter 4 addresses the PSL optimization problem. In Section 4.1, a simple and efficient algorithm based on a heuristic search by shotgun hill climbing to construct binary sequences with small peak sidelobe levels is suggested. The algorithm is applied for the generation of binary sequences of lengths between 106 and 300. Improvements are obtained in almost half of the considered lengths while for the rest of the lengths, binary sequences with the same PSL values as reported in the state-of-the-art publications are found. Then, in Section 4.2, a method to generate long binary sequences with low PSL value is proposed. Both the time and memory complexities of the proposed algorithm are reduced to $\mathcal{O}(n)$. During our experiments, we repeatedly reach better PSL values than the currently known state of art constructions, such as Legendre sequences, with or without rotations, Rudin-Shapiro sequences or m-sequences, with or without rotations, by always reaching record-breaking PSL values strictly less than $\sqrt{n}$. Furthermore, the efficiency and simplicity of the proposed method are particularly beneficial to the lightweightness of the implementation, which allowed us to reach record-breaking PSL values for less than a second. In Section 4.3 we continue our research with the exploration of hybrid algorithms for achieving binary sequences with arbitrary lengths and high PSL values. By combining some of our previous works, together with some mathematical insights, a few hybrid heuristic algorithms were created. During our experiments, and by using the aforementioned algorithms, we were able to find PSL-optimal binary sequences for all those lengths, which were previously found during exhaustive searches by various papers. Then, by using a general-purpose computer, we further demonstrate the effectiveness of the proposed algorithms by revealing binary sequences with lengths between 106 and 300, the majority of which possess record-breaking PSL values. Then, by using some well-known algebraic constructions, we outline a few strategies for finding highly-competitive binary sequences, which could be efficiently optimized, in terms of PSL, by the proposed algorithms. Finally, in Section 4.3.3, a well-known computational problem is finding the lowest possible PSL among the set of a binary sequence B, and all binary sequences generated by rotations of B is discussed. Some useful properties of rotated binary sequences are discovered, which allowed us to project the aforementioned problem to a perfectly balanced parallelizable algorithm. The proposed algorithm, altogether with its graphics processing unit (GPU) implementation,

[1] Although the demonstrated anomalies are visible on paper, reading the electronic version is greatly encouraged.

is significantly faster than the existing instruments. We were able to exhaust the search space of all m-sequences with lengths $2^n - 1$, for $18 \leq n \leq 20$, and to reveal a complete list of all PSL-optimal Legendre sequences, with or without rotations, for lengths up to 432100 - out of computational reach until now. The numerical experiments suggest that the PSL value of all PSL-optimal Legendre sequences, with or without rotations, and with lengths N greater than 235723, are strictly greater than $\sqrt{N}$.

Chapter 5 deals with the Merit Factor (MF) problem. It was conjectured that stochastic search procedures will not yield merit factors higher than 5 for long binary sequences (sequences with lengths greater than 200). Some useful mathematical properties related to the flip operation of the skew-symmetric binary sequences are presented in Section 5.1. By exploiting those properties, the memory complexity of state-of-the-art stochastic MF optimization algorithms could be reduced from $O(n^2)$ to $O(n)$. As a proof of concept, a lightweight stochastic algorithm was constructed, which can optimize pseudo-randomly generated skew-symmetric binary sequences with long lengths (up to $10^5 + 1$) to skew-symmetric binary sequences with a merit factor greater than 5. An approximation of the required time is also provided. The numerical experiments suggest that the algorithm is universal and could be applied to skew-symmetric binary sequences with arbitrary lengths.

Golay introduced one beneficial class of sequences, called skew-symmetric sequences; finite binary sequences with odd lengths with alternate autocorrelation values equal to 0. Their special construction greatly reduces the computational efforts of finding binary sequences with odd lengths and high MF. Having this in mind, the majority of papers to be found in the literature are focused solely on this class, preferring them over binary sequences with even lengths. In Section 5.1.2, a new class of finite binary sequences with even lengths with alternate autocorrelation values equal to ± 1 is presented (see also [46]). We show that the MF values of the new class are closely related to the MF values of adjacent classes of skew-symmetric sequences. We further introduce new sub-classes of sequences using the partition number problem and the notion of potentials, measured by helper ternary sequences. Throughout our experiments, MF records for binary sequences with many lengths less than 225, and all lengths greater than 225, are discovered. Binary sequences of all lengths, odd or even, less than 2^8 and with MF > 8, and all lengths, odd or even, less than 2^9 and with MF > 7, are now revealed. We demonstrate the efficiency of the proposed algorithm by launching it on two extremely hard search spaces of binary sequences of lengths 573 and 1009. The choice of those two specific lengths is motivated by the approximation numbers given in [24], Figure 7, presented during a discussion of how much time the state-of-the-art stochastic solver lssOrel_8 will need to reach binary sequences with the aforementioned lengths and merit factors close to 6.34. It was estimated that finding solutions with a merit factor of 6.34 for a binary sequence with length 573 requires around 32 years, while for binary sequences with length 1009, the average runtime prediction to reach the merit factor of 6.34 was 46774481153 years. By using the proposed in Section 5.1.2 algorithm, we were able to reach such binary sequences within several hours. Finally, in Section 5.2, a method addressing the S-box reverse engineering problem using spectrography on aperiodic autocorrelation functions is presented.

Chapter 2

Vector Boolean Functions and Cryptography

2.1 Boolean Functions

Definition 2.1.1 (Boolean Function & Truth Tables)**.** Let us define the set $B = \{0,1\}$. A Boolean function $f(x)$ of n variables $x_1,...,x_n$ is a mapping $f : B^n \mapsto B$ from n binary inputs $x = (x_1,x_2,\cdots,x_n) \in B^n$ to one binary output $y = f(x) \in B$. The binary truth table (BTT) of an n-variable Boolean function $f(x)$ is the vector of all the consecutive outputs of the Boolean function:

$$[f(x)] = [f(0,0,\cdots,0), f(0,0,\cdots,1), \cdots, f(1,1,\cdots,1)]$$

The polarity truth table (PTT) of an n-variable Boolean function $f(x)$ is derived from the binary truth table. We define the PTT by $[\hat{f}(x)] = [1-2f(x)]$. By the definition of the polarity truth table, follows:

$$f(x) = 0 \Leftrightarrow \hat{f}(x) = 1; f(x) = 1 \Leftrightarrow \hat{f}(x) = -1$$

Definition 2.1.2 (Algebraic Normal Form)**.** The algebraic normal form of an n-variable Boolean function $f(x)$, denoted by ANF_f, is given by the following equation: $ANF_f = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n \oplus a_{1,2}x_1x_2 \oplus \cdots \oplus a_{1,2,\cdots,n}x_1x_2\cdots x_n$, where the coefficients a belongs to B.

Definition 2.1.3 (Algebraic Degree)**.** The algebraic degree of a Boolean function $f(x)$, denoted by $deg(f)$, is equal to the number of variables in the longest item of its ANF_f.

Definition 2.1.4 (Hamming Distance)**.** The Hamming distance between two n-variable Boolean functions $f(x)$ and $g(x)$, denoted by $d_H(f,g)$, represents the number of differing elements in the corresponding positions of their truth tables.

Definition 2.1.5 (Linear Boolean Function)**.** Any n-variable Boolean function of the form:

$$l_w(x) =< w,x >= w_1x_1 \oplus w_2x_2 \oplus \cdots \oplus w_nx_n,$$

where $w,x \in B^n$, is called a linear function.

Definition 2.1.6 (Affine Boolean Function)**.** Any n-variable Boolean function of the form:

$$l_w(x) =< w,x >= w_0 \oplus w_1x_1 \oplus w_2x_2 \oplus \cdots \oplus w_nx_n,$$

where $w_0 \in B$ and $w,x \in B^n$, is called an affine function.

Definition 2.1.7 (Walsh-Hadamard Transform)**.** For an n-variable Boolean function $f(x)$, represented by its polarity table $[\hat{f}(x)]$, the Walsh-Hadamard transform, or WHT, $\hat{F}_f : B^n \to Z$, is defined by:

$$\hat{F}_f(w) = \sum_{x \in B^n} \hat{f}(x)(-1)^{<w,x>}$$

Definition 2.1.8 (Absolute Indicator)**.** For an n-variable Boolean function $f(x)$, we denote the absolute indicator of f as Δ_f. For all $u \in F_2^n$, except the zero vector, write

$$\Delta_f(u) = \sum_x (-1)^{f(x)+f(x+u)}$$

The absolute indicator of f is calculated by

$$\Delta_f = \max_u |\ \Delta_f(u)\ | \tag{2.1}$$

2.2 Vector Boolean Functions (S-boxes)

Definition 2.2.1 (Vectorial Boolean Function – Substitution Table – S-box)**.** An n-binary input to m-binary output mapping $S : B^n \Leftrightarrow B^m$, which assigns some $y = (y_1, y_2, \cdots, y_m) \in B^m$ by $S(x) = y$ to each $x = (x_1, x_2, \cdots, x_n) \in B^n$, is called an (n,m) substitution table (S-box) and is denoted by $S(n,m)$.

Definition 2.2.2 (Bijective S-box)**.** An S-box $S(n,m)$ is said to be bijective, if it maps each input $x \in B^n$ to a distinct output $y = S(x) \in B^m$ and all possible 2^m outputs are present.

Definition 2.2.3 (S-box Look-up Table – LUT)**.** The look-up table LUT of an S-box $S(n,m)$ is an (2^n x m) binary matrix S, which rows consist of all outputs of $S(n,m)$, corresponding to all possible 2^n inputs ordered lexicographically. Since the mapping defined by $S(n,m)$ consists of m Boolean functions $f_1, f_2, \cdots, f_m$, we could write down S_{LUT} as follows:

$$S_{LUT} = \begin{bmatrix} f_1(0,0,\cdots,0) & f_2(0,0,\cdots,0) & \cdots & f_m(0,0,\cdots,0) \\ f_1(0,0,\cdots,1) & f_2(0,0,\cdots,1) & \cdots & f_m(0,0,\cdots,1) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(1,1,\cdots,0) & f_2(1,1,\cdots,0) & \cdots & f_m(1,1,\cdots,0) \\ f_1(1,1,\cdots,1) & f_2(1,1,\cdots,1) & \cdots & f_m(1,1,\cdots,1) \end{bmatrix} \tag{2.2}$$

Definition 2.2.4 (S-box Coordinates)**.** We define each column of the $S(n,m)$ LUT as a coordinate of $S(n,m)$. Each column represents the truth table of some Boolean function f_i. If $S(n,m)$ is bijective vectorial Boolean function it follows that $n = m$ and we have exactly n coordinates.

Definition 2.2.5 (Polarity Look-up Table – PLUT)**.** The polarity look-up table PLUT of an S-box $S(n,m)$, denoted by S_{PLUT}, is an $(2^n, m)$ matrix with elements in $\{-1,1\}$, where each element on row j and column k, denoted by $S_{PLUT}[j][k]$, for $j = 1,2,\cdots,2^n$ and $k = 1,2,\cdots,m$, is derived from $S_{LUT}[j][k]$ by

$$S_{PLUT}[j][k] = (-1)^{S_{LUT}[j][k]} = 1 - 2S_{LUT}[j][k]$$

Since the mapping defined by $S(n,m)$ consists of m Boolean functions $f_1, f_2, \cdots, f_m$, we could write down S_{PLUT} as follows:

$$S_{PLUT} = \begin{bmatrix} \hat{f}_1(0,0,\cdots,0) & \hat{f}_2(0,0,\cdots,0) & \cdots & \hat{f}_m(0,0,\cdots,0) \\ \hat{f}_1(0,0,\cdots,1) & \hat{f}_2(0,0,\cdots,1) & \cdots & \hat{f}_m(0,0,\cdots,1) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{f}_1(1,1,\cdots,0) & \hat{f}_2(1,1,\cdots,0) & \cdots & \hat{f}_m(1,1,\cdots,0) \\ \hat{f}_1(1,1,\cdots,1) & \hat{f}_2(1,1,\cdots,1) & \cdots & \hat{f}_m(1,1,\cdots,1) \end{bmatrix}, \tag{2.3}$$

where $\hat{f}_i(\alpha) = (-1)^{f_i(\alpha)} = 1 - 2f_i(\alpha)$.

Definition 2.2.6 (S-box Extended WHT Spectrum Matrix – EWHTSM)**.** The extended Walsh-Hadamard transform spectrum matrix (EWHTSM) of an S-box $S(n,m)$ is a $(2^n, 2^m)$ matrix $\hat{F}_{ExtS}$, which columns are represented by the Walsh-Hadamard transform spectra $[\hat{F}_{g_v}(w)]$ of the Boolean functions $g_v(x) = v_1 f_1(x) \oplus v_2 f_2(x) \oplus \cdots \oplus v_m f_m(x)$, where w and v are arranged lexicographically respectively in B^n and B^m.

$$\hat{F}_{ExtS} = \begin{bmatrix} \hat{F}_{g_0}(0,0,\cdots,0) & \cdots & \hat{F}_{g_{2^m-1}}(0,0,\cdots,0) \\ \hat{F}_{g_0}(0,0,\cdots,1) & \cdots & \hat{F}_{g_{2^m-1}}(0,0,\cdots,1) \\ \vdots & \ddots & \vdots \\ \hat{F}_{g_0}(1,1,\cdots,0) & \cdots & \hat{F}_{g_{2^m-1}}(1,1,\cdots,0) \\ \hat{F}_{g_0}(1,1,\cdots,1) & \cdots & \hat{F}_{g_{2^m-1}}(1,1,\cdots,1) \end{bmatrix} \tag{2.4}$$

The importance of the S-box extended Walsh-Hadamard transform matrix is to quantitatively describe the distance with a special measure, alike the Hamming distance, between each linear combination of coordinates in the given S-box and each possible linear function.

Definition 2.2.7 (Linear Approximation Table – LAT)**.** The linear approximation table of an S-box $S(n,m)$, denoted by LAT_S or S_{LAT}, is a table with 2^n rows and 2^m columns, which entries are given by:

$$S_{LAT}[X][Y] = LAT_S[X][Y] = 2^{n-1} - d_H(X,Y), \tag{2.5}$$

where Y is a consequent linear combination of coordinates of the current S-box and X is the consequent linear function with length n.

Definition 2.2.8 (S-box Nonlinearity)**.** The nonlinearity of an S-box $S(n,m)$, denoted by S_{NL}, is defined as:

$$S_{NL} = 2^{n-1} - \max(\{\mid w_i \mid\}), \tag{2.6}$$

where $\{\mid w_i \mid\}$ is the set of all absolute values of elements in LAT, except the uppermost left one.

Definition 2.2.9 (S-box ACNV)**.** The average coordinate nonlinearity value, or S_{ACNV}, of a given S-box S, is the average value of all nonlinearities of coordinates of S.

Definition 2.2.10 (S-box Decimal Look-up Table – DLUT)**.** Each S-box is uniquely defined by its LUT. Translating each row of the LUT as a decimal number uniquely defines the same S-box as a decimal look-up table (DLUT).

A bijective S-box $S(n,n)$ has exactly n coordinates. Each coordinate is defined by its truth table, which consists of 2^n elements from B. So, we have a total of $n2^n$ elements from B which uniquely define S. Following this observation, we have a total of 2^{n2^n} possible choices of S-boxes. But not all of them are bijective. To further restrict our choices to bijective

Table 2.1 DLUT example of a randomly-generated bijective $(3,3)$ S-box.

Input bits	**Output bits**	**Decimal**
000	001	1
001	110	6
010	010	2
011	111	7
100	101	5
101	011	3
110	100	4
111	000	0

S-boxes only, we need to restrict the set of possible S-boxes with the observation that all elements in the DLUT of S should be distinct. This reduces the possible choices of S-boxes from 2^{n2^n} to $2^n!$.

Definition 2.2.11 (XOR Table)**.** The XOR table of an S-box $S(n,m)$ is a $(2^n \times 2^m)$ binary matrix S_{XORT}, which columns consist of all linear combinations of S_{LUT} columns ordered lexicographically.

Definition 2.2.12 (S-box Minimal Algebraic Degree)**.** The minimal algebraic degree of an S-box $S(n,m)$ is the minimum algebraic degree among all component functions of S.

$$\begin{aligned} S_{DEG} &= min_{(v \in B^m)} deg(g_v) = \\ &= min_{((v_1,v_2,\cdots,v_m) \in B^m)} deg(v_1 f_1(x) \oplus v_2 f_2(x) \oplus \cdots \oplus v_m f_m(x)), \end{aligned} \tag{2.7}$$

where $f_1, f_2, \cdots, f_m$ are the coordinate Boolean functions of $S(n,m)$.

Definition 2.2.13 (S-box Absolute Indicator)**.** The absolute indicator of a given S-box S, denoted as S_{AC}, is equal to the maximal absolute indicator among all absolute indicators of component functions of S.

Definition 2.2.14 (S-box Differential Uniformity)**.** Differential uniformity, or δ-uniformity of a given S-box $S(n,m)$, denoted by S_δ, is defined by:

$$S_\delta = \max_{\alpha \in B^n \setminus \{0\}} \max_{\beta \in B^m} \left| \{ x \in B^n \mid S(x) \oplus S(x \oplus \alpha) = \beta) \} \right|$$

2.3 Cryptographic Properties of Some Popular S-boxes

The cryptographic properties of vector boolean functions are thoroughly examined by introducing a rich list of desirable parameters an S-box should have to guarantee an acceptable resistance to sophisticated cryptographic attacks such as the linear cryptanalysis [103][17], the differential cryptanalysis [18], boomerang attack [147] or interpolation attack [79]. S-boxes are widely used in modern cryptographic algorithms like AES [40], Whirlpool [11], Camellia [7] and many others (see Table A.1 in the Appendix). For a given S-box S the goal of the designer is to achieve high values of S_{NL} and S_{DEG}, as well as small values of S_{δ} and S_{AC}.

The S-boxes, created with the Finite Field Inversion method [114], as the Rijndael S-box used in AES [40], have the best currently known cryptographic properties among all 8×8 S-boxes. However, some concerns about constructing S-boxes by using a purely algebraic approach can make them vulnerable to algebraic attacks [34]. Hence, in some applications, randomly or heuristically generated S-boxes are used. In table A.1 a collection of well-known and published S-boxes used in popular cryptographic algorithms are analyzed, and one can see that only 11 S-boxes, out of 47, are AES-alike. For a more detailed picture, the LAT Spectras of the S-boxes is also provided, i.e. the real-valued vector of all absolute values of LAT coefficients. The distribution of the S_{LAT} coefficients of a given S-box S could also provide some more insights into how S is constructed when the construction method is not announced (intentionally or not) by the designers of S.

2.4 Design Strategies for Constructing S-boxes

The rich variety of proposed S-boxes constructions can be classified into four categories. The first category $\boldsymbol{T_1}$ for finding S-boxes with good cryptographic properties uses the pseudo-random generation method. The highest reported nonlinearity (NL) of an $(8,8)$ S-box generated by this approach is 100 [110]. Table 2.2 presents statistics of our experiments about pseudo-randomly generated S-boxes. We generated over one billion S-boxes $(1,387,914,282)$ and, for example, find that the probability to randomly construct an $(8,8)$ S-box with NL 100 is $2^{-25.978}$. Thus, the probability to find an S-box of NL 100, or higher, at random is rather small.

The second category $\boldsymbol{T_2}$ uses a more straightforward (deterministic) approach, like an algebraic constructions like finite field inversion method, cellular automata based methods [16], quasi-cyclic codes methods [25][19], affine-power-affine methods [38] or using some other deterministic approach as Feistel and Misty constructions [29].

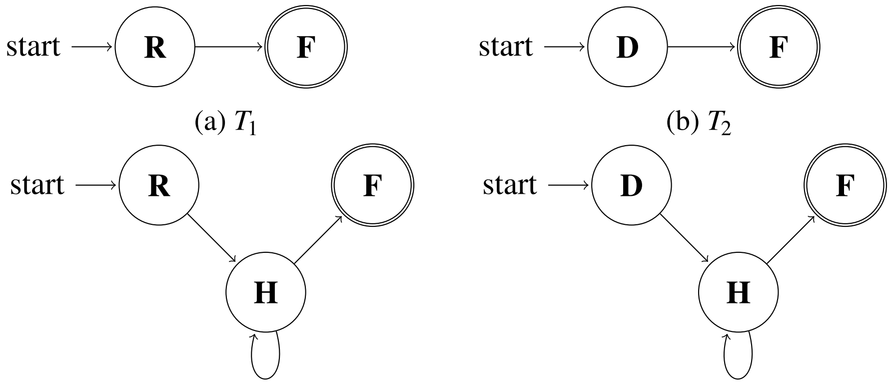
Table 2.2 Statistics for $(8,8)$ Sboxes generated by using T_1

Nonlinearity	Found	Approx. probability
66	1	$2^{-30.370}$
68	7	$2^{-27.563}$
70	35	$2^{-25.241}$
72	252	$2^{-22.393}$
74	1467	$2^{-19.852}$
76	8372	$2^{-17.339}$
78	44954	$2^{-14.914}$
80	223694	$2^{-12.599}$
82	1032177	$2^{-10.393}$
84	4412551	$2^{-8.297}$
86	17459934	$2^{-6.313}$
88	62726236	$2^{-4.468}$
90	192298910	$2^{-2.851}$
92	430567292	$2^{-1.689}$
94	515198571	$2^{-1.430}$
96	161572964	$2^{-3.103}$
98	2366844	$2^{-9.196}$
100	21	$2^{-25.978}$
102	0	NA

The third category $\boldsymbol{T_3}$ is about applying heuristic search methods to optimize pseudo-randomly generated S-boxes. Members of this category are methods like hill climbing [107], simulated annealing [32], genetic algorithms [108], special genetic algorithms combined with total tree searching [145], special immune algorithms [78], and others [142][121].

The fourth category $\boldsymbol{T_4}$ is using hybrid search, i.e starting from an S-box generated by some T_2 construction, and then obtaining a new one by using some T_3 algorithm. Such methods are suggested in [85][31][76][101][42][77][4]. It should be noted that categories T_3 and T_4 looks similar. However, the comparison between T_3 and T_4 methods is not entirely fair, since the authors of the latest do not start from a pseudo-random state. Instead, they initialize their algorithm with some highly competitive candidate. The same observation is made in [121], p.9.

The logic flow of the aforementioned categories is summarized in Figure 2.1. **R** denotes some pseudo-random generated bijective S-box, **H** is a notation for some heuristic algorithm, **D** is a notation for some deterministic construction, while **F** is the final state.



(a) T_1 (b) T_2

(c) T_3 (d) T_4

Fig. 2.1 Automata representation of S-box generation categories.

We should also address the S-box chaos-based constructions methods. They could belong to either of categories T_2, T_3 or T_4. However, in [50], S-boxes generated by using chaotic functions (CF) are analyzed to measure their actual resistance to linear cryptanalysis. It appears that most of the aforementioned papers emphasize the average nonlinearity of the S-box coordinates (ACNV) only, ignoring the rest of the S-box components in the process. Having this in mind, the majority of those studies should be re-evaluated. Integrating such S-boxes in a given cryptosystem should be done with considerable caution. Furthermore, we show that in the context of the nonlinearity optimization problem the profit of using chaos structures appears to be negligible. By using two heuristic methods and starting from pseudo-random S-boxes, we repeatedly reached S-boxes, that significantly outperform all previously published CF-based S-boxes, in those cryptographic terms, that the aforementioned papers utilize for comparison. Moreover, we have linked the multi-armed bandit problem to the problem of maximizing an S-box average coordinate nonlinearity value, which further allowed us to reach near-optimal average coordinate nonlinearity values significantly greater than those known in the literature.

The methods involved in CF S-box constructions are manifold (see the comparison table provided in [50]). The actual nonlinearity of an S-box is calculated by the minimum nonlinearity of all the components of the S-box. For example, let us take an arbitrary S-box $F(5,5)$ with $F_{LUT} = [f_0, f_1, f_2, f_3, f_4]$. Each column of F_{LAT} is determined by some linear combination of coordinates of F, sorted lexicographically, from left to right, by the binary representation of the column index, zero-filled to 5. Let $F_{LAT}[i]$ denotes the i-th column

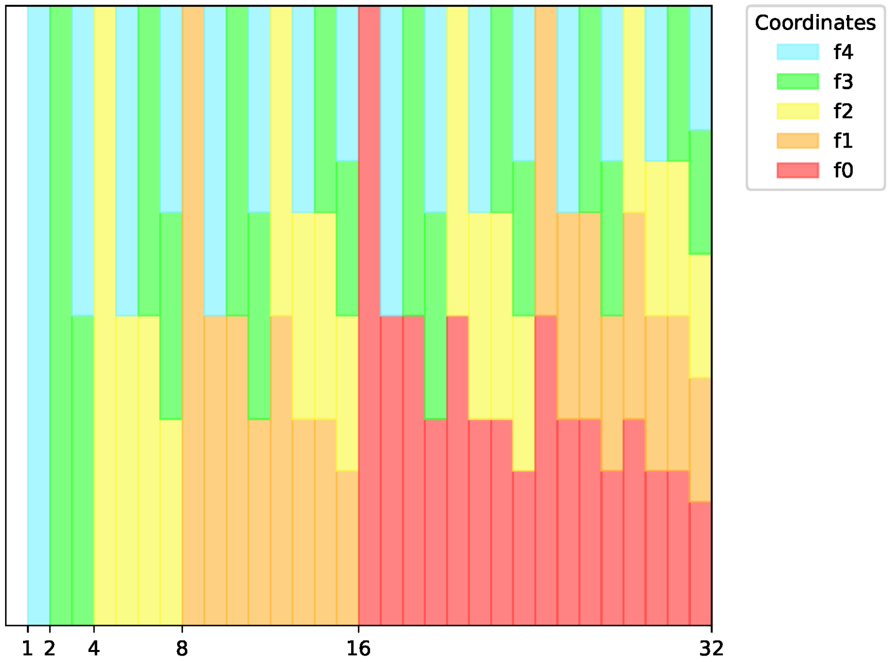


Fig. 2.2 Coordinate decomposition of a $(5,5)$ S-box LAT

of F_{LAT}. Then, for example, the $F_{LAT}[11]$ column holds the nonlinear characteristics of the Boolean function $f_1 \oplus f_3 \oplus f_4$, while $F_{LAT}[4]$ holds the nonlinear characteristics of the Boolean function f_3. In Figure 2.2 the coordinate decomposition of F_{LAT} is visualized. Each coordinate is associated with a distinct color. The number of segments in each column corresponds to the number of terms in the respective linear combination of coordinates. Since $F_{LAT}[0]$ is the trivial linear combination (all coefficients are equal to zero), we leave the first column of Figure 2.2 colorless. For technical reasons and better illustration, the coordinate decomposition example is based on a $(5,5)$ S-box. However, it applies to S-boxes of any dimension.

As defined in Definition 2.2.8, we seek the maximum absolute value v of all the elements in S-box $S(n,n)$ LAT, to find the nonlinearity of S, i.e. $S_{NL} = 2^{n-1} - v$. In the context of block ciphers, a low nonlinearity S-box value is associated with the cipher linear cryptanalysis resistance [103][17][74]. As shown in [50], the average value of the nonlinearities of the coordinates of a given S-box S doesn't correspond to the actual nonlinearity of S. However, from the designer's perspective, when a higher value of ACNV is desirable, a simple heuristic construction could be used instead.

In general, if we want to improve the nonlinearity of a given bijective S-box $S(n,n)$, a strategy of lowering the absolute value of coefficients in S_{LAT} makes sense. Moreover, the elements of each column of S_{LAT} are entangled by Parceval's theorem [104]. Let's denote as

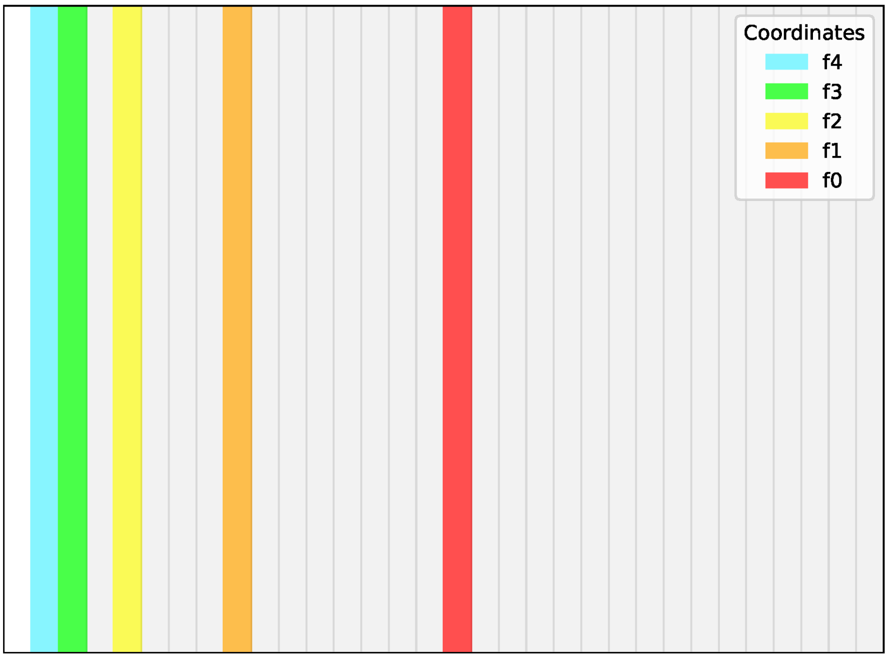


Fig. 2.3 Columns of interest of a $(5,5)$ S-box LAT

C_i the array composed of the elements of $S_{LAT}[i]$. Since we want to lower the nonlinearities of coordinates of S only, an evaluating function $E(S)$ is created, s.t. $E(S) = \sum_{p=0}^{n-1} \sum_{x \in C_{2^p}} |x|^M$, where M denotes a magnitude of our choice. The restriction $x \in C_{2^p}$ narrows down the set of possible columns of S_{LAT} to be optimized, in terms of nonlinearity, to the set of coordinates of S. As an example, in the case of a $S(5,5)$ S-box, the evaluation function threats as significant the elements inside the colored columns of S_{LAT} illustrated in Figure 2.3.

By using stochastic[1] hill climbing as a heuristic function, starting from arbitrary pseudo-random S-box construction and by using $E(S)$, algorithm 1 is proposed.

[1] hill climbing without neighborhood search

Algorithm 1 An algorithm for an S-box ACNV optimization

$s \leftarrow R(n)$ ▷ the function R(n) generates pseudo-random bijective S-box $S(n,n)$
repeat
    $sdupl \leftarrow s$
    RT(sdupl) ▷ the function RT(S) make a random transposition in S
    if $\mathrm{E}(sdupl) < \mathrm{E}(s)$ **then**
        $s \leftarrow sdupl$
    end if
until STOP condition is reached ▷ reaching $\frac{n(n-1)}{4}$ cycles

Given an S-box $S(n,n)$, and by using just one transposition, we can reach a total of $\binom{n}{2}$ S-boxes. Let denote this set as S^T. We further define a set S^I, s.t. $W \in S^I \iff W \in S^T \wedge E(W) < E(S)$. In case $|S^I| = 1$, and we are allowed to randomly pick $\frac{|S^T|}{2}$ elements from S^T, the probability some of the picked elements to belong to S^I is $\frac{1}{2}$. The threshold value of the stop condition in Algorithm 1 is constructed on this observation.

By using a magnitude of 10, we repeatedly generated S-boxes with high coordinate nonlinearities. During our experiments, we tried various magnitude values. However, larger or smaller values of the magnitude are respectively too aggressive or too tolerant to the largest elements of the S-box LAT. In Figure 2.4 the DLUT, in a hexadecimal format, of an optimized S-box $S_c(8,8)$ is presented. The first row and column of the table correspond respectively to the first and second half of the input in hexadecimal format. For example, the input **11110101**, equal to **f5**, is transformed by S_c to **5d**.

By using Algorithm 1 we could repeatedly optimize pseudo-randomly generated S-boxes to ACNV of 114.0, the highest reported in the literature. Moreover, by exploiting the techniques discussed in the multi-armed bandit problem [15], we were able to reach ACNV of 114.5 (see [50]). Algorithm 1 was implemented with the built-in tools provided by the open-source mathematical software system SageMath [43].

2.5 Nonlinearity Optimization Using SAT Solvers

In this section, an interconnection between the S-box nonlinearity optimization problem and binary integer programming is shown. A lightweight optimization routine is proposed, which does not cause any significant computational burden. Moreover, the toolbox could be utilized as proof of infeasibility.

A major drawback of the state-of-the-art heuristic techniques is their aggressiveness on the initial S-box. Hence, in most cases, it is difficult to link the resulting S-box with the

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	ab	f0	5e	3f	fa	e2	6f	8e	3c	36	30	db	29	73	da	45
10	87	f9	60	3b	bf	a4	c7	0c	a9	c0	f3	cb	68	ff	ee	a6
20	90	57	f2	77	ef	c2	78	b7	94	32	e6	4d	53	6d	26	98
30	c1	2c	2a	9a	12	2b	ea	e8	17	7c	5c	6e	50	d9	f6	88
40	83	69	5a	67	af	b9	1a	b8	8a	d4	b4	a0	cc	e1	24	c6
50	be	1f	a1	51	9f	64	4e	4f	2f	85	6b	76	86	35	4b	ed
60	81	84	39	13	62	c3	9e	dc	d0	66	5f	44	de	1c	bd	34
70	1d	1e	2d	6c	a2	46	97	c5	37	61	a3	56	fe	f7	d5	38
80	ce	05	09	18	aa	fc	91	28	9b	10	e9	0b	71	dd	e7	23
90	7f	72	59	6a	43	fd	d1	e4	f8	0d	55	74	c8	f5	27	65
a0	93	c4	19	49	00	20	3d	2e	a8	d3	01	7d	25	0e	f4	33
b0	02	04	0a	14	16	ae	31	11	cf	79	8f	d8	8b	d7	ca	b3
c0	bb	3e	0f	92	df	40	4c	cd	ac	22	5b	a5	bc	f1	75	89
d0	96	b1	e3	d2	7a	1b	70	58	03	47	80	9c	06	ba	c9	54
e0	ad	41	99	48	7e	3a	95	e0	ec	07	63	7b	b2	21	b0	4a
f0	8d	d6	15	fb	9d	5d	8c	42	08	b6	eb	a7	b5	e5	52	82

Fig. 2.4 An optimized S-box $S_c(8,8)$ using Algorithm 1, having ACNV of 114.0

initial S-box. It is difficult to prove that such a link exists in the first place. The fine-grained optimization routine proposed in [51] allows us to optimize the nonlinearity value of a given S-box with as minimum changes as possible. From the designer's perspective, this property is particularly beneficial, since we could focus the optimization routine on the weak components of a given S-box, without degrading the remaining ones. The effectiveness of the proposed algorithm is further demonstrated by increasing the nonlinearity of the Skipjack S-box, developed by NSA, and Kuznyechik S-box, developed by the Russian Federation's standardization agency, by tweaking respectively 4 and 12 (out of 2048) bits only.

The currently known maximum nonlinearity value for 8-variable balanced Boolean functions is 116 [122]. Furthermore, as shown in [133], the nonlinearity value of 8-variable balanced Boolean functions is upper bounded by 120, which means that the maximum theoretical ACNV of (8,8) bijective S-boxes is less or equal to 118.0. If a bijective S-box with ACNV greater than 116.0 is found, at least one of its eight coordinates will possess a nonlinearity value of 118, which will finally answer the long-standing problem of the maximum possible nonlinearity value for 8-variable balanced Boolean functions. However, there is academic skepticism that 8-variable balanced Boolean functions with nonlinearity value 118 exist. Having this in mind, one open question to be answered is: *Does bijective (8,8) S-box with an ACNV value of 116 exist*? By using the SAT solving techniques, we showed that bijective (8,8) S-boxes with an ACNV value of 116.0 exist. However, despite our attempts, we were not able to find an 8-variable balanced Boolean function with a nonlinearity of 118.

We first introduce the concept of couplings, coordinate decomposition, degree of descendibility, S-box coordinate extended linear approximation table (CELAT), as well as some useful properties and inner relationships. For convenience, let us denote as $f(n)^i$ the integer extracted from n, by flipping its i-th bit of its binary representation. Obviously, $f(f(n)^i)^i = n$.

Lemma 2.5.1 (The Parity Lemma)**.** Tweaking a bijective S-box S by flipping just one bit in its corresponding Look-up Table (LUT) will convert S to a non-bijective S-box.

Proof 2.5.1 (Proof of Lemma 2.5.1)**.** We take an arbitrary bijective S-box $S(n,n)$ and its corresponding Look-up S_{LUT} and Decimal Look-up S_{DLUT} tables. We pick the flipped bit to be somewhere inside the row with index i of S_{LUT}. The resulted Look-up Table will be denoted as S'_{LUT}. We will prove that the S-box S' is not bijective.

Indeed, if $S_{DLUT} = [d_0, d_1, \cdots, d_{2^n-1}]$, then the resulted Decimal Look-up Table of the S-box S' is equal to $S'_{DLUT} = [d_0, d_1, \cdots, \overline{d_i}, \cdots, d_{2^n-1}]$, where $\overline{d_i}$ is the decimal integer, which corresponds to the bit-concatenation of all the bits from the i-th row of S'_{LUT}. However, from the bijectivity property it follows that $\forall i : i \neq j \rightarrow d_i \neq d_j$. Furthermore, by definition (see Lemma 2.2.2), S_{DLUT} is in fact a permutation of all the 2^n integers in the interval $[0, 2^n - 1]$.

Since S is with dimensions (n,n), each element of S_{DLUT} is represented by exactly 2^n bits. Having this in mind, the number of possible distinct values of $\overline{d_i}$ is 2^n (d_i with the first bit flipped, d_i with the second bit flipped, ..., or d_i with the last bit flipped). Since the binary representations of all those distinct values consist of exactly n bits, their decimal representations are values less or equal to $2^n - 1$. Therefore, no matter which bit of d_i is flipped, $\overline{d_i}$ will collide with exactly one d_j, for some $j \neq i$. Hence, S'_{DLUT} will hold two different elements, $\overline{d_i}$ and d_j, with equal values, and therefore, S' is a non-bijective S-box. □

It is not possible to get a bijective S-box by modifying (flipping) a single bit of the S_{LUT} of another bijective S-box. However, as shown in the next Lemma, the minimum count of bits we need to change in the LUT of a random bijective S-box to get a new bijective S-box is 2.

Lemma 2.5.2 (Couplings Lemma)**.** The smallest nonzero number of bits from the LUT of a random bijective S-box that needed to be modified to obtain another bijective S-box is 2.

Proof 2.5.2 (Proof of Lemma 2.5.2)**.** Let us take a bijective S-box $S(n,n)$ and define the DLUT of S as an array $S_{DLUT} = [d_0, d_1, \cdots, d_{2^n-1}]$. Since S is bijective, it follows that $\forall i : i \neq j \rightarrow d_i \neq d_j$. We recall that S_{DLUT} is a permutation of all the 2^n integers in the interval $[0, 2^n - 1]$.

Without loss of generality, we pick the first flipped bit to be somewhere inside the row with index i of S_{LUT}. Let us denote the resulted Look-up Table, when this bit is flipped, as

S'_{LUT}. As shown in the previous lemma, the S-box S', which corresponds to the Look-up Table S'_{LUT}, is not bijective. However, we will show that we could always flip another distinct (non-trivial) bit, which could transform the S-box S' to some bijective S-box S'', where $S'' \neq S$.

Using the notations introduced throughout the proof of the previous lemma, we have

$$S_{DLUT} = [d_0, d_1, \cdots, d_{2^n-1}],$$

and

$$S'_{DLUT} = [d_0, d_1, \cdots, \overline{d_i}, \cdots, d_{2^n-1}],$$

where $\overline{d_i}$ is the decimal integer, which corresponds to the bit-concatenation of all the bits from the i-th row of S'_{LUT}. Following the same observation made in Lemma 2.5.1, no matter which bit of d_i is flipped, $\overline{d_i}$ will collide with exactly one d_j, for some $j \neq i$, and S'_{DLUT} will hold two different elements, $\overline{d_i}$ and d_j, with equal values

$$S'_{DLUT} = [d_0, d_1, \cdots, d_j, \cdots, \overline{d_i}, \cdots, d_{2^n-1}].$$

However, all the remaining $2^n - 2$ elements from S'_{DLUT}, i.e. all the elements in S'_{DLUT} with d_j and $\overline{d_i}$ excluded, differ from each other. Since $d_j = \overline{d_i}$, and $d_i \neq \overline{d_i}$, we could highlight two reasons for the non-bijectivity of the S-box S':

- The value d_i is missing from S'_{DLUT}.
- There are two identical values in S'_{DLUT} - d_j and $\overline{d_i}$.

Having this in mind, if we could modify d_j to d_i, by using just a single flip, we could convert S'_{DLUT} to S''_{DLUT}, where the S-box S'', which corresponds to the Decimal Look-up Table S''_{DLUT}, is bijective. It is trivial to be shown that this modification is possible. Let us recall that $\overline{d_i}$ is created by flipping a single bit in d_i on position, for example, x. Therefore, since $d_j = \overline{d_i}$, flipping the bit on position x in d_j will convert d_j back to d_i, and we will have the DLUT S''_{DLUT}, s.t:

$$S''_{DLUT} = [d_0, d_1, \cdots, d_i, \cdots, d_j, \cdots, d_{2^n-1}].$$

Since the elements in the S-box S'', which corresponds to the Decimal Look-up Table S''_{DLUT}, are now a permutation of all the 2^n integers in the interval $[0, 2^n - 1]$, S'' is bijective. Furthermore, the permutation S''_{DLUT} is exactly one transposition away from the permutation S_{DLUT}. □

We have shown that if we start from a random bijective S-box S it is possible to construct another bijective S-box $S^{''}$ by flipping exactly two bits from the LUT of S. It appears that the first flip can be on a random element in the LUT, but the second flip is uniquely determined by the first one. We define each such pair as coupling.

Definition 2.5.1 (Couplings)**.** Let us take a bijective S-box $S(n,n)$ and its corresponding DLUT

$$S_{DLUT} = [d_0, d_1, \cdots, d_i, \cdots, d_{2^n-1}].$$

We define as a **coupling** each set $\{d_s, f(d_s)^j\}$, while the set of all couplings in S as $\{S \updownarrow\}$.

Lemma 2.5.3 (Couplings Set Cardinality)**.** Given a bijective S-box $S(n,n)$:

$$|\{S \updownarrow\}| = n2^{n-1}.$$

Proof 2.5.3 (Proof of Lemma 2.5.3)**.** If we flip a bit on column i in the LUT of S, the corresponding unique second flip we need to perform to guarantee the bijectivity property of the newly created S-box has to be on column i as well, i.e. we are flipping two distinct bits sharing the same S-box coordinate i. Thus, we have exactly n coordinates, each having $\frac{2^n}{2}$ distinct couplings, or a total of $n2^{n-1}$ couplings. □

Definition 2.5.2 (Couplings Pivot Set)**.** We define the set $\{S \updownarrow^i\}$ as the maximum subset of the coupling set of a bijective S-box S(n,n), which holds couplings operating only on column i of the S_{LUT}, i.e. couplings of the form $\{d_x, f(d_x)^i\}$. We call each such maximum subset $\{S \updownarrow^i\}$ a couplings pivot set operating on column i of S_{LUT}.

Corollary 2.5.1 (Properties of Couplings Pivot Sets)**.** Considering the definitions of the Couplings Pivot Sets on bijective S-box S(n,n), the following properties hold:

- $\forall i \neq j, \{S \updownarrow^i\} \cap \{S \updownarrow^j\} = \emptyset$
- $\forall i, |\{S \updownarrow^i\}| = 2^{n-1}$
- $|\bigcup_{i=1}^{n} \{S \updownarrow^i\}| = n2^{n-1}$

Definition 2.5.3 (Coordinate Decomposition)**.** Let S be an (n,n) bijective S-box. We take a random element with coordinates (x,y) of its corresponding linear approximation table S_{LAT}. We denote the binary representation of y as:

$$y_{(2)} = b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \cdots + b_1 2^1 + b_0 2^0$$

The coordinate decomposition of an element with coordinates (x,y), denoted by $\triangle_{x,y}$, is the set:

$$\triangle_{x,y} = \bigcup_{i=0, b_i \neq 0}^{n-1} \{b_i(n-i-1)\}$$

Definition 2.5.4 (Nonlinearity Bottleneck Snapshot – NBS)**.** We define the nonlinearity bottleneck snapshot S_{NBS} of a bijective S-box $S(n,n)$ as a set of tuples holding all coordinates of the elements of S_{LAT}, which are holding down the nonlinearity value S_{NL} of S, i.e.

$$(x,y) \in S_{NBS} \Leftrightarrow |LAT_S[x][y]| = 2^{n-1} - S_{NL}$$

Definition 2.5.5 (NBS Coordinate Decomposition – NBSCD)**.** We define the nonlinearity bottleneck snapshot coordinate decomposition of a bijective S-box $S(n,n)$, denoted by Δ_S, as a set of all S_{NBS} coordinate decompositions, i.e.:

$$\Delta_S = \bigcup_{(x,y) \in S_{NBS}} \triangle_{x,y}$$

Definition 2.5.6 (Degree of Descendibility – Λ_S)**.** For a given bijective S-box $S(n,n)$, we define a family of sets Ψ_S, s.t.:

$$E \in \Psi_S \Leftrightarrow \forall Q \in \Delta_S \; \exists q \in Q : \; q \in E$$

The degree of descendibility of S is the minimum cardinality of a set in Ψ_S, i.e.:

$$\Lambda_S = \min_{\forall A \in \Psi_S} |A|$$

Corollary 2.5.2 (Basic properties of Λ_S)**.** For a given bijective S-box $S(n,n)$:

- $\Lambda_S \in \mathbb{N}$
- $\Lambda_S \in [1,n]$
- $\Lambda_S = 1 \Leftrightarrow | \bigcap_{s \in \Delta_S} | \geq 1$
- $\Lambda_S > 1 \Leftrightarrow \bigcap_{s \in \Delta_S} = \emptyset$

Definition 2.5.7 (Descendible Coordinate)**.** For a given bijective S-box $S(n,n)$, we say that coordinate j is descendible if the following properties hold:

- $\Lambda_S = 1$

- $j \in \bigcap_{s \in \Delta_S}$

Definition 2.5.8 (Couplings Transformation)**.** For a given bijective S-box $S(n,n)$ and some coupling c_i, we denote as S^{c_i} the S-box created by applying coupling c_i on S. We define this transform as coupling transform denoting it with the operator $\circ$, i.e.

$$S^{c_i} = S \circ c_i$$

When we have a list of couplings $\{c_1, c_2, \cdots, c_i\}$, which we want to use for transformation of S in this exact order, we will use the following expression:

$$S^{c_1, c_2, \cdots, c_i} = S \circ c_1 \circ c_2 \circ \cdots \circ c_i$$

Lemma 2.5.4 (Couplings Inverse)**.** For a given bijective S-box S and any coupling c, the following property holds:

$$S = S \circ c \circ c$$

Proof 2.5.4. Since c is, in fact, a transposition in the Decimal Look-up Table S_{DLUT} of S (swapping two elements in S_{DLUT}) applying the same transposition twice would cancel its effect out. □

Definition 2.5.9 (Coupling Transformation Matrix – CTM)**.** For a given bijective S-box $S(n,n)$ and some coupling c_i, we denote as $S^{c_i}{}_{LAT}$ the transformed LAT of S caused by c_i. We define the coupling transformation matrix of c_i on S, as:

$$S^{c_i}_{CTM} = S^{c_i}{}_{LAT} - S_{LAT}$$

Lemma 2.5.5 (Pivot Couplings Commutativity)**.** For a given bijective S-box $S(n,n)$, for any two couplings c_a and c_b, which belongs to the same couplings pivot set $\{S \updownarrow^i\}$, we have the following property:

$$S \circ c_a \circ c_b = S \circ c_b \circ c_a$$

Proof 2.5.5. In case $c_a \equiv c_b$ the theorem follows from lemma 2.5.4, i.e.

$$S \circ c_a \circ c_b = S \circ c_a \circ c_a = S$$

In the case when $c_a \neq c_b$, since they belong to the same coupling pivot set, it follows that $c_a \cap c_b = \varnothing$, which concludes the proof. □

Corollary 2.5.3. For a given bijective S-box $S(n,n)$, for any couplings c_j, which belongs to the same couplings pivot set $\{S \updownarrow^i\}$, we have the following properties:

$$S_{LAT}^{c_a,c_b} = S_{LAT}^{c_b,c_a} = S_{LAT} + S_{CTM}^{c_a} + S_{CTM}^{c_b}$$

$$S_{LAT}^{c_1,c_2,\cdots,c_k} = S_{LAT} + \sum_{i=1}^{k} S_{CTM}^{c_i}$$

Lemma 2.5.6 (CTM Values)**.** The value of each element in a CTM is -2, 0, or 2.

Proof 2.5.6. For a given bijective S-box $S(n,n)$ and some coupling $c = \left\{d_x, f(d_x)^j\right\}$, we denote as $S^c{}_{LAT}$ the transformed LAT of S caused by c. Let us take some element $e_{x,y}$ from the LAT of S before the coupling transformation. We have:

$$e_{x,y} = 2^{n-1} - d_H(L_q, b_1 b_2 \cdots b_{2^n}),$$

for some linear function L_q and some linear combination in binary representation of the coordinates of $S : b = b_1 b_2 \cdots b_{2^n}$. If $j \notin \Delta_{x,y}$, $e_{x,y}$ is not affected after applying the coupling. However, if $j \in \Delta_{x,y}$, we know that exactly two of the bits of the linear combination b are flipped. We denote them as b_s and b_t. Let us denote the element on position (x,y) on the newly created LAT as $e'_{x,y}$.

$$\begin{aligned} e'_{x,y} &= 2^{n-1} - d_H(l_q, b_1 b_2 \cdots \bar{b}_s \cdots \bar{b}_t \cdots b_{2^n}) \\ &= 2^{n-1} - d_H(l_q, b_1 b_2 \cdots b_s \cdots \bar{b}_t \cdots b_{2^n}) \pm 1 \\ &= 2^{n-1} - d_H(l_q, b_1 b_2 \cdots b_s \cdots b_t \cdots b_{2^n}) \pm 1 \pm 1 \\ &= e_{x,y} \pm 1 \pm 1 \end{aligned} \tag{2.8}$$

Since the expression $\pm 1 \pm 1$ is equal to one of the three possible values: -2, 0, and 2, the proof concludes. □

Corollary 2.5.4. For a given bijective S-box $S(n,n)$, let us apply transformations of couplings $c_1, c_2, \cdots, c_k$, which belongs to the same couplings pivot set $\{S \updownarrow^i\}$. The elements of the resulting CTM are numbers in the interval $[-2k, -2(k-1), \cdots, -2, 0, 2, \cdots, 2(k-1), 2k]$.

Definition 2.5.10 (S-box Coordinate Extended LAT – CELAT)**.** For a given bijective S-box $S(n,n)$, and a given coordinate i, we can define the one-dimensional linear approximation table of S as:

$$S_{LAT_{1D}}[x] = S_{LAT}[x \ / \ 2^n][x \ \% \ 2^n]$$

Furthermore, we denote all the couplings in the couplings pivot set $\{S\updownarrow^i\}$ as $c_1, c_2, \cdots, c_{2^{n-1}}$. We have:

$$\begin{aligned} S^{c_1}_{CTM} &= S^{c_1}_{LAT} - S_{LAT} \\ S^{c_2}_{CTM} &= S^{c_2}_{LAT} - S_{LAT} \\ &\cdots \\ S^{c_{2^{n-1}}}_{CTM} &= S^{c_{2^{n-1}}}_{LAT} - S_{LAT} \end{aligned} \tag{2.9}$$

Following the same concept used in the construction of one-dimensional LAT of S, we can define one-dimensional CTM, i.e.:

$$\begin{aligned} S^{c_1}_{CTM_{1D}} &= S^{c_1}_{LAT_{1D}} - S_{LAT_{1D}} \\ S^{c_2}_{CTM_{1D}} &= S^{c_2}_{LAT_{1D}} - S_{LAT_{1D}} \\ &\cdots \\ S^{c_{2^{n-1}}}_{CTM_{1D}} &= S^{c_{2^{n-1}}}_{LAT_{1D}} - S_{LAT_{1D}} \end{aligned} \tag{2.10}$$

Finally, we define S-box i-th Coordinate Extended LAT S^i_{CELAT} as the following table:

$$S^i_{CELAT} = \begin{bmatrix} S_{LAT_{1D}} \\ S^{c_1}_{CTM_{1D}} \\ S^{c_2}_{CTM_{1D}} \\ \cdots \\ S^{c_{2^{n-1}}}_{CTM_{1D}} \end{bmatrix}$$

S^i_{CELAT} has $2^{n-1} + 1$ rows and 2^{2n} columns.

For example, let us consider an S-box $S(2,2)$ with $S_{DLUT} = [0,2,1,3]$. For $n = 2$ the S^2_{CELAT} has $2^2 - 1 = 3$ rows and $2^2 * 2 = 16$ columns. Considering coordinate 2, we have:

$$\{S\updownarrow^2\} = \{c_1, c_2\} = \{\{0,1\},\{2,3\}\} \quad S^{c_1}_{DLUT} = [1,2,0,3] \quad S^{c_2}_{DLUT} = [0,3,1,2]$$

$$S^2_{CELAT} =$$

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -2 & 0 & -2 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 2 & 0 & 2 & 0 & -2 \end{bmatrix}$$

Definition 2.5.11 (Integer Programming – Optimization Problem)**.** A pure integer linear program is a problem of the form:

$$\begin{array}{rccl} \max & cx & & \\ \text{subject to} & Ax & \leq & b \\ & x & \geq & 0 \quad \text{integral} \end{array}$$

where the data consists of the row vector c with size n, (m,n) matrix A, and column vectors b and x with respective sizes of m and n. The column vector x contains the variables to be optimized. We say that the set S is the set of feasible solutions, i.e.:

$$S := \left\{x \in Z_+^n : Ax \leq b\right\}$$

Definition 2.5.12 (Binary Integer Linear Programming – BILP)**.** A pure binary integer linear program is a problem of the form:

$$\begin{array}{rccl} \max & cx & & \\ \text{subject to} & Ax & \leq & b \\ & x & \geq & 0 \quad \text{binary} \end{array}$$

where the data consists of the row vector c with size n, (m,n) matrix A, and column vectors b and x with respective sizes of m and n. The column vector x contains the binary variables to be optimized. We say that the set S is the set of feasible solutions, i.e.:

$$S := \left\{x \in B^n : Ax \leq b\right\}$$

Definition 2.5.13 (Binary Integer Programming – Feasibility or SAT Problem)**.** A feasibility binary integer program is a problem of the form:

$$\begin{array}{rccl} \text{subject to} & Ax & \leq & b \\ & x & \geq & 0 \quad \text{binary} \end{array}$$

where the data consists of (m,n)-matrix A and column vectors b and x with respective sizes of m and n. The column vector x contains the binary variables to be optimized. We say that the set S is the set of feasible solutions, i.e.:

$$S := \left\{x \in B^n : Ax \leq b\right\}$$

In the context of the feasibility problem we are looking for just one element in the set S, not the optimal one.

For an (n,n) S-box S, we denote 2^{n-1} by r and 2^{2n} by m. Let us construct its CELAT using coordinate i i.e:

$$S^i_{CELAT} = \begin{bmatrix} S_{LAT_{1D}} \\ S^{c_1}_{CTM_{1D}} \\ S^{c_2}_{CTM_{1D}} \\ \cdots \\ S^{c_{2^{n-1}}}_{CTM_{1D}} \end{bmatrix} = \begin{bmatrix} l_1 & l_2 & \cdots & l_m \\ c_{11} & c_{12} & \cdots & c_{1m} \\ c_{21} & c_{22} & \cdots & c_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ c_{r1} & c_{r2} & \cdots & c_{rm} \end{bmatrix}$$

We want to apply some coupling transformations subset $P = p_1, p_2, \cdots, p_k$ which belongs to the pivot coupling set $\{S \updownarrow^i\}$. From corollary 2.5.3 it follows that:

$$S^{p_1,p_2,\cdots,p_k}_{LAT} = S_{LAT} + \sum_{i=1}^{k} S^{p_i}_{CTM}$$

We denote

$$S^{p_1,p_2,\cdots,p_k}_{LAT_{1D}} = [q_1, q_2, \cdots, q_m]$$

Then, we can construct the following system of equations:

$$\begin{aligned} q_1 &= l_1 + c_{11}x_1 + c_{21}x_2 + \cdots + c_{r1}x_r \\ q_2 &= l_2 + c_{12}x_1 + c_{22}x_2 + \cdots + c_{r2}x_r \\ \cdots \\ q_m &= l_m + c_{1m}x_1 + c_{2m}x_2 + \cdots + c_{rm}x_r \end{aligned} \tag{2.11}$$

where $x = (x_1, x_2, \cdots, x_r) \in B^r$, and $x_t = 1$ iff $p_t \in P$. We have $S_{NL} = 2^{n-1} - \max_{j=1}^{m} abs(l_j)$. If coordinate i is descendable, we can construct the following binary integer programming feasibility problem:

$$\begin{array}{llll} \text{subject to} & < {S^i_{CELAT}}^T, x > & \leq & A \\ \text{subject to} & < {S^i_{CELAT}}^T, x > & \geq & B \\ & x & \geq & 0 \quad \text{binary} \end{array}$$

where A is a column vector with $2^{n-1} + 1$ elements, each equal to $2^{n-1} - S_{NL} - 2$, while B is a column vector with $2^{n-1} + 1$ elements, each equal to $S_{NL} - 2^{n-1} + 2$. Let us denote the SAT problem descending on coordinate i in equation 2.5 as $\Omega_{S,i}$. This is NP-hard[2] problem with a total of 2^{n-1} binary variables and $2^n + 2$ restrictions. However, we can further divide the problem to an union of subproblems, i.e.:

$$\Omega_{S,i} = \bigcup_{d=1}^{n-1} \Omega_{S,i}^{d}$$

where each subproblem $\Omega_{S,i}^{d}$ is modelled using the following restrictions:

$$\begin{array}{llcll}
\text{subject to} & < {S_{CELAT}^{i}}^{T}, x > & \leq & A & \\
\text{subject to} & < {S_{CELAT}^{i}}^{T}, x > & \geq & B & \\
\text{subject to} & \sum_{j=1}^{r} x_j & = & d & \\
 & x & \geq & 0 & \text{binary}
\end{array}$$

Solving any of the subproblems will yield a solution to the original problem.

For subproblems $\Omega_{S,i}^{d}$ of a binary integer programming feasibility problem $\Omega_{S,i}$, the following property holds:

$$\bigcap_{d=1}^{n-1} \Omega_{S,i}^{d} = \emptyset$$

It is easy to show that the search space of the subproblem $\Omega_{S,i}^{d}$ for the bijective S-box $S(n,n)$ is $\binom{2^{n-1}}{d}$.

Theorem 2.5.1. For a subproblem $\Omega_{S,i}^{d}$, all restrictions with the participation of some l_j for which the following inequalities hold:

$$\begin{aligned} l_j &\leq 2^{n-1} - S_{NL} - 2d - 2 \\ l_j &\geq S_{NL} - 2^{n-1} + 2d + 2 \end{aligned} \tag{2.12}$$

are always satisfied.

[2]The complexity class of decision problems that are intrinsically harder than those that can be solved by a nondeterministic Turing machine in polynomial time.

