Universidade Estadual de Campinas
Instituto de Computação

Ana Clara Zoppi Serpa

Study of Diffusion and MDS Matrices in Symmetric Block Ciphers

Estudo de Difusão e Matrizes MDS em Cifras de Bloco Simétricas

CAMPINAS
2023

Ana Clara Zoppi Serpa

Study of Diffusion and MDS Matrices in Symmetric Block Ciphers

Estudo de Difusão e Matrizes MDS em Cifras de Bloco Simétricas

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestra em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Ricardo Dahab
Co-supervisor/Coorientador: Dr. Jorge Nakahara Jr.

Este exemplar corresponde à versão final da Dissertação defendida por Ana Clara Zoppi Serpa e orientada pelo Prof. Dr. Ricardo Dahab.

CAMPINAS
2023

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

	Serpa, Ana Clara Zoppi, 1999-
Se67e	Estudo de difusão e matrizes MDS em cifras de bloco simétricas / Ana Clara Zoppi Serpa. – Campinas, SP : [s.n.], 2023.
	Orientador: Ricardo Dahab.
	Coorientador: Jorge Nakahara Junior.
	Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.
	1. Criptografia. I. Dahab, Ricardo, 1957-. II. Nakahara Junior, Jorge. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações Complementares

Título em outro idioma: Study of diffusion and MDS matrices in symmetric block ciphers
Palavras-chave em inglês:
Cryptography
Área de concentração: Ciência da Computação
Titulação: Mestra em Ciência da Computação
Banca examinadora:
Ricardo Dahab [Orientador]
Daniel Panario
Marcos Antonio Simplicio Junior
Thais Bardini Idalino
Data de defesa: 03-08-2023
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0009-0009-9868-9769
- Currículo Lattes do autor: http://lattes.cnpq.br/5467174242604206

Universidade Estadual de Campinas
Instituto de Computação

Ana Clara Zoppi Serpa

Study of Diffusion and MDS Matrices in Symmetric Block Ciphers

Estudo de Difusão e Matrizes MDS em Cifras de Bloco Simétricas

Campinas, 03 de agosto de 2023

Dedicated to everyone who inspired me, each in their own unique way.

Acknowledgements

This work was conducted with the support of QuintoAndar, through a partnership with the Institute of Computing to provide support to postgraduate students in their research projects. Similarly, this work is also the outcome of a collaboration between the Institute of Computing (through FUNCAMP – Foundation for the Development of Unicamp) and Samsung Eletrônica da Amazônia Ltda., within the framework of the "Post-Quantum Cryptography" project. Therefore, I express my gratitude to both companies for their funding support. This study was also financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. I express my gratitude to CAPES as well.

In regard to the individuals with whom I have interacted and collaborated, there are numerous people whom I would like to express my appreciation to. Each one of them has, in various ways, supported me throughout the development of this work.

Firstly, I would like to acknowledge my advisors, Prof. Dr. Ricardo Dahab and Dr. Jorge Nakahara Jr. Their expertise, guidance, patience, and unwavering commitment to my work have allowed me to delve into Cryptology beyond what I had ever imagined. I would also like to express my gratitude to the other members of the Laboratory of Security and Cryptography (LASCA) with whom I have collaborated. Through these collaborations, I have not only gained invaluable knowledge but have also shared memorable moments. Among them, I would like to highlight Prof. Dr. Julio López, Caio Teixeira, Tomás Silva, Félix Rodrigues, Jheyne Ortiz, Décio Gazzoni Filho, Giuliano Sider, and Hayato Fujii.

Furthermore, I thank the staff members of the Institute of Computing, ranging from the administrative personnel in both the undergraduate and postgraduate departments to the security personnel and janitorial staff, highlighting Wilson and Priscilla. All these contributions made the Institute of Computing always a peaceful and comfortable space for my studies, a second home to me. Additionally, I extend my appreciation to all the professors at the Institute of Computing for all I have learned with their courses and exchange of ideas during our time together.

To my colleagues at Amazon, I would like to convey my gratitude for the good moments, lessons learned, and encouragement they have provided. The knowledge and skills I acquired at Amazon as a Software Development Engineer have enabled me to approach problem-solving in my master's degree journey from a fresh perspective. Although these two domains of work are distinct, the maturity and insights gained from each have been paramount for my success.

Furthermore, I am deeply grateful to my parents, Rita Aparecida Zoppi and Alberto Luiz Serpa, and to other family members for their unwavering efforts in ensuring that I had access to the best educational opportunities and the time and space necessary to pursue my studies. Lastly, but by no means least, I would like to express my appreciation to my friends for their constant companionship. Each of them has made my days pleasant and fun, always bringing light to my life. I express special thanks to Esther, Flávia, Sarah, Thiago, Gustavo, Andreis, Miguel, Erick, Carolina, Henrique, Pedro and Guilherme.

Resumo

Esta dissertação de mestrado apresenta um estudo abrangente da propriedade de difusão em algoritmos criptográficos simétricos, especificamente o AES e o DES, com análise das Formas Normais Algébricas (ANFs) de suas caixas de substituição, tabelas de distribuição de diferenças (DDTs) e tabelas de aproximação linear (LATs), discutindo vulnerabilidades e resistência desses algoritmos contra ataques de criptoanálise diferencial e linear. Além disso, um modelo do algoritmo DES como um grafo para análise de sua difusão é exposto, o qual poderia ser estendido para outros cifradores de bloco e funções hash, aprimorando a compreensão de suas características de difusão.

Apresentamos também o primeiro catálogo de matrizes MDS (Maximum Distance Separable) com aproximadamente 150 matrizes, abrangendo 22 anos de literatura, com suas contagens de operações **xtime** e **xor**, oferecendo um recurso valioso para o *design* de algoritmos criptográficos. Além disso, a pesquisa identifica matrizes não MDS previamente relatadas incorretamente como MDS na literatura, retificando possíveis interpretações equivocadas e contribuindo para a precisão de futuras análises.

Para facilitar o estudo e a reprodução dos resultados apresentados, todos os códigos e implementações estão disponíveis abertamente no GitHub, garantindo transparência e permitindo pesquisas adicionais no campo.

A dissertação também explora o uso de Algoritmos Genéticos (GA) para construção de matrizes MDS como uma abordagem inovadora. Várias sugestões de melhoria são fornecidas com base nos resultados de nossos experimentos com GA, os quais, embora não tenham produzido resultados significativamente inovadores, apresentam informações valiosas para futuras direções de pesquisa e aprimoramentos potenciais na construção de matrizes MDS.

Matrizes MDS estão altamente ligadas ao design de algoritmos criptográficos, particularmente cifras de bloco da família *Substitution Permutation Network*. Os projetistas muitas vezes precisam escolher uma matriz MDS para ser usada em seu algoritmo. Portanto, uma aplicação do nosso trabalho reside no fato de que novas abordagens para obtenção de matrizes MDS facilitam o processo de design.

A outra aplicação está relacionada a reduzir o custo computacional das implementações em software e/ou hardware o máximo possível. A técnica de construção de matrizes MDS proposta tenta não apenas garantir a propriedade MDS, mas também reduzir as contagens de **xor** e **xtime** das matrizes. Quanto menores as contagens de **xor** e **xtime**, menor será o custo computacional da matriz, contribuindo para uma implementação de baixo custo computacional.

Em resumo, esta dissertação oferece uma análise abrangente da propriedade de difusão e sua relevância para algoritmos criptográficos simétricos, apresenta o primeiro catálogo extenso de matrizes MDS da literatura, e fornece implementações de Algoritmos Genéticos para busca de matrizes MDS e sugestões de melhorias. Essas contribuições visam avançar a compreensão, análise e projeto de algoritmos criptográficos simétricos seguros. Além disso, os códigos em Python desenvolvidos neste trabalho podem facilitar a avaliação de matrizes (eles podem ser usados para verificar se uma determinada matriz é MDS e informar suas contagens de **xor** e **xtime**), a avaliação da propriedade de difusão de algoritmos, auxiliar outros pesquisadores no estudo deste tema, ou serem utilizados para o ensino do assunto.

Abstract

This Master thesis presents a comprehensive study of the diffusion property in symmetric cryptographic algorithms, specifically AES and DES, with analysis of the Algebraic Normal Forms (ANFs) of their substitution boxes, DDTs (difference distribution tables) and LATs (linear approximation tables), discussing vulnerabilities and resistance of these algorithms against differential and linear cryptanalysis attacks. Additionally, a model of the DES algorithm as a graph for diffusion analysis is explained, which could be extended to other block ciphers and hash functions, enhancing the understanding of their diffusion characteristics.

We also present the first catalogue of ≈ 150 MDS (Maximum Distance Separable) matrices covering 22 years of literature with their **xtime** and **xor** counts, offering a valuable resource for cryptographic algorithm design. Furthermore, the research identifies non-MDS matrices previously misreported as MDS in the literature, rectifying potential misinterpretations and contributing to the accuracy of future cryptographic analysis.

To facilitate the study and reproduction of the presented results, all codes and implementations are made openly available on GitHub, ensuring transparency and enabling further research in the field.

The thesis also explores the use of Genetic Algorithms (GA) for MDS matrix construction as a novel approach. Various suggestions for improvement are provided based on the outcomes of our GA experiments, which, although not yielding significantly novel results, present valuable insights for future research directions and potential enhancements in the construction of MDS matrices.

MDS matrices are highly linked to cryptographic algorithms design, particularly block ciphers of the Substitution Permutation Network family. The designers often must choose a MDS matrix to be used in their algorithm. Therefore, one application of our work lies in the fact that new approaches to obtain MDS matrices facilitate this design process.

The other application is related to lowering the computational cost of the software and/or hardware implementations as much as possible. Our proposed MDS matrix construction technique attempts not only to ensure MDS property but also to reduce the **xor** and **xtime** counts of the matrices. The lower the **xor** and **xtime** counts, the lower the computational cost of the matrix, contributing to a low cost implementation.

Overall, this thesis offers a comprehensive analysis of the diffusion property and its relevance for symmetric cryptographic algorithms, presents a comprehensive MDS matrix catalogue, and provides code implementations of Genetic Algorithms for MDS matrix search and suggestions for improvement. These contributions aim to advance the understanding, analysis, and design of secure symmetric cryptographic algorithms. Furthermore, the Python codes developed in this work may ease the evaluation of matrices (they can be used to check if a given matrix is MDS and output its **xor** and **xtime** cost), the evaluation of a cipher's diffusion property, help other researchers in the study of this topic or be used for teaching.

List of Figures

List of Tables

List of Algorithms

Contents

Introduction

In this work, we study the complete diffusion property in symmetric block ciphers, with special attention to the application of MDS (*Maximum Distance Separable*) codes and their usage in matrix construction throughout the years. We provide a catalogue of ≈ 200 MDS matrices that are present in the literature at the time of this work's writing, and their respective computational costs in number of necessary **xtime** and **xor** operations for matrix multiplication. We also briefly investigate MDS matrix construction through *genetic algorithms.*

Motivation

Complete diffusion is a necessary requirement for security against differential and linear cryptanalysis. As mathematical components that ensure complete diffusion is achieved within few iterations of a cipher's round function, MDS matrices are highly relevant for modern block cipher design. However, there are many challenges regarding MDS matrix construction, for instance, finding matrices which are MDS (and therefore appropriately secure for usage in ciphers) whereas presenting low computational cost. There have been some strategies towards MDS matrices construction, such as instantiation of special types of matrices (e.g Cauchy, Vandermonde, Hadamard, circulant) that diminished the number of required initial element choices. For example, some of these constructions require that only the n elements of the first row are chosen, while the remaining are derived from the first row, instead of all the n^2 elements of a complete $n \times n$ matrix. Another aspect of MDS matrix construction is the search for involutory matrices, since the matrix and its inverse have the same **xtime** and **xor** counts thus leading to equal computational cost for encryption and decryption.

Notwithstanding, the construction of efficient MDS matrices with genetic algorithms has not yet been explored, to the best of our knowledge. We then choose to investigate genetic algorithms usage not only due to novelty of the approach, but also because of their applicability in optimization problems, modeling MDS matrix construction as an optimization problem where we want to optimize the computational cost of the matrices. The computational cost is measured in terms of the count of **xtime** and **xor** operations required for matrix multiplication, therefore matrices with low counts are the ones with less computational cost and thus the most attractive for our problem.

Contribution

We begin our work with preliminary studies about diffusion layers, comparing how complete diffusion is attained by two classical ciphers, DES and AES. We show a detailed analysis of how diffusion is achieved at each round of each cipher. For DES, our analysis models the DES state bits as nodes in a graph. We believe this form of diffusion analysis we propose could be applied to study diffusion layers in other ciphers and hash functions, but we leave it as a future work. We also believe this approach towards diffusion analysis has not yet been detailed in the literature, thus making it one of the contributions of this work. After the diffusion analysis, we discuss differential and linear cryptanalysis, explaining the relationship between the diffusion property and security against these two classical attacks.

Then, we present our MDS matrix catalogue, which contains the **xtime** and **xor** counts of each matrix found in our literature review, together with the cost of its inverse transform for the correspondent finite field. We also indicate the matrix construction type (e.g Vandermonde, circulant, Hadamard) and whether it is involutory or not. To the best of our knowledge, this is the first complete catalogue of this kind. Furthermore, it covers 22 years of cryptography literature, since we start with the SHARK [62] cipher, conceived in 1996, and end with Duwal's MDS matrices [28], published in 2018. We also pinpoint three matrices presented as MDS in the literature which our practical experiments revealed not to be MDS due to the presence of singular submatrices. Finally, we explain how to model MDS matrix construction in the genetic algorithms setting and discuss results of our endeavours regarding MDS matrix search with genetic algorithms.

Text Organization

Chapters 1, 2 and 3 consist of our preliminary studies of the diffusion property. Specifically, Chapter 1 introduces and defines the diffusion property in Cryptography, and analyses it in the substitution boxes (S-boxes) of three ciphers: DES [58], AES [59], and SAFER [52]. The analysis is conducted extracting the algebraic normal forms of each S-box. Chapter 2 expands diffusion analysis to the whole DES cipher, modeling it as a graph. Chapter 3 explains how complete diffusion is achieved by AES. Chapters 4 and 5 are, respectively, about differential [11] and linear [54] cryptanalysis, establishing a relationship between the diffusion property and security against the two classical attacks. Chapter 6 compares AES and DES with respect to the diffusion property and expains why AES can withstand both attacks while DES cannot, and why, since AES, several block cipher designs opt to use MDS matrices combined with permutations for their diffusion layers. Chapter 7 details linear error correcting codes theory and MDS matrix related concepts, and presents our MDS matrix catalogue. Chapter 8 explains our experiments regarding MDS matrix search with Genetic Algorithms. Appendix A contains background and specifications of AES and DES, definitions of block ciphers, Feistel Networks and Substitution Permutation Networks. Although these structures, as well as AES and DES, are classical concepts in Cryptology, we provide them in the text for convenience. However, we leave it at the end since it is mostly background.

Note to readers

Each chapter was developed separately throughout our work, as partial, isolated reports on a sepecifc subject, e.g a report solely on DES, a report only on AES, a report focused on linear cryptanalysis, and so forth. Because of that, each chapter may contain mathematical notations, acronyms, appendix and conclusion sections of its own.

Chapter 1

Analysis of the diffusion property in the DES, AES and SAFER S-Boxes

In this chapter, we obtain the Algebraic Normal Forms (ANFs) of the S-Boxes (Substitution Boxes) of three ciphers (DES [58], AES [59] and SAFER [52]), and use them to analyze and compare these S-Boxes with respect to the complete diffusion (see Definition 1) cryptographic property.

We assume the reader is familiar with the following concepts:

- the XOR Boolean function
- block ciphers (see Appendix A)
- vectors and vector spaces
- the AES cipher [59] (see Appendix A)
- the DES cipher [58] (see Appendix A)

1.1 Notation

- $\mathbb{F}_2$: the finite field $(\{0,1\},+,\cdot)$, also denoted by $GF(2)$ in the literature
- $\mathbb{F}_2^n$, $\mathbb{F}_2^m$: vector spaces of dimension n (m) with elements of $\mathbb{F}_2$, i.e "bit vectors"
- n: dimension of a vector space, also, the number of variables (inputs) of a vectorial Boolean function (S-Box)
- m: dimension of a vector space, also, the number of outputs of a vectorial Boolean function (S-Box)
- f: a Boolean function
- x_i: an input variable of a Boolean function
- x: ordered tuple $(x_1, ..., x_n)$ with the inputs of a Boolean function
- u, x: monomials of the Algebraic Normal Form of a Boolean function
- a_i: a coefficient of the Algebraic Normal Form of a Boolean function
- y_i: an output bit of a vectorial Boolean function
- y: ordered tuple $(y_1, ..., y_m)$ with the outputs of a vectorial Boolean function
- $x_i x_j$: multiplication of x_i and x_j in $\mathbb{F}_2$, i.e $x_i \cdot x_j$, in all the ANFs presented in this chapter
- $S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8$: S-Boxes of the DES algorithm

1.2 Acronyms

- ANF: Algebraic Normal Form of a Boolean function
- S-Box: Substitution Box
- DES: Data Encryption Standard
- AES: Advanced Encryption Standard
- SAFER: Secure And Fast Encryption Routine

1.3 Preliminaries

Claude Shannon introduced, in [67], two concepts that became essential for the design of secure ciphers: *confusion* and *diffusion*. He addresses these concepts considering the encryption of English texts, thus, in his paper, the plaintext and ciphertext symbols are the alphabet letters. However, we can easily adjust the concepts to discuss relationships between bits instead of relationships between letters, since any information is represented by bits in a computer, or, formally, by elements of the finite field $\mathbb{F}_2 = \{0, 1\}$.

(Confusion) *Refers to how ciphertext bits relate to plaintext (key) bits. The relationship must be complex, so that it is computationally unfeasible to be exploited in an attack.*

(Diffusion) *Refers to the influence of each plaintext (key) bit upon the ciphertext bits. This influence must be as widespread as possible.*

These two properties are extremely relevant for the design of modern ciphers — all of them must strive for confusion and diffusion of plaintext and key bits. Actually, all modern ciphers should achieve *complete diffusion* (see Definition 1) in order to be secure. Modern block ciphers particularly achieve this through iterations, substitutions, linear operations and permutations.

It's relevant to note that complete diffusion should be achieved (and evaluated whilst designing a cipher) for both plaintext and key, although in this chapter we analyze only plaintext diffusion.

Definition 1 (Complete Diffusion) *When each ciphertext bit depends on all plaintext (key) bits, complete diffusion has been achieved by the cipher.*

An S-Box (Substitution Box) is a component, commonly used by block ciphers, to achieve confusion and diffusion. S-Box usage was introduced by Feistel [30]. Furthermore, [30] proposes that S-Boxes and permutations be combined in order to build strong ciphers, i.e they "work together", playing central roles towards ensuring confusion and diffusion. In other words, using only S-Boxes does not provide complete diffusion/confusion. Similarly, using only permutations would not attain complete diffusion/confusion either. Both components are relevant for the overall security of the cipher with respect to these properties.

(S-Box) *An S-Box (Substitution Box) is a non-linear mapping from an n-bit vector to an m-bit vector, usually implemented as lookup table.*

In the case of DES, for instance, the S-Boxes are presented as integer-to-integer lookup tables. For AES, the S-Box is most commonly represented as a lookup table of integers in hexadecimal notation.

In order to analyze S-Boxes with respect to the diffusion property, we represent them as *vectorial Boolean Functions* (see Definitions 2 and 3), since integers are vectors of bits (formally, an n-bit integer is a vector of n elements from $\mathbb{F}_2$).

Definition 2 (Boolean function) *A boolean function of n variables is a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$, i.e, it maps n input bits to a single bit.*

Definition 3 (Vectorial Boolean function) *A vectorial Boolean function of n inputs and m outputs is a function mapping $\mathbb{F}_2^n$ into $\mathbb{F}_2^m$. Furthermore, each component of the output is called a* coordinate. *Each coordinate is a Boolean function, since it is a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2$.*

Definition 4 (Truth table of a Boolean function) *The truth table of a Boolean function f of n variables gives the image of f for each input in $\mathbb{F}_2^n$. As an example, see Table 1.1.*

x_1	0	0	1	1
x_2	0	1	0	1
$f(x_1, x_2)$	0	1	1	0

Table 1.1: Truth table of the XOR Boolean function

Boolean functions are represented by *truth tables* (see Definition 2) and, from a truth table, we can find the *Algebraic Normal Form* (ANF). The ANF is a *multivariate polynomial* which allows us to analyze the relationship between each input and output bit of an S-Box. A method to obtain ANFs is described in Section 1.4.

Definition 5 (Degree of an ANF) *The degree of an ANF is the number of variables present in the monomial with the maximum number of variables (the "largest product").*

For example, for $f(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_1$, the largest products are x_1x_2 and x_2x_3, with two variables. Therefore, the degree of f is 2.

1.4 Finding the ANF of a Boolean function

To find the ANF of a boolean function given its truth table, we use the method described in [13] by Anne Canteaut.