Proof 2.5.7. For a subproblem $\Omega^d_{S,i}$ we have the following restrictions:

$$\begin{aligned}
l_1 + c_{11}x_1 + c_{21}x_2 + \cdots + c_{r1}x_r &\le 2^{n-1} - S_{NL} - 2 \\
l_1 + c_{11}x_1 + c_{21}x_2 + \cdots + c_{r1}x_r &\ge S_{NL} - 2^{n-1} + 2 \\
l_2 + c_{12}x_1 + c_{22}x_2 + \cdots + c_{r2}x_r &\le 2^{n-1} - S_{NL} - 2 \\
l_2 + c_{12}x_1 + c_{22}x_2 + \cdots + c_{r2}x_r &\ge S_{NL} - 2^{n-1} + 2 \\
&\cdots \\
l_m + c_{1m}x_1 + c_{2m}x_2 + \cdots + c_{rm}x_r &\le 2^{n-1} - S_{NL} - 2 \\
l_m + c_{1m}x_1 + c_{2m}x_2 + \cdots + c_{rm}x_r &\ge S_{NL} - 2^{n-1} + 2 \\
x_1 + x_2 + \cdots + x_r &= d
\end{aligned} \tag{2.13}$$

From lemma 2.5.6, we know that the possible values of the elements c_{ij} are -2, 0 or 2. Hence:

$$\min_{i,j} c_{ij} = -2$$

$$\max_{i,j} c_{ij} = 2$$

Since $\sum x_j = d$, we have:

$$\min_j (c_{1j}x_1 + c_{2j}x_2 + \cdots + c_{rj}x_r) = -2d$$

$$\max_j (c_{1j}x_1 + c_{1j}x_2 + \cdots + c_{rj}x_r) = 2d$$

If for some l_j the following inequalities hold:

$$\begin{aligned}
l_j &\le 2^{n-1} - S_{NL} - 2d - 2 \\
l_j &\ge S_{NL} - 2^{n-1} + 2d + 2
\end{aligned} \tag{2.14}$$

then

$$\begin{aligned}
l_1 + c_{11}x_1 + c_{21}x_2 + \cdots + c_{r1}x_r &\le \\
\le l_1 + \max(c_{11}x_1 + c_{21}x_2 + \cdots + c_{r1}x_r) &\le \\
\le l_1 + 2d &\le \\
\le 2^{n-1} - S_{NL} - 2d - 2 + 2d &\le \\
\le 2^{n-1} - S_{NL} - 2 &
\end{aligned} \tag{2.15}$$

and on the other hand

$$\begin{aligned}
&l_1 + c_{11}x_1 + c_{21}x_2 + \cdots + c_{r1}x_r \geq \\
&\geq l_1 + \min(c_{11}x_1 + c_{21}x_2 + \cdots + c_{r1}x_r) \geq \\
&\geq l_1 - 2d \geq \\
&\geq S_{NL} - 2^{n-1} + 2d + 2 - 2d \geq \\
&\geq S_{NL} - 2^{n-1} + 2
\end{aligned} \tag{2.16}$$

which completes the proof. □

Definition 2.5.14 (CELAT with radius R)**.** For a given bijective S-box $S(n,n)$, and a given coordinate i, we have:

$$S^i_{CELAT} = \begin{bmatrix} l_1 & l_2 & \cdots & l_m \\ c_{11} & c_{12} & \cdots & c_{1m} \\ c_{21} & c_{22} & \cdots & c_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ c_{r1} & c_{r2} & \cdots & c_{rm} \end{bmatrix}$$

We define as $S^{i,R}_{CELAT}$ a matrix constructed of those columns of S^i_{CELAT} with first element ρ, for which the following inequalities hold:

$$\begin{aligned}
&\rho > 2^{n-1} - S_{NL} - 2R - 2 \\
&\rho < S_{NL} - 2^{n-1} + 2R + 2
\end{aligned} \tag{2.17}$$

Hence, a given suproblem $\Omega^d_{S,i}$ could be further reduced and launched on $S^{i,d}_{CELAT}$, instead of its corresponding full (unreduced) version S^i_{CELAT}.

By using automata notation, Figure 2.5 presents the distinct steps of the optimization process. State **S** is the initial state of the automata. In this phase, we initialize and process the input. We make some additional checks about the properties of the S-box. For example, we check the bijectivity property of the S-box. We further analyze and extract the descendable coordinates (if such exist).

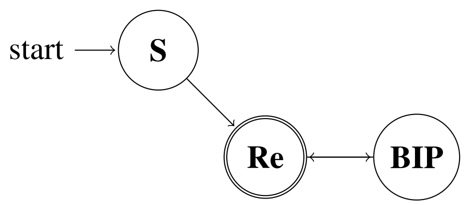


Fig. 2.5 Automata representation of the optimization process

A few important properties of the automata should be emphasized. For a given S-box S, if $\Lambda_S = 1$, then at least one descendible coordinate, for example j, does exist. Thus, if a feasible solution of $\Omega^R_{S,j}$ is found, the nonlinearity of S could be increased by activating exactly R couplings. Therefore, we could not only optimize the nonlinearity of S but dictate the impact of our changes to the original S-box as well - increasing the value of R will increase the total count of flipped bits in S.

For example, if we first choose the coordinate j to descend into, we further calculate the corresponding matrix S^j_{CELAT}. Then, the adjacent state *Re*, by further processing the generated matrix, and by using some radius R, generates the matrix $S^{j,R}_{CELAT}$. Finally, state *BIP* is translating the problem to a binary integer programming feasibility problem. In case a feasible solution or proof that the problem is infeasible is found, the result is reported back to state Re, the major role of which is to orchestrate the behavior of the optimization routine - increasing the radius, changing the descendible coordinate, or giving up.

In those cases where $\Lambda_S > 1$, the aforementioned algorithm, as we will later demonstrate, is still applicable. We just pick a random coordinate $j \in \Delta_S$ instead of a descendible one. As a consequence, finding a solution to the $\Omega^R_{S,j}$ problem will not increase the nonlinearity of S. However, it will decrease the value of Λ_S by 1. Thus, by repeating the reduction phase, we would eventually reduce the initial problem to a problem having $\Lambda_{S'} = 1$, for some S-box S', yielded by the optimization routine, or the composition of optimization routines, performed on S.

We have implemented the algorithm by using Python, for states S and Re, and the Gurobi SAT Solver [71], for the BIP state itself. We analyzed two famous S-boxes: the Skipjack S-box, developed by the U.S. National Security Agency (NSA) [138], which we will denote as S_k, and the Kuznyechik S-box, standardized by the Russian Federation's standardization agency [53], which we will denote as K_k.

2.5.1 Skipjack (Case S_k)

The characteristics of Sk are $Sk_{NL} = 100$, $Sk_{NBS} = \{(138, 89), (125, 168), (77, 168)\}$, $\Delta_{Sk} = \{(1, 3, 4, 7), (0, 2, 4)\}$ and $\Lambda_{Sk} = 1$. Since the coordinate with index 4 is descendible, we first try to solve the problem $\Omega^1_{Sk,4}$, trying the minimum possible radius value of 1. The translated BIP model consists of 51 rows, 128 columns, and 3420 nonzeros. By using a general-purpose CPU, it took approximately 0.115 seconds to prove that $\Omega^1_{Sk,4}$ is infeasible. However, by increasing the radius value by 1, the translated BIP model of $\Omega^2_{Sk,4}$, consisting of 189 rows, 128 columns, and 12640 nonzeros, a solution is found. The time required was 0.301 seconds. The found solution coupling set is $\{(130, 138), (183, 191)\}$. Indeed, the resulting S-box does possess a nonlinearity of 102. Furthermore, the found solution required the flipping of only

4 bits, since, by design, each activated coupling modifies exactly 2 bits in the S-box it was launched on.

On the other hand, if we require a higher nonlinearity, combined with a greater impact of the structure of Sk, we could significantly increase the value of R. Indeed, using a radius value of 10, the translated BIP model of $\Omega^{10}_{Sk,4}$ consists of 25125 rows, 128 columns, and 1621980 nonzeros. Despite the greater model, after 14.542 seconds, a solution was found, yielding an S-box with nonlinearity 102, constructed from Sk by flipping exactly 20 bits.

2.5.2 Kuznyechik (Case K_k)

The characteristics of Kk are Kk_{NL} = 100, Kk_{NBS} = { (90,47), (184,105), (55,165), (102,103), (222,151), (62,105), (72,85), (237,98), (110,15), (246,28), (65,106), (135,171), (76,167), (251,54) }, Δ_{Kk} = { (2,3,5,6), (1,2,5,6,7), (0,2,5,7), (1,3,5,7), (1,2,4,7), (0,2,5,6,7), (0,3,5,6,7), (3,4,5), (0,2,4,6,7), (1,2,6), (1,2,4,6), (2,4,5,6,7), (4,5,6,7)} and Λ_{Kk} = 2. Since the degree of descendibility is greater than 1, more precisely $\Delta_{Kk} = \{2,5\}$, we pick a random coordinate from Δ_{Kk}. $\Omega^1_{Kk,5}$ and $\Omega^2_{Kk,5}$ are reported back as infeasible. However, $\Omega^3_{Kk,5}$, consisting of 319 rows, 128 columns, and 20904 nonzeros, is feasible. Again, the solver took less than a second to find a solution, i.e. the coupling set $\{(0,4),(67,71),(136,140)\}$. Let's denote the resulting S-box as $\widehat{Kk}$. The characteristics of $\widehat{Kk}$ are $\widehat{Kk}_{NL}$ = 100, Kk_{NBS} = $\{(65,106),(135,171),(62,105),(237,98),(184,105)\}$, $\Delta_{\widehat{Kk}}$ = { (0,2,4,6,7), (1,2,6), (1,2,4,6), (1,2,4,7)} and $\Lambda_{\widehat{Kk}}$ = 1. As expected, the value of Λ is decreased by 1. Thus, we continue the optimization process by the descendible coordinate with index 2. First, the infeasibility of the two models $\Omega^1_{\widehat{Kk},2}$ and $\Omega^2_{\widehat{Kk},2}$ are proved. However, the BIP model of $\Omega^3_{\widehat{Kk},2}$, consisting of 379 rows, 128 columns, and 25344 nonzeros, yielded a solution for less than a second. Indeed, the found coupling set $\{(95,127),(207,239),(108,157)\}$ increased the nonlinearity of $\widehat{Kk}$ from 100 to 102. Hence, we have shown how Kk could be optimized to an S-box with higher nonlinearity by tweaking just 12 bits.

2.5.3 The ACNV problem

The ACNV optimization problem could be represented as a special, and significantly lighter, in terms of computational burden, case of $S^{i,R}_{CELAT}$, where S denotes the initial S-box and i denotes the coordinate of S to be optimized. Since our goal is ACNV optimization only, we could significantly reduce the size of $S^{i,R}_{CELAT}$. Let us denote as $S^{i,R}_{ACNV}$, the matrix formed by the matrix $S^{i,R}_{CELAT}$, with columns corresponding to linear combinations of coordinates of S removed. Indeed, this is a significant reduction of the feasibility model. For example, if S is of dimension (n,n), $S^{i,R}_{CELAT}$ has $2^{n-1}+1$ rows and 2^{2n} columns, while $S^{i,R}_{ACNV}$ has $2^{n-1}+1$

rows and 2^n columns. We further denote the corresponding feasibility problem corresponding to $S^{i,R}_{ACNV}$ as $\Psi_{S,i}$. As usual, we could divide the problem $\Psi_{S,i}$ to a union of subproblems $\Psi^d_{S,i}$.

We have initiated the optimization routine on a bijective S-box, for simplicity denoted as $\mathcal{S}$, from [50], possessing the highest, currently known, ACNV of 114.5. It is composed of 6 coordinates with a nonlinearity value of 114 and 2 coordinates with a nonlinearity value of 116. We will outline a possible trace of improvement, which led to an S-box with an ACNV of 116.0.

- We launched $\Psi^{\leq 9}_{\mathcal{S},4}$. After around 153 seconds a feasible solution, with exactly 9 couplings, was found. We activated the couplings to get $\mathcal{S}_1$. ACNV was lifted to 114.75.

- We launched $\Psi^{\leq 8}_{\mathcal{S}_1,1}$. After around 16 seconds a feasible solution, with exactly 8 couplings, was found. We activated the couplings to get $\mathcal{S}_2$. ACNV was lifted to 115.

- We consequently launched $\Psi^{\leq 9}_{\mathcal{S}_2,2}$, $\Psi^{\leq 9}_{\mathcal{S}_2,3}$ and $\Psi^{\leq 9}_{\mathcal{S}_2,5}$, to prove their infeasibility for respectively 181, 274 and 173 seconds. However, by launching $\Psi^{\leq 9}_{\mathcal{S}_2,7}$, after 257 seconds, a feasible solution with exactly 9 couplings was found. We activated the couplings to get $\mathcal{S}_3$. ACNV was lifted to 115.25.

- We consequently launched $\Psi^{\leq 9}_{\mathcal{S}_3,2}$, $\Psi^{10}_{\mathcal{S}_3,2}$ and $\Psi^{11}_{\mathcal{S}_3,2}$, to prove their infeasibility for respectively 151, 698 and 3457 seconds. Then, we continued with $\Psi^{\leq 9}_{\mathcal{S}_3,3}$ and $\Psi^{10}_{\mathcal{S}_3,3}$ to prove their infeasibility for respectively 240 and 1015 seconds. However, $\Psi^{11}_{\mathcal{S}_3,3}$ yielded a solution after 5171 seconds. We activated the 11 couplings to get $\mathcal{S}_4$. ACNV was lifted to 115.5.

- We continued with $\Psi^{\leq 9}_{\mathcal{S}_4,2}$ and $\Psi^{10}_{\mathcal{S}_4,2}$ to prove their infeasibility for respectively 170 and 715 seconds. However, $\Psi^{11}_{\mathcal{S}_4,2}$ yielded a solution after 1145 seconds. We activated the 11 couplings to get $\mathcal{S}_5$. ACNV was lifted to 115.75.

- Finally, we launched $\Psi^{\leq 9}_{\mathcal{S}_5,5}$, and a feasible solution was found after 69 seconds. We activated the 9 couplings to get $\mathcal{S}_6$. ACNV was lifted to 116.

We present $\mathcal{S}_6$ in Figure 2.6 in a hexadecimal format.

The overall nonlinearity of $\mathcal{S}_6$ is 92.

Significant efforts were made to reach higher ACNV - reaching higher ACNV would reveal a balanced Boolean function having a nonlinearity of 118. Unfortunately, all instances $\Psi^x_{\mathcal{S}_6,y}$, for $\forall x, y : x \leq 21, y \in [1, 8]$, were proofed infeasible. We want to emphasize that the search routine is highly efficient. For example, $\Psi^{17}_{\mathcal{S}_6,y}$, for some y, was proved infeasible for 1065 seconds or approximately 18 minutes. Since the given search space size is equal

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	a9	7b	99	49	0a	45	b3	c1	a3	5a	24	26	bf	72	b6	05
10	4a	73	e2	f1	2a	d1	25	92	64	f3	f5	d7	ff	dc	cb	4e
20	de	7a	22	98	f9	87	b1	a5	28	9a	b0	55	16	67	61	0c
30	27	33	53	2d	c7	58	7e	f6	37	71	1e	10	d0	e0	b7	c9
40	9e	91	6e	20	d9	5b	fb	13	8a	db	ad	a1	8c	39	a2	ee
50	89	4f	50	1a	07	35	65	bd	9f	18	cd	17	41	be	2f	00
60	ca	0d	ae	3a	94	f7	a8	93	aa	f8	e9	e6	b2	54	01	69
70	a7	81	5c	86	77	f4	29	d2	ec	0e	e4	56	90	2e	1d	40
80	4c	51	75	11	3e	d3	3d	8d	9c	6c	95	ef	76	c4	8b	dd
90	23	b4	ce	43	62	d6	74	fe	82	02	7c	80	32	2b	78	fc
a0	c0	21	af	e3	68	6f	e1	eb	03	38	09	c2	d4	ed	bc	12
b0	15	fa	5e	bb	c8	e7	c6	14	a4	b9	9d	04	cc	d8	3f	9b
c0	e5	4d	31	63	79	1c	d5	f0	47	7f	0b	46	f2	2c	70	b5
d0	cf	8e	4b	36	1f	da	a6	6a	6b	42	19	57	5d	48	ac	1b
e0	44	3c	5f	ea	a0	85	8f	30	ba	ab	34	c3	59	96	fd	08
f0	b8	e8	84	6d	66	7d	df	60	52	83	88	3b	0f	97	c5	06

Fig. 2.6 An optimized S-box $S_c(8,8)$ with ACNV of 116.0 using SAT techniques

to $\binom{128}{17}$, or approximately 2^{69}, this results in checking simultaneously roughly 2^{59} distinct cases per second. The results are published in [51].

Chapter 3

On the S-box Reverse Engineering

3.1 Introduction and motivation

The reasons for obfuscating the design of a given S-box are manifold. For example, the initial S-boxes used in the Data Encryption Standard (DES) [55] were originally modified by NSA. The reasons for applying those modifications were not known. However, in [33], D. Coppersmith announces the motivation behind the S-box modifications. It appears that the agency knew about the existence of differential attacks about 20 years before the academic world. However, they kept that in secrecy. D. Coppersmith further commented on this secrecy decision by saying:

> *... that was because [differential cryptanalysis] can be a very powerful tool, used against many schemes, and there was concern that such information in the public domain could adversely affect national security.*

Another reason for hiding a given S-box design could be related to some hidden structure, the knowledge of which could be exploited to gain a significant advantage in terms of hardware implementation. For example, as discovered in [21], the S-boxes used in the hash function Streebog and the 128-bit block cipher Kuznyechik, standardized by the Russian Federation, are designed with such a hidden structure. A user knowing the not published decomposition could implement the given S-box with a significantly smaller hardware footprint, allowing him to reach an up to 8 times faster S-box look-up.

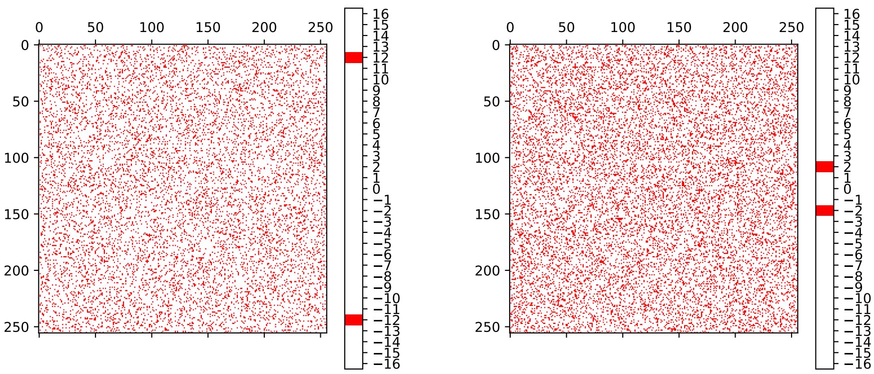
Another practical reason for hiding the design of a given S-box could be related to an encapsulated trapdoor as discussed in [128]. Although the aforementioned trapdoor can be easily detected, as shown in [151], the motivation for finding other trapdoor S-box techniques should not be underestimated.

There are various tools and techniques, which could help us to initiate some S-box reverse engineering (see [119][20][120]). In the next section, the concept of S-box spectrography is presented. A good example of using spectrography for S-box reverse engineering purposes is the Pollock representation (see [21]).

3.2 S-box spectrography

We can isolate the coordinates, in terms of row and column indexes, of those elements of the LAT of a given S-box $S(n,m)$, which are equal to some fixed value or, in the more unrestricted case, belong to some set of values of our choice. We define each distinct isolation as a **spectra channel**. For convenience, we denote as $\S_S^E$ the spectra channel isolated from an S-box S, by using restriction set E. We can further visualize the channel as a $(2^n \times 2^m)$ matrix plot – those elements, which belong to the restriction set are colored in red, while the remaining elements are left colorless.

In Figures 3.1a and 3.1b two spectra channels of the popular Rijndael S-box [39] are presented. Since its dimensions, i.e. $(8,8)$, the LAT table has 2^8 rows and 2^8 columns, or a total of 2^{16} elements. For example, in Figure 3.1a only the elements from the corresponding S-box LAT equal to -12 or 12 are colored, while in Figure 3.1b the restriction set E is equivalent to $\{-2,2\}$.



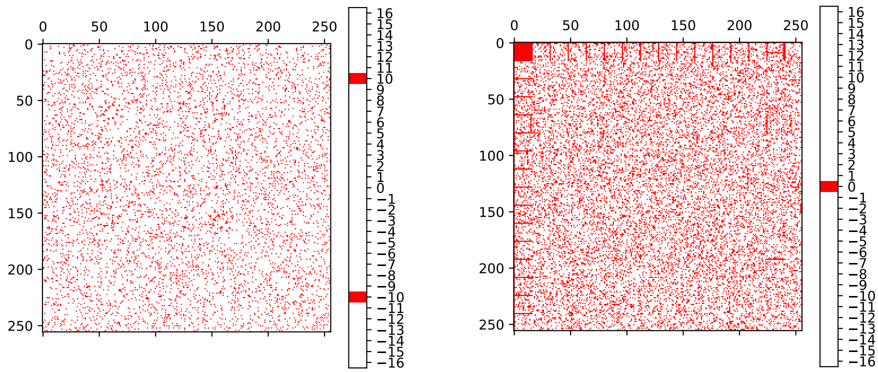
(a) $\S_{AES}^{-12,12}$ (b) $\S_{AES}^{-2,2}$

Fig. 3.1 Some spectra channels of Rijndael S-box

During our experiments, we repeatedly generated random bijective $(8,8)$ S-boxes and thoroughly analyzed their spectra channels. However, we didn't find any anomalies, symmetries, or visual patterns. It is really difficult to distinguish visually their spectra channel plots from plots populated with randomly scattered points.

In [132] a rich database of popular S-boxes is published. The rest of this section presents our results in applying spectra channel analysis on the aforementioned S-box collection. Anubis is a block cipher, which was submitted to the NESSIE project [127]. The Anubis S-box is constructed by using involutions. It appears that such constructions are easily detected by using some spectra channel plot of the form $\S^{-x,x}$. Indeed, as shown in Figure 3.2a, by applying the restriction $\{-10,10\}$ the plot is symmetric with respect to the main diagonal.

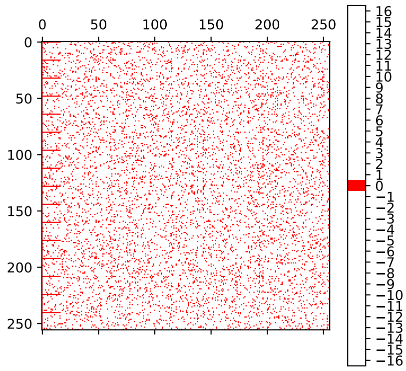
CLEFIA is a 128-bit block cipher supporting key lengths of 128, 192 and 256 bits [137]. The two S-boxes used by CLEFIA employ two different types of $(8,8)$ S-boxes: the first S_0 is based on four 4-bit random S-boxes, while the second S_1 is based on the inverse function over GF(2^8). To achieve better hardware implementation, S_0 is designed by using a combination of 4 smaller linked S-boxes. We analyzed S_0 to find anomalies in $\S^0_{Clefia_{S_0}}$ plot (see Figure 3.2b). There are respectively vertical and horizontal red lines immediately next to the x and y axis, while a complete red square is visible in the upper-left of the matrix plot.



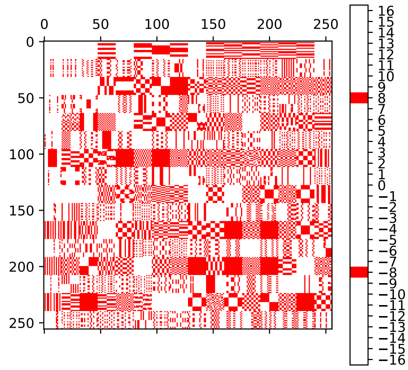
(a) $\S^{-10,10}_{Anubis}$ (b) $\S^0_{Clefia_{S_0}}$

Fig. 3.2 Some spectra channels of Anubis and Clefia S-boxes

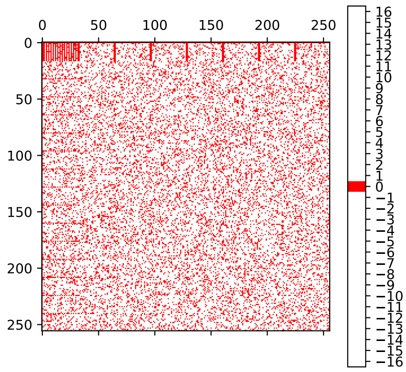
The Cellular Message Encryption Algorithm (CMEA) is a US block cipher that was used for securing mobile phone communications [126]. By analyzing the $\S^0_{CMEA}$ plot we found anomalies immediately next to the y-axis horizontal red lines (see Figure 3.3a). Crypton is a new 128-bit block cipher algorithm proposal for AES. The S-box in the first version (S_0) [93] was further revised and replaced by four S-boxes (S_1, S_2, S_3 and S_4) [94]. In Figure 3.3b the anomalies found in S_0 are depicted, which are clearly visible by restriction $\{-8,8\}$.



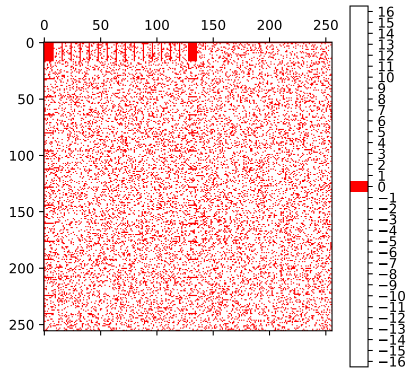
(a) $\S^{0}_{CMEA}$



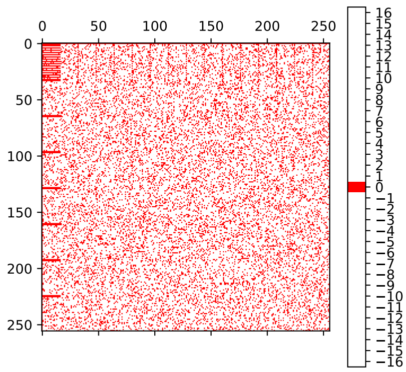
(b) Crypton $\S_0^{-8,8}$



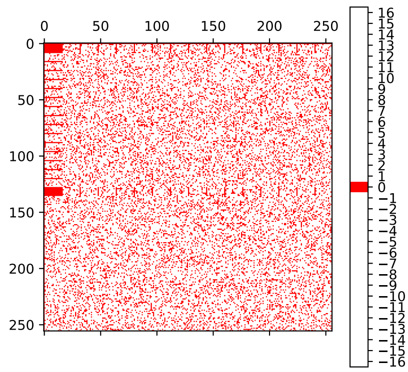
(c) Crypton $\S_1^{0}$



(d) Crypton $\S_2^{0}$



(e) Crypton $\S_3^{0}$

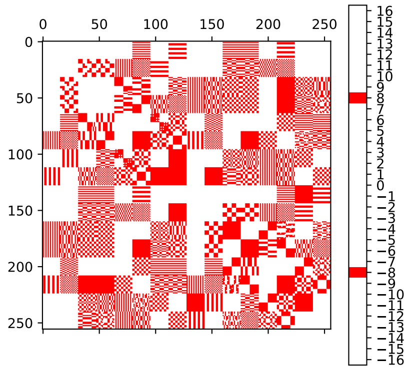


(f) Crypton $\S_4^{0}$

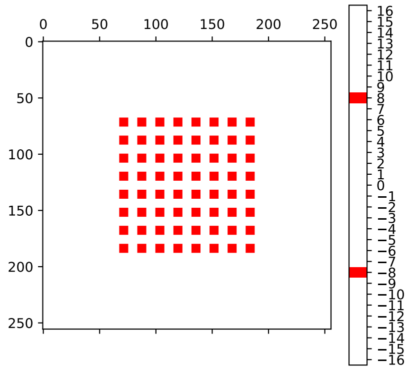
Fig. 3.3 Some spectra channels of CMEA and Crypton S-boxes

All revised Crypton S-boxes possess anomalies in their spectra channel plots. As shown in Figures from 3.3c to 3.3f, the plots are populated with horizontal and vertical lines by the restriction $\{0\}$.

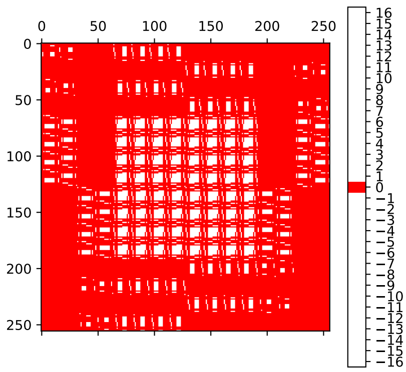
Another NESSIE project block cipher submission is the CS-cipher [144]. By using spectra channel $\S^{0}_{CS}$ a picturesque plot was discovered (see Figure 3.4a).



(a) $\S^{-8,8}_{CS}$



(b) $\S^{-8,8}_{CSS}$

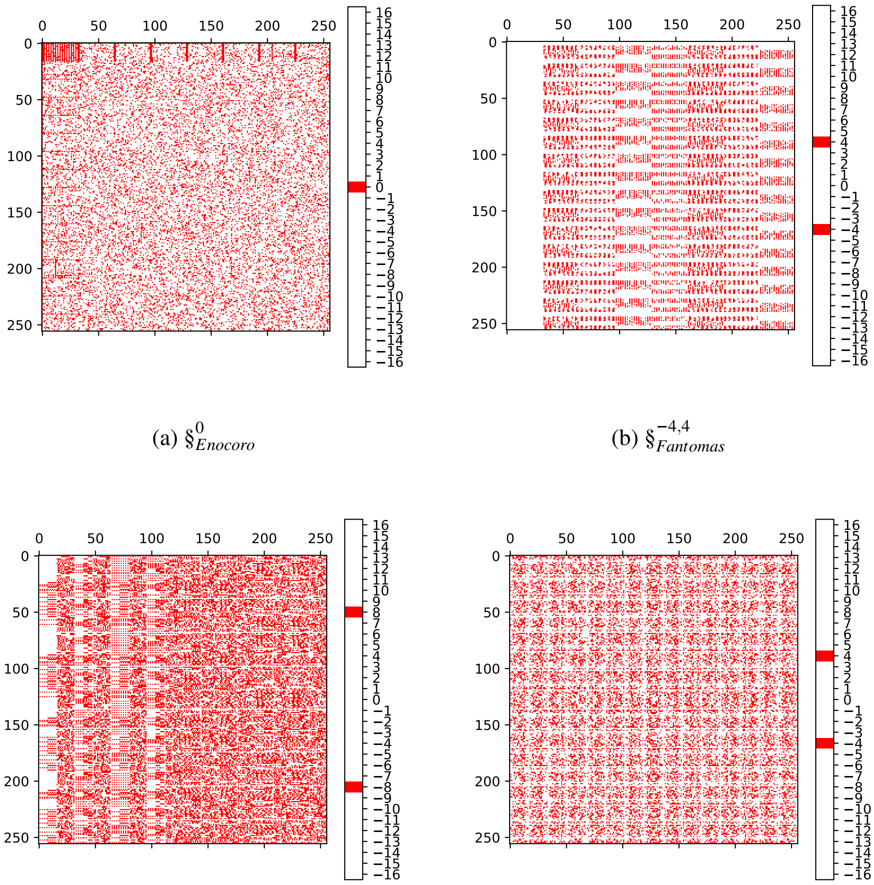


(c) $\S^{0}_{CSS}$

Fig. 3.4 Some spectra channels for CS and CSS S-boxes

The content scrambling system (CSS) [13] is used to encode DVDs. We analyzed the S-box implemented and the results are given in Figures 3.4b and 3.4c.

We further analyzed the S-boxes published in Enocoro [148], Fantomas [69], FLY [83], Fox [82] and Iceberg [143]. Enocoro anomalies are clearly visible in spectra channel with restriction $E = \{0\}$ (see Figure 3.5a). White rectangular covering the lower values on x-axis of Fantomas is detected on spectra channel $\S^{-4,4}_{Fantomas}$ (see Figure 3.5b), while smaller almost perfect rectangulars are visible on the x-axis of FLY spectra channel $\S_{FLY}^{-8,8}$ (see 3.5c). Analyzing Fox by applying spectra channel $\S_{Fox}^{-4,4}$ reveals a grid-alike structure (see 3.5d). The Iceberg S-box is involution (see 3.6a).



(a) $\S_{Enocoro}^{0}$

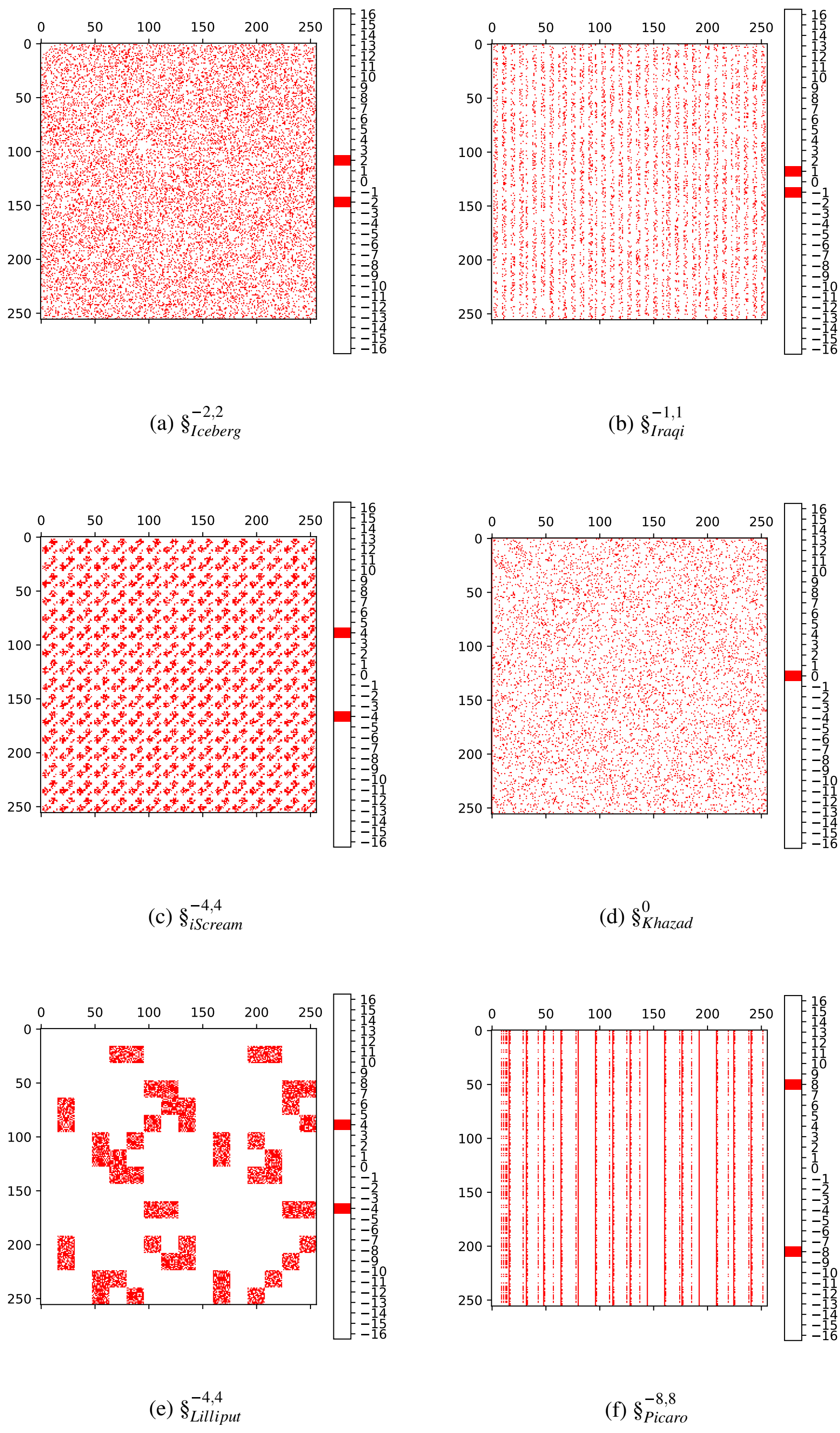
(b) $\S_{Fantomas}^{-4,4}$

(c) $\S_{FLY}^{-8,8}$

(d) $\S_{Fox}^{-4,4}$

Fig. 3.5 Some spectra channels for Enocoro, Fantomas, FLY and Fox S-boxes

Anomalies are found in Iraqi [150], iScream [70], Khazad [10], Lilliput [3] and Picaro [123]. In Figure 3.6b the spectra channel for $\S_{Iraqi}^{-1,1}$ is given. It is distinguishable from pseudo-randomly generated S-box by the striped-alike structure. Furthermore, we can deduce from $\S_{Iraqi}^{-1,1}$ that the Iraqi S-box is not bijective. Fractal-alike structure is revealed in plot $\S_{iScream}^{-4,4}$ (see Figure 3.6c), while involution is observed in $\S_{Khazad}^{0}$ (see Figure 3.6d). Analyzing the Lilliput S-box a Tetris-alike structure is revealed on spectra channel $\S_{Lilliput}^{-4,4}$ (see Figure 3.6e), while fence-alike structure is clearly visible in Picaro S-box on spectra channel $\S_{Picaro}^{-8,8}$ (see Figure 3.6f).



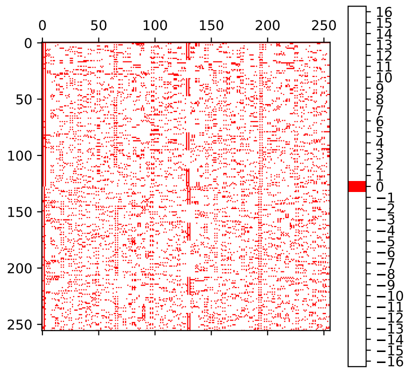
(a) $\S^{-2,2}_{Iceberg}$ (b) $\S^{-1,1}_{Iraqi}$

(c) $\S^{-4,4}_{iScream}$ (d) $\S^{0}_{Khazad}$

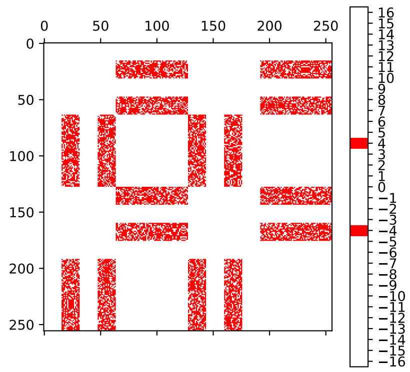
(e) $\S^{-4,4}_{Lilliput}$ (f) $\S^{-8,8}_{Picaro}$

Fig. 3.6 Some spectra channels for Iceberg, Iraqi, iScream, Khazad, Lilliput and Picaro S-boxes

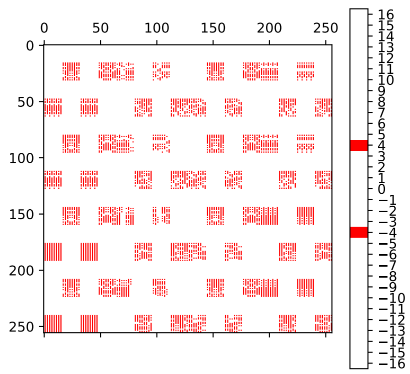
By applying the same method we were able to detect anomalies in Safer [102], Scream [30], SKINNY [5], SNOW 3G [117] and Twofish [134]. In Figure 3.7a the spectra channel $\S^{0}_{Safer}$ is plotted, while in Figure 3.7b $\S^{-4,4}_{Scream}$ a very curious pattern in Scream S-box is revealed. $\S^{-4,4}_{SKINNY}$ is heavily partitioned (see 3.7c), while $\S^{-2,2}_{SNOW3G}$ is completely blank (see Figure 3.7d), which, for example, is completely unusual for a pseudo-randomly generated S-box. In Twofish, two S-boxes π_0 and π_1 are used. Both of them are very similar in terms of their spectra channels (see Figures 3.8a and 3.8b). Furthermore, they are distinguishable from pseudo-randomly generated S-boxes as well (lines on the y-axis are visible).



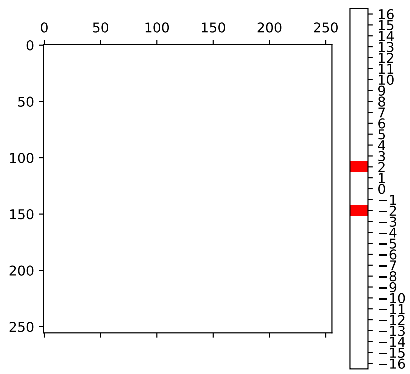
(a) $\S^{0}_{Safer}$



(b) $\S^{-4,4}_{Scream}$

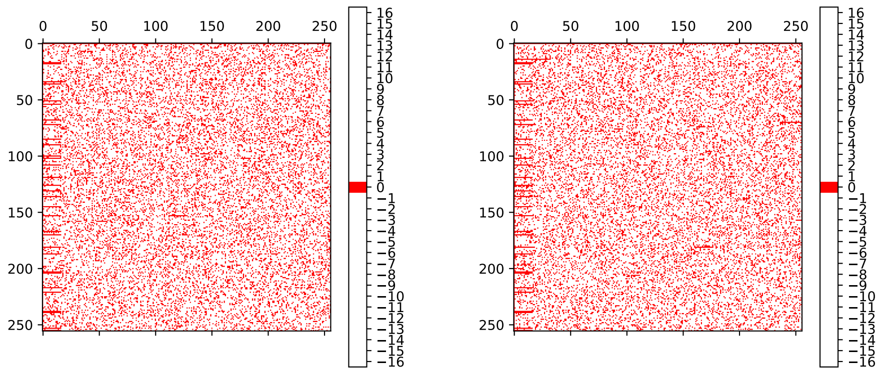


(c) $\S^{-4,4}_{SKINNY}$



(d) $\S^{-2,2}_{SNOW3G}$

Fig. 3.7 Some spectra channels for Safer, Scream, SKINNY and SNOW3G S-boxes



(a) $\S^{0}_{Twofish\pi_0}$ (b) $\S^{0}_{Twofish\pi_1}$

Fig. 3.8 Some spectra channels for Twofish S-boxes

Finally, we analyzed the S-boxes used in Whirlpool [12], Zorro [58] and ZUC [152] by using spectra channels $\S^{0}_{Whirlpool}$ (see Figure 3.9a), $\S^{-2,2}_{Zorro}$ (see Figure 3.9b) and $\S^{-8,8}_{ZUCS_0}$ (see Figure 3.9c).

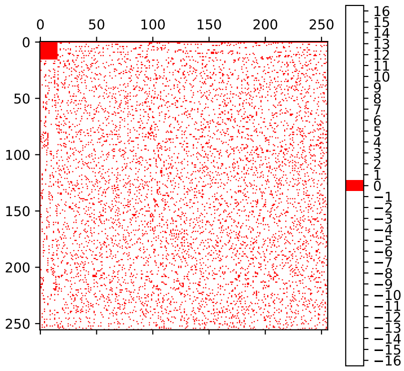
3.3 Automatic spectral analysis of S-box LAT, DDT, XORT, ACT spectras

We could automate the process of anomaly discovery in a given S-box S LAT spectra. Moreover, it could be easily generalized for other spectras of S like the DDT, ACT, and XORT.

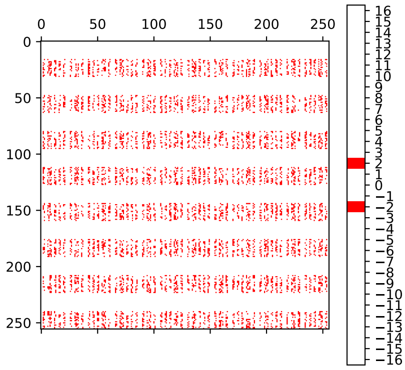
S_{LAT} has 2^n columns. We denote as $S^T_{LAT}[i]$ the i-th column of S_{LAT}. We further denote as $\sigma(S,i,e)$ the total number of occurrences of e and $-e$ in $S^T_{LAT}[i]$, while $\sigma_{ind}(S,e)$ denotes the set of indexes of columns of S_{LAT}, s.t:

$$\forall_{i_1 \neq i_2, i_1, i_2 \in \sigma_{ind}(S,e)} : \sigma(S,i_1,e) \equiv \sigma(S,i_2,e).$$

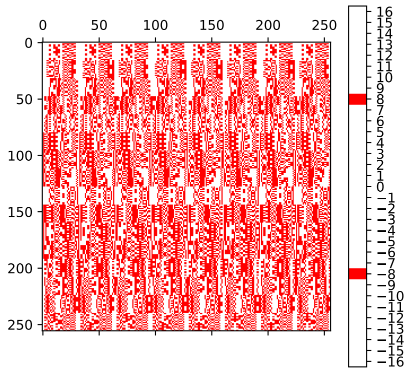
For some reasonable threshold value t and two different values e_1 and e_2, in respect of pseudo-randomly generated S-box, $\sigma_{ind}(S,e_1) \equiv \sigma_{ind}(S,e_2)$, where $\sigma(S,i,e_1) \geq t$ and $\sigma(S,i,e_2) \geq t$, is highly unlikely. During our experiments, we generated more than 10^5 pseudo-random S-boxes. Only in 0.3% of all generated S-boxes a collision was found and always with length 8. Let's denote such collision as $\Gamma(S,t,e_1,e_2,I)$, where I is a set of indexes of columns of S_{LAT}. In Table A.2 (in the Appendix) we give the collisions found



(a) $\S^{0}_{Whirlpool}$



(b) $\S^{-2,2}_{Zorro}$

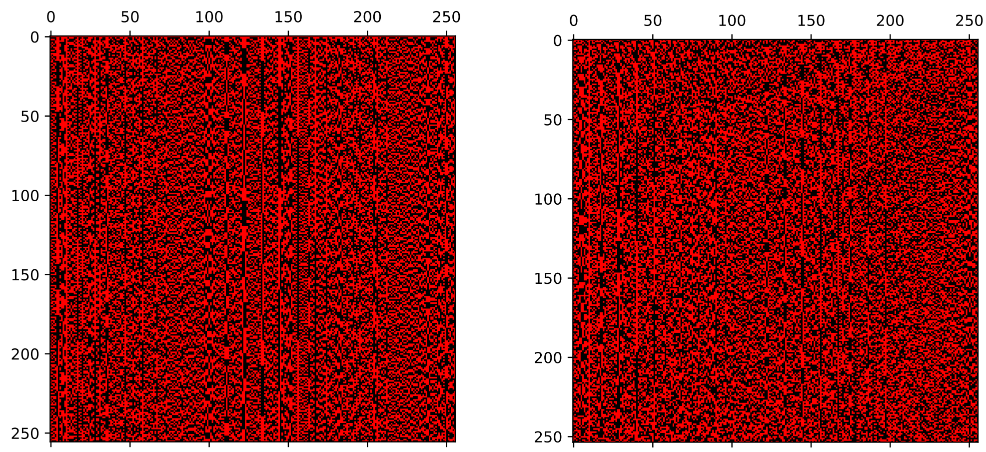


(c) $\S^{-8,8}_{ZUCS_0}$

Fig. 3.9 Some spectra channels for Whirlpool, Zorro and ZUC S_0 S-boxes

in some S-boxes. We have found a collision in the Russian Federation's standardization agency Kuznyechik S-box [68], which was not visible during our spectra channel analysis. The indexes of the collision confirm the observations made in [21]. We can apply the same strategy on LAT rows (instead of columns). In Table A.3 the collisions found in the state standard of Republic of Belarus (BelT) [131] are given. We further analyzed various S-box DDT spectra. The collisions found are given in Table A.4. We have found a collision in π_3 S-box of the new encryption standard of Ukraine Kalyna [116]. By applying the same method to the transformed DDT, more collisions are found in Kalyna and BelT (see Table A.5). We searched for collisions in S-boxes ACT spectra. In Table A.6 the found collisions are given.

We further analyzed the XORT of various S-boxes. A visual interpretation of some XORT relies on the order in which the columns of XORT are populated. In the original definition, the columns are populated in lexicographical order. However, we can tweak that order and populate the XORT by first plotting the n coordinates of a given S-box $S(n,n)$, then all linear combinations of S coordinates with two terms, three terms, and so on, until the last column, which is the XOR of all n coordinates. Such rearrangement makes sense since we group the XORs of the main building blocks of the S-box (the coordinates) into the most significant columns of XORT (the left ones). For example, In Figures 3.10a and 3.10b we give the respectively XORT and rearranged XORT plot of BelT S-box. The lexicographically sorted XOR reveals some vertical lines, which is not unusual for XOR tables of pseudo-randomly generated S-boxes. However, the rearranged XORT reveals some interesting leafs-alike patterns in the upper left section. Furthermore, each consequent column is similar to the previous column when upward-slide.



(a) BelT XORT plot

(b) BelT rearranged XORT plot

Fig. 3.10 Some XORT spectra channels for BelT S-box

Chapter 4

Binary Sequences and Their Autocorrelation

Sequences with low autocorrelation functions are necessary for a variety of signal and information-processing applications. For example, in pulse codes-based compression for radars and sonars, such sequences are used to obtain high resolution. The shifts of sequences with low autocorrelation can be also used for better synchronization purposes or to identify users in multi-user systems. Due to their big practical importance, these sequences have been widely studied and various methods for constructing sequences with small values of autocorrelation are developed.

Let $B = (b_0, b_1, \cdots, b_{n-1})$ be a binary sequence of length $n > 1$, where $b_i \in \{-1, 1\}, 0 \leq i \leq n-1$. The **aperiodic autocorrelation function** (AACF) of B is given by

$$C_u(B) = \sum_{j=0}^{n-u-1} b_j b_{j+u}, \;\; for\; u \in \{0, 1, \cdots, n-1\}.$$

We will note that the AACF is originally defined in the interval

$$\{-n+1, -n+2, \cdots, -2, -1, 0, 1, 2, \cdots, n-1\}.$$

As the AACF is an even function with $C_u(B) = -C_u(B)$, we will consider it for the interval $\{0, 1, \cdots, n-1\}$ only. The $C_0(B)$ is called **mainlobe** and the rest $C_u(B)$ for $u \in \{1, \cdots, n-1\}$ are called **sidelobe** levels. We define the **peak sidelobe level (PSL)** [146] of B as

$$PSL(B) = \max_{0<u<n} |C_u(B)|.$$

The value of the PSL can also be represented in decibels

$$PSL_{db}(B) = 20\log\left(\frac{PSL(B)}{n}\right).$$

Another important measure of an AACF is the **merit factor** (MF), which gives the ratio of the energy of the mainlobe level to the energy of sidelobe levels, i.e.

$$MF(B) = \frac{C_0(B)}{2\sum_{u=1}^{n-1}|C_u(B)|^2}.$$

The binary sequences of low autocorrelation are of special interest and some of the well known such sequences are the Barker codes [9], M-sequences [67], Gold codes [66], Kasami codes [84], Weil sequences [130], Legendre sets [124] and others (see [92][139]). Barker sequences are known to have the best autocorrelation properties, but the longest such sequence is of length 13. M-sequences, Gold codes, and Kasami sequences have ideal periodic autocorrelation functions but have no constraints on the sidelobes of their aperiodic autocorrelation functions. As summarized in [111], during the years a variety of analytical constructions and computer search methods are developed to construct binary sequences with relatively minimal PSL. By an exhaustive search the minimum values of the PSL for $n \leq 40$[96], $n \leq 48$[8], $n = 64$[35], $n \leq 68$[88], $n \leq 74$[90], $n \leq 80$ [91], $n \leq 82$ [89] and $n \leq 84$ [87] are obtained. The best currently known values for PSL for $85 \leq n \leq 105$ are published in [112], and for $n \geq 106$ in [54].

4.1 Efficient Generation of Low Autocorrelation Binary Sequences

In this section an efficient and easy-to-implement heuristic algorithm is suggested and, as an illustration of its effectiveness, it was further utilized for the generation of binary sequences with lengths between 106 and 300. The generated sequences are better, in terms of PSL values than a significant part of those obtained in [54] ones. The algorithm can also be used for the generation of sequences with lengths greater than 300.

Since our goal is to lower the PSL of a given binary sequence, i.e. to lower the value of PSL(B), it makes sense to simultaneously lower the values of each $C_u(B)$, for $u \in$

$\{1, \cdots, n-1\}$. By making this observation, we define the following fitness function:

$$F(B) = \sum_{u=1}^{n-1} |C_u(B)|^P = \sum_{u=1}^{n-1} \left(|\sum_{j=0}^{n-u-1} b_j b_{j+u}| \right)^P,$$

where P is the magnitude of the fitness function, i.e. the higher the magnitude is the higher the fitness function intolerance to large absolute values of $C_u(B)$'s will be. We made experiments with various values of P and the best results were obtained for values in the interval $[3, 5]$. Lower values of P make the fitness function too tolerant to higher absolute values of the PSLs $C_u(B)$, while higher values of P are heavily populating the heuristic topology with local minimums. We have fixed the magnitude P of the fitness function to 4.

Let's denote the i-th position of a binary sequence B of length n as b_i. Flipping the i-th position of B is to interchange b_i with $-b_i$. By the neighborhood of the binary sequence B, denoted by $N(B)$, we define the set of all binary sequences constructed from B by making a single flip in B.

The optimization process takes as input the length of the binary sequence n, the fitness function F, the threshold value t, the two integers h_{min} and h_{max} defining the flipping allowance interval, and the goal G which is the desired final PSL value to be reached.

In the beginning, we generate a random binary sequence B of length n. Then, by searching the neighborhood of B, we look for a better binary sequence, i.e. a binary sequence with a smaller fitness value. If some X out of the neighbors of B has PSL equal to G we output X and quit. If during the search of the neighborhood no better binary sequence is found, we are stuck in some local minimum B'. In order to escape the local minimum we flip h randomly chosen elements of B', where $h \in [h_{min}, h_{max}]$. We will call such try a **quake**. In the case when t consecutive quakes are not sufficient to escape the local minimum, we start the process from the beginning by randomly generating a new binary sequence, i.e. the shotgun hill-climbing approach. The algorithm stops when a binary sequence with the searched value of the PSL is found or when the preliminary defined number of restarts is reached. The pseudo-code of the shotgun hill climbing (SHC) algorithm is given in Algorithm 2.

Algorithm 2 Shotgun Hill Climbing algorithm for PSL optimization

```
procedure HC(n, F, t, h_min, h_max, G)
 BinSeq ← R(n)                                ▷ random binary sequence BinSeq with length n
 thresholdLeft ← t; bestFit ← F(BinSeq)
 globFit ← bestFit; BinSeqCopy ← BinSeq
 repeat
  NB ← N(BinSeq)                              ▷ generation of all neighbors
  FLAG ← True
  for X ∈ NB do
   if PSL(X)==G then Output X and Quit
   end if
   if F(X) < bestFit then                     ▷ a better candidate is found
    bestFit ← F(X)
    BinSeq ← X
    FLAG ← False
   end if
  end for
  if FLAG then
   if bestFit < globFit then                  ▷ a better candidate is found
    globFit ← bestFit; BinSeqCopy ← BinSeq
    thresholdLeft ← t
   else                                       ▷ a better candidate was not found
    thresholdLeft ← thresholdLeft − 1
    if thresholdLeft > 0 then
     BinSeq ← BinSeqCopy
     h ← RI(h_min, h_max)                     ▷ h is random integer ∈ [h_min, h_max]
     FLIP(BinSeq, h)                          ▷ flip h random bits in BinSeq
     bestFit ← F(BinSeq)
    else                                      ▷ the threshold is reached
     BinSeq ← R(n)
     thresholdLeft ← t
     bestFit ← F(BinSeq)
     globFit ← bestFit
     BinSeqCopy ← BinSeq
    end if
   end if
  end if
 until STOP condition reached                 ▷ reaching 10^5 restarts
end procedure
```

The fitness function is the critical resource demanding routine of the algorithm. However, its complexity is comparable to the complexity of the binary sequence PSL calculation itself. The additional negligible overheat is caused by the calculation of the sum of all the P-powered mainlobes.

The parameter h_{min} should be tolerant to possible optimizations involving any small number of flips. Having this in mind and without any restrictions, we choose $h_{min} = 1$. On the other hand, fixing a value of the parameter h_{max} is a trade-off between accuracy and flexibility - smaller values of h_{max} will decrease the algorithm's chances to escape from a given local minimum, while higher values of h_{max} will greatly defocus the climbing routines (for example, hopping from hill A to another hill B, before reaching the local minimum of A). During our experiments, we have fixed the value of h_{max} as $\lceil \sqrt{n} \rceil$, where n is the length of the starting binary sequence.

Another important parameter is the threshold value t. Choosing a small value of t allows us to restart the process of searching a binary sequence with a low PSL value and, instead of losing more time in trying to escape the current local minimum we have stuck at, we reinitialize the searching procedure by starting from the beginning.

We have tried different meta-heuristic strategies like, for example, the simulated annealing method and tabu search. However, it appears that regularly reinitializing the current state of the algorithm, i.e. the core concept of the shotgun hill-climbing method is a more productive strategy to utilize than the aforementioned ones. The initial state does matter and by having a low value of t we increase our chances to reinitialize the algorithm from a highly-competitive candidate. During our experiments, we used a threshold of $t = 10^3$.

We present in Table B.1 the obtained by Algorithm 2 results for binary sequences of lengths from 106 to 300. The second column contains the best-known by us value of the PSL for the corresponding length. In the third column, we present the best value of the PSL obtained by the Algorithm 2 and in the fourth the corresponding sequence with this value of the PSL. The sequences are given in a hexadecimal format where -1's are replaced by zeros and the leading -1's are omitted. For example, the binary sequence of length 11 $B = (-1, -1, 1, 1, -1, 1, 1, -1, 1, 1, 1)$ is given by $1b7$. The decoding procedure requires the length of the binary sequence. The corresponding values of the PSL in decibels and of the merit factor are calculated and given in the fifth and sixth columns respectively.

We improve the PSL values for 95 from the included 195 lengths. The remaining 100 binary sequences have the same values of the PSL as the currently known best ones. Furthermore, all of them are unique and unpublished before. The obtained in this thesis results and the best previously known ones are plotted in Fig. 4.1.

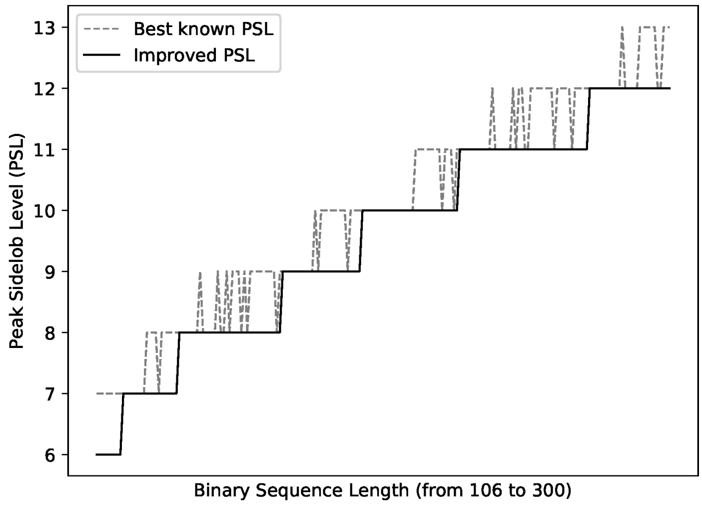


Fig. 4.1 An overview of the shotgun hill climbing algorithm results

The suggested in this section algorithm is highly parallelizable so that a multicore architecture can be fully utilized. It is implemented in Python on a single mid-range computer with an octa-core CPU. During our experiments, the time required to reach a given PSL goal was between a few minutes to several hours. Furthermore, with each instance of the algorithm, we repeatedly reached binary sequences with lower or the same PSL than the state-of-the-art algorithms. The results are published in [48].

4.2 On the Generation of Long Binary Sequences with Record-Breaking PSL Values

M-sequences, Gold codes, and Kasami sequences have ideal periodic autocorrelation functions but have no constraints on the sidelobes of their aperiodic autocorrelation functions, i.e. their PSL value is not pre-determined. The same is true for Legendre sets and Rudin-Shapiro sequences. Furthermore, it is difficult to calculate the growth of the PSL of the aforementioned families of binary sequences. It is conjectured that the PSL values of m-sequences grow like $\mathcal{O}(\sqrt{n})$, making them one of the best methods to straightforwardly construct a binary sequence with near-optimal PSL value. However, as stated in [81]:

The claim that the PSL of m-sequences grows like $\mathcal{O}(\sqrt{n})$, which appears frequently in the radar literature, is concluded to be unproven and not currently supported by data.

As summarized in [111], during the years a variety of analytical constructions and computer search methods are developed to construct binary sequences with relatively minimal PSL. It appears that the current state of art computer search methods, like CAN [73], ITROX [140], MWISL-Diag, MM-PSL [141] or DPM [86], could yield better, or at least not worse PSL values, than the algebraic constructions. However, when the length of the generated by a given heuristic algorithm binary sequences rises, so is the overall time and memory complexity of the routine. As concluded in [109]:

As an indication of the runtime complexity of our EA[1]*, the computing time is 58009 s or 16.1136 h for L=1019. For lengths up to 4096, the computing time required empirically shows a seemingly quadratic growth with L.*

Thus, the main motivation of this section is to create an efficient and lightweight algorithm, in terms of time and memory complexity, to address the heuristic generation of very long binary sequences with near-optimal PSL values.

Let us denote $C_{n-i-1}(B)$ by $\hat{C}_i(B)$. Since this is just a rearrangement of the sidelobes of B, it follows that:

$$B_{PSL} = \max_{0<u<n} |C_u(B)| = \max_{0\le u<n-1} |\hat{C}_u(B)|.$$

We will graphically represent the calculation of values of $\hat{C}_i(B)$ for a binary sequence of length 8 in Figure 4.2. The x-axis indexes represent the elements of $B = (b_0, b_1, \cdots, b_7)$, while the y axes represents the elements of B in reverse order, i.e. $(b_7, b_6, \cdots, b_0)$. Each cell of the graphics corresponds to the product provided by the x and y-axis values. To calculate $\hat{C}_i(B) = \sum_{j=0}^{i} b_j b_{j+n-i-1}$ for some i ($0 \le i \le 7$), we start from the cell with coordinates (b_i, b_7). Then, by decreasing both indexes of the current cell by 1 we jump to the next cell (b_{i-1}, b_6) which will be added to the sum. We continue this process until we reach the cell (b_0, b_{7-i}).

As the value of the mainlobe $\hat{C}_7(B)$ is always 8, we can exclude it from the PSL calculation. Having this in mind, we can define the PSL of the binary sequence B as the diagonal in Figure 4.2 with the highest absolute sum of its elements compared to all other diagonals, excluding the main one.

Let us denote by $\overline{b_i}$ the flipped bit b_i, i.e. $\overline{b_i} = -b_i$ and by $\hat{C}_i(B_j)$ the sidelobe of the binary sequence B_j, obtained from B by flipping the bit on position j.

[1]EA stands for Evolutionary Algorithm

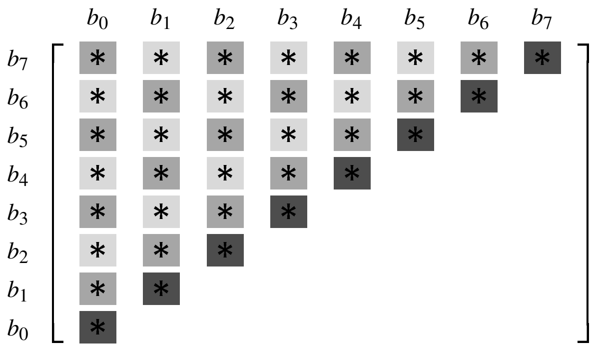


Fig. 4.2 A visual interpretation of the sidelobe calculation process, for a binary sequence with length 8

We can further exploit the relations between the value of the sidelobe $\hat{C}_i(\Psi)$ of a given binary sequence Ψ with length n, and the value of the sidelobe $\hat{C}_i(\Psi_f)$, s.t. the binary sequence Ψ_f is equal to the binary sequence Ψ with the bit on position f flipped. We denote as Ω_Ψ the array of all the consequent sidelobes of Ψ, i.e:

$$\Omega_\Psi = \left[\hat{C}_0(\Psi), \hat{C}_1(\Psi), \cdots, \hat{C}_{n-2}(\Psi)\right]$$

We denote as Ω_{Ψ_f} the array of all the consequent sidelobes of Ψ_f, i.e:

$$\Omega_{\Psi_f} = \left[\hat{C}_0(\Psi_f), \hat{C}_1(\Psi_f), \cdots, \hat{C}_{n-2}(\Psi_f)\right]$$

For convenience, we further denote the i-th element of a given array A as $A[i]$. For example, $\Omega_\Psi[3] = \hat{C}_2(\Psi)$.

The calculation of Ω_Ψ, corresponding to some random binary sequence Ψ, is not linear. The time complexity of the trivial computational approach is $\mathcal{O}(n^2)$ (two nested **for** cycles). However, as shown in Wiener–Khinchin–Einstein theorem [149], the autocorrelation function of a wide-sense-stationary random process has a spectral decomposition given by the power spectrum of that process, we can use one regular and one inverse Fast Fourier Transform (FFT), to achieve a faster way of calculating Ω_Ψ. Despite its time complexity of $\mathcal{O}(n \log n)$, its memory requirement is significantly higher than the trivial computational approach.

By exploiting the observations made in this section, we present an algorithm that can calculate the array Ω_{Ψ_f}, if we hold the array Ω_Ψ in memory, with time and memory complexity of $\mathcal{O}(n)$. The pseudo-code of the algorithm is given in Algorithm 3. The following notations are used:

- $\min_{(x,y)}$: returns x, if $x \le y$; otherwise, returns y.

- $\max_{(x,y)}$: returns x, if $x \geq y$; otherwise, returns y.
- $x -= y$: same as $x = x - y$
- $x *= y$: same as $x = x * y$

Algorithm 3 An algorithm for an in-memory flip inside a binary sequence

procedure FLIP(f, Ψ, Ω_Ψ, n)
  $\delta_{min} \leftarrow \min_{(n-f-1,f)}$
  $\delta_{max} \leftarrow \max_{(n-f,f)}$
  if $f \leq \frac{n-1}{2}$ **then**
    for $q \in [0, \delta_{max} - \delta_{min} - 1)$ **do**
      $\Omega_\Psi[\delta_{min} + q] -= 2\Psi[f]\Psi[n-q-1]$
    end for
  else
    for $q \in [0, \delta_{max} - \delta_{min})$ **do**
      $\Omega_\Psi[\delta_{min} + q] -= 2\Psi[f]\Psi[q]$
    end for
  end if
  if $f \leq \frac{n-1}{2}$ **then**
    for $q \in [0, n - \delta_{max})$ **do**
      $\Omega_\Psi[\delta_{max} + q - 1] -= 2\Psi[f](\Psi[2f-q] + \Psi[q])$
    end for
  else
    for $q \in [0, n - \delta_{max} - 1)$ **do**
      $\Omega_\Psi[\delta_{max} + q] -=$
      $2\Psi[f](\Psi[\delta_{max} - \delta_{min} + q] + \Psi[n-q-1])$
    end for
  end if
  $\Psi[f] *= -1$
end procedure

The procedure introduced in Algorithm 3 performs an in-place memory update of Ω_Ψ when a single bit on position f of Ψ is flipped. Therefore, when the procedure ends, both Ψ and Ω are transformed to Ψ_f and Ω_{Ψ_f}. We will note that the procedure is reversible, i.e. if an in-place memory update of Ω_{Ψ_f} is made, when a single bit on position f of Ψ_f is flipped, both Ψ_f and Ω_{Ψ_f} are transformed back to Ψ and Ω_Ψ.

The basic ingredients of some heuristic algorithms could be summarized as:

- $\mathcal{A}$: **metaheuristic algorithm**, like hill climbing, simulated annealing, tabu search, etc.
- $\mathcal{I}$: **search operator**, which is used to generate the candidates

- $\mathcal{F}$: **fitness function**, which is used to compare the candidates

In the previous section, we have used the shotgun hill-climbing as $\mathcal{A}$, a neighborhood search as $\mathcal{I}$, and the following fitness function as $\mathcal{F}$:

$$F(B) = \sum_{u=1}^{n-1} |C_u(B)|^4 = \sum_{u=1}^{n-1} \left(|\sum_{j=0}^{n-u-1} b_j b_{j+u}| \right)^4,$$

where B is a binary sequence with length n. However, using shotgun hill-climbing metaheuristic algorithm for finding very long binary sequences with low PSL is not time efficient because the number of hops required to reach some local optimum grows exponentially when the length of the binary sequence increases.

Using a neighborhood search to consequently pick the best candidate among all neighbors could be beneficial in finding LBS with low PSL. However, in the aspect of very long binary sequences, this search strategy is extremely slow. For example, in the case of a binary sequence with length 2^{16}, and $\mathcal{I}$ equivalent to a single flip, in each optimization step we need to fitness all the 2^{16} neighbors of the current state S and to pick the one with the best score yielded by $\mathcal{F}$. This observation is still true, even if all the neighbors of S have better scores.

To overcome the disadvantages mentioned above, we choose the following strategy:

- $\mathcal{A}$: stochastic hill climbing metaheuristic algorithm. We visit a random neighbor of the current state S and accept it if it is a better candidate than S. Otherwise, we pick another neighbor of S and repeat the process.

- $\mathcal{I}$: we choose a single flip as the search operator, so we can exploit the memory and time efficiency of Algorithm 3.

- $\mathcal{F}$: since $\hat{C}(B)$s are rearrangements of the sidelobes of B, we can use the same fitness function $F(B)$ as in [48], i.e:

$$F(B) = \sum_{u=0}^{n-2} |\hat{C}_u(B)|^4 = \sum_{u=0}^{n-2} \hat{C}_u(B)^4$$

We need to further address the strategy described in $\mathcal{A}$ of picking the next candidate, or neighbor, of S. Let us consider an approach of consistently probing x pseudo-randomly chosen neighbors. In case a better candidate is found, we accept it; otherwise, we try again, until we have accumulated a total number of t consequent fails. Then, we announce that we have reached a local optimum. This model can be described by the Bernoulli distribution.

The probability to achieve exactly r successes in N trials is equal to:

$$P(X = r) = \binom{N}{r} p^r q^{N-r},$$

where p and q are the probabilities of success and failure respectively, i.e. $q = 1 - p$. We can easily calculate $P(X = 0)$:

$$P(X = 0) = \binom{N}{0} p^0 q^{N-0} = q^N = (1 - p)^N$$

We further calculate $P(X \geq 1)$:

$$P(X \geq 1) = 1 - P(X = 0) = 1 - (1 - p)^N$$

Thus, relying solely on pseudorandom choices of neighbors is not efficient and there is always a chance to miss the better candidate. We can increase the probability of finding the eventual better candidate, but that significantly overhead the optimization process. Missing a better candidate is undesirable behavior of the optimization process, especially when we are dealing with very long binary sequences.

The number of neighbors of a binary sequence B with length n is n. Let us denote those neighbors as $i_1, i_2, \cdots, i_n$, where the j-th neighbor i_j is equal to B with flipped bit on position j. We suggest the following simple search strategy::

1. we pick a pseudorandomly generated neighbor i_r
2. we consequently try, for all $x \in [1, n-1]$, the neighbors $i_{(r+x) \mod n}$

We want to emphasize the extreme situation when the local optimum is already reached, i.e. $k = 0$. The suggested search strategy will detect that in exactly n steps, which is an optimal scenario. Furthermore and more importantly, we never miss a better candidate, if any, and we keep the non-deterministic nature of the search routine at the same time.

We suggest Algorithm 4 for finding very long binary sequences with low PSL which is based on the above described $(\mathcal{A}, \mathcal{I}, \mathcal{F})$. The following notations and functions are used in the pseudo-code:

- Ψ is a random (initial) binary sequence.
- $x, y \leftarrow a, b$ is equivalent to the statements $x = a$ and $y = b$.
- $R(n)$: a function, which generates a pseudo-random integer number $\in [0, n)$.

- $Q(x, B, \Omega_B)$: a function, which makes x flips at random bit positions in B. We pass Ω_B as an argument, so we can use the in-place memory function $Flip$. We apply this function to escape the local minimum when we are stuck in such.
- **beacon**: we further implant a beacon in the cost function F, so we can simultaneously calculate the PSL of the given binary sequence. Such an approach adds a negligible overhead, if any, to the cost function routine.

Algorithm 4 An algorithm for long binary sequences PSL optimization

```
BestCost, Cost ← F(Ω_Ψ), 0
isGImpr, isLImpr ← True, False
while true do
  if isGImpr then
    r ← R(n)
    for i ∈ [0,n) do
      Flip((r+i)%n, Ψ, Ω_Ψ, n)
      Cost ← F(Ω_Ψ)                          ▷ * the beacon is here *
      if BestCost > Cost then
        BestCost ← Cost
        isLImpr ← True
        break
      else
        Flip((r+i)%n, Ψ, Ω_Ψ, n)
      end if
    end for
    if isLImpr then
      isGImpr, isLImpr ← True, False
      continue
    else
      isGImpr ← False
    end if
  else
    r ← R(4)
    Q(1+r, Ψ, Ω_Ψ)
    isGImpr, isLImpr ← True, False
  end if
end while
```

We emphasize that the complexity of Algorithm 4 mainly depends on the complexity of Algorithm 3, because in each iteration during the optimization process, Algorithm 3 is called twice, in case the new candidate is worse than the current one, and once, if the new candidate is better. The in-memory flip function applied in Algorithm 3 passes only once through Ω_Ψ array, without creating any memory overheads, to reach time and memory complexities of $\mathcal{O}(n)$. The same observation is true for the simple cost function F - it passes only once through Ω_Ψ to sum all quadrupled values of its elements. The function Q is a random number of calls of F (between 1 and 4). The remaining part of Algorithm 4 consists of simple automaton, which rule the continuous optimization process. Therefore, both time and memory complexities of Algorithm 4 are $\mathcal{O}(n)$.

We have implemented Algorithm 4 by using the C language and a mid-range computer station. Given the linear time and memory complexity of the algorithm, we were able to repeatedly generate binary sequences with record-breaking PSL values for less than a second. As stated in [109], the time required to reach a PSL value 26, for a binary sequence with length 1019, is 58009 seconds or 16.1136 hours. For comparison, by using Algorithm 4, we reach this value for less than a second.

We present the results achieved by Algorithm 4, for binary sequences with lengths x^2 for $x \in [18, 44]$, compared with the currently known state of art algorithms found in the literature, like CAN [73], ITROX [140], MWISL-Diag, MM-PSL [141], DPM [86], 1bCAN [95]. We will refer to this collection of algorithms as collection **A**. We want to emphasize, that the differences between the proposed algorithm with algorithms from collection A are manifold. For example, we do not use converging functions, mini regular or quadratic optimization problems, or floating-based arithmetic. Furthermore, the provided algorithm does not suffer from a unique navigation trace through the sequence search space. The experiments were based on 12 instances of each algorithm (each ran to a distinct thread of the processor). Furthermore, the lifetime of our algorithm is restricted to 1 minute. As shown in Figure 4.3, we significantly outperform the best results achieved by state of art algorithms. In fact, for some of the lengths, less than a second was needed to reach a record-breaking PSL.

In contrast to some other state-of-the-art algorithms, the computing complexity of the algorithm presented in this work does not grow quadratically. Maybe this is the reason for the lack of published results for binary sequences of lengths greater than 2^{12}. Nevertheless, the results with which we can further compare are *m*-sequences. However, such sequences exists only for lengths $2^n - 1$, $n \geq 1, n \in N$.

In Table 4.11 we present the best PSL values of binary m-sequences with length n (with or without rotation), yielded by some primitive polynomial of degree n over $GF(2)$ from [52] denoted by $\mathbb{M}_n^{\mathbb{F}}$ and the binary sequences generated by Algorithm 4 denoted by $\mathbb{A}_n$ for lengths $2^n - 1$ and $13 \leq n \leq 17$. As it can be seen from Table 4.11, our results significantly outperform the best results achieved by *m*-sequences. The results are published in [49].

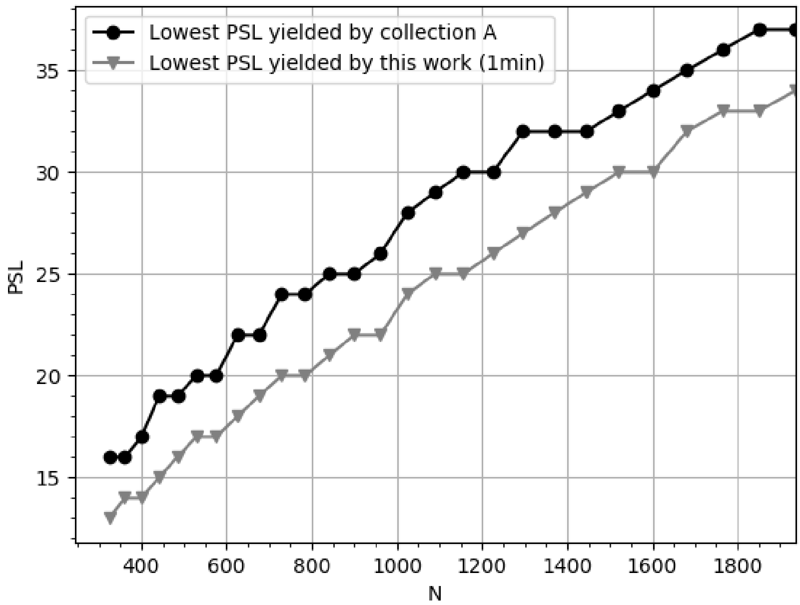


Fig. 4.3 Comparison to other state of the art algorithms known in literature

Table 4.1 Reached PSL values compared to known results from m-sequences exhaustive search

n	$2^n - 1$	$\mathbb{M}_n^{\mathbb{F}}$	$\mathbb{A}_n$
13	8191	85	77
14	16383	125	115
15	32767	175	171
16	65535	258	254
17	131071	363	360

4.3 Hybrid Constructions of Binary Sequences with Low Autocorrelation Sidelobes

An m-sequence $M = (x_0, x_1, \cdots, x_{2^m-2})$ of length $2^m - 1$ is defined by:

$$x_i = (-1)^{Tr(\beta\alpha^i)}, \text{ for } 0 \leq i < 2^m - 1,$$

where α is a primitive element of the field $\mathbb{F}_{2^m}$, $\beta \in \mathbb{F}_{2^m}$, and Tr is denoting the trace function from $\mathbb{F}_{2^m}$ to $\mathbb{F}_2$.

Given an odd prime p, a Legendre sequence L with length p is defined by:

$$L_i = \begin{cases} 1, \text{ if } i \text{ is a quadratic residue mod } p \\ -1, \text{ otherwise.} \end{cases}$$

We denote as $B \leftarrow \rho$ the binary sequence obtained from B, by left-rotating it ρ times. By definition, $B \leftarrow |B| \equiv B$. Furthermore, if b_i is the element of B on position i, we will denote as $b_i^{\leftarrow \rho}$ the element of $B \leftarrow \rho$ on position i.

A comparison, in terms of algorithm efficiency (the ratio of the beneficial work performed by the algorithm to the total energy invested) and actual effectiveness (the quality of the achieved results), was made. The best results were achieved by the SHC algorithm, regarding the binary sequences with lengths less than 300, and HC, for all the remaining lengths. However, the approximated binary sequence's length, from which HC starts outperforming SHC, is fuzzy and yet to be determined.

In Table 4.2 a comparison between the most significant components of SHC and HC was made. In summary, both heuristic algorithms are not deterministic, i.e. starting from two identical states rarely results in two identical ending states. The search operator used in both SHC and HC is the single flip operator. Thus, each modification is a simple composition of single flips. One major difference between the two algorithms is their complexity. Indeed, in HC the time complexity of the flip operation is linear, which is a significant advantage compared to the quadratic one to be found in SHC. Another major difference between HC and SHC is the probability of missing (failing to detect) a better binary sequence, which is just 1 flip away from the current position.

As observed in Section 4.2 of the thesis or in our work [49], the PSL-optimization process of very long binary sequences is a time-consuming routine, despite the algorithm's linear time and memory complexities. Thus, HC avoids restarts, i.e. re-initializing the starting state with a pseudo-random binary sequence. However, re-initialization appears to be significantly beneficial when dealing with PSL optimization of binary sequences with relatively small lengths, such as the SHC algorithm.

By considering the observations made above, we have revisited the SHC algorithm:

- The quadratic flip operator was interchanged with the linear flip operator.
- The probing strategy (searching for better candidates) was interchanged with the more efficient probing strategy introduced in HC.

The complete pseudo-code of the kernel of the revisited SHC algorithm is summarized in Algorithm 5. For brevity, the following notations were used:

Table 4.2 A comparison between SHC and HC

	SHC	HC
Deterministic	No	No
Search Operator	Flip	Flip
Complexity	$O(n^2)$	$O(n)$
Fitness Function	x^4	x^4
Restarts	Yes	No
Missing Probability	> 0	$= 0$

- n - the binary sequence's length
- $\mathbb{T}$ - the threshold value of the instance
- F - a fixed fitness function
- V, V^* - respectively the current best and the overall best fitness value
- c - the counter. The algorithm quits if the counter c reaches the threshold $\mathbb{T}$
- $\mathbb{Z}_n^+$ - the set of all positive integer numbers strictly less than n
- $\mathbb{L}$, $\mathbb{G}$ - binary variables: $\mathbb{L}$ (local) is activated if V is improved, while $\mathbb{G}$ (global) is activated if V^* is improved
- $\mathbb{B}^n$ - the set of all n-dimensional binary sequences with elements from $\{-1, 1\}$
- Q - the quaking function as defined in Section 4.2. For example, if the input triplet of Q is x, L, SL, the function flips x random bits in L, and at the same time, in-memory updating the sidelobe array SL

Considering the significant changes made in the SHC algorithm, the fitness function parameters are carefully analyzed, re-evaluated, and updated. Given a binary sequence Ψ, both algorithms (SHC and HC) are sharing the same fitness function F, s.t:

$$F(\Psi) = \sum_{x \in \Omega_\Psi} |x|^4 = \sum_{x \in \Omega_\Psi} x^4$$

During our previous experiments, we reached to the conclusion that interchanging the power 4 with larger or smaller value, is respectively too intolerant or too tolerant to the largest elements in Ω_Ψ. However, since significant changes to the kernel of SHC were made, this

Algorithm 5 The Shotgun Hill Climbing revisited kernel

```
procedure SHC(n, 𝕋)
  pick Ψ ∈ 𝔹^n
  V*, V, 𝔾, 𝕃, c ← F(Ω_Ψ), 0, True, False, 0
  while c < 𝕋 do
    c += 1
    if 𝔾 then
      pick r ∈ ℤ_n^+
      for i ∈ [0, n) do
        flip((r + i) %n, Ψ, Ω_Ψ)
        if V* > F(Ω_Ψ) then
          V*, 𝕃 ← F(Ω_Ψ), True
          break
        else
          flip((r + i) %n, Ψ, Ω_Ψ)
        end if
      end for
      if 𝕃 then
        𝔾, 𝕃 ← True, False
        continue
      else
        𝔾 ← False
      end if
    else
      pick r ∈ ℤ_4^+
      Q(1+r, Ψ, Ω_Ψ)
      𝔾, 𝕃 ← True, False
    end if
  end while
end procedure
```

observation is to be re-evaluated by a series of experiments. More precisely, given a fixed threshold $\mathbb{T}$, and the fitness function $\sum_{x\in\Omega_\Psi}|x|^\alpha$, a comparison between the efficiency of different α values is measured.

In Table 4.3 the results regarding binary sequences with length 100 are given. Each row of the table corresponds to a different experiment. For a more informative measurement of the overall efficiency of the experiments, another variable $V^\triangledown$ was introduced. It measures the median value of all the best values V^*. More formally, if t_i denotes the thread i of a given experiment $\mathbb{E}$ with $\mathbb{R}$ restarts, and if the best results achieved by t_i is denoted as V_i^*, then

$$V^\triangledown = \frac{\sum_{i\in\mathbb{E}} V_i^*}{\mathbb{R}}$$

At first, the numerical experiments suggest $\alpha = 3$ as a near-optimal value for achieving the best results. Indeed, given a binary sequence with length 100, and $(\alpha,\mathbb{R},\mathbb{T}) = (3,10^2,10^4)$, the value of $V^\triangledown$ is smaller compared to the other experiments' values. This observation is more clearly visible throughout the experiments with binary sequences having length 256 summarized in Table 4.4 and binary sequences with length 500 (see Table 4.5 and the triplet $(\alpha,\mathbb{R},\mathbb{T}) = (3,10^2,10^4)$ with $V^\triangledown = 11.51$). However, this tendency of $\alpha = 3$ supremacy over integer values of α is not observable throughout larger values of n. As summarized in Table 4.6, the triplet $(\alpha,\mathbb{R},\mathbb{T}) = (4,10^2,10^4)$ yields better characteristics than $(\alpha,\mathbb{R},\mathbb{T}) = (3,10^2,10^4)$. In fact, the quality of the binary sequences yielded by the triplet $(\alpha,\mathbb{R},\mathbb{T}) = (4,10^2,10^3)$, having $V^\triangledown$ equal to 24.81, is almost the same as those binary sequences generated by the triplet $(\alpha,\mathbb{R},\mathbb{T}) = (3,10^2,10^4)$ with $V^\triangledown = 24.98$. Since the first threshold value (10^3) is ten times smaller than the second one (10^4), and given the negligible difference of the binary sequences' quality (0.17), this correlation is particularly beneficial and could be further exploited to reduce the overall time needed for the binary sequences optimization routines.

During the final two experiments, considering the bigger sizes of the binary sequences, the threshold value is fixed at 10^3. However, the data gathered throughout the previous experiments suggested that if we have a triplet $(n,\mathbb{R},\mathbb{T}_1)$ measured with $V_1^\triangledown$, then, given $\mathbb{T}_1 \geq 10^3$ and some threshold value $\mathbb{T}_2 >> \mathbb{T}_1$, such that the triplet $(n,\mathbb{R},\mathbb{T}_2)$ is measured with $V_2^\triangledown$, then $V_2^\triangledown < V_1^\triangledown$.

In Tables 4.7 and 4.8, triplets of the form $(\alpha,10^2,10^3)$ were analyzed, corresponding to binary sequences with respective lengths of 2048 and 4096. It appears that the longer the binary sequence is (n), the larger the aggression of the optimization routine should be (α). Indeed, in the case of $n = 2048$, the best value of $V^\triangledown = 36.74$ is calculated by using $\alpha = 5$,

Table 4.3 Efficiency and comparison of various triplets $(\alpha, \mathbb{T}, 100)$

n	α	$\mathbb{R}$	$\mathbb{T}$	V^*	$V^{\triangledown}$
100	1	10^2	10^3	7	7.63
100	1	10^2	10^4	6	7.00
100	2	10^2	10^3	6	6.95
100	2	10^2	10^4	6	6.72
100	3	10^2	10^3	6	6.94
100	3	10^2	10^4	6	6.70
100	4	10^2	10^3	7	7.00
100	4	10^2	10^4	6	6.94
100	5	10^2	10^3	7	7.00
100	5	10^2	10^4	6	6.95
100	6	10^2	10^3	7	7.10
100	6	10^2	10^4	7	7.00
100	7	10^2	10^3	7	7.23
100	8	10^2	10^3	8	8.26

Table 4.4 Efficiency and comparison of various triplets $(\alpha, \mathbb{T}, 256)$

n	α	$\mathbb{R}$	$\mathbb{T}$	V^*	$V^{\triangledown}$
256	1	10^2	10^3	13	14.66
256	1	10^2	10^4	13	13.94
256	2	10^2	10^3	11	11.98
256	2	10^2	10^4	11	11.72
256	3	10^2	10^3	11	11.92
256	3	10^2	10^4	11	11.51
256	4	10^2	10^3	11	11.99
256	4	10^2	10^4	11	11.84
256	5	10^2	10^3	12	12.22

Table 4.5 Efficiency and comparison of various triplets $(\alpha, \mathbb{T}, 500)$

n	α	$\mathbb{R}$	$\mathbb{T}$	V^*	V^∇
500	1	10^2	10^3	21	23.19
500	1	10^2	10^4	21	22.10
500	2	10^2	10^3	17	17.83
500	2	10^2	10^4	16	17.04
500	3	10^2	10^3	16	16.94
500	3	10^2	10^4	16	16.61
500	4	10^2	10^3	16	17.04
500	4	10^2	10^4	16	16.89

Table 4.6 Efficiency and comparison of various triplets $(\alpha, \mathbb{T}, 1024)$

n	α	$\mathbb{R}$	$\mathbb{T}$	V^*	V^∇
1024	1	10^2	10^3	34	38.50
1024	1	10^2	10^4	34	35.96
1024	2	10^2	10^3	27	28.27
1024	2	10^2	10^4	26	27.12
1024	3	10^2	10^3	24	25.43
1024	3	10^2	10^4	24	24.81
1024	4	10^2	10^3	24	24.98
1024	4	10^2	10^4	24	24.16
1024	5	10^2	10^3	25	25.32
1024	6	10^2	10^3	25	25.98

Table 4.7 Efficiency and comparison of various triplets $(\alpha, \mathbb{T}, 2048)$

n	α	$\mathbb{R}$	$\mathbb{T}$	V^*	V^∇
2048	1	10^2	10^3	58	65.64
2048	2	10^2	10^3	41	44.32
2048	3	10^2	10^3	37	38.27
2048	4	10^2	10^3	36	36.99
2048	5	10^2	10^3	36	36.74
2048	6	10^2	10^3	36	36.91

Table 4.8 Efficiency and comparison of various triplets $(\alpha, \mathbb{T}, 4096)$

n	α	$\mathbb{R}$	$\mathbb{T}$	V^*	V^∇
4096	1	10^2	10^3	99	110.11
4096	2	10^2	10^3	64	68.48
4096	3	10^2	10^3	55	57.47
4096	4	10^2	10^3	53	54.91
4096	5	10^2	10^3	53	54.17
4096	6	10^2	10^3	53	54.16
4096	7	10^2	10^3	53	54.28

while in the case of binary sequences with lengths $n = 4096$, the best value of $V^\nabla = 54.16$ is yielded by using $\alpha = 6$.

As previously discussed, binary sequences with lengths up to 84 and PSL-optimal values have been already discovered by using various exhaustive search strategies. This data is particularly beneficial for measuring the efficiency of a given PSL-optimizing algorithm. In other words, given a search space with binary sequences with some fixed length $n \leq 84$, and some PSL-optimizing algorithm $\mathbb{A}$ with a reasonable threshold value, the best results achieved by $\mathbb{A}$ could be compared with the already known optimal PSL values.

During our experiments, we used a single general-purpose computer with a 6-cored central processing unit architecture, capable of running 12 threads simultaneously. Surprisingly, by using the SHC revisited kernel, as well as a fixed value of $\alpha = 2$, we were able to reach binary sequences with optimal PSL values for each length in $[1, 82]$. Given the linear time and memory complexities of the algorithm, for the majority of those lengths, the PSL-optimal binary sequences were reached for less than a minute. However, for some border cases, the needed time was a few hours. The best results yielded by our experiments are summarized in

Table B.2. A remark should be made, that we have included just one PSL-optimal binary sequence for a given length. However, for almost each fixed length, the algorithm was able to find more than one binary sequence having an optimal PSL value. The binary sequences are given in a hexadecimal format, by omitting the leading zeroes. In the last column of Table B.2, beside the corresponding optimal PSL value of the hexadecimal binary sequence given in column 2, the symbol ⧖ was used to illustrate some approximation of the time needed for Algorithm 2 to reach a PSL-optimal binary sequence:

- ⧖ ≈ minute
- ⧖⧖ ≈ hour
- ⧖ ⧖ ⧖ ≈ day

For all other cases, the algorithm was able to reach the optimal PSL for less than a minute, and in some cases, for less than a second.

The results achieved throughout the experiments described in this section demonstrated the efficiency of Algorithm 5. Thus, we have further launched the algorithm on binary sequences with lengths up to 300. The results are given in Table B.3. The binary sequences with record-breaking PSL values are further highlighted with the symbol ▼ (the black triangle pointing down). Almost all of the results known in the literature were improved. More precisely, we have improved 179 out of 195 cases. Curiously, for some lengths, we have even revealed binary sequences with record-breaking PSL values, having a distance of 2 to the previously known PSL record value. We will mark those improvements with a double black triangle symbol. An example of such length is 229.

In [36], the best results achieved by the D-Wave 2 quantum computer for binary sequences with length 128 is PSL 8, while Algorithm 5 could reach PSL 6. For longer lengths, for example, binary sequences with lengths 256, the best PSL achieved by the D-Wave 2 quantum computer was 12, while during our experiments we reached PSL values of 10. We reached PSL values of 10 for binary sequences up to 271. For completeness, since the D-Wave 2 quantum computer is tested on binary sequences with length 426, we have further launched Algorithm 2 on the same length. Surprisingly, the algorithm was able to find binary sequences with PSL values of 17 (the best value achieved by the quantum computer) for less than a second. It reached PSL values of 16, and even 15, for less than a second as well. However, PSL value of 14 (see Table B.4) was noticeable harder to reach (199 seconds). During this optimization routine, and driven by the results provided in Table 4.5 (since 500 is close to 426), we have updated the α value to 3.

Recently, in [37] a multi-thread evolutionary search algorithm was proposed. By using Algorithm 5 we were able to improve almost all of the best PSL values from the aforementioned paper - usually for less than a second. For example, the best PSL value for binary sequences with length 3000 achieved in [37] is 51. We have launched Algorithm 2 on binary sequences with the same length. It should be emphasized (see Tables 4.7 and 4.8), that the α parameter should be increased to 6. Record-breaking PSL values of 44 and 43 were reached for respectively 111 and 371 seconds. In Table B.4 an example of such binary sequence (2nd row) is given. The last column of the table provides a more quantitative measure of the record: ▼x denotes that the corresponding binary sequences possess a record-breaking PSL equal to $P - x$, where P was the previously known record.

The reasoning behind announcing one binary sequence as long, or short, is ambiguous. Measuring the largeness of a given binary sequence is probably more related to the capabilities of the used algorithm than the actual length itself. From a practical point of view, some algorithms, or their implementations, would not even start the optimization (or construction) process, since their computational capabilities (or hardware restrictions) would not be able to process the desired length. For example, as discussed in [36], the usage of a 512-qubit D-Wave 2 quantum computer limits the code length that can be handled, to at most 426, due to a combination of overhead operations and qubits unavailability. Moreover, it was estimated that a 2048-qubit D-Wave computer could handle binary sequences with lengths up to 2000. Hence, the exact fixed value differentiating short from long binary sequences is still unclear.

In Table 4.9 some detailed time measurements of binary sequences with lengths $2^g - 1$, for $g \in N, g \in [13, 17]$ are given. The binary sequences are specially chosen to exactly match the lengths of the well-known m-sequences, generated by some primitive polynomial of degree g over $GF(2)$ denoted by $\mathbb{M}$ (see [52]) and the binary sequences generated by Algorithm 5 denoted by $\mathbb{A}$. The α parameter was fixed to 4. The last column ($\mathbb{A}$) denotes the time needed for Algorithm 5 to reach the corresponding PSL (s, m, h, and D denote respectively seconds, minutes, hours and days). Evidently, the longer the m-sequence, the harder for Algorithm 5 to find binary sequences with better PSL values is. For example, Algorithm 5 required approximately 3 days to find a binary sequence of length 131071 with lower PSL than the optimal m-sequence having the same size. Given a PSL-optimizing algorithm $\mathscr{A}$ we will reference the length n of a binary sequence as $\mathscr{A}$-long if the expected time from $\mathscr{A}$, starting from a pseudo-randomly generated binary sequence with length n, to reach a binary sequence with PSL p, s.t. $p \leq \lfloor \sqrt{n} \rfloor$, and by using single general purpose processor, is more than 1 day. Otherwise, we will reference it as $\mathscr{A}$-short. Throughout the radar literature statements that the asymptotic PSL of m-sequences grows no faster than order $\sqrt{n}$, were frequently made. However, as shown in [81], this assumption was not supported by theory or by data. Nevertheless, it appears that the PSL-optimal m-sequences are very

Table 4.9 Time required to find better PSL values compared to known results from m-sequences exhaustive search

g	$n = 2^g - 1$	$\mathbb{M}_n^{\mathbb{F}}$ (PSL)	$\mathbb{A}$ (PSL)	$\mathbb{T}$
13	8191	85	84	19s
13	8191	85	83	23s
13	8191	85	82	28s
13	8191	85	81	1.5m
13	8191	85	80	6.95m
13	8191	85	79	4.37h
13	8191	85	78	8.04h
13	8191	85	77	13.24h
14	16383	125	124	44s
14	16383	125	123	1.16m
14	16383	125	122	4.70m
14	16383	125	121	4.72m
14	16383	125	120	5.30m
14	16383	125	119	14.15m
14	16383	125	118	20.26m
14	16383	125	117	20.37m
14	16383	125	116	1.49h
14	16383	125	115	1.49h
15	32767	175	174	47.27m
15	32767	175	173	47.28m
15	32767	175	172	3.09h
15	32767	175	171	3.10h
16	65535	258	257	9.42m
16	65535	258	256	22.79m
16	65535	258	255	22.80m
16	65535	258	254	22.81m
17	131071	363	362	2.95D
17	131071	363	361	2.95D
17	131071	363	360	2.95D

close to $\sqrt{n}$ (see [44]). Thus, the threshold value of $\lfloor\sqrt{n}\rfloor$ is based on the expectation that the optimal PSL value for a given binary sequences with length n is less than $\lceil\sqrt{n}\rceil$.

From now on, we denote Algorithm 5 as $\mathscr{A}$ with fixed α value to 4 if not specified otherwise. During our experiments and by using $\mathscr{A}$, we have reached the conclusion that all binary sequences with lengths n, s.t. $n > 10^5$ are $\mathscr{A}$-long. In this section, we have investigated some hybrid constructions which could be applied in those cases when the binary sequences are $\mathscr{A}$-long.

4.3.1 Using $\mathscr{A}$ as an m-sequences extension

The following procedure is proposed:

- Choose a primitive polynomial f over F_{2^m}
- Fix an initial element a over F_{2^m}
- Convert f to a linear-feedback shift register $\mathscr{L}$
- Expand the $\mathscr{L}$ to a binary sequence L, $|L| = 2^m - 1$.
- Launch $\mathscr{A}$ with L as an input

The primitive polynomials over F_{2^m} could be calculated in advance. Furthermore, the PSL of L, where L is seeded by some initial element a over F_{2^m}, could be specially chosen to have the minimum possible value. This is easily achievable by using the theorems discussed later in this chapter (see Subsection 4.3.3):

The aforementioned procedure could be better illustrated by an example. If we fix $m = 17$, we could pick the primitive polynomial $f = x^{17} + x^{14} + x^{12} + x^{10} + x^9 + x + 1$ over $F_{2^{17}}$. Before converting f to a linear-feedback shift register $\mathscr{L}$, we should fix the starting state of $\mathscr{L}$. Throughout this example, a is fixed to the initial state of $\mathscr{L}$:

$$[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1]$$

Then, $\mathscr{L}$ is expanded to L. By using single instruction, multiple data (SIMD) capable device and starting with L, we could efficiently enumerate all 2^{17} different binary sequences generated by all possible starting states, to find the one generating the minimum PSL value. More formally, a value ρ_{max}, s.t. $\forall\rho : (L \leftarrow \rho_{max})_{PSL} \leq (L \leftarrow \rho)_{PSL}$. Considering f and the fixed value of a, in this specific case the value of ρ_{max} is 15150, or more precisely, $(L \leftarrow \rho_{max})_{PSL} = 363$.

Experiments with initializing $\mathscr{A}$ (α=6) with $L \leftarrow \rho_{max}$, instead of pseudo-randomly generated binary sequences, were made. We were able to repeatedly reach record-breaking binary sequences of length 131071 having PSL equal to 359. The time required was less than 2 minutes, which was a significant improvement over the time required for $\mathscr{A}$ (starting from pseudo-randomly generated sequences) to reach binary sequences with PSL close to 359: approximately 3 days. Leaving $\mathscr{A}$ to work for another 46 minutes it even reached binary sequences of length 131071 with PSL 356.

The proposed procedure, as demonstrated, is highly efficient and is capable to reach binary sequences with $\mathscr{A}$-long lengths and record-breaking PSL values for a few minutes. Unfortunately, it is applicable on binary sequences with lengths of the form $2^n - 1$ only. However, throughout the next section, we provide another procedure that can generate binary sequences with length p and record-breaking PSL values, where p is a prime number.

4.3.2 Using $\mathscr{A}$ as an Legendre-sequences extension

The following procedure is proposed:

- Choose a prime number p
- Generate the sequence $L = \left[t_1, t_2, \cdots, t_p\right]$
- For i, s.t. $i \in N$, $1 \leq i \leq p$, and in case i is a quadratic residue mod p, replace t_i with 1. Otherwise, replace t_i with -1.
- Launch $\mathscr{A}$ with L as an input

As the numerical experiments suggested in [44], it is highly unlikely that a Legendre sequence with length p, for $p > 235723$, or any rotation of it, would yield a PSL value less than $\sqrt{p}$. Having this in mind, experiments with initializing $\mathscr{A}$ (α=8) with a rotation of Legendre sequence with length 235747 were made (the next prime number after 235723). Again, by using SIMD-capable devices, we have extracted the PSL-optimal rotation among all possible rotations of a Legendre sequence with length 235747. More precisely, on rotation 60547, a binary sequence with PSL equal to 508 was yielded. Surprisingly, $\mathscr{A}$ was able to significantly optimize this binary sequence. As shown in Table 4.10, for less than 25 minutes, using only 1 thread of a Xeon-2640 CPU with a base frequency of 2.50 GHz, a binary sequence with PSL equal to 408 was found.

Since $\sqrt{235747} \approx 485.54$, it follows that 408 is significantly smaller than the expected value of 485.54. In fact, by leaving $\mathscr{A}$ for a total of 2.21 hours, a binary sequence with length 235747 and PSL 400, or 108▼, was reached. More details could be found in [47].

Table 4.10 Time required for $\mathscr{A}$ to reach smaller PSL values, when launched from a rotated Legendre sequence with length 235747 and rotation value 60547.

PSL	*T*
496	1s
482	6s
462	15s
442	24s
422	9.75m
411	12.4m
410	18.9m
409	23.4m
408	23.7m

4.3.3 On the Aperiodic Autocorrelations of Rotated Binary Sequences

The maximal length shift register sequences, or m-sequences, is a well-known algebraic design [67]. Unfortunately, they are defined for lengths $2^n - 1$ only ($n \in N$). Nevertheless, as shown in [52], their extensive study could provide valuable insights into understanding the world of binary sequences possessing low aperiodic autocorrelation characteristics. However, finding the PSL-optimal m-sequences is a rigid and tedious task - during each iteration, the PSL value of a given binary sequence B, altogether with all possible rotations of B, should be calculated. In [81], an exhaustive search of PSL-optimal m-sequence with lengths up to $2^{15} - 1$ is given. Later, in [52], the exhaustive search study was extended with results regarding m-sequences with lengths $2^{16} - 1$ and $2^{17} - 1$. Since then, no progress was made.

Similar to the problem of finding PSL-optimal m-sequences, finding PSL-optimal Legendre sequences involves a significant computational burden - during each iteration, the PSL value of the binary sequence, altogether with all possible rotations of B, should be calculated. This explains why the numerical results regarding the PSL-optimal Legendre sequences are scarce. For example, in [135], Fig.4, a list of all PSL-optimal Legendre sequences, up to length 3500 only, is given.

The routine of finding the minimum PSL among all the possible rotations of a given binary sequence plays an important role in the overall computational burden. By making some observations of the behavior of the sidelobes array in a rotated sequence, we were able to project the routine to a perfectly balanced parallelizable algorithm. This allows us to efficiently utilize the computational possibilities of modern GPUs. Hence, we were able to exhaustive search all m-sequences with lengths $2^{18} - 1$, $2^{19} - 1$ and $2^{20} - 1$, as well as finding all optimal Legendre sequences with lengths up to 432100 - something out of reasonable computational reach until now.

We denote as $B \leftarrow \rho$ the binary sequence obtained from B, by left-rotating it ρ times. By definition, $B \leftarrow |B| \equiv B$. Furthermore, if b_i is the element of B on position i, we will denote as $b_i^{\leftarrow\rho}$ the element of $B \leftarrow \rho$ on position i.

Theorem 4.3.1. Given a binary sequence $B = b_0 b_1 \cdots b_{n-1}$ with length n, the following property holds:

$$\hat{C}_i(B \leftarrow 1) - \hat{C}_i(B) = b_0 \left(b_{i+1} - b_{n-i-1}\right)$$

Proof 4.3.1. By definition,

$$\hat{C}_i(B) = \sum_{j=0}^{i} b_j b_{j+n-i-1}$$

Since $B \leftarrow 1$ is the left-rotated version of B,

$$\hat{C}_i(B \leftarrow 1) = \sum_{j=0}^{i} b_{(j+1 \bmod n)} b_{(j+1+n-i-1 \bmod n)}$$

Thus, $\hat{C}_i(B \leftarrow 1) - \hat{C}_i(B)$ is equal to:

$$\sum_{j=0}^{i} b_{(j+1 \bmod n)} b_{(j+1+n-i-1 \bmod n)} - \sum_{j=0}^{i} b_j b_{j+n-i-1} = \tag{4.1}$$

$$\sum_{j=0}^{i} b_{(j+1 \bmod n)} b_{(j+n-i \bmod n)} - \sum_{j=0}^{i} b_j b_{j+n-i-1} = \tag{4.2}$$

$$\sum_{j=0}^{i} b_{(j+1 \bmod n)} b_{(j+n-i \bmod n)} - \sum_{j=-1}^{i-1} b_{j+1} b_{j+1+n-i-1} = \tag{4.3}$$

$$\sum_{j=0}^{i} b_{(j+1 \bmod n)} b_{(j+n-i \bmod n)} - \sum_{j=-1}^{i-1} b_{j+1} b_{j+n-i} = \tag{4.4}$$

$$\sum_{j=0}^{i-1} b_{(j+1 \bmod n)} b_{(j+n-i \bmod n)} + b_{(i+1 \bmod n)} b_{(i+n-i \bmod n)} - \tag{4.5}$$

$$\sum_{j=0}^{i-1} b_{j+1} b_{j+n-i} - b_{-1+1} b_{-1+n-i} \tag{4.6}$$

Since $i < n-1$ and $j \le i-1$, we have $j < i < n-1$. Thus, $(j+1) \bmod n = j+1$. However, since $j < i$, or $j - i < 0$, we have $j - i + n < n$. Thus, $(j+n-i) \bmod n = j+n-i$. Hence,

$\hat{C}_i(B \leftarrow 1) - \hat{C}_i(B)$ could be further simplified to:

$$\sum_{j=0}^{i-1} b_{j+1}b_{j+n-i} + b_{i+1}b_{(n \bmod n)} - \tag{4.7}$$

$$\sum_{j=0}^{i-1} b_{j+1}b_{j+n-i} - b_0 b_{n-i-1} = \tag{4.8}$$

$$b_{i+1}b_0 - b_0 b_{n-i-1} = b_0(b_{i+1} - b_{n-i-1}) \tag{4.9}$$

□

Theorem 4.3.2. Given a binary sequence $B = b_0 b_1 \cdots b_{n-1}$ with length n, the difference $\hat{C}_i(B \leftarrow \rho) - \hat{C}_i(B \leftarrow (\rho - 1))$ is equal to $b_{(\rho-1) \bmod n}(b_{(i+\rho) \bmod n} - b_{(n-i+\rho-2) \bmod n})$.

Proof 4.3.2. Since $B = b_0 b_1 \cdots b_{n-1}$, it follows that $B \leftarrow 1 = b_0^{\leftarrow 1} b_1^{\leftarrow 1} \cdots b_{n-1}^{\leftarrow 1}$, or in the more general case $B \leftarrow j = b_0^{\leftarrow j} b_1^{\leftarrow j} \cdots b_{n-1}^{\leftarrow j}$. Thus, by using Theorem 4.3.1:

$$\hat{C}_i(B \leftarrow \rho) - \hat{C}_i(B \leftarrow (\rho - 1)) = \tag{4.10}$$

$$b_0^{\leftarrow(\rho-1)} \left(b_{i+1}^{\leftarrow(\rho-1)} - b_{n-i-1}^{\leftarrow(\rho-1)} \right) \tag{4.11}$$

However, by definition, $b_i^{\leftarrow 1}$ is the element of $B \leftarrow 1$ on position i. Thus, $b_i^{\leftarrow 1} = b_{(i+1) \bmod n}$. In the general case, $b_i^{\leftarrow x} = b_{(i+x) \bmod n}$. By using those relations, we can substitute:

$$b_0^{\leftarrow(\rho-1)} \left(b_{i+1}^{\leftarrow(\rho-1)} - b_{n-i-1}^{\leftarrow(\rho-1)} \right) = \tag{4.12}$$

$$b_{(0+\rho-1) \bmod n} \left(b_{(i+1+\rho-1) \bmod n} - b_{(n-i-1+\rho-1) \bmod n} \right) = \tag{4.13}$$

$$b_{(\rho-1) \bmod n} \left(b_{(i+\rho) \bmod n} - b_{(n-i+\rho-2) \bmod n} \right) \tag{4.14}$$

□

Let us denote as Ω_B the array of all the sidelobes of a some binary sequence B with length n, or more formally: $\Omega_B = [\hat{C}_0(B), \hat{C}_1(B), \cdots, \hat{C}_{n-2}(B)]$. By using Theorem 4.3.2 and the inherited relationship between elements of $\Omega_{B\leftarrow\rho}$ and $\Omega_{B\leftarrow(\rho-1)}$, we can calculate $\Omega_{B\leftarrow\rho}$, given $\Omega_{B\leftarrow(\rho-1)}$, by using $n-1$ distinct parallel threads. Two very beneficial properties should be emphasized:

- The threads are independent of each other.
- The pool of the threads is perfectly balanced in terms of synchronization, i.e. if we have two distinct threads t_i and t_j, the arithmetic operations involved throughout the calculation process of t_i and t_j are the same.

This scenario suits well in the context of the single instruction, multiple data (SIMD) model [56]. We could dedicate the calculation of $\hat{C}_i(B \leftarrow \rho)$ to a thread t_i only since the aforementioned calculation is independent of other threads' results. Moreover, to optimize the routine further, we could just in-memory replace the values of $\hat{C}_i(B)$, i.e. Ω_B, with the consequent values of $\hat{C}_i(B \leftarrow \rho)$, i.e. $\Omega_{B\leftarrow\rho}$, for $\rho \in [1, n-1]$. The pseudo-code of the algorithm is given in Algorithm 6. Throughout the pseudo-code, we have used the following notations:

- $x \leftarrow y$: same as $x = y$
- $x+ = y$: same as $x = x + y$
- $\Omega_B[i]$: the i-th element of Ω_B, i.e. $\hat{C}_i(B)$
- $\max|\Omega_B|$: the maximum absolute value of Ω_B
- **s**y**n**c**: we await all threads to synchronize (otherwise $\max|\Omega_B|$ could lead to an ambiguous result)

Algorithm 6 A GPU algorithm for extracting the minimum PSL value of B, and all possible rotations of B

procedure EXTRACT(B, n)
  $\Omega_B \leftarrow \left[\hat{C}_0(B), \hat{C}_1(B), \cdots, \hat{C}_{n-2}(B)\right]$
  $minPSL \leftarrow n$
  for $\rho \in [0, n-1]$ **do**
    $t_i : \Omega_B[i] + =$
    $b_{(\rho-1) \bmod n}\left(b_{(i+\rho) \bmod n} - b_{(n-i+\rho-2) \bmod n}\right)$
    sy**n**c**
    if $\max|\Omega_B| < minPSL$ **then**
      $minPSL \leftarrow \max|\Omega_B|$
    end if
  end for
end procedure

The observations made in the previous section allow us to design a fast routine for finding the minimum PSL among all the possible rotations of a given binary sequence. Our first practical application was an exhaustive search of all m-sequences with fixed lengths. The proposed algorithm could be summarized as follow:

- Choose a primitive polynomial f over F_{2^m}
- Fix an initial element β over F_{2^m}.

- Convert f to a linear-feedback shift register Γ with a starting state set to β.
- Expand the Γ period to a binary sequence L with length $2^m - 1$.
- Launch Algorithm 6.
- Output the value reached by the previous step.

Since the period of the LSFR L is equal to $2^m - 1$, the proposed algorithm would iterate through all possible starting states of expanded LSFRs constructed from f. Thus, we can conclude that the aforementioned algorithm will find the smallest achievable PSL by m-sequence generated by f with all possible starting states.

The proposed algorithm could be also successfully utilized in finding optimal Legendre sequences. Thus, the following routine is proposed:

- Choose a prime number p.
- Generate the sequence $X = (x_1, x_2, \cdots, x_p)$.
- For i, s.t. $i \in N$, $1 \leq i \leq p$, and in case i is a quadratic residue mod p, replace x_i with 1. Otherwise, replace x_i with -1.
- Launch Algorithm 6.
- Output the value reached by the previous step.

We have implemented the m-sequence exhaustive search algorithm by using an amalgam of programming languages[2] and GPUs as SIMD-capable devices. To analyze the efficiency of our implementation, we have further compared it to the popular scientific computing library NumPy [115]. More specifically, we compare the proposed algorithm with the naive approach of PSL re-calculation by using NumPy. The following notations are used:

- s, m, h, D, Y - seconds, minutes, hours, days, years
- ***X*** S - the time library **X** required for a single PSL calculation
- ***X*** R - the overall time which a given library **X** required for the PSL calculations of all the distinct rotations of a given binary sequence

[2] C, Python, SageMath, CUDA

Table 4.11 GPU algorithm vs CPU NumPy naive approach

n	$2^n - 1$	NumPy S	NumPy R (estimation)	CUDA R
15	32767	1.3s	11.8h	1s
16	65535	4.3s	78.3h	2s
17	131071	16.3s	25D	4s
18	262143	1m4s	194D	14s
19	524287	4m15s	4.2Y	48s
20	1048575	18m5s	36Y	3m11s

During the comparison, a mid-range GPU with approximately 1200 CUDA cores and a mid-range CPU with 6 cores (12 threads) were used. The results are given in Table 4.11. For example, by using a single mid-range GPU, altogether with the aforementioned algorithm, the time required to find the PSL-optimal binary sequence, among the set comprised of a binary sequence B of length $2^{20} - 1$ and all the possible rotations of B, would be 191 seconds. For completing the same calculation on a mid-range CPU, and by using a single thread, the required time would be approximately, based on estimation, 36 years. This results in an approximate speed-up factor of $2^{22.5}$.

The proposed algorithm allowed us to successfully exhaust search all possible m-sequences with lengths $2^{18} - 1$, $2^{19} - 1$ and $2^{20} - 1$. To achieve that, we first created a list of all the primitive polynomials of the corresponding degree. Then, for each polynomial, we launched the proposed algorithm. In Table 4.12 we present the optimal PSL values achieved by the exhaustive search routine. The best PSL values known before this work, to be found in [52], are denoted as π, while the optimal PSL values achieved by our work are denoted as Π. It should be emphasized, that the comparison provided in Table 4.11, for some n, does not reflect the actual time needed for exhaustive search of all m-sequences with length $2^n - 1$, but just the minimum PSL yielded by all possible rotations of a binary sequence generated by just one primitive polynomial. For example, the actual time needed for finding the optimal PSL value among all m-sequences with length $2^{20} - 1$, with or without rotations, is approximately 191 seconds multiplied by $\frac{\phi(2^{20}-1)}{20} = 24000$, or a total of 54 GPU days. However, by using NumPy and a naive approach, the time required would be approximately 864000 (single-thread) CPU years. Example of primitive polynomials yielding the PSL-optimal m-sequences, when rotated, could be found in 4.15, 4.16 and 4.17.

$$x^{18} + x^{15} + x^{14} + x^{13} + x^6 + x^3 + x^2 + x + 1 \tag{4.15}$$

$$x^{19} + x^{17} + x^{14} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^3 + x^2 + 1 \tag{4.16}$$

Table 4.12 Optimum PSL values achieved during the exhaustive search

n	$2^n - 1$	π	Π	$\lfloor\sqrt{2^n - 1}\rfloor$
18	262143	544	507	511
19	524287	775	731	724
20	1048575	1066	1024	1023

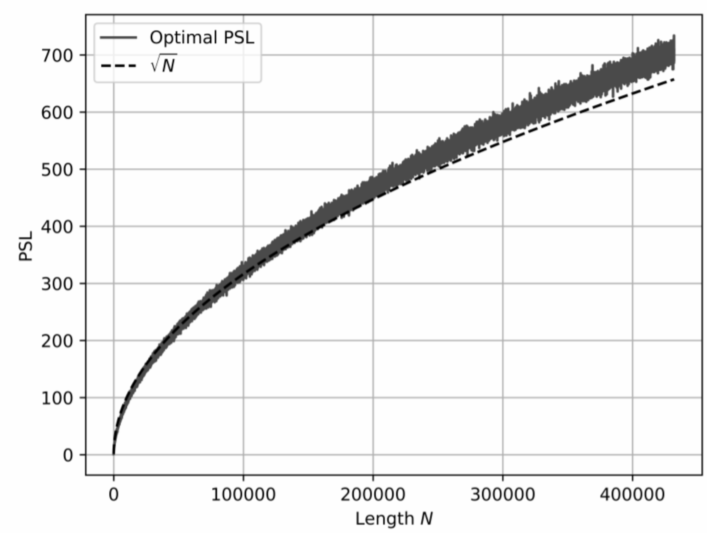


Fig. 4.4 A complete map of the optimal PSL values of all the Legendre sequences with lengths less than 432100, with or without rotation.

$$x^{20} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^4 + x + 1 \tag{4.17}$$

We were able to successfully reveal all the optimal PSL values for Legendre sequences up to length 432100. In Figure 4.4, an overview of the optimal Legendre sequences is given.

It could be observed (depicted with a dashed line), in the beginning, the resulting trace stays very close to the line $y = \sqrt{n}$. It appears, that the beam comprised of the PSL-optimal Legendre sequence values, at least up to length 2^{17}, is still able to cover the y trace. However, a tendency of overall PSL-increasing, compared to line y, could be noticed. Indeed, during our experiments, we were not able to find a Legendre sequence B, having length n greater than 235723, such that B, or rotations of B, yield a PSL value less or equal to $\sqrt{n}$. Combining this fact with the overall tendency of PSL beam increasing, we conjecture that all Legendre sequences, with or without rotation, and with lengths $n > 235723$, could not reach a PSL value less or equal to $\sqrt{n}$. The results are published in [44].