According to [13], any Boolean function can be uniquely represented by a multivariate polynomial with degree at most 1 in each input variable. So, for instance, if we have a Boolean function f of three variables, namely x_1, x_2 and x_3, its ANF is

$$f(x) = a_0 + a_1 \cdot x_1 + a_2 \cdot x_2 + a_3 \cdot x_3 + a_4 \cdot x_1 \cdot x_2 + a_5 \cdot x_1 \cdot x_3 + a_6 \cdot x_2 \cdot x_3 + a_7 \cdot x_1 \cdot x_2 \cdot x_3,$$

where $a_0, a_1, a_2, a_3, a_4, a_5, a_6$ and a_7 are the coefficients, the ordered tuple $x = (x_1, x_2, x_3)$ is the input, $\cdot$ denotes multiplication in $\mathbb{F}_2$ and $+$ denotes addition in $\mathbb{F}_2$. Each input variable is an element of $\mathbb{F}_2$, and each coefficient is also an element of $\mathbb{F}_2$. Therefore, addition $(+)$ is equivalent to an exclusive-or (XOR) operation and multiplication $(\cdot)$ is equivalent to an AND operation.

Each coefficient multiplies a monomial, and we represent a monomial in the following way: for any $u \in \mathbb{F}_2^n$, x^u denotes the monomial defined by

$$\prod_{i=1}^{n} x_i^{u_i}.$$

In other words, each monomial is represented as an n-bit vector. The i-th bit indicates whether the i-th input variable belongs to the monomial (1 if it does, 0 if it does not). The following example illustrates this for 3 variables.

$1 \to 000$, since none of the input variables belong to the monomial.
$x_1 \to 100$, since x_1 belongs to the monomial, whilst x_2 and x_3 do not.
$x_2 \to 010$, since x_2 belongs to the monomial, whilst x_1 and x_3 do not.
$x_3 \to 001$, since x_3 belongs to the monomial, whilst x_1 and x_2 do not.
$x_1x_2 \to 110$, since only x_1 and x_2 belong to the monomial.
$x_1x_3 \to 101$, since only x_1 and x_3 belong to the monomial.
$x_2x_3 \to 011$, since only x_2 and x_3 belong to the monomial.
$x_1x_2x_3 \to 111$, since all of them belong to the monomial.

Hence the ANF for this 3 variable example is given by

$$f(x) = a_{000} + a_{100} \cdot x_1 + a_{010} \cdot x_2 + a_{001} \cdot x_3 + a_{110} \cdot x_1 \cdot x_2 + a_{101} \cdot x_1 \cdot x_3 + a_{011} \cdot x_2 \cdot x_3 + a_{111} \cdot x_1 \cdot x_2 \cdot x_3.$$

This process can be applied to any n we wish, and each coefficient a_u can be found by means of the formula

$$a_u = \sum_{x \preceq u} f(x),$$

where u and x are monomials, as explained above, and $x \preceq u$ if and only if $x_i \leq u_i$ for all $1 \leq i \leq n$. Hence, for our 3 variable example,

$$a_{000} = f(000),$$

$$a_{100} = f(100) + f(000),$$

$$a_{010} = f(010) + f(000),$$

$$a_{110} = f(110) + f(010) + f(100) + f(000),$$

$$a_{001} = f(001) + f(000),$$

$$a_{101} = f(101) + f(001) + f(100) + f(000),$$

$$a_{011} = f(011) + f(001) + f(010) + f(000),$$

$$a_{111} = f(111) + f(110) + f(101) + f(100) + f(011) + f(010) + f(001) + f(000).$$

The values $f(000), f(001)$ and so forth are given by the truth table, thus we use it in order to obtain all the coefficients of the ANF. Table 1.2 shows an example truth table for f.

$x = (x_1, x_2, x_3)$	$f(x)$
000	0
001	1
010	0
011	0
100	0
101	1
110	1
111	1

Table 1.2: Truth table example for 3 variables

Using these example values, we can obtain the ANF of f:

$$a_{000} = f(000) = 0,$$

$$a_{100} = f(100) + f(000) = 0 + 0 = 0,$$

$$a_{010} = f(010) + f(000) = 0 + 0 = 0,$$

$$a_{110} = f(110) + f(010) + f(100) + f(000) = 1 + 0 + 0 + 0 = 1,$$

$$a_{001} = f(001) + f(000) = 1 + 0 = 1,$$

$$a_{101} = f(101) + f(001) + f(100) + f(000) = 1 + 1 + 0 + 0 = 0,$$

$$a_{011} = f(011) + f(001) + f(010) + f(000) = 0 + 1 + 0 + 0 = 1,$$

$$a_{111} = f(111) + f(110) + f(101) + f(100) + f(011) + f(010) + f(001) + f(000) = 0.$$

Finally

$$\begin{aligned} f(x) &= a_{000} + a_{100} \cdot x_1 + a_{010} \cdot x_2 + a_{001} \cdot x_3 + a_{110} \cdot x_1 \cdot x_2 + a_{101} \cdot x_1 \cdot x_3 + a_{011} \cdot x_2 \cdot x_3 + a_{111} \cdot x_1 \cdot x_2 \cdot x_3 \\ &= x_3 + x_1 \cdot x_2 + x_2 \cdot x_3. \end{aligned}$$

This method allows us to obtain the Algebraic Normal Form of a single output bit of an S-Box. Applying it for each of the m output bits gives us the Algebraic Normal Forms of all of them, which allows us to thoroughly analyze the S-Box with respect to the diffusion property.

Algorithm 1 explains this approach for computing the ANF of a Boolean function. For a Boolean function of n variables, with the monomials represented by bit vectors as outlined previously, there are 2^n monomials u. For each u, we must compute the coefficient a_u. To compute a_u, we need to check, for each monomial x, if we should include $f(x)$ in our sum for a_u or not. The check requires n steps. Therefore, this approach takes roughly $2^n \cdot 2^n \cdot n$ steps, i.e $O(n \cdot 2^{2n})$ runtime. In [13], an optimized method using Möbius transforms is presented, with $O(n \cdot 2^n)$ complexity.

Algorithm 1 Finding the ANF of a Boolean function

```
for $u$ from 0 to $2^n - 1$ do                          ▷ Compute each $a_u$
    $a_u \leftarrow 0$
    for $x$ from 0 to $2^n - 1$ do                      ▷ Compute each $x$
        for $i$ from 1 to $n$ do                        ▷ Check if $f(x)$ should be summed
            if $x_i > u_i$ then
                SHOULD_SUM ← false
                break
            end if
        end for
        if SHOULD_SUM = true then
            $a_u \leftarrow a_u + f(x)$
        end if
    end for
end for
return the $a_u$ values computed
```

Given that the ANF of a Boolean function can be computed with $O(n \cdot 2^n)$ complexity, obtaining the ANFs of all output bits of an m-bit output S-Box requires $O(m \cdot n \cdot 2^n)$ complexity.

1.5 ANFs for DES S-Boxes

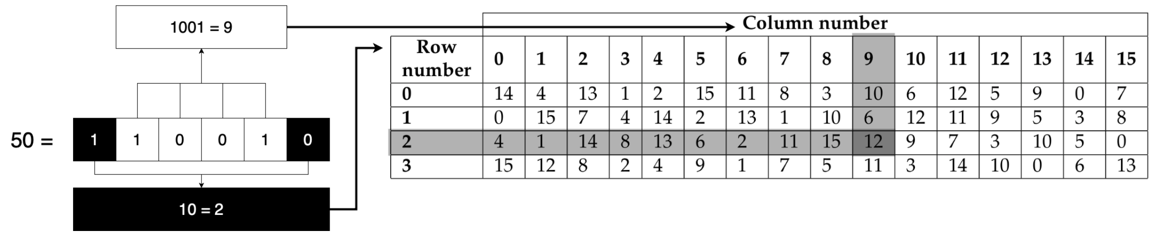
The DES algorithm [58] has 8 S-Boxes: S_1, S_2, S_3, S_4, S_5, S_6, S_7 and S_8. Each of them maps a 6-bit vector to a 4-bit vector, thus each of them is a vectorial Boolean function (see Definition 3) from $\mathbb{F}_2^6$ to $\mathbb{F}_2^4$. Here, we outline the process for S_1, but it applies to all 8 S-Boxes. Table 1.3 shows S_1.

An input integer is denoted by $x = (x_1, x_2, x_3, x_4, x_5, x_6)$, and the tuple is ordered considering the binary representation of the integer from left to right, hence x_1 is the first bit from left to right and x_6 is the last bit, also from left to right. For example, for the integer $x = 50$, the binary representation is 110010, hence $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0, x_5 = 1$ and $x_6 = 0$.

	Column number (x_2, x_3, x_4, x_5)															
Row number (x_1, x_6)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Table 1.3: DES S-Box S_1

The 4-bit output of S-Box S_1 for a given 6-bit input is obtained in the following way: the concatenation of bits x_1 and x_6 determine the row number, and the concatenation of bits x_2 to x_5 determine the column number. For $x = 50$, for instance, since $x_1 = 1$ and $x_6 = 0$, the row number is 2, since 10 is the binary representation of 2. Similarly, since the four middle bits are 1001, which is the binary representation of 9, the column number is 9. The value at row 2, column 9 is 12, thus $S_1(50) = 12$. Figure 1.1 illustrates this process.



Row number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Figure 1.1: Usage of the S_1 DES S-Box

An output integer is denoted by $y = (y_1, y_2, y_3, y_4)$, also considering the binary representation of the integer from left to right. For $x = 50$, the S_1 output is 12, thus $y_1 = 1, y_2 = 1, y_3 = 0$ and $y_4 = 0$, since the binary representation of 12 is 1100. In vectorial Boolean function notation, $S_1(1, 1, 0, 0, 1, 0) = (1, 1, 0, 0)$.

Applying the method of Section 1.4, we obtain the ANFs for each output bit y_i, $1 \leq i \leq 4$ of S_1, and they directly show the relationship with the each input bit x_j, $1 \leq j \leq 6$, allowing us to assess the diffusion property.

Equations 1.1, 1.2, 1.3 and 1.4 are the obtained Algebraic Normal Forms of y_1, y_2, y_3 and y_4, respectively, regarding the S-Box S_1 (see Table 1.3) of DES.

$$\begin{aligned} y_1 = 1 &+ x_6 + x_5 + x_4x_5x_6 + x_3 + x_3x_4 + x_3x_4x_6 + x_3x_4x_5 + x_2 + x_2x_3 + x_2x_3x_4 + x_1 \\ &+ x_1x_5 + x_1x_4 + x_1x_4x_6 + x_1x_3x_5 + x_1x_3x_4 + x_1x_3x_4x_6 + x_1x_3x_4x_5 + x_1x_2x_5x_6 \\ &+ x_1x_2x_4 + x_1x_2x_4x_6 + x_1x_2x_4x_5 + x_1x_2x_3 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4 + x_1x_2x_3x_4x_6, \end{aligned} \tag{1.1}$$

$$\begin{aligned} y_2 = 1 &+ x_6 + x_5x_6 + x_4x_6 + x_4x_5 + x_3 + x_3x_5 + x_3x_5x_6 + x_3x_4x_6 + x_3x_4x_5x_6 + x_2 + x_2x_6 + x_2x_4 + x_2x_4x_6 \\ &+ x_2x_4x_5 + x_2x_3x_6 + x_1x_6 + x_1x_5 + x_1x_4x_5 + x_1x_3 + x_1x_3x_5x_6 + x_1x_3x_4 + x_1x_3x_4x_5x_6 + x_1x_2 + x_1x_2x_6 \\ &+ x_1x_2x_5 + x_1x_2x_4x_5x_6 + x_1x_2x_3 + x_1x_2x_3x_6 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4 + x_1x_2x_3x_4x_6, \end{aligned} \tag{1.2}$$

$$\begin{aligned} y_3 = 1 &+ x_6 + x_5 + x_4 + x_4x_5 + x_4x_5x_6 + x_3x_6 + x_3x_5 + x_3x_4 + x_3x_4x_6 + x_2x_6 + x_2x_5 + x_2x_4 \\ &+ x_2x_4x_6 + x_2x_4x_5x_6 + x_2x_3 + x_2x_3x_6 + x_2x_3x_5 + x_2x_3x_4 + x_2x_3x_4x_6 + x_1 + x_1x_5 + x_1x_5x_6 \\ &+ x_1x_3x_4 + x_1x_3x_4x_5x_6 + x_1x_2 + x_1x_2x_6 + x_1x_2x_5x_6 + x_1x_2x_4 + x_1x_2x_4x_6 + x_1x_2x_4x_5 \\ &+ x_1x_2x_4x_5x_6 + x_1x_2x_3 + x_1x_2x_3x_6 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4 + x_1x_2x_3x_4x_6, \end{aligned} \tag{1.3}$$

$$\begin{aligned} y_4 = x_5x_6 &+ x_4 + x_3x_5 + x_2 + x_2x_6 + x_2x_5 + x_2x_4x_6 + x_2x_4x_5 + x_2x_3x_6 + x_2x_3x_5x_6 + x_1x_6 + x_1x_5 \\ &+ x_1x_5x_6 + x_1x_4 + x_1x_4x_6 + x_1x_4x_5 + x_1x_3 + x_1x_3x_5 + x_1x_3x_4 + x_1x_3x_4x_6 + x_1x_3x_4x_5 \\ &+ x_1x_3x_4x_5x_6 + x_1x_2x_5 + x_1x_2x_5x_6 + x_1x_2x_4x_5 + x_1x_2x_3 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4 + x_1x_2x_3x_4x_6. \end{aligned} \tag{1.4}$$

For convenience, the concept of diffusion is presented here again.

(Diffusion) *Refers to the influence of each plaintext (key) bit upon the ciphertext bits. This influence must be as widespread as possible.*

Although it refers to plaintext and ciphertext bits, for the moment, we restrict the analysis to the S-boxes — we evaluate the influence of each input bit upon the output bits. For S_1, we see clearly that y_1 depends on x_1, x_2, x_3, x_4, x_5 and x_6, since they are present in the ANF. The same holds for y_2, y_3 and y_4, thus y_2 depends on all input bits, and so do y_3 and y_4. Each input bit spreads its influence to all output bits. Therefore, S_1 *achieves complete diffusion.*

Table 1.4 shows S_2, and Equations 1.5 to 1.8 show the respective ANFs.

	Column number (x_2, x_3, x_4, x_5)															
Row number (x_1, x_6)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Table 1.4: DES S-Box S_2

$$\begin{aligned} y_1 = {} & 1 + x_6 + x_5 + x_4x_5 + x_3 + x_2x_6 + x_2x_4 + x_2x_4x_5 + x_2x_3 + x_2x_3x_6 + x_1 + x_1x_5x_6 + x_1x_4x_5 \\ & + x_1x_4x_5x_6 + x_1x_3x_5x_6 + x_1x_2x_6 + x_1x_2x_5x_6 + x_1x_2x_4x_5 + x_1x_2x_4x_5x_6 + x_1x_2x_3 + x_1x_2x_3x_6, \end{aligned} \tag{1.5}$$

$$\begin{aligned} y_2 = {} & 1 + x_6 + x_5 + x_4 + x_4x_5x_6 + x_3x_6 + x_3x_4x_5x_6 + x_2 + x_2x_4 + x_2x_4x_6 \\ & + x_2x_3 + x_1 + x_1x_2x_4x_5 + x_1x_2x_4x_5x_6 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_6, \end{aligned} \tag{1.6}$$

$$\begin{aligned} y_3 = {} & 1 + x_5 + x_4 + x_3x_5 + x_3x_4 + x_3x_4x_6 + x_3x_4x_5 + x_2 + x_2x_5x_6 + x_2x_4x_6 + x_2x_4x_5x_6 + x_2x_3x_6 \\ & + x_1 + x_1x_5x_6 + x_1x_4x_5 + x_1x_3 + x_1x_3x_5 + x_1x_3x_4 + x_1x_3x_4x_6 + x_1x_3x_4x_5 + x_1x_2 + x_1x_2x_6 \\ & + x_1x_2x_5 + x_1x_2x_4 + x_1x_2x_4x_6 + x_1x_2x_4x_5x_6 + x_1x_2x_3 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4, \end{aligned} \tag{1.7}$$

$$\begin{aligned} y_4 = {} & 1 + x_4 + x_4x_5x_6 + x_3 + x_3x_6 + x_3x_5 + x_2x_6 + x_2x_4x_5 + x_2x_4x_5x_6 + x_2x_3x_5 \\ & + x_2x_3x_5x_6 + x_1 + x_1x_6 + x_1x_5x_6 + x_1x_4x_5x_6 + x_1x_3 + x_1x_3x_6 + x_1x_3x_5 \\ & + x_1x_3x_5x_6 + x_1x_2 + x_1x_2x_5 + x_1x_2x_5x_6 + x_1x_2x_4x_6 + x_1x_2x_3x_6 + x_1x_2x_3x_5x_6. \end{aligned} \tag{1.8}$$

Table 1.5 shows S_3, and Equations 1.9 to 1.12 show the respective ANFs.

	Column number (x_2, x_3, x_4, x_5)															
Row number (x_1, x_6)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Table 1.5: DES S-Box S_3

$$\begin{aligned} y_1 = {} & 1 + x_6 + x_5 + x_4x_5 + x_3 + x_2x_6 + x_2x_4 + x_2x_4x_5 + x_2x_3 + x_2x_3x_6 + x_1 + x_1x_5x_6 + x_1x_4x_5 \\ & + x_1x_4x_5x_6 + x_1x_3x_5x_6 + x_1x_2x_6 + x_1x_2x_5x_6 + x_1x_2x_4x_5 + x_1x_2x_4x_5x_6 + x_1x_2x_3 + x_1x_2x_3x_6, \end{aligned} \tag{1.9}$$

$$\begin{aligned} y_2 = {} & 1 + x_6 + x_5 + x_4 + x_4x_5x_6 + x_3x_6 + x_3x_4x_5x_6 + x_2 + x_2x_4 + x_2x_4x_6 \\ & + x_2x_3 + x_1 + x_1x_2x_4x_5 + x_1x_2x_4x_5x_6 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_6, \end{aligned} \tag{1.10}$$

$$\begin{aligned} y_3 = {} & 1 + x_5 + x_4 + x_3x_5 + x_3x_4 + x_3x_4x_6 + x_3x_4x_5 + x_2 + x_2x_5x_6 + x_2x_4x_6 + x_2x_4x_5x_6 + x_2x_3x_6 \\ & + x_1 + x_1x_5x_6 + x_1x_4x_5 + x_1x_3 + x_1x_3x_5 + x_1x_3x_4 + x_1x_3x_4x_6 + x_1x_3x_4x_5 + x_1x_2 + x_1x_2x_6 \\ & + x_1x_2x_5 + x_1x_2x_4 + x_1x_2x_4x_6 + x_1x_2x_4x_5x_6 + x_1x_2x_3 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4, \end{aligned} \tag{1.11}$$

$$\begin{aligned} y_4 = {} & 1 + x_4 + x_4x_5x_6 + x_3 + x_3x_6 + x_3x_5 + x_2x_6 + x_2x_4x_5 + x_2x_4x_5x_6 + x_2x_3x_5 \\ & + x_2x_3x_5x_6 + x_1 + x_1x_6 + x_1x_5x_6 + x_1x_4x_5x_6 + x_1x_3 + x_1x_3x_6 + x_1x_3x_5 \\ & + x_1x_3x_5x_6 + x_1x_2 + x_1x_2x_5 + x_1x_2x_5x_6 + x_1x_2x_4x_6 + x_1x_2x_3x_6 + x_1x_2x_3x_5x_6. \end{aligned} \tag{1.12}$$

	Column number (x_2, x_3, x_4, x_5)															
Row number (x_1, x_6)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Table 1.6: DES S-Box S_4

Table 1.6 shows S_4, and Equations 1.13 to 1.16 show the respective ANFs.

$$\begin{aligned} y_1 = {} & x_6 + x_5 + x_5x_6 + x_4 + x_4x_6 + x_4x_5x_6 + x_3x_6 + x_3x_5 + x_2x_6 + x_2x_5 + x_2x_5x_6 \\ & + x_2x_4x_5 + x_2x_4x_5x_6 + x_2x_3 + x_2x_3x_5 + x_2x_3x_5x_6 + x_2x_3x_4x_6 + x_1 + x_1x_5x_6 \\ & + x_1x_4 + x_1x_4x_6 + x_1x_3x_5x_6 + x_1x_3x_4 + x_1x_3x_4x_6 + x_1x_3x_4x_5 + x_1x_3x_4x_5x_6 \\ & + x_1x_2x_5 + x_1x_2x_5x_6 + x_1x_2x_4 + x_1x_2x_4x_5 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4, \end{aligned} \tag{1.13}$$

$$\begin{aligned} y_2 = {} & 1 + x_5x_6 + x_4x_6 + x_4x_5 + x_4x_5x_6 + x_3 + x_3x_6 + x_3x_5 + x_2 + x_2x_6 + x_2x_5x_6 + x_2x_4x_5x_6 + x_2x_3 \\ & + x_2x_3x_5x_6 + x_2x_3x_4 + x_2x_3x_4x_6 + x_1 + x_1x_5 + x_1x_5x_6 + x_1x_4x_6 + x_1x_3x_5 + x_1x_3x_5x_6 \\ & + x_1x_3x_4x_6 + x_1x_3x_4x_5x_6 + x_1x_2x_5x_6 + x_1x_2x_4 + x_1x_2x_4x_5 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4, \end{aligned} \tag{1.14}$$

$$\begin{aligned} y_3 = {} & 1 + x_6 + x_5 + x_5x_6 + x_4x_6 + x_4x_5 + x_3 + x_3x_4x_5 + x_3x_4x_5x_6 + x_2 + x_2x_6 + x_2x_5x_6 \\ & + x_2x_4x_5x_6 + x_2x_3x_6 + x_2x_3x_4 + x_2x_3x_4x_6 + x_1x_6 + x_1x_5 + x_1x_5x_6 + x_1x_4 \\ & + x_1x_4x_6 + x_1x_4x_5 + x_1x_4x_5x_6 + x_1x_3x_6 + x_1x_3x_5 + x_1x_3x_4x_5 + x_1x_3x_4x_5x_6 + x_1x_2 \\ & + x_1x_2x_5 + x_1x_2x_4 + x_1x_2x_4x_5 + x_1x_2x_3x_6 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4, \end{aligned} \tag{1.15}$$

$$\begin{aligned} y_4 = {} & 1 + x_5x_6 + x_4 + x_4x_6 + x_4x_5 + x_3 + x_3x_4x_5x_6 + x_2x_6 + x_2x_5 + x_2x_5x_6 + x_2x_4x_5 + x_2x_4x_5x_6 + x_2x_3 \\ & + x_2x_3x_6 + x_2x_3x_4x_6 + x_1 + x_1x_6 + x_1x_5x_6 + x_1x_4x_6 + x_1x_4x_5x_6 + x_1x_3 + x_1x_3x_6 + x_1x_3x_5 \\ & + x_1x_3x_4x_5x_6 + x_1x_2 + x_1x_2x_5 + x_1x_2x_4 + x_1x_2x_4x_5 + x_1x_2x_3 + x_1x_2x_3x_6 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4. \end{aligned} \tag{1.16}$$

Table 1.7 shows S_5, and Equations 1.17 to 1.20 show the respective ANFs.