Chapter 5

Binary Sequences and the Merit Factor Problem

The merit factor problem is of practical importance to manifold domains, such as digital communications engineering, radars, system modulation, system testing, information theory, physics, and chemistry. However, the merit factor problem is referenced as one of the most difficult optimization problems and it was further conjectured that stochastic search procedures will not yield merit factors higher than 5 for long binary sequences (sequences with lengths greater than 200). Some useful mathematical properties related to the flip operation of the skew-symmetric binary sequences are presented in this chapter. By exploiting those properties, the memory requirement of state-of-the-art stochastic merit factor optimization algorithms could be reduced from $O(n^2)$ to $O(n)$. As a proof of concept, a lightweight stochastic algorithm was constructed, which can optimize pseudo-randomly generated skew-symmetric binary sequences with long lengths (up to $10^5 + 1$) to skew-symmetric binary sequences with a merit factor greater than 5. An approximation of the required time is also provided. The numerical experiments suggest that the algorithm is universal and could be applied to skew-symmetric binary sequences with arbitrary lengths.

5.1 On the Skew-Symmetric Binary Sequences and the Merit Factor Problem

If F_n denotes the optimal (greatest) value of the merit factor among all sequences of length n, then the merit factor problem could be described as finding the value of $\limsup_{n\to\infty} F_n$. Several conjectures regarding the $\limsup_{n\to\infty} F_n$ value should be mentioned. The first conjecture published in [75] assumes that $\limsup_{n\to\infty} F_n = 6$. A more extreme conjecture that $\limsup_{n\to\infty} F_n = \infty$ is given by Littlewood [97]. In [28], it was conjectured that $\limsup_{n\to\infty} F_n = 5$. Golay [63] assumed that the expected value of $\limsup_{n\to\infty} F_n$ is very close to 12.32. However, in [64] he added that "...no systematic synthesis will ever be found which will yield higher merit factors [than 6]...". Nevertheless, in [22] it was conjectured that $\limsup_{n\to\infty} F_n > 6.34$. The latest assumption is based on the specially constructed infinite family of sequences.

Since the merit factor problem has resisted more than 50 years of theoretical attacks, a significant number of computational pieces of evidence were collected. They could be divided into exhaustive search methods and heuristic methods.

Regarding the exhaustive search methods, the optimal merit factors for all binary sequences with lengths $n \leq 60$ are given in [105]. Twenty years later, the list of optimal merit factors was extended to $n \leq 66$ [118]. The two largest known values of F_n are 14.1 and 12.1 for n equals respectively 13 and 11. It should be mentioned that both of those binary sequences are comprised of the Barker sequences [9]. In fact, in [80] the author published a personal selection of challenges concerning the merit factor problem, arranged in order of increasing significance. The first suggested challenge is to find a binary sequence X of length $n > 13$ for which $F(X) \geq 10$.

A reasonable strategy for finding binary sequences with near-optimal merit factors is to introduce some restriction on the sequences' structure. A well-studied restriction on the structure of the sequence has been defined by the skew-symmetric binary sequences, which were introduced by Golay [60]. Having a binary sequence $(b_0, b_1, \cdots, b_{2l})$ of odd length $n = 2l + 1$, the restriction is defined by

$$b_{l+i} = (-1)^i b_{l-i} \text{ for } i = 1, 2, \cdots, l.$$

Golay observed that odd-length Barker sequences are skew-symmetric. Therefore, an idea of binary sequences' sieving was proposed [62]. Furthermore, as shown in [60], all aperiodic autocorrelations of a skew-symmetric sequence with even indexes are equal to 0.

The optimal merit factors for all skew-symmetric sequences of odd length $n \leq 59$ were given by Golay himself [62]. Later, the optimal merit factors for skew-symmetric sequences with lengths $n \leq 69$ and $n \leq 71$ were revealed respectively in [65] and [41], while the optimal skew-symmetric solutions for $n \leq 89$ and $n \leq 119$ were given in respectively [125] and [118].

It should be noted, that the problem of minimizing F_n is also known as the "low autocorrelated binary string problem", or the LABS problem. It has been well studied in theoretical physics and chemistry. For example, the LABS problem is correlated with the quantum models of magnetism. Having this in mind, the merit factor problem was attacked by various search algorithms, such as the branch and bound algorithm proposed in [118], as well as stochastic search algorithms like tabu search [72], memetic algorithm combined with tabu search [57], as well as evolutionary and genetic algorithms [41] and [106]. However, since the search space grows like 2^n, the difficulty of finding long binary sequences with near-optimal F_n significantly increases. Bernasconi predicted that [14] " ... stochastic search procedures will not yield merit factors higher than about $F_n = 5$ for long sequences". By long sequences, Bernasconi was referring to binary sequences with lengths greater than 200. Furthermore, in [41] the problem was described as " ... amongst the most difficult optimization problems".

The principle behind basic search methods could be summarized as moving through the search space by doing tiny changes inside the current binary sequence. In the case of skew-symmetric binary sequences, Golay suggested [61] that only one or two elements should be changed at a given optimization step. In case the new candidate has a better merit factor, the search method accepts it as a new current state and continues the optimization process. Having this in mind, a strategy of how to choose a new sequence when no acceptable neighbor sequence exists should be considered as well.

The best results regarding skew-symmetric binary sequences with high merit factors are achieved by [24], [26], [27], and [57]. In [57], the authors introduced a memetic algorithm with an efficient method to recompute the characteristics of a given binary sequence L', such that L' is one flip away from L, and assuming that some products of elements from L have been already stored in memory. More precisely, a square $(n-1, n-1)$ tau table $\mathcal{T}(S)$, such that $\mathcal{T}(S)_{ij} = s_j s_{i+j}$ for $j \leq n-i$ was introduced. Later, in [24] the principle of self-avoiding walk [100] was considered. By using Hasse graphs the authors demonstrated that considering the LABS problem, a basic stochastic search method could be easily trapped in a cycle. To avoid this scenario, the authors suggested the usage of a self-avoiding walk strategy accompanied by a hash table for efficient memory storage of the pivot coordinates. Then, in [26] an algorithm called xLastovka was presented. The concept of a priority queue was introduced. In summary, during the optimization process, a queue of pivot coordinates altogether with their energy values is maintained. Recently, some skew-symmetric binary sequences with record-breaking merit factors for lengths from 301 to 401 were revealed [27].

The aforementioned state-of-the-art algorithms are benefiting from the tau table $\mathcal{T}(S)$ previously discussed. It significantly increases the speed of evaluating a given one-flip-away neighbor, reaching a time complexity of $O(n)$. However, the memory requirement of maintaining $\mathcal{T}(S)$ is like $O(n^2)$. Having this in mind, the state-of-the-art algorithms could be inapplicable to very long binary sequences due to hardware restrictions.

In this section, by using some mathematical insights, an alternative to the $\mathcal{T}(S)$ table is suggested, the usage of which significantly reduces the memory requirement of the discussed state-of-the-art algorithms from $O(n^2)$ to $O(n)$. This enhancement could be easily integrated. For example, in an online repository [23] a collection of currently known best merit factors for skew-symmetric sequences with lengths from 5 to 449 is given. The longest binary sequence is of length 449, having a merit factor of 6.5218. As a proof of concept, by using just a single budget processor Xeon-2640 CPU with a base frequency of 2.50 GHz, the price of which at the time of writing this work is about 15 dollars, and our tweaked implementation of the lssOrel algorithm introduced in [23], we were able to find a skew-symmetric binary sequence with better merit factor of 6.5319. The time required was approximately one day. As a comparison, the currently known optimal results were acquired by using the Slovenian Initiative for National Grid (SLING) infrastructure (100 processors) and a 4-day threshold limitation per length.

It should be noted, that despite the significant memory complexity optimization introduced, the state-of-the-art algorithms could still suffer from memory and speed issues. As previously discussed, additional memory-requiring structures were needed, such as, for example, a set of all previously visited pivots [24] or a priority queue with 640 000 coordinates and a total size of 512MB [26].

Another issue is the "greedy" approach of collecting all the neighbors to determine the best one. This could dramatically decrease the optimization process, especially when very long binary sequences are involved. This side-effect is already discussed in Section 4.2.

Having those observations in mind, an almost memory-free optimization algorithm is suggested. More precisely, both the time and memory complexities of the algorithm are linear. This could be particularly beneficial for multi-thread architectures or graphical processing units. During our experiments, and by using the aforementioned algorithm, we were able to find skew-symmetric sequences with merit factors strictly greater than $F_n = 5$ for all the tested lengths up to $10^5 + 1$. Thus, Bernasconi's prediction that no stochastic search procedure will yield merit factors higher than $F_n = 5$ for binary sequences with lengths greater than 200 was very pessimistic.

Let us consider a skew-symmetric binary sequence defined by an array $L = [b_0, b_1, \cdots, b_{n-1}]$ with an odd length $n = 2l + 1$. If the corresponding to L sidelobes' array is denoted by an array W, we have:

$$W = [C_{n-1}(L), C_{n-2}(L), \cdots, C_1(L), C_0(L)],$$

where

$$C_u(L) = \sum_{j=0}^{n-u-1} b_j b_{j+u}, \quad for\ u \in \{0, 1, \cdots, n-1\}.$$

In this section, for convenience, we will use the reversed version of W, denoted by S, s.t:

$$S = \left[\hat{C}_0(L), \hat{C}_1(L), \cdots, \hat{C}_{n-2}(L), \hat{C}_{n-1}(L)\right],$$

where $\hat{C}_{n-i-1}(L) = C_i(L)$, for $i \in \{0, 1, \cdots, n-1\}$. Thus,

$$\hat{C}_i(L) = C_{n-i-1}(L) = \sum_{j=0}^{n-(n-i-1)-1} b_j b_{j+(n-i-1)}.$$

Hence,

$$\hat{C}_i(L) = \sum_{j=0}^{i} b_j b_{j+n-i-1}, \quad for\ i \in \{0, 1, \cdots, n-1\}.$$

Furthermore, we will denote the i-th element of a given array A as $A[i]$. It should be noted that the first index of an array is 0, not 1. For example,

$$W[n-1] = S[0] = \hat{C}_0[L] = C_{n-1}(L).$$

Since L is a skew-symmetric binary sequence, the following properties hold:

- $S[i] = 0$, for odd values of i.
- $L[l-i] = (-1)^i L[l+i]$.

Having this in mind, the array of sidelobes S could be represented as follows:

$$S = \left[\hat{C}_0(L), 0, \hat{C}_2(L), 0, \cdots, 0, \hat{C}_{n-3}(L), 0, \hat{C}_{n-1}(L)\right].$$

For convenience, we will use the notation S_i which represents the $(i-1)$-th element of a given array S, or more formally $S_i = S[i-1]$.

Thus, for every odd value r, we have

$$S_r = \hat{C}_{r-1}(L) = \sum_{j=0}^{r-1} b_j b_{j+n-r+1-1} = \sum_{j=0}^{r-1} b_j b_{j+n-r} = \sum_{j=1}^{r} b_{j-1} b_{j-1+n-r}.$$

In terms of L, the previous relationship could be written down as follows:

$$S_r = \sum_{j=1}^{r} b_{j-1} b_{j-1+n-r} = \sum_{i=1}^{r} L[i-1]L[n+i-r-1].$$

We could further substitute $i = l - q$, for $q \in \{0, 1, \cdots, l\}$ into the major property of the skew-symmetric sequences to show that:

$$L[l - l + q] = (-1)^{l-q} L[l + l - q] \implies$$
$$L[q] = (-1)^{l-q} L[l + l + 1 - q - 1] \implies L[q] = (-1)^{l-q} L[n - q - 1].$$

Hence, given a skew-symmetric sequence L with length $n = 2l + 1$, if we flip both the elements on positions q and $n - q - 1$, for some fixed $q \in \{0, 1, \cdots, l\}$, the resulted binary sequence L^q will be skew-symmetric as well. Let's denote the array of sidelobes of L^q as S^q:

$$S^q = \left[\hat{C}_0(L^q), 0, \hat{C}_2(L^q), 0, \cdots, 0, \hat{C}_{n-3}(L^q), 0, \hat{C}_{n-1}(L^q)\right].$$

As a consequence of the previously aforementioned observations, we have:

$$S_r^q = \sum_{i=1}^{r} L^q[i-1] L^q[n+i-r-1].$$

In Theorem 5.1.1 a more detailed picture of the S array transformation to the S^q array is provided.

Theorem 5.1.1. Given two skew-symmetric sequences L and L^q with length $n = 2l + 1$, and with sidelobes arrays respectively S and S^q, where $q < l$, the following properties hold:

I For $\forall e$, s.t. e is an even number, $S_e^q - S_e = 0$.

II If r is an odd number and $r \leq q$, $S_r^q - S_r = 0$.

III If r is an odd number and $r > q$, and $r < n - q$, and $q \neq r - q - 1$, then:

$$S_r^q - S_r = -2\left(L[q]L[n+q-r] + L[r-q-1]L[n-q-1]\right).$$

IV If r is an odd number and $r > q$, and $r < n - q$, and $q = r - q - 1$, then $S_r^q - S_r = 0$.

V If r is an odd number and $r \geq n - q$, and $q \neq r - q - 1$, then:

$$S_r^q - S_r = -2L[n-q-1]L[2n-q-r-1] - 2L[q+r-n]L[q] -$$
$$-2L[q]L[n+q-r] - 2L[r-q-1]L[n-q-1].$$

VI If r is an odd number and $r \geq n - q$, and $q = r - q - 1$, then:

$$S_r^q - S_r = -2L[n-q-1]L[2n-q-r-1] - 2L[q+r-n]L[q].$$

Proof 5.1.1. Property I: For $\forall e$, s.t. e is an even number, $S_e = 0$ and $S_e^q = 0$, since both S and S^q are skew-symmetric sequences. Therefore, $S_e^q - S_e = 0$.

Property II: If r is an odd number and $r \leq q$, then

$$S_r^q - S_r = \sum_{i=1}^{r} L^q[i-1]L^q[n+i-r-1] - \sum_{i=1}^{r} L[i-1]L[n+i-r-1].$$

By construction, $L^q[q] \neq L[q]$, $L^q[n-q-1] \neq L[n-q-1]$ and $\forall x \in [0,1,\cdots,n-1], x \neq q \;\&\; x \neq n-q-1 : L^q[x] = L[x]$. Since, by considering the initial condition $r \leq q$, it follows that $r-1 < q$. Therefore, for $i \in \{1,2,\cdots,r\}$, $i-1 \leq r-1 < q$ and $L^q[i-1] = L[i-1]$. On the other hand, for $i \in \{1,2,\cdots,r\}$, $n+i-r-1 \geq n+1-r-1 = n-r$, but since $r \leq q$, then $n-r \geq n-q > n-q-1$, which means that $L^q[n+i-r-1] = L[n+i-r-1]$.

By combining the aforementioned observations:

$$\begin{aligned} S_r^q - S_r &= \sum_{i=1}^{r} L^q[i-1]L^q[n+i-r-1] - \sum_{i=1}^{r} L[i-1]L[n+i-r-1] = \\ &= \sum_{i=1}^{r} L[i-1]L[n+i-r-1] - \sum_{i=1}^{r} L[i-1]L[n+i-r-1] = 0. \end{aligned} \tag{5.1}$$

Property III We consider r as an odd number, $r > q$, $r < n-q$, and $q \neq r-q-1$. Since $r > q$, we have $r-1 \geq q$, which means that at least one element from the elements defined by $L^q[i-1]$, for $i \in \{1,2,\cdots,r\}$, will coincide with $L^q[q]$. However, since $r < n-q$, or $r-1 < n-q-1$, there will be no element from the elements defined by $L^q[i-1]$, for $i \in \{1,2,\cdots,r\}$, that will coincide with $L^q[n-q-1]$.

For $i \in \{1,2,\cdots,r\}$, $n+i-r-1 \geq n-r$. If $n-r \leq q$ then $n-q \leq r$, which contradicts the initial condition of $r < n-q$. Therefore, $n-r > q$ and $n+i-r-1 > q$, and there will be no element from the elements defined by $L^q[n+i-r-1]$, for $i \in \{1,2,\cdots,r\}$, that will coincide with $L^q[q]$. On the other hand, for $i \in \{1,2,\cdots,r\}$, $n+i-r-1 \geq n-r$, and since $r > q$, $n-r < n-q$. Thus $n-r \leq n-q-1$, which means there will be an element from the elements defined by $L^q[n+i-r-1]$, for $i \in \{1,2,\cdots,r\}$, which will coincide with $L^q[n-q-1]$.

$$S_r^q - S_r = \sum_{i=1}^{r} L^q[i-1]L^q[n+i-r-1] - \sum_{i=1}^{r} L[i-1]L[n+i-r-1] =$$
$$= (\sum_{i=1}^{q} L^q[i-1]L^q[n+i-r-1]) + L^q[q]L^q[n+q-r] + (\sum_{i=q+2}^{r} L^q[i-1]L^q[n+i-r-1]) -$$
$$- (\sum_{i=1}^{q} L[i-1]L[n+i-r-1]) - L[q]L[n+q-r] - \sum_{i=q+2}^{r} L[i-1]L[n+i-r-1]. \tag{5.2}$$

However, since it is given that $q \neq r-q-1$, then $n+q-r \neq n+r-q-1-r = n-q-1$. Thus, the coinciding elements are still to be determined inside the sequences defined for $i \in \{q+2, q+3, \cdots, r\}$. Furthermore, as previously shown, we have:

$$\sum_{i=1}^{q} L^q[i-1]L^q[n+i-r-1] = \sum_{i=1}^{q} L[i-1]L[n+i-r-1].$$

Hence:

$$\begin{aligned}
&S_r^q - S_r = \\
&= L^q[q]L^q[n+q-r] + (\sum_{i=q+2}^{r} L^q[i-1]L^q[n+i-r-1]) - \\
&- L[q]L[n+q-r] - \sum_{i=q+2}^{r} L[i-1]L[n+i-r-1] = \\
&= L^q[q]L^q[n+q-r] + (\sum_{i=q+2}^{r-q-1} L^q[i-1]L^q[n+i-r-1]) + \\
&+ L^q[r-q-1]L^q[n+r-q-r-1] + \\
&+ (\sum_{i=r-q+1}^{r} L^q[i-1]L^q[n+i-r-1]) - \\
&- L[q]L[n+q-r] - \sum_{i=q+2}^{r-q-1} L[i-1]L[n+i-r-1] - \\
&- L[r-q-1]L[n+r-q-r-1] - \sum_{i=r-q+1}^{r} L[i-1]L[n+i-r-1].
\end{aligned} \tag{5.3}$$

Since we have isolated all coincidences, it follows:

$$\sum_{i=q+2}^{r-q-1} L^q[i-1]L^q[n+i-r-1] = \sum_{i=q+2}^{r-q-1} L[i-1]L[n+i-r-1].$$

$$\sum_{i=r-q+1}^{r} L^q[i-1]L^q[n+i-r-1] = \sum_{i=r-q+1}^{r} L[i-1]L[n+i-r-1].$$

Thus,

$$\begin{aligned} S_r^q - S_r &= L^q[q]L^q[n+q-r] + L^q[r-q-1]L^q[n-q-1]- \\ &- L[q]L[n+q-r] - L[r-q-1]L[n-q-1]. \end{aligned} \tag{5.4}$$

However, since L^q is identical to L with q-th and $n-q-1$-th bits flipped, we have $L^q[q] = -L[q]$ and $L^q[n-q-1] = -L[n-q-1]$.

$$\begin{aligned} S_r^q - S_r &= -L[q]L^q[n+q-r] - L^q[r-q-1]L[n-q-1]- \\ &- L[q]L[n+q-r] - L[r-q-1]L[n-q-1] = \\ &= -L[q]L[n+q-r] - L[r-q-1]L[n-q-1]- \\ &- L[q]L[n+q-r] - L[r-q-1]L[n-q-1] = \\ &= -2\left(L[q]L[n+q-r] + L[r-q-1]L[n-q-1]\right). \end{aligned} \tag{5.5}$$

Property IV This property is almost identical to Property III. However, this time the fact that $q = r-q-1$ should be considered. More precisely, we should revisit the equation:

$$\begin{aligned} S_r^q - S_r &= L^q[q]L^q[n+q-r] + \Big(\sum_{i=q+2}^{r} L^q[i-1]L^q[n+i-r-1]\Big)- \\ &- L[q]L[n+q-r] - \sum_{i=q+2}^{r} L[i-1]L[n+i-r-1]. \end{aligned} \tag{5.6}$$

Since $q = r-q-1$, or $2q = r-1$, and $n+q-r = n+q-2q-1 = n-q-1$, both coincides appeared on the same monomial:

$$\sum_{i=q+2}^{r} L^q[i-1]L^q[n+i-r-1] = \sum_{i=q+2}^{r} L[i-1]L[n+i-r-1].$$

Therefore,

$$
\begin{aligned}
S_r^q - S_r &= L^q[q]L^q[n+q-r] - L[q]L[n+q-r] = \\
&= L^q[q]L^q[n-q-1] - L[q]L[n-q-1] = \\
&= -L[q]L^q[n-q-1] - L[q]L[n-q-1] = \\
&= L[q]L[n-q-1] - L[q]L[n-q-1] = 0.
\end{aligned} \tag{5.7}
$$

Property V We have that $r \geq n-q$, while in the same time $q \neq r-q-1$. We continue the proof of this and the consequence properties by following the same method and observations made throughout the proof of Properties III and IV. A total of 4 coincides between L^q and L are possible:

- $i-1 = q$, or $i = q+1$.
- $n+i-r-1 = q$, or $i = q+r-n+1$.
- $i-1 = n-q-1$, or $i = n-q$.
- $n+i-r-1 = n-q-1$, or $i = r-q$.

$$
\begin{aligned}
S_r^q - S_r = &\sum_{i=1}^{r} L^q[i-1]L^q[n+i-r-1] - \sum_{i=1}^{r} L[i-1]L[n+i-r-1] = \\
= &\sum_{i=1, i \notin \{q+1, q+r-n+1, n-q, r-q\}}^{r} L^q[i-1]L^q[n+i-r-1] + \\
&+ L^q[q]L^q[n+q-r] + L^q[q+r-n]L^q[q] + \\
&+ L^q[n-q-1]L^q[2n-q-r-1] + \\
&+ L^q[r-q-1]L^q[n-q-1] - \\
- &\sum_{i=1, i \notin \{q+1, q+r-n+1, n-q, r-q\}}^{r} L[i-1]L[n+i-r-1] - \\
&- L[q]L[n+q-r] - L[q+r-n]L[q] - \\
&- L[n-q-1]L[2n-q-r-1] - \\
&- L[r-q-1]L[n-q-1].
\end{aligned} \tag{5.8}
$$

Since L^q is identical to L with q-th and $n-q-1$-th bits flipped, it follows that both sums are comprised of non-flipped bits, and therefore they are equal. Thus:

$$
\begin{aligned}
S_r^q - S_r = {} & L^q[q]L^q[n+q-r] + L^q[q+r-n]L^q[q]+ \\
& + L^q[n-q-1]L^q[2n-q-r-1]+ \\
& + L^q[r-q-1]L^q[n-q-1]- \\
& - L[q]L[n+q-r] - L[q+r-n]L[q]- \\
& - L[n-q-1]L[2n-q-r-1]- \\
& - L[r-q-1]L[n-q-1] = \\
& = -L[q]L[n+q-r] - L[q+r-n]L[q]- \\
& - L[n-q-1]L[2n-q-r-1]- \\
& - L[r-q-1]L[n-q-1]- \\
& - L[q]L[n+q-r] - L[q+r-n]L[q]- \\
& - L[n-q-1]L[2n-q-r-1]- \\
& - L[r-q-1]L[n-q-1] = \\
& = -2*(L[q]L[n+q-r] + L[q+r-n]L[q]+ \\
& + L[n-q-1]L[2n-q-r-1] + L[r-q-1]L[n-q-1]).
\end{aligned}
\tag{5.9}
$$

Property VI This property is very similar to the previous Property V. However, since $q = r - q - 1$, and by using the similar approach shown throughout the proof of Property IV, we could exactly pinpoint those monomials that include a double coincide. Indeed, when $q = r - q - 1$, $n + q - r = n + (r - q - 1) - r = n - q - 1$. Thus:

$$
\begin{aligned}
S_r^q - S_r = {} & \sum_{i=1}^{r} L^q[i-1]L^q[n+i-r-1] - \sum_{i=1}^{r} L[i-1]L[n+i-r-1] = \\
= {} & \sum_{i=1, i\notin\{q+1, q+r-n+1, n-q, r-q\}}^{r} L^q[i-1]L^q[n+i-r-1]+ \\
& + L^q[q]L^q[n+q-r] + L^q[q+r-n]L^q[q]+ \\
& + L^q[n-q-1]L^q[2n-q-r-1]- \\
- {} & \sum_{i=1, i\notin\{q+1, q+r-n+1, n-q, r-q\}}^{r} L[i-1]L[n+i-r-1]- \\
& - L[q]L[n+q-r] - L[q+r-n]L[q]- \\
& - L[n-q-1]L[2n-q-r-1].
\end{aligned}
\tag{5.10}
$$

However:

$$\begin{aligned}
&L^q[q]L^q[n+q-r]-L[q]L[n+q-r]=\\
&=L^q[q]L^q[n-q-1]-L[q]L[n-q-1]=\\
&=(-1)L[q](-1)L[n-q-1]-L[q]L[n-q-1]=0.
\end{aligned} \tag{5.11}$$

Thus:

$$\begin{aligned}
&S_r^q-S_r=\\
&=\sum_{i=1,i\notin\{q+1,q+r-n+1,n-q,r-q\}}^{r} L^q[i-1]L^q[n+i-r-1]+\\
&L^q[q+r-n]L^q[q]+\\
&+L^q[n-q-1]L^q[2n-q-r-1]-\\
&-\sum_{i=1,i\notin\{q+1,q+r-n+1,n-q,r-q\}}^{r} L[i-1]L[n+i-r-1]-\\
&L[q+r-n]L[q]-\\
&-L[n-q-1]L[2n-q-r-1].
\end{aligned} \tag{5.12}$$

Following the same observations made throughout the proof of Property V, the equation could be further simplified to:

$$S_r^q-S_r=-2(L[q+r-n]L[q]+L[n-q-1]L[2n-q-r-1]). \tag{5.13}$$

□

We should emphasize, that Theorem 5.1.1 covers all the possible sidelobes positions and all the possible flip bit choices. Indeed, let's define the sidelobe position as s, while the flip bit position as q. Furthermore, we denote property X as δ_X. Then:

$$\begin{aligned}
&\forall s\forall q\equiv(\forall e:e\equiv 0 \bmod 2)\forall q\bigcup(\forall r:r\equiv 1 \bmod 2)\forall q\equiv\\
&\equiv\delta_1\bigcup(\forall r:r\equiv 1 \bmod 2)(\forall q:r\leq q)\bigcup\\
&\bigcup(\forall r:r\equiv 1 \bmod 2)(\forall q:r>q)=\\
&=\delta_1\bigcup\delta_2\bigcup(\forall r:r\equiv 1 \bmod 2)(\forall q:r>q,r<n-q)\bigcup\\
&\bigcup(\forall r:r\equiv 1 \bmod 2)(\forall q:r>q,r\geq n-q).
\end{aligned} \tag{5.14}$$

For convenience, we will substitute $(\forall r : r \equiv 1 \bmod 2)$ as $\forall r \in \mathbb{O}$:

$$\begin{aligned}
&\delta_1 \bigcup \delta_2 \bigcup (\forall r \in \mathbb{O})(\forall q : r > q, r < n-q) \bigcup \\
&\bigcup (\forall r \in \mathbb{O})(\forall q : r > q, r \geq n-q) = \\
&= \delta_1 \bigcup \delta_2 \bigcup (\forall r \in \mathbb{O})(\forall q : r > q, r < n-q, q \neq r-q-1) \bigcup \\
&\bigcup (\forall r \in \mathbb{O})(\forall q : r > q, r < n-q, q = r-q-1) \bigcup \\
&\bigcup (\forall r \in \mathbb{O})(r \geq n-q) = \bigcup_{i=1}^{4} \delta_i \bigcup (\forall r \in \mathbb{O})(r \geq n-q) = \\
&= \bigcup_{i=1}^{4} \delta_i \bigcup (\forall r \in \mathbb{O})(r \geq n-q, q \neq r-q-1) \bigcup \\
&\bigcup (\forall r \in \mathbb{O})(r \geq n-q, q = r-q-1) = \bigcup_{i=1}^{6} \delta_i .
\end{aligned} \tag{5.15}$$

Furthermore, $\bigcap_{i=1}^{6} \delta_i = \varnothing$. Theorem 5.1.1, as well as the observations made throughout this section, are summarized as a pseudo-code in Algorithm 7. The following notations were used:

- $n = 2l + 1$: the odd length of the sequence.
- q: the bit position which is to be flipped. Defined for $q < l$. Please note, that besides q, the algorithm is going to flip $n - q - 1$ as well, since we want to keep the skew-symmetric property of the binary sequence.
- L: a binary skew-symmetric sequence.
- S: the sidelobes array corresponding to L.

When the algorithm finishes, L is going to be modified to L^q, while S is going to correspond to the sidelobes array of L^q. This is accomplished in $O(n)$ for both time and memory complexities.

Theorem 5.1.2. Given two skew-symmetric sequences L and L^q with length $n = 2l + 1$, where L^q corresponds to L with q-th and $n - q - 1$-th bit flipped for some fixed $q < l$, and with sidelobes arrays denoted respectively as S and S^q, the following property holds:

Algorithm 7 An algorithm for in-memory flip of skew-symmetric binary sequence in linear time and memory complexities

procedure FLIP(q, L, S)
  for $r = 1; r < n-1; r+ = 2$ **do**
    if $r \leq q$ **then**
      continue
    end if
    $\varepsilon_1 = L[q], \varepsilon_2 = L[n+q-r], \varepsilon_3 = L[r-q-1]$
    $\varepsilon_4 = L[n-q-1], \varepsilon_5 = L[2n-q-r-1], \varepsilon_6 = L[q+r-n]$
    if $r < n-q$ **then**
      if $q \neq r-q-1$ **then**
        $S_r = S_r - 2(\varepsilon_1\varepsilon_2 + \varepsilon_3\varepsilon_4)$
      end if
    else
      if $q \neq r-q-1$ **then**
        $S_r = S_r - 2(\varepsilon_1\varepsilon_2 + \varepsilon_3\varepsilon_4 + \varepsilon_4\varepsilon_5 + \varepsilon_6\varepsilon_1)$
      else
        $S_r = S_r - 2(\varepsilon_4\varepsilon_5 + \varepsilon_6\varepsilon_1)$
      end if
    end if
  end for
  $L[q] = -L[q], L[n-q-1] = -L[n-q-1]$
end procedure

$$\mathbb{E}(L^q) = \mathbb{E}(L) + \sum_{r=q+1, r\neq 2q+1}^{n-q-1} (16+\sigma\kappa\varepsilon_1) + \sum_{r=n-q, r\neq 2q+1}^{n-1} (\kappa(\varepsilon_2+\sigma\varepsilon_1)+32+32\sigma\varepsilon_1\varepsilon_2) + \\ + \sum_{r\geq n-q, r\leq n-1, r=2q+1} (16+\kappa\varepsilon_2), \tag{5.16}$$

where $\sigma = (-1)^{l-q}$, $\kappa = -8S_r L[q]$, $\varepsilon_1(r) = L[r-q-1]$, $\varepsilon_2(r) = L[q+r-n]$.

Proof 5.1.2.

$$\begin{aligned} \mathbb{E}(L^q) - \mathbb{E}(L) &= \sum_{i=1}^{n-1} (S_i^q)^2 - \sum_{i=1}^{n-1} (S_i)^2 = \sum_{i=1}^{n-1} ((S_i^q)^2 - (S_i)^2) = \\ &= \sum_{j=1}^{6} \sum_{i\in\mathbb{D}(\delta_j)} ((S_i^q)^2 - (S_i)^2) = \sum_{j=1}^{6} \sum_{i\in\mathbb{D}(\delta_j)} ((S_i+\delta_j)^2 - (S_i)^2) = \\ &= \sum_{j=1}^{6} \sum_{i\in\mathbb{D}(\delta_j)} (2S_i\delta_j + \delta_j^2). \end{aligned} \tag{5.17}$$

We proceed with the calculation of δ_i^2, for $i \in [3,5,6]$.

$$\begin{aligned} \delta_3^2 &= (-2\,(L[q]L[n+q-r] + L[r-q-1]L[n-q-1]))^2 = \\ &= 4(L[q]^2 L[n+q-r]^2 + L[r-q-1]^2 L[n-q-1]^2 + \\ &+ 2L[q]L[n+q-r]L[r-q-1]L[n-q-1]). \end{aligned} \tag{5.18}$$

However, $L[x]^2 = 1$ for any x, therefore:

$$\delta_3^2 = 4(1+1+2L[q]L[n+q-r]L[r-q-1]L[n-q-1]). \tag{5.19}$$

Furthermore, from the main property of the skew-symmetric binary sequences, we know that $L[q] = (-1)^{l-q} L[n-q-1]$. Thus:

$$\begin{aligned} L[n+q-r] &= (-1)^{l-(n+q-r)} L[n-(n+q-r)-1] = \\ &= (-1)^{l-n-q+r} L[n-n-q+r-1] = \\ &= (-1)^{l-n-q+r} L[r-q-1]. \end{aligned} \tag{5.20}$$

However, since $r \equiv n \equiv 1 \mod 2$, we know that $r - n \equiv 0 \mod 2$ and therefore $(-1)^{l-n-q+r} = (-1)^{l-q}$. Having this in mind, we can further simplify δ_3^2:

$$\begin{aligned}
\delta_3^2 &= 8 + 8L[q]L[n+q-r]L[r-q-1]L[n-q-1]) = \\
&= 8 + 8L[q](-1)^{l-q}L[r-q-1])L[r-q-1](-1)^{l-q}L[q] = \\
&= 8 + 8L[q]^2(-1)^{2(l-q)}L[r-q-1]^2 = 8 + 8 = 16.
\end{aligned} \tag{5.21}$$

The calculation of δ_5^2 is similar to the calculation of δ_3^2. Indeed:

$$\begin{aligned}
\delta_5^2 &= (-2L[n-q-1]L[2n-q-r-1] - 2L[q+r-n]L[q]- \\
&- 2L[q]L[n+q-r] - 2L[r-q-1]L[n-q-1]))^2.
\end{aligned} \tag{5.22}$$

We could simplify $L[2n-q-r-1]$:

$$\begin{aligned}
&L[2n-q-r-1] = \\
&= (-1)^{l-(2n-q-r-1)}L[n-(2n-q-r-1)-1] = \\
&= (-1)^{l-2n+q+r+1}L[-n+q+r].
\end{aligned} \tag{5.23}$$

Since r is odd, $r+1$ is even, and therefore $r+1-2n \equiv 0 \mod 2$. Therefore, $(-1)^{l-2n+q+r+1)} = (-1)^{l+q} = (-1)^{l-q}(-1)^{2q} = (-1)^{l-q}$. Thus:

$$\begin{aligned}
\delta_5^2 &= 4(L[n-q-1]L[2n-q-r-1] + L[q+r-n]L[q]+ \\
&+ L[q]L[n+q-r] + L[r-q-1]L[n-q-1]))^2 = \\
&= 4((-1)^{l-q}L[q](-1)^{l-q}L[r+q-n] + L[q+r-n]L[q]+ \\
&+ L[q](-1)^{l-q}L[r-q-1] + L[r-q-1](-1)^{l-q}L[q])^2 = \\
&= 4(2L[q]L[r+q-n] + 2L[q]L[r-q-1](-1)^{l-q})^2 = \\
&= 16L[q]^2(L[r+q-n] + L[r-q-1](-1)^{l-q})^2 = \\
&= 16(L[r+q-n]^2 + (L[r-q-1](-1)^{l-q})^2+ \\
&+ 2L[r+q-n]L[r-q-1](-1)^{l-q}) = \\
&= 32 + 32L[r+q-n]L[r-q-1](-1)^{l-q}.
\end{aligned} \tag{5.24}$$

Finally, we simplify δ_6^2:

$$\begin{aligned}
\delta_6^2 &= 4(L[n-q-1]L[2n-q-r-1]+L[q+r-n]L[q])^2 = \\
&= 4(L[n-q-1]^2L[2n-q-r-1]^2+L[q+r-n]^2L[q]^2+ \\
&+2L[n-q-1]L[2n-q-r-1]L[q+r-n]L[q]) = \\
&= 4(2+2(-1)^{l-q}L[q](-1)^{l-q}L[r+q-n]L[q+r-n]L[q] = \\
&= 4(2+2(-1)^{2(l-q)}L[q]^2L[q+r-n]^2) = 16.
\end{aligned} \tag{5.25}$$

We have:

$$\begin{aligned}
\mathbb{E}(L^q)-\mathbb{E}(L) &= \sum_{j=1}^{6}\sum_{i\in\mathbb{D}(\delta_j)}(2S_i\delta_j+\delta_j^2) = \\
&= \sum_{j\in\{1,2,4\}}\sum_{i\in\mathbb{D}(\delta_j)}(2S_i\delta_j+\delta_j^2)+\sum_{j\in\{3,5,6\}}\sum_{i\in\mathbb{D}(\delta_j)}(2S_i\delta_j+\delta_j^2).
\end{aligned} \tag{5.26}$$

However, since δ_j, for $j\in\{1,2,4\}$ is 0, we have:

$$\begin{aligned}
\mathbb{E}(L^q)-\mathbb{E}(L) &= \sum_{j\in\{3,5,6\}}\sum_{i\in\mathbb{D}(\delta_j)}(2S_i\delta_j+\delta_j^2) = \\
&= \sum_{r=q+1,r\neq 2q+1}^{n-q-1}(2S_r\delta_3+\delta_3^2)+\sum_{r=n-q,r\neq 2q+1}^{n-1}(2S_r\delta_5+\delta_5^2)+ \\
&+\sum_{r\geq n-q,r\leq n-1,r=2q+1}(2S_r\delta_6+\delta_6^2).
\end{aligned} \tag{5.27}$$

and:

$$\begin{aligned}
\delta_3 &= -2\,(L[q]L[n+q-r]+L[r-q-1]L[n-q-1]) = \\
&= -2\left(L[q](-1)^{l-q}L[r-q-1]+L[r-q-1](-1)^{l-q}L[q]\right) = \\
&= -4(-1)^{l-q}L[q]L[r-q-1] = -4\sigma L[q]\varepsilon_1.
\end{aligned} \tag{5.28}$$

$$\begin{aligned}
\delta_5 &= -2(L[n-q-1]L[2n-q-r-1]+L[q+r-n]L[q]+ \\
&+L[q]L[n+q-r]+L[r-q-1]L[n-q-1]) = \\
&= -4(L[q]L[r+q-n]+L[q]L[r-q-1](-1)^{l-q}) = \\
&= -4L[q](L[r+q-n]+L[r-q-1](-1)^{l-q}) = \\
&= -4L[q](\varepsilon_2+\varepsilon_1\sigma).
\end{aligned} \tag{5.29}$$

$$\begin{aligned}\delta_6 &= -2(L[n-q-1]L[2n-q-r-1]+L[q+r-n]L[q]) = \\ &= -2((-1)^{l-q}L[q](-1)^{l-q}L[q+r-n]+L[q+r-n]L[q]) = \\ &= -4(-1)^{l-q}L[q]L[q+r-n] = -4\sigma L[q]\varepsilon_2.\end{aligned} \tag{5.30}$$

we could substitute and further simplify the difference between the merit factors of L^q and L, i.e:

$$\begin{aligned}\mathbb{E}(L^q)-\mathbb{E}(L) &= \sum_{r=q+1,r\neq 2q+1}^{n-q-1}(2S_r(-4\sigma L[q]\varepsilon_1)+16)+ \\ &+ \sum_{r=n-q,r\neq 2q+1}^{n-1}(2S_r(-4L[q](\varepsilon_2+\varepsilon_1\sigma))+32+32\varepsilon_2\varepsilon_1\sigma)+ \\ &+ \sum_{r\geq n-q,r\leq n-1,r=2q+1}(2S_r(-4\sigma L[q]\varepsilon_2)+16).\end{aligned} \tag{5.31}$$

However, if we use κ, where $\kappa = -8S_rL[q]$:

$$\begin{aligned}\mathbb{E}(L^q)-\mathbb{E}(L) &= \sum_{r=q+1,r\neq 2q+1}^{n-q-1}(-8S_r\sigma L[q]\varepsilon_1+16)+ \\ &+ \sum_{r=n-q,r\neq 2q+1}^{n-1}(-8S_rL[q](\varepsilon_2+\varepsilon_1\sigma))+32+32\varepsilon_2\varepsilon_1\sigma)+ \\ &+ \sum_{r\geq n-q,r\leq n-1,r=2q+1}(-8S_r\sigma L[q]\varepsilon_2+16) = \sum_{r=q+1,r\neq 2q+1}^{n-q-1}(\kappa\sigma\varepsilon_1+16)+ \\ &+ \sum_{r=n-q,r\neq 2q+1}^{n-1}(\kappa(\varepsilon_2+\varepsilon_1\sigma))+32+32\varepsilon_2\varepsilon_1\sigma)+\sum_{r\geq n-q,r\leq n-1,r=2q+1}(\kappa\sigma\varepsilon_2+16).\end{aligned} \tag{5.32}$$

□

In Algorithm 8 a pseudo-code of the derivative function is given. The input of the function consists of a bit position q to be flipped, a skew-symmetric sequence L with an odd length $n = 2l+1$, as well as the corresponding sidelobe array S. We recall that besides the bit position q, s.t. $q < l$, the bit position $n-q-1$ is flipped as well, to keep the skew-symmetric property of the binary sequence. The output of the function consists of a single integer value Δ, which corresponds to the difference between the energies of L and L^q. In other words, if $\Delta < 0$, then the energy of the sequence L^q is lower than the merit factor of the sequence

Table 5.1 A comparison between the memory required by the tau table and the memory required by the proposed in-memory flip algorithm.

n	The memory required by using the tau table	The memory required by using the proposed method
256	256.0 KB	1.0 KB
512	1.0 MB	2.0 KB
1024	4.0 MB	4.0 KB
5000	95.37 MB	19.53 KB
20000	1525.88 MB	78.12 KB
99999	37.25 GB	390.62 KB

L. Therefore, the merit factor of L is going to be higher than the merit factor of L^q. More formally,

$$\begin{aligned}&\Delta < 0 \implies \mathbb{E}(L^q) - \mathbb{E}(L) < 0 \implies \mathbb{E}(L^q) < \mathbb{E}(L) \implies 2\mathbb{E}(L^q) < 2\mathbb{E}(L) \implies \\ &\implies \frac{1}{2\mathbb{E}(L^q)} > \frac{1}{2\mathbb{E}(L)} \implies \frac{n^2}{2\mathbb{E}(L^q)} > \frac{n^2}{2\mathbb{E}(L)} \implies \mathbb{MF}(L^q) > \mathbb{MF}(L).\end{aligned} \tag{5.33}$$

The derivative function allows us to reduce the memory requirement of some state-of-the-art algorithms from $O(n^2)$ to $O(n)$. In Table 5.1, a comparison between the space required by the tau table and the memory requirement by the proposed method is presented. During the calculations, an assumption was that both memory structures are comprised of integers (4 Bytes). For example, by using just one thread of the processors, the tau table corresponding to binary sequences with length 5000 would require approximately 95.37 Megabytes to be allocated for the tau table expansion routine, while the sidelobe array presented in this work would require the allocation of approximately 19.53 Kilobytes. It should be emphasized, that interchanging the tau table used by the state-of-the-art algorithms with the proposed sidelobe array structure would not impact the time complexity of the tweaked algorithm. However, from a practical point of view, the significant memory reduction could greatly enhance the overall time performance of a tweaked algorithm, since the size of the sidelobe array could be usually saved inside the CPU cache layers, instead of saving it to the slower memory banks. Furthermore, interchanging the tau table with the proposed sidelobe array could allow the multithreading capabilities of modern CPUs, and even GPUs, to be fully utilized.

For example, we have implemented a lightweight version of the lssOrel algorithm [24] with the tau table reduced. The pseudo-code of the enhanced implementation is given in Algorithm 9. The following notations were used:

- Ψ - a binary sequence with length n.
- Ω_Ψ - the corresponding sidelobe array of Ψ - the replacement of the tau table.
- $\mathbb{H}$ - a set of fingerprints, or hashes, of visited candidates.
- $\mathbb{T}_i$ - an inner threshold value. When the inner counter w_i reaches $\mathbb{T}_i$, the set is flushed and the whole routine restarts. The threshold value $\mathbb{T}_i$ constrains the size of the set $\mathbb{H}$.
- $\mathbb{T}_o$ - an outer threshold value. When the outer counter w_o reaches $\mathbb{T}_o$, the program is terminated. However, $\mathbb{T}_o$ could be an expression as well.
- $\mathbb{H}$.add(hash(Ψ)) - adding the hash of the binary sequence Ψ to the set $\mathbb{H}$.
- $C(\Omega_\Psi)$ - the cost function, i.e. the sum of the squares of all elements in the sidelobe array Ω_Ψ, which is equal to the energy of Ψ, or $\mathbb{E}(\Psi)$.
- pickBestNeighbor($\Psi, \Omega_\Psi, \mathbb{H}$) - a function, which returns the index of the best unexplored neighbor of Ψ, i.e. the binary sequence Ψ^f with a distance of exactly 1 flip away from Ψ, s.t. hash(Ψ^f) does not belong to the set $\mathbb{H}$. The pseudo-code of this helper function is given in Algorithm 10.

Several notations were used throughout the pseudo-code presentation shown in Algorithm 10.

- $\mathbb{MAX}$ - the maximum possible value, which the type of the variable **bestDelta** could hold. For example, if the variable **bestDelta** is of type **INT** (4 Bytes) then $\mathbb{MAX} = 7FFFFFFF_{16} = 2,147,483,647$
- $\mathbb{P}, \mathbb{Q}$ - two odd prime numbers, which are used to calculate the hash of the binary sequence. During our experiments, they were fixed to $\mathbb{P} = 315223$ and $\mathbb{Q} = 99041$. It should be noted, that no additional efforts were made to find better, in terms of hash collision false positives or false negatives rates, values of $\mathbb{P}$ and $\mathbb{Q}$.

Algorithm 9 was implemented (C++) on a general-purpose computer equipped with a budget processor Xeon-2640 CPU, having a base frequency of 2.50 GHz. A skew-symmetric binary sequence with length 449 and a record-breaking merit factor of 6.5319 was found after approximately one day. It should be noted that all 12 threads of the CPU were launched in parallel. As a comparison, the currently known optimal results (a merit factor of 6.5218) were acquired by using the Slovenian Initiative for National Grid (SLING) infrastructure (100 processors) and 4-day threshold limitation [23]. The binary sequence is given in a hexadecimal format in Table 5.2.

Algorithm 8 Lightweight flip probing of skew-symmetric binary sequences in linear both time and memory complexities

```
function DERIVATIVE(q, L, S)
  Δ = 0
  σ = (−1)^(l−q)
  for r = 1; r < n − 1; r+ = 2 do
    if r ≤ q then
      continue
    end if
    κ = −8 S_r L[q]
    ε1 = L[r − q − 1]
    ε2 = L[q + r − n]
    if r < n − q then
      if q ≠ r − q − 1 then
        Δ = Δ + 16 + κσε1
      end if
    else
      if q ≠ r − q − 1 then
        Δ = Δ + 32 + κ(ε2 + ε1σ) + 32ε2ε1σ
      else
        Δ = Δ + 16 + κσε2
      end if
    end if
  end for
  return Δ
end function
```

Algorithm 9 Heuristic algorithm, with tau table reduction, searching for binary skew-symmetric sequences with a high merit factor.

```
procedure MF(n, 𝕋_i, 𝕋_o)
  bestMF, w_o ← 0,0
  while True do
    ℍ, w_i, ← {∅}, 0
    Ψ ← random
    ℍ.add(hash(Ψ))
    V ← C(Ω_Ψ)
    while True do
      bestN ← pickBestNeighbor(Ψ, Ω_Ψ, ℍ)
      if bestN == −1 then
        break
      end if
      Flip(bestN, Ψ, Ω_Ψ)
      V ← C(Ω_Ψ)
      w_i += 1
      ℍ.add(hash(Ψ))
      if n^2/(2V) > bestMF then
        bestMF ← n^2/(2V)
      end if
      if w_i > 𝕋_i then
        w_o += 1
        break
      end if
    end while
    if w_o > 𝕋_o then
      break
    end if
  end while
end procedure
```

Algorithm 10 Pseudo-code of the helper function pickBestNeighbor

```
function PICKBESTNEIGHBOR(Ψ, Ω_Ψ, ℍ)
  bestN = -1
  bestDelta = 𝕄𝔸𝕏
  for q = 0; q < ⌈n/2⌉; q++ do
    δ = Derivative(q, Ψ, Ω_Ψ)
    if δ ≤ bestDelta then
      hash=ℙ
      for i = 0; i < ⌈n/2⌉; i++ do
        if q == i then
          hash=hash∗ℚ - Ψ[i]
        else
          hash=hash∗ℚ + Ψ[i]
        end if
      end for
      if hash(Ψ) ∈ ℍ then
        continue
      end if
      bestDelta = δ
      bestN=q
    end if
  end for
  return bestN
end function
```

Table 5.2 An example of a skew-symmetric binary sequence with length 449 and a record merit factor found by Algorithm 9. The sequence is presented in HEX with leading zeroes omitted.

n	Sequence in HEX	MF
449	96f633d86fe825794ed23a9dfd7d4c3 abd080cf76cbf9bdab9a7b2533e3161 901d1950c774ca8bd012cfd7d5d8123 c4f97e285469d327478	6.5319

It should be emphasized that the flip operation for the middle index of the skew-symmetric binary sequence Ψ is not permitted. However, this is not affecting the search space by cutting some parts of it. Indeed, let's define the binary sequence $\mathbb{B} = b_1 b_2 \cdots b_l M b_{l+1} b_{l+2} \cdots b_{2l}$ of length $n = 2l + 1$ and the binary sequence $\overline{\mathbb{B}}$ as the binary sequence $\mathbb{B}$ with all the bits flipped, i.e. $\overline{\mathbb{B}} = \overline{b_1 b_2} \cdots \overline{b_l} \overline{M} \overline{b_{l+1} b_{l+2}} \cdots \overline{b_{2l}}$. It could be easily shown that all sidelobes of $\mathbb{B}$ and $\overline{\mathbb{B}}$ are identical. Indeed,

$$C_u(\overline{\mathbb{B}}) = \sum_{j=0}^{n-u-1} \overline{b_j b_{j+u}} = \sum_{j=0}^{n-u-1} (-1)^2 b_j b_{j+u} = C_u(\mathbb{B}).$$

5.1.1 On the Bernasconi Conjecture

As previously discussed, in [14] Bernasconi conjectured that stochastic search procedures will not yield merit factors higher than 5 for long sequences (greater than 200). It should be mentioned that this prediction was made in 1987. Since then, many years have passed and pieces of evidence that stochastic search procedures could perform better than the prediction's expectations were found. Indeed, heuristic algorithms that could find odd binary sequences with lengths up to about 500 and merit factors greater than 5 were discovered. However, the Bernasconi conjecture appears valid when the threshold of the binary sequence's length is updated and lifted. Since during the last 35 years the computational capabilities of modern CPUs are rising almost exponentially such actualization would be fair. However, if a stochastic search procedure is found, a procedure that could reach extremely long binary sequences with merit factors greater than 5, by using a mid-range general-purpose computer, then the barriers predicted by Bernasconi could be very pessimistic.

Some more experiments were made by using Algorithm 9 and skew-symmetric binary sequences with lengths greater than 1000. For example, within several seconds, a binary sequence with length 1001 and a merit factor greater than 5 was discovered. By leaving the routine for a minute, binary sequences with merit factors up to 5.65 were reached. Then, within several seconds as well, a binary sequence with a length of 2001 and a merit factor greater than 5 was discovered. However, this time the routine needed almost an hour to reach binary sequences with merit factors up to 5.40. When the length is increased to 5001, the algorithm required half a day to reach a binary sequence with MF greater than 5.10. Finally, the algorithm failed to reach a binary sequence with length 10001 and a merit factor greater than 5 within 24 hours (by using all the twelve threads of the processor). The numerical experiments suggest that Algorithm 9 is not able to find binary sequences with lengths greater than 10000 and merit factor greater than 5.

Indeed, the Algorithm 9 property of avoiding Hasse cycles, or the self-avoiding walk (SAW) property, yields binary sequences with near-optimal merit factors. However, the efficiency of this strategy melts away when binary sequences with bigger lengths are used. This is not surprising, since the bigger the length, the larger the search space is. For example, the search space of the set of all skew-symmetric binary sequences with length 10001 is 2^{5001}. More importantly, several more computational burdens were introduced by Algorithm 9 itself:

- The pickBestNeighbor function (see Algorithm 10) is looking for the best neighbor of the current binary sequence Ψ. Thus, each calling of the function would trigger the Derivative function exactly n times.
- As previously discussed, Algorithm 9 is using a hashing technique to keep an unordered set of the already visited notes. Such an approach is causing a significant computation burden to the algorithm for larger values of n:
    1. The unordered set strategy requires at least $\mathbb{G}\mathbb{X}n\mathbb{T}_o$ bytes of memory, where $\mathbb{G}$ is the count of the threads used by the processor, while $\mathbb{X}$ is the size in bytes of the used variable type.
    2. Frequently, when a candidate Ψ^q with lower score δ is found (see line 6 from Algorithm 10), a hash of the candidate should be calculated, so to be further checked was the binary sequence Ψ^q met before.

To annihilate all the aforementioned computational burdens, an Algorithm 11 is proposed. In summary, the following simplifications were introduced:

1. The pickBestNeighbor function straightforwardly accepts the first met neighbor having a strictly better score.

2. By introducing the previous tweak, the algorithm cycle trapping is avoided. It should be noted that if small values of n are used, this could greatly worsen the quality, in terms of the high merit factor, of the binary sequences found. However, when considering larger values of n, the numerical experiments suggest that this tweak could be highly efficient. Thus, the need of using an unordered set could be annihilated and the memory complexity of the algorithm significantly reduced.

3. Since the unordered set was annihilated, the hash routines are removed as well.

In Algorithm 11 the following notations were used:

- $\mathbb{T}$ - the threshold value of the instance.
- C - the cost function.
- V, V^* - respectively the current best and the overall best score values.
- c - the counter. The algorithm quits if the counter c reaches the threshold $\mathbb{T}$.
- $\mathbb{L}$, $\mathbb{G}$ - binary variables: $\mathbb{L}$ (local) is activated if V is improved, while $\mathbb{G}$ (global) is activated if V^* is improved.
- Quake function - the function flips $\mathbb{Q}$ random bits in Ψ.

During our experiments, by using Algorithm 11, we were able to reach skew-symmetric binary sequences with lengths up to 100,001 and merit factors greater than 5. However, the greater the length of the binary sequence is, the larger the value of $\mathbb{Q}$ should be. Some of those $\mathbb{Q}$ values, used during our experiments, are given in Table 5.3. It should be emphasized, that those $\mathbb{Q}$ values guarantee to reach a skew-symmetric binary sequence with merit factors greater than 5.0, but it is highly unlikely that exactly those values would yield the best results.

For example, by using Algorithm 11, a binary sequence with length 10,001 and a merit factor greater than 5 was reached for approximately one minute. Leaving the algorithm for another minute would reach merit factors of 5.10 and higher. Doubling the length of the binary sequence to 20,001 required from Algorithm 11 approximately 4 minutes to reach a skew-symmetric binary sequence with a merit factor greater than 5.

Binary sequences with a length of 50,001 and a merit factor greater than 5 were reached for leaving the algorithm for approximately 40 minutes, while binary sequences with a length of 100,001 and a merit factor greater than 5 were reached for approximately 5 hours. However, it should be emphasized that the larger the sequence, the larger the number of quakes $\mathbb{Q}$ should be. In Table 5.3 the values of $\mathbb{Q}$ corresponding to the binary sequences'

Algorithm 11 A heuristic algorithm, with a tau table, unordered set, and hashing routines reduced, for searching long skew-symmetric binary sequences with a high merit factor. Both the time and memory complexity of the algorithm are $O(n)$.

```
procedure SHC(n, 𝕋)
  Ψ ← random
  V*, V, 𝔾, 𝕃, c ← C(Ω_Ψ), 0, True, False, 0
  while c < 𝕋 do
    c += 1
    if 𝔾 then
      pick random r ∈ [0, ⌊n/2⌋)
      for q ∈ [0, ⌊n/2⌋) do
        δ = Derivative((r + q) mod ⌊n/2⌋, Ψ, Ω_Ψ)
        if δ > 0 then
          continue
        end if
        Flip((r + q) mod ⌊n/2⌋, Ψ, Ω_Ψ)
        V += δ
        if V* > C(Ω_Ψ) then
          V*, 𝕃 ← C(Ω_Ψ), True
          break
        else
          Flip((r + q) mod ⌊n/2⌋, Ψ, Ω_Ψ)
        end if
      end for
      if 𝕃 then
        𝔾, 𝕃 ← True, False
        continue
      else
        𝔾 ← False
      end if
    else
      Quake(ℚ, Ψ, Ω_Ψ)
      𝔾, 𝕃 ← True, False
    end if
  end while
end procedure
```

lengths used throughout the experiments are given. Small cuts from the history of the search traces are provided within the four complimentary files. Each file holds skew-symmetric binary sequences of fixed length - $2^4 5^4 + 1$, $2^5 5^4 + 1$, $2^4 5^5 + 1$ or $2^5 5^5 + 1$. All sequences possess merit factors greater than 5.

Table 5.3 The number of quakes used throughout our experiments.

Length n	Quake $\mathbb{Q}$
999	1
1499	2
1999	3
2999	4
4999	6
10001	14
20001	30
50001	70
100001	160

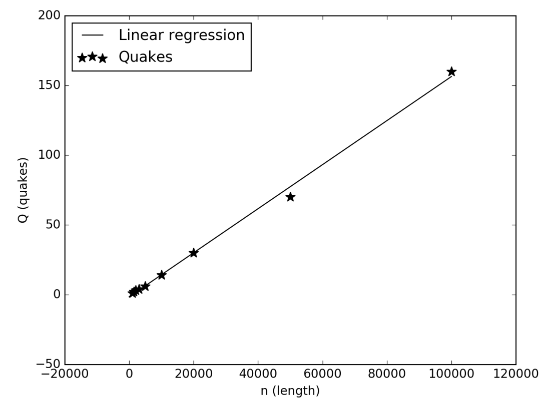


Fig. 5.1 A linear regression made to all the $(n, \mathbb{Q})$ pairs from Table 5.3. The equation representing the linear fit is $\mathbb{Q} = 0.001578787n - 1.546093$.

The numerical experiments suggest that the value of $\mathbb{Q}$ grows linear with the length of the binary sequence. This is visible in Figure 5.1. The time required (in seconds) to reach binary sequences with a merit factor strictly greater than 5 is given in Figure 5.2. As expected, the time required to reach a binary sequence with a merit factor greater than 5 grows quadratic with the size of the binary sequences n.

Both the regression models are rough approximations of the algorithm's behavior. For a more precise estimation - more instances of the algorithm should be analyzed. However, one very important property of Algorithm 11 should be further highlighted. When a counter to the function Quake is attached, during the optimization routine a total of approximately 2000-2500 calls to the function are made before a binary sequence with a merit factor greater than 5 is reached. This observation, as well as the numerical pieces of evidence found through our experiments, suggest that given an arbitrary binary sequence $\mathbb{B}$ with length n, and by using a general-purpose computer with 12 threads, as well as C++ implementation of Algorithm 11 launched with variable $\mathbb{Q}$ close to $\lceil 0.001578787n - 1.546093 \rceil$, $\mathbb{B}$ could be optimized to a binary sequence with merit factor greater than 5, after an approximately $177.2867 - 0.0562043n + 0.000002340029n^2$ seconds.