	Column number (x_2, x_3, x_4, x_5)															
Row number (x_1, x_6)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Table 1.7: DES S-Box S_5

$$\begin{aligned} y_1 = {} & x_6 + x_5 + x_5x_6 + x_4x_6 + x_4x_5 + x_3x_6 + x_3x_4 + x_3x_4x_6 + x_3x_4x_5 + x_3x_4x_5x_6 + x_2 + x_2x_4 \\ & + x_2x_4x_6 + x_2x_4x_5 + x_2x_3x_6 + x_2x_3x_5x_6 + x_1x_5 + x_1x_5x_6 + x_1x_4x_6 + x_1x_3 + x_1x_3x_6 + x_1x_3x_5x_6 \\ & + x_1x_3x_4x_5 + x_1x_2x_5x_6 + x_1x_2x_4 + x_1x_2x_4x_6 + x_1x_2x_4x_5 + x_1x_2x_4x_5x_6 + x_1x_2x_3x_6 + x_1x_2x_3x_4, \end{aligned} \tag{1.17}$$

$$\begin{aligned} y_2 = {} & x_6 + x_5 + x_4 + x_3 + x_3x_6 + x_3x_5x_6 + x_3x_4x_6 + x_3x_4x_5x_6 + x_2x_4 + x_2x_3x_6 + x_2x_3x_4x_6 + x_1 + x_1x_5x_6 \\ & + x_1x_4x_5 + x_1x_4x_5x_6 + x_1x_3x_4x_5 + x_1x_2x_6 + x_1x_2x_4x_6 + x_1x_2x_3 + x_1x_2x_3x_6 + x_1x_2x_3x_4 + x_1x_2x_3x_4x_6, \end{aligned} \tag{1.18}$$

$$\begin{aligned} y_3 = {}& 1 + x_5 + x_5x_6 + x_4 + x_4x_6 + x_4x_5 + x_3x_6 + x_3x_5 + x_3x_4 + x_3x_4x_6 + x_3x_4x_5 + x_3x_4x_5x_6 + x_2 + x_2x_5 \\ & + x_2x_5x_6 + x_2x_4x_6 + x_2x_4x_5 + x_2x_3x_5 + x_2x_3x_5x_6 + x_2x_3x_4 + x_2x_3x_4x_6 + x_1 + x_1x_6 + x_1x_5x_6 \\ & + x_1x_4 + x_1x_4x_5 + x_1x_3 + x_1x_3x_6 + x_1x_3x_5 + x_1x_3x_4 + x_1x_3x_4x_6 + x_1x_3x_4x_5 + x_1x_3x_4x_5x_6 \\ & + x_1x_2x_6 + x_1x_2x_5 + x_1x_2x_4 + x_1x_2x_4x_5x_6 + x_1x_2x_3 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4 + x_1x_2x_3x_4x_6, \end{aligned} \tag{1.19}$$

$$\begin{aligned} y_4 = {}& x_5x_6 + x_4x_5 + x_3 + x_3x_6 + x_3x_5 + x_3x_5x_6 + x_3x_4x_6 + x_3x_4x_5 + x_3x_4x_5x_6 + x_2x_6 \\ & + x_2x_5 + x_2x_5x_6 + x_2x_4 + x_2x_4x_6 + x_2x_4x_5x_6 + x_2x_3x_5 + x_1x_6 + x_1x_4 + x_1x_4x_5 + x_1x_3 \\ & + x_1x_3x_6 + x_1x_3x_4x_6 + x_1x_3x_4x_5 + x_1x_3x_4x_5x_6 + x_1x_2 + x_1x_2x_6 + x_1x_2x_5 + x_1x_2x_5x_6 \\ & + x_1x_2x_4 + x_1x_2x_4x_5 + x_1x_2x_3 + x_1x_2x_3x_6 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4. \end{aligned} \tag{1.20}$$

Table 1.8 shows S_6, and Equations 1.21 to 1.24 show the respective ANFs.

	Column number (x_2, x_3, x_4, x_5)															
Row number (x_1, x_6)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Table 1.8: DES S-Box S_6

$$\begin{aligned} y_1 = {}& 1 + x_5 + x_5x_6 + x_4x_6 + x_4x_5 + x_4x_5x_6 + x_3x_6 + x_3x_5x_6 + x_3x_4 + x_3x_4x_6 + x_3x_4x_5 + x_3x_4x_5x_6 \\ & + x_2 + x_2x_3 + x_2x_3x_4x_6 + x_1x_6 + x_1x_5 + x_1x_5x_6 + x_1x_4x_6 + x_1x_4x_5x_6 + x_1x_3 + x_1x_3x_6 \\ & + x_1x_3x_5 + x_1x_3x_5x_6 + x_1x_2x_4x_6 + x_1x_2x_4x_5x_6 + x_1x_2x_3x_6 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4x_6, \end{aligned} \tag{1.21}$$

$$\begin{aligned} y_2 = {}& 1 + x_6 + x_5 + x_4 + x_3 + x_3x_5 + x_3x_4x_5 + x_2 + x_2x_4 + x_2x_4x_5x_6 + x_1 + x_1x_4x_5 + x_1x_4x_5x_6 + x_1x_3 + x_1x_3x_6 \\ & + x_1x_3x_5x_6 + x_1x_3x_4x_5 + x_1x_2x_4x_5 + x_1x_2x_3 + x_1x_2x_3x_6 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4x_6, \end{aligned} \tag{1.22}$$

$$\begin{aligned} y_3 = {}& x_6 + x_4 + x_4x_5x_6 + x_3x_5 + x_2x_5x_6 + x_2x_4x_5 + x_2x_3 + x_2x_3x_5 + x_1x_6 + x_1x_5 + x_1x_4x_5x_6 + x_1x_3 \\ & + x_1x_3x_6 + x_1x_3x_5 + x_1x_3x_5x_6 + x_1x_2 + x_1x_2x_4x_5 + x_1x_2x_4x_5x_6 + x_1x_2x_3 + x_1x_2x_3x_5x_6, \end{aligned} \tag{1.23}$$

$$\begin{aligned} y_4 = {}& x_5 + x_4x_5x_6 + x_3 + x_3x_4 + x_3x_4x_6 + x_3x_4x_5 + x_3x_4x_5x_6 + x_2x_4 + x_2x_4x_5x_6 \\ & + x_2x_3 + x_2x_3x_4 + x_2x_3x_4x_6 + x_1 + x_1x_6 + x_1x_4x_5 + x_1x_4x_5x_6 + x_1x_3x_5 + x_1x_3x_4 \\ & + x_1x_3x_4x_6 + x_1x_3x_4x_5 + x_1x_3x_4x_5x_6 + x_1x_2x_6 + x_1x_2x_4x_6 + x_1x_2x_4x_5x_6 + x_1x_2x_3x_6. \end{aligned} \tag{1.24}$$

Table 1.9 shows S_7, and Equations 1.25 to 1.28 show the respective ANFs.

	Column number (x_2, x_3, x_4, x_5)															
Row number (x_1, x_6)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Table 1.9: DES S-Box S_7

$$\begin{aligned} y_1 = {}& x_6 + x_5 + x_3 + x_3x_4x_5 + x_3x_4x_5x_6 + x_2x_4 + x_2x_3 + x_2x_3x_6 + x_2x_3x_4 + x_2x_3x_4x_6 + x_1x_6 \\ & + x_1x_5 + x_1x_5x_6 + x_1x_4 + x_1x_4x_5x_6 + x_1x_3x_6 + x_1x_3x_5 + x_1x_3x_4x_5 + x_1x_3x_4x_5x_6 + x_1x_2 \\ & + x_1x_2x_4 + x_1x_2x_4x_5 + x_1x_2x_3 + x_1x_2x_3x_6 + x_1x_2x_3x_5 + x_1x_2x_3x_4 + x_1x_2x_3x_4x_6, \end{aligned} \tag{1.25}$$

$$\begin{aligned} y_2 = {} & 1 + x_5 + x_4 + x_3x_4x_5x_6 + x_2 + x_2x_6 + x_2x_4 + x_2x_4x_5x_6 + x_2x_3 + x_1 + x_1x_6 + x_1x_4 \\ & + x_1x_3 + x_1x_3x_4x_5 + x_1x_2 + x_1x_2x_4x_6 + x_1x_2x_4x_5x_6 + x_1x_2x_3x_6 + x_1x_2x_3x_4, \end{aligned} \tag{1.26}$$

$$\begin{aligned} y_3 = {} & x_5 + x_5x_6 + x_4 + x_4x_5 + x_4x_5x_6 + x_3 + x_3x_6 + x_3x_4x_6 + x_3x_4x_5x_6 + x_2 + x_2x_4x_5 + x_2x_4x_5x_6 \\ & + x_2x_3x_4x_6 + x_1x_6 + x_1x_5 + x_1x_5x_6 + x_1x_3 + x_1x_3x_5 + x_1x_3x_5x_6 + x_1x_3x_4x_6 + x_1x_3x_4x_5x_6 \\ & + x_1x_2x_4 + x_1x_2x_4x_5 + x_1x_2x_3 + x_1x_2x_3x_6 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4x_6, \end{aligned} \tag{1.27}$$

$$\begin{aligned} y_4 = {} & x_6 + x_5 + x_4x_5 + x_3 + x_3x_4 + x_3x_4x_5 + x_2 + x_2x_4x_6 + x_2x_4x_5x_6 + x_2x_3 + x_1 \\ & + x_1x_4x_6 + x_1x_4x_5x_6 + x_1x_3x_4x_6 + x_1x_3x_4x_5x_6 + x_1x_2x_5x_6 + x_1x_2x_4x_6 + x_1x_2x_3x_6. \end{aligned} \tag{1.28}$$

Table 1.10 shows S_8, and Equations 1.29 to 1.32 show the respective ANFs.

	Column number (x_2, x_3, x_4, x_5)															
Row number (x_1, x_6)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Table 1.10: DES S-Box S_8

$$\begin{aligned} y_1 = {} & 1 + x_6 + x_5 + x_4x_6 + x_4x_5x_6 + x_3 + x_3x_4 + x_3x_4x_6 + x_2x_6 + x_2x_5 + x_2x_5x_6 + x_2x_4 + x_2x_4x_6 \\ & + x_2x_4x_5 + x_2x_3x_4 + x_2x_3x_4x_6 + x_1 + x_1x_6 + x_1x_5x_6 + x_1x_4x_5 + x_1x_4x_5x_6 + x_1x_3x_6 + x_1x_3x_5 \\ & + x_1x_3x_4 + x_1x_3x_4x_6 + x_1x_2x_4x_6 + x_1x_2x_4x_5x_6 + x_1x_2x_3x_6 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4 + x_1x_2x_3x_4x_6, \end{aligned} \tag{1.29}$$

$$\begin{aligned} y_2 = {} & 1 + x_6 + x_5 + x_4 + x_3x_5 + x_2 + x_2x_5 + x_2x_4 + x_2x_4x_5 + x_2x_3 + x_1x_5x_6 + x_1x_4 + x_1x_4x_6 + x_1x_3 + x_1x_3x_5 \\ & + x_1x_3x_5x_6 + x_1x_3x_4 + x_1x_3x_4x_6 + x_1x_2x_5 + x_1x_2x_4 + x_1x_2x_4x_5 + x_1x_2x_3 + x_1x_2x_3x_4 + x_1x_2x_3x_4x_6, \end{aligned} \tag{1.30}$$

$$\begin{aligned} y_3 = {} & x_5 + x_4x_5 + x_3 + x_3x_5 + x_2 + x_2x_6 + x_2x_5x_6 + x_2x_4x_6 + x_2x_4x_5x_6 \\ & + x_2x_3x_6 + x_2x_3x_4x_6 + x_1 + x_1x_5 + x_1x_5x_6 + x_1x_4 + x_1x_4x_6 + x_1x_4x_5 \\ & + x_1x_4x_5x_6 + x_1x_3x_5 + x_1x_2x_5 + x_1x_2x_4x_5x_6 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_6, \end{aligned} \tag{1.31}$$

$$\begin{aligned} y_4 = {} & 1 + x_5 + x_5x_6 + x_4 + x_4x_6 + x_4x_5 + x_3 + x_3x_5x_6 + x_3x_4x_6 + x_3x_4x_5x_6 + x_2 + x_2x_5x_6 + x_2x_4x_5 \\ & + x_2x_3x_6 + x_1x_6 + x_1x_5 + x_1x_4x_5x_6 + x_1x_3 + x_1x_3x_5 + x_1x_3x_5x_6 + x_1x_3x_4x_6 + x_1x_2x_5x_6 \\ & + x_1x_2x_4 + x_1x_2x_4x_6 + x_1x_2x_4x_5 + x_1x_2x_3 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4x_6. \end{aligned} \tag{1.32}$$

From the ANFs of S_2, S_3, S_4, S_5, S_6, S_7 and S_8, it is possible to observe that *they all achieve complete diffusion as well*, since all input bits are present in the ANFs of all output bits. It is relevant to note that, for diffusion purposes, linear dependence is sufficient. However, linear dependence would not ensure a secure cipher, for linear dependence does not attain the confusion property — linear dependence is not complex enough for the confusion property to be ensured. All the ANFs of DES have degree equal to 5, and there are several products of input bits in the expressions in order to ensure confusion. However, a thorough confusion analysis is beyond the scope of this chapter.

1.6 ANFs for AES S-Box

The same method can be applied to analyse the diffusion property for the AES S-Box, or for any S-Box that is an n-bit integer to m-bit integer mapping.

In the case of AES, there is only one S-Box. Furthermore, $n = m = 8$. There are thus 8 input variables, $x_1, x_2, x_3, x_4, x_5, x_6, x_7$ and x_8, and 8 output variables: $y_1, y_2, y_3, y_4, y_5, y_6, y_7$ and y_8. Analogously to the analysis conducted for DES, this notation considers input and output bits *from left to right*. Formally, the AES S-Box is a mapping from $\mathbb{F}_2^8$ to $\mathbb{F}_2^8$.

Additionally, in the case of AES, the S-Box is invertible, so it is possible to obtain Algebraic Normal Forms for the inverse S-Box. For DES, that is not possible, since the S-Boxes are not bijective. The inverse S-Box is used in the AES decryption process.

The AES S-Box in hexadecimal notation is shown by Table 1.11. Given an 8-bit vector $x = (x_l, x_r)$, the S-box representation in 1.11 can be seen as a 2-dimensional table where x_l indexes the row and x_r indexes the column. For example, for $x = 53_x$, the output value is $y = ed_x$.

	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	a_x	b_x	c_x	d_x	e_x	f_x
0_x	63_x	$7c_x$	77_x	$7b_x$	$f2_x$	$6b_x$	$6f_x$	$c5_x$	30_x	01_x	67_x	$2b_x$	fe_x	$d7_x$	ab_x	76_x
1_x	ca_x	82_x	$c9_x$	$7d_x$	fa_x	59_x	47_x	$f0_x$	ad_x	$d4_x$	$a2_x$	af_x	$9c_x$	$a4_x$	72_x	$c0_x$
2_x	$b7_x$	fd_x	93_x	26_x	36_x	$3f_x$	$f7_x$	cc_x	34_x	$a5_x$	$e5_x$	$f1_x$	71_x	$d8_x$	31_x	15_x
3_x	04_x	$c7_x$	23_x	$c3_x$	18_x	96_x	05_x	$9a_x$	07_x	12_x	80_x	$e2_x$	eb_x	27_x	$b2_x$	75_x
4_x	09_x	83_x	$2c_x$	$1a_x$	$1b_x$	$6e_x$	$5a_x$	$a0_x$	52_x	$3b_x$	$d6_x$	$b3_x$	29_x	$e3_x$	$2f_x$	84_x
5_x	53_x	$d1_x$	00_x	ed_x	20_x	fc_x	$b1_x$	$5b_x$	$6a_x$	cb_x	be_x	39_x	$4a_x$	$4c_x$	58_x	cf_x
6_x	$d0_x$	ef_x	aa_x	fb_x	43_x	$4d_x$	33_x	85_x	45_x	$f9_x$	02_x	$7f_x$	50_x	$3c_x$	$9f_x$	$a8_x$
7_x	51_x	$a3_x$	40_x	$8f_x$	92_x	$9d_x$	38_x	$f5_x$	bc_x	$b6_x$	da_x	21_x	10_x	ff_x	$f3_x$	$d2_x$
8_x	cd_x	$0c_x$	13_x	ec_x	$5f_x$	97_x	44_x	17_x	$c4_x$	$a7_x$	$7e_x$	$3d_x$	64_x	$5d_x$	19_x	73_x
9_x	60_x	81_x	$4f_x$	dc_x	22_x	$2a_x$	90_x	88_x	46_x	ee_x	$b8_x$	14_x	de_x	$5e_x$	$0b_x$	db_x
a_x	$e0_x$	32_x	$3a_x$	$0a_x$	49_x	06_x	24_x	$5c_x$	$c2_x$	$d3_x$	ac_x	62_x	91_x	95_x	$e4_x$	79_x
b_x	$e7_x$	$c8_x$	37_x	$6d_x$	$8d_x$	$d5_x$	$4e_x$	$a9_x$	$6c_x$	56_x	$f4_x$	ea_x	65_x	$7a_x$	ae_x	08_x
c_x	ba_x	78_x	25_x	$2e_x$	$1c_x$	$a6_x$	$b4_x$	$c6_x$	$e8_x$	dd_x	74_x	$1f_x$	$4b_x$	bd_x	$8b_x$	$8a_x$
d_x	70_x	$3e_x$	$b5_x$	66_x	48_x	03_x	$f6_x$	$0e_x$	61_x	35_x	57_x	$b9_x$	86_x	$c1_x$	$1d_x$	$9e_x$
e_x	$e1_x$	$f8_x$	98_x	11_x	69_x	$d9_x$	$8e_x$	94_x	$9b_x$	$1e_x$	87_x	$e9_x$	ce_x	55_x	28_x	df_x
f_x	$8c_x$	$a1_x$	89_x	$0d_x$	bf_x	$e6_x$	42_x	68_x	41_x	99_x	$2d_x$	$0f_x$	$b0_x$	54_x	bb_x	16_x

Table 1.11: The AES S-box

Equation 1.33 shows the Algebraic Normal Form for the y_1 bit of the AES S-Box.

$$\begin{aligned}
y_1 = {} & x_6 + x_6x_8 + x_6x_7 + x_5x_6x_8 + x_5x_6x_7 + x_5x_6x_7x_8 + x_4 + x_4x_7x_8 + x_4x_6 + x_4x_6x_7x_8 + x_4x_5x_7x_8 \\
& + x_4x_5x_6x_7 + x_4x_5x_6x_7x_8 + x_3 + x_3x_7x_8 + x_3x_6x_8 + x_3x_6x_7x_8 + x_3x_5 + x_3x_5x_8 + x_3x_5x_7 + x_3x_5x_6 \\
& + x_3x_5x_6x_8 + x_3x_5x_6x_7 + x_3x_4x_8 + x_3x_4x_6x_7 + x_3x_4x_5 + x_3x_4x_5x_7x_8 + x_3x_4x_5x_6x_8 + x_2x_8 \\
& + x_2x_7x_8 + x_2x_6 + x_2x_6x_7 + x_2x_5x_8 + x_2x_5x_7 + x_2x_5x_7x_8 + x_2x_5x_6x_8 + x_2x_5x_6x_7x_8 + x_2x_4 \\
& + x_2x_4x_6 + x_2x_4x_6x_8 + x_2x_4x_6x_7 + x_2x_4x_6x_7x_8 + x_2x_4x_5x_8 + x_2x_4x_5x_6x_8 + x_2x_3x_8 + x_2x_3x_6 \\
& + x_2x_3x_5x_8 + x_2x_3x_5x_6x_7 + x_2x_3x_4 + x_2x_3x_4x_7x_8 + x_2x_3x_4x_6x_7 + x_2x_3x_4x_5 + x_2x_3x_4x_5x_8 \\
& + x_2x_3x_4x_5x_6 + x_2x_3x_4x_5x_6x_8 + x_1 + x_1x_8 + x_1x_7 + x_1x_6x_8 + x_1x_6x_7x_8 + x_1x_5x_8 + x_1x_5x_6x_8 \\
& + x_1x_5x_6x_7 + x_1x_4x_7 + x_1x_4x_7x_8 + x_1x_4x_6x_8 + x_1x_4x_5x_8 + x_1x_4x_5x_7 + x_1x_4x_5x_7x_8 + x_1x_4x_5x_6 \\
& + x_1x_4x_5x_6x_8 + x_1x_3 + x_1x_3x_6x_7x_8 + x_1x_3x_5 + x_1x_3x_5x_8 + x_1x_3x_4 + x_1x_3x_4x_7 + x_1x_3x_4x_7x_8 \\
& + x_1x_3x_4x_6x_7x_8 + x_1x_3x_4x_5x_7x_8 + x_1x_3x_4x_5x_6x_8 + x_1x_2x_8 + x_1x_2x_6 + x_1x_2x_6x_7x_8 + x_1x_2x_5x_8 \\
& + x_1x_2x_5x_7 + x_1x_2x_5x_6x_7x_8 + x_1x_2x_4 + x_1x_2x_4x_8 + x_1x_2x_4x_7 + x_1x_2x_4x_6 + x_1x_2x_4x_6x_7 \\
& + x_1x_2x_4x_5x_8 + x_1x_2x_4x_5x_7x_8 + x_1x_2x_4x_5x_6x_8 + x_1x_2x_3x_7 + x_1x_2x_3x_7x_8 + x_1x_2x_3x_6 \\
& + x_1x_2x_3x_6x_7 + x_1x_2x_3x_5x_8 + x_1x_2x_3x_5x_7 + x_1x_2x_3x_5x_7x_8 + x_1x_2x_3x_5x_6x_8 + x_1x_2x_3x_4 \\
& + x_1x_2x_3x_4x_7 + x_1x_2x_3x_4x_6x_8 + x_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5x_7 + x_1x_2x_3x_4x_5x_7x_8 + x_1x_2x_3x_4x_5x_6x_8
\end{aligned} \tag{1.33}$$

Equation 1.34 shows the Algebraic Normal Form for the y_2 bit of the AES S-Box.

$$\begin{aligned}
y_2 = {} & 1 + x_5 + x_5x_7 + x_5x_7x_8 + x_5x_6 + x_4x_8 + x_4x_7x_8 + x_4x_6x_8 + x_4x_6x_7x_8 + x_4x_5x_7 + x_4x_5x_7x_8 \\
& + x_4x_5x_6 + x_4x_5x_6x_7 + x_3 + x_3x_8 + x_3x_7x_8 + x_3x_6x_8 + x_3x_6x_7 + x_3x_6x_7x_8 + x_3x_5 + x_3x_5x_8 \\
& + x_3x_5x_6x_8 + x_3x_5x_6x_7 + x_3x_5x_6x_7x_8 + x_3x_4x_8 + x_3x_4x_6x_8 + x_3x_4x_6x_7 + x_3x_4x_5x_7x_8 + x_3x_4x_5x_6 \\
& + x_3x_4x_5x_6x_8 + x_3x_4x_5x_6x_7 + x_2 + x_2x_6x_8 + x_2x_6x_7 + x_2x_5x_8 + x_2x_5x_7 + x_2x_5x_7x_8 + x_2x_5x_6x_8 \\
& + x_2x_5x_6x_7 + x_2x_5x_6x_7x_8 + x_2x_4 + x_2x_4x_8 + x_2x_4x_7 + x_2x_4x_6 + x_2x_4x_6x_8 + x_2x_4x_5 + x_2x_4x_5x_8 \\
& + x_2x_4x_5x_7 + x_2x_4x_5x_6 + x_2x_3x_8 + x_2x_3x_7 + x_2x_3x_5x_6 + x_2x_3x_5x_6x_8 + x_2x_3x_5x_6x_7x_8 + x_2x_3x_4 \\
& + x_2x_3x_4x_6x_7 + x_2x_3x_4x_6x_7x_8 + x_2x_3x_4x_5x_7 + x_2x_3x_4x_5x_7x_8 + x_1x_8 + x_1x_7 + x_1x_6x_7 + x_1x_5 \\
& + x_1x_5x_7x_8 + x_1x_5x_6 + x_1x_5x_6x_8 + x_1x_5x_6x_7x_8 + x_1x_4x_8 + x_1x_4x_7 + x_1x_4x_6 + x_1x_4x_6x_7 \\
& + x_1x_4x_5x_8 + x_1x_4x_5x_7 + x_1x_4x_5x_6x_7 + x_1x_4x_5x_6x_7x_8 + x_1x_3 + x_1x_3x_8 + x_1x_3x_6x_8 + x_1x_3x_5 \\
& + x_1x_3x_5x_7 + x_1x_3x_5x_6 + x_1x_3x_5x_6x_7 + x_1x_3x_4x_7 + x_1x_3x_4x_7x_8 + x_1x_3x_4x_6x_8 + x_1x_3x_4x_6x_7x_8 \\
& + x_1x_3x_4x_5x_7 + x_1x_3x_4x_5x_7x_8 + x_1x_2x_7 + x_1x_2x_7x_8 + x_1x_2x_5 + x_1x_2x_5x_7x_8 + x_1x_2x_4x_8 \\
& + x_1x_2x_4x_7 + x_1x_2x_4x_6x_8 + x_1x_2x_4x_6x_7x_8 + x_1x_2x_4x_5x_8 + x_1x_2x_4x_5x_7x_8 + x_1x_2x_4x_5x_6 \\
& + x_1x_2x_3 + x_1x_2x_3x_6x_8 + x_1x_2x_3x_6x_7 + x_1x_2x_3x_6x_7x_8 + x_1x_2x_3x_5x_7x_8 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_4 \\
& + x_1x_2x_3x_4x_8 + x_1x_2x_3x_4x_7 + x_1x_2x_3x_4x_6x_7x_8 + x_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5x_7 + x_1x_2x_3x_4x_5x_7x_8
\end{aligned} \tag{1.34}$$

Equation 1.35 shows the Algebraic Normal Form for the y_3 bit of the AES S-Box.

$$\begin{aligned}
y_3 = {} & 1 + x_6x_7x_8 + x_5x_8 + x_5x_7x_8 + x_5x_6x_7x_8 + x_4 + x_4x_7x_8 + x_4x_6 + x_4x_6x_8 + x_4x_6x_7 + x_4x_5 + x_4x_5x_7x_8 \\
& + x_4x_5x_6x_8 + x_4x_5x_6x_7x_8 + x_3x_7 + x_3x_7x_8 + x_3x_6x_7 + x_3x_6x_7x_8 + x_3x_5x_8 + x_3x_5x_7 + x_3x_5x_6x_8 \\
& + x_3x_5x_6x_7 + x_3x_5x_6x_7x_8 + x_3x_4x_7x_8 + x_3x_4x_6 + x_3x_4x_6x_8 + x_3x_4x_6x_7 + x_3x_4x_6x_7x_8 + x_3x_4x_5 \\
& + x_3x_4x_5x_6 + x_3x_4x_5x_6x_7x_8 + x_2 + x_2x_7 + x_2x_7x_8 + x_2x_6x_8 + x_2x_6x_7 + x_2x_5x_7 + x_2x_5x_6 \\
& + x_2x_5x_6x_7 + x_2x_5x_6x_7x_8 + x_2x_4 + x_2x_4x_7 + x_2x_4x_7x_8 + x_2x_4x_6x_7x_8 + x_2x_4x_5x_7 + x_2x_4x_5x_6 \\
& + x_2x_4x_5x_6x_7 + x_2x_3x_8 + x_2x_3x_7 + x_2x_3x_7x_8 + x_2x_3x_5x_7 + x_2x_3x_5x_6 + x_2x_3x_5x_6x_7x_8 + x_2x_3x_4x_7x_8 \\
& + x_2x_3x_4x_6x_7 + x_2x_3x_4x_6x_7x_8 + x_2x_3x_4x_5 + x_2x_3x_4x_5x_8 + x_2x_3x_4x_5x_7 + x_2x_3x_4x_5x_6 + x_1 \\
& + x_1x_7x_8 + x_1x_5x_7 + x_1x_5x_7x_8 + x_1x_5x_6 + x_1x_5x_6x_7x_8 + x_1x_4x_8 + x_1x_4x_7 + x_1x_4x_7x_8 + x_1x_4x_6 \\
& + x_1x_4x_6x_7 + x_1x_4x_5x_8 + x_1x_4x_5x_7 + x_1x_4x_5x_6 + x_1x_4x_5x_6x_8 + x_1x_4x_5x_6x_7 + x_1x_3 + x_1x_3x_7 \\
& + x_1x_3x_7x_8 + x_1x_3x_6 + x_1x_3x_5 + x_1x_3x_5x_7 + x_1x_3x_5x_7x_8 + x_1x_3x_5x_6x_8 + x_1x_3x_5x_6x_7x_8 + x_1x_3x_4 \\
& + x_1x_3x_4x_7 + x_1x_3x_4x_7x_8 + x_1x_3x_4x_6 + x_1x_3x_4x_6x_8 + x_1x_3x_4x_6x_7x_8 + x_1x_3x_4x_5 + x_1x_3x_4x_5x_8 \\
& + x_1x_2x_7 + x_1x_2x_6 + x_1x_2x_6x_7x_8 + x_1x_2x_5x_6x_8 + x_1x_2x_4x_8 + x_1x_2x_4x_7x_8 + x_1x_2x_4x_5 + x_1x_2x_4x_5x_7 \\
& + x_1x_2x_4x_5x_6x_8 + x_1x_2x_4x_5x_6x_7 + x_1x_2x_3 + x_1x_2x_3x_8 + x_1x_2x_3x_7x_8 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_5x_6x_7 \\
& + x_1x_2x_3x_5x_6x_7x_8 + x_1x_2x_3x_4x_8 + x_1x_2x_3x_4x_6 + x_1x_2x_3x_4x_6x_8 + x_1x_2x_3x_4x_6x_7 + x_1x_2x_3x_4x_6x_7x_8
\end{aligned} \tag{1.35}$$

Equation 1.36 shows the Algebraic Normal Form for the y_4 bit of the AES S-Box.

$$\begin{aligned}
y_4 = {} & x_8 + x_7 + x_7x_8 + x_6 + x_5 + x_5x_6 + x_5x_6x_8 + x_4x_8 + x_4x_7 + x_4x_6x_7 + x_4x_5 + x_4x_5x_8 + x_4x_5x_6 \\
& + x_4x_5x_6x_8 + x_4x_5x_6x_7 + x_4x_5x_6x_7x_8 + x_3 + x_3x_8 + x_3x_7 + x_3x_6 + x_3x_5 + x_3x_5x_8 + x_3x_5x_7 \\
& + x_3x_5x_7x_8 + x_3x_5x_6 + x_3x_5x_6x_7 + x_3x_4 + x_3x_4x_8 + x_3x_4x_7 + x_3x_4x_7x_8 + x_3x_4x_6 + x_3x_4x_6x_7x_8 \\
& + x_3x_4x_5 + x_3x_4x_5x_8 + x_3x_4x_5x_7 + x_3x_4x_5x_6x_8 + x_3x_4x_5x_6x_7x_8 + x_2x_8 + x_2x_7 + x_2x_6x_8 + x_2x_6x_7x_8 \\
& + x_2x_5x_7x_8 + x_2x_5x_6 + x_2x_5x_6x_7x_8 + x_2x_4 + x_2x_4x_8 + x_2x_4x_7x_8 + x_2x_4x_6x_7 + x_2x_4x_5 + x_2x_4x_5x_8 \\
& + x_2x_4x_5x_7x_8 + x_2x_4x_5x_6x_8 + x_2x_4x_5x_6x_7 + x_2x_3x_7x_8 + x_2x_3x_6 + x_2x_3x_5 + x_2x_3x_5x_8 + x_2x_3x_5x_6 \\
& + x_2x_3x_5x_6x_8 + x_2x_3x_5x_6x_7 + x_2x_3x_5x_6x_7x_8 + x_2x_3x_4x_8 + x_2x_3x_4x_7x_8 + x_2x_3x_4x_6x_7 + x_2x_3x_4x_6x_7x_8 \\
& + x_2x_3x_4x_5x_7 + x_2x_3x_4x_5x_7x_8 + x_2x_3x_4x_5x_6 + x_2x_3x_4x_5x_6x_7 + x_1x_8 + x_1x_5 + x_1x_5x_7x_8 + x_1x_5x_6x_7x_8 \\
& + x_1x_4x_8 + x_1x_4x_6 + x_1x_4x_5 + x_1x_4x_5x_8 + x_1x_4x_5x_7 + x_1x_4x_5x_6 + x_1x_4x_5x_6x_7x_8 + x_1x_3 + x_1x_3x_7 \\
& + x_1x_3x_7x_8 + x_1x_3x_6x_8 + x_1x_3x_6x_7 + x_1x_3x_6x_7x_8 + x_1x_3x_5 + x_1x_3x_5x_8 + x_1x_3x_5x_7x_8 + x_1x_3x_5x_6 \\
& + x_1x_3x_5x_6x_8 + x_1x_3x_5x_6x_7 + x_1x_3x_4 + x_1x_3x_4x_6x_7 + x_1x_3x_4x_5 + x_1x_3x_4x_5x_7 + x_1x_3x_4x_5x_6 \\
& + x_1x_3x_4x_5x_6x_8 + x_1x_3x_4x_5x_6x_7 + x_1x_3x_4x_5x_6x_7x_8 + x_1x_2 + x_1x_2x_8 + x_1x_2x_7 + x_1x_2x_7x_8 \\
& + x_1x_2x_6 + x_1x_2x_6x_7 + x_1x_2x_6x_7x_8 + x_1x_2x_5 + x_1x_2x_5x_8 + x_1x_2x_5x_7x_8 + x_1x_2x_4 + x_1x_2x_4x_8 \\
& + x_1x_2x_4x_6x_8 + x_1x_2x_4x_5 + x_1x_2x_4x_5x_8 + x_1x_2x_4x_5x_6 + x_1x_2x_4x_5x_6x_7 + x_1x_2x_4x_5x_6x_7x_8 \\
& + x_1x_2x_3 + x_1x_2x_3x_7x_8 + x_1x_2x_3x_6x_7 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_8 + x_1x_2x_3x_5x_7 + x_1x_2x_3x_5x_6x_7 \\
& + x_1x_2x_3x_4x_8 + x_1x_2x_3x_4x_7 + x_1x_2x_3x_4x_6 + x_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5x_8 + x_1x_2x_3x_4x_5x_6x_8
\end{aligned} \tag{1.36}$$

Equation 1.37 shows the Algebraic Normal Form for the y_5 bit of the AES S-Box.

$$\begin{aligned}
y_5 = {} & x_8 + x_6x_7 + x_5x_8 + x_5x_7x_8 + x_5x_6 + x_5x_6x_8 + x_5x_6x_7 + x_5x_6x_7x_8 + x_4 + x_4x_7x_8 + x_4x_6x_8 + x_4x_6x_7x_8 \\
& + x_4x_5x_8 + x_4x_5x_7 + x_4x_5x_6 + x_4x_5x_6x_7x_8 + x_3x_7x_8 + x_3x_6x_7 + x_3x_6x_7x_8 + x_3x_5x_6 + x_3x_5x_6x_7 \\
& + x_3x_5x_6x_7x_8 + x_3x_4 + x_3x_4x_8 + x_3x_4x_6 + x_3x_4x_6x_7 + x_3x_4x_5x_7 + x_3x_4x_5x_7x_8 + x_3x_4x_5x_6 \\
& + x_3x_4x_5x_6x_8 + x_2 + x_2x_7x_8 + x_2x_6x_8 + x_2x_6x_7 + x_2x_5 + x_2x_5x_8 + x_2x_5x_7x_8 + x_2x_5x_6x_7 + x_2x_4x_8 \\
& + x_2x_4x_7x_8 + x_2x_4x_6x_8 + x_2x_4x_5x_8 + x_2x_4x_5x_7 + x_2x_4x_5x_7x_8 + x_2x_4x_5x_6x_7x_8 + x_2x_3 + x_2x_3x_7 \\
& + x_2x_3x_7x_8 + x_2x_3x_6x_8 + x_2x_3x_6x_7x_8 + x_2x_3x_5 + x_2x_3x_5x_7 + x_2x_3x_5x_7x_8 + x_2x_3x_5x_6x_8 + x_2x_3x_5x_6x_7 \\
& + x_2x_3x_5x_6x_7x_8 + x_2x_3x_4x_8 + x_2x_3x_4x_7 + x_2x_3x_4x_6 + x_2x_3x_4x_6x_8 + x_2x_3x_4x_6x_7 + x_2x_3x_4x_5 \\
& + x_2x_3x_4x_5x_8 + x_2x_3x_4x_5x_6 + x_2x_3x_4x_5x_6x_7 + x_2x_3x_4x_5x_6x_7x_8 + x_1 + x_1x_8 + x_1x_7 + x_1x_7x_8 + x_1x_6x_8 \\
& + x_1x_6x_7 + x_1x_5 + x_1x_5x_8 + x_1x_5x_6 + x_1x_5x_6x_8 + x_1x_5x_6x_7 + x_1x_5x_6x_7x_8 + x_1x_4x_6x_8 + x_1x_4x_6x_7 \\
& + x_1x_4x_6x_7x_8 + x_1x_4x_5 + x_1x_4x_5x_7 + x_1x_4x_5x_7x_8 + x_1x_4x_5x_6x_7x_8 + x_1x_3 + x_1x_3x_6 + x_1x_3x_6x_7 \\
& + x_1x_3x_6x_7x_8 + x_1x_3x_5 + x_1x_3x_5x_6x_8 + x_1x_3x_5x_6x_7x_8 + x_1x_3x_4x_7 + x_1x_3x_4x_7x_8 + x_1x_3x_4x_6 \\
& + x_1x_3x_4x_6x_8 + x_1x_3x_4x_6x_7x_8 + x_1x_3x_4x_5x_8 + x_1x_3x_4x_5x_7x_8 + x_1x_3x_4x_5x_6 + x_1x_3x_4x_5x_6x_7 + x_1x_2 \\
& + x_1x_2x_7x_8 + x_1x_2x_6x_8 + x_1x_2x_6x_7 + x_1x_2x_5x_8 + x_1x_2x_5x_6x_8 + x_1x_2x_5x_6x_7x_8 + x_1x_2x_4x_7 + x_1x_2x_4x_6 \\
& + x_1x_2x_4x_6x_7x_8 + x_1x_2x_4x_5 + x_1x_2x_4x_5x_8 + x_1x_2x_4x_5x_7 + x_1x_2x_4x_5x_6x_8 + x_1x_2x_4x_5x_6x_7 + x_1x_2x_3 \\
& + x_1x_2x_3x_8 + x_1x_2x_3x_7 + x_1x_2x_3x_7x_8 + x_1x_2x_3x_6x_8 + x_1x_2x_3x_6x_7 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_8 \\
& + x_1x_2x_3x_5x_7 + x_1x_2x_3x_5x_6x_8 + x_1x_2x_3x_5x_6x_7 + x_1x_2x_3x_5x_6x_7x_8 + x_1x_2x_3x_4 + x_1x_2x_3x_4x_8 \\
& + x_1x_2x_3x_4x_6 + x_1x_2x_3x_4x_6x_7 + x_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5x_7x_8 + x_1x_2x_3x_4x_5x_6 + x_1x_2x_3x_4x_5x_6x_8
\end{aligned} \tag{1.37}$$