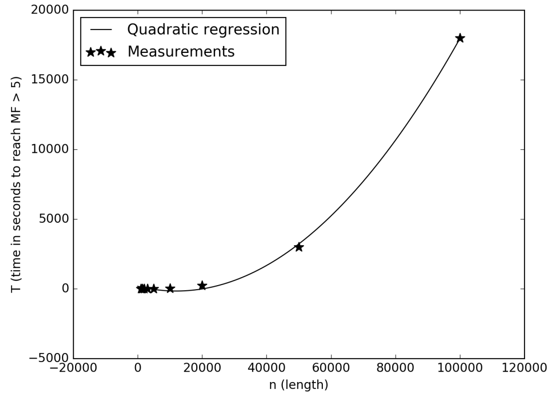


Fig. 5.2 A quadratic regression of all $(n, \mathbb{T})$ measurements. The equation representing the quadratic fit is $\mathbb{T} = 177.2867 - 0.0562043n + 0.000002340029n^2$.

5.1.2 New Classes of Binary Sequences with High Merit Factor

Despite the rich results regarding the skew-symmetric binary sequences, the search for binary sequences with even lengths and high MF was scarcely researched. This is not surprising, since the sieving proposed by Golay applies to odd-length sequences only.

In this section, motivated by the absence of computationally efficient sieving for binary sequences with even lengths and high merit factor values, several new classes of binary sequences are proposed. We start with the definition of a class of finite binary sequences, called pseudo-skew-symmetric, with alternate auto-correlation absolute values equal to one. The class is defined by using sieving suitable for even-length binary sequences. Then, by using some mathematical observations, we show how state-of-the-art algorithms for searching skew-symmetric binary sequences with high merit factor and length $2n + 1$ could be easily converted to algorithms searching pseudo-skew-symmetric binary sequences with high merit factor and lengths $2n$ or $2n+2$. More importantly, this conversion does not degrade the performance of the modified algorithm.

Then, by using number partitions [6], an additional sieving strategy for both skew-symmetric and pseudo-skew-symmetric sequences is proposed. A method of finding sub-classes of binary sequences with high MF is further discussed. The experiments revealed that the classes defined in this section are highly promising. By using a single mid-range computer, we were able to improve all records for skew-symmetric binary sequences with lengths above 225, which were recently reached by various algorithms and a supercomputer grid. We further revealed that binary sequences with even or odd length n, for $n \leq 2^8$, and with merit factor strictly greater than 8, and binary sequences with even or odd length n, for $n \leq 2^9$ and with a merit factor strictly greater than 7 do exist.

Finally, we demonstrate the efficiency of the proposed algorithm by launching it on two extremely hard search spaces of binary sequences of lengths 573 and 1009. The choice of those two specific lengths is motivated by the approximation numbers given in [24], Figure 7, showing how much time the state-of-the-art stochastic solver lssOrel_8 need to reach binary sequences with the aforementioned lengths and merit factors close to 6.34. More precisely, it was estimated that finding solutions with a merit factor of 6.34 for a binary sequence with length 573 requires around 32 years, while for a binary sequence with length 1009, the average runtime prediction to reach the merit factor of 6.34 is 46774481153 years. By using the proposed in this section algorithm, we were able to reach such candidates within several hours.

Definition 5.1.1 (Pseudo-Skew-Symmetric Binary Sequence)**.** We call a given sequence $P = a||X = Y||b$ a pseudo-skew-symmetric binary sequence, if either X or Y are skew-symmetric binary sequences, for some $a \in \{-1, 1\}$ or $b \in \{-1, 1\}$.

Proposition 5.1.1. The sidelobes array of pseudo-skew-symmetric binary sequences consists of alternating $\pm$ ones.

Proof 5.1.3. Let us denote the pseudo-skew-symmetric binary sequence as P. By definition, P could be represented as $a||A$ or $B||b$, for some skew-symmetric binary sequences A or B.

If $P = B||b$, for some skew-symmetric binary sequence $B = (b_0, b_1, \cdots, b_{n-1})$, then $P = (b_0, b_1, \cdots, b_{n-1}, b_n)$, where $b_n = b$.

Thus,

$$\hat{C}_i(P) = \sum_{j=0}^{i} b_j b_{j+n-i}, \;\; for\; i \in \{0, 1, \cdots, n\}.$$

Therefore, the sidelobes of P could be further simplified:

$$\hat{C}_i(P) = \sum_{j=0}^{i} b_j b_{j+n-i} = \sum_{j=0}^{i-1} b_j b_{j+n-i} + b_i b_{i+n-i} = \sum_{j=0}^{i-1} b_j b_{j+n-i} + b_i b_n = \hat{C}_{i-1}(B) + b_i b_n.$$

The last substitution arises from the definition of the sidelobe array:

$$\hat{C}_{i-1}(B) = \sum_{j=0}^{i-1} b_j b_{j+n-(i-1)-1} = \sum_{j=0}^{i-1} b_j b_{j+n-i}.$$

The sidelobe array of P, denoted as S_P, could be then simplified:

$$\begin{aligned} S_P &= \left[\hat{C}_0(P), \hat{C}_1(P), \cdots, \hat{C}_{n-1}(P), \hat{C}_n(P)\right] = \\ &= \left[\hat{C}_0(P), \hat{C}_0(B) + b_1 b_n, \hat{C}_1(B) + b_2 b_n, \cdots, \hat{C}_{n-2}(B) + b_{n-1} b_n, \hat{C}_{n-1}(B) + b_n b_n\right] = \\ &= \left[b_0 b_n, \hat{C}_0(B) + b_1 b_n, b_2 b_n, \cdots, b_{n-1} b_n, \hat{C}_{n-1}(B) + b_n b_n\right]. \end{aligned} \tag{5.34}$$

Note that $\hat{C}_x(B) = 0$, for odd values of x. Since $b_i \in \{-1, 1\}$, $\hat{C}_x(P) = b_x b_n = \pm 1$, for even values of x, which completes the proof of the first case, or more formally:

$$S_P = \left[\pm 1, \hat{C}_0(B) + b_1 b_n, \pm 1, \cdots, \pm 1, \hat{C}_{n-1}(B) + b_n b_n\right].$$

Let us consider the second case. If $P = a||A$, then $P^{rev} = A^{rev}||a$ will possess the same sidelobes array as P, where A^{rev} denotes the reversed version of a given binary sequence A. Since A is skew-symmetric sequence, A^{rev} is a skew-symmetric as well. Thus, by applying the first case, we have:

$$S_P = S_{P^{rev}} = \left[\pm 1, \hat{C}_0(A^{rev}) + a_1 a_n, \pm 1, \cdots, \pm 1, \hat{C}_{n-1}(A^{rev}) + a_n a_n\right],$$

where $A^{rev} = (a_0, a_1, \cdots, a_{n-1}, a_n)$ and $a_n = a$, which completes the proof. □

This property is beneficial for the energy $E(P)$ of the pseudo-skew-symmetric binary sequence P. Indeed,

$$\mathbb{E}(P) = \sum_{u=0}^{n-1} \hat{C}_u(P)^2 = \sum_{u=0, u_{even}}^{n-1} \hat{C}_u(P)^2 + \sum_{u=0, u_{odd}}^{n-1} \hat{C}_u(P)^2 =$$

$$= \sum_{u=0, u_{even}}^{n-1} \pm 1^2 + \sum_{u=0, u_{odd}}^{n-1} \hat{C}_u(P)^2 = \lfloor \frac{n}{2} \rfloor + \sum_{u=1, u_{odd}}^{n} \hat{C}_u(P)^2.$$

The following property allows us to convert an existing algorithm for searching skew-symmetric binary sequences with high merit factor to an algorithm searching pseudo-skew-symmetric binary sequences and high merit factor.

Proposition 5.1.2. Given a skew-symmetric binary sequence $B = (b_0, b_1, \cdots, b_{n-1})$ with sidelobes array

$$S_B = \left[\hat{C}_0(B), \hat{C}_1(B), \cdots, \hat{C}_{n-2}(B), \hat{C}_{n-1}(B)\right],$$

the following property holds:

$$\mathbb{E}(P) = \mathbb{E}(B) + n + 2b_n\delta,$$

where P is the pseudo-skew-symmetric sequence $B||b_n$ and $\delta = \sum_{u=0,u_{even}}^{n-2} \hat{C}_u(B) b_{u+1}$.

Proof 5.1.4. Using the result from the previous proposition proof we have:

$$S_P = \left[\pm 1, \hat{C}_0(B) + b_1 b_n, \pm 1, \cdots, \pm 1, \hat{C}_{n-1}(B) + b_n b_n\right].$$

By using the definition of energy of a binary sequence we have:

$$\begin{aligned}
&\mathbb{E}(P) - \mathbb{E}(B) = \sum_{u=0}^{n-1} \hat{C}_u(P)^2 - \sum_{u=0}^{n-2} \hat{C}_u(B)^2 = 1 + \sum_{u=1}^{n-1} \hat{C}_u(P)^2 - \sum_{u=0}^{n-2} \hat{C}_u(B)^2 = \\
&1 + \sum_{u=0}^{n-2} \hat{C}_{u+1}(P)^2 - \hat{C}_u(B)^2 = \\
&= 1 + \sum_{u=0,u_{even}}^{n-2} \hat{C}_{u+1}(P)^2 - \hat{C}_u(B)^2 + \sum_{u=0,u_{odd}}^{n-2} \hat{C}_{u+1}(P)^2 - \hat{C}_u(B)^2 = \\
&= 1 + \sum_{u=0,u_{even}}^{n-2} \left(\hat{C}_u(B) + b_{u+1}b_n\right)^2 - \hat{C}_u(B)^2 + \sum_{u=0,u_{odd}}^{n-2} \pm 1^2 - 0^2 = \\
&= 1 + \sum_{u=0,u_{even}}^{n-2} \left(\hat{C}_u(B) + b_{u+1}b_n\right)^2 - \hat{C}_u(B)^2 + \lfloor \frac{n-1}{2} \rfloor = \\
&= 1 + \sum_{u=0,u_{even}}^{n-2} 2\hat{C}_u(B) b_{u+1} b_n + (b_{u+1}b_n)^2 + \lfloor \frac{n-1}{2} \rfloor = \\
&= 1 + \sum_{u=0,u_{even}}^{n-2} 2\hat{C}_u(B) b_{u+1} b_n + \sum_{u=0,u_{even}}^{n-2} (b_{u+1}b_n)^2 + \lfloor \frac{n-1}{2} \rfloor = \\
&= 1 + \sum_{u=0,u_{even}}^{n-2} 2\hat{C}_u(B) b_{u+1} b_n + \lfloor \frac{n-1}{2} \rfloor + \lfloor \frac{n-1}{2} \rfloor = n + 2b_n \sum_{u=0,u_{even}}^{n-2} \hat{C}_u(B) b_{u+1}. \quad \square
\end{aligned} \tag{5.35}$$

The last property is of significant importance when converting an algorithm searching for skew-symmetric binary sequences, denoted as $\mathscr{A}$, to an algorithm searching for pseudo-skew-symmetric binary sequences $\mathscr{B}$ and a high merit factor. Indeed, despite the complexity of algorithm $\mathscr{A}$ we can decompose it to a tape $\cdots||\mathbb{L}_1||\cdots||\mathbb{L}_2||\cdots||\mathbb{L}_n||\cdots$, where $\mathbb{L}_i$ are stages of $\mathscr{A}$, where better candidates could be announced. They are known as local optimums in heuristic search literature. We could easily replace $\mathbb{L}_i$ with $\mathbb{L}_i||\mathbb{T}_i$, where $\mathbb{T}_i$ is a simple routine with memory and time complexity of $O(n)$, which calculates the pseudo-skew-symmetric sequences $L_i||1$ and $L_i||-1$ merit factors, where L_i is the current best candidate. It should be noted that $\mathscr{B} = \cdots||\mathbb{L}_1||\mathbb{T}_1||\cdots||\mathbb{L}_2||\mathbb{T}_2||\cdots||\mathbb{L}_n||\mathbb{T}_n||\cdots$ does not interfere with the normal work of $\mathscr{A}$ by design. Furthermore, since those linear time complexity checkups are initiated on local optimums only, the delay of $\mathscr{B}$ compared to $\mathscr{A}$ caused by the additional instructions $\mathbb{T}_i$ is negligible.

We could further extend the search of highly-competitive pseudo-skew-symmetric sequences by the following observation:

Proposition 5.1.3. Given a skew-symmetric binary sequence $B = b_0||B^{'}||b_{n-1}$ both binary sequences $b_0||B^{'}$ and $B^{'}||b_{n-1}$ are pseudo-skew-symmetric.

Proof 5.1.5. From the main property of the skew-symmetric sequences follows that $B^{'}$ is skew-symmetric as well. Thus, the pseudo-skew-symmetry of $b_0||B^{'}$ and $B^{'}||b_{n-1}$ follows directly from definition 5.1.1. $\square$

Proposition 5.1.4. Given a skew-symmetric binary sequence $B = (b_0, b_1, \cdots, b_{n-1}) = b_0||B^{'}||b_{n-1}$ with sidelobes array

$$S_B = \left[\hat{C}_0(B), \hat{C}_1(B), \cdots, \hat{C}_{n-2}(B), \hat{C}_{n-1}(B)\right],$$

the following property holds:

$$\mathbb{E}(P) = \mathbb{E}(B) + n - 3 + 2b_{n-1}\delta,$$

where P is the pseudo-skew-symmetric sequence $b_0||B^{'}$ and $\delta = \sum_{u=1, u_{even}}^{n-2} -\hat{C}_u(B)b_u$.

Proof 5.1.6. We have $B = (b_0, b_1, \cdots, b_{n-1})$ and $P = (b_0, b_1, \cdots, b_{n-2})$. Furthermore,

$$\hat{C}_i(P) = \sum_{j=0}^{i} b_j b_{j+n-2-i}, \;\; for\; i \in \{0, 1, \cdots, n-2\}.$$

Decomposing $\hat{C}_i(B)$ reveals the following:

$$\hat{C}_i(B) = \sum_{j=0}^{i} b_j b_{j+n-1-i} = \sum_{j=0}^{i-1} b_j b_{j+n-1-i} + b_i b_{i+n-1-i} = b_i b_{n-1} + \hat{C}_{i-1}(P)$$

In other words, $\hat{C}_{i-1}(P) = \hat{C}_i(B) - b_i b_{n-1}$. Thus, by using the sidelobes array of B,

$$S_B = \left[\hat{C}_0(B), 0, \hat{C}_2(B), 0, \cdots, 0, \hat{C}_{n-3}(B), 0, \hat{C}_{n-1}(B)\right],$$

we could represent the sidelobes array of P:

$$\begin{aligned} S_P &= \left[\hat{C}_0(P), \hat{C}_1(P), \hat{C}_2(P), \cdots, \hat{C}_{n-2}(P)\right] = \\ &= \left[\hat{C}_1(B) - b_1 b_{n-1}, \hat{C}_2(B) - b_2 b_{n-1}, \hat{C}_3(B) - b_3 b_{n-1}, \cdots, \hat{C}_{n-1}(B) - b_{n-1} b_{n-1}\right]. \end{aligned} \tag{5.36}$$

By using the definition of energy of a binary sequence we have:

$$\begin{aligned} \mathbb{E}(P) - \mathbb{E}(B) &= \sum_{u=0}^{n-3} \hat{C}_u(P)^2 - \sum_{u=0}^{n-2} \hat{C}_u(B)^2 = \sum_{u=0}^{n-3} \hat{C}_u(P)^2 - \left(1 + \sum_{u=1}^{n-2} \hat{C}_u(B)^2\right) = \\ &= -1 + \sum_{u=1}^{n-2} \hat{C}_{u-1}(P)^2 - \hat{C}_u(B)^2 = \\ &= -1 + \sum_{u=1, u_{even}}^{n-2} \hat{C}_{u-1}(P)^2 - \hat{C}_u(B)^2 + \sum_{u=1, u_{odd}}^{n-2} \hat{C}_{u-1}(P)^2 - \hat{C}_u(B)^2 = \\ &= -1 + \sum_{u=1, u_{even}}^{n-2} \left(\hat{C}_u(B) - b_u b_{n-1}\right)^2 - \hat{C}_u(B)^2 + \sum_{u=1, u_{odd}}^{n-2} \pm 1^2 - 0^2 = \\ &= -1 + \sum_{u=1, u_{even}}^{n-2} \left(-2\hat{C}_u(B) b_u b_{n-1} + (b_u b_{n-1})\right)^2 + \lfloor \frac{n-2}{2} \rfloor = \\ &= -1 + \sum_{u=1, u_{even}}^{n-2} -2\hat{C}_u(B) b_u b_{n-1} + \sum_{u=1, u_{even}}^{n-2} (b_u b_{n-1})^2 + \lfloor \frac{n-2}{2} \rfloor = \\ &= -1 + \sum_{u=1, u_{even}}^{n-2} -2\hat{C}_u(B) b_u b_{n-1} + \lfloor \frac{n-2}{2} \rfloor + \lfloor \frac{n-2}{2} \rfloor = n - 3 + 2b_{n-1} \sum_{u=1, u_{even}}^{n-2} -\hat{C}_u(B) b_u. \end{aligned} \tag{5.37}$$

□

The last property further enhances the power of the algorithm. Thus now we can modify each algorithm $\mathscr{A}$, searching for skew-symmetric binary sequences with odd length n and high merit factor, to an algorithm $\mathscr{B}$, searching simultaneously skew-symmetric binary sequences with odd length n and pseudo-skew-symmetric binary sequences with even lengths $n-1$ and $n+1$.

Definition 5.1.2 (Restriction Class of Binary Sequence)**.** We will call the class of binary sequences of length n, with the first k elements fixed, a restriction class of order k on binary sequences with length n. We will denote this set as R_n^k. If the binary sequence is skew-symmetric we will use the notation $\mathscr{R}_n^k$.

It should be noted that $\mathscr{R}_n^k \subset R_n^k$. More precisely, the magnitude of R_n^k is 2^{n-k}, while the magnitude of $\mathscr{R}_n^k$ is 2^{l-k+1}, where $n = 2l+1$, since $\mathscr{R}_n^k$ is defined over the skew-symmetric binary sequences only.

A well-studied area in number theory and combinatorics is the number partition problem - distinct ways of writing a given integer number n as a sum of positive integers. We define the number of possible partitions of a non-negative integer n as the partition function $p(n)$. No closed-form expression for $p(n)$ is known. However, the partition functions for some different values of n could be found in the online encyclopedia of integer numbers (OEIS), sequence A000041 [1].

Theoretically, searching for skew-symmetric binary sequences of length n with high merit factors could be parallelized to $\mid \mathscr{R}_n^k \mid$ instances. To minimize the total number of instances needed, we should consider several actions to a given skew-symmetric binary sequence $B = (b_0, b_1, \cdots, b_{n-1})$:

- Reversing B defined as operator δ_1: $\delta_1(B) = (b_{n-1}, \cdots, b_1, b_0)$
- Complementing B defined as operator δ_2: $\delta_2(B) = (\overline{b_0}, \overline{b_1}, \cdots, \overline{b_{n-1}})$, where $\overline{b_i} = -b_i$
- Alternating complementing of B defined as operator δ_3 :
  $\delta_3(B) = (\cdots, \overline{b_{i-2}}, b_{i-1}, \overline{b_i}, b_{i+1}, \overline{b_{i+2}}, \cdots)$

All three operators leave the energy of B intact. If we further add the identity operator δ_0 we construct a group G of order 8. By using some group theory [118], we could derive a closed formula of the exact number of symmetry classes with length k: $2^{k-3} + 2^{\lfloor \frac{k}{2} \rfloor - 2 + (k \bmod 2)}$. The same formula arises from the row sums of the Losanitsch's triangle (OEIS, sequence A005418 [2]) - named after the S. Lozanić, in his work related to the symmetries exhibited by rows of paraffins [99]. This fact could be used to partition the search space from $p(k)$ covering subsets to $2^{k-3} + 2^{\lfloor \frac{k}{2} \rfloor - 2 + (k \bmod 2)}$ non-covering subsets. A similar partitioning was used in [118] to efficiently parallelize a branch and bound algorithm for exhaustively searching binary sequences with optimal merit factors. Since exhaustive search is inapplicable for large values of n, the following characteristic is proposed:

Definition 5.1.3 (Potential of a Restriction Subclass)**.** For a skew-symmetric binary sequence $B = (b_0, b_1, \cdots, b_{n-1})$, we fix a partitioning with length k: $t_0, t_1, \cdots, t_g$, s.t. $\sum_{i=0}^{g} t_i = k$. The partitioning could be projected to a skew-symmetric binary sequence with the following procedure:

$$R = \underbrace{a\cdots a}_{t_0}\underbrace{\overline{a}\cdots\overline{a}}_{t_1}\underbrace{a\cdots a}_{t_2}\underbrace{\overline{a}\cdots\overline{a}}_{t_3}\cdots\underbrace{(-1)^g a\cdots(-1)^g a}_{t_g}\underbrace{u_1u_2u_3\cdots u_{n-2k-2}u_{n-2k-1}u_{n-2k}}_{\text{non-fixed (free) elements}}\underbrace{f_1f_2f_3\cdots f_{k-2}f_{k-1}f_k}_{\text{last elements are fixed}}$$

The last k elements f_i are fixed due to the first k elements of the sequence and its skew-symmetric property. Please note that all elements $a, \overline{a}, (-1)^g a, u_i, f_i \in \{-1, 1\}$. We define the potential of the binary skew-symmetric sequence R as the energy of the ternary sequence R^z, where:

$$R^z = \underbrace{a\cdots a}_{t_0}\underbrace{\overline{a}\cdots\overline{a}}_{t_1}\underbrace{a\cdots a}_{t_2}\underbrace{\overline{a}\cdots\overline{a}}_{t_3}\cdots\underbrace{(-1)^g a\cdots(-1)^g a}_{t_g}\quad\underbrace{000\cdots000}_{n-2k\text{ zeroed elements}}\quad\underbrace{f_1f_2f_3\cdots f_{k-2}f_{k-1}f_k}_{\text{last elements are fixed}}$$

R^z is ternary since we have introduced a new element 0. This way we could not only focus on the complete sidelobes of R but take under consideration the non-complete fragments of sidelobes of R, where the fixed elements of the sequence play a role. For example, let us consider a skew-symmetric binary sequence Q with length $n = 21$, a restriction $k = 6$ and a partition $1, 1, 2, 2$:

$$Q = \underbrace{a}_{1}\underbrace{\overline{a}}_{1}\underbrace{aa}_{2}\underbrace{\overline{a}\overline{a}}_{2}\quad\underbrace{u_1u_2u_3\cdots u_9}_{\text{non-fixed (free) elements}}\quad\underbrace{f_1f_2f_3f_4f_5f_6}_{\text{elements are fixed}}$$

Since Q is skew-symmetric we know that $Q[l-i] = (-1)^i Q[l+i]$, for $n = 2l+1$. If we take $i = l$ we have $Q[0] = (-1)^l Q[n-1]$. In the current example, $n = 21$ and $l = 10$. Therefore $f_6 = Q[20] = Q[0](-1)^l = Q[0]$. By following the same routine we could reveal all values of f_i:

$$Q = \underbrace{a}_{1}\underbrace{\overline{a}}_{1}\underbrace{aa}_{2}\underbrace{\overline{a}\overline{a}}_{2}\quad\underbrace{u_1u_2u_3\cdots u_9}_{\text{non-fixed (free) elements}}\quad\underbrace{a}_{1}\underbrace{\overline{a}\overline{a}}_{2}\underbrace{aaa}_{3}$$

We could easily derive Q^z:

$$Q^z = \underbrace{a}_{1}\underbrace{\overline{a}}_{1}\underbrace{aa}_{2}\underbrace{\overline{a}\overline{a}}_{2}\underbrace{000000000}_{9}\underbrace{a}_{1}\underbrace{\overline{a}\overline{a}}_{2}\underbrace{aaa}_{3}$$

Table 5.4 A list of unique partitions in $\mathbb{R}_{21}^{6}$

Partition	± Notation
6	['+', '+', '+', '+', '+', '+']
5,1	['+', '+', '+', '+', '+', '-']
4,1,1	['+', '+', '+', '+', '-', '+']
4,2	['+', '+', '+', '+', '-', '-']
3,1,2	['+', '+', '+', '-', '+', '+']
3,2,1	['+', '+', '+', '-', '-', '+']
3,3	['+', '+', '+', '-', '-', '-']
2,1,2,1	['+', '+', '-', '+', '+', '-']
2,1,1,2	['+', '+', '-', '+', '-', '-']
2,2,2	['+', '+', '-', '-', '+', '+']

Without loss of generality, let us fix $a = 1$. Then, we have $\overline{a} = -1$, and

$$Q^z = (1,-1,1,1,-1,-1,0,0,0,0,0,0,0,0,0,1,-1,-1,1,1,1)$$

Thus, the potential of the partition $1,1,2,2$ is equal to $\mathbb{E}(Q^z)$. The sidelobes' array of Q^z is:

$$S_{Q^z} = [1,0,1,0,1,0,-5,0,3,0,-1,0,0,0,0,0,0,0,0,-4,0],$$

therefore $\mathbb{E}(Q^z) = \sum_u {S_{Q^z}}_u^2 = 54$. The cardinality of the set $\mathscr{R}_{21}^{6}$ is $|\ \mathscr{R}_{21}^{6}\ | = 2^{6-3} + 2^{\lfloor \frac{6}{2} \rfloor - 2 + (6 \bmod 2)} = 2^3 + 2^1 = 10$. A list of unique partitions in $\mathscr{R}_{21}^{6}$ could be find in Table 5.4. For simplicity, we denote as $\mathscr{R}_n^{k|m}$ those partitions of size k over n, which posses exactly m elements. For example, referring Table 5.4, the partition (6) is in $\mathscr{R}_{21}^{6|1}$, partitions $(5,1)$, $(4,2)$ and $(3,3)$ are in $\mathscr{R}_{21}^{6|2}$, partitions $(4,1,1)$, $(3,1,2)$, $(3,2,1)$ and $(2,2,2)$ are in $\mathscr{R}_{21}^{6|3}$, while partitions $(2,1,2,1)$ and $(2,1,1,2)$ are in $\mathscr{R}_{21}^{6|4}$.

Given a partition $t_0, t_1, \cdots, t_g$ of size k, we will denote the set of skew-symmetric binary sequences defined by the partition as $\mathbb{B}_n^{t_0,t_1,\cdots,t_g}$. Please note that $\mathbb{B}_n^{t_0,t_1,\cdots,t_g} \subset \mathscr{R}_n^{k|g+1} \subset \mathscr{R}_n^{k}$. Finally, the potential of a given partition set S is denoted as $\mathscr{U}(S)$.

A few remarks regarding the sidelobes of a given potential ternary sequence should be made. In case we are interested in the potential of $\mathscr{R}_n^{k}$, the sidelobes of the ternary sequence could be divided into three distinct sections:

- **Head**: the first k sidelobes. They are shared among all sequences in this class, i.e. they are immutable.
- **Body**: the mid $n - 2k$ sidelobes. They are all equal to zero.

Table 5.5 Some partitions with optimal and normalized potentials

Class	$\mathscr{U}$ optimal	$\mathscr{U}$	$\mathscr{U}^\star$ optimal	$\mathscr{U}^\star$
$\mathscr{R}_n^{39\|4}$	18,11,6,4	3731	18,11,6,4	1082
$\mathscr{R}_n^{41\|6}$	17,9,6,4,3,2	2217	17,9,6,4,3,2	813
$\mathscr{R}_n^{47\|9}$	18,8,5,4,3,3,2,2,2	1859	11,9,5,5,5,3,3,3,3	830
$\mathscr{R}_n^{56\|4}$	27,14,9,6	12856	27,14,9,6	3472
$\mathscr{R}_n^{68\|7}$	25,11,10,7,5,5,5	9596	25,12,9,8,6,4,4	3040
$\mathscr{R}_n^{79\|9}$	26,12,10,7,6,6,6,4,2	11667	28,14,10,7,6,6,4,2,2	3702

- **Tail**: the last k sidelobes. Ignoring the sidelobes equal to zero, the remaining sidelobes are even numbers. They are not shared among the sequences in this class, i.e. they are mutable. However, partial information about their final value is gathered.

The actual calculation of the potential $\mathscr{U}(\mathscr{R}_n^k)$ gives an equal priority to the value of the elements in the head and the tail. However, we could tweak the actual energy calculation while minimizing the energy of the elements to prefer minimizing the elements in the head more, than minimizing the elements in the tail. All elements inside the tail are even numbers. This arises from the simple observation, that each summand of the form b_ib_j, which participates in a given sidelobe inside the tail, is accompanied by the symmetry summand b_jb_i. Having this in mind, if we prefer to minimize the energy of the summands, rather than minimizing their overall sums, we could normalize the tail by dividing its sidelobes values by 2. We will define this value as a normalized potential, denoted as $\mathscr{U}^\star(\mathscr{R}_n^k)$.

As a final remark, please note that despite $\mathscr{R}_n^k, \mathscr{R}_{n+1}^k, \mathscr{R}_{n+2}^k, \cdots$ is an infinite sequence of non-intersecting finite sets, their potentials and normalized potentials are equal. More formally, $\forall i \geq n \forall j \geq i : \mathscr{U}(\mathscr{R}_i^k) = \mathscr{U}(\mathscr{R}_j^k) \& \mathscr{U}^\star(\mathscr{R}_i^k) = \mathscr{U}^\star(\mathscr{R}_j^k)$.

During our research, by using an exhaustive search, we have calculated all the potentials, as well as normalized potentials, of set partitions of the form $\mathscr{R}_n^{k|g}$, for $38 < k < 115$ and some values of $g \in [4, 12]$. For speeding up the exhaustive routine, the following restriction of the partitions were further applied: $\forall i : t_i \geq t_{i+1}$. As an illustration, various partitions having an optimal potential and normalized potential are given in Table 5.5.

5.1.3 Algorithm for Finding Binary Sequences with Arbitrary Length and High Merit Factor

By achieving both linear time and memory complexities, we can utilize all the threads of a given central processing unit. Furthermore, the memory requirements of a given algorithm are significantly reduced.

In Algorithm 12 a pseudo-code of the proposed routine is presented. The following additional notations and remarks should be considered:

- n - an odd integer number
- $t_0, t_1, \cdots, t_g$ - the partition search space to search through.
- $\mathbb{T}_i$ - an inner threshold value. When the inner counter w_i reaches $\mathbb{T}_i$, the set is flushed and the whole routine restarts. The threshold value $\mathbb{T}_i$ constrains the size of the set $\mathbb{H}$.
- $\mathbb{T}_o$ - an outer threshold value. When the outer counter w_o reaches $\mathbb{T}_o$, the program is terminated.
- $\mathbb{T}_a$ - an activator threshold value. For example, the probability of finding a pseudo-skew-symmetric sequence with length $n-1$ or $n+1$ and merit factor X, from a skew-symmetric sequence with length n and merit factor $X-1$, is negligible for higher values of X. Thus, we could save time and effort to repeatedly probe the adjacent pseudo-skew-symmetric sequences.
- $\perp_{n-1}, \perp_n, \perp_{n+1}$ - the best candidates found, in terms of merit factor value, for respectively pseudo and not pseudo-skew-symmetric sequences of lengths $n-1, n$ and $n+1$.
- $\mathbb{H}$ - a set of hashes of the visited candidates. We make sure to avoid already visited nodes.
- $\mathbb{H}$.add(hash(B)) - adding the hash of the binary sequence B to the set $\mathbb{H}$.
- pickBetterNeighborIndex - a function, which returns the index of a better-unexplored neighbor of B, i.e. the binary sequence with a distance of exactly 1 flip away from B, s.t. its hash does not belong to the set $\mathbb{H}$. An optimized derivative-based pseudo-code of this helper function is discussed in our previous work [45].

Algorithm 12 was implemented (C++) on a general-purpose computer equipped with a central processing unit with 8 cores and 16 threads. Despite using just a single low-budget personal computer, we were able to improve all the results, for all skew-symmetric

Algorithm 12 An algorithm for searching skew-symmetric and pseudo-skew-symmetric binary sequences with arbitrary lengths and high merit factors.

procedure MF($n, t_0, t_1, \cdots, t_g, \mathbb{T}_i, \mathbb{T}_o, \mathbb{T}_a$)
  $\perp_{n-1}, \perp_n, \perp_{n+1}, w_o \leftarrow 0, 0, 0, 0$
  while True **do**
    $\mathbb{H}, w_i, \leftarrow \{\varnothing\}, 0$
    $B \leftarrow$ random , s.t. $B \in \mathbb{B}_n^{t_0, t_1, \cdots, t_g} \subset \mathscr{R}_n^k$, for $k = \sum_{i=0}^{g} t_i$.
    $\mathbb{H}$.add(hash(B))
    $V \leftarrow \mathbb{E}(B)$
    while True **do**
      bestN $\leftarrow$ pickBetterNeighborIndex(B)
      if bestN $== -1$ **then**
        break
      end if
      Flip(bestN, B)
      $V \leftarrow \mathbb{E}(B)$
      $w_i \mathrel{+}= 1$
      $\mathbb{H}$.add(hash(B))
      if $\frac{n^2}{2V} > \perp_n$ **then**
        $\perp_n \leftarrow \frac{n^2}{2V}$
      end if
      if $\frac{n^2}{2V} \geq \mathbb{T}_a$ **then**
        if $\frac{(n+1)^2}{2(V+n+2b_n\delta)} > \perp_{n+1}$ **then**
          $\perp_{n+1} \leftarrow \frac{(n+1)^2}{2(V+n+2b_n\delta)}$
        end if
        if $\frac{(n-1)^2}{2(V+n-3+2b_{n-1}\delta)} > \perp_{n-1}$ **then**
          $\perp_{n-1} \leftarrow \frac{(n-1)^2}{2(V+n-3+2b_{n-1}\delta)}$
        end if
      end if
      if $w_i > \mathbb{T}_i$ **then**
        $w_o \mathrel{+}= 1$
        break
      end if
    end while
    if $w_o > \mathbb{T}_o$ **then**
      break
    end if
  end while
end procedure

lengths in the range 225-451, announced in literature and reached by using a supercomputer grid. Furthermore, by using classes of pseudo-skew-symmetric sequences, we were able to simultaneously reach binary sequences of even lengths between 225 and 512, and beyond, with merit factors greater than 7. We demonstrate the efficiency of our approach by publishing a complete list of binary sequences, for both even and odd lengths up to 2^8, and merit factors greater than 8. The list is further accompanied by a complete list of binary sequences, for both even and odd lengths up to 2^9, and merit factors greater than 7 (see Tables B.5 - B.15).

We further demonstrate the power and efficiency of the proposed algorithm by launching it on binary sequences of lengths 573 and 1009. As mentioned earlier, the choice of those two specific lengths is motivated by the approximation numbers given in [24], Figure 7, presented during a discussion of how much time the state-of-the-art stochastic solver lssOrel_8 will need to reach binary sequences with the aforementioned lengths and merit factors close to 6.34. It was estimated that finding solutions with a merit factor of 6.34 for a binary sequence with length 573 requires around 32 years, while for binary sequences with length 1009, the average runtime prediction to reach the merit factor of 6.34 is 46774481153 years. By using the proposed algorithm, we were able to reach such candidates within several hours (see Table B.16). By further applying some operators on the skew-symmetric binary sequence of length 1009 found, several sequences of lengths 1006, 1007, 1008, and 1010 with MF greater than 6.34 were also revealed. The same argument is true for the other sequence of length 573, but since the results are too many we omit the data.

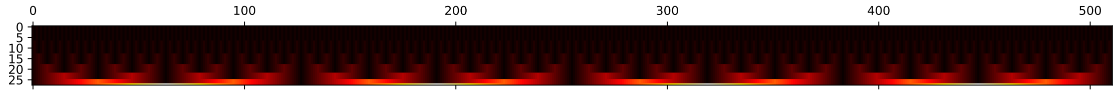
For convenience, we denote the operators acting on binary sequences as shown in Table 5.6. Please note that operator η_0 activated on a given skew-symmetric binary sequence $a||L||b$ will yield another skew-symmetric binary sequence L, while all other operators activated on the same skew-symmetric binary sequence will yield a pseudo-skew-symmetric sequence. Throughout the tables with reported records, the classes denoted with Ω represent the best-known result to be found in the literature for the current length (all in Table B.5, for the lengths between 172 and 226). It should be emphasized, that all records achieved by starting from a sequence of class Ω, are directly calculated without the usage of any additional stochastic routine. All other records throughout the tables (classes $\mathbb{B}$) are achieved by using a heuristic search. All sequences are presented in hexadecimal format with zeroes omitted. It should be noted, that as soon as the algorithm finds a record sequence of a given length, it automatically continues to the next search space. In some cases, we required a little bit more demanding goal, i.e. MF greater than 8 (for sequences with lengths less than about 256), or MF greater than 7 (for sequences with lengths less than about 512). Some records were found for several minutes, while others required a little bit more effort of several hours.

Table 5.6 A list of used operators acting on binary sequences

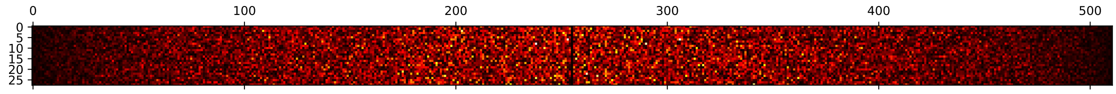
Operator	Action
η_0	$a\|\|L\|\|b \circ \eta_0 = L$
η_1	$L \circ \eta_1 = L\|\|1$
η_2	$L \circ \eta_2 = L\|\| - 1$
η_3	$a\|\|L \circ \eta_3 = L$
η_4	$L\|\|b \circ \eta_4 = L$
η_5	$L \circ \eta_5 = 1\|\|L$
η_6	$L \circ \eta_6 = -1\|\|L$

5.2 Using Aperiodic Autocorrelation functions for an S-box reverse engineering

We can treat all $\binom{n}{2}$ columns of two-term linear combinations of coordinates of an S-box $S(n,n)$ as binary sequences and analyze their sidelobe levels. Such a strategy makes sense since sidelobe levels can reveal hidden inner relationships between the coordinates of S. In Figure 5.3 the obtained results are given. The absolute values of side lobes values are interchanged with a gradient palette starting from darker (lower values) to lighter (higher values). In Figure 5.3a the side lobes plot of the trivial $(8,8)$ S-box, i.e. the identity permutation, is plotted, while Figure 5.3b is an example of a random $(8,8)$ S-box side lobes plot. The anomalies in S-boxes of BelT, CSS, Safer, and SKINNY are visible.

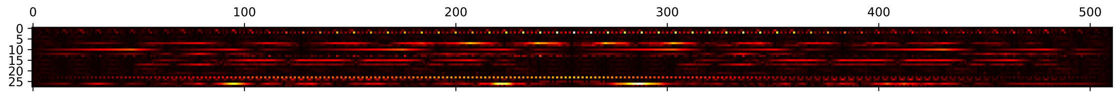


(a) Identity permutation

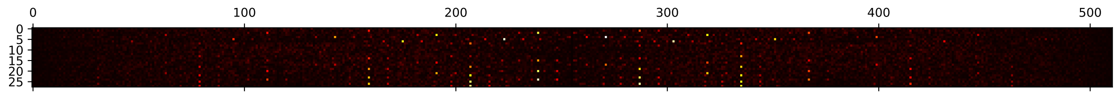


(b) Random permutation

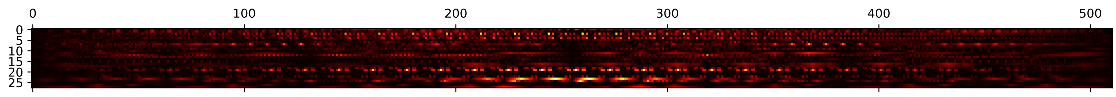
(c) BelT



(d) CSS



(e) Safer



(f) SKINNY

Fig. 5.3 Anomalies detected in various S-boxes' side lobes spectra

References

[1] Oeis a000041. https://oeis.org/A000041. Accessed: 2022-05-30.

[2] Oeis a005418. https://oeis.org/A005418. Accessed: 2022-05-30.

[3] Adomnicai, A., Berger, T. P., Clavier, C., Francq, J., Huynh, P., Lallemand, V., Le Gouguec, K., Minier, M., Reynaud, L., and Thomas, G. (2019). Lilliput-ae: a new lightweight tweakable block cipher for authenticated encryption with associated data. *Submitted to NIST Lightweight Project*.

[4] Ahmad, M., Bhatia, D., and Hassan, Y. (2015). A novel ant colony optimization based scheme for substitution box design. *Procedia Computer Science*, 57:572–580.

[5] Andreeva, E., Lallemand, V., Purnal, A., Reyhanitabar, R., Roy, A., and Vizár, D. (2019). Forkae v. *Submission to NIST lightweight cryptography project*.

[6] Andrews, G. E. (1998). *The theory of partitions*. Number 2. Cambridge university press.

[7] Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., and Tokita, T. (2000). Camellia: A 128-bit block cipher suitable for multiple platforms—design andanalysis. In *International Workshop on Selected Areas in Cryptography*, pages 39–56. Springer.

[8] Baden, J. and Cohen, M. (1990). Optimal peak sidelobe filters for biphase pulse compression. In *IEEE International Conference on Radar*, pages 249–252. IEEE.

[9] Barker, R. H. and Jackson, W. (1953). Group synchronization of binary digital systems in Communication Theory. *Academic Press, New York*, pages 273–287.

[10] Barreto, P. and Rijmen, V. (2000). The khazad legacy-level block cipher. *Primitive submitted to NESSIE*, 97:106.

[11] Barreto, P., Rijmen, V., et al. (2000a). The Whirlpool hashing function. In *First open NESSIE Workshop, Leuven, Belgium*, volume 13, page 14. Citeseer.

[12] Barreto, P., Rijmen, V., et al. (2000b). The whirlpool hashing function. In *First open NESSIE Workshop, Leuven, Belgium*, volume 13, page 14. Citeseer.

[13] Becker, M. and Desoky, A. (2004). A study of the dvd content scrambling system (css) algorithm. In *Proceedings of the Fourth IEEE International Symposium on Signal Processing and Information Technology, 2004.*, pages 353–356. IEEE.

[14] Bernasconi, J. (1987). Low autocorrelation binary sequences: statistical mechanics and configuration space analysis. *Journal de Physique*, 48(4):559–567.

[15] Berry, D. A. and Fristedt, B. (1985). Bandit problems: sequential allocation of experiments (Monographs on statistics and applied probability). *London: Chapman and Hall*, 5:71–87.

[16] Bhattacharya, D., Bansal, N., Banerjee, A., and RoyChowdhury, D. (2007). A near optimal S-box design. In *International Conference on Information Systems Security*, pages 77–90. Springer.

[17] Biham, E. (1994). On Matsui's linear cryptanalysis. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 341–355. Springer.

[18] Biham, E. and Shamir, A. (1991). Differential cryptanalysis of DES-like cryptosystems. *Journal of CRYPTOLOGY*, 4(1):3–72.

[19] Bikov, D., Bouyukliev, I., and Bouyuklieva, S. (2019). Bijective S-boxes of different sizes obtained from quasi-cyclic codes. *Journal of Algebra Combinatorics Discrete Structures and Applications*, 6(3):123–134.

[20] Biryukov, A. and Perrin, L. (2015). On reverse-engineering s-boxes with hidden design criteria or structure. In *Annual Cryptology Conference*, pages 116–140. Springer.

[21] Biryukov, A., Perrin, L., and Udovenko, A. (2016). Reverse-engineering the s-box of streebog, kuznyechik and stribobr1. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 372–402. Springer.

[22] Borwein, P., Choi, K.-K., and Jedwab, J. (2004). Binary sequences with merit factor greater than 6.34. *IEEE transactions on information theory*, 50(12):3234–3249.

[23] Boškovic, B., Brglez, F., and Brest, J. (2016). A github archive for solvers and solutions of the labs problem. *For updates, see https://github. com/borkob/git_labs (January 2016)*.

[24] Bošković, B., Brglez, F., and Brest, J. (2017). Low-autocorrelation binary sequences: On improved merit factors and runtime predictions to achieve them. *Applied Soft Computing*, 56:262–285.

[25] Bouyukliev, I., Bikov, D., and Bouyuklieva, S. (2017). S-boxes from binary quasi-cyclic codes. *Electronic Notes in Discrete Mathematics*, 57:67–72.

[26] Brest, J. and Bošković, B. (2018). A heuristic algorithm for a low autocorrelation binary sequence problem with odd length and high merit factor. *IEEE Access*, 6:4127–4134.

[27] Brest, J. and Bošković, B. (2020). In searching of long skew-symmetric binary sequences with high merit factors. *arXiv preprint arXiv:2011.00068*.

[28] Byrnes, J. and Newman, D. J. (1990). The l4 norm of a polynomial with coefficients± 1. *Amer. Math. Monthly*, 97:42–45.

[29] Canteaut, A., Duval, S., and Leurent, G. (2015a). Construction of lightweight S-boxes using Feistel and MISTY structures. In *International Conference on Selected Areas in Cryptography*, pages 373–393. Springer.

[30] Canteaut, A., Duval, S., and Leurent, G. (2015b). Construction of lightweight s-boxes using feistel and misty structures. In *International Conference on Selected Areas in Cryptography*, pages 373–393. Springer.

[31] Chen, G. (2008). A novel heuristic method for obtaining S-boxes. *Chaos, Solitons & Fractals*, 36(4):1028–1036.

[32] Clark, J. A., Jacob, J. L., and Stepney, S. (2005). The design of S-boxes by simulated annealing. *New Generation Computing*, 23(3):219–231.

[33] Coppersmith, D. (1994). The data encryption standard (des) and its strength against attacks. *IBM journal of research and development*, 38(3):243–250.

[34] Courtois, N. T. and Pieprzyk, J. (2002). Cryptanalysis of block ciphers with overdefined systems of equations. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 267–287. Springer.

[35] Coxson, G. and Russo, J. (2005). Efficient exhaustive search for optimal-peak-sidelobe binary codes. *IEEE Transactions on Aerospace and Electronic Systems*, 41(1):302–308.

[36] Coxson, G. E., Hill, C. R., and Russo, J. C. (2014). Adiabatic quantum computing for finding low-peak-sidelobe codes. In *2014 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6. IEEE.

[37] Coxson, G. E., Russo, J. C., and Luther, A. (2020). Long low-psl binary codes by multi-thread evolutionary search. In *2020 IEEE International Radar Conference (RADAR)*, pages 256–261. IEEE.

[38] Cui, L. and Cao, Y. (2007). A new S-box structure named Affine-Power-Affine. *International Journal of Innovative Computing, Information and Control*, 3(3):751–759.

[39] Daemen, J. and Rijmen, V. (2013a). *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media.

[40] Daemen, J. and Rijmen, V. (2013b). *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media.

[41] De Groot, C., Würtz, D., and Hoffmann, K. H. (1992). Low autocorrelation binary sequences: Exact enumeration and optimization by evolutionary strategies. *Optimization*, 23(4):369–384.

[42] de la Cruz Jiménez, R. A. (2017). Generation of 8-Bit S-Boxes Having Almost Optimal Cryptographic Properties Using Smaller 4-Bit S-Boxes and Finite Field Multiplication. In *International Conference on Cryptology and Information Security in Latin America*, pages 191–206. Springer.

[43] Developers, T. S. (2016). Sagemath.

[44] Dimitrov, M. (2020a). On the aperiodic autocorrelations of rotated binary sequences. *IEEE Communications Letters*, 25(5):1427–1430.

[45] Dimitrov, M. (2021a). On the skew-symmetric binary sequences and the merit factor problem. *arXiv preprint arXiv:2106.03377*.

[46] Dimitrov, M. (2022). New classes of binary sequences with high merit factor. *arXiv preprint arXiv:2206.12070*.

[47] Dimitrov, M., Baicheva, T., and Nikolov, N. (2021). Hybrid constructions of binary sequences with low autocorrelation sideobes. *IEEE Access*, 9:112400–112410.

[48] Dimitrov, M., Baitcheva, T., and Nikolov, N. (2020a). Efficient generation of low autocorrelation binary sequences. *IEEE Signal Processing Letters*, 27:341–345.

[49] Dimitrov, M., Baitcheva, T., and Nikolov, N. (2020b). On the generation of long binary sequences with record-breaking PSL values. *IEEE Signal Processing Letters*, 27:1904–1908.

[50] Dimitrov, M. M. (2020b). On the design of chaos-based s-boxes. *IEEE Access*, 8:117173–117181.

[51] Dimitrov, M. M. (2021b). A framework for fine-grained nonlinearity optimization of boolean and vectorial boolean functions. *IEEE Access*, 9:124910–124920.

[52] Dmitriev, D. and Jedwab, J. (2007). Bounds on the growth rate of the peak sidelobe level of binary sequences. *Advances in Mathematics of Communications*, 1(4):461.

[53] Dolmatov, V. (2016). Gost r 34.12-2015: Block cipher “kuznyechik”. *Transformation*, 50:10.

[54] Du, K. L., Wu, W. H., and Mow, W. H. (2013). Determination of long binary sequences having low autocorrelation functions. US Patent 8,493,245.

[55] FIPS, P. (1999). 46-3. data encryption standard (des). *National Institute of Standards and Technology*, 25(10):1–22.

[56] Flynn, M. J. (1972). Some computer organizations and their effectiveness. *IEEE transactions on computers*, 100(9):948–960.

[57] Gallardo, J. E., Cotta, C., and Fernández, A. J. (2009). Finding low autocorrelation binary sequences with memetic algorithms. *Applied Soft Computing*, 9(4):1252–1262.

[58] Gérard, B., Grosso, V., Naya-Plasencia, M., and Standaert, F.-X. (2013). Block ciphers that are easier to mask: How far can we go? In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 383–399. Springer.

[59] Gilbert, H. and Peyrin, T. (2010). Super-Sbox cryptanalysis: improved attacks for AES-like permutations. In *International Workshop on Fast Software Encryption*, pages 365–383. Springer.

[60] Golay, M. (1972). A class of finite binary sequences with alternate auto-correlation values equal to zero (corresp.). *IEEE Transactions on Information Theory*, 18(3):449–450.

[61] Golay, M. (1975). Hybrid low autocorrelation sequences (corresp.). *IEEE Transactions on Information Theory*, 21(4):460–462.

[62] Golay, M. (1977). Sieves for low autocorrelation binary sequences. *IEEE Transactions on information theory*, 23(1):43–51.

[63] Golay, M. (1982). The merit factor of long low autocorrelation binary sequences (corresp.). *IEEE Transactions on Information Theory*, 28(3):543–549.

[64] Golay, M. (1983). The merit factor of legendre sequences (corresp.). *IEEE Transactions on Information Theory*, 29(6):934–936.

[65] Golay, M. J. and Harris, D. B. (1990). A new search for skewsymmetric binary sequences with optimal merit factors. *IEEE Transactions on Information Theory*, 36(5):1163–1166.

[66] Gold, R. (1967). Optimal binary sequences for spread spectrum multiplexing (Corresp.). *IEEE Transactions on Information Theory*, 13(4):619–621.

[67] Golomb, S. W. et al. (1967). *Shift register sequences*. Aegean Park Press.

[68] GOST, R. (2015). R 34.12-2015. *Information Technology. Cryptographic Protection of Information. Block Ciphers]. Moscow, Standartinform.*

[69] Grosso, V., Leurent, G., Standaert, F.-X., and Varıcı, K. (2014a). LS-designs: Bitslice encryption for efficient masked software implementations. In *International Workshop on Fast Software Encryption*, pages 18–37. Springer.

[70] Grosso, V., Leurent, G., Standaert, F.-X., Varici, K., Durvaux, F., Gaspar, L., and Kerckhof, S. (2014b). Scream & iscream side-channel resistant authenticated encryption with masking. *Submission to CAESAR*.

[71] Gurobi Optimization, I. (2018). Gurobi optimizer reference manual. *URL http://www. gurobi. com*.

[72] Halim, S., Yap, R. H., and Halim, F. (2008). Engineering stochastic local search for the low autocorrelation binary sequence problem. In *International Conference on Principles and Practice of Constraint Programming*, pages 640–645. Springer.

[73] He, H., Stoica, P., and Li, J. (2009). Designing unimodular sequence sets with good correlations—including an application to mimo radar. *IEEE Transactions on Signal Processing*, 57(11):4391–4405.

[74] Heys, H. M. (2002). A tutorial on linear and differential cryptanalysis. *Cryptologia*, 26(3):189–221.

[75] Hoholdt, T. and Jensen, H. E. (1988). Determination of the merit factor of legendre sequences. *IEEE Transactions on Information Theory*, 34(1):161–164.

[76] Isa, H., Jamil, N., and Z'aba, M. R. (2013). S-box construction from non-permutation power functions. In *Proceedings of the 6th International Conference on Security of Information and Networks*, pages 46–53. ACM.

[77] Isa, H., Jamil, N., and Z'aba, M. R. (2016). Construction of cryptographically strong S-Boxes inspired by bee waggle dance. *New generation computing*, 34(3):221–238.

[78] Ivanov, G., Nikolov, N., and Nikova, S. (2015). Cryptographically strong S-boxes generated by modified immune algorithm. In *International Conference on Cryptography and Information Security in the Balkans*, pages 31–42. Springer.

[79] Jakobsen, T. and Knudsen, L. R. (1997). The interpolation attack on block ciphers. In *International Workshop on Fast Software Encryption*, pages 28–40. Springer.

[80] Jedwab, J. (2004). A survey of the merit factor problem for binary sequences. In *International Conference on Sequences and Their Applications*, pages 30–55. Springer.

[81] Jedwab, J. and Yoshida, K. (2006). The peak sidelobe level of families of binary sequences. *IEEE transactions on information theory*, 52(5):2247–2254.

[82] Junod, P. and Vaudenay, S. (2004). Fox: a new family of block ciphers. In *International Workshop on Selected Areas in Cryptography*, pages 114–129. Springer.

[83] Karpman, P. and Grégoire, B. (2016). The littlun s-box and the fly block cipher. In *Lightweight Cryptography Workshop*, pages 17–18.

[84] Kasami, T. (1966). Weight distribution formula for some class of cyclic codes. *Coordinated Science Laboratory Report no. R-285*.

[85] Kazymyrov, O., Kazymyrova, V., and Oliynykov, R. (2013). A Method For Generation Of High-Nonlinear S-Boxes Based On Gradient Descent. *IACR Cryptology ePrint Archive*, 2013:578.

[86] Kerahroodi, M. A., Aubry, A., De Maio, A., Naghsh, M. M., and Modarres-Hashemi, M. (2017). A coordinate-descent framework to design low psl/isl sequences. *IEEE Transactions on Signal Processing*, 65(22):5942–5956.

[87] Leukhin, A., Parsaev, N., Bezrodnyi, V., and Kokovihina, N. (2017). The exhaustive search for optimum minimum peak sidelobe binary sequences. *Bulletin of the Russian Academy of Sciences: Physics*, 81(5):575–578.

[88] Leukhin, A. and Potehin, E. (2012). Binary sequences with minimum peak sidelobe level up to length 68. *arXiv preprint arXiv:1212.4930*.

[89] Leukhin, A. and Potekhin, E. (2015). A Bernasconi model for constructing ground-state spin systems and optimal binary sequences. In *Journal of Physics: Conference Series*, volume 613, page 012006. IOP Publishing.

[90] Leukhin, A. N. and Potekhin, E. N. (2013). Optimal peak sidelobe level sequences up to length 74. In *2013 European Radar Conference*, pages 495–498. IEEE.

[91] Leukhin, Anatolii N and Potekhin, Egor N (2014). Exhaustive search for optimal minimum peak sidelobe binary sequences up to length 80. In *International Conference on Sequences and Their Applications*, pages 157–169. Springer.

[92] Levanon, N. and Mozeson, E. (2004). *Radar signals*. John Wiley & Sons.

[93] Lim, C. H. (1998). Crypton: A new 128-bit block cipher. *NIsT AEs Proposal*.

[94] Lim, C. H. (1999). A revised version of crypton: Crypton v1. 0. In *International Workshop on Fast Software Encryption*, pages 31–45. Springer.

[95] Lin, R., Soltanalian, M., Tang, B., and Li, J. (2019). Efficient design of binary sequences with low autocorrelation sidelobes. *IEEE Transactions on Signal Processing*, 67(24):6397–6410.

[96] Lindner, J. (1975). Binary sequences up to length 40 with best possible autocorrelation function. *Electronics letters*, 11(21):507–507.

[97] Littlewood, J. (1966). On Polynomials $\sum^n \pm z^m$, $\sum^n e^{\alpha_m i} z^m$, $z = e^{\theta i}$. *Journal of the London Mathematical Society*, 1(1):367–376.

[98] Littlewood, J. E. (1968). *Some problems in real and complex analysis*. DC Heath.

[99] Losanitsch, S. (1897). Die isomerie-arten bei den homologen der paraffin-reihe. *Berichte der deutschen chemischen Gesellschaft*, 30(2):1917–1926.

[100] Madras, N. and Slade, G. (2013). *The self-avoiding walk*. Springer Science & Business Media.

[101] Mamadolimov, A., Isa, H., and Mohamad, M. S. (2013). Practical bijective S-box design. *arXiv preprint arXiv:1301.4723*.

[102] Massey, J. L. (1993). Safer k-64: A byte-oriented block-ciphering algorithm. In *International Workshop on Fast Software Encryption*, pages 1–17. Springer.

[103] Matsui, M. (1993). Linear cryptanalysis method for DES cipher. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 386–397. Springer.

[104] Meier, W. and Staffelbach, O. (1989). Nonlinearity criteria for cryptographic functions. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 549–562. Springer.

[105] Mertens, S. (1996). Exhaustive search for low-autocorrelation binary sequences. *Journal of Physics A: Mathematical and General*, 29(18):L473.

[106] Militzer, B., Zamparelli, M., and Beule, D. (1998). Evolutionary search for low autocorrelated binary sequences. *IEEE Transactions on Evolutionary Computation*, 2(1):34–39.

[107] Millan, W. (1998). How to improve the nonlinearity of bijective S-boxes. In *Australasian Conference on Information Security and Privacy*, pages 181–192. Springer.

[108] Millan, W., Burnett, L., Carter, G., Clark, A., and Dawson, E. (1999). Evolutionary heuristics for finding cryptographically strong S-boxes. In *International Conference on Information and Communications Security*, pages 263–274. Springer.

[109] Mow, W. H., Du, K.-L., and Wu, W. H. (2015). New evolutionary search for long low autocorrelation binary sequences. *IEEE Transactions on aerospace and electronic systems*, 51(1):290–303.

[110] Mroczkowski, P. (2009). Generating Pseudorandom S-Boxes-a Method of Improving the Security of Cryptosystems Based on Block Ciphers. *Journal of Telecommunications and Information Technology*, pages 74–79.

[111] Nasrabadi, M. A. and Bastani, M. H. (2010). A survey on the design of binary pulse compression codes with low autocorrelation. In *Trends in Telecommunications Technologies*. IntechOpen.

[112] Nunn, C. J. and Coxson, G. E. (2008). Best-known autocorrelation peak sidelobe levels for binary codes of length 71 to 105. *IEEE transactions on Aerospace and Electronic Systems*, 44(1):392–395.

[113] Nyberg, K. (1991a). Perfect nonlinear S-boxes. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 378–386. Springer.