Equation 1.38 shows the Algebraic Normal Form for the y_6 bit of the AES S-Box.

$$\begin{aligned}
y_6 = {} & x_8 + x_7 + x_6x_8 + x_5x_8 + x_5x_7x_8 + x_5x_6 + x_5x_6x_8 + x_4x_8 + x_4x_7 + x_4x_7x_8 + x_4x_6x_8 + x_4x_6x_7 + x_4x_5 \\
& + x_4x_5x_8 + x_4x_5x_7 + x_4x_5x_7x_8 + x_4x_5x_6 + x_4x_5x_6x_8 + x_4x_5x_6x_7 + x_4x_5x_6x_7x_8 + x_3 + x_3x_8 + x_3x_7x_8 \\
& + x_3x_6x_8 + x_3x_6x_7 + x_3x_6x_7x_8 + x_3x_5x_8 + x_3x_5x_7 + x_3x_5x_7x_8 + x_3x_5x_6x_8 + x_3x_5x_6x_7 + x_3x_5x_6x_7x_8 \\
& + x_3x_4x_8 + x_3x_4x_7 + x_3x_4x_6 + x_3x_4x_6x_7x_8 + x_3x_4x_5 + x_3x_4x_5x_6x_7 + x_3x_4x_5x_6x_7x_8 + x_2x_8 \\
& + x_2x_7x_8 + x_2x_6x_7 + x_2x_5x_8 + x_2x_5x_7x_8 + x_2x_5x_6 + x_2x_5x_6x_7 + x_2x_5x_6x_7x_8 + x_2x_4x_8 + x_2x_4x_7x_8 \\
& + x_2x_4x_6x_8 + x_2x_4x_5 + x_2x_4x_5x_8 + x_2x_4x_5x_7x_8 + x_2x_4x_5x_6 + x_2x_4x_5x_6x_7 + x_2x_4x_5x_6x_7x_8 + x_2x_3 \\
& + x_2x_3x_7 + x_2x_3x_7x_8 + x_2x_3x_5 + x_2x_3x_5x_8 + x_2x_3x_5x_6 + x_2x_3x_5x_6x_7x_8 + x_2x_3x_4x_6 + x_2x_3x_4x_6x_8 \\
& + x_2x_3x_4x_6x_7 + x_2x_3x_4x_6x_7x_8 + x_2x_3x_4x_5 + x_2x_3x_4x_5x_7 + x_2x_3x_4x_5x_6x_8 + x_1 + x_1x_8 + x_1x_7x_8 \\
& + x_1x_6x_8 + x_1x_6x_7 + x_1x_6x_7x_8 + x_1x_5x_8 + x_1x_5x_7 + x_1x_5x_6 + x_1x_5x_6x_8 + x_1x_5x_6x_7x_8 + x_1x_4 + x_1x_4x_8 \\
& + x_1x_4x_7 + x_1x_4x_6x_8 + x_1x_4x_6x_7 + x_1x_4x_6x_7x_8 + x_1x_4x_5x_8 + x_1x_4x_5x_7x_8 + x_1x_4x_5x_6 + x_1x_4x_5x_6x_8 \\
& + x_1x_4x_5x_6x_7x_8 + x_1x_3x_8 + x_1x_3x_7 + x_1x_3x_6x_7 + x_1x_3x_6x_7x_8 + x_1x_3x_5x_8 + x_1x_3x_5x_7 + x_1x_3x_5x_7x_8 \\
& + x_1x_3x_5x_6x_8 + x_1x_3x_4x_7 + x_1x_3x_4x_6 + x_1x_3x_4x_6x_7 + x_1x_3x_4x_6x_7x_8 + x_1x_3x_4x_5x_8 + x_1x_3x_4x_5x_7x_8 \\
& + x_1x_3x_4x_5x_6x_7 + x_1x_2 + x_1x_2x_8 + x_1x_2x_6 + x_1x_2x_6x_7 + x_1x_2x_6x_7x_8 + x_1x_2x_5x_7 + x_1x_2x_5x_7x_8 \\
& + x_1x_2x_5x_6x_7 + x_1x_2x_4 + x_1x_2x_4x_7x_8 + x_1x_2x_4x_6 + x_1x_2x_4x_6x_7 + x_1x_2x_4x_5x_7 + x_1x_2x_4x_5x_7x_8 \\
& + x_1x_2x_4x_5x_6 + x_1x_2x_4x_5x_6x_8 + x_1x_2x_4x_5x_6x_7 + x_1x_2x_4x_5x_6x_7x_8 + x_1x_2x_3x_7 + x_1x_2x_3x_6 \\
& + x_1x_2x_3x_6x_8 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_8 + x_1x_2x_3x_5x_7 + x_1x_2x_3x_5x_7x_8 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_5x_6x_8 \\
& + x_1x_2x_3x_5x_6x_7x_8 + x_1x_2x_3x_4 + x_1x_2x_3x_4x_8 + x_1x_2x_3x_4x_7 + x_1x_2x_3x_4x_7x_8 + x_1x_2x_3x_4x_6x_8 \\
& + x_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5x_7x_8 + x_1x_2x_3x_4x_5x_6 + x_1x_2x_3x_4x_5x_6x_8 + x_1x_2x_3x_4x_5x_6x_7
\end{aligned} \tag{1.38}$$

Equation 1.39 shows the Algebraic Normal Form for the y_7 bit of the AES S-Box.

$$\begin{aligned}
y_7 = {} & 1 + x_8 + x_7x_8 + x_6x_8 + x_5 + x_5x_8 + x_5x_7 + x_5x_7x_8 + x_5x_6 + x_5x_6x_8 + x_5x_6x_7 + x_4x_8 + x_4x_7 + x_4x_7x_8 \\
& + x_4x_6x_7 + x_4x_5x_8 + x_4x_5x_7 + x_4x_5x_7x_8 + x_4x_5x_6 + x_4x_5x_6x_7x_8 + x_3x_5x_7 + x_3x_5x_6 + x_3x_5x_6x_7 + x_3x_4 \\
& + x_3x_4x_8 + x_3x_4x_7x_8 + x_3x_4x_6x_8 + x_3x_4x_6x_7x_8 + x_3x_4x_5x_8 + x_3x_4x_5x_7 + x_3x_4x_5x_6 + x_3x_4x_5x_6x_8 + x_2 \\
& + x_2x_7x_8 + x_2x_6 + x_2x_6x_7x_8 + x_2x_5x_7 + x_2x_5x_7x_8 + x_2x_5x_6 + x_2x_5x_6x_8 + x_2x_5x_6x_7x_8 + x_2x_4 + x_2x_4x_8 \\
& + x_2x_4x_6x_7x_8 + x_2x_4x_5 + x_2x_4x_5x_8 + x_2x_4x_5x_7x_8 + x_2x_4x_5x_6x_7 + x_2x_4x_5x_6x_7x_8 + x_2x_3x_7 + x_2x_3x_7x_8 \\
& + x_2x_3x_6x_8 + x_2x_3x_6x_7 + x_2x_3x_5 + x_2x_3x_5x_7 + x_2x_3x_5x_7x_8 + x_2x_3x_5x_6x_7 + x_2x_3x_5x_6x_7x_8 + x_2x_3x_4x_8 \\
& + x_2x_3x_4x_7x_8 + x_2x_3x_4x_6x_8 + x_2x_3x_4x_6x_7 + x_2x_3x_4x_5 + x_2x_3x_4x_5x_7 + x_2x_3x_4x_5x_7x_8 + x_2x_3x_4x_5x_6x_8 \\
& + x_2x_3x_4x_5x_6x_7 + x_1 + x_1x_8 + x_1x_7 + x_1x_6 + x_1x_6x_8 + x_1x_5 + x_1x_5x_7 + x_1x_5x_6x_7x_8 + x_1x_4x_8 + x_1x_4x_7 \\
& + x_1x_4x_7x_8 + x_1x_4x_6x_7 + x_1x_4x_6x_7x_8 + x_1x_4x_5 + x_1x_4x_5x_7 + x_1x_4x_5x_7x_8 + x_1x_4x_5x_6 + x_1x_4x_5x_6x_8 \\
& + x_1x_4x_5x_6x_7x_8 + x_1x_3x_8 + x_1x_3x_6 + x_1x_3x_6x_7 + x_1x_3x_5 + x_1x_3x_5x_7 + x_1x_3x_5x_7x_8 + x_1x_3x_5x_6 \\
& + x_1x_3x_5x_6x_8 + x_1x_3x_4x_8 + x_1x_3x_4x_7 + x_1x_3x_4x_6 + x_1x_3x_4x_6x_7x_8 + x_1x_3x_4x_5 + x_1x_3x_4x_5x_8 \\
& + x_1x_3x_4x_5x_7 + x_1x_3x_4x_5x_6 + x_1x_3x_4x_5x_6x_8 + x_1x_2x_8 + x_1x_2x_6 + x_1x_2x_6x_7 + x_1x_2x_6x_7x_8 + x_1x_2x_5 \\
& + x_1x_2x_5x_7x_8 + x_1x_2x_5x_6x_8 + x_1x_2x_4x_8 + x_1x_2x_4x_7 + x_1x_2x_4x_6 + x_1x_2x_4x_6x_7x_8 + x_1x_2x_4x_5 \\
& + x_1x_2x_4x_5x_7x_8 + x_1x_2x_4x_5x_6 + x_1x_2x_4x_5x_6x_8 + x_1x_2x_4x_5x_6x_7 + x_1x_2x_4x_5x_6x_7x_8 + x_1x_2x_3 \\
& + x_1x_2x_3x_7x_8 + x_1x_2x_3x_6x_8 + x_1x_2x_3x_6x_7x_8 + x_1x_2x_3x_5x_8 + x_1x_2x_3x_5x_6x_8 + x_1x_2x_3x_5x_6x_7 \\
& + x_1x_2x_3x_5x_6x_7x_8 + x_1x_2x_3x_4x_8 + x_1x_2x_3x_4x_6 + x_1x_2x_3x_4x_6x_7x_8 + x_1x_2x_3x_4x_5x_7 + x_1x_2x_3x_4x_5x_7x_8
\end{aligned} \tag{1.39}$$

Equation 1.40 shows the Algebraic Normal Form for the y_8 bit of the AES S-Box.

$$\begin{aligned}
y_8 = {} & 1 + x_8 + x_7x_8 + x_6 + x_6x_7 + x_5 + x_5x_7 + x_5x_6 + x_5x_6x_7 + x_5x_6x_7x_8 + x_4 + x_4x_8 + x_4x_7 + x_4x_7x_8 \\
& + x_4x_6 + x_4x_6x_8 + x_4x_6x_7 + x_4x_5x_8 + x_4x_5x_7 + x_4x_5x_6x_7x_8 + x_3x_8 + x_3x_6x_8 + x_3x_6x_7x_8 + x_3x_5x_8 \\
& + x_3x_5x_6 + x_3x_5x_6x_8 + x_3x_5x_6x_7 + x_3x_4x_7x_8 + x_3x_4x_6x_8 + x_3x_4x_6x_7x_8 + x_3x_4x_5x_6x_8 + x_2x_8 \\
& + x_2x_7 + x_2x_7x_8 + x_2x_6 + x_2x_6x_8 + x_2x_6x_7 + x_2x_6x_7x_8 + x_2x_5x_8 + x_2x_5x_6x_7 + x_2x_5x_6x_7x_8 + x_2x_4 \\
& + x_2x_4x_8 + x_2x_4x_7 + x_2x_4x_6x_7 + x_2x_4x_5x_8 + x_2x_4x_5x_7 + x_2x_4x_5x_7x_8 + x_2x_4x_5x_6x_8 + x_2x_4x_5x_6x_7x_8 \\
& + x_2x_3 + x_2x_3x_7 + x_2x_3x_6 + x_2x_3x_6x_8 + x_2x_3x_5x_8 + x_2x_3x_5x_7x_8 + x_2x_3x_5x_6x_7 + x_2x_3x_4 + x_2x_3x_4x_8 \\
& + x_2x_3x_4x_7 + x_2x_3x_4x_7x_8 + x_2x_3x_4x_6 + x_2x_3x_4x_6x_7 + x_2x_3x_4x_5x_8 + x_2x_3x_4x_5x_6 + x_2x_3x_4x_5x_6x_8 \\
& + x_1x_7x_8 + x_1x_6 + x_1x_6x_8 + x_1x_6x_7x_8 + x_1x_5x_7 + x_1x_5x_6 + x_1x_5x_6x_8 + x_1x_5x_6x_7 + x_1x_5x_6x_7x_8 \\
& + x_1x_4x_8 + x_1x_4x_7x_8 + x_1x_4x_6 + x_1x_4x_6x_8 + x_1x_4x_6x_7 + x_1x_4x_6x_7x_8 + x_1x_4x_5 + x_1x_4x_5x_6x_7x_8 \\
& + x_1x_3 + x_1x_3x_6 + x_1x_3x_6x_8 + x_1x_3x_6x_7x_8 + x_1x_3x_5 + x_1x_3x_5x_7x_8 + x_1x_3x_5x_6 + x_1x_3x_5x_6x_7 \\
& + x_1x_3x_5x_6x_7x_8 + x_1x_3x_4x_8 + x_1x_3x_4x_7 + x_1x_3x_4x_6 + x_1x_3x_4x_6x_8 + x_1x_3x_4x_6x_7x_8 + x_1x_3x_4x_5x_8 \\
& + x_1x_3x_4x_5x_7 + x_1x_3x_4x_5x_6 + x_1x_3x_4x_5x_6x_8 + x_1x_2 + x_1x_2x_6 + x_1x_2x_6x_7 + x_1x_2x_6x_7x_8 + x_1x_2x_5 \\
& + x_1x_2x_5x_7 + x_1x_2x_5x_7x_8 + x_1x_2x_5x_6 + x_1x_2x_5x_6x_7x_8 + x_1x_2x_4x_8 + x_1x_2x_4x_6x_8 + x_1x_2x_4x_6x_7 \\
& + x_1x_2x_4x_5x_8 + x_1x_2x_4x_5x_7x_8 + x_1x_2x_4x_5x_6x_7 + x_1x_2x_4x_5x_6x_7x_8 + x_1x_2x_3 + x_1x_2x_3x_7 + x_1x_2x_3x_7x_8 \\
& + x_1x_2x_3x_6x_7 + x_1x_2x_3x_6x_7x_8 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_8 + x_1x_2x_3x_5x_6x_8 + x_1x_2x_3x_4x_7x_8 \\
& + x_1x_2x_3x_4x_6 + x_1x_2x_3x_4x_6x_7 + x_1x_2x_3x_4x_6x_7x_8 + x_1x_2x_3x_4x_5x_7 + x_1x_2x_3x_4x_5x_6 + x_1x_2x_3x_4x_5x_6x_8
\end{aligned} \tag{1.40}$$

For the inverse S-Box, the inputs are denoted by $y_1, ..., y_8$ and the outputs by $x_1, ..., x_8$, and the ANF extraction process is the same. Equations 1.41 to 1.48 show the ANFs for the inverse S-Box of AES.

$$\begin{aligned} x_1 = {} & y_7y_8 + y_6y_7 + y_5 + y_5y_8 + y_5y_6y_8 + y_5y_6y_7 + y_5y_6y_7y_8 + y_4y_8 + y_4y_7y_8 + y_4y_6 + y_4y_6y_7 + y_4y_6y_7y_8 \\ & + y_4y_5 + y_4y_5y_8 + y_4y_5y_7y_8 + y_4y_5y_6y_7 + y_4y_5y_6y_7y_8 + y_3y_7 + y_3y_6 + y_3y_6y_7 + y_3y_6y_7y_8 + y_3y_5y_7 \\ & + y_3y_5y_7y_8 + y_3y_5y_6y_8 + y_3y_4y_8 + y_3y_4y_7y_8 + y_3y_4y_6y_8 + y_3y_4y_6y_7y_8 + y_3y_4y_5y_7 + y_3y_4y_5y_6y_7 + y_2y_8 \\ & + y_2y_7 + y_2y_7y_8 + y_2y_6 + y_2y_5y_7y_8 + y_2y_5y_6y_8 + y_2y_4y_7 + y_2y_4y_7y_8 + y_2y_4y_6 + y_2y_4y_6y_7y_8 + y_2y_3 \\ & + y_2y_3y_8 + y_2y_3y_7y_8 + y_2y_3y_6y_7y_8 + y_2y_3y_5 + y_2y_3y_5y_8 + y_2y_3y_5y_7y_8 + y_2y_3y_5y_6y_8 + y_2y_3y_5y_6y_7 \\ & + y_2y_3y_5y_6y_7y_8 + y_2y_3y_4y_8 + y_2y_3y_4y_6y_8 + y_2y_3y_4y_6y_7 + y_2y_3y_4y_5 + y_2y_3y_4y_5y_7 + y_2y_3y_4y_5y_7y_8 \\ & + y_2y_3y_4y_5y_6 + y_2y_3y_4y_5y_6y_7 + y_1y_8 + y_1y_6y_8 + y_1y_6y_7y_8 + y_1y_5y_7y_8 + y_1y_5y_6y_7y_8 + y_1y_4 + y_1y_4y_7 \\ & + y_1y_4y_6 + y_1y_4y_6y_8 + y_1y_4y_6y_7y_8 + y_1y_4y_5y_7y_8 + y_1y_4y_5y_6 + y_1y_4y_5y_6y_8 + y_1y_3y_7 + y_1y_3y_6 + y_1y_3y_6y_7 \\ & + y_1y_3y_6y_7y_8 + y_1y_3y_5 + y_1y_3y_5y_8 + y_1y_3y_5y_7 + y_1y_3y_5y_6 + y_1y_3y_5y_6y_8 + y_1y_3y_4y_6y_7y_8 + y_1y_3y_4y_5 \\ & + y_1y_3y_4y_5y_7y_8 + y_1y_3y_4y_5y_6 + y_1y_3y_4y_5y_6y_7 + y_1y_2y_8 + y_1y_2y_6y_8 + y_1y_2y_6y_7y_8 + y_1y_2y_5y_8 + y_1y_2y_5y_7y_8 \\ & + y_1y_2y_5y_6y_7y_8 + y_1y_2y_4 + y_1y_2y_4y_7 + y_1y_2y_4y_7y_8 + y_1y_2y_4y_6y_7y_8 + y_1y_2y_4y_5y_6y_8 + y_1y_2y_3y_7y_8 \\ & + y_1y_2y_3y_6 + y_1y_2y_3y_6y_8 + y_1y_2y_3y_5 + y_1y_2y_3y_5y_7 + y_1y_2y_3y_5y_6 + y_1y_2y_3y_5y_6y_8 + y_1y_2y_3y_5y_6y_7 \\ & + y_1y_2y_3y_4 + y_1y_2y_3y_4y_8 + y_1y_2y_3y_4y_6y_8 + y_1y_2y_3y_4y_6y_7 + y_1y_2y_3y_4y_6y_7y_8 + y_1y_2y_3y_4y_5y_6 \end{aligned} \tag{1.41}$$

$$\begin{aligned} x_2 = {} & 1 + y_8 + y_7y_8 + y_6 + y_6y_8 + y_6y_7y_8 + y_5 + y_5y_6 + y_5y_6y_8 + y_5y_6y_7 + y_5y_6y_7y_8 + y_4y_8 + y_4y_7 \\ & + y_4y_7y_8 + y_4y_6y_8 + y_4y_5y_6 + y_4y_5y_6y_8 + y_3y_8 + y_3y_7 + y_3y_7y_8 + y_3y_6y_7 + y_3y_5 + y_3y_5y_6y_7 + y_3y_4 \\ & + y_3y_4y_8 + y_3y_4y_6 + y_3y_4y_6y_8 + y_3y_4y_6y_7 + y_3y_4y_6y_7y_8 + y_3y_4y_5 + y_3y_4y_5y_7 + y_3y_4y_5y_6y_7 + y_2y_8 \\ & + y_2y_7y_8 + y_2y_5 + y_2y_5y_8 + y_2y_5y_7y_8 + y_2y_5y_6y_8 + y_2y_5y_6y_7y_8 + y_2y_4y_8 + y_2y_4y_7 + y_2y_4y_7y_8 \\ & + y_2y_4y_6 + y_2y_4y_6y_7 + y_2y_4y_5y_6 + y_2y_3 + y_2y_3y_7 + y_2y_3y_6 + y_2y_3y_6y_7y_8 + y_2y_3y_5y_7y_8 + y_2y_3y_5y_6y_8 \\ & + y_2y_3y_4y_8 + y_2y_3y_4y_7 + y_2y_3y_4y_6y_8 + y_2y_3y_4y_6y_7 + y_2y_3y_4y_6y_7y_8 + y_2y_3y_4y_5y_8 + y_2y_3y_4y_5y_7 \\ & + y_2y_3y_4y_5y_6y_8 + y_1 + y_1y_8 + y_1y_6y_8 + y_1y_5 + y_1y_5y_8 + y_1y_5y_7 + y_1y_5y_6 + y_1y_5y_6y_8 + y_1y_4y_8 \\ & + y_1y_4y_7y_8 + y_1y_4y_6y_7y_8 + y_1y_4y_5 + y_1y_4y_5y_8 + y_1y_4y_5y_7 + y_1y_4y_5y_6 + y_1y_4y_5y_6y_8 + y_1y_4y_5y_6y_7 \\ & + y_1y_4y_5y_6y_7y_8 + y_1y_3 + y_1y_3y_8 + y_1y_3y_7y_8 + y_1y_3y_6y_7 + y_1y_3y_6y_7y_8 + y_1y_3y_5 + y_1y_3y_5y_7y_8 \\ & + y_1y_3y_5y_6y_8 + y_1y_3y_5y_6y_7y_8 + y_1y_3y_4 + y_1y_3y_4y_8 + y_1y_3y_4y_7 + y_1y_3y_4y_5 + y_1y_3y_4y_5y_8 + y_1y_3y_4y_5y_7 \\ & + y_1y_3y_4y_5y_7y_8 + y_1y_3y_4y_5y_6 + y_1y_2y_7y_8 + y_1y_2y_6 + y_1y_2y_6y_8 + y_1y_2y_6y_7 + y_1y_2y_6y_7y_8 + y_1y_2y_5 \\ & + y_1y_2y_5y_8 + y_1y_2y_5y_7 + y_1y_2y_5y_7y_8 + y_1y_2y_5y_6y_7 + y_1y_2y_5y_6y_7y_8 + y_1y_2y_4 + y_1y_2y_4y_7y_8 + y_1y_2y_4y_6y_7 \\ & + y_1y_2y_4y_6y_7y_8 + y_1y_2y_4y_5y_8 + y_1y_2y_4y_5y_7y_8 + y_1y_2y_4y_5y_6y_8 + y_1y_2y_4y_5y_6y_7 + y_1y_2y_3y_7y_8 + y_1y_2y_3y_6 \\ & + y_1y_2y_3y_6y_7y_8 + y_1y_2y_3y_5 + y_1y_2y_3y_5y_7 + y_1y_2y_3y_5y_6 + y_1y_2y_3y_5y_6y_8 + y_1y_2y_3y_5y_6y_7 + y_1y_2y_3y_5y_6y_7y_8 \\ & + y_1y_2y_3y_4 + y_1y_2y_3y_4y_8 + y_1y_2y_3y_4y_7y_8 + y_1y_2y_3y_4y_6y_8 + y_1y_2y_3y_4y_5 + y_1y_2y_3y_4y_5y_8 + y_1y_2y_3y_4y_5y_6 \end{aligned} \tag{1.42}$$

$$\begin{aligned} x_3 = {} & y_7 + y_7y_8 + y_6 + y_6y_7 + y_6y_7y_8 + y_5 + y_5y_8 + y_5y_7 + y_5y_7y_8 + y_5y_6y_7 + y_5y_6y_7y_8 + y_4 + y_4y_7 \\ & + y_4y_6y_8 + y_4y_5 + y_4y_5y_7y_8 + y_3y_8 + y_3y_7 + y_3y_7y_8 + y_3y_6y_7 + y_3y_5y_8 + y_3y_4 + y_3y_4y_7y_8 + y_3y_4y_6y_8 \\ & + y_3y_4y_6y_7 + y_3y_4y_6y_7y_8 + y_3y_4y_5 + y_3y_4y_5y_7y_8 + y_3y_4y_5y_6 + y_3y_4y_5y_6y_7 + y_3y_4y_5y_6y_7y_8 + y_2 \\ & + y_2y_7 + y_2y_7y_8 + y_2y_6y_8 + y_2y_6y_7 + y_2y_5y_7 + y_2y_5y_6 + y_2y_5y_6y_8 + y_2y_5y_6y_7y_8 + y_2y_4 + y_2y_4y_6 \\ & + y_2y_4y_6y_7 + y_2y_4y_5 + y_2y_4y_5y_8 + y_2y_4y_5y_7 + y_2y_4y_5y_6y_7 + y_2y_3 + y_2y_3y_8 + y_2y_3y_6 + y_2y_3y_6y_7y_8 \\ & + y_2y_3y_5y_6 + y_2y_3y_4 + y_2y_3y_4y_8 + y_2y_3y_4y_7y_8 + y_2y_3y_4y_6 + y_2y_3y_4y_6y_8 + y_2y_3y_4y_6y_7 + y_2y_3y_4y_6y_7y_8 \\ & + y_2y_3y_4y_5y_8 + y_2y_3y_4y_5y_6 + y_1 + y_1y_8 + y_1y_5y_8 + y_1y_5y_7y_8 + y_1y_5y_6y_8 + y_1y_5y_6y_7y_8 + y_1y_4y_7 \\ & + y_1y_4y_6y_7y_8 + y_1y_4y_5 + y_1y_4y_5y_8 + y_1y_4y_5y_6y_8 + y_1y_3 + y_1y_3y_8 + y_1y_3y_6 + y_1y_3y_5y_7 + y_1y_3y_5y_7y_8 \\ & + y_1y_3y_5y_6y_7 + y_1y_3y_4 + y_1y_3y_4y_7y_8 + y_1y_3y_4y_6 + y_1y_3y_4y_6y_8 + y_1y_3y_4y_6y_7y_8 + y_1y_3y_4y_5 + y_1y_3y_4y_5y_8 \\ & + y_1y_3y_4y_5y_6 + y_1y_3y_4y_5y_6y_7 + y_1y_2y_8 + y_1y_2y_7 + y_1y_2y_6 + y_1y_2y_6y_8 + y_1y_2y_6y_7 + y_1y_2y_5y_8 \\ & + y_1y_2y_5y_7y_8 + y_1y_2y_5y_6 + y_1y_2y_5y_6y_7 + y_1y_2y_5y_6y_7y_8 + y_1y_2y_4 + y_1y_2y_4y_8 + y_1y_2y_4y_7 + y_1y_2y_4y_7y_8 \\ & + y_1y_2y_4y_6y_8 + y_1y_2y_4y_5y_7 + y_1y_2y_4y_5y_6y_8 + y_1y_2y_4y_5y_6y_7 + y_1y_2y_3 + y_1y_2y_3y_8 + y_1y_2y_3y_6y_7y_8 \\ & + y_1y_2y_3y_5y_8 + y_1y_2y_3y_5y_6 + y_1y_2y_3y_5y_6y_8 + y_1y_2y_3y_5y_6y_7y_8 + y_1y_2y_3y_4 + y_1y_2y_3y_4y_8 + y_1y_2y_3y_4y_7 \\ & + y_1y_2y_3y_4y_7y_8 + y_1y_2y_3y_4y_5 + y_1y_2y_3y_4y_5y_7y_8 + y_1y_2y_3y_4y_5y_6 + y_1y_2y_3y_4y_5y_6y_8 + y_1y_2y_3y_4y_5y_6y_7 \end{aligned} \tag{1.43}$$

$$\begin{aligned}
x_4 = 1 &+ y_8 + y_7 + y_6y_8 + y_6y_7y_8 + y_5y_6 + y_5y_6y_8 + y_4y_7 + y_4y_6y_8 + y_4y_6y_7y_8 + y_4y_5y_7 + y_4y_5y_7y_8 + y_4y_5y_6y_8 \\
&+ y_4y_5y_6y_7 + y_3y_8 + y_3y_7 + y_3y_6 + y_3y_6y_8 + y_3y_5 + y_3y_5y_7 + y_3y_5y_7y_8 + y_3y_5y_6y_7 + y_3y_5y_6y_7y_8 + y_3y_4 \\
&+ y_3y_4y_7 + y_3y_4y_6 + y_3y_4y_5y_7 + y_3y_4y_5y_6y_8 + y_2y_8 + y_2y_7 + y_2y_7y_8 + y_2y_6 + y_2y_6y_8 + y_2y_6y_7 + y_2y_5y_8 \\
&+ y_2y_5y_7y_8 + y_2y_5y_6y_8 + y_2y_5y_6y_7 + y_2y_4 + y_2y_4y_8 + y_2y_4y_7 + y_2y_4y_7y_8 + y_2y_4y_6y_8 + y_2y_4y_6y_7 \\
&+ y_2y_4y_6y_7y_8 + y_2y_4y_5 + y_2y_4y_5y_7y_8 + y_2y_4y_5y_6 + y_2y_4y_5y_6y_8 + y_2y_4y_5y_6y_7y_8 + y_2y_3y_8 + y_2y_3y_7y_8 \\
&+ y_2y_3y_6 + y_2y_3y_6y_7 + y_2y_3y_6y_7y_8 + y_2y_3y_5 + y_2y_3y_5y_6y_7 + y_2y_3y_5y_6y_7y_8 + y_2y_3y_4y_7y_8 + y_2y_3y_4y_6 \\
&+ y_2y_3y_4y_6y_8 + y_2y_3y_4y_6y_7 + y_2y_3y_4y_6y_7y_8 + y_2y_3y_4y_5y_7y_8 + y_2y_3y_4y_5y_6 + y_2y_3y_4y_5y_6y_7 + y_1y_8 \\
&+ y_1y_7 + y_1y_7y_8 + y_1y_6 + y_1y_6y_8 + y_1y_6y_7 + y_1y_6y_7y_8 + y_1y_5y_7 + y_1y_5y_7y_8 + y_1y_5y_6y_8 + y_1y_5y_6y_7 \\
&+ y_1y_5y_6y_7y_8 + y_1y_4y_8 + y_1y_4y_7 + y_1y_4y_7y_8 + y_1y_4y_5 + y_1y_4y_5y_7 + y_1y_4y_5y_6 + y_1y_4y_5y_6y_8 + y_1y_4y_5y_6y_7y_8 \\
&+ y_1y_3 + y_1y_3y_6 + y_1y_3y_6y_8 + y_1y_3y_6y_7 + y_1y_3y_5 + y_1y_3y_5y_7 + y_1y_3y_5y_6y_7 + y_1y_3y_4y_7y_8 + y_1y_3y_4y_6 \\
&+ y_1y_3y_4y_6y_7 + y_1y_3y_4y_5 + y_1y_3y_4y_5y_7 + y_1y_3y_4y_5y_7y_8 + y_1y_3y_4y_5y_6 + y_1y_3y_4y_5y_6y_8 + y_1y_2y_8 \\
&+ y_1y_2y_7y_8 + y_1y_2y_6 + y_1y_2y_6y_8 + y_1y_2y_6y_7 + y_1y_2y_5y_8 + y_1y_2y_5y_7y_8 + y_1y_2y_5y_6 + y_1y_2y_5y_6y_8 \\
&+ y_1y_2y_5y_6y_7 + y_1y_2y_4y_8 + y_1y_2y_4y_7 + y_1y_2y_4y_7y_8 + y_1y_2y_4y_6y_8 + y_1y_2y_4y_5y_8 + y_1y_2y_4y_5y_7 \\
&+ y_1y_2y_4y_5y_7y_8 + y_1y_2y_4y_5y_6y_8 + y_1y_2y_4y_5y_6y_7 + y_1y_2y_4y_5y_6y_7y_8 + y_1y_2y_3y_7 + y_1y_2y_3y_7y_8 + y_1y_2y_3y_6y_7 \\
&+ y_1y_2y_3y_5 + y_1y_2y_3y_5y_7 + y_1y_2y_3y_5y_7y_8 + y_1y_2y_3y_5y_6y_8 + y_1y_2y_3y_4 + y_1y_2y_3y_4y_7 + y_1y_2y_3y_4y_6 \\
&+ y_1y_2y_3y_4y_6y_8 + y_1y_2y_3y_4y_6y_7y_8 + y_1y_2y_3y_4y_5y_7 + y_1y_2y_3y_4y_5y_7y_8 + y_1y_2y_3y_4y_5y_6 + y_1y_2y_3y_4y_5y_6y_8
\end{aligned} \tag{1.44}$$

$$\begin{aligned}
x_5 = y_8 &+ y_7 + y_6y_8 + y_6y_7 + y_6y_7y_8 + y_5 + y_5y_6 + y_4 + y_4y_7 + y_4y_6y_7 + y_4y_5y_7y_8 + y_4y_5y_6 + y_4y_5y_6y_8 \\
&+ y_4y_5y_6y_7 + y_4y_5y_6y_7y_8 + y_3y_7 + y_3y_7y_8 + y_3y_6y_7 + y_3y_6y_7y_8 + y_3y_5y_8 + y_3y_5y_7 + y_3y_5y_6y_7 \\
&+ y_3y_5y_6y_7y_8 + y_3y_4y_8 + y_3y_4y_7y_8 + y_3y_4y_6y_8 + y_3y_4y_6y_7 + y_3y_4y_5y_7y_8 + y_3y_4y_5y_6 + y_3y_4y_5y_6y_7 \\
&+ y_2y_7 + y_2y_7y_8 + y_2y_6y_8 + y_2y_5 + y_2y_5y_8 + y_2y_5y_7 + y_2y_5y_7y_8 + y_2y_5y_6y_8 + y_2y_5y_6y_7 + y_2y_4y_7 \\
&+ y_2y_4y_7y_8 + y_2y_4y_6y_8 + y_2y_4y_6y_7y_8 + y_2y_4y_5y_8 + y_2y_4y_5y_7y_8 + y_2y_4y_5y_6 + y_2y_4y_5y_6y_8 + y_2y_3y_6 \\
&+ y_2y_3y_6y_8 + y_2y_3y_6y_7y_8 + y_2y_3y_5y_8 + y_2y_3y_5y_6 + y_2y_3y_5y_6y_8 + y_2y_3y_4 + y_2y_3y_4y_8 + y_2y_3y_4y_7y_8 \\
&+ y_2y_3y_4y_6y_7 + y_2y_3y_4y_5y_7y_8 + y_2y_3y_4y_5y_6y_7 + y_1 + y_1y_7y_8 + y_1y_6y_8 + y_1y_6y_7y_8 + y_1y_5y_8 + y_1y_5y_7y_8 \\
&+ y_1y_5y_6 + y_1y_4y_5 + y_1y_4y_5y_8 + y_1y_4y_5y_7y_8 + y_1y_4y_5y_6y_8 + y_1y_4y_5y_6y_7 + y_1y_3 + y_1y_3y_8 + y_1y_3y_7 \\
&+ y_1y_3y_7y_8 + y_1y_3y_6 + y_1y_3y_6y_7y_8 + y_1y_3y_5y_8 + y_1y_3y_5y_7 + y_1y_3y_5y_6 + y_1y_3y_5y_6y_8 + y_1y_3y_5y_6y_7y_8 \\
&+ y_1y_3y_4y_7y_8 + y_1y_3y_4y_6y_7 + y_1y_3y_4y_5y_8 + y_1y_3y_4y_5y_7 + y_1y_3y_4y_5y_6y_7y_8 + y_1y_2y_8 + y_1y_2y_7y_8 \\
&+ y_1y_2y_5 + y_1y_2y_5y_7 + y_1y_2y_5y_6y_7 + y_1y_2y_5y_6y_7y_8 + y_1y_2y_4y_7 + y_1y_2y_4y_6 + y_1y_2y_4y_6y_8 + y_1y_2y_4y_6y_7 \\
&+ y_1y_2y_4y_5 + y_1y_2y_4y_5y_8 + y_1y_2y_4y_5y_7 + y_1y_2y_4y_5y_6y_8 + y_1y_2y_4y_5y_6y_7y_8 + y_1y_2y_3y_8 + y_1y_2y_3y_7 \\
&+ y_1y_2y_3y_7y_8 + y_1y_2y_3y_6 + y_1y_2y_3y_6y_7y_8 + y_1y_2y_3y_5y_8 + y_1y_2y_3y_5y_7 + y_1y_2y_3y_5y_6y_8 + y_1y_2y_3y_4y_8 \\
&+ y_1y_2y_3y_4y_6 + y_1y_2y_3y_4y_6y_8 + y_1y_2y_3y_4y_5 + y_1y_2y_3y_4y_5y_7 + y_1y_2y_3y_4y_5y_6y_8 + y_1y_2y_3y_4y_5y_6y_7
\end{aligned} \tag{1.45}$$

$$\begin{aligned}
x_6 = y_7y_8 &+ y_6y_8 + y_6y_7 + y_6y_7y_8 + y_5 + y_5y_8 + y_5y_7 + y_5y_7y_8 + y_5y_6 + y_5y_6y_7 + y_4 + y_4y_8 + y_4y_7 + y_4y_6 + y_4y_6y_8 \\
&+ y_4y_6y_7 + y_4y_6y_7y_8 + y_4y_5 + y_4y_5y_7 + y_4y_5y_7y_8 + y_4y_5y_6y_7 + y_4y_5y_6y_7y_8 + y_3 + y_3y_8 + y_3y_7y_8 + y_3y_6y_8 \\
&+ y_3y_6y_7y_8 + y_3y_5 + y_3y_5y_7 + y_3y_4y_8 + y_3y_4y_7 + y_3y_4y_6 + y_3y_4y_6y_7 + y_3y_4y_5y_8 + y_3y_4y_5y_7y_8 + y_3y_4y_5y_6 \\
&+ y_3y_4y_5y_6y_8 + y_3y_4y_5y_6y_7 + y_3y_4y_5y_6y_7y_8 + y_2y_7 + y_2y_7y_8 + y_2y_6 + y_2y_6y_7 + y_2y_6y_7y_8 + y_2y_5y_8 \\
&+ y_2y_5y_7y_8 + y_2y_5y_6y_8 + y_2y_5y_6y_7 + y_2y_5y_6y_7y_8 + y_2y_4y_7 + y_2y_4y_7y_8 + y_2y_4y_6y_8 + y_2y_4y_6y_7 + y_2y_4y_5y_8 \\
&+ y_2y_4y_5y_7 + y_2y_4y_5y_6 + y_2y_4y_5y_6y_7 + y_2y_4y_5y_6y_7y_8 + y_2y_3 + y_2y_3y_8 + y_2y_3y_7 + y_2y_3y_7y_8 + y_2y_3y_6y_7 \\
&+ y_2y_3y_6y_7y_8 + y_2y_3y_5 + y_2y_3y_5y_7 + y_2y_3y_5y_7y_8 + y_2y_3y_5y_6 + y_2y_3y_5y_6y_8 + y_2y_3y_5y_6y_7 + y_2y_3y_4 \\
&+ y_2y_3y_4y_8 + y_2y_3y_4y_6 + y_2y_3y_4y_5y_7 + y_2y_3y_4y_5y_6y_7 + y_2y_3y_4y_5y_6y_7y_8 + y_1y_7y_8 + y_1y_6 + y_1y_6y_8 + y_1y_6y_7 \\
&+ y_1y_5y_7 + y_1y_5y_6 + y_1y_4y_8 + y_1y_4y_7 + y_1y_4y_7y_8 + y_1y_4y_6y_8 + y_1y_4y_6y_7 + y_1y_4y_6y_7y_8 + y_1y_4y_5 + y_1y_4y_5y_8 \\
&+ y_1y_4y_5y_6 + y_1y_4y_5y_6y_7 + y_1y_4y_5y_6y_7y_8 + y_1y_3y_7 + y_1y_3y_6 + y_1y_3y_6y_8 + y_1y_3y_6y_7 + y_1y_3y_6y_7y_8 \\
&+ y_1y_3y_5y_7 + y_1y_3y_5y_7y_8 + y_1y_3y_5y_6 + y_1y_3y_5y_6y_7y_8 + y_1y_3y_4 + y_1y_3y_4y_7y_8 + y_1y_3y_4y_6y_8 + y_1y_3y_4y_6y_7 \\
&+ y_1y_3y_4y_6y_7y_8 + y_1y_3y_4y_5 + y_1y_3y_4y_5y_8 + y_1y_3y_4y_5y_7 + y_1y_3y_4y_5y_6 + y_1y_3y_4y_5y_6y_8 + y_1y_3y_4y_5y_6y_7 \\
&+ y_1y_3y_4y_5y_6y_7y_8 + y_1y_2 + y_1y_2y_7y_8 + y_1y_2y_6 + y_1y_2y_6y_8 + y_1y_2y_6y_7 + y_1y_2y_5y_7 + y_1y_2y_5y_6y_8 + y_1y_2y_4y_7 \\
&+ y_1y_2y_4y_7y_8 + y_1y_2y_4y_6y_8 + y_1y_2y_4y_6y_7y_8 + y_1y_2y_4y_5y_7 + y_1y_2y_4y_5y_7y_8 + y_1y_2y_4y_5y_6 + y_1y_2y_4y_5y_6y_8 \\
&+ y_1y_2y_3 + y_1y_2y_3y_7 + y_1y_2y_3y_7y_8 + y_1y_2y_3y_6 + y_1y_2y_3y_5 + y_1y_2y_3y_5y_6 + y_1y_2y_3y_5y_6y_7 + y_1y_2y_3y_4y_8 \\
&+ y_1y_2y_3y_4y_7 + y_1y_2y_3y_4y_6 + y_1y_2y_3y_4y_6y_7y_8 + y_1y_2y_3y_4y_5y_8 + y_1y_2y_3y_4y_5y_7y_8 + y_1y_2y_3y_4y_5y_6y_8
\end{aligned} \tag{1.46}$$