[114] Nyberg, K. (1991b). Perfect nonlinear S-boxes. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 378–386. Springer.

[115] Oliphant, T. E. (2006). *A guide to NumPy*, volume 1. Trelgol Publishing USA.

[116] Oliynykov, R., Gorbenko, I., Kazymyrov, O., Ruzhentsev, V., Kuznetsov, O., Gorbenko, Y., Dyrda, O., Dolgov, V., Pushkaryov, A., Mordvinov, R., et al. (2015). A new encryption standard of ukraine: The kalyna block cipher. *IACR Cryptology ePrint Archive*, 2015:650.

[117] Orhanou, G., El Hajji, S., and Bentaleb, Y. (2010). Snow 3g stream cipher operation and complexity study. *Contemporary Engineering Sciences-Hikari Ltd*, 3(3):97–111.

[118] Packebusch, T. and Mertens, S. (2016). Low autocorrelation binary sequences. *Journal of Physics A: Mathematical and Theoretical*, 49(16):165001.

[119] Perrin, L. P. (2017). *Cryptanalysis, reverse-engineering and design of symmetric cryptographic algorithms*. PhD thesis, University of Luxembourg, Luxembourg, Luxembourg.

[120] Perrin, L. P. and Udovenko, A. (2017). Exponential s-boxes: a link between the s-boxes of belt and kuznyechik/streebog. *IACR Transactions on Symmetric Cryptology*, 2016(2):99–124.

[121] Picek, S., Cupic, M., and Rotim, L. (2016a). A new cost function for evolution of s-boxes. *Evolutionary computation*, 24(4):695–718.

[122] Picek, S., Santana, R., and Jakobovic, D. (2016b). Maximal nonlinearity in balanced boolean functions with even number of inputs, revisited. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 3222–3229. IEEE.

[123] Piret, G., Roche, T., and Carlet, C. (2012). Picaro–a block cipher allowing efficient higher-order side-channel resistance. In *International Conference on Applied Cryptography and Network Security*, pages 311–328. Springer.

[124] Pott, A. (2006). *Finite geometry and character theory*. Springer.

[125] Prestwich, S. D. (2013). Improved branch-and-bound for low autocorrelation binary sequences. *arXiv preprint arXiv:1305.6187*.

[126] Reeds III, J. A. (1992). Cryptosystem for cellular telephony. US Patent 5,159,634.

[127] Rijmen, V. and Barreto, P. (2000). The anubis block cipher. *Submission to NESSIE.*

[128] Rijmen, V. and Preneel, B. (1997). A family of trapdoor ciphers. In *International Workshop on Fast Software Encryption*, pages 139–148. Springer.

[129] Rudin, W. (1959). Some theorems on fourier coefficients. *Proceedings of the American Mathematical Society*, 10(6):855–859.

[130] Rushanan, J. J. (2006). Weil sequences: A family of binary sequences with good correlation properties. In *2006 IEEE International Symposium on Information Theory*, pages 1648–1652. IEEE.

[131] SageMath. Preliminary State Standard of Republic of Belarus (STB P 34.101.31–2007). http://apmi.bsu.by/assets/files/std/belt-spec27.pdf.

[132] SageMath. SageMath Sbox library. https://github.com/sagemath/sage/blob/master/src/sage/crypto/sboxes.py.

[133] Sarkar, P. and Maitra, S. (2000). Nonlinearity bounds and constructions of resilient boolean functions. In *Annual International Cryptology Conference*, pages 515–532. Springer.

[134] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., and Ferguson, N. (1998). Twofish: A 128-bit block cipher. aes submission, 15 june 1998.

[135] Schotten, H. D. and Lüke, H. D. (2005). On the search for low correlated binary sequences. *AEU-International Journal of Electronics and Communications*, 59(2):67–78.

[136] Shapiro, H. S. (1952). *Extremal problems for polynomials and power series*. PhD thesis, Massachusetts Institute of Technology.

[137] Shirai, T., Shibutani, K., Akishita, T., Moriai, S., and Iwata, T. (2007). The 128-bit blockcipher clefia. In *International workshop on fast software encryption*, pages 181–195. Springer.

[138] Skipjack, N. (1998). KEA algorithm specifications. *Online document: http://csrc. nist. org/encryption/skipjack/skipjack. pdf.*

[139] Skolnik, M. I. (1970). Radar handbook.

[140] Soltanalian, M. and Stoica, P. (2012). Computational design of sequences with good correlation properties. *IEEE Transactions on Signal processing*, 60(5):2180–2193.

[141] Song, J., Babu, P., and Palomar, D. P. (2015). Sequence design to minimize the weighted integrated and peak sidelobe levels. *IEEE Transactions on Signal Processing*, 64(8):2051–2064.

[142] Souravlias, D., Parsopoulos, K. E., and Meletiou, G. C. (2017). Designing bijective S-boxes using Algorithm Portfolios with limited time budgets. *Applied Soft Computing*, 59:475–486.

[143] Standaert, F.-X., Piret, G., Rouvroy, G., Quisquater, J.-J., and Legat, J.-D. (2004). Iceberg: An involutional cipher efficient for block encryption in reconfigurable hardware. In *International Workshop on Fast Software Encryption*, pages 279–298. Springer.

[144] Stern, J. and Vaudenay, S. (1998). Cs-cipher. In *International Workshop on Fast Software Encryption*, pages 189–204. Springer.

[145] Tesař, P. (2010). A new method for generating high non-linearity s-boxes. *Radioengineering*, 19(1):23–26.

[146] Turyn, R. et al. (1968). Sequences with small correlation. In *Error correcting codes*, pages 195–228. Wiley New York.

[147] Wagner, D. (1999). The boomerang attack. In *International Workshop on Fast Software Encryption*, pages 156–170. Springer.

[148] Watanabe, D., Furuya, S., Yoshida, H., Takaragi, K., and Preneel, B. (2002). A new keystream generator mugi. In *International Workshop on Fast Software Encryption*, pages 179–194. Springer.

[149] Wiener, N. (1964). *Extrapolation, interpolation, and smoothing of stationary time series*. The MIT press.

[150] Wikipedia source (1999). Wikipedia. https://en.wikipedia.org/wiki/Iraqi_block_cipher.

[151] Wu, H., Bao, F., Deng, R. H., and Ye, Q.-Z. (1998). Cryptanalysis of rijmen-preneel trapdoor ciphers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 126–132. Springer.

[152] Xiu-tao, F. (2011). Zuc algorithm: 3gpp lte international encryption standard [j]. *Information Security and Communications Privacy*, 12.

Appendix A

S-box Characteristics and Collisions

A.1 Detailed characteristics of popular S-boxes

Table A.1 S-boxes overview.

S	S_{NL}	S_δ	S_{AC}	S_{DEG}	Spectra
Anubis	94	8	96	7	6896, 12520, 11331, 9978, 7750, 5970, 4335, 2760, 1861, 1046, 572, 282, 138, 78, 10, 2, 2, 4
BelT	102	8	88	6	6277, 11745, 11387, 9943, 8250, 6301, 4793, 3344, 1836, 979, 428, 190, 52, 10
CLEFIA S_0	100	10	96	6	14160, 0, 22280, 0, 15596, 0, 8387, 0, 3535, 0, 1185, 0, 340, 0, 52
CMEA	96	12	104	6	7338, 12050, 11742, 9575, 7930, 5931, 4212, 2773, 1798, 1046, 576, 286, 162, 72, 30, 11, 3
Crypton_1_0 S_0 Crypton_1_0 S_1 Crypton_1_0 S_2 Crypton_1_0 S_3	96	10	96	6	13926, 0, 22058, 0, 15948, 0, 8460, 0, 3731, 0, 1094, 0, 276, 0, 36, 0, 6

Continue on the next page

S-boxes overview (continued).

S	S_{NL}	S_{δ}	S_{AC}	S_{DEG}	Spectra
Crypton_0_5	88	16	128	4	20891, 0, 11596, 0, 22018, 0, 4812, 0, 5236, 0, 468, 0, 370, 0, 20, 0, 112, 0, 0, 0, 12
CSA	94	12	104	7	7073, 12586, 11246, 9747, 7865, 6041, 4231, 2777, 1733, 1117, 581, 308, 139, 54, 22, 7, 5, 3
CS_cipher	96	16	128	3	33183, 0, 0, 0, 23264, 0, 0, 0, 8448, 0, 0, 0, 288, 0, 0, 0, 352
Enocoro	96	10	128	6	14248, 0, 21982, 0, 15824, 0, 8369, 0, 3572, 0, 1179, 0, 300, 0, 54, 0, 7
E2	100	10	104	6	6730, 12172, 11248, 9841, 8000, 6217, 4644, 2983, 1765, 991, 507, 272, 120, 36, 9,
Fantomas	96	16	128	2	26877, 0, 11568, 0, 15584, 0, 4220, 0, 5816, 0, 572, 0, 544, 0, 24, 0, 330
Fox	96	16	128	4	19196, 0, 18171, 0, 15888, 0, 6405, 0, 4280, 0, 983, 0, 352, 0, 41, 0, 219
Iceberg	96	8	96	7	6929, 12610, 11291, 9774, 7881 , 6060, 4166, 2892, 1887, 940, 566, 288, 137, 62, 33, 14, 5
iScream	96	16	128	4	23103, 0, 14728, 0, 15888, 0, 4824, 0, 5536, 0, 904, 0, 240, 0, 24, 0, 288
Kalyna π_0	104	8	72	7	6317, 11616, 10829, 9829, 8542, 6834, 4912, 3317, 1880, 897, 371, 147, 44

Continue on the next page

S-boxes overview (continued).

S	S_{NL}	S_{δ}	S_{AC}	S_{DEG}	Spectra
Kalyna π_1	104	8	72	7	6280, 11426, 10769, 9948, 8500, 6970, 5112, 3267, 1855, 907, 351, 122, 28
Kalyna π_2	104	8	72	7	6307, 11643, 10808, 9681, 8540, 6895, 4981, 3415, 1854, 898, 363, 108, 42
Kalyna π_3	104	8	72	7	6371, 11451, 10804, 9887, 8422, 6985, 5019, 3310, 1878, 883, 361, 124, 40
Khazad	96	8	104	7	7030, 12594, 11426, 9876, 7573, 5938, 4268, 2836, 1877, 1058, 524, 264, 153, 58, 30, 16, 14
Kuznechik	100	8	96	7	6534, 11645, 10761, 10166, 8793, 6804, 4474, 2796, 1693, 971, 535, 219, 91, 39, 14
MD2	90	10	88	6	7082, 12669, 11306, 9702, 7819, 5899, 4244, 2850, 1776, 1033, 619, 286, 151, 60, 22, 11, 3, 1, 1, 1
newDES	92	12	96	6	6995, 12491, 11309, 9828, 7860, 6021, 4355, 2784, 1765, 1020, 557, 282, 154, 71, 25, 14, 1, 1, 2
Scream	96	8	128	3	19583, 0, 11392, 0, 23952, 0, 4416, 0, 5344, 0, 576, 0, 112, 0, 0, 0, 160,
SKINNY8	64	64	128	2	40383, 0, 5810, 0, 9904, 0, 1846, 0, 5624, 0, 388, 0, 464, 0, 118, 0, 892, 0, 22, 0, 0, 0, 4, 0, 24, 0, 2, 0, 0, 0, 2, 0, 52

Continue on the next page

S-boxes overview (continued).

S	S_{NL}	S_δ	S_{AC}	S_{DEG}	Spectra
Skipjack	100	12	96	6	7005, 12456, 11244, 9799, 7882, 6032, 4354, 2813, 1814, 1041, 567, 317, 154, 54, 3
SNOW3G	96	8	96	5	25715, 0, 0, 0, 32324, 0, 0, 0, 6924, 0, 0, 0, 556, 0, 0, 0, 16
Streebog	100	8	96	7	6534, 11645, 10761, 10166, 8793, 6804, 4474, 2796, 1693, 971, 535, 219, 91, 39, 14
Turing	94	12	96	6	6998, 12383, 11444, 9705, 7997, 5922, 4320, 2852, 1800, 1045, 551, 265, 145, 72, 21, 11, 3, 1
Twofish p_0	96	10	128	6	14221, 0, 22376, 0, 15506, 0, 8298, 0, 3545, 0, 1194, 0, 314, 0, 68, 0, 13
Twofish p_1	96	10	112	6	14151, 0, 22385, 0, 15667, 0, 8209, 0, 3512, 0, 1189, 0, 353, 0, 57, 0, 12
AES ARIA S_2 Camellia CLEFIA S_1 DBlock Hierocrypt3 Hierocrypt31 SEED S_0 SEED S_1 SMS4 ZUC S_1	112	4	32	7	4590, 12240, 9180, 10200, 8670, 6120, 9180, 4080, 1275
Whirlpool	100	8	96	7	7399, 12400, 11084, 9960, 7580, 5882, 4229, 3028, 1870, 1048, 563, 260, 134, 62, 36

Continue on the next page

S-boxes overview (continued).

S	S_{NL}	S_{δ}	S_{AC}	S_{DEG}	Spectra
ZUC S_0	96	8	128	5	26655, 0, 0, 0, 31200, 0, 0, 0, 7232, 0, 0, 0, 288, 0, 0, 0, 160
Zorro	96	10	112	5	10115, 6322, 17157, 4921, 11652, 3036, 6519, 1393, 2562, 536, 891, 131, 208, 36, 41, 9, 6

A.2 Collisions search by using absolute LAT spectra

Table A.2 Collisions search by using absolute LAT spectra

Sbox	Collision Γ	$\|I\|$	I
FLY	$\Gamma(\cdot,4,0,8,I)$	12	22, 54, 86, 97, 99, 101, 103, 105, 107, 118, 150, 182
FLY	$\Gamma(\cdot,4,0,8,I)$	8	60, 94, 188, 195, 203, 207, 229, 252
FLY	$\Gamma(\cdot,4,0,8,I)$	12	30, 126, 159, 189, 190, 219, 225, 231, 235, 239, 249, 254
PICARO	$\Gamma(\cdot,4,2,6,I)$	96	0, 1, 9, 11, 13, 14, 16, 17, 25, 27, 29, 30, 32, 33, 41, 43, 45, 46, 48, 49, 57, 59, 61, 62, 64, 65, 73, 75, 77, 78, 80, 81, 89, 91, 93, 94, 96, 97, 105, 107, 109, 110, 112, 113, 121, 123, 125, 126, 128, 129, 137, 139, 141, 142, 144, 145, 153, 155, 157, 158, 160, 161, 169, 171, 173, 174, 176, 177, 185, 187, 189, 190, 192, 193, 201, 203, 205, 206, 208, 209, 217, 219, 221, 222, 224, 225, 233, 235, 237, 238, 240, 241, 249, 251, 253, 254
PICARO	$\Gamma(\cdot,4,0,8,I)$	16	9, 11, 13, 14, 29, 43, 57, 78, 109, 125, 137, 174, 185, 219, 238, 251

Continue on the next page

Collisions search by using absolute LAT spectra (continued)

Sbox	Collision Γ	$\|I\|$	I
PICARO	$\Gamma(\cdot,4,2,10,I)$	96	0, 1, 9, 11, 13, 14, 16, 17, 25, 27, 29, 30, 32, 33, 41, 43, 45, 46, 48, 49, 57, 59, 61, 62, 64, 65, 73, 75, 77, 78, 80, 81, 89, 91, 93, 94, 96, 97, 105, 107, 109, 110, 112, 113, 121, 123, 125, 126, 128, 129, 137, 139, 141, 142, 144, 145, 153, 155, 157, 158, 160, 161, 169, 171, 173, 174, 176, 177, 185, 187, 189, 190, 192, 193, 201, 203, 205, 206, 208, 209, 217, 219, 221, 222, 224, 225, 233, 235, 237, 238, 240, 241, 249, 251, 253, 254
PICARO	$\Gamma(\cdot,4,2,10,I)$	12	19, 34, 52, 69, 99, 115, 132, 165, 180, 210, 229, 242
PICARO	$\Gamma(\cdot,4,2,10,I)$	12	26, 44, 63, 72, 106, 122, 143, 168, 191, 220, 232, 252
PICARO	$\Gamma(\cdot,4,2,10,I)$	12	82, 83, 84, 85, 146, 147, 148, 149, 194, 195, 196, 197
PICARO	$\Gamma(\cdot,4,2,10,I)$	12	23, 39, 54, 70, 103, 119, 134, 166, 182, 215, 230, 247
PICARO	$\Gamma(\cdot,4,4,12,I)$	204	0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 42, 43, 44, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 60, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 74, 76, 78, 79, 80, 82, 83, 84, 85, 86, 87, 88, 90, 92, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 106, 108, 109, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 122, 124, 125, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 140, 143, 144, 146, 147, 148, 149, 150, 151, 152, 154, 156, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 170, 172, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 188, 191, 192, 194, 195, 196, 197, 198, 199, 200, 202, 204, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 218, 219, 220, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 234, 236, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 250, 251, 252, 255

Continue on the next page

Collisions search by using absolute LAT spectra (continued)

Sbox	Collision Γ	$\|I\|$	I
PICARO	$\Gamma(\cdot,4,10,14,I)$	96	0, 1, 9, 11, 13, 14, 16, 17, 25, 27, 29, 30, 32, 33, 41, 43, 45, 46, 48, 49, 57, 59, 61, 62, 64, 65, 73, 75, 77, 78, 80, 81, 89, 91, 93, 94, 96, 97, 105, 107, 109, 110, 112, 113, 121, 123, 125, 126, 128, 129, 137, 139, 141, 142, 144, 145, 153, 155, 157, 158, 160, 161, 169, 171, 173, 174, 176, 177, 185, 187, 189, 190, 192, 193, 201, 203, 205, 206, 208, 209, 217, 219, 221, 222, 224, 225, 233, 235, 237, 238, 240, 241, 249, 251, 253, 254
PICARO	$\Gamma(\cdot,4,10,14,I)$	12	28, 42, 56, 79, 108, 124, 136, 175, 184, 218, 239, 250
PICARO	$\Gamma(\cdot,4,12,14,I)$	12	23, 39, 54, 70, 103, 119, 134, 166, 182, 215, 230, 247
Iraqi	$\Gamma(\cdot,4,0,4,I)$	130	0, 2, 3, 4, 5, 10, 11, 12, 13, 18, 19, 20, 21, 26, 27, 28, 29, 32, 33, 38, 39, 40, 41, 42, 46, 47, 48, 49, 54, 55, 56, 57, 62, 63, 66, 67, 68, 69, 74, 75, 76, 77, 82, 83, 84, 85, 90, 91, 92, 93, 96, 97, 102, 103, 104, 105, 110, 111, 112, 113, 118, 119, 120, 121, 126, 127, 128, 129, 134, 135, 136, 137, 142, 143, 144, 145, 150, 151, 152, 153, 158, 159, 162, 163, 164, 165, 170, 171, 172, 173, 178, 179, 180, 181, 186, 187, 188, 189, 192, 193, 198, 199, 200, 201, 206, 207, 208, 209, 214, 215, 216, 217, 222, 223, 226, 227, 228, 229, 234, 235, 236, 237, 242, 243, 244, 245, 250, 251, 252, 253
Iraqi	$\Gamma(\cdot,4,2,6,I)$	129	0, 2, 3, 4, 5, 10, 11, 12, 13, 18, 19, 20, 21, 26, 27, 28, 29, 32, 33, 38, 39, 40, 41, 46, 47, 48, 49, 54, 55, 56, 57, 62, 63, 66, 67, 68, 69, 74, 75, 76, 77, 82, 83, 84, 85, 90, 91, 92, 93, 96, 97, 102, 103, 104, 105, 110, 111, 112, 113, 118, 119, 120, 121, 126, 127, 128, 129, 134, 135, 136, 137, 142, 143, 144, 145, 150, 151, 152, 153, 158, 159, 162, 163, 164, 165, 170, 171, 172, 173, 178, 179, 180, 181, 186, 187, 188, 189, 192, 193, 198, 199, 200, 201, 206, 207, 208, 209, 214, 215, 216, 217, 222, 223, 226, 227, 228, 229, 234, 235, 236, 237, 242, 243, 244, 245, 250, 251, 252, 253

Continue on the next page

Collisions search by using absolute LAT spectra (continued)

Sbox	Collision Γ	$\lvert I\rvert$	I
FOX	$\Gamma(\cdot,4,4,12,I)$	16	0, 17, 34, 51, 68, 85, 102, 119, 136, 153, 170, 187, 204, 221, 238, 255
Fantomas	$\Gamma(\cdot,4,0,8,I)$	10	102, 110, 114, 122, 195, 215, 230, 238, 242, 250
Fantomas	$\Gamma(\cdot,4,4,12,I)$	128	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 36, 40, 44, 48, 52, 56, 60, 64, 65, 68, 69, 72, 73, 76, 77, 80, 81, 84, 85, 88, 89, 92, 93, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 154, 156, 158, 160, 164, 168, 172, 176, 180, 184, 188, 192, 195, 196, 199, 200, 203, 204, 207, 208, 211, 212, 215, 216, 219, 220, 223, 224, 226, 228, 230, 232, 234, 236, 238, 240, 242, 244, 246, 248, 250, 252, 254
Lilliput	$\Gamma(\cdot,4,0,8,I)$	16	80, 83, 85, 86, 112, 115, 117, 118, 224, 226, 229, 231, 232, 234, 237, 239
Lilliput	$\Gamma(\cdot,4,4,12,I)$	64	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191
Lilliput	$\Gamma(\cdot,4,4,12,I)$	32	19, 23, 25, 29, 80, 86, 98, 106, 115, 117, 131, 134, 137, 140, 163, 166, 169, 172, 192, 194, 196, 198, 200, 202, 204, 206, 224, 226, 233, 235, 243, 255
Lilliput	$\Gamma(\cdot,4,4,12,I)$	8	20, 26, 50, 60, 132, 142, 164, 174
CMEA	$\Gamma(\cdot,4,0,4,I)$	5	0, 56, 118, 178, 252
CMEA	$\Gamma(\cdot,4,2,6,I)$	12	0, 5, 61, 75, 78, 115, 138, 143, 183, 193, 196, 249
CryptonS0	$\Gamma(\cdot,4,0,8,I)$	19	96, 103, 104, 105, 111, 117, 124, 162, 164, 167, 168, 171, 173, 174, 197, 207, 231, 235, 236
CryptonS0	$\Gamma(\cdot,4,4,12,I)$	12	0, 16, 18, 47, 76, 97, 142, 163, 192, 239, 253, 255
CryptonS0	$\Gamma(\cdot,4,4,12,I)$	5	99, 158, 179, 208, 210
SKINNY	$\Gamma(\cdot,4,0,8,I)$	8	193, 197, 201, 205, 225, 229, 233, 237
SKINNY	$\Gamma(\cdot,4,0,8,I)$	8	74, 78, 106, 110, 202, 206, 234, 238

Continue on the next page

Collisions search by using absolute LAT spectra (continued)

Sbox	Collision Γ	$\lvert I\rvert$	I
SKINNY	$\Gamma(\cdot,4,0,8,I)$	12	21, 23, 29, 31, 55, 63, 151, 159, 181, 183, 189, 191
SKINNY	$\Gamma(\cdot,4,4,12,I)$	128	0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 194, 196, 198, 200, 202, 204, 206, 208, 210, 212, 214, 216, 218, 220, 222, 224, 226, 228, 230, 232, 234, 236, 238, 240, 242, 244, 246, 248, 250, 252, 254
SKINNY	$\Gamma(\cdot,4,4,12,I)$	12	19, 21, 27, 29, 147, 149, 155, 157, 195, 199, 203, 207
SKINNY	$\Gamma(\cdot,4,4,12,I)$	8	83, 87, 91, 95, 243, 247, 251, 255
SKINNY	$\Gamma(\cdot,4,4,12,I)$	36	17, 23, 25, 31, 49, 55, 57, 63, 81, 85, 89, 93, 117, 125, 145, 151, 153, 159, 177, 183, 185, 191, 193, 197, 201, 205, 213, 221, 225, 229, 233, 237, 241, 245, 249, 253
SKINNY	$\Gamma(\cdot,4,0,16,I)$	8	30, 62, 90, 122, 158, 190, 218, 250
SKINNY	$\Gamma(\cdot,4,12,20,I)$	8	133, 135, 141, 143, 165, 167, 173, 175
SKINNY	$\Gamma(\cdot,4,16,24,I)$	12	21, 23, 29, 31, 55, 63, 151, 159, 181, 183, 189, 191
ZUCS0	$\Gamma(\cdot,4,0,8,I)$	16	22, 23, 54, 55, 86, 87, 118, 119, 150, 151, 182, 183, 214, 215, 246, 247
ZUCS0	$\Gamma(\cdot,4,0,8,I)$	32	6, 7, 26, 27, 38, 39, 58, 59, 70, 71, 90, 91, 102, 103, 122, 123, 134, 135, 154, 155, 166, 167, 186, 187, 198, 199, 218, 219, 230, 231, 250, 251
Kuznyechik	$\Gamma(\cdot,4,2,14,I)$	16	0, 26, 32, 58, 68, 94, 100, 126, 138, 144, 170, 176, 206, 212, 238, 244
Kuznyechik	$\Gamma(\cdot,4,2,16,I)$	16	0, 26, 32, 58, 68, 94, 100, 126, 138, 144, 170, 176, 206, 212, 238, 244
Scream	$\Gamma(\cdot,4,0,8,I)$	8	67, 71, 75, 79, 99, 103, 107, 111
Scream	$\Gamma(\cdot,4,0,8,I)$	8	200, 201, 202, 203, 204, 205, 206, 207
Scream	$\Gamma(\cdot,4,0,8,I)$	12	88, 90, 92, 94, 112, 114, 116, 118, 121, 123, 125, 127

Continue on the next page

Collisions search by using absolute LAT spectra (continued)

Sbox	Collision Γ	$\|I\|$	I
Scream	$\Gamma(\cdot,4,4,12,I)$	64	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191
Scream	$\Gamma(\cdot,4,4,12,I)$	6	23, 25, 49, 52, 58, 63
CSS	$\Gamma(\cdot,4,0,16,I)$	128	4, 5, 6, 7, 8, 9, 10, 11, 20, 21, 22, 23, 24, 25, 26, 27, 36, 37, 38, 39, 40, 41, 42, 43, 52, 53, 54, 55, 56, 57, 58, 59, 64, 65, 66, 67, 76, 77, 78, 79, 80, 81, 82, 83, 92, 93, 94, 95, 96, 97, 98, 99, 108, 109, 110, 111, 112, 113, 114, 115, 124, 125, 126, 127, 128, 129, 130, 131, 140, 141, 142, 143, 144, 145, 146, 147, 156, 157, 158, 159, 160, 161, 162, 163, 172, 173, 174, 175, 176, 177, 178, 179, 188, 189, 190, 191, 196, 197, 198, 199, 200, 201, 202, 203, 212, 213, 214, 215, 216, 217, 218, 219, 228, 229, 230, 231, 232, 233, 234, 235, 244, 245, 246, 247, 248, 249, 250, 251
SNOW3G	$\Gamma(\cdot,4,0,8,I)$	8	56, 98, 100, 104, 172, 202, 216, 232
SNOW3G	$\Gamma(\cdot,4,0,8,I)$	8	59, 81, 131, 141, 205, 207, 213, 251
SNOW3G	$\Gamma(\cdot,4,0,8,I)$	8	22, 28, 38, 122, 150, 178, 180, 198
SNOW3G	$\Gamma(\cdot,4,0,8,I)$	8	32, 42, 44, 112, 130, 144, 160, 228
SNOW3G	$\Gamma(\cdot,4,0,8,I)$	8	9, 67, 119, 127, 129, 143, 145, 149
SNOW3G	$\Gamma(\cdot,4,0,8,I)$	8	61, 71, 89, 97, 137, 159, 235, 247
SNOW3G	$\Gamma(\cdot,4,0,8,I)$	10	7, 13, 40, 53, 83, 96, 99, 101, 153, 187
SNOW3G	$\Gamma(\cdot,4,0,8,I)$	8	25, 115, 133, 135, 157, 179, 197, 203
SNOW3G	$\Gamma(\cdot,4,0,8,I)$	24	14, 15, 17, 23, 26, 41, 52, 87, 105, 107, 117, 118, 156, 163, 169, 189, 191, 193, 210, 215, 238, 239, 241, 246
SNOW3G	$\Gamma(\cdot,4,0,8,I)$	8	19, 121, 155, 165, 173, 181, 227, 231
SNOW3G	$\Gamma(\cdot,4,0,8,I)$	8	31, 33, 35, 95, 167, 185, 225, 245
iScream	$\Gamma(\cdot,4,0,8,I)$	8	129, 145, 161, 177, 193, 209, 225, 241
iScream	$\Gamma(\cdot,4,4,12,I)$	32	0, 1, 16, 17, 32, 33, 48, 49, 64, 65, 80, 81, 96, 97, 112, 113, 128, 129, 144, 145, 160, 161, 176, 177, 192, 193, 208, 209, 224, 225, 240, 241

Continue on the next page

Collisions search by using absolute LAT spectra (continued)

Sbox	Collision Γ	$\lvert I \rvert$	I
iScream	$\Gamma(\cdot,4,4,12,I)$	8	34, 38, 50, 54, 71, 87, 99, 115
Zorro	$\Gamma(\cdot,4,2,6,I)$	128	0, 1, 4, 5, 10, 11, 14, 15, 18, 19, 22, 23, 24, 25, 28, 29, 34, 35, 38, 39, 40, 41, 44, 45, 48, 49, 52, 53, 58, 59, 62, 63, 64, 65, 68, 69, 74, 75, 78, 79, 82, 83, 86, 87, 88, 89, 92, 93, 98, 99, 102, 103, 104, 105, 108, 109, 112, 113, 116, 117, 122, 123, 126, 127, 128, 129, 132, 133, 138, 139, 142, 143, 146, 147, 150, 151, 152, 153, 156, 157, 162, 163, 166, 167, 168, 169, 172, 173, 176, 177, 180, 181, 186, 187, 190, 191, 192, 193, 196, 197, 202, 203, 206, 207, 210, 211, 214, 215, 216, 217, 220, 221, 226, 227, 230, 231, 232, 233, 236, 237, 240, 241, 244, 245, 250, 251, 254, 255
CS	$\Gamma(\cdot,4,0,8,I)$	24	36, 37, 38, 39, 40, 41, 42, 43, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175
CS	$\Gamma(\cdot,4,0,8,I)$	16	224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239
CS	$\Gamma(\cdot,4,0,8,I)$	16	80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95
CS	$\Gamma(\cdot,4,0,8,I)$	16	176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191
CS	$\Gamma(\cdot,4,0,8,I)$	16	112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127

A.3 Collisions search by using absolute transposed LAT spectra

Table A.3 Collisions search by using absolute transposed LAT spectra

Sbox	Collision Γ	$\|I\|$	I
BelT	$\Gamma(\cdot,4,0,8,I^T)$	16	1, 2, 3, 4, 6, 8, 12, 16, 24, 32, 48, 64, 96, 128, 192
BelT	$\Gamma(\cdot,4,0,8,I^T)$	6	7, 14, 28, 56, 80, 160
BelT	$\Gamma(\cdot,4,8,12,I^T)$	6	7, 14, 28, 56, 80, 160

A.4 Collisions search by using DDT spectra

Table A.4 Collisions search by using DDT spectra

Sbox	Collision Γ	$\|I\|$	I
FLY	$\Gamma(\cdot,4,0,4,I)$	8	6, 7, 14, 15, 96, 112, 224, 240
FLY	$\Gamma(\cdot,4,4,8,I)$	8	6, 7, 14, 15, 96, 112, 224, 240
PICARO	$\Gamma(\cdot,0,2,12,I)$	12	23, 57, 73, 93, 107, 121, 139, 157, 167, 183, 205, 235
PICARO	$\Gamma(\cdot,0,2,12,I)$	12	19, 53, 69, 95, 97, 117, 129, 159, 163, 179, 207, 225
PICARO	$\Gamma(\cdot,0,2,12,I)$	10	2, 4, 6, 7, 9, 10, 11, 12, 13, 14
Lilliput	$\Gamma(\cdot,0,4,8,I)$	8	16, 32, 48, 80, 96, 112, 160, 224
SKINNY	$\Gamma(\cdot,0,0,2,I)$	12	232, 233, 234, 235, 236, 237, 248, 249, 250, 251, 252, 253
SKINNY	$\Gamma(\cdot,0,0,4,I)$	8	53, 54, 98, 99, 114, 115, 197, 213
SKINNY	$\Gamma(\cdot,0,4,12,I)$	6	24, 25, 26, 27, 42, 43
Kalyna π_3	$\Gamma(\cdot,0,0,2,I)$	6	65, 67, 127, 145, 151, 250
ZUCS0	$\Gamma(\cdot,0,0,2,I)$	8	133, 135, 141, 143, 149, 151, 157, 159
ZUCS0	$\Gamma(\cdot,0,0,2,I)$	16	66, 70, 74, 78, 82, 86, 90, 94, 209, 211, 213, 215, 217, 219, 221, 223
ZUCS0	$\Gamma(\cdot,0,0,4,I)$	5	4, 6, 10, 20, 28
ZUCS0	$\Gamma(\cdot,0,4,6,I)$	8	224, 226, 228, 230, 240, 242, 244, 246
ZUCS0	$\Gamma(\cdot,0,0,8,I)$	8	35, 39, 43, 47, 51, 55, 59, 63

Continue on the next page

Collisions search by using DDT spectra (continued)

Sbox	Collision Γ	$\|I\|$	I
ZUCS0	$\Gamma(\cdot,0,0,8,I)$	8	161, 163, 169, 171, 177, 179, 185, 187
Kuznyechik	$\Gamma(\cdot,0,2,4,I)$	5	4, 18, 36, 38, 48
Scream	$\Gamma(\cdot,0,0,2,I)$	8	33, 49, 97, 113, 161, 177, 225, 241
Scream	$\Gamma(\cdot,0,0,8,I)$	8	32, 64, 80, 96, 160, 192, 208, 224
CSS	$\Gamma(\cdot,0,0,16,I)$	108	17, 18, 19, 21, 22, 23, 25, 26, 27, 29, 30, 31, 33, 34, 35, 37, 38, 39, 41, 42, 43, 45, 46, 47, 53, 54, 55, 57, 58, 59, 85, 86, 87, 89, 90, 91, 101, 102, 103, 105, 106, 107, 113, 114, 115, 117, 118, 119, 121, 122, 123, 125, 126, 127, 149, 150, 151, 153, 154, 155, 165, 166, 167, 169, 170, 171, 177, 178, 179, 181, 182, 183, 185, 186, 187, 189, 190, 191, 209, 210, 211, 213, 214, 215, 217, 218, 219, 221, 222, 223, 225, 226, 227, 229, 230, 231, 233, 234, 235, 237, 238, 239, 245, 246, 247, 249, 250, 251
SNOW3G	$\Gamma(\cdot,0,0,2,I)$	8	67, 87, 97, 103, 138, 170, 192, 243
SNOW3G	$\Gamma(\cdot,0,0,2,I)$	8	128, 133, 135, 144, 238, 248, 251, 255
SNOW3G	$\Gamma(\cdot,0,0,4,I)$	8	67, 87, 97, 103, 138, 170, 192, 243
SNOW3G	$\Gamma(\cdot,0,2,4,I)$	24	22, 23, 42, 63, 70, 83, 98, 101, 110, 111, 113, 118, 122, 123, 124, 125, 137, 155, 169, 175, 194, 196, 227, 241
CS	$\Gamma(\cdot,0,0,2,I)$	8	131, 147, 163, 179, 195, 211, 227, 243
CS	$\Gamma(\cdot,0,0,2,I)$	8	11, 27, 43, 59, 203, 219, 235, 251

A.5 Collisions search by using transposed DDT spectra

Table A.5 Collisions search by using transposed DDT spectra

Sbox	Collision Γ	$\|I\|$	I
Fox	$\Gamma(\cdot,4,0,8,I^T)$	8	34, 51, 102, 119, 153, 170, 204, 255
Kalyna π_3	$\Gamma(\cdot,0,0,2,I^T)$	5	1, 12, 55, 76, 199
BelT	$\Gamma(\cdot,0,0,2,I^T)$	9	28, 31, 67, 98, 114, 124, 201, 216, 223

Continue on the next page

Collisions search by using transposed DDT spectra (continued)

Sbox	Collision Γ	$\|I\|$	I
BelT	$\Gamma(\cdot,0,0,2,I^T)$	6	57, 64, 143, 156, 184, 224
BelT	$\Gamma(\cdot,0,0,2,I^T)$	6	24, 32, 46, 56, 139, 195
BelT	$\Gamma(\cdot,0,0,2,I^T)$	5	16, 30, 34, 113, 197
BelT	$\Gamma(\cdot,0,0,2,I^T)$	5	11, 17, 88, 120, 176
BelT	$\Gamma(\cdot,0,2,4,I^T)$	5	11, 17, 88, 120, 176
BelT	$\Gamma(\cdot,0,2,4,I^T)$	5	16, 30, 34, 113, 197
BelT	$\Gamma(\cdot,0,2,4,I^T)$	6	24, 32, 46, 56, 139, 195

A.6 Collisions search by using ACT spectra

Table A.6 Collisions search by using ACT spectra (some results are omitted)

Sbox	Collision Γ	$\|I\|$	I
FLY	$\Gamma(\cdot,4,0,16,I)$	12	1, 3, 5, 7, 13, 15, 16, 48, 80, 112, 208, 240
PICARO	$\Gamma(\cdot,4,0,16,I)$	12	88, 90, 92, 95, 152, 154, 156, 159, 200, 202, 204, 207
Iraqi	$\Gamma(\cdot,4,0,8,I)$	129	0, 2, 3, 4, 5, 10, 11, 12, 13, 18, 19, 20, 21, 26, 27, 28, 29, 32, 33, 38, 39, 40, 41, 46, 47, 48, 49, 54, 55, 56, 57, 62, 63, 66, 67, 68, 69, 74, 75, 76, 77, 82, 83, 84, 85, 90, 91, 92, 93, 96, 97, 102, 103, 104, 105, 110, 111, 112, 113, 118, 119, 120, 121, 126, 127, 128, 129, 134, 135, 136, 137, 142, 143, 144, 145, 150, 151, 152, 153, 158, 159, 162, 163, 164, 165, 170, 171, 172, 173, 178, 179, 180, 181, 186, 187, 188, 189, 192, 193, 198, 199, 200, 201, 206, 207, 208, 209, 214, 215, 216, 217, 222, 223, 226, 227, 228, 229, 234, 235, 236, 237, 242, 243, 244, 245, 250, 251, 252, 253
Fox	$\Gamma(\cdot,4,8,24,I)$	16	0, 17, 34, 51, 68, 85, 102, 119, 136, 153, 170, 187, 204, 221, 238, 255
Fantomas	$\Gamma(\cdot,4,0,64,I)$	5	84, 136, 148, 200, 212

Continue on the next page

Collisions search by using ACT spectra (continued)

Sbox	Collision Γ	$\lvert I\rvert$	I
Lilliput	$\Gamma(\cdot,4,8,24,I)$	24	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 40, 41, 42, 43, 44, 45, 46, 47
Crypton S0	$\Gamma(\cdot,4,8,24,I)$	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
SKINNY	$\Gamma(\cdot,4,0,64,I)$	16	20, 24, 52, 56, 84, 88, 116, 120, 132, 140, 164, 172, 196, 204, 228, 236
ZUCS0	$\Gamma(\cdot,4,0,8,I)$	16	20, 21, 52, 53, 84, 85, 116, 117, 148, 149, 180, 181, 212, 213, 244, 245
Kuznyechik	$\Gamma(\cdot,4,0,16,I)$	5	68, 94, 138, 144, 212
Kuznyechik	$\Gamma(\cdot,4,32,40,I)$	16	0, 26, 32, 58, 68, 94, 100, 126, 138, 144, 170, 176, 206, 212, 238, 244
Scream	$\Gamma(\cdot,4,8,24,I)$	32	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47
CSS	$\Gamma(\cdot,4,0,64,I)$	64	68, 69, 70, 71, 72, 73, 74, 75, 84, 85, 86, 87, 88, 89, 90, 91, 100, 101, 102, 103, 104, 105, 106, 107, 116, 117, 118, 119, 120, 121, 122, 123, 132, 133, 134, 135, 136, 137, 138, 139, 148, 149, 150, 151, 152, 153, 154, 155, 164, 165, 166, 167, 168, 169, 170, 171, 180, 181, 182, 183, 184, 185, 186, 187
SNOW3G	$\Gamma(\cdot,4,0,8,I)$	8	9, 67, 119, 127, 129, 143, 145, 149
iScream	$\Gamma(\cdot,4,0,16,I)$	8	129, 145, 161, 177, 193, 209, 225, 241
Zorro	$\Gamma(\cdot,4,8,24,I)$	8	0, 49, 86, 103, 129, 176, 215, 230
CS	$\Gamma(\cdot,4,0,8,I)$	16	224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239

Appendix B

Binary Sequences

B.1 Shotgun Hill climbing results

Table B.1 An overview of the shotgun hill climbing algorithm results

n	Old	New	Binary sequence in HEX	db	MF
106	7	6	1366453fff339abc3d613eab4f2	-24.943	5.030
107	7	6	3e525b707207bb6280c08c733aa	-25.025	4.497
108	7	6	9d31b81bc465b48ab7ae0801834	-25.105	5.533
109	7	6	1c80e7c337e7ea64d55da750ca5b	-25.186	5.636
110	7	6	825bebaee519f060d42d81cc8d4	-23.926	5.984
111	7	6	1cb387b52c8ed4cfeb048855305c	-24.004	5.138
112	7	6	68a5614a61368ddf1743207fe706	-24.082	4.931
113	7	6	1ae5cb4fe90feae29779ec120644e	-24.160	4.409
114	7	6	19ed6101bcf959e19a5583a622e81	-24.236	5.375
115	7	7	56d413e9ca1c1992f37994f88c502	-24.312	3.952
116	7	7	43f475cbd4e3b98d5d0cb6c4840db	-24.387	3.925
117	7	7	5d8caed643dfa1480b11c347164c1	-24.462	4.108
118	7	7	3ce9d9c9ad524fb5f415fade2e1186	-24.536	3.976
119	7	7	4b24ce6b455b8b02001de1753c5297	-24.609	4.331
120	7	7	d91e13e197ad463b9e2d9d5fed2544	-24.682	4.639
121	7	7	3fbd241b987f4b8ed966614a888e89	-24.754	4.141
122	7	7	28d7ab4e488ce60018781f34d704ae9	-24.825	3.999
123	8	7	7d9b6c7bf11e94507c2556d6e6a8c31	-24.896	3.736

Continue on the next page

An overview of the shotgun hill climbing algorithm results (continued)

n	Old	New	Binary sequence in HEX	db	MF
124	8	7	703ffe14662bdc7cd3f4eb262a49a93	-24.966	4.884
125	8	7	a8e0e42fc6af59cfb7b640cff64bb2c	-25.036	4.134
126	8	7	ca666b72aa45167f4cc39f00521c2d2	-25.105	4.544
127	7	7	73fef5c8d1d95d05cc26917ce097bc2d	-25.174	4.815
128	8	7	915fca044f23e83a942393ada7bb73e7	-25.242	4.911
129	8	7	1856351aa9ada9798eb0070b267d80836	-25.310	3.955
130	8	7	7ea03973917046150ca103459afb7b49	-25.377	4.692
131	8	7	169633c2e13890d5e540afdd64c811c09	-25.443	4.403
132	8	7	f4bf06b4afe88af3c79dd76badcd94c8	-24.350	4.431
133	8	7	18fe45f33afd90cba4888b9d2b534841e1	-24.415	4.323
134	8	8	2b35983c61b4f3bbf3752c69fabe0897a8	-24.480	3.742
135	8	8	12c3755bb64459418f4a242e731e1697e	-24.545	3.876
136	8	8	30ed813f6f583c925aaa2f53e6722f5bcf	-24.609	3.925
137	8	8	d569ca74eebccc573b0208187a6f82fa09	-24.673	4.066
138	8	8	3d128917da3431938e6dfd1ef7a2e68bc2f	-24.736	3.771
139	8	8	3c0e1d9b35f9bd5342a80db491c406d6f10	-24.798	3.808
140	8	8	bdcf8e3944f5b152fbbbf01b66a2d0b890a	-24.861	4.026
141	9	8	115e1f52e273d156c9af48cc8007b6c649e 5	-24.923	3.923
142	8	8	71338901166bd08b7d05ac1a4edf87d1531	-24.984	3.724
143	8	8	67aa81c2c56fde794f6365fc0b30db92253 7	-25.045	3.940
144	8	8	39716d38490502a3765215eb20ee1bb84ca 3	-25.105	3.886
145	8	8	1791bb0ba63bccda7c2a3678dfd6825c792 a0	-25.166	4.477
146	8	8	3708999ea4c1f08e12ae8ebcdf092d1215a 20	-25.225	3.975
147	9	8	40c48cac0843a2f917ccab14215dd87b792 c7	-25.285	4.122
148	8	8	2c24f9cb675dcd540bb0943d629030d83cd c0	-25.343	5.291

Continue on the next page

An overview of the shotgun hill climbing algorithm results (continued)

n	Old	New	Binary sequence in HEX	db	MF
149	8	8	5a0f857ae7b62266299eee68a141d70085a 58	-25.402	3.698
150	9	8	3d63df1b948ddc2689a895072984b2ba7e6 008	-25.460	4.329
151	8	8	2640cc90388e31881fe5535d5ac2c2456f2 f16	-25.518	3.999
152	9	8	6501a71c13b1fec21d82cfddb2bb3a5d569 536	-25.575	4.068
153	9	8	1752e3434eae633cc3817375b05becd5f40 5224	-25.632	3.917
154	9	8	3cbe58528eb47f0efe6afbc2ed521dcf988 626d	-25.689	3.864
155	8	8	aaa430985f6a183d3fc9edd8217b0732ef1 b74	-25.745	3.987
156	9	8	dcaf489c2264f8ff9aeb8d7433f708165a1 6928	-25.801	4.713
157	8	8	8c91dbe342975ba661d860071a06d745771 7b0b	-25.856	4.241
158	9	8	11f07cda85b2c794875eea635521ffdf727 5c666	-25.911	4.479
159	9	8	665f717b678d7c472844d61aad3a2e77814 1dbfb	-25.966	4.366
160	9	8	62088e74b483f5cf4daeb02e3d169de44e9 cd5df	-26.021	3.765
161	9	8	e720b7b8987caaa3ca7e454a0ecc9108245 a5cf	-26.075	4.096
162	9	8	112db024584a1c7a44aa9b729ab138c0531 f8bf83	-26.129	4.408
163	9	8	5af97f061a5a10317fa15510778b32ce219 9c89c2	-26.182	4.104
164	9	8	10f81f8297d4226c9428d39b575b9cab2f3 f9a18a	-26.235	4.189

Continue on the next page

An overview of the shotgun hill climbing algorithm results (continued)

n	Old	New	Binary sequence in HEX	db	MF
165	9	8	b8089b446cab2ffa99c97939df6953879e4 6bc6f8	-26.288	4.394
166	9	8	856dff4fad1b0b93a6195558e3130d69940 67e81a	-26.340	4.212
167	8	8	ecad8e1a6be074ff88322c5b3cd5680dc82 5a1198	-26.393	4.656
168	9	8	37f80dbe33864f68ef1fa5a951eeecd274d 5c6c506	-25.421	4.324
169	9	9	18a745f218b371c6f21132f7f2f3ec9d290 f9eaae6f	-25.473	3.636
170	9	9	3835d25fe470f32ed7c4ccabe4f2b5e6601 1584a5bb	-25.524	4.234
171	9	9	20acaa4d24c6028139c5fd39f3065ca87cd 082f5f84	-25.575	4.171
172	9	9	9c92f90c3ec2109c08862ec8ea5be45911d 7abb6143	-25.626	3.928
173	9	9	1e58f6cad6917eeaee691536d57df81c5cb 901c43387	-25.676	4.025
174	9	9	6c99808556a9e44f04a4af397f90dac63b5 c151f770	-25.726	3.797
175	9	9	810552f57861b5543b90c9bc298de721699 f922627	-25.776	3.923
176	9	9	ac277413353446ebbec34fbda6a08305ea7 07e8b14a3	-25.825	3.792
177	9	9	bfd3bffc44db1369bde8c4956de06a2f3cc e38a9d0f9	-25.875	4.366
178	9	9	c40a317538cacb189615811a82f8a6da26c bc12fff85	-25.924	3.905
179	9	9	755e7001560439f469090f9492191af2766 0ba19b2555	-25.972	3.953
180	10	9	20e89f547a266727ad2c0e2dfbfab4eb790 0d6f11e714	-26.021	4.147

Continue on the next page

An overview of the shotgun hill climbing algorithm results (continued)

n	Old	New	Binary sequence in HEX	db	MF
181	9	9	112569db006c60067a7aee0fc75d29142a7 734da259170	-26.069	4.143
182	10	9	55c099a8f91f8000d786cd73ce63b798a96 866a94bab6	-26.117	4.434
183	10	9	567d0f51bc62247232345e7bd5c5073a4b4 d5002d822d	-26.164	3.946
184	10	9	9fb510fddd6402a513b7317c6506389a1e0 59a4b11bc65	-26.212	4.121
185	10	9	992f9283fb8240a96fc5d5862d296463ce5 debb71d8ccd	-26.259	4.055
186	10	9	2efef5d4dde19fe9026e6db13acb718d287 83c9f8ef52a8	-26.305	4.052
187	10	9	14f80f5c2591e69ce6e755251fbd683512c 2b6376eeedb	-26.352	4.899
188	10	9	d22ffdd5f6a233a8bea58a16e81943e370e 6912d33c3136	-26.398	3.936
189	10	9	123dbffccf13e5b1b781ed982dba92a278e 2573d64eaa9d	-26.444	4.663
190	10	9	2bfec663b8b80160e29f16b506d8b6e8955 261676066b042	-26.490	4.246
191	9	9	11eac5b0b8ca5ad4c2d2744038c59fe6fe4 d07dd6c98b3f1	-26.536	4.192
192	10	9	ad3aaa94f48d92334e31e476fe2f033dffc 37f9042c32697	-26.581	5.236
193	10	9	aeb347c1d1da654e18f519cce85fc9df2c3 23bf65bfebc90	-26.626	4.272
194	10	9	152b11e12881902387f696de45c5a36c92f 8a0ac77638caa7	-25.756	4.200
195	10	9	1d47bac00fecaac330e5c6d93a68ce265e9 4ba9db0b030128	-26.716	3.980
196	10	10	e1be82e1e81af93cca3cd9dd75ec888046b 132b152c78404b	-25.845	4.436

Continue on the next page

An overview of the shotgun hill climbing algorithm results (continued)

n	Old	New	Binary sequence in HEX	db	MF
197	10	10	1e71d8a9e4c75c8904f6dfea5e35495f00d ae91a0a1326ae05	-25.889	3.716
198	10	10	2edba6c2298993d0ff35b502b939c8283fc 5bdf78ab63e79d4	-25.933	3.959
199	10	10	213212bd5e84d6fbe8f059e2e39fbcb6399 b22ae39859b705f	-25.977	4.012
200	10	10	97f408aee9f17082e252ed9dd6354035128 4c780c85cf0cd0d	-26.021	4.016
201	10	10	1b8e271803e8f153e16ed49261efaeeda0d b5e9ac6ea62467f5	-26.064	4.558
202	10	10	28896a25f8804e58cd76e40638bd0786ebc e96957888301b22a	-26.107	3.896
203	10	10	169c3c36a07652906d0deec88865527c4e8 c03e629baefe639e	-26.150	3.765
204	10	10	cf67c809f660c8a7d9bc7aa4763d21c2105 135a2f235294545e	-26.193	4.130
205	10	10	159b65cba243039145c6500c7c65a7fe42d 0077ac87d1be36a54	-26.235	4.511
206	10	10	12765377a7b55d926a14886701cfa80e3b0 5009f57a430e28cf8	-26.277	4.546
207	10	10	3eda512837a306f55c4e618f6282b984c0a 22449efc32625e92f	-26.319	4.124
208	10	10	d8c49d521383658069e764209165efb173a c434b843e15d4756b	-26.361	3.913
209	10	10	92bb7527a734817aab8268f1be66a10f871 3dc86dca35bd6dfe7	-26.403	4.024
210	10	10	2e4b2cdb5d5d06708dddbda17e1097f8294 5cce2040c1e27438b7	-26.444	4.416
211	10	10	3006f3f70992440f19518c5b08c22b12234 35582bfa5f3d26b7c0	-26.486	3.853
212	10	10	2c1c395d9b2bad230839514a11bc85c866a 6389a27fac0fa2107e	-26.527	4.197

Continue on the next page

An overview of the shotgun hill climbing algorithm results (continued)

n	Old	New	Binary sequence in HEX	db	MF
213	10	10	1ecb7a82ac839c1634e9a3c03160de3d009 43a2f549afed919bdc8	-26.568	4.314
214	11	10	4391784839e2ba9e384fe40899ac6c696fd 5eba9949d3feb66914	-26.608	4.004
215	11	10	54eba39307033259c5dd1ae000ba95a041b ef2b9be2d87f2e35ac2	-26.649	4.294
216	11	10	70835039c47a166461a51e2e0bb2a4d756f 29f7f04bfbc9920127d	-26.689	4.160
217	11	10	11c2d59cc49c9469e9d6922094e8dba2617 501ec028d3fc705f3fcd	-26.729	3.999
218	11	10	12e61da78ed3e653f9cb64b6e8bf2145ee8 06877e7e76a8a819a9a1	-26.769	4.162
219	11	10	2b37e41114e882ec5e59c25a9c57a203c0c 6b9699493c357c59ddf7	-26.809	4.174
220	11	10	62a2bc0a38b1605f8321a7c8a13719d34a9 6f3446f6effc21148636	-26.848	4.229
221	11	10	f45bafe7673953bce07d5e74b7c041472ed a23e2cb7d49d32b1260b	-26.888	4.093
222	11	10	d91d6ed119acea81c5f47ec6bd6d3be95a1 9ef9e465a0159070f764	-26.927	4.046
223	10	10	4a70894496d298c01381155df82667e4cb3 21f97347c235e38170ae7	-26.966	3.734
224	11	10	1e2e7c3249469a3537e2fe24612a5c9f520 5f4fa9a9bdec67bee2bb2	-27.005	4.353
225	11	10	1dea3e715a9881e3e0054954159db182909 d36f961a4743e446b34ff1	-27.044	4.609
226	11	10	32cc5e0c945afb4c12f3de9199312138c1d 88669015a8da3fd5474581	-27.082	4.244
227	10	10	22ebf7574cc9779ebc090324b0cc61927b4 257f143313950f857ea553	-27.121	4.251
228	11	10	fa53a40f36c2f6374864b9c2c9ef7b2a284 c5fa79677ee1fea555b141	-27.159	3.988

Continue on the next page

An overview of the shotgun hill climbing algorithm results (continued)

n	Old	New	Binary sequence in HEX	db	MF
229	11	11	a75ce55b5d23ecac9137d372bf947ea0c3a221a1b30befb4b108fcf72	-26.369	3.719
230	11	11	f7332341300147a52cd1491971e815e65f1036b8769a8aaf7159f2c47	-26.407	3.854
231	11	11	2a808dc4d85ca8bd9682006611f9a363c8e9ea6bebd2348d72c51a7c43	-26.444	3.861
232	11	11	5656966e6e18f1dc48803edef7d24bb54eed93e77334ebe02d3aab03d8	-26.482	3.748
233	11	11	17d4bccaaf3086f2ab017b84178db7eec81e279f5cbca7cbe68b5cd8da9	-26.519	3.898
234	11	11	2007f0ac7762cac4e0e43831c4aa1a2240a3dbb58536dead2cd534c61b6	-26.556	4.009
235	11	11	479662251d2130781bcca255d6a87bbc42c407c05258e8eac92838dbb66	-26.594	3.830
236	11	11	240060c71fd710e97cbacb6a9de5b0aeb674353f352edc33609dd2f1337	-26.630	3.901
237	11	11	19cc87e8436ee1b65ea0c8410034dd70a6478e0d6a9d1575c5b89cb537	-26.667	4.368
238	11	11	14cc4b3fad9b12199c1f4e96dfa8f5cd30e7b50817c2f41ab8a362cf7a9a	-26.704	3.985
239	11	11	266ffb94a4f5bea647aa418dc69d151f1a29e6818ec9e5ee6e80f900720e	-26.740	3.560
240	12	11	fe9c900f2c6ade00a1e0b104e12ce6b0fdd2d54466a2146cfa2789ddb059	-26.776	4.179
241	11	11	1c6b10f278e927d5b453595862437ec1f73b713a9b86042153e2ec0054e8	-26.812	4.170
242	11	11	83a8ab66dbf3e2e774631ee7e01f0d8957e20e723dfc9512d2e3069a5eb4	-26.848	4.425
243	11	11	4bb8a96e2929d4ed371fe8b99b623e16350ffe48c167f6f3c22b9021952a8	-26.884	4.251
244	11	11	a0e8e4f0e137dc06decc6ad51bc2b11e12d085843a610d47ffb4b20449b31	-26.920	4.220

Continue on the next page

An overview of the shotgun hill climbing algorithm results (continued)

n	Old	New	Binary sequence in HEX	db	MF
245	11	11	15e76533db7e903cd514700224afd24b2b4 033672fc1528f4308fa91ce0f1b	-26.955	3.877
246	11	11	abfcd0b3f80b03974d8248b8a2b39b7fa5f 8c1ac676f61cd5fb7e729512bc	-26.991	3.756
247	12	11	55779587d0bc0a753acb17dbc71ae24d857 b967ef8529f3dfdd8466cb4c4cc	-27.026	3.966
248	11	11	fee01b4a6b639587aefd2079b02a0fe1f51 a71ac419b55b5cb6666ebe7fa61	-27.061	4.243
249	12	11	e69021ef914b3dc4b720d828f4e78ad391e 8d0671766745d035a2ac6441440	-27.096	4.444
250	12	11	1d82eb11b055fee7570494f67cadeeadebd 60e61c4e48a30f2b0495bd826c9e	-27.131	4.658
251	11	11	185b66591f9adfd4fcb9711a1ed865fd10e b1d31b5da95875bc4222eeef0b04	-27.166	4.101
252	11	11	89e034220ae08d514bdaa363aaa4c2b7ed7 308c45bcdb44de44c3c7023cd85a	-27.200	4.033
253	12	11	1ed2db2821fbfae1870f40e99545e8e8f72 856cccdea1deb2ec37f91da769ac6	-27.235	3.996
254	12	11	2e00e40a057f47b7764b2e91f2e1dc36752 0e74fc9857f5e9298cf5f6b6b1ac7	-27.269	4.468
255	12	11	4e48792994ce3896f2363f70b53c43853aa aaed7c0b528101a17f4018136c933	-27.303	3.993
256	12	11	9b77e41cc0d9278fdd5a54b331946a53564 37b53baa902f780a61805078f2083	-27.337	4.264
257	12	11	797b093ac095d0d53d4ce60de43928b1442 cb679e16ef7b80d5e76eddf8b45c9	-27.371	4.116
258	12	11	f19ccb67644aab3fac44bc02a8b7e62f7f4 ed5f6179f428da5d9b4983dc73c2c	-27.405	3.962
259	12	11	44c930912a770de24230e07dd434aca15a1 9580de8ab79ea8b37f1d90987cc182	-27.438	4.534
260	12	11	a50e2e9f7f7c415d2eb2cfab9be4ea46ab1 980f27c4cce6edc475ae09d216d382	-27.472	4.370