$$\begin{aligned}
x_7 = 1 &+ y_8 + y_6 + y_6y_7y_8 + y_5y_7y_8 + y_5y_6y_7 + y_5y_6y_7y_8 + y_4 + y_4y_6y_8 + y_4y_6y_7y_8 + y_4y_5y_7 + y_4y_5y_7y_8 \\
&+ y_4y_5y_6 + y_4y_5y_6y_8 + y_4y_5y_6y_7y_8 + y_3 + y_3y_6y_8 + y_3y_5 + y_3y_5y_7 + y_3y_5y_7y_8 + y_3y_5y_6 + y_3y_4 + y_3y_4y_6 \\
&+ y_3y_4y_6y_8 + y_3y_4y_6y_7y_8 + y_3y_4y_5y_8 + y_3y_4y_5y_6 + y_3y_4y_5y_6y_8 + y_3y_4y_5y_6y_7 + y_3y_4y_5y_6y_7y_8 + y_2y_6 \\
&+ y_2y_6y_7 + y_2y_6y_7y_8 + y_2y_5 + y_2y_5y_8 + y_2y_5y_7y_8 + y_2y_5y_6y_7 + y_2y_5y_6y_7y_8 + y_2y_4y_8 + y_2y_4y_6y_8 \\
&+ y_2y_4y_6y_7 + y_2y_4y_5y_7 + y_2y_4y_5y_6 + y_2y_4y_5y_6y_8 + y_2y_4y_5y_6y_7 + y_2y_3y_8 + y_2y_3y_7 + y_2y_3y_7y_8 + y_2y_3y_6y_8 \\
&+ y_2y_3y_6y_7 + y_2y_3y_6y_7y_8 + y_2y_3y_5 + y_2y_3y_5y_7 + y_2y_3y_5y_7y_8 + y_2y_3y_5y_6y_7 + y_2y_3y_4y_8 + y_2y_3y_4y_7y_8 \\
&+ y_2y_3y_4y_6y_8 + y_2y_3y_4y_6y_7y_8 + y_2y_3y_4y_5 + y_2y_3y_4y_5y_8 + y_2y_3y_4y_5y_7 + y_2y_3y_4y_5y_7y_8 + y_2y_3y_4y_5y_6 \\
&+ y_2y_3y_4y_5y_6y_8 + y_2y_3y_4y_5y_6y_7 + y_2y_3y_4y_5y_6y_7y_8 + y_1y_7 + y_1y_7y_8 + y_1y_6 + y_1y_6y_8 + y_1y_6y_7y_8 \\
&+ y_1y_5y_8 + y_1y_5y_7 + y_1y_5y_6 + y_1y_5y_6y_8 + y_1y_5y_6y_7y_8 + y_1y_4 + y_1y_4y_7y_8 + y_1y_4y_5y_8 + y_1y_4y_5y_7 \\
&+ y_1y_4y_5y_6 + y_1y_4y_5y_6y_8 + y_1y_3 + y_1y_3y_7 + y_1y_3y_7y_8 + y_1y_3y_6 + y_1y_3y_6y_8 + y_1y_3y_5 + y_1y_3y_5y_6 \\
&+ y_1y_3y_5y_6y_8 + y_1y_3y_5y_6y_7 + y_1y_3y_5y_6y_7y_8 + y_1y_3y_4y_7 + y_1y_3y_4y_6 + y_1y_3y_4y_6y_7y_8 + y_1y_3y_4y_5 \\
&+ y_1y_3y_4y_5y_7y_8 + y_1y_3y_4y_5y_6y_7 + y_1y_2y_7y_8 + y_1y_2y_6y_8 + y_1y_2y_6y_7 + y_1y_2y_6y_7y_8 + y_1y_2y_5y_7y_8 \\
&+ y_1y_2y_5y_6 + y_1y_2y_5y_6y_8 + y_1y_2y_5y_6y_7 + y_1y_2y_4 + y_1y_2y_4y_6 + y_1y_2y_4y_6y_7 + y_1y_2y_4y_6y_7y_8 + y_1y_2y_4y_5 \\
&+ y_1y_2y_4y_5y_8 + y_1y_2y_4y_5y_7y_8 + y_1y_2y_3 + y_1y_2y_3y_7y_8 + y_1y_2y_3y_6 + y_1y_2y_3y_6y_8 + y_1y_2y_3y_6y_7 \\
&+ y_1y_2y_3y_5y_8 + y_1y_2y_3y_5y_7 + y_1y_2y_3y_5y_6y_7 + y_1y_2y_3y_5y_6y_7y_8 + y_1y_2y_3y_4 + y_1y_2y_3y_4y_7 + y_1y_2y_3y_4y_6y_7 \\
&+ y_1y_2y_3y_4y_6y_7y_8 + y_1y_2y_3y_4y_5y_8 + y_1y_2y_3y_4y_5y_7 + y_1y_2y_3y_4y_5y_7y_8 + y_1y_2y_3y_4y_5y_6 + y_1y_2y_3y_4y_5y_6y_7
\end{aligned} \tag{1.47}$$

$$\begin{aligned}
x_8 = y_8 &+ y_6y_8 + y_6y_7 + y_6y_7y_8 + y_5 + y_5y_6y_7 + y_5y_6y_7y_8 + y_4y_7 + y_4y_6 + y_4y_6y_7y_8 + y_4y_5 + y_4y_5y_8 \\
&+ y_4y_5y_7y_8 + y_4y_5y_6 + y_4y_5y_6y_8 + y_3y_7y_8 + y_3y_5 + y_3y_5y_8 + y_3y_5y_7 + y_3y_5y_7y_8 + y_3y_5y_6y_8 \\
&+ y_3y_5y_6y_7y_8 + y_3y_4y_8 + y_3y_4y_6 + y_3y_4y_5 + y_3y_4y_5y_8 + y_3y_4y_5y_7y_8 + y_2y_8 + y_2y_6y_8 + y_2y_6y_7 \\
&+ y_2y_6y_7y_8 + y_2y_5 + y_2y_5y_6 + y_2y_5y_6y_7y_8 + y_2y_4y_7 + y_2y_4y_5 + y_2y_4y_5y_7y_8 + y_2y_4y_5y_6y_7 + y_2y_3y_7 \\
&+ y_2y_3y_5y_7 + y_2y_3y_5y_6y_8 + y_2y_3y_5y_6y_7 + y_2y_3y_5y_6y_7y_8 + y_2y_3y_4y_8 + y_2y_3y_4y_7 + y_2y_3y_4y_6y_8 \\
&+ y_2y_3y_4y_6y_7 + y_2y_3y_4y_5 + y_2y_3y_4y_5y_8 + y_2y_3y_4y_5y_7y_8 + y_2y_3y_4y_5y_6y_7 + y_1y_7 + y_1y_7y_8 + y_1y_6 \\
&+ y_1y_6y_7 + y_1y_5y_7 + y_1y_5y_7y_8 + y_1y_5y_6y_7 + y_1y_5y_6y_7y_8 + y_1y_4y_8 + y_1y_4y_7y_8 + y_1y_4y_6 + y_1y_4y_6y_8 \\
&+ y_1y_4y_5y_6y_8 + y_1y_3 + y_1y_3y_8 + y_1y_3y_7y_8 + y_1y_3y_6 + y_1y_3y_6y_8 + y_1y_3y_6y_7 + y_1y_3y_5y_8 + y_1y_3y_5y_7y_8 \\
&+ y_1y_3y_5y_6 + y_1y_3y_5y_6y_8 + y_1y_3y_5y_6y_7y_8 + y_1y_3y_4 + y_1y_3y_4y_7y_8 + y_1y_3y_4y_6 + y_1y_3y_4y_6y_8 + y_1y_3y_4y_5y_8 \\
&+ y_1y_3y_4y_5y_7y_8 + y_1y_3y_4y_5y_6y_7 + y_1y_2 + y_1y_2y_6y_8 + y_1y_2y_6y_7 + y_1y_2y_5y_8 + y_1y_2y_5y_7 + y_1y_2y_5y_6y_8 \\
&+ y_1y_2y_5y_6y_7y_8 + y_1y_2y_4 + y_1y_2y_4y_7y_8 + y_1y_2y_4y_6y_8 + y_1y_2y_4y_6y_7y_8 + y_1y_2y_4y_5 + y_1y_2y_4y_5y_8 \\
&+ y_1y_2y_4y_5y_7y_8 + y_1y_2y_4y_5y_6 + y_1y_2y_4y_5y_6y_7 + y_1y_2y_3y_8 + y_1y_2y_3y_7 + y_1y_2y_3y_7y_8 + y_1y_2y_3y_6y_7 \\
&+ y_1y_2y_3y_6y_7y_8 + y_1y_2y_3y_5 + y_1y_2y_3y_5y_7y_8 + y_1y_2y_3y_5y_6 + y_1y_2y_3y_5y_6y_8 + y_1y_2y_3y_4 + y_1y_2y_3y_4y_8 \\
&+ y_1y_2y_3y_4y_7 + y_1y_2y_3y_4y_7y_8 + y_1y_2y_3y_4y_6 + y_1y_2y_3y_4y_6y_8 + y_1y_2y_3y_4y_5 + y_1y_2y_3y_4y_5y_7y_8
\end{aligned} \tag{1.48}$$

From the ANFs, we observe that the AES S-Box and its inverse both achieve complete diffusion, since each output bit depends on all input bits.

1.7 ANFs for SAFER S-Box

The SAFER [52] cipher also uses an S-Box. Similarly to the AES S-Box, it maps from $\mathbb{F}_2^8$ to $\mathbb{F}_2^8$. Given an 8-bit input integer x, the output integer is

$$S(x) = (45^x \bmod 257) \bmod 256.$$

Analogously to the ANFs obtained for AES, the inputs are $x_1, x_2, x_3, x_4, x_5, x_6, x_7$ and x_8, since the integer x is an ordered tuple $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$, where the bits are ordered from left to right. The output is the ordered tuple $y = (y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8)$, also from left to right.

The SAFER S-Box *does not achieve complete diffusion*, because the output bit y_7 does not depend on the input bit x_1. Furthermore, the eighth output bit (y_8) depends only linearly on x_1, which might indicate weak confusion. The inverse SAFER S-Box, however, achieves complete diffusion.

The ANFs for each output bit, from y_1 to y_8, are shown by Equations 1.49 to 1.56.

$$\begin{aligned}
y_1 = {} & x_7 + x_6 + x_6x_8 + x_5x_6x_7 + x_4x_7 + x_4x_7x_8 + x_4x_6x_7x_8 + x_4x_5 + x_4x_5x_8 + x_4x_5x_7 + x_4x_5x_6 + x_4x_5x_6x_7 \\
& + x_4x_5x_6x_7x_8 + x_3x_7 + x_3x_7x_8 + x_3x_6 + x_3x_6x_8 + x_3x_6x_7x_8 + x_3x_5 + x_3x_5x_7x_8 + x_3x_5x_6x_7 \\
& + x_3x_5x_6x_7x_8 + x_3x_4 + x_3x_4x_7x_8 + x_3x_4x_6x_8 + x_3x_4x_6x_7 + x_3x_4x_5 + x_3x_4x_5x_7 + x_3x_4x_5x_7x_8 \\
& + x_2 + x_2x_8 + x_2x_7 + x_2x_7x_8 + x_2x_6x_8 + x_2x_6x_7 + x_2x_6x_7x_8 + x_2x_5x_8 + x_2x_5x_7x_8 + x_2x_5x_6 \\
& + x_2x_5x_6x_7 + x_2x_5x_6x_7x_8 + x_2x_4x_8 + x_2x_4x_7x_8 + x_2x_4x_6x_8 + x_2x_4x_6x_7 + x_2x_4x_6x_7x_8 + x_2x_4x_5x_8 \\
& + x_2x_4x_5x_7 + x_2x_4x_5x_6 + x_2x_4x_5x_6x_7 + x_2x_4x_5x_6x_7x_8 + x_2x_3 + x_2x_3x_6 + x_2x_3x_6x_8 + x_2x_3x_6x_7 \\
& + x_2x_3x_5x_8 + x_2x_3x_5x_6x_8 + x_2x_3x_5x_6x_7 + x_2x_3x_5x_6x_7x_8 + x_2x_3x_4 + x_2x_3x_4x_8 + x_2x_3x_4x_6x_8 \\
& + x_2x_3x_4x_6x_7x_8 + x_2x_3x_4x_5x_7 + x_2x_3x_4x_5x_6x_8 + x_1x_8 + x_1x_7 + x_1x_7x_8 + x_1x_6 + x_1x_6x_8 + x_1x_6x_7 \\
& + x_1x_6x_7x_8 + x_1x_5 + x_1x_5x_8 + x_1x_5x_7 + x_1x_5x_7x_8 + x_1x_5x_6 + x_1x_5x_6x_8 + x_1x_5x_6x_7 + x_1x_5x_6x_7x_8 \\
& + x_1x_4 + x_1x_4x_8 + x_1x_4x_7 + x_1x_4x_7x_8 + x_1x_4x_6 + x_1x_4x_6x_8 + x_1x_4x_6x_7 + x_1x_4x_6x_7x_8 + x_1x_4x_5 \\
& + x_1x_4x_5x_8 + x_1x_4x_5x_7 + x_1x_4x_5x_7x_8 + x_1x_4x_5x_6 + x_1x_4x_5x_6x_8 + x_1x_4x_5x_6x_7 + x_1x_4x_5x_6x_7x_8 \\
& + x_1x_3 + x_1x_3x_8 + x_1x_3x_7 + x_1x_3x_7x_8 + x_1x_3x_6 + x_1x_3x_6x_8 + x_1x_3x_6x_7 + x_1x_3x_6x_7x_8 + x_1x_3x_5 \\
& + x_1x_3x_5x_8 + x_1x_3x_5x_7 + x_1x_3x_5x_7x_8 + x_1x_3x_5x_6 + x_1x_3x_5x_6x_8 + x_1x_3x_5x_6x_7 + x_1x_3x_5x_6x_7x_8 \\
& + x_1x_3x_4 + x_1x_3x_4x_8 + x_1x_3x_4x_7 + x_1x_3x_4x_7x_8 + x_1x_3x_4x_6 + x_1x_3x_4x_6x_8 + x_1x_3x_4x_6x_7 \\
& + x_1x_3x_4x_6x_7x_8 + x_1x_3x_4x_5 + x_1x_3x_4x_5x_8 + x_1x_3x_4x_5x_7 + x_1x_3x_4x_5x_7x_8 + x_1x_3x_4x_5x_6 \\
& + x_1x_3x_4x_5x_6x_8 + x_1x_3x_4x_5x_6x_7 + x_1x_3x_4x_5x_6x_7x_8 + x_1x_2 + x_1x_2x_8 + x_1x_2x_7 + x_1x_2x_7x_8 \\
& + x_1x_2x_6 + x_1x_2x_6x_8 + x_1x_2x_6x_7 + x_1x_2x_6x_7x_8 + x_1x_2x_5 + x_1x_2x_5x_8 + x_1x_2x_5x_7 + x_1x_2x_5x_7x_8 \\
& + x_1x_2x_5x_6 + x_1x_2x_5x_6x_8 + x_1x_2x_5x_6x_7 + x_1x_2x_5x_6x_7x_8 + x_1x_2x_3 + x_1x_2x_3x_8 + x_1x_2x_3x_7 \\
& + x_1x_2x_3x_7x_8 + x_1x_2x_3x_6 + x_1x_2x_3x_6x_8 + x_1x_2x_3x_6x_7 + x_1x_2x_3x_6x_7x_8 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_8 \\
& + x_1x_2x_3x_5x_7 + x_1x_2x_3x_5x_7x_8 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_5x_6x_8 + x_1x_2x_3x_5x_6x_7 + x_1x_2x_3x_5x_6x_7x_8
\end{aligned} \tag{1.49}$$

$$\begin{aligned}
y_2 = {} & x_7 + x_7x_8 + x_6x_8 + x_6x_7 + x_5 + x_5x_8 + x_5x_6 + x_5x_6x_7 + x_4x_8 + x_4x_7 + x_4x_6 + x_4x_6x_8 + x_4x_6x_7x_8 \\
& + x_4x_5 + x_4x_5x_7x_8 + x_4x_5x_6x_7 + x_4x_5x_6x_7x_8 + x_3 + x_3x_8 + x_3x_7 + x_3x_7x_8 + x_3x_6x_8 + x_3x_5 \\
& + x_3x_5x_7 + x_3x_5x_7x_8 + x_3x_5x_6 + x_3x_5x_6x_7x_8 + x_3x_4x_8 + x_3x_4x_6x_8 + x_3x_4x_6x_7 + x_3x_4x_6x_7x_8 \\
& + x_3x_4x_5x_8 + x_3x_4x_5x_7 + x_3x_4x_5x_6 + x_3x_4x_5x_6x_8 + x_3x_4x_5x_6x_7x_8 + x_2 + x_2x_8 + x_2x_7 + x_2x_6 \\
& + x_2x_6x_7 + x_2x_6x_7x_8 + x_2x_5x_8 + x_2x_5x_7x_8 + x_2x_5x_6x_7 + x_2x_5x_6x_7x_8 + x_2x_4 + x_2x_4x_7x_8 + x_2x_4x_6 \\
& + x_2x_4x_6x_8 + x_2x_4x_5x_6 + x_2x_3x_7 + x_2x_3x_7x_8 + x_2x_3x_6x_8 + x_2x_3x_5 + x_2x_3x_5x_8 + x_2x_3x_5x_6 \\
& + x_2x_3x_5x_6x_7x_8 + x_2x_3x_4 + x_2x_3x_4x_7 + x_2x_3x_4x_7x_8 + x_2x_3x_4x_6 + x_2x_3x_4x_6x_7 + x_2x_3x_4x_5 \\
& + x_2x_3x_4x_5x_6 + x_2x_3x_4x_5x_6x_8 + x_2x_3x_4x_5x_6x_7 + x_1x_8 + x_1x_7 + x_1x_7x_8 + x_1x_6 + x_1x_6x_8 + x_1x_6x_7 \\
& + x_1x_6x_7x_8 + x_1x_5 + x_1x_5x_8 + x_1x_5x_7 + x_1x_5x_7x_8 + x_1x_5x_6 + x_1x_5x_6x_8 + x_1x_5x_6x_7 + x_1x_5x_6x_7x_8 \\
& + x_1x_4 + x_1x_4x_8 + x_1x_4x_7 + x_1x_4x_7x_8 + x_1x_4x_6 + x_1x_4x_6x_8 + x_1x_4x_6x_7 + x_1x_4x_6x_7x_8 + x_1x_4x_5 \\
& + x_1x_4x_5x_8 + x_1x_4x_5x_7 + x_1x_4x_5x_7x_8 + x_1x_4x_5x_6 + x_1x_4x_5x_6x_8 + x_1x_4x_5x_6x_7 + x_1x_4x_5x_6x_7x_8 \\
& + x_1x_3x_5x_8 + x_1x_3x_5x_7x_8 + x_1x_3x_5x_6x_8 + x_1x_3x_5x_6x_7x_8 + x_1x_3x_4x_5x_8 + x_1x_3x_4x_5x_7x_8 \\
& + x_1x_3x_4x_5x_6x_8 + x_1x_3x_4x_5x_6x_7x_8 + x_1x_2 + x_1x_2x_8 + x_1x_2x_7 + x_1x_2x_7x_8 + x_1x_2x_6 + x_1x_2x_6x_8 \\
& + x_1x_2x_6x_7 + x_1x_2x_6x_7x_8 + x_1x_2x_5 + x_1x_2x_5x_8 + x_1x_2x_5x_7 + x_1x_2x_5x_7x_8 + x_1x_2x_5x_6 + x_1x_2x_5x_6x_8 \\
& + x_1x_2x_5x_6x_7 + x_1x_2x_5x_6x_7x_8 + x_1x_2x_3x_5x_8 + x_1x_2x_3x_5x_7x_8 + x_1x_2x_3x_5x_6x_8 + x_1x_2x_3x_5x_6x_7x_8 \\
& + x_1x_2x_3x_4 + x_1x_2x_3x_4x_8 + x_1x_2x_3x_4x_7 + x_1x_2x_3x_4x_7x_8 + x_1x_2x_3x_4x_6 + x_1x_2x_3x_4x_6x_8 \\
& + x_1x_2x_3x_4x_6x_7 + x_1x_2x_3x_4x_6x_7x_8 + x_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5x_7 + x_1x_2x_3x_4x_5x_6 + x_1x_2x_3x_4x_5x_6x_7
\end{aligned} \tag{1.50}$$