Continue on the next page

An overview of the shotgun hill climbing algorithm results (continued)

n	Old	New	Binary sequence in HEX	db	MF
261	11	11	10867d02c11be5517e4f4f5cbd4135e3f29	-27.505	3.920
			b15ffecae6e0d2d66479d064a678e79		
262	12	11	128b8716cdead0448f3f6e7265ac6435c10	-27.538	4.198
			cefa2987f8c417035121484d40f452f		
263	12	11	c5e44bf69b228002a7b29e90ef252a10727	-27.571	4.430
			0065cca0f0fea6d579de3acdc732ec		
264	12	11	838bacc30044321f263b7f2245bed79543d	-27.604	4.264
			b437f5612e9d956a63389f8177469e7		
265	12	11	1832e784ed916573709c6abcffb07ab5fea	-27.637	3.928
			0b3d5998abc9f0161ea37f7ad965a69		
266	12	11	330bd8bc510f9d0045d9af80815954ee7d9	-27.670	4.073
			0a321066096387ec978dd496872d233c		
267	11	11	df45ddba45d345ced1fb81f37be31a52a00	-27.702	4.327
			14bdf11cacc3a3e3589f5a5e490ca4f		
268	12	11	52b43d5792b8524e4edf6efb9b965597cf2	-28.563	4.549
			53c12f86ee5320c66efa122ff629c730		
269	12	11	152b43d5792b8524e4edf6efb9b965597cf	-27.767	4.388
			253c12f86ee5320c66efa122ff629c730		
270	12	11	21e5eea4f7cf140f85bea242277dde7bcd9	-27.799	4.237
			ca65dc4afae6f990be2a0678b4a966270		
271	12	11	344052dfa92b00930cd10f1c58098a2a1cd	-27.832	3.838
			afa3b9b962b0724c86837e291fc18ae56		
272	12	11	fcdde3a5833a16db6ed41cb0d2cc19c6fae	-27.864	4.393
			cacb3a7ffa0dab51ba1cf6281b9d570fa		
273	12	12	44fd6cbb59dc119fee359596843d96f3db2	-27.140	4.141
			8c5eab59b0e2febc09f04560c206e4ab7		
274	12	12	2084a897ae41a524bbff40ff05d12b96043	-27.171	3.609
			b5385d1fbb747137baa7399e5bc6c74dcd		
275	12	12	4092e0bb2873535a4739c7cf18ade8c273c	-27.203	3.980
			08cced32765fe95a0f45f6d66d564fafd5		
276	12	12	3ad05cc5750b304c44d870be582126af4a6	-27.235	3.563
			7af40533e139a6afbdc6463ce0768206d8		

Continue on the next page

An overview of the shotgun hill climbing algorithm results (continued)

n	Old	New	Binary sequence in HEX	db	MF
277	12	12	183c883b5cb9366c4426e16eb70b50d862eae61914bfa6805a78a29402e20758df57f	-27.266	3.912
278	12	12	3c474578ffdc1943abea11a0613a85b2970d2665b3a7a4d4216113e233f348859d013c	-27.297	4.077
279	12	12	7c815fd557ac5dce82804d1cf4b59b3ca8ce63cc72d2b270145a7220d82501fe8e049	-27.328	3.983
280	12	12	d27923a83fe74ff88a80248e14ad48d99ea5ecf0d1f6d5dc6c18b773a8b167bb8c49ed	-27.360	4.465
281	12	12	15a63b833b92922bda94f25432f9906e7d6cb080b802c9d120101f66ae0d857078d3c63	-27.391	3.823
282	12	12	4a0aa392e5296934d26cf8b8c007b8599bee514e4e7040326316ec4722f5abf06e4fef	-27.421	3.949
283	12	12	131adc329deb49d4484ac1abfb560dd06c6e9bb893abf288981e6107a08775c30a25f18	-27.452	3.844
284	13	12	750721671bd43b577672bdbb85d72e9eb6d8c2f778197470da082cefd9bfd061b61f636	-27.483	3.961
285	12	12	cfefe3c9a98b339b78784d6de752452df674bf76ef115867605ae316a075c142fe2451a	-27.513	4.153
286	12	12	1e5f4df33a080874311aecb106e6bcf8aa4e9fe29d34b36e7e427f23d71a8fbca3f4e2d5	-27.544	4.409
287	12	12	2d6fff0088403555d21c1be4513646065a42c2cde2742f397650ef9c8b432e8e5c0f6b14	-27.574	4.220
288	12	12	d19696cc4945e90993653bcafae44afe6bf3e1c872f1dfbf815e2a8c82f037d74dea9e72	-27.604	4.002

Continue on the next page

An overview of the shotgun hill climbing algorithm results (continued)

n	Old	New	Binary sequence in HEX	db	MF
289	12	12	185bffdf5540ebb9cc935f8bcc4dabf1974 b54c1d5a9cf42b6383636e49c33ef889bc8 287	-27.634	3.803
290	13	12	1954ea11d44ddb89d82999525ce70f41858 f24832d2eb011c1c81e6b0047edf5b71b47 e88	-27.664	3.673
291	13	12	25e47f4fc982622a311acd9dcbbec0f2e9a 9f0cfda9dbe1823df5d7638c6d6194afe08 296	-27.694	4.513
292	13	12	a3b9c7d13899707f33a824ea27782dbb9bc e68925256b14fbb10795a0528e89010c3c7 68c	-27.724	4.072
293	13	12	17a57eb128c330309f040d6583d5e227709 75373e10547bf24a2e93d0cc8f996730a50 14c5	-27.754	3.940
294	13	12	8dc9840017534c6eaf61cf1fbc58f568fa9 4403739476e14d72826e0bd289a4792fcd0 9c1	-27.783	3.870
295	13	12	681faa2e4adff123767204b2af92e18b266 51a67e7c718c0619580c14a2b3f110cb42a 6da5	-27.813	4.341
296	12	12	fb1038796d64e801dc88d702cad89970a41 3091a431977d7be4ba8aa4eb721e3ba1409 acea	-27.842	4.238
297	12	12	6d0fb7212d379086a9e86c2a54fcc87ccfb a7ff6b9d4eca11b8f1e6c13eabafc448a39 a7d7	-27.872	3.976
298	13	12	3aff50e3839b63e273c7ed3402274894e21 3b169fb37555558be5cc425a760b79f690c eba5f	-27.901	4.131
299	13	12	68ef75bbad75d36c63e30b296cd65f93e7e 0141f5bb84b81738c63ee47adab72a0f3b0 2cfb	-27.930	4.139

Continue on the next page

An overview of the shotgun hill climbing algorithm results (continued)

n	Old	New	Binary sequence in HEX	db	MF
300	13	12	b25be8354bc61f73a63b94ea06430063068 27e386dc8e36058b22aabb5a123b284c9fd f9504	-27.959	4.365

B.2 Reached optimal PSL solutions

Table B.2 Reached optimal solutions

n	Sequence in HEX	PSL
10	37a	3
11	712	1
12	b3	2
13	a60	1
14	2a60	2
15	3dba	2
16	a447	2
17	1c0a6	2
18	2650f	2
19	52447	2
20	87b75	2
21	129107	2
22	14f668	3
23	56ce01	3
24	4a223c	3
25	9b501c	2
26	2e7e935	3
27	24bb9f1	3
28	e702a49	2
29	10e2225b	3
30	2a31240f	3
31	2d079910	3

Continue on the next page

Reached optimal solutions (continued)

n	Sequence in HEX	PSL
32	2857d373	3
33	16915cc18	3
34	1a43808dd	3
35	5569e0199	3
36	87885776d	3
37	10c1237a2b	3
38	7caacc212	3
39	29ca6c7c80	3
40	22471e86fa	3
41	7c64d77ade	3
42	4447b874b4	3
43	550e7f99b49	3
44	cb4b8778888	3
45	b6cab731e3f	3
46	16959a2e3003	3
47	69a7e851988	3
48	e6e9bd5bc10f	3
49	103f6eda6ae71	4
50	31dceade9920f	4
51	71c077376adb4	3
52	600dc3cb4cd56	4
53	1671848a940fcb	4
54	2622a797806912	4
55	6006a578ea6933	4
56	61e4b3229420af	4
57	143606103beca35	4
58	215081f5644f2ce	4
59	3b06774134bdf5e	4
60	4df905215263a39	4
61	193c99e12d6010aa	4
62	25695564e679ff83	4
63	707d54b9c99ef690	4
64	d4ef33d372e082be	4

Continue on the next page

Reached optimal solutions (continued)

n	Sequence in HEX	PSL
65	1f75f6c8f84c6b50	4
66	28a59401e57b1c993	4
67	5ba4d417723078421	4
68	d155a49d98c7bf7e1	4
69	18ff3eb05d654b6665	4
70	2b5aae6765e79b600f	4
71	8cea0ff5e92cb9726	4
72	dbcf036102615ab2a	$4^{⧗}$
73	164da9aab5398f1ffe1	$4^{⧗}$
74	8c9c6dab51e57580f	$4^{⧗}$
75	5ff692ba8d62f1e3326	4
76	87ad414fa9fcbb99a6c	4
77	fe00861c0d932958aca	4
78	328b457f0461e4ed7b73	$4^{⧗⧗}$
79	55fae4fdb42732de2ce2	$4^{⧗}$
80	fe00a22a539352e3659e	4
81	dc9df3ff085a6c3aae53	$4^{⧗⧗}$
82	2bf0fceee2499527bc61a	$4^{⧗⧗⧗}$

B.3 Revised Shotgun Hill climbing results

Table B.3 An overview of the revised shotgun hill climbing algorithm results

n	Sequence in HEX	PSL
106	35101a2373a0160d982f6b4e39a	6
107	2408504b2beac46b8d93cc85f86	6
108	727184e79679234058155e880bd	6
109	5db00f58363f65c08452544632b	6
110	2b5085f188c82cbb79e1ae25c1bb	6
111	700f7ceb4b8a926c793caafcdcee	6
112	1c62bf5e0e2bf9bdb9db524d921b	6

Continue on the next page

An overview of the revised shotgun hill climbing algorithm results (continued)

n	Sequence in HEX	PSL
113	10e8e632f9a52d803cd7eac6eddd5	6
114	3fad9a9fa616431ee6a6b8746ba74	6
115	637c6cdec32bd4cbaecaf2ffe1610	6▼
116	a03feff259d626e9c4f46471a5168	6▼
117	1b33da4cc6d5dc7f8a55c9007cb8f0	6▼
118	23c598f4ac7f6afde47b84c05dd592	6▼
119	60835d6bb25f775d6b588d9e361f81	6▼
120	98cc2e429c2f810668dfdf14bab0b2	6▼
121	178ffe7181c3f443365313724aac95a	6▼
122	30d4e9ae516cf0320ad003177377485	6▼
123	369ec917afe507e53bdc97151138738	6▼
124	f15ce151edfd7f0ca9eb4496d833233	6▼
125	1b8730333bcdf414d92203c581a554a5	6▼
126	3b9275a7ba7661bb8dbf8e078ad41257	6▼
127	2933b32d40937c4b6f08e03a851c2c2a	6▼
128	84528942da6f07e733404ee8ba70c3ae	6▼
129	1f80f99bf3cc5c3d6f1aacd4209aa925b	6▼
130	2678ae07e71929fb587022ed6bfdb576d	6▼
131	3cbf4b091ea86cea277167ac6304c812	6▼
132	410028af0ea52e93f029f908ce74d8c99	6▼
133	10c27978f1888d4fb0a97c9326ecafe97f	6▼
134	3f01b89e464dccaabce38e920492b56810	6▼▼
135	550c944868887c4b7b8709d8263de6c81a	7▼
136	dc789e3aa4f65db16085033ab4b40aee42	7▼
137	1bdfe2817aaa3b39d39daf366d86bc0f49 2	7▼
138	1e618e9ba6c707dc94f05ad723357b2bff d	7▼
139	2bd70f3cde89ad5316439120fe3b9b480b 5	7▼
140	79f8036d08785fbef98ba3b2eb54652eb3 3	7▼

Continue on the next page

An overview of the revised shotgun hill climbing algorithm results (continued)

n	Sequence in HEX	PSL
141	fdf5f77808f6cf055b0dd9295c878ad32e2	7▼
142	343638ce5ed915e8abcc9a0beef812814894	7▼
143	554e194c63ca5a65f47de2fd999fc0227ebd	7▼
144	e757f83fd667a6c479d5296908879f6c8d2e	7▼
145	1051d14d00c893e49498fdba0570862f539ca	7▼
146	1c8f3f584efe71220e0da5d4d58d10ed11ec1	7▼
147	2072a669eade89e6058251c9cc2628a5f5602	7▼
148	136cbb11363a7078d55f1dc696f217b588520	7▼
149	152214204e428bf4553661919fe41c0db69e18	7▼
150	4fc361d2f104c9510a54c53e9afa6123467f6	7▼
151	3b93bb695ed592557b82497047438f87c4318 0	7▼
152	62913b08d6326c46c082e52e3feb0e0b6d7505	7▼
153	152b80f95df5a4a0e1a2e30cbf68cf76e9ccdeb	7▼
154	183f0383fe80cccbf38ac495965dae7bd79a695	7▼
155	1431f0be9440e92cdd1c4d5680659df9377adca	7▼
156	bbbd712a19673174fdbc9ad3e78d06f40bb1a84	7▼

Continue on the next page

An overview of the revised shotgun hill climbing algorithm results (continued)

n	Sequence in HEX	PSL
157	1444313cfc12546b26e36eb70568dc11b7 06bcf5	7▼
158	18785b52d7074935b31708ef988f769911 040aa7	7▼
159	41c5d5f8d8012c40f00d6ba24d35a539cd 8a573a	7▼
160	7277c5d1140ae6e5638c47ab40937830f2 1b684b	7▼
161	1c9d7e8ec413f2eddacc7be1a45a4318ba 3ab2b46	7▼
162	36280d42653b385e990c70aec3d64845dd 59413e	7▼
163	ff4a2a50fadfb069cb64a79bb8eafdf55e 660cc4	7▼
164	8cbd592237b8e9d5dd7fddb148e13a7c0e f03696c	8
165	bdfe78f96cd0a73e41fa8764667b9e82d1 54d6544	8
166	12c242012b761f803271ab9649f67432ee 288d398a	8
167	349aab4a5752c6459e4cd43f18708fd980 18044fc3	8
168	a18f9ca18bdb8eaf44a84db7f3f92ddd36 0ec23bc4	8
169	f77f9c30338bec86cb76455ec4af4d4394 769e17de	8▼
170	1647c513e17c8b5ac12f169cf4008e77de aeedd71d9	8▼
171	455bce3cd34aa5199a53b3f9900ed684d8 11607828e	8▼
172	8a862f714aa517b5e1d4c9e784b66d0c07 eccfd9f61	8▼

Continue on the next page

An overview of the revised shotgun hill climbing algorithm results (continued)

n	Sequence in HEX	PSL
173	9087c81b16785b95b1f63942ab8829d1da e83048267	8▼
174	3a595abedb5fb13e998250683feaa608f1 b10f721e8c	8▼
175	75f9a9db5111b640009d36bc18a71887a8 d4f60f5079	8▼
176	f8b81eb83c80faa526c53d6c43bbb18d34 c2b7df7bb3	8▼
177	ceca7a7c3d3d4ed8893081464daa5d50cf 40905fe630	8▼
178	240603787909825762b567fe0a338e0aeb 85db46e98a4	8▼
179	d27d4a3d8ce26560a137f967fd5b2a22fe 7ea4e9cb1e	8▼
180	45f880bba360f4fe67321649f77be67971 e729d54b4a5	8▼
181	16e90d659806f9ad47ab399481f7975581 4e0cc9805074	8▼
182	2ad575a76ca6eb9f36b207790cec2047db f70dd1f07095	8▼
183	3080a6b5d518e2437cbe03e83276091f3d 9ad5bb717275	8▼
184	bd09e4073d719c5290dc81d3edc090b050 3345a2aaddb7	8▼
185	1b0245da96f15f3faaf6f0e71c5a6c6e2e 7ba4fa2190aff	8▼
186	d6c3079d747a0496d4eab7337a91236c73 cefba0eff4f4	8▼
187	34999dc9c93025871f7aaceb517d0e451c 07b504a75da01	8▼
188	36766a797988a55100a42a91e73c43f005 b76d60705f364	8▼

Continue on the next page

An overview of the revised shotgun hill climbing algorithm results (continued)

n	Sequence in HEX	PSL
189	880723487ff2acbc3e65d1eba13327b9a0 5965bd52d14e7	8▼
190	21e50af105ba1d87a44214221d935bba27 35951f776101cf	8▼
191	6122466d46065abb2e2595ed350f45d4a7 f173881f4c33ef	8▼
192	1fbfab7bc285711fb852eb5f00b2ba9c36 98e27cd26a66c9	8▼
193	11e5e2e1ea52cd9c13f6ec031979a99549 b90fb8c2600a288	8▼
194	35d745068d86b74ca0a6d8c73a39676ea7 7bd2b4bc0fc0267	8▼
195	1c841bd699c259b0d801b20e4fd8bebe1c 6567ae3abd08a95	8▼
196	b6a64ce8063c6116f91dd3cfc332f8aac5 f7bdc8a0bad6d2a	9▼
197	2b7ceef5fba16ec29257b30a65a26ac34f 1841ddc7c0e0de7	9▼
198	2da669214cb962a811544e5d3d37a000f8 c0c60dced1ed0bce	9▼
199	1144b275da9c8adb8fffce37c87ba0d2c3 c6bda983f4dc032b	9▼
200	66c30c122f4ee5d8b01ab9155a1ca5afed 0d37d4df0775bd84	9▼
201	82a1c892ca09589a5f1ba194c682ef0f71 d182378a64895ff4	9▼
202	1d045e3d7d3e006c938fb456d5f2a4bf5e 4dcce9c41ca663186	9▼
203	6413fc8964522104171ca948e5d4c4e1cf ade1a82d03b3e640d	9▼
204	6730c61d894ad6db47d7db1707d109a8fd 7e9912cfee2df887d	9▼

Continue on the next page

An overview of the revised shotgun hill climbing algorithm results (continued)

n	Sequence in HEX	PSL
205	d24ff6dfd7766450d28c6f1d08aa13c6f5 060b93ef182d5e847	9▼
206	7372ccbe4d517dc500e9ed586a99c9fc60 a442016a06fd0c961	9▼
207	5b92dad3371cc960e08e1993a80ac0a9f5 73c2708165ba02bf5f	9▼
208	d47fffd42e8257a630ef1673359f05eb26 ce173462e0ecb498d2	9▼
209	1bf64d73ea8531230afd6c614fceee5aad 2714cd7c1674125f01e	9▼
210	31da42975a5a3c741f6506fc77598874bf 77e37f2ae29fb1304dd	9▼
211	252e50a7cd40fd82e13aae3096361608b2 3030076fbbd84ca636e	9▼
212	87d63ff093d2c932221b74ae6e9443ac63 33b42e0b890a5754141	9▼
213	54614010e30c87b5366b6baa6400fc7e8b 57067a894e9b3f898e4	9▼
214	e7d4a6d69fda9cf9843db94242a88c5cd7 77cd24165c2f913e0fc	9▼
215	301c7898c56aa56687800ffbf3e65a5787 6867c9426eb3dd5d46d3	9▼
216	d332ccdcab19af1972f93007baf8af8057 c3af4b59e4d040624b52	9▼
217	a87867118f48a6922d161093f015d7f8dd b57c80cb5aedd1b0b177	9▼
218	4c91d36554864c73c5ae223a17dd60ec62 96849685d7fb81f3f881	9▼
219	536a2df324baa32c8488880d9ae152f5bd 0b808ebcf131fc0c293c7	9▼
220	bc39257be78b79101abf2c3edb9b3c01e4 157240d46a6319c5789d2	9▼

Continue on the next page

An overview of the revised shotgun hill climbing algorithm results (continued)

n	Sequence in HEX	PSL
221	fadda9f6d109fcc882a91bab8478e6ebc5 713826d19fd06b485a061	9▼
222	1e1a28caa7a070d16e6300965eba9752b3 4e37e66d02139025cfc84f	9▼
223	49e28e14ca6daa3c6fc973368464a08c55 94bf408129cfa607b303d1	9▼
224	b5a435ab97a31e722120bbf812cd251cc7 032281cc0aa29f07f9b66e	9▼
225	1f61bc4168c021782f9e50c6b52a1c546a da0864fd9313b32a9bfae98	9▼
226	3bbbf1c19f6ec551ff0ab982ab334dae01 29a63cc6b58b61e968d0d80	9▼
227	378320814f8021439fe15e5a12add18b76 0cd0788aba8ed3630926333	9▼
228	ac5471683c569456b21141ec539fa32e00 78998f3800d377665b36fa	9▼
229	a18ea64e0c7d887c6fb51278b686a8b401 66199ef8050906a7c1516b3	9▼▼
230	25474a4ba6e7c3434d1c724ef643ea3181 2c8fa1bbfc877bf6ee5488f7	9▼▼
231	180e1d36289672c3086a1511d58dfeb4a7 f13ca44b44fec5d024664dde	9▼▼
232	886bf85a5bf40b7b2fab51ee8712e2bba7 5b358384435d3ccc99dbf7e6	9▼▼
233	5dda7518a3629e66f6ec3823f6cc6c373b 4bac795efacf416fe1a1ab00	9▼▼
234	f8e141d03a5bb1d91ce20721cdb6207f56 b699d33bf575955694dfa930	10▼
235	3feaa21651ef22c8cb05ab35df33b138a0 8c83e1a1ed24685592035e152	10▼
236	5453ea9ffc1e60c3285de3d07b64a1bcc0 95366d4c437dfccd5d8f4fcea	10▼

Continue on the next page

An overview of the revised shotgun hill climbing algorithm results (continued)

n	Sequence in HEX	PSL
237	165567767124fabcb4d08f0da7140abd81 f42e5c9a831dda76fffd894c71	10▼
238	c81ea65bf4b9df2e7f7066454c2d3c8e6a 2841e27963c8229db40a0afd8	10▼
239	6a66b95e25a3cb20e16c7b36b1b22e5988 21242ffc69eeaed03bf9f9d753	10▼
240	dcd3bec7a1856d4ea4febb5c0dcc52e119 ffaa69d4c86df1470530793374	10▼
241	15cabbcb3c965d13d1baf6581833a05593 c6ff73c18dfca9e96272a467f29	10▼
242	2cf51c98f2793804326afb59471b2243a9 12fa50b7abce08ef22607d03941	10▼
243	66346d9c9f2d3393fdeaa0075e7f573ef2 7a1b4b8630b4a322df02f9ba47d	10▼
244	cde4bae1750d2d31e5e3b193df44580f92 245b262ffaaf6e42c6bde7b9532	10▼
245	1af4e7a8ed850811188970ae2af8180736 3afb0113d0f9166b49916df928d6	10▼
246	30b36a460dbd8ab690c173b8d8ca8c0351 cb3a170bba020a9417843dd76dd3	10▼
247	5de6adef6aa7775d3812b0cd7831689b5e 39682e61899e9ba3f039b00e27b4	10▼
248	f41f437cb07cf0a0aadf0c67b3f7fe114c c66b766ccd153185293943089549	10▼
249	10242f665effd3eb4875a1fab42f9d4515 fc9e251dae3c607319a69e49366ef	10▼
250	2007616f89095843f3ced5634bf501cfff 55adb4589658662e8ba374f65c676	10▼
251	275419d5069976e3bdde14b3329284641e 6164276b8012963f4d383e161fcba	10▼
252	1b55a5dadcac2ee8c3ef41026edcc98eab f592878208e314f6349886407e13d	10▼

Continue on the next page

An overview of the revised shotgun hill climbing algorithm results (continued)

n	Sequence in HEX	PSL
253	16fa06a49b5776c2a804a3f64b59e4fd20 3a358e8a77d8f79f159d7c34654e60	10▼
254	20d5c99925b7a51f543e49ff428d5d4e54 8a26e1280a1a2d9fc5cc33018c70ce	10▼
255	10008133c4e8b9aa47e1546b8b75a0a4fb cc1d2c7925637235e4866f23d20cf2	10▼
256	6e6053b51d9f80a561e97e2cc13cae1d56 38728f2013377e867fbbee26bada65	10▼
257	1a24b6e6c465cf993425fe01cb10c2ac88 2285c51cea5697d378bc40305c6e753	10▼
258	dafc4a13dbc909c653b76970b24085986f f0fd93e73d6bcd8bba9aae855ce8af	10▼
259	23c7a45f27e3ff8fb66d31f630620d9f6f 959318ea2754cff5256657508bdad2c	10▼
260	94db24992764caf16520a31303c3d0a967 2e74e01e8012d787381aaaeae319def	10▼
261	28d24a7097956e9f7a63b183c0d97211ee 4f99b9f94a3de360ff75f75082fb55d	10▼
262	2547150b862f86ac277033a8d7de1cfd81 8cd1012db7104817cbf15c29924695c9	10▼
263	58333ccc921a4318cbddf299f4a0d055a3 3a13554056c856a9380b4ff0e1c60d3f	10▼
264	7a4dd00f8bbafc5095a2f5f00da7131ba7 d6f7ac4ce20662e388a6b0c21273204b	10▼
265	154ab3ecf9568391efd8918b059f988d67 a21805a46107cb6b89bd30f4c47405c51	10▼
266	3c690152ba0daf7d5b4f7a3ee3c88ab33f 6bb8252dc786c8ccd668169c4bbc4cfc2	10▼
267	15e1810bfa1308e523b851c7078b2be464 f66df69c7492775594b91644a16e77aff	10▼
268	32c38e387faae3e8b74eb7d4675bfa49f5 00cac6c56b4de44a8b7f9d8372666090b	10▼

Continue on the next page

An overview of the revised shotgun hill climbing algorithm results (continued)

n	Sequence in HEX	PSL
269	e6fbaa465ee2a294646b484fbf7d498512 837f32dc48de2872f0781741941892680	10▼
270	ca6d5d2e2898349ca6f36814244ad6e204 8fc50210d8a0fe07fcd5d7f135261c718	10▼
271	4fd6ffb4ef673619e25b08bdb8157332e6 15587d1c72ee2d9302c5feb706b0acdab4	10▼
272	eb8a2f2227f60eca7d47a60d44193beef1 4b2502f3b5a198f69d3ed7dfb4eca72b41	11
273	1b007f99648f37cf3f43ffdb61260d2d33 b65231ad1cbc3353a1ec6e4bc5555d5ab9 5	11
273	1d92f5d3696863c9fa0972f85e9023bf72 63e0d0472f3a817d42462388332a9db3ba d	11▼
274	2a377a8cd8e836fc187135c97cb4f69fad ef3a4367b96014b1b9a79bb40d1120baa7 5	11▼
275	5991082785400a4fec7053b34aeba361d9 542b51c7533d37b28524c29f747f285b8c	11▼
276	af8eb78a4018df61ad9e2d5c980dd38ea4 dbd3cc1d37126245796adbfad9dccfe27	11▼
277	4a97467cb36e66d3c4062908017d0aa39c 6a04ad0f2f27b6c10b1dbaec226a396dc1 d	11▼
278	96655611a994569ea5924430f8fbaace17 8f1df22f07a48c180bef02336e65223642 b	11▼
279	7425c1ec9da091b4d0ee98297cf8a600cb b43c455e0031c4f15c642251892bdbcbd5 d7	11▼
280	55bef3e1c6a79c1a03aad609724c2da00b ba2ad6484112fe95db18d81f99948c6f0b 32	11▼

Continue on the next page

An overview of the revised shotgun hill climbing algorithm results (continued)

n	Sequence in HEX	PSL
281	1424e102f4fde1aa05941514283b49ba3e 1786dae904facc4c6db8ac6632ef12acc8 9ce	11▼
282	3e88983536cd657fd8069a3360e796e35a dc35cb8ab5c6f0ebeeefee25ad9daecc06 81f	11▼
283	5f08ad54acf01756103661f0e35a1c815c d9465bf0909bdcbca2081b5ce79bbb96e7 4d	11▼
284	d1195f2440f108379fb13357dd894f83ff 89e0e313269f6b5f48f675acb1218d5769 aeb	11▼
285	1fd2b9a522fa0ba0363cefa32874704a1a f558b374eb3eadff9593c7925bbc98e4f6 62d7	11▼
286	12410ee79818cd60c7230ad510aadec039 2e476e0f0a036f167bf2be2cab02c0d6d4 4d81	11▼
287	63dac2f781e251694b5e9978aa03ca24f2 a20ad51bbba930e99d93590608f330099d e606	11▼
288	69a4d15a39cd274d62e3f41c235b3b280f 0336af0833a646b21eb0a04085c40b5fab 1aae	11▼
289	16d9909ffc2421af02de219e1d86e042cd bfaff9be97237531d2e1ab96739b5eab8b 166aa	11▼
290	15c7ff0aef22f9dbdf7394c8094b13871a c35a9bcd81a472251e5024efd3605951fa 0d157	11▼
291	a1c202c94a731846da997686016197dbcd 6a6ca7cc264437646ac0c0fcbd11ff5ae0 7555	11▼

Continue on the next page

An overview of the revised shotgun hill climbing algorithm results (continued)

n	Sequence in HEX	PSL
292	6e7871e089cc8db9274cf3a22f22d3d452 3d272db2952ab2ac27188b6fbf47faef01 bbdf6	11▼
293	4e6aa8af6a0ead457af0ad0ca55efe940e 310e4e6f21b73cf0006124c90360db0b98 7db6c	11▼
294	1a5b6d16eca315188a6c5271c7a7ab9eb3 a5ee0efdaabf07b9578110e7fcfe06ccd0 a47ecc	11▼
295	4707bfcc051613d674df4982da568161be 90f8cb12bf339535d1a7488f0468c03112 ae1157	11▼
296	b71a2ac9b154ef459223e3b03cd2394c7e 3c15f1ac6a2272deea2c235a20fb4bdbdf 3b7fb4	11▼
297	2ff21574c741f68aa9d0872b97acc757c6 fdb374791dee45c4a41c274d6c6df59200 62de92	11▼
298	32c7b45b87be1884edfec5712d7e3efcef 2e825460a6d5dc1b9d4335581e4e33b454 d9126b4	11▼
299	587814353b51d6b1d029f7fe73bd9c88ad 984a394357ee609923a3ec1923e0bb4047 492ea83	11▼
300	5b2a550cad8a468cac4ac82be6ad333849 c865361c6d800818ef9387d6513e8cb81d 0f23ff6	11▼

B.3.1 Revised Shotgun Hill climbing results for longer binary sequences

Table B.4 An overview of the revised shotgun hill climbing algorithm results for longer binary sequences

n	Sequence in HEX	PSL	▼
426	3075e0e3e3c1581d2af808dfee48904 226a942d671d897292c4613c5b19a5d d22a6799309414418db4ba724a9fd8f fd1dd109b71493	14	▼3
3000	9c9d1dd018fecf19c744616ad4b166 50e04945bf3f38486f3e52499f8687b d6a090f4b79735ec64f9987f6ac4985 ba941983ecdb9d1d0fd861dfdc7ed4a dd34ac12f08b559aa8c22cf70b4724a d819dbb4ddf4678582db3601786cc56 7f8d290b90cbf46e2939152989bba06 5e1644ec8b1d995e9d8d68221ff5166 66bff43b0a993eaa9ba440f0b79f00e 083acd93b1b64ee5acc52cb1a3bc77e 7c01a14a8a7d8003c62fc5778be6a05 df09b9fc03b70dc2df6850a61ed7045 398c52aa1b5baf036848553d7dd27f8 cb72ed847c6796f7216a975dc497149 ef6eab576508ac77dc3c8837d54d952 1d151694dea17e2bb4969a2c4461616 fafaacb172e35685b3bd63152287a79 e329c65b01a41030bf595ec7ef87188 b37a4d3552e73fadefcdf57b05cc618 904a2fdfd52ff7e8a8c1ea9fdf9db08 957495f01fd6ca7ff219ae3c4624100 d4eee30cc0db5aa8e9f548c31b10593 f138b2c7d22c3f7c16279b7b2f65de7 d17494944967d341c6c0c4e70863b00 201984a	43	▼8

B.4 New Classes of Binary Sequences with High (RECORD) Merit Factor

Table B.5 A list of binary sequences with record merit factor values and lengths between 172 and 237

n	Class	Record sequence in HEX	Old MF	New MF
172	$\Omega_{173} \circ \eta_4$	fe03184fe780309206b6663e571d355a6ac59356a	8.8363	9.052631578947368
178	$\Omega_{179} \circ \eta_4$	2c7bd3a7034ccbfe886e8a084688550ccf2b613d16c	8.2125	9.29149560117302
180	$\Omega_{179} \circ \eta_1$	b1ef4e9c0d332ffa21ba28211a2154333cad84f45b0	8.5353	8.553326293558607
182	$\Omega_{183} \circ \eta_4$	3cfe712191dc7d1c57c81ec5a8d7edbd6ddb9a3b654d2	8.2194	8.928301886792452
184	$\Omega_{183} \circ \eta_2$	f3f9c4864771f4715f207b16a35fb6f5b76e68ed9534a	8.2980	8.636734693877552
186	$\Omega_{185} \circ \eta_1$	233da5ef19ed00149bc0c4644d4b9c1550e992e878b375	8.3606	8.627431421446383
190	$\Omega_{187} \circ \eta_0 \circ \eta_0 \circ \eta_4$	190c8d2692191f17dba56aff75407e11d7b5b9a3863c8cb9	8.5021	9.195109526235354
192	$\Omega_{195} \circ \eta_4 \circ \eta_4 \circ \eta_4$	190c8d2692191f17dba56aff75407e11d7b5b9a3863c8cb9	8.0000	8.930232558139535
193	$\Omega_{195} \circ \eta_4 \circ \eta_4$	32191a4d24323e2fb74ad5feea80fc23af6b73470c791973	8.6868	9.11179060665362
193	$\Omega_{195} \circ \eta_0 \circ \eta_0 \circ \eta_1 \circ \eta_1$	c864693490c8f8bedd2b57fbaa03f08ebdadcd1c31e465cf	8.6868	9.238343253968255
194	$\Omega_{195} \circ \eta_0 \circ \eta_0 \circ \eta_1 \circ \eta_1 \circ \eta_1$	190c8d2692191f17dba56aff75407e11d7b5b9a3863c8cb9f	8.0522	8.644005512172715
196	$\Omega_{199} \circ \eta_4 \circ \eta_4 \circ \eta_4$	937e64c9f4bc13e78367a16729653ad0f58ce65738aaa	8.0910	8.521739130434783
198	$\Omega_{199} \circ \eta_4$	24df99327d2f04f9e0d9e859ca594eb43d633995ce2aaa	8.3662	8.786194531600179
200	$\Omega_{199} \circ \eta_2$	937e64c9f4bc13e78367a16729653ad0f58ce65738aaaa	8.2919	8.756567425569177
202	$\Omega_{199} \circ \eta_2 \circ \eta_1 \circ \eta_6$	126fcc993e97827cf06cf42ce52ca75a1eb19ccae715555	8.0291	8.568668626627467
204	$\Omega_{205} \circ \eta_4$	3c7877d72fc246a45d9aaedfb63a8eff98847e474af20a25a4b	8.1858	8.276849642004773
206	$\Omega_{207} \circ \eta_4$	2492010c9e4ed276a103f76a13a07752ba0763c4e58cbaa38e2	7.7921	8.436580516898609
208	$\Omega_{211} \circ \eta_0 \circ \eta_4$	38e383be228cf9981fc150c0fafad4d014a9599acdf76bb5b6db	7.9529	8.96849087893864
209	$\Omega_{211} \circ \eta_4 \circ \eta_4$	38e383be228cf9981fc150c0fafad4d014a9599acdf76bb5b6db	8.6394	8.849473257698541
209	$\Omega_{211} \circ \eta_0$	71c7077c4519f3303f82a181f5f5a9a02952b3359beed76b6db6	8.6394	9.254449152542373
210	$\Omega_{211} \circ \eta_4$	71c7077c4519f3303f82a181f5f5a9a02952b3359beed76b6db6	7.9862	9.153175591531756
212	$\Omega_{213} \circ \eta_4$	2a2fbaa406cf5693c8d89b658f12d8639d8dcb1e02ce547fbaf7f	7.8082	9.262984336356142
214	$\Omega_{213} \circ \eta_1$	a8beea901b3d5a4f23626d963c4b618e76372c780b3951feebdfd	8.1808	9.17755511022044
216	$\Omega_{215} \circ \eta_1$	7c361f6410df47fc0c506cc111bbacc6becad56f5cbae75a72d6b	7.8598	8.589101620029455
218	$\Omega_{217} \circ \eta_1$	1a403f08432daddf13836660a67d66635b1288f8f345d2b547955	7.4888	8.328776726253066
220	$\Omega_{221} \circ \eta_4$	3f06f260dece7d91c596a6d0009d550e7e184918a6ce8d672e52b5	7.2848	8.479327259985984
222	$\Omega_{223} \circ \eta_4$	1f1f0fa3be027fcc798db8f201889aa3491c996cd562a51214b5b5a	7.6742	8.71666077113548
224	$\Omega_{223} \circ \eta_2$	7c7c3e8ef809ff31e636e3c806226a8d247265b3558a944852d6d6a	7.3962	8.521739130434783
226	$\Omega_{225} \circ \eta_1$	e060603dfef3807b8dce68f58e36d82de6c8dba55b2ea8b565656d5	7.1555	8.487205051512131
227	$\mathbb{B}_n^{15,11,7}$	7fff001fc3cdfb67e939a58cc0d8658d4cd879b1ea63a8cb4a9552aaa	7.5578	8.069057312871907
228	$\mathbb{B}_{228}^{17}$	ffff9c31639d28ccf5ab85cba46d3c6e10de901f4cc83d927b2d95555	7.2888	8.18903591682
229	$\mathbb{B}_{228} \circ \eta_5$	1ffff9c31639d28ccf5ab85cba46d3c6e10de901f4cc83d927b2d95555	7.5476	8.55202217873
230	$\mathbb{B}_{228} \circ \eta_1 \circ \eta_5$	3ffff3862c73a5199eb570b9748da78dc21bd203e99907b24f65b2aaab	7.1739	8.16610064835
231	$\mathbb{B}_{231}^{16}$	7fffb612c7d8bc368ed13b8379234c371a35bb10ede34bd8a4f163aaaa	7.4381	8.18671371586
232	$\mathbb{B}_{233} \circ \eta_4$	fffad0296b1ca397ca63d8cedc5b991edc4c9d260d7920db0782bc1555	7.1727	8.05748502994
233	$\mathbb{B}_{233}^{13}$	1fff5a052d639472f94c7b19db8b7323db8993a4c1af241b60f05782aaa	7.3522	8.22560606061
234	$\mathbb{B}_{233} \circ \eta_5$	3fff5a052d639472f94c7b19db8b7323db8993a4c1af241b60f05782aaa	7.1651	8.27379873073
235	$\mathbb{B}_{233}^{15} \circ \eta_5 \circ \eta_5$	7fff5a052d639472f94c7b19db8b7323db8993a4c1af241b60f05782aaa	7.3496	8.44677271337
236	$\mathbb{B}_{233} \circ \eta_1 \circ \eta_5 \circ \eta_5$	fffeb40a5ac728e5f298f633b716e647b7132749835e4836c1e0af05555	7.5797	8.37282020445
237	$\mathbb{B}_{239} \circ \eta_0$	1fff3863ff634e14a46fe933d8c162c27ac9d338546e0fa4f27552693555	7.8230	8.65203327172

Table B.6 A list of binary sequences with record merit factor values and lengths between 238 and 278

n	Class	Record sequence in HEX	Old MF	New MF
238	$\mathbb{B}_{239}^{14} \circ \eta_4$	3fff3863ff634e14a46fe933d8c162c27ac9d338546e0fa4f2755 2693555	7.7573	8.34226804124
239	$\mathbb{B}_{239}^{19}$	7ffff0c39e1f03f01ef8967c933666e66331ca61dae952b52969b 4d2aaaa	7.6962	8.19056495555
240	$\mathbb{B}_{239}^{14} \circ \eta_2$	fffcff800f3c398721e6e43326f40dcaf46332e465a36992d34aa 954d554	7.2948	8.24742268041
241	$\mathbb{B}_{241}$	1ffffc4235e31e3c1ee6079bc2973b0b13782d196a645ad25b25f 22ed5555	8.0668	8.26893507973
242	$\mathbb{B}_{241} \circ \eta_2$	3ffff8846bc63c783dcc0f37852e761626f05a32d4c8b5a4b64be 45daaaaa	7.1893	8.14973559699
243	$\mathbb{B}_{243}^{18}$	7ffff8c0e7381f840fe3960f3a4c66ce64c7b2d61b6ad45a95b26 d4daaaaa	7.2488	8.46216680997
244	$\mathbb{B}_{243} \circ \eta_1$	fffff181ce703f081fc72c1e7498cd9cc98f65ac36d5a8b52b64d a9b55555	7.1730	8.51974813967
245	$\mathbb{B}_{243} \circ \eta_1 \circ \eta_5$	1fffff181ce703f081fc72c1e7498cd9cc98f65ac36d5a8b52b64 da9b55555	7.3237	8.5799028016
246	$\mathbb{B}_{243} \circ \eta_1 \circ \eta_1 \circ \eta_5$	3ffffe3039ce07e103f8e583ce9319b39931ecb586dab516a56c9 b536aaaab	7.1650	8.33324153126
247	$\mathbb{B}_{243} \circ \eta_1 \circ \eta_1 \circ \eta_2 \circ \eta_5$	7ffffc60739c0fc207f1cb079d2633673263d96b0db56a2d4ad93 6a6d55556	7.109	8.22889128675
248	$\mathbb{B}_{243} \circ \eta_1 \circ \eta_1 \circ \eta_2 \circ \eta_5 \circ \eta_6$	7ffffc60739c0fc207f1cb079d2633673263d96b0db56a2d4ad93 6a6d55556	-	8.01668404588
249	$\mathbb{B}_{249}^{16}$	1ffffc212fe40e94e19e33d2972665b1b1e663783d3259a4f84ae 543a2d5555	8.1323	8.20119047619
250	$\mathbb{B}_{249} \circ \eta_2$	3ffff8425fc81d29c33c67a52e4ccb6363ccc6f07a64b349f095c a8745aaaaa	7.1988	8.19564647259
251	$\mathbb{B}_{249} \circ \eta_2 \circ \eta_1$	7ffff084bf903a538678cf4a5c9996c6c7998de0f4c96693e12b9 50e8b55555	7.5632	8.2354248366
252	$\mathbb{B}_{253} \circ \eta_4$	ffffe03198f80ce61e0f2cf07a25ce2176c877a52cf2d6966cd5a d99356aaaa	7.2394	8.49438202247
253	$\mathbb{B}_{253}^{19}$	1ffffc06331f019cc3c1e59e0f44b9c42ed90ef4a59e5ad2cd9ab 5b326ad5555	7.3036	8.95481253497
254	$\mathbb{B}_{253} \circ \eta_2$	3ffff80c663e03398783cb3c1e8973885db21de94b3cb5a59b356 b664d5aaaaa	7.0325	8.51808819646
255	$\mathbb{B}_{253} \circ \eta_2 \circ \eta_2$	7ffff018cc7c06730f0796783d12e710bb643bd296796b4b366ad 6cc9ab55554	7.2849	8.22892938497
256	$\mathbb{B}_{259} \circ \eta_0 \circ \eta_4$	fff65cbd0a16c7841bd24913277f2064c6572a27311c70b945a4e 17d0bc862aa	7.1483	8.12698412698
257	$\mathbb{B}_{259} \circ \eta_0$	1ffffc380df2781e7233e42db41b66c64e6c671af1c2e532365a9 635ca92d5555	7.3847	8.32270665323
258	$\mathbb{B}_{259} \circ \eta_4$	3ffffc380df2781e7233e42db41b66c64e6c671af1c2e532365a9 635ca92d5555	7.0738	8.25241755517
259	$\mathbb{B}_{259}^{19}$	7ffff8701be4f03ce467c85b6836cd8c9cd8ce35e385ca646cb52 c6b9525aaaaa	8.0918	8.18659995118
260	$\mathbb{B}_{259}^{16} \circ \eta_1$	ffff3e0648fcf2598f931e43a9538a717b6093f812e5b39499e34 d48e6a53555	-	8.26405867971
261	$\mathbb{B}_{261}$	1fffff21274a5ec18d9e601cf948f139c2d93b48f94daa659c9ac 5e0f63a35555	-	8.04452054795
262	$\mathbb{B}_{263}^{16} \circ \eta_4$	3ffff3d98cc67b60b077887e1c1eed486c68fc45ada568976b0a7 166cc99d3555	-	8.08146927243
263	$\mathbb{B}_{263}^{16}$	7fffe7b3198cf6c160ef10fc383dda90d8d1f88b5b4ad12ed614e 2cd9933a6aaa	7.2006	8.22852724245
264	$\mathbb{B}_{265} \circ \eta_4$	ffffec3731980ffc25863ed306f12b6b503e3f12e530eb65874aa d5993234eaaa	-	8.17260787993
265	$\mathbb{B}_{267} \circ \eta_0$	1ffffe3943ca79588ed4f2ccc37e13e24dce253a572ccc34fc489 f960d2f925555	7.0963	8.71710526316
266	$\mathbb{B}_{265} \circ \eta_1$	3ffffb0dcc6603ff09618fb4c1bc4adad40f8fc4b94c3ad961d2a b5664c8d3aaaa	-	8.10492554410
267	$\mathbb{B}_{267}^{20}$	7ffffc728794f2b11da9e59986fc27c49b9c4a74ae599869f8913 f2c1a5f24aaaa	7.0765	8.07715839565
268	$\mathbb{B}_{267} \circ \eta_2 \circ \eta_2 \circ \eta_3$	fffff1ca1e53cac476a796661bf09f126e7129d2b96661a7e244f cb0697c92aaaa	7.0004	8.01965163019
269	$\mathbb{B}_{269}^{22,11}$	1fffff800ff0699b6c1f21f0db3632786c6c69632731cb5a35ac7 1986b54aa955555	7.3092	7.4414849856
270	$\mathbb{B}_{269} \circ \eta_1$	3fffff001fe0d336d83e43e1b66c64f0d8d8d2c64e6396b46b58e 330d6a9552aaaab	7.0056	7.27399720615
271	$\mathbb{B}_{271}^{21}$	7ffffe89502ef15b2327cc786c3c6784ad0fc5a64b4e5a4ca7373 812ef501deaaaaa	7.5386	7.69015706806
272	$\mathbb{B}_{271} \circ \eta_1$	fffffd12a05de2b6464f98f0d878cf095a1f8b4c969cb4994e6e7 025dea03bd55555	-	7.35719968178
273	$\mathbb{B}_{273}^{22,11}$	1fffff800ff0338dbc2761b32787386c3e4e52c693696331a762d 1c932b54aa955555	-	7.21062306502
274	$\mathbb{B}_{273} \circ \eta_2$	3fffff001fe0671b784ec3664f0e70d87c9ca58d26d2c6634ec5a 392656a9552aaaaa	-	7.1843062201
275	$\mathbb{B}_{275}^{22,11}$	7ffffe003f63918fc8c7a478f24e63c1e247694b66c72da47a4dc ad91b62b556aaaaa	-	7.50099186669
276	$\mathbb{B}_{275} \circ \eta_1$	fffffc007ec7231f918f48f1e49cc783c48ed296cd8e5b48f49b9 5b236c56aad55555	-	7.47703180212
277	$\mathbb{B}_{275} \circ \eta_1 \circ \eta_5$	1fffffc007ec7231f918f48f1e49cc783c48ed296cd8e5b48f49b 95b236c56aad55555	-	7.45520792849
278	$\mathbb{B}_{275} \circ \eta_1 \circ \eta_2 \circ \eta_5$	3fffff800fd8e463f231e91e3c9398f07891da52d9b1cb691e937 2b646d8ad55aaaaaa	-	7.12557624931

Table B.7 A list of binary sequences with record merit factor values and lengths between 279 and 312

n	Class	Record sequence in HEX	Old MF	New MF
279	$\mathbb{B}_{275} \circ \eta_1 \circ \eta_2 \circ \eta_1 \circ \eta_5$	7fffff001fb1c8c7e463d23c792731e0f123b4a5b36396d23d26e56c8db15aab555555	-	7.05209277043
280	$\mathbb{B}_{281} \circ \eta_4$	fffffc007936c187ec0dd8f03db13cda71b39278cb138b52d88d4ea594e31a554aaaaa	-	7.27002967359
281	$\mathbb{B}_{281}^{22,11}$	1fffff800f26d830fd81bb1e07b6279b4e36724f1962716a5b11a9d4b29c634aa955555	7.5058	7.7050156128
282	$\mathbb{B}_{281} \circ \eta_2$	3fffff001e4db061fb03763c0f6c4f369c6ce49e32c4e2d4b62353a96538c69552aaaaa	-	7.38933283776
283	$\mathbb{B}_{283}^{22,11}$	7ffffe003f9c0f3c38d867139330e53e47a17a46b06d331b12658db4b2d49ab556aaaaa	7.5088	8.17067945317
284	$\mathbb{B}_{283} \circ \eta_1$	fffffc007f381e7871b0ce272661ca7c8f42f48d60da663624cb1b6965a9356aad55555	-	7.89815902859
285	$\mathbb{B}_{285}^{22,11}$	1fffff800ff218fc31e07b06d24f26c6c6d999c6c6c634e3c6b16a5b2d49a354aa955555	7.0142	7.46827877896
286	$\mathbb{B}_{285} \circ \eta_1$	3fffff001fe431f863c0f60da49e4d8d8db3338d8d8c69c78d62d4b65a9346a9552aaaab	-	7.22707192083
287	$\mathbb{B}_{287}^{22,11}$	7ffffe003fe06d8db331e63e16e4b4c78c6d0e4da4c3c6e16b6693338d8e56ab556aaaaa	-	7.47314461985
288	$\mathbb{B}_{287} \circ \eta_1$	fffffc007fc0db1b6663cc7c2dc9698f18da1c9b49878dc2d6cd26671b1cad56aad55555	-	7.21503131524
289	$\mathbb{B}_{289}^{22,11,6}$	1fffff800ff619ccc3c4e6723d3a5e35b0f24e34b1f25e13d23664ed2ccd9a754aa955555	-	7.1312329235
290	$\mathbb{B}_{291} \circ \eta_4$	3fffff001fec4a76992c5a7072d0cd969cc9b18cd879ccbc36b61ec398760ec55aab55555	-	7.36040609137
291	$\mathbb{B}_{291}^{22,11,6}$	7ffffe003fd894ed3258b4e0e5a19b2d39936319b0f399786d6c3d8730ec1d8ab556aaaaa	-	7.71370012753
292	$\mathbb{B}_{291} \circ \eta_1$	fffffc007fb129da64b169c1cb43365a7326c63361e732f0dad87b0e61d83b156aad55555	-	7.58846564614
293	$\mathbb{B}_{291} \circ \eta_1 \circ \eta_5$	1fffffc007fb129da64b169c1cb43365a7326c63361e732f0dad87b0e61d83b156aad55555	-	7.47032718413
294	$\mathbb{B}_{291} \circ \eta_1 \circ \eta_1 \circ \eta_5$	3fffff800ff6253b4c962d38396866cb4e64d8c66c3ce65e1b5b0f61cc3b0762ad55aaaaab	-	7.33129770992
295	$\mathbb{B}_{295}^{22,11,6}$	7ffffe003fc319e87b0f1cc672969a589c69c49e49d879e1f264c92d3a5e9934ab556aaaaa	-	7.28730530899
296	$\mathbb{B}_{295} \circ \eta_1$	fffffc007f8633d0f61e398ce52d34b138d3893c93b0f3c3e4c9925a74bd326956aad55555	-	7.13950456323
297	$\mathbb{B}_{297}^{22,11,6}$	1fffff800fce123d389c6a5e972c63932799399399633926c3785e06d893d23a4d4aa955555	-	7.1551752109
298	$\mathbb{B}_{297} \circ \eta_2$	3fffff001f9c247a7138d4bd2e58c7264f32732732c6724d86f0bc0db127a4749a9552aaaaa	-	7.26829268293
299	$\mathbb{B}_{297} \circ \eta_2 \circ \eta_5$	7fffff001f9c247a7138d4bd2e58c7264f32732732c6724d86f0bc0db127a4749a9552aaaaa	-	7.38485048736
300	$\mathbb{B}_{301} \circ \eta_4$	b7e00048d23dbb673e05a41e139b4cb590bd183cc39b16947856b263b8b70dc5556a3d55	-	7.67132628708
301	$\mathbb{B}_{301}^{12}$	16fc00091a47b76ce7c0b483c2736996b217a307987362d28f0ad64c7716e1b8aaad47aaa	7.7173	8.24544958136
302	$\mathbb{B}_{301} \circ \eta_1$	2df80012348f6ed9cf81690784e6d32d642f460f30e6c5a51e15ac98ee2dc371555a8f555	-	8.092635315
303	$\mathbb{B}_{301} \circ \eta_1 \circ \eta_1$	5bf00024691eddb39f02d20f09cda65ac85e8c1e61cd8b4a3c2b5931dc5b86e2aab51eaab	7.9488	8.16370264983
304	$\mathbb{B}_{301} \circ \eta_1 \circ \eta_1 \circ \eta_1$	b7e00048d23dbb673e05a41e139b4cb590bd183cc39b16947856b263b8b70dc5556a3d557	-	8.24553890079
305	$\mathbb{B}_{301} \circ \eta_1 \circ \eta_1 \circ \eta_1 \circ \eta_2$	16fc00091a47b76ce7c0b483c2736996b217a307987362d28f0ad64c7716e1b8aaad47aaae	7.5117	8.12587351502
306	$\mathbb{B}_{301} \circ \eta_1 \circ \eta_1 \circ \eta_1 \circ \eta_6 \circ \eta_5$	2000b7e00048d23dbb673e05a41e139b4cb590bd183cc39b16947856b263b8b70dc5556a3d557	-	8.18353434714
307	$\mathbb{B}_{301} \circ \eta_1 \circ \eta_1 \circ \eta_1 \circ \eta_6 \circ \eta_5 \circ \eta_6$	2000b7e00048d23dbb673e05a41e139b4cb590bd183cc39b16947856b263b8b70dc5556a3d557	7.4932	8.27180972442
308	$\mathbb{B}_{301} \circ \eta_1 \circ \eta_1 \circ \eta_1 \circ \eta_2 \circ \eta_6 \circ \eta_5 \circ \eta_6$	40016fc00091a47b76ce7c0b483c2736996b217a307987362d28f0ad64c7716e1b8aaad47aaae	-	8.15823873409
309	$\mathbb{B}_{301} \circ \eta_1 \circ \eta_1 \circ \eta_1 \circ \eta_2 \circ \eta_2 \circ \eta_6 \circ \eta_5 \circ \eta_6$	8002df80012348f6ed9cf81690784e6d32d642f460f30e6c5a51e15ac98ee2dc371555a8f555c	7.5229	8.08338977311
310	$\mathbb{B}_{309} \circ \eta_6$	8002df80012348f6ed9cf81690784e6d32d642f460f30e6c5a51e15ac98ee2dc371555a8f555c	-	7.96716962361
311	$\mathbb{B}_{309} \circ \eta_6 \circ \eta_5$	48002df80012348f6ed9cf81690784e6d32d642f460f30e6c5a51e15ac98ee2dc371555a8f555c	7.4229	7.88786494862
312	$\mathbb{B}_{309} \circ \eta_2 \circ \eta_6 \circ \eta_5$	90005bf00024691eddb39f02d20f09cda65ac85e8c1e61cd8b4a3c2b5931dc5b86e2aab51eaab8	-	7.69152970923

Table B.8 A list of binary sequences with record merit factor values and lengths between 313 and 345

n	Class	Record sequence in HEX	Old MF	New MF
313	$\mathbb{B}_{313}^{24,11,9,4}$	1fffffe003fe04b878f0b32666cda7c2c5a4f98f4994e1ec2d61c c666330b49690ea552aa555555	7.5547	7.62048848787
314	$\mathbb{B}_{313} \circ \eta_1$	3fffffc007fc0970f1e1664ccd9b4f858b49f31e9329c3d85ac39 8ccc661692d21d4aa554aaaaab	-	7.31315828512
315	$\mathbb{B}_{315}^{23,10,7,4}$	7fffff003f87c0f184fe339cf0c6179e38de666466668db69a164 d2c9b36ac592d4a5ab552aaaaa	7.4661	7.5204638472
316	$\mathbb{B}_{315} \circ \eta_2$	fffffe007f0f81e309fc6739e18c2f3c71bcccc8cccd1b6d342c9 a59366d58b25a94b56aa555554	-	7.25908694388
317	$\mathbb{B}_{317}$	1ffffff003f920d9f36093b4ec686396db0f225e25e234b1c7926 86c4f138a7359ca3952ab555555	-	7.46131571132
318	$\mathbb{B}_{317} \circ \eta_2$	3ffffffe007f241b3e6c12769d8d0c72db61e44bc4bc469638f24d 0d89e2714e6b39472a556aaaaaa	-	7.23658222413
319	$\mathbb{B}_{319}^{25,10,7}$	7fffffc00fe0674e1939ce09ccf92963cd83c378da34b58cb61f1 acc9d6c9b196c2656ad54aaaaaa	7.4224	7.45720357614
320	$\mathbb{B}_{319} \circ \eta_2$	ffffff801fc0ce9c32739c1399f252c79b0786f1b4696b196c3e3 5993ad93632d84cad5aa9555554	-	7.48976009362
321	$\mathbb{B}_{323} \circ \eta_0$	1fffffc003f9807e330e1ce9c63d0e61e0cd83c9998d29cca5a64 bd26d84da4b3256a9952aad55555	7.3183	7.73116746699
322	$\mathbb{B}_{323} \circ \eta_4$	3fffffc003f9807e330e1ce9c63d0e61e0cd83c9998d29cca5a64 bd26d84da4b3256a9952aad55555	-	7.76891952645
323	$\mathbb{B}_{323}^{24,12,7}$	7fffff8007f300fc661c39d38c7a1cc3c19b0793331a53994b4c9 7a4db09b49664ad532a555aaaaaa	7.743	7.80788804071
324	$\mathbb{B}_{323} \circ \eta_2$	ffffff000fe601f8cc3873a718f4398783360f266634a73296992 f49b613692cc95aa654aab555554	-	7.68491947291
325	$\mathbb{B}_{323} \circ \eta_2 \circ \eta_1$	1ffffe001fcc03f19870e74e31e8730f066c1e4ccc694e652d32 5e936c26d25992b54ca9556aaaaa9	7.5167	7.61206399539
326	$\mathbb{B}_{323} \circ \eta_2 \circ \eta_1 \circ \eta_6$	1ffffe001fcc03f19870e74e31e8730f066c1e4ccc694e652d32 5e936c26d25992b54ca9556aaaaa9	-	7.50642746151
327	$\mathbb{B}_{327}^{25,12,11,6}$	7fffffc003ff81ec63781ce43c19c39e1e66786619665a666969b 4994b46c95a364e95aab554aaaaaa	7.3009	7.76761586518
328	$\mathbb{B}_{323} \circ \eta_2 \circ \eta_1 \circ \eta_1 \circ \eta_6 \circ \eta_6$	3fffffc003f9807e330e1ce9c63d0e61e0cd83c9998d29cca5a64 bd26d84da4b3256a9952aad555553	-	7.22622246104
329	$\mathbb{B}_{327} \circ \eta_1 \circ \eta_2$	1ffffff000ffe07b18de07390f0670e787999e19865996999a5a6 d2652d1b2568d93a56aad552aaaaaa	7.2782	7.33340108401
330	$\mathbb{B}_{327} \circ \eta_1 \circ \eta_2 \circ \eta_1$	3ffffffe001ffc0f631bc0e721e0ce1cf0f333c330cb32d3334b4d a4ca5a364ad1b274ad55aaa5555555	-	7.22819593787
331	$\mathbb{B}_{331}^{24,12,8,2}$	7fffff8007f83c0fcc36f0de9667261b24cb1a5b63638793cc739 672661e8d2e34cad4b5aa555aaaaaa	7.3501	7.44603778714
332	$\mathbb{B}_{331} \circ \eta_2$	ffffff000ff0781f986de1bd2cce4c36499634b6c6c70f2798e72 ce4cc3d1a5c6995a96b54aab555554	-	7.2117246794
333	$\mathbb{B}_{333}^{24,12,8,3}$	1fffffe001fe3181fc303c963e42f18cc6c63a670b70b66126c6c c9b42e5278d2b2d5a9b255aaa555555	7.2743	7.29724927613
334	$\mathbb{B}_{333} \circ \eta_2$	3fffffc003fc6303f860792c7c85e3198d8c74ce16e16cc24d8d9 93685ca4f1a565ab5364ab554aaaaaa	-	7.06855911798
335	$\mathbb{B}_{335}^{25,12,8,2}$	7fffffc003fc81e67b3698760f1b0f0cb46e46b61e6963e46e43c d2d392d6259e33a6695cab554aaaaaa	7.3302	7.46276100545
336	$\mathbb{B}_{335} \circ \eta_2$	ffffff8007f903ccf66d30ec1e361e1968dc8d6c3cd2c7c8dc879 a5a725ac4b3c674cd2b956aa9555554	-	7.27422680412
337	$\mathbb{B}_{337}^{25,12,8,2}$	1ffffff000ff219cc6f03790f81e4f25a6931cb399b19930db386 1e34e5a94b972b46cd9a354aab555555	7.3327	7.35550518135
338	$\mathbb{B}_{337} \circ \eta_2$	3ffffffe001fe43398de06f21f03c9e4b4d26396733633261b670c 3c69cb52972e568d9b346a9556aaaaaa	-	7.26280991736
339	$\mathbb{B}_{339}^{26,13,10,2}$	7ffffffe000ffc231ec37835a670f85a749ccd8db61b9638d8cc9c 2785ad267835a34e9374aad556aaaaaa	7.3961	7.77228459353
340	$\mathbb{B}_{339} \circ \eta_1$	ffffffc001ff8463d86f06b4ce1f0b4e9399b1b6c372c71b19938 4f0b5a4cf06b469d26e955aaad555555	-	7.7500670421
341	$\mathbb{B}_{339} \circ \eta_1 \circ \eta_5$	1ffffffc001ff8463d86f06b4ce1f0b4e9399b1b6c372c71b1993 84f0b5a4cf06b469d26e955aaad555555	7.317	7.72939377825
342	$\mathbb{B}_{339} \circ \eta_1 \circ \eta_1 \circ \eta_5$	3ffffff8003ff08c7b0de0d699c3e169d2733636d86e58e363327 09e16b499e0d68d3a4dd2ab555aaaaaab	-	7.67984241628
343	$\mathbb{B}_{339} \circ \eta_1 \circ \eta_1 \circ \eta_5 \circ \eta_6$	3ffffff8003ff08c7b0de0d699c3e169d2733636d86e58e363327 09e16b499e0d68d3a4dd2ab555aaaaaab	7.1921	7.63260672116
344	$\mathbb{B}_{339} \circ \eta_1 \circ \eta_1 \circ \eta_1 \circ \eta_5 \circ \eta_6$	7ffffff0007fe118f61bc1ad3387c2d3a4e66c6db0dcb1c6c664e 13c2d6933c1ad1a749ba556aab5555557	-	7.36286709806
345	$\mathbb{B}_{345}^{27,13,10,6}$	1ffffffc001ff8187e1f2364edc38f18e730f933a53a13e13394b 3649b492dc4e7235a569a955aaad555555	7.2612	7.48773276296

Table B.9 A list of binary sequences with record merit factor values and lengths between 346 and 380

n	Class	Record sequence in HEX	Old MF	New MF
346	$\mathbb{B}_{339} \circ \eta_1 \circ \eta_1 \circ \eta_1 \circ \eta_1 \circ \eta_2 \circ \eta_5 \circ \eta_6$	1ffffffc001ff8463d86f06b4ce1f0b4e9399b1b6c372c71b1993 84f0b5a4cf06b469d26e955aaad555555e	-	7.10818192614
347	$\mathbb{B}_{347}^{26,13,10,0}$	7fffffe000ffcc0f1e19c327c33966f03f21d8e17a6367a16d897 2b52e61b34a73499692d4caad556aaaaaa	7.1698	7.23177177177
348	$\mathbb{B}_{347} \circ \eta_2$	ffffffc001ff981e3c33864f8672cde07e43b1c2f4c6cf42db12e 56a5cc36694e6932d25a9955aaad555554	-	7.13888233907
349	$\mathbb{B}_{349}^{24,9,9,9}$	1fffffe00ff8033e76c49b0a79274c36a5ad0f0399b3931992b4b c1e072cf63960b18ec76532a954aa555555	7.1295	7.24488460623
350	$\mathbb{B}_{349} \circ \eta_2$	3fffffc01ff0067ced893614f24e986d4b5a1e073367263325697 83c0e59ec72c1631d8eca6552a954aaaaaa	-	7.14619064287
351	$\mathbb{B}_{349} \circ \eta_2 \circ \eta_1$	7fffff803fe00cf9db126c29e49d30da96b43c0e66ce4c664ad2f 0781cb3d8e582c63b1d94caa552a9555555	7.0911	7.14043120436
352	$\mathbb{B}_{349} \circ \eta_2 \circ \eta_1 \circ \eta_5$	ffffff803fe00cf9db126c29e49d30da96b43c0e66ce4c664ad2f 0781cb3d8e582c63b1d94caa552a9555555	-	7.05603644647
353	$\mathbb{B}_{353}^{23,9,9,9}$	1fffffc01ff006cf9f21e06d8e74c72667963c1b8b6b4f07091ad 27966636cf649c6a5a3594c6ab55aad55555	7.1385	7.26498367537
354	$\mathbb{B}_{355} \circ \eta_4$	3fffff803fe00f39e1b8784d2f26ccd9cc63c0f307c24e2d6b34a d26cd9ccc6343ce9691a5934aa552a955555	-	7.18472652219
355	$\mathbb{B}_{355}^{23,9,9,9}$	7fffff007fc01e73c370f09a5e4d99b398c781e60f849c5ad6695 a4d9b3998c6879d2d234b26954aa552aaaaa	7.232	7.39496537965
356	$\mathbb{B}_{355} \circ \eta_2$	fffffe00ff803ce786e1e134bc9b3367318f03cc1f0938b5acd2b 49b3673318d0f3a5a46964d2a954aa555554	-	7.20582215147
357	$\mathbb{B}_{357}^{26,12,6}$	1ffffff8007f876cc33e58e06f198373723c78d293d29983d383c 96d23737299b46a49e532cc76956aa9555555	7.2201	7.48467230444
358	$\mathbb{B}_{357} \circ \eta_2$	3ffffff000ff0ed9867cb1c0de3306e6e478f1a527a53307a7079 2da46e6e53368d493ca6598ed2ad552aaaaaa	-	7.27625752243
359	$\mathbb{B}_{357} \circ \eta_2 \circ \eta_2$	7ffffffe001fe1db30cf96381bc660dcdc8f1e34a4f4a660f4e0f2 5b48dcdca66d1a92794cb31da55aaa5555554	7.1896	7.33361784454
360	$\mathbb{B}_{357} \circ \eta_2 \circ \eta_2 \circ \eta_5$	ffffffe001fe1db30cf96381bc660dcdc8f1e34a4f4a660f4e0f2 5b48dcdca66d1a92794cb31da55aaa5555554	-	7.12714474263
361	$\mathbb{B}_{357} \circ \eta_2 \circ \eta_2 \circ \eta_5 \circ \eta_6$	ffffffe001fe1db30cf96381bc660dcdc8f1e34a4f4a660f4e0f2 5b48dcdca66d1a92794cb31da55aaa5555554	7.1229	7.17310656099
362	$\mathbb{B}_{363} \circ \eta_4$	3ffffff000fde381fac783318cb427c3396999ce1ccbc4ed0cda4 d9987932d62f0c9b3296c15a925d4aab555555	-	7.86295451818
363	$\mathbb{B}_{363}^{26,12,6}$	7ffffffe001fbc703f58f066319684f8672d3339c399789da19b49 b330f265ac5e1936652d82b524ba9556aaaaaa	7.6	7.92929353713
364	$\mathbb{B}_{363} \circ \eta_1$	ffffffc003f78e07eb1e0cc632d09f0ce5a66738732f13b433693 6661e4cb58bc326cca5b056a49752aad555555	-	7.68003709715
365	$\mathbb{B}_{363} \circ \eta_1 \circ \eta_2$	1ffffff8007ef1c0fd63c198c65a13e19cb4cce70e65e276866d2 6ccc3c996b17864d994b60ad492ea555aaaaaaa	7.2421	7.46274927179
366	$\mathbb{B}_{363} \circ \eta_1 \circ \eta_2 \circ \eta_2$	3ffffff000fde381fac783318cb427c3396999ce1ccbc4ed0cda4 d9987932d62f0c9b3296c15a925d4aab5555554	-	7.43291532571
367	$\mathbb{B}_{363} \circ \eta_1 \circ \eta_2 \circ \eta_2 \circ \eta_2$	7ffffffe001fbc703f58f066319684f8672d3339c399789da19b49 b330f265ac5e1936652d82b524ba9556aaaaaa8	7.0216	7.28600021638
368	$\mathbb{B}_{363} \circ \eta_1 \circ \eta_2 \circ \eta_2 \circ \eta_2 \circ \eta_2$	ffffffc003f78e07eb1e0cc632d09f0ce5a66738732f13b433693 6661e4cb58bc326cca5b056a49752aad5555550	-	7.07692307692
369	$\mathbb{B}_{363} \circ \eta_5 \circ \eta_5 \circ \eta_6 \circ \eta_5 \circ \eta_6 \circ \eta_6$	5ffffffe001fbc703f58f066319684f8672d3339c399789da19b4 9b330f265ac5e1936652d82b524ba9556aaaaaa	7.074	7.19667019027
370	$\mathbb{B}_{369} \circ \eta_6$	5ffffffe001fbc703f58f066319684f8672d3339c399789da19b4 9b330f265ac5e1936652d82b524ba9556aaaaaa	-	7.02988600185
371	$\mathbb{B}_{371}^{23,9,9,9}$	7fffff007fc01f9de489b87b439839334cc34a5a736938cdb1e32 787c34cc331b59b43a5b9dc689a954aa552aaaaa	7.333	7.44730007575
372	$\mathbb{B}_{371} \circ \eta_1$	fffffe00ff803f3bc91370f687307266998694b4e6d2719b63c64 f0f869986636b36874b73b8d1352a954aa555555	-	7.53561315618
373	$\mathbb{B}_{371} \circ \eta_1 \circ \eta_5$	1fffffe00ff803f3bc91370f687307266998694b4e6d2719b63c6 4f0f869986636b36874b73b8d1352a954aa555555	7.2147	7.62601403201
374	$\mathbb{B}_{371} \circ \eta_1 \circ \eta_2 \circ \eta_5$	3fffffc01ff007e779226e1ed0e60e4cd330d2969cda4e336c78c 9e1f0d330cc6d66d0e96e771a26a552a954aaaaaa	-	7.51133068414
375	$\mathbb{B}_{371} \circ \eta_1 \circ \eta_2 \circ \eta_5 \circ \eta_5$	7fffffc01ff007e779226e1ed0e60e4cd330d2969cda4e336c78c 9e1f0d330cc6d66d0e96e771a26a552a954aaaaaa	7.0011	7.40209495736
376	$\mathbb{B}_{371} \circ \eta_1 \circ \eta_2 \circ \eta_2 \circ \eta_1 \circ \eta_5$	ffffff007fc01f9de489b87b439839334cc34a5a736938cdb1e32 787c34cc331b59b43a5b9dc689a954aa552aaaaa9	-	7.28141738772
377	$\mathbb{B}_{377}^{27,13,10,5}$	1ffffffc001ff80e783ce03cce4f872786f07b0db0d99c6666d99 cb1cb16b46963694e4cd2a4d2964a955aaad555555	7.2249	7.35048613984
378	$\mathbb{B}_{377} \circ \eta_2$	3ffffff8003ff01cf079c0799c9f0e4f0de0f61b61b338cccdb33 963962d68d2c6d29c99a549a52c952ab555aaaaaaa	-	7.11786390356
379	$\mathbb{B}_{379}^{27,13,10,5}$	7ffffff0007fe0761e4f90f21fc30f06d8d86cf09b319a6367993 39d2ce58d8e52d34a972d1ac696256aa5552aaaaaa	7.2422	7.33685769742
380	$\mathbb{B}_{379} \circ \eta_1 \circ \eta_1 \circ \eta_3$	ffffffc001ff81d8793e43c87f0c3c1b6361b3c26cc6698d9e64c e74b39636394b4d2a5cb46b1a5895aa9554aaaaaab	-	7.01924946529

Table B.10 A list of binary sequences with record merit factor values and lengths between 381 and 414

n	Class	Record sequence in HEX	Old MF	New MF
381	$\mathbb{B}_{381}^{27,13,10,5}$	1ffffffc001ff818f0787c0d9e13e0cf0e49e43cc39c667333666 d92cd2e58e4b4ca53a59cad696b49a955aaad555555	7.106	7.2018753721
382	$\mathbb{B}_{381} \circ \eta_2$	3ffffff8003ff031e0f0f81b3c27c19e1c93c8798738cce666ccd b259a5cb1c96994a74b395ad2d69352ab555aaaaaaa	-	7.08437712399
383	$\mathbb{B}_{383}^{27,13,10,5}$	7ffffff0007fe07319cce478e730f318f01bc1e3963c3f27272b4 b61b694b952d932d326da46cc993256aa5552aaaaaa	7.0314	7.07480466866
384	$\mathbb{B}_{383} \circ \eta_1 \circ \eta_2 \circ \eta_3$	ffffffc001ff81cc673391e39cc3cc63c06f078e58f0fc9c9cad2 d86da52e54b64cb4c9b691b3264c95aa9554aaaaaaa	-	7.08923076923
385	$\mathbb{B}_{385}^{27,13,10,7}$	1ffffffc001ff80d8c72d86c73343b439960f721f8d2d61b331a7 c3c95a374a7992f12f336c69c36c9ca955aaad555555	7.0772	7.24887519562
386	$\mathbb{B}_{385} \circ \eta_2$	3ffffff8003ff01b18e5b0d8e668768732c1ee43f1a5ac366634f 8792b46e94f325e25e66d8d386d93952ab555aaaaaaa	-	7.1296774811
387	$\mathbb{B}_{387}^{27,13,10,5}$	7ffffff0007fe033c1f66139c2d327c96786f078d8dc39b33339b 48d8da52e5a61ca730f49b166294b356aa5552aaaaaa	7.1502	7.19974040958
388	$\mathbb{B}_{387} \circ \eta_1$	ffffffe000ffc06783ecc27385a64f92cf0de0f1b1b8736666736 91b1b4a5cb4c394e61e9362cc52966ad54aaa5555555	-	7.06248827172
389	$\mathbb{B}_{391} \circ \eta_0$	1ffffff8003ff01c39c3e47876c1da64f931b70cda1e723633272 365a1ccb71b394e61dac7696e52d92dab552aa9555555	7.0461	7.39305256987
390	$\mathbb{B}_{391} \circ \eta_4$	3ffffff8003ff01c39c3e47876c1da64f931b70cda1e723633272 365a1ccb71b394e61dac7696e52d92dab552aa9555555	-	7.27403156385
391	$\mathbb{B}_{391}^{27,13,10,7,1}$	7ffffff0007fe0387387c8f0ed83b4c9f2636e19b43ce46c664e4 6cb43996e36729cc3b58ed2dca5b25b56aa5552aaaaaa	7.1553	7.16070257611
392	$\mathbb{B}_{391} \circ \eta_1$	ffffffe000ffc070e70f91e1db076993e4c6dc336879c8d8cc9c8 d968732dc6ce539876b1da5b94b64b6ad54aaa5555555	-	7.1832460733
393	$\mathbb{B}_{391} \circ \eta_1 \circ \eta_2$	1ffffffc001ff80e1ce1f23c3b60ed327c98db866d0f391b19939 1b2d0e65b8d9ca730ed63b4b7296c96d5aa9554aaaaaaa	7.0952	7.19304210134
394	$\mathbb{B}_{391} \circ \eta_1 \circ \eta_2 \circ \eta_5$	3ffffffc001ff80e1ce1f23c3b60ed327c98db866d0f391b19939 1b2d0e65b8d9ca730ed63b4b7296c96d5aa9554aaaaaaa	-	7.22095078612
395	$\mathbb{B}_{391} \circ \eta_1 \circ \eta_2 \circ \eta_5 \circ \eta_5$	7ffffffc001ff80e1ce1f23c3b60ed327c98db866d0f391b19939 1b2d0e65b8d9ca730ed63b4b7296c96d5aa9554aaaaaaa	7.0991	7.23610982284
396	$\mathbb{B}_{395} \circ \eta_1$	ffffffe000ffc0664e333c1e1e703721f9927c19cf1a65b0f34b1 e61b4d9ad63995a372b65a5ad3324e66ad54aaa5555555	-	7.15270935961
397	$\mathbb{B}_{397}^{27,13,8}$	1ffffffc001ff9213e36cf18664c8978353c49d2c9da358e948f8 49f21d8c3d8ed3f29788ce669b4c7253a3955aaad555555	7.0829	7.19675799087
398	$\mathbb{B}_{399} \circ \eta_4$	3ffffff8003fe69c336f097b072cd296cf04ed8b53ad278d99999 c963c13f09c4eb4c783cc36b178b4732d86552aa9555555	-	7.12056099973
399	$\mathbb{B}_{399}^{27,13,8}$	7ffffff0007fcd3866de12f60e59a52d9e09db16a75a4f1b33333 92c7827e1389d698f07986d62f168e65b0caa5552aaaaaa	7.1487	7.23180703189
400	$\mathbb{B}_{399} \circ \eta_1$	ffffffe000ff9a70cdbc25ec1cb34a5b3c13b62d4eb49e3666667 258f04fc2713ad31e0f30dac5e2d1ccb61954aaa5555555	-	7.10479573712
401	$\mathbb{B}_{401}^{27,13,9}$	1ffffffc001ff901e1c3cd32370b1b0f16e06e4392664db258b09 e31ce66392e46a47b4b1b0b7233cd2da5ab955aaad555555	7.0084	7.01820006983
402	$\mathbb{B}_{407} \circ \eta_0 \circ \eta_0 \circ \eta_4$	3ffffffe001fc8787c06f0f067337903d892d23cc1a79db198e764 99b1d961acd23c389d2b973366b4b46ad6968d5aaa555555	-	7.05755961219
403	$\mathbb{B}_{407} \circ \eta_0 \circ \eta_0$	7ffffffc003f90f0f80de1e0ce66f207b125a479834f3b6331cec9 3363b2c359a478713a572e66cd6968d5ad2d1ab554aaaaaa	-	7.10264147643
404	$\mathbb{B}_{407} \circ \eta_0 \circ \eta_4$	ffffffc003f90f0f80de1e0ce66f207b125a479834f3b6331cec9 3363b2c359a478713a572e66cd6968d5ad2d1ab554aaaaaa	-	7.02911283376
405	$\mathbb{B}_{407} \circ \eta_4 \circ \eta_4$	1ffffffc003f90f0f80de1e0ce66f207b125a479834f3b6331cec 93363b2c359a478713a572e66cd6968d5ad2d1ab554aaaaaa	-	7.09082656061
406	$\mathbb{B}_{407} \circ \eta_4$	3ffffff8007f21e1f01bc3c19ccde40f624b48f3069e76c6639d9 266c76586b348f0e274ae5ccd9ad2d1ab5a5a356aa9555555	-	7.03765690377
407	$\mathbb{B}_{407}^{27,12,6}$	7ffffff000fe43c3e0378783399bc81ec49691e60d3ced8cc73b2 4cd8ecb0d6691e1c4e95cb99b35a5a356b4b46ad552aaaaaa	-	7.11856467555
408	$\mathbb{B}_{407} \circ \eta_2$	ffffffe001fc8787c06f0f067337903d892d23cc1a79db198e764 99b1d961acd23c389d2b973366b4b46ad6968d5aaa5555554	-	7.1801242236
409	$\mathbb{B}_{407} \circ \eta_2 \circ \eta_6$	ffffffe001fc8787c06f0f067337903d892d23cc1a79db198e764 99b1d961acd23c389d2b973366b4b46ad6968d5aaa5555554	-	7.24285590578
410	$\mathbb{B}_{407} \circ \eta_2 \circ \eta_2 \circ \eta_6$	1ffffffc003f90f0f80de1e0ce66f207b125a479834f3b6331cec 93363b2c359a478713a572e66cd6968d5ad2d1ab554aaaaaa8	-	7.0767028711
411	$\mathbb{B}_{411}^{28,14,11,8,4}$	7ffffff8001ffc03c6b678721f0d30dce4da41f861d984f2399c9 9b72c598965a9478c6c8d30d29725a63e4b54aa9555aaaaaaa	-	7.13047699451
412	$\mathbb{B}_{411} \circ \eta_1$	fffffff0003ff8078d6cf0e43e1a61b9c9b483f0c3b309e473393 36e58b312cb528f18d91a61a52e4b4c7c96a9552aab5555555	-	7.01537444206
413	$\mathbb{B}_{413}^{28,14,10,6}$	1fffffffe0007fe06f0cfc3189e43ce49c9c9c333c0fc3c338cf0c b4c932d2d4ad332d8d8d8e4d2e589b2d4cb46a556aaa5555555	-	7.02855612329
414	$\mathbb{B}_{415} \circ \eta_4$	3ffffffc000ffc0783ccf807b61bc3c9e4b198d398721b4a76699 86760f1a36993c99b0e58d2d1a716a94cd296ad54aaad555555	-	7.09714285714

Table B.11 A list of binary sequences with record merit factor values and lengths between 415 and 441

n	Class	Record sequence in HEX	Old MF	New MF
415	$\mathbb{B}_{415}^{28,14,10,6}$	7ffffff8001ff80f0799f00f6c378793c96331a730e43694ecd33 0cec1e346d32793361cb1a5a34e2d5299a52d5aa9555aaaaaaa	-	7.25891427126
416	$\mathbb{B}_{415} \circ \eta_1$	fffffff0003ff01e0f33e01ed86f0f2792c6634e61c86d29d9a66 19d83c68da64f266c39634b469c5aa5334a5ab552aab5555555	-	7.04855001629
417	$\mathbb{B}_{417}^{28,13,12,8,6,6,4}$	1ffffffe000fff00fc0fc92787993c3e0793934f867264730c739 936cb36e636694f39396a52d39969638d4ad4ab554aaa5555555	-	7.22610538564
418	$\mathbb{B}_{417} \circ \eta_2$	3ffffffc001ffe01f81f924f0f32787c0f27269f0ce4c8e618e73 26d966dcc6cd29e7272d4a5a732d2c71a95a956aa9554aaaaaaa	-	7.13683522588
419	$\mathbb{B}_{417} \circ \eta_2 \circ \eta_5$	7ffffffc001ffe01f81f924f0f32787c0f27269f0ce4c8e618e73 26d966dcc6cd29e7272d4a5a732d2c71a95a956aa9554aaaaaaa	-	7.05120893244
420	$\mathbb{B}_{421} \circ \eta_4$	ffffff0007ff807e07f031b2d86c9cc670cd8786d23c9c399b0c d399b49cb70e5a58cd264c9ce58f39352a56a55aaa5552aaaaaa	-	7.04585397028
421	$\mathbb{B}_{421}^{28,13,12,8,6,6,4}$	1ffffffe000fff00fc0fe06365b0d9398ce19b0f0da4793873361 9a73369396e1cb4b19a4c9939cb1e726a54ad4ab554aaa5555555	-	7.25327385824
422	$\mathbb{B}_{421} \circ \eta_1$	3ffffffc001ffe01f81fc0c6cb61b27319c3361e1b48f270e66c3 34e66d272dc39696334993273963ce4d4a95a956aa9554aaaaaab	-	7.05842251288
423	$\mathbb{B}_{423}^{28,14,11,8,4}$	7ffffff8001ffc03cc366f03d907b07993e4e1f21e1c3c66d2727 27270e64b49697296c6b19a53a518b52e634cb54aa9555aaaaaaa	-	7.17150300601
424	$\mathbb{B}_{423} \circ \eta_1$	fffffff0003ff807986cde07b20f60f327c9c3e43c3878cda4e4e 4e4e1cc9692d2e52d8d6334a74a316a5cc6996a9552aab5555555	-	7.08448928121
425	$\mathbb{B}_{425}^{28,14,10,6}$	1ffffffe0007fe03c25bc93927f0bc3326799334f07e4c9d90db7 0b71cb9d8ce56b4f339966332d0b563938d1e2d2a556aaa555555 5	-	7.0667057903
426	$\mathbb{B}_{429} \circ \eta_0 \circ \eta_4$	3ffffff8001ffc03f07e1330b4e0e721a5e24d8e664e34e1e49b0 dcb18e5a4f24e6649ce25e1a364a4f0b33a56b52ad55aaa955555 5	-	7.05528341498
427	$\mathbb{B}_{429} \circ \eta_0$	7ffffff0003ff807e0fc266169c1ce434bc49b1ccc9c69c3c9361 b9631cb49e49ccc939c4bc346c949e16674ad6a55aab5552aaaaa a	-	7.22782050266
428	$\mathbb{B}_{429} \circ \eta_4$	fffffff0003ff807e0fc266169c1ce434bc49b1ccc9c69c3c9361 b9631cb49e49ccc939c4bc346c949e16674ad6a55aab5552aaaaa a	-	7.13889321902
429	$\mathbb{B}_{429}^{28,14,11,8,4}$	1ffffffe0007ff00fc1f84cc2d3839c86978936399938d387926c 372c639693c93999273897868d9293c2cce95ad4ab556aaa55555 55	-	7.05354131535
430	$\mathbb{B}_{431} \circ \eta_4$	3ffffffc000ffe01e786f1c27b0cf61f0e633c1e9c3998ce46e52 d9c3e46e4c9992d85ad3264b5a74cb162db46965aa554aaad5555 55	-	7.0021964705
431	$\mathbb{B}_{431}^{28,14,11,8,4}$	7ffffff8001ffc03cf0de384f619ec3e1cc6783d3873319c8dca5 b387c8dc993325b0b5a64c96b4e9962c5b68d2cb54aa9555aaaaa aa	-	7.08849118522
432	$\mathbb{B}_{431} \circ \eta_1$	fffffff0003ff8079e1bc709ec33d87c398cf07a70e663391b94b 670f91b932664b616b4c992d69d32c58b6d1a596a9552aab55555 55	-	7.04560555723
433	$\mathbb{B}_{433}^{28,13,12,8,6,6,4}$	1ffffffe000fff00fc0fc333247c1e1e0c7387999a5b8d3c619ce 1cda4d9a6d3c91e19996936ca5a5ad6e3332d4ad4ab554aaa5555 555	-	7.05482390126
434	$\mathbb{B}_{433} \circ \eta_2$	3ffffffc001ffe01f81f866648f83c3c18e70f3334b71a78c339c 39b49b34da7923c3332d26d94b4b5adc6665a95a956aa9554aaaa aaa	-	7.06140811277
435	$\mathbb{B}_{433} \circ \eta_2 \circ \eta_1$	7ffffff8003ffc03f03f0ccc91f0787831ce1e66696e34f186738 73693669b4f24786665a4db29696b5b8cccb52b52ad552aa95555 555	-	7.06854688084
436	$\mathbb{B}_{433} \circ \eta_2 \circ \eta_1 \circ \eta_5$	fffffff8003ffc03f03f0ccc91f0787831ce1e66696e34f186738 73693669b4f24786665a4db29696b5b8cccb52b52ad552aa95555 555	-	7.07307635065
437	$\mathbb{B}_{433} \circ \eta_2 \circ \eta_1 \circ \eta_5 \circ \eta_5$	1fffffff8003ffc03f03f0ccc91f0787831ce1e66696e34f18673 873693669b4f24786665a4db29696b5b8cccb52b52ad552aa9555 5555	-	7.07816901408
438	$\mathbb{B}_{441} \circ \eta_4 \circ \eta_4 \circ \eta_4$	3ffffff0007fe0479c679c39c3873b1ce523c4e1f09784e46c3f0 cdc8cd2b4e46c5a1d296c4b706c93b25b49b49a649a456aa5552a aaaa	-	7.01645819618
439	$\mathbb{B}_{441} \circ \eta_0$	7fffffc001ff811e719e70e70e1cec73948f1387c25e1391b0fc3 372334ad391b16874a5b12dc1b24ec96d26d269926915aa9554aa aaaa	-	7.07544606799
440	$\mathbb{B}_{441} \circ \eta_4$	ffffffc001ff811e719e70e70e1cec73948f1387c25e1391b0fc3 372334ad391b16874a5b12dc1b24ec96d26d269926915aa9554aa aaaa	-	7.16400236827
441	$\mathbb{B}_{441}^{26,13,10,6}$	1ffffff8003ff023ce33ce1ce1c39d8e7291e270f84bc272361f8 66e46695a72362d0e94b625b83649d92da4da4d324d22b552aa95 55555	-	7.25458818263

Table B.12 A list of binary sequences with record merit factor values and lengths between 442 and 464

n	Class	Record sequence in HEX	Old MF	New MF
442	$\mathbb{B}_{441} \circ \eta_2$	3ffffff0007fe0479c679c39c3873b1ce523c4e1f09784e46c3f0 cdc8cd2b4e46c5a1d296c4b706c93b25b49b49a649a456aa5552a aaaaa	-	7.07994491556
443	$\mathbb{B}_{441} \circ \eta_2 \circ \eta_1$	7fffffe000ffc08f38cf3873870e7639ca4789c3e12f09c8d87e1 9b919a569c8d8b43a52d896e0d92764b6936934c9348ad54aaa55 55555	-	7.06694274397
444	$\mathbb{B}_{441} \circ \eta_2 \circ \eta_1 \circ \eta_2$	ffffffc001ff811e719e70e70e1cec73948f1387c25e1391b0fc3 372334ad391b16874a5b12dc1b24ec96d26d269926915aa9554aa aaaaa	-	7.07798362775
445	$\mathbb{B}_{445}^{27,13,10,7,3,3}$	1ffffffc001ff80e1cce3c34c6783d2663e14b913b461ec631e07 98736996a5b26c5a6f13b90fa52663d2966cf2d24cda4a955aaad 555555	-	7.05317709075
446	$\mathbb{B}_{445} \circ \eta_2$	3ffffff8003ff01c399c78698cf07a4cc7c29722768c3d8c63c0f 30e6d32d4b64d8b4de27721f4a4cc7a52cd9e5a499b4952ab555a aaaaaa	-	7.12296784359
447	$\mathbb{B}_{445} \circ \eta_2 \circ \eta_5$	7ffffff8003ff01c399c78698cf07a4cc7c29722768c3d8c63c0f 30e6d32d4b64d8b4de27721f4a4cc7a52cd9e5a499b4952ab555a aaaaaa	-	7.19410239793
448	$\mathbb{B}_{445} \circ \eta_2 \circ \eta_2 \circ \eta_5$	fffffff0007fe0387338f0d319e0f4998f852e44ed187b18c781e 61cda65a96c9b169bc4ee43e94998f4a59b3cb4933692a556aab5 555554	-	7.168
449	$\mathbb{B}_{445} \circ \eta_2 \circ \eta_2 \circ \eta_5 \circ \eta_6$	fffffff0007fe0387338f0d319e0f4998f852e44ed187b18c781e 61cda65a96c9b169bc4ee43e94998f4a59b3cb4933692a556aab5 555554	6.5218	7.1428925737
450	$\mathbb{B}_{451} \circ \eta_4$	3ffffffc000ffe01e1a70db06de07e34b5a46c93661e499938678 3733337296693998e5a6738c6e1f0f256a5c6b1cb61a5aa554aaa d555555	-	7.01517356059
451	$\mathbb{B}_{451}^{28,14,11,8,4}$	7ffffff8001ffc03c34e1b60dbc0fc696b48d926cc3c933270cf0 6e6666e52cd27331cb4ce718dc3e1e4ad4b8d6396c34b54aa9555 aaaaaaa	-	7.17362629611
452	$\mathbb{B}_{451} \circ \eta_2$	fffffff0003ff807e1f0fc1a7c0f349b1e89ce1c6cc6693e24f0d 8726369cb4e253866cc6da4d885b18f34ad61ad4b5a56a9552aab 5555554	-	7.04399393187
453	$\mathbb{B}_{453}^{28,14,11,8,4}$	1fffffffe0007ff00fc0fcf03c66679933387859e0db07938c7987 8c79396c96996c9396b1ca59e969333996666d2b4d4ad4ab556aa a5555555	-	7.04991754844
454	$\mathbb{B}_{453} \circ \eta_2$	3ffffffc000ffe01f81f9e078cccf326670f0b3c1b60f2718f30f 18f272d92d32d9272d6394b3d2d266732cccda569a95a956aad55 4aaaaaaa	-	7.04381108605
455	$\mathbb{B}_{455}^{28,14,11,8,4,4}$	7ffffff8001ffc03c3781e199cc785bc31ccc6696607f094f216d 8cc724cd8e172c1d2a5661e64cc934b85a4c999695a34b54aa955 5aaaaaaa	-	7.10791045801
456	$\mathbb{B}_{455} \circ \eta_2$	fffffff0003ff80786f03c33398f0b7863998cd2cc0fe129e42db 198e499b1c2e583a54acc3cc99926970b499332d2b4696a9552aa b5555554	-	7.01159967629
457	$\mathbb{B}_{457}^{28,14,11,8,4,4,3}$	1fffffffe0007ff00f0f9cce60f3e19e4f24f6316e163391cb370b c70b670b6d0b730db9327a47b274e34e59a534a64cd94b4ab556a aa5555555	-	7.01966254369
458	$\mathbb{B}_{461} \circ \eta_4 \circ \eta_4 \circ \eta_4$	3ffffffc000ffe01e1ec66f03f834b34b1b598dce1e729cda34d8 d86691e658d8c378c9f2696c8d98393c33c35ab52e64e96956aad 554aaaaaa	-	7.11884884273
459	$\mathbb{B}_{461} \circ \eta_0$	7ffffff0003ff80787b19bc0fe0d2cd2c6d66373879ca7368d363 619a479963630de327c9a5b23660e4f0cf0d6ad4b993a5a55aab5 552aaaaaa	-	7.25635461872
460	$\mathbb{B}_{461} \circ \eta_4$	fffffff0003ff80787b19bc0fe0d2cd2c6d66373879ca7368d363 619a479963630de327c9a5b23660e4f0cf0d6ad4b993a5a55aab5 552aaaaaa	-	7.39549839228
461	$\mathbb{B}_{461}^{28,14,11,8,4,4}$	1fffffffe0007ff00f0f633781fc1a59a58dacc6e70f394e6d1a6c 6c3348f32c6c61bc64f934b646cc1c9e19e1ad5a973274b4ab556 aaa5555555	-	7.53941393501
462	$\mathbb{B}_{461} \circ \eta_1$	3ffffffc000ffe01e1ec66f03f834b34b1b598dce1e729cda34d8 d86691e658d8c378c9f2696c8d98393c33c35ab52e64e96956aad 554aaaaaab	-	7.38509445713
463	$\mathbb{B}_{461} \circ \eta_1 \circ \eta_1$	7ffffff8001ffc03c3d8cde07f069669636b31b9c3ce539b469b1 b0cd23ccb1b186f193e4d2d91b3072786786b56a5cc9d2d2ad55a aa95555557	-	7.25248663644
464	$\mathbb{B}_{461} \circ \eta_1 \circ \eta_1 \circ \eta_1$	fffffff0003ff80787b19bc0fe0d2cd2c6d66373879ca7368d363 619a479963630de327c9a5b23660e4f0cf0d6ad4b993a5a55aab5 552aaaaaaf	-	7.13467656416

Table B.13 A list of binary sequences with record merit factor values and lengths between 465 and 485

n	Class	Record sequence in HEX	Old MF	New MF
465	$\mathbb{B}_{461} \circ \eta_1 \circ \eta_1 \circ \eta_1 \circ \eta_1$	1ffffffe0007ff00f0f633781fc1a59a58dacc6e70f394e6d1a6c 6c3348f32c6c61bc64f934b646cc1c9e19e1ad5a973274b4ab556 aaa5555555f	–	7.08469855832
466	$\mathbb{B}_{461} \circ \eta_1 \circ \eta_1 \circ \eta_1 \circ \eta_1 \circ \eta_2$	3ffffffc000ffe01e1ec66f03f834b34b1b598dce1e729cda34d8 d86691e658d8c378c9f2696c8d98393c33c35ab52e64e96956aad 554aaaaaabe	–	7.09057663423
467	$\mathbb{B}_{467}^{30,15,12,10,6,6,6}$	7ffffffe0003ffc00fc0fc1e0c39e19ce1c725e1399933cc8db83 6691e6691e635b8dccb3199b16872496c9969b4d694ad4ad54aab 5556aaaaaaa	–	7.0075509286
468	$\mathbb{B}_{469} \circ \eta_4$	fffffffe0003ffc03f03f19c781f870e6631e8c63ccc78f138f1c c9c96ce1c9cc92db12da4ccb64de93666d25a95a4992b52b54aab 5556aaaaaaa	–	7.16795392067
469	$\mathbb{B}_{469}$	1fffffffc0007ff807e07e338f03f0e1ccc63d18c7998f1e271e3 99392d9c3939925b625b49996c9bd26ccda4b52b493256a56a955 6aaad5555555	–	7.3743127263
470	$\mathbb{B}_{469} \circ \eta_1$	3ffffffff8000fff00fc0fc671e07e1c3998c7a318f331e3c4e3c7 32725b38727324b6c4b69332d937a4d99b496a569264ad4ad52aa d555aaaaaaab	–	7.25165780316
471	$\mathbb{B}_{469} \circ \eta_1 \circ \eta_2$	7fffffff0001ffe01f81f8ce3c0fc3873318f4631e663c789c78e 64e4b670e4e6496d896d2665b26f49b33692d4ad24c95a95aa555 aaab55555556	–	7.12719270064
472	$\mathbb{B}_{469} \circ \eta_1 \circ \eta_2 \circ \eta_2$	fffffffe0003ffc03f03f19c781f870e6631e8c63ccc78f138f1c c9c96ce1c9cc92db12da4ccb64de93666d25a95a4992b52b54aab 5556aaaaaaac	–	7.07340614681
473	$\mathbb{B}_{473}^{29,15,11,9,6,6,4}$	1fffffff0001ffc01f81f2661c3ec1b324ce61e3c363633c396cd 0edc2e4e42dc4bcc792d327272d25a64ce331ac52da6635a95aad 55aaab5555555	–	7.38087226181
474	$\mathbb{B}_{473} \circ \eta_2$	3ffffffe0003ff803f03e4cc387d8366499cc3c786c6c67872d9a 1db85c9c85b89798f25a64e4e5a4b4c99c66358a5b4cc6b52b55a ab5556aaaaaaa	–	7.31129189717
475	$\mathbb{B}_{473} \circ \eta_2 \circ \eta_5$	7ffffffe0003ff803f03e4cc387d8366499cc3c786c6c67872d9a 1db85c9c85b89798f25a64e4e5a4b4c99c66358a5b4cc6b52b55a ab5556aaaaaaa	–	7.24410839273
476	$\mathbb{B}_{473} \circ \eta_2 \circ \eta_2 \circ \eta_5$	fffffffc0007ff007e07c99870fb06cc9339878f0d8d8cf0e5b34 3b70b9390b712f31e4b4c9c9cb49699338cc6b14b6998d6a56ab5 56aaad5555554	–	7.07430997877
477	$\mathbb{B}_{477}^{29,15,11,9,6,6,4}$	1fffffff0001ffc01f81e6f03c78c3e61e463e0f3333cccb4cb78 c67878c2c96966c970cf0ccd33334a526e5a652c96d2b465a95aa d55aaab5555555	–	7.27859884837
478	$\mathbb{B}_{477} \circ \eta_2$	3ffffffe0003ff803f03cde078f187cc3c8c7c1e66679996996f1 8cf0f18592d2cd92e19e199a666694a4dcb4ca592da568cb52b55 aab5556aaaaaaa	–	7.18548336373
479	$\mathbb{B}_{479}^{29,15,11,9,6,6,4}$	7ffffffc0007ff007e07e72331c993c61b619d25ccb0f03c4b31b 0c3c1b4c394b4d3933c4b52d3cc8709963964b19c933726a56a55 2aa5554aaaaaaa	–	7.12063186643
480	$\mathbb{B}_{479} \circ \eta_1$	fffffff8000ffe00fc0fce466393278c36c33a4b9961e07896636 1878369872969a7267896a5a7990e132c72c96339266e4d4ad4aa 554aaa95555555	–	7.05882352941
481	$\mathbb{B}_{481}^{29,15,11,9,6,6,4}$	1fffffff0001ffc01f81fc27e1b6cc92f0cda6627a52f2d33b273 9a49e2d33c258e19363133c343e162661ccb438cc71a562d5a95a ad55aaab5555555	–	7.26636306533
482	$\mathbb{B}_{481} \circ \eta_1$	3ffffffe0003ff803f03f84fc36d9925e19b4cc4f4a5e5a6764e7 3493c5a6784b1c326c62678687c2c4cc399687198e34ac5ab52b5 5aab5556aaaaaab	–	7.15944530046
483	$\mathbb{B}_{481} \circ \eta_1 \circ \eta_2$	7ffffffc0007ff007e07f09f86db324bc3369989e94bcb4cec9ce 69278b4cf0963864d8c4cf0d0f858998732d0e331c6958b56a56a b556aaad5555556	–	7.17503229378
484	$\mathbb{B}_{481} \circ \eta_1 \circ \eta_2 \circ \eta_6$	7ffffffc0007ff007e07f09f86db324bc3369989e94bcb4cec9ce 69278b4cf0963864d8c4cf0d0f858998732d0e331c6958b56a56a b556aaad5555556	–	7.08406919076
485	$\mathbb{B}_{481} \circ \eta_1 \circ \eta_2 \circ \eta_6 \circ \eta_6$	7ffffffc0007ff007e07f09f86db324bc3369989e94bcb4cec9ce 69278b4cf0963864d8c4cf0d0f858998732d0e331c6958b56a56a b556aaad5555556	–	7.11165195308

Table B.14 A list of binary sequences with record merit factor values and lengths between 486 and 505

n	Class	Record sequence in HEX	Old MF	New MF
486	$\mathbb{B}_{487} \circ \eta_4$	3fffffff0007ff807e07cf038f8c79c1e60f8799b0dc9f30f3391 b0e730f48f4b364b1b9334b358dcb199694a65ad96c9492b4d6a5 6a9556aab5555555	-	7.14230420321
487	$\mathbb{B}_{487}^{30,13,12,8,6,6,5}$	7ffffffe000fff00fc0f9e071f18f383cc1f0f3361b93e61e6723 61ce61e91e966c9637266966b1b96332d294cb5b2d9292569ad4a d52aad556aaaaaaa	-	7.19784522003
488	$\mathbb{B}_{487} \circ \eta_1$	fffffffc001ffe01f81f3c0e3e31e707983e1e66c3727cc3cce46 c39cc3d23d2cd92c6e4cd2cd6372c665a52996b65b2524ad35a95 aa555aaad5555555	-	7.19033816425
489	$\mathbb{B}_{489}^{30,15,12,8,6,4}$	1ffffffff8000fff00fc2c7e07878e1ec1f8c6d927c99626679339 86d24f264e634e3c6993396662798d639c6c95ac5a49696a56c2d 4ab554aaa95555555	-	7.20070464948
490	$\mathbb{B}_{489} \circ \eta_2$	3fffffff0001ffe01f858fc0f0f1c3d83f18db24f932c4ccf2673 0da49e4c9cc69c78d32672ccc4f31ac738d92b58b492d2d4ad85a 956aa95552aaaaaaa	-	7.09305760709
491	$\mathbb{B}_{489} \circ \eta_2 \circ \eta_1$	7ffffffe0003ffc03f0b1f81e1e387b07e31b649f2658999e4ce6 1b493c99398d38f1a64ce59989e6358e71b256b16925a5a95b0b5 2ad552aaa55555555	-	7.08519955328
492	$\mathbb{B}_{495} \circ \eta_0 \circ \eta_4$	fffffff0001ffc01f81f06d836cc2fc30d9c3b659b4e4ed38d8da 4e72666961e66726c78d8db0ec6c39863b498d34af4ce358e5295 a954aa95552aaaaaa	-	7.07045215563
493	$\mathbb{B}_{495} \circ \eta_4 \circ \eta_4$	1fffffff0001ffc01f81f06d836cc2fc30d9c3b659b4e4ed38d8d a4e72666961e66726c78d8db0ec6c39863b498d34af4ce358e529 5a954aa95552aaaaaa	-	7.0810220254
494	$\mathbb{B}_{495} \circ \eta_1 \circ \eta_3 \circ \eta_3$	3ffffff8000ffe00fc0f836c1b6617e186ce1db2cda72769c6c6d 27393334b0f3339363c6c6d876361cc31da4c69a57a671ac7294a d4aa554aaa95555555	-	7.07637882039
495	$\mathbb{B}_{495}^{29,15,11,9,6,6,4}$	7ffffffc0007ff007e07c1b60db30bf0c3670ed966d393b4e3636 939c999a587999c9b1e3636c3b1b0e618ed2634d2bd338d6394a5 6a552aa5554aaaaaaa	-	7.01233472612
496	$\mathbb{B}_{495} \circ \eta_1$	fffffff8000ffe00fc0f836c1b6617e186ce1db2cda72769c6c6d 27393334b0f3339363c6c6d876361cc31da4c69a57a671ac7294a d4aa554aaa95555555	-	7.01459854015
497	$\mathbb{B}_{495} \circ \eta_1 \circ \eta_5$	1fffffff8000ffe00fc0f836c1b6617e186ce1db2cda72769c6c6 d27393334b0f3339363c6c6d876361cc31da4c69a57a671ac7294 ad4aa554aaa95555555	-	7.01730113636
498	$\mathbb{B}_{497}^{27,15,11,11,7,7,7,7} \circ \eta_2$	3ffffff8000ffe003f80fe03c9c86321b0e61a7387983ccb724e6 696c3d26636670b4e1e66c723ccb59a5b27966d397365c9cb56ad 5ab556aad555aaaaaaa	-	7.00378424174
499	$\mathbb{B}_{499}^{29,15,11,9,6,6,4}$	7ffffffc0007ff007e07d87431ed839c0e4f86ce46978d8f49b67 2d9b19b4c9cc399398f2639c2d8da1e46ce5ac6d49b58e934258a 56a552aa5554aaaaaaa	-	7.16797167367
500	$\mathbb{B}_{499} \circ \eta_1$	fffffff8000ffe00fc0fb0e863db07381c9f0d9c8d2f1b1e936ce 5b3633699398732731e4c7385b1b43c8d9cb58da936b1d2684b14 ad4aa554aaa95555555	-	7.04542892571
501	$\mathbb{B}_{501}^{30,15,12,8,6,4}$	1fffffff8000fff00fc1e4f30f865f207e0f34da63598c5e0d939 bc360e4d8ccc9ce4a72d1939ca5ec99f261cf34a56a35e694b34e 5ad4ab554aaa95555555	-	7.16572456321
502	$\mathbb{B}_{501} \circ \eta_2$	3fffffff0001ffe01f83c9e61f0cbe40fc1e69b4c6b318bc1b273 786c1c9b199939c94e5a327394bd933e4c39e694ad46bcd29669c b5a956aa95552aaaaaaa	-	7.27872451043
503	$\mathbb{B}_{501} \circ \eta_2 \circ \eta_5$	7ffffffff0001ffe01f83c9e61f0cbe40fc1e69b4c6b318bc1b273 786c1c9b199939c94e5a327394bd933e4c39e694ad46bcd29669c b5a956aa95552aaaaaaa	-	7.39489682586
504	$\mathbb{B}_{501} \circ \eta_2 \circ \eta_2 \circ \eta_5$	fffffffe0003ffc03f0793cc3e197c81f83cd3698d663178364e6 f0d83936333273929cb464e7297b267c9873cd295a8d79a52cd39 6b52ad552aaa55555554	-	7.21964529332
505	$\mathbb{B}_{501} \circ \eta_2 \circ \eta_2 \circ \eta_1 \circ \eta_5$	1fffffffc0007ff807e0f27987c32f903f079a6d31acc62f06c9c de1b0726c6664e7253968c9ce52f64cf930e79a52b51af34a59a7 2d6a55aaa5554aaaaaaa9	-	7.09980512249

Table B.15 A list of binary sequences with record merit factor values and lengths between 506 and 527

n	Class	Record sequence in HEX	Old MF	New MF
506	$\mathbb{B}_{501} \circ \eta_2 \circ \eta_2 \circ \eta_1 \circ \eta_1 \circ \eta_5$	3fffffff8000fff00fc1e4f30f865f207e0f34da63598c5e0d939 bc360e4d8ccc9ce4a72d1939ca5ec99f261cf34a56a35e694b34e 5ad4ab554aaa955555553	-	7.05994595489
507	$\mathbb{B}_{507}^{29,15,11,9,6,6,4}$	7ffffffc0007ff007e07c1b30db303f0e1c9c6d99996e58f4f0e7 863948cf183592cdc1b65a6d2c2d86e19998e49c96d2b5338d339 4a56a552aa5554aaaaaaa	-	7.23144657627
508	$\mathbb{B}_{507} \circ \eta_1$	fffffff8000ffe00fc0f83661b6607e1c3938db3332dcb1e9e1cf 0c72919e306b259b836cb4da585b0dc33331c9392da56a671a672 94ad4aa554aaa95555555	-	7.10293955741
509	$\mathbb{B}_{509}^{26,13,12,9,7,6,6,4,2}$	1ffffff8003ffc01fc0fc3078f266781f81f393278cda4cb1c78d 2649c6963ccccd2786d8e63c96db0ce1cc9633935a95a96663496 b2d4ad5aad552aa9555555	-	7.03489193005
510	$\mathbb{B}_{511} \circ \eta_4$	3ffffffe0007ffc007f01fc079b43c5b0ce72761e437872d2cccb 4e4db61b18ccc9b1a71ce4f0ccc3c36972e5a76364cb1ed2f196a d5ab56aad556aaa5555555	-	7.00889248181
511	$\mathbb{B}_{511}^{27,15,11,11,7,7,7,7}$	7ffffff0001ffc007f01fc079258d264c3613c61b1b1c8d8665a5 8c78c6c4b4b3c3c4e4da4d8786658dc9393964b1634c670d871a5 4a952a554aa95552aaaaaa	-	7.22166602135
512	$\mathbb{B}_{511} \circ \eta_1$	ffffffe0003ff800fe03f80f24b1a4c986c278c3636391b0ccb4b 18f18d8969678789c9b49b0f0ccb1b927272c962c698ce1b0e34a 952a54aa9552aaa5555555	-	7.09417622862
513	$\mathbb{B}_{513}^{29,14,13,10,7,7,7,7,2}$	1fffffff0003ffe007f01fc078c6f0cc799c9e1b36b4e3ca52d61 c7a49ce66679966664d8e16da7c3e0d24f0731a58d996ccb46c96 ad5ab56aa5552aab5555555	-	7.11729229771
514	$\mathbb{B}_{513} \circ \eta_2$	3ffffffe0007ffc00fe03f80f18de198f3393c366d69c794a5ac3 8f4939ccccf32cccc9b1c2db4f87c1a49e0e634b1b32d9968d92d 5ab56ad54aaa5556aaaaaaa	-	7.02014136153
515	$\mathbb{B}_{517} \circ \eta_0$	7fffffff0001ffe007e07c3c1d83e493b670b658c79c3a5c3cdb1 32c66c70f263672d24e64f3138cb487b49a4d863d263b1c6b5894 b4a56a556aa95552aaaaaaa	-	7.00874689498
516	$\mathbb{B}_{517} \circ \eta_4$	ffffffff0001ffe007e07c3c1d83e493b670b658c79c3a5c3cdb1 32c66c70f263672d24e64f3138cb487b49a4d863d263b1c6b5894 b4a56a556aa95552aaaaaaa	-	7.01190350785
517	$\mathbb{B}_{517}^{27,15,11,11,7,7,7,7}$	1ffffffc0007ff001fc07f01e1e0f0f270cf03e19cc3c664d86d8 7c63932d999ccd999c33926d69c69ce66d2cd9a52b4cb634b4a5a 5ab56ad5aab556aaad555555	-	7.12922756855
518	$\mathbb{B}_{517} \circ \eta_2$	3ffffff8000ffe003f80fe03c3c1e1e4e19e07c339878cc9b0db0 f8c7265b33399b33386724dad38d39ccda59b34a56996c69694b4 b56ad5ab556aad555aaaaaaa	-	7.20719849584
519	$\mathbb{B}_{519}^{27,15,11,11,7,7,7,7}$	7ffffff0001ffc007f01fc07c36493e1b64d25b0f87cc793ccf98 da7271b0f1999992d3927278d9accb1a4ca5ad3870c6396b1c634 a54a952a554aa95552aaaaa	-	7.18717647687
520	$\mathbb{B}_{519} \circ \eta_1$	ffffffe0003ff800fe03f80f86c927c36c9a4b61f0f98f2799f31 b4e4e361e3333325a724e4f1b3599634994b5a70e18c72d638c69 4a952a54aa9552aaa555555	-	7.06078963860
521	$\mathbb{B}_{521}^{27,15,11,11,7,7,7,7}$	1ffffffc0007ff001fc07f01b2c4996c7c36d0f60f358f0e63639 c9d878cce1e4f4e5a4cc969d8d927264b49f34a74bc72d6c798ec 31ab56ad5aab556aaad55555	-	7.01760599793
522	$\mathbb{B}_{521} \circ \eta_2$	3ffffff8000ffe003f80fe03658932d8f86da1ec1e6b1e1cc6c73 93b0f199c3c9e9cb49992d3b1b24e4c9693e694e978e5ad8f31d8 6356ad5ab556aad555aaaaaa	-	7.06759350521
523	$\mathbb{B}_{521} \circ \eta_2 \circ \eta_5$	7ffffff8000ffe003f80fe03658932d8f86da1ec1e6b1e1cc6c73 93b0f199c3c9e9cb49992d3b1b24e4c9693e694e978e5ad8f31d8 6356ad5ab556aad555aaaaaa	-	7.11833133816
524	$\mathbb{B}_{521} \circ \eta_2 \circ \eta_2 \circ \eta_5$	fffffff0001ffc007f01fc06cb1265b1f0db43d83cd63c398d8e7 2761e3338793d39693325a763649c992d27cd29d2f1cb5b1e63b0 c6ad5ab56aad55aaab555555	-	7.08181161663
525	$\mathbb{B}_{521} \circ \eta_2 \circ \eta_2 \circ \eta_5 \circ \eta_6$	fffffff0001ffc007f01fc06cb1265b1f0db43d83cd63c398d8e7 2761e3338793d39693325a763649c992d27cd29d2f1cb5b1e63b0 c6ad5ab56aad55aaab555555	-	7.04634931997
526	$\mathbb{B}_{521} \circ \eta_2 \circ \eta_2 \circ \eta_1 \circ \eta_1 \circ \eta_5$	3ffffffc0007ff001fc07f01b2c4996c7c36d0f60f358f0e63639 c9d878cce1e4f4e5a4cc969d8d927264b49f34a74bc72d6c798ec 31ab56ad5aab556aaad555555	-	7.03401637260
527	$\mathbb{B}_{517} \circ \eta_2 \circ \eta_2 \circ \eta_1 \circ \eta_1 \circ \eta_2 \circ \eta_5 \circ \eta_6 \circ \eta_6 \circ \eta_5 \circ \eta_5$	67ffffff8000ffe003f80fe03c3c1e1e4e19e07c339878cc9b0db 0f8c7265b33399b33386724dad38d39ccda59b34a56996c69694b 4b56ad5ab556aad555aaaaaaa6	-	7.08239404294

Table B.16 A list of binary sequences with record merit factor values of lengths 573, 1006, 1007, 1008, 1009 and 1010

n	Class	Record sequence in HEX	Old MF	New MF
573	$\mathbb{B}_{573}^{27,15,11,11,7,7,7,7}$	1ffffffc0007ff001fc07f01e7c2787b0e1e5a7e3cf0f64bc39ce 372d9b399ac9b26e333333246318c199319c3724d92d0e74b4d25 61e5a4b16962d65ab56ad5aab556aaad55555	-	6.82937432399
1006	$\mathbb{B}_{1009} \circ \eta_0 \circ \eta_4$	3fffffffff000007fff0001ff801fe01e03fc0be03e9c60fb70e1 f039b0e3c762c73ca479ccc6f0d9c9327c97879938d8dcd937330 e1793867d8761e19f1b59f1b59a5a769d669397a4b33739cdc9c9 3996978d6338d9cb46ccd96e0d36c276d24b192b5a4b714a6d852 a50ad52a5aa55aa955aaab5556aaaab55555555	-	6.35677047348
1007	$\mathbb{B}_{1009} \circ \eta_0$	7ffffffffe00000fffe0003ff003fc03c07f817c07d38c1f6e1c3 e07361c78ec58e7948f3998de1b39264f92f0f3271b1b9b26e661 c2f270cfb0ec3c33e36b3e36b34b4ed3acd272f49666e739b9392 732d2f1ac671b3968d99b2dc1a6d84eda4963256b496e294db0a5 4a15aa54b54ab552ab5556aaad55556aaaaaaaa	-	6.41941303825
1008	$\mathbb{B}_{1009} \circ \eta_4$	fffffffffe00000fffe0003ff003fc03c07f817c07d38c1f6e1c3 e07361c78ec58e7948f3998de1b39264f92f0f3271b1b9b26e661 c2f270cfb0ec3c33e36b3e36b34b4ed3acd272f49666e739b9392 732d2f1ac671b3968d99b2dc1a6d84eda4963256b496e294db0a5 4a15aa54b54ab552ab5556aaad55556aaaaaaaa	-	6.41811107180
1009	$\mathbb{B}_{1009}^{39,21,15,15,10,10,8,8,4}$	1ffffffffffc00001fffc0007fe007f80780ff02f80fa7183edc38 7c0e6c38f1d8b1cf291e7331bc36724c9f25e1e64e3637364dccc 385e4e19f61d87867c6d67c6d66969da759a4e5e92ccdce737272 4e65a5e358ce3672d1b3365b834db09db492c64ad692dc529b614 a942b54a96a956aa556aaad555aaaaad55555555	-	6.41690827959
1010	$\mathbb{B}_{1009} \circ \eta_2$	3ffffffff800003fff8000ffc00ff00f01fe05f01f4e307db870 f81cd871e3b1639e523ce663786ce4993e4bc3cc9c6c6e6c9b998 70bc9c33ec3b0f0cf8dacf8dacd2d3b4eb349cbd2599b9ce6e4e4 9ccb4bc6b19c6ce5a3666cb7069b613b69258c95ad25b8a536c29 52856a952d52ad54aad555aaab55555aaaaaaaaa	-	6.36726796080