$$\begin{aligned} y_3 = {} & x_8 + x_7 + x_6 + x_5 + x_5x_6 + x_5x_6x_8 + x_5x_6x_7x_8 + x_4x_7 + x_4x_7x_8 + x_4x_6x_8 + x_4x_6x_7x_8 + x_4x_5x_7x_8 \\ & + x_4x_5x_6x_8 + x_3x_7 + x_3x_7x_8 + x_3x_6 + x_3x_6x_7 + x_3x_6x_7x_8 + x_3x_5x_7 + x_3x_5x_6x_8 + x_3x_5x_6x_7x_8 \\ & + x_3x_4 + x_3x_4x_8 + x_3x_4x_7 + x_3x_4x_7x_8 + x_3x_4x_6 + x_3x_4x_6x_7 + x_3x_4x_6x_7x_8 + x_3x_4x_5x_8 \\ & + x_3x_4x_5x_7x_8 + x_3x_4x_5x_6x_8 + x_3x_4x_5x_6x_7x_8 + x_2 + x_2x_8 + x_2x_7 + x_2x_7x_8 + x_2x_6 + x_2x_6x_7x_8 \\ & + x_2x_5x_7x_8 + x_2x_5x_6 + x_2x_5x_6x_8 + x_2x_4 + x_2x_4x_7x_8 + x_2x_4x_6x_7 + x_2x_4x_5x_8 + x_2x_4x_5x_6 \\ & + x_2x_4x_5x_6x_8 + x_2x_4x_5x_6x_7 + x_2x_3 + x_2x_3x_8 + x_2x_3x_7 + x_2x_3x_7x_8 + x_2x_3x_6x_7x_8 + x_2x_3x_5 \\ & + x_2x_3x_5x_8 + x_2x_3x_5x_7 + x_2x_3x_5x_6 + x_2x_3x_4 + x_2x_3x_4x_8 + x_2x_3x_4x_6x_8 + x_2x_3x_4x_6x_7x_8 \\ & + x_2x_3x_4x_5x_7 + x_1x_8 + x_1x_7 + x_1x_7x_8 + x_1x_6 + x_1x_6x_8 + x_1x_6x_7 + x_1x_6x_7x_8 + x_1x_5 + x_1x_5x_8 \\ & + x_1x_5x_7 + x_1x_5x_7x_8 + x_1x_5x_6 + x_1x_5x_6x_8 + x_1x_5x_6x_7 + x_1x_5x_6x_7x_8 + x_1x_4 + x_1x_4x_8 \\ & + x_1x_4x_7 + x_1x_4x_7x_8 + x_1x_4x_6 + x_1x_4x_6x_8 + x_1x_4x_6x_7 + x_1x_4x_6x_7x_8 + x_1x_4x_5 + x_1x_4x_5x_8 \\ & + x_1x_4x_5x_7 + x_1x_4x_5x_7x_8 + x_1x_4x_5x_6 + x_1x_4x_5x_6x_8 + x_1x_4x_5x_6x_7 + x_1x_4x_5x_6x_7x_8 + x_1x_3x_5x_8 \\ & + x_1x_3x_5x_7x_8 + x_1x_3x_5x_6x_8 + x_1x_3x_5x_6x_7x_8 + x_1x_3x_4x_5x_8 + x_1x_3x_4x_5x_7x_8 + x_1x_3x_4x_5x_6x_8 \\ & + x_1x_3x_4x_5x_6x_7x_8 + x_1x_2 + x_1x_2x_8 + x_1x_2x_7 + x_1x_2x_7x_8 + x_1x_2x_6 + x_1x_2x_6x_8 + x_1x_2x_6x_7 \\ & + x_1x_2x_6x_7x_8 + x_1x_2x_5 + x_1x_2x_5x_8 + x_1x_2x_5x_7 + x_1x_2x_5x_7x_8 + x_1x_2x_5x_6 + x_1x_2x_5x_6x_8 \\ & + x_1x_2x_5x_6x_7 + x_1x_2x_5x_6x_7x_8 + x_1x_2x_4x_5x_6x_7x_8 + x_1x_2x_3x_5x_8 + x_1x_2x_3x_5x_7x_8 + x_1x_2x_3x_4x_5x_6x_8 \end{aligned} \tag{1.51}$$

$$\begin{aligned} y_4 = {} & x_7x_8 + x_6 + x_6x_8 + x_6x_7x_8 + x_5 + x_5x_8 + x_5x_7 + x_5x_6 + x_5x_6x_7x_8 + x_4x_6 + x_4x_6x_8 + x_4x_5x_8 + x_4x_5x_7 \\ & + x_4x_5x_7x_8 + x_4x_5x_6 + x_3x_8 + x_3x_7x_8 + x_3x_6x_8 + x_3x_5 + x_3x_5x_6 + x_3x_5x_6x_7x_8 + x_3x_4 + x_3x_4x_7x_8 \\ & + x_3x_4x_6 + x_3x_4x_6x_8 + x_3x_4x_6x_7 + x_3x_4x_5x_8 + x_3x_4x_5x_6x_7x_8 + x_2 + x_2x_7 + x_2x_5x_7 + x_2x_5x_7x_8 + x_2x_4 \\ & + x_2x_4x_8 + x_2x_4x_7x_8 + x_2x_4x_6 + x_2x_4x_6x_7x_8 + x_2x_4x_5x_7 + x_2x_4x_5x_6 + x_2x_4x_5x_6x_8 + x_2x_4x_5x_6x_7x_8 \\ & + x_2x_3 + x_2x_3x_7 + x_2x_3x_7x_8 + x_2x_3x_6x_8 + x_2x_3x_5 + x_2x_3x_5x_8 + x_2x_3x_5x_6 + x_2x_3x_5x_6x_8 + x_2x_3x_4x_8 \\ & + x_2x_3x_4x_6x_8 + x_2x_3x_4x_6x_7x_8 + x_2x_3x_4x_5x_7x_8 + x_2x_3x_4x_5x_6x_7 + x_2x_3x_4x_5x_6x_7x_8 + x_1x_8 + x_1x_7 \\ & + x_1x_7x_8 + x_1x_6 + x_1x_6x_8 + x_1x_6x_7 + x_1x_6x_7x_8 + x_1x_5 + x_1x_5x_8 + x_1x_5x_7 + x_1x_5x_7x_8 + x_1x_5x_6 \\ & + x_1x_5x_6x_8 + x_1x_5x_6x_7 + x_1x_5x_6x_7x_8 + x_1x_4 + x_1x_4x_8 + x_1x_4x_7 + x_1x_4x_7x_8 + x_1x_4x_6 + x_1x_4x_6x_8 \\ & + x_1x_4x_6x_7 + x_1x_4x_6x_7x_8 + x_1x_4x_5 + x_1x_4x_5x_8 + x_1x_4x_5x_7 + x_1x_4x_5x_7x_8 + x_1x_4x_5x_6 + x_1x_4x_5x_6x_8 \\ & + x_1x_4x_5x_6x_7 + x_1x_4x_5x_6x_7x_8 + x_1x_3x_6 + x_1x_3x_6x_8 + x_1x_3x_6x_7 + x_1x_3x_6x_7x_8 + x_1x_3x_5x_8 \\ & + x_1x_3x_5x_7x_8 + x_1x_3x_5x_6 + x_1x_3x_5x_6x_7 + x_1x_3x_5x_6x_7x_8 + x_1x_3x_4x_6 + x_1x_3x_4x_6x_8 + x_1x_3x_4x_6x_7 \\ & + x_1x_3x_4x_6x_7x_8 + x_1x_3x_4x_5 + x_1x_3x_4x_5x_7 + x_1x_3x_4x_5x_6x_7x_8 + x_1x_2x_5x_8 + x_1x_2x_5x_7x_8 + x_1x_2x_5x_6x_8 \\ & + x_1x_2x_5x_6x_7x_8 + x_1x_2x_4 + x_1x_2x_4x_8 + x_1x_2x_4x_6 + x_1x_2x_4x_6x_8 + x_1x_2x_4x_5 + x_1x_2x_4x_5x_7x_8 \\ & + x_1x_2x_4x_5x_6 + x_1x_2x_3 + x_1x_2x_3x_8 + x_1x_2x_3x_7 + x_1x_2x_3x_7x_8 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_8 \\ & + x_1x_2x_3x_5x_7 + x_1x_2x_3x_5x_7x_8 + x_1x_2x_3x_5x_6x_8 + x_1x_2x_3x_4 + x_1x_2x_3x_4x_8 + x_1x_2x_3x_4x_6x_7 \\ & + x_1x_2x_3x_4x_6x_7x_8 + x_1x_2x_3x_4x_5x_8 + x_1x_2x_3x_4x_5x_7x_8 + x_1x_2x_3x_4x_5x_6 + x_1x_2x_3x_4x_5x_6x_7 \end{aligned} \tag{1.52}$$

$$\begin{aligned} y_5 = {} & x_8 + x_7x_8 + x_6 + x_6x_7 + x_6x_7x_8 + x_5 + x_5x_7 + x_5x_6 + x_5x_6x_8 + x_4 + x_4x_7x_8 + x_4x_6 + x_4x_6x_7 + x_4x_5 \\ & + x_4x_5x_8 + x_4x_5x_7x_8 + x_4x_5x_6 + x_4x_5x_6x_8 + x_3x_8 + x_3x_7 + x_3x_7x_8 + x_3x_6 + x_3x_6x_8 + x_3x_6x_7 \\ & + x_3x_6x_7x_8 + x_3x_5 + x_3x_5x_7 + x_3x_5x_6 + x_3x_5x_6x_7x_8 + x_3x_4x_8 + x_3x_4x_7 + x_3x_4x_6x_7 + x_3x_4x_6x_7x_8 \\ & + x_3x_4x_5x_8 + x_2x_8 + x_2x_7 + x_2x_7x_8 + x_2x_6x_8 + x_2x_6x_7 + x_2x_6x_7x_8 + x_2x_5x_8 + x_2x_5x_7x_8 \\ & + x_2x_5x_6x_7x_8 + x_2x_4 + x_2x_4x_7 + x_2x_4x_6 + x_2x_4x_6x_7 + x_2x_4x_6x_7x_8 + x_2x_4x_5 + x_2x_4x_5x_8 + x_2x_4x_5x_6 \\ & + x_2x_4x_5x_6x_7 + x_2x_3x_8 + x_2x_3x_6x_7 + x_2x_3x_5 + x_2x_3x_5x_8 + x_2x_3x_4x_6 + x_2x_3x_4x_6x_8 + x_2x_3x_4x_6x_7 \\ & + x_2x_3x_4x_6x_7x_8 + x_2x_3x_4x_5 + x_2x_3x_4x_5x_7x_8 + x_2x_3x_4x_5x_6x_8 + x_1x_8 + x_1x_7 + x_1x_7x_8 + x_1x_6 \\ & + x_1x_6x_8 + x_1x_6x_7 + x_1x_6x_7x_8 + x_1x_5x_6 + x_1x_5x_6x_7 + x_1x_4x_7 + x_1x_4x_7x_8 + x_1x_4x_5 + x_1x_4x_5x_7x_8 \\ & + x_1x_3x_7 + x_1x_3x_7x_8 + x_1x_3x_6 + x_1x_3x_6x_8 + x_1x_3x_5 + x_1x_3x_5x_6 + x_1x_3x_5x_6x_8 + x_1x_3x_4 + x_1x_3x_4x_8 \\ & + x_1x_3x_4x_7 + x_1x_3x_4x_7x_8 + x_1x_3x_4x_6x_7 + x_1x_3x_4x_6x_7x_8 + x_1x_3x_4x_5 + x_1x_3x_4x_5x_8 + x_1x_3x_4x_5x_7 \\ & + x_1x_3x_4x_5x_6 + x_1x_3x_4x_5x_6x_8 + x_1x_2x_5x_8 + x_1x_2x_5x_7 + x_1x_2x_5x_6 + x_1x_2x_5x_6x_8 + x_1x_2x_4x_6x_7 \\ & + x_1x_2x_4x_6x_7x_8 + x_1x_2x_4x_5x_7 + x_1x_2x_4x_5x_7x_8 + x_1x_2x_4x_5x_6 + x_1x_2x_3 + x_1x_2x_3x_8 + x_1x_2x_3x_6x_7 \\ & + x_1x_2x_3x_6x_7x_8 + x_1x_2x_3x_5 + x_1x_2x_3x_5x_8 + x_1x_2x_3x_5x_7 + x_1x_2x_3x_5x_6 + x_1x_2x_3x_5x_6x_8 + x_1x_2x_3x_4x_7 \\ & + x_1x_2x_3x_4x_7x_8 + x_1x_2x_3x_4x_6x_8 + x_1x_2x_3x_4x_6x_7x_8 + x_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5x_8 + x_1x_2x_3x_4x_5x_7x_8 \end{aligned} \tag{1.53}$$

$$\begin{aligned} y_6 = {} & x_8 + x_7x_8 + x_6 + x_6x_8 + x_6x_7x_8 + x_5x_8 + x_5x_7 + x_5x_7x_8 + x_5x_6 + x_5x_6x_8 + x_5x_6x_7x_8 + x_4x_7x_8 \\ & + x_4x_6 + x_4x_6x_8 + x_4x_5 + x_4x_5x_8 + x_4x_5x_7 + x_4x_5x_7x_8 + x_4x_5x_6x_8 + x_3x_7x_8 + x_3x_6 + x_3x_6x_8 \\ & + x_3x_6x_7x_8 + x_3x_5x_6 + x_3x_5x_6x_8 + x_3x_5x_6x_7x_8 + x_3x_4 + x_3x_4x_8 + x_3x_4x_7x_8 + x_3x_4x_6 + x_3x_4x_6x_8 \\ & + x_3x_4x_5x_7 + x_3x_4x_5x_7x_8 + x_3x_4x_5x_6x_8 + x_2x_8 + x_2x_7 + x_2x_6x_8 + x_2x_5x_8 + x_2x_5x_6 + x_2x_5x_6x_8 \\ & + x_2x_4x_8 + x_2x_4x_7 + x_2x_4x_7x_8 + x_2x_4x_6 + x_2x_4x_6x_8 + x_2x_4x_6x_7 + x_2x_4x_5x_8 + x_2x_4x_5x_7x_8 \\ & + x_2x_4x_5x_6 + x_2x_3 + x_2x_3x_7 + x_2x_3x_7x_8 + x_2x_3x_5x_7x_8 + x_2x_3x_5x_6 + x_2x_3x_5x_6x_8 + x_2x_3x_5x_6x_7x_8 \\ & + x_2x_3x_4x_6 + x_2x_3x_4x_6x_7 + x_2x_3x_4x_6x_7x_8 + x_2x_3x_4x_5 + x_2x_3x_4x_5x_6 + x_2x_3x_4x_5x_6x_7 + x_1x_7 + x_1x_6 \\ & + x_1x_6x_8 + x_1x_5x_8 + x_1x_5x_6 + x_1x_4x_8 + x_1x_4x_7 + x_1x_4x_7x_8 + x_1x_4x_6x_7 + x_1x_4x_5x_6 + x_1x_4x_5x_6x_8 \\ & + x_1x_4x_5x_6x_7 + x_1x_4x_5x_6x_7x_8 + x_1x_3x_7 + x_1x_3x_6 + x_1x_3x_6x_7 + x_1x_3x_5 + x_1x_3x_5x_6 + x_1x_3x_4 \\ & + x_1x_3x_4x_8 + x_1x_3x_4x_7 + x_1x_3x_4x_6x_8 + x_1x_3x_4x_6x_7x_8 + x_1x_3x_4x_5x_7x_8 + x_1x_3x_4x_5x_6x_8 + x_1x_2x_8 \\ & + x_1x_2x_7x_8 + x_1x_2x_6 + x_1x_2x_6x_7x_8 + x_1x_2x_5x_7 + x_1x_2x_5x_6 + x_1x_2x_5x_6x_8 + x_1x_2x_4x_8 + x_1x_2x_4x_6 \\ & + x_1x_2x_4x_6x_7 + x_1x_2x_4x_6x_7x_8 + x_1x_2x_4x_5x_8 + x_1x_2x_4x_5x_7 + x_1x_2x_4x_5x_6x_7 + x_1x_2x_4x_5x_6x_7x_8 \\ & + x_1x_2x_3x_8 + x_1x_2x_3x_6x_8 + x_1x_2x_3x_6x_7x_8 + x_1x_2x_3x_5x_7 + x_1x_2x_3x_5x_6x_8 + x_1x_2x_3x_4 \end{aligned} \tag{1.54}$$

$$\begin{aligned} y_7 = {} & x_7 + x_6 + x_6x_8 + x_5x_8 + x_5x_6 + x_4x_8 + x_4x_7 + x_4x_7x_8 + x_4x_6x_7 + x_4x_5x_6 + x_4x_5x_6x_8 + x_4x_5x_6x_7 \\ & + x_4x_5x_6x_7x_8 + x_3x_7 + x_3x_6 + x_3x_6x_7 + x_3x_5 + x_3x_5x_6 + x_3x_4 + x_3x_4x_8 + x_3x_4x_7 + x_3x_4x_6x_8 \\ & + x_3x_4x_6x_7x_8 + x_3x_4x_5x_7x_8 + x_3x_4x_5x_6x_8 + x_2x_8 + x_2x_7x_8 + x_2x_6 + x_2x_6x_7x_8 + x_2x_5x_7 + x_2x_5x_6 \\ & + x_2x_5x_6x_8 + x_2x_4x_8 + x_2x_4x_6 + x_2x_4x_6x_7 + x_2x_4x_6x_7x_8 + x_2x_4x_5x_8 + x_2x_4x_5x_7 + x_2x_4x_5x_6x_7 \\ & + x_2x_4x_5x_6x_7x_8 + x_2x_3x_8 + x_2x_3x_6x_8 + x_2x_3x_6x_7x_8 + x_2x_3x_5x_7 + x_2x_3x_5x_6x_8 + x_2x_3x_4 \end{aligned} \tag{1.55}$$

Note that the term x_1 is not present in the ANF of y_7 (Equation 1.55).

$$\begin{aligned} y_8 = {} & 1 + x_7 + x_7x_8 + x_6 + x_6x_8 + x_6x_7x_8 + x_5 + x_5x_8 + x_5x_7x_8 + x_5x_6 + x_5x_6x_7x_8 + x_4 + x_4x_8 + x_4x_7x_8 \\ & + x_4x_6x_8 + x_4x_6x_7x_8 + x_4x_5x_8 + x_4x_5x_7x_8 + x_4x_5x_6x_8 + x_3 + x_3x_8 + x_3x_7 + x_3x_5x_8 + x_3x_4x_7 \\ & + x_3x_4x_6x_8 + x_3x_4x_5 + x_3x_4x_5x_8 + x_3x_4x_5x_6 + x_3x_4x_5x_6x_8 + x_3x_4x_5x_6x_7 + x_3x_4x_5x_6x_7x_8 + x_2x_7 \\ & + x_2x_5x_7 + x_2x_5x_7x_8 + x_2x_4 + x_2x_4x_8 + x_2x_4x_7x_8 + x_2x_4x_6 + x_2x_4x_6x_7x_8 + x_2x_4x_5x_7 + x_2x_4x_5x_6 \\ & + x_2x_4x_5x_6x_8 + x_2x_3x_8 + x_2x_3x_6 + x_2x_3x_6x_7 + x_2x_3x_6x_7x_8 + x_2x_3x_5x_8 + x_2x_3x_5x_7 + x_2x_3x_5x_6x_7 \\ & + x_2x_3x_5x_6x_7x_8 + x_2x_3x_4x_8 + x_2x_3x_4x_6x_8 + x_2x_3x_4x_6x_7x_8 + x_2x_3x_4x_5x_7 + x_2x_3x_4x_5x_6x_8 + x_1 \end{aligned} \tag{1.56}$$

We can note that x_1 is present only once in the ANF of y_8 (Equation 1.56), and is not multiplied by other terms. This shows that y_8 depends linearly on x_1.

The SAFER S-Box is bijective, hence we can also obtain ANFs for its inverse. For the inverse, variables from x_1 to x_8 are the outputs, and variables from y_1 to y_8 are the inputs. The ANFs for the inverse are shown below, from Equation 1.57 to Equation 1.64.

$$\begin{aligned} x_1 = {} & 1 + y_8 + y_6 + y_6y_7 + y_6y_7y_8 + y_5 + y_5y_8 + y_5y_7 + y_5y_6 + y_5y_6y_8 + y_5y_6y_7 + y_4y_6y_8 + y_4y_6y_7y_8 \\ & + y_4y_5y_8 + y_4y_5y_7 + y_4y_5y_7y_8 + y_4y_5y_6y_8 + y_4y_5y_6y_7 + y_4y_5y_6y_7y_8 + y_3 + y_3y_7 + y_3y_7y_8 + y_3y_6 \\ & + y_3y_5y_6y_8 + y_3y_5y_6y_7 + y_3y_5y_6y_7y_8 + y_3y_4 + y_3y_4y_8 + y_3y_4y_7 + y_3y_4y_6 + y_3y_4y_6y_8 + y_3y_4y_6y_7y_8 \\ & + y_3y_4y_5y_7 + y_3y_4y_5y_7y_8 + y_3y_4y_5y_6y_7 + y_3y_4y_5y_6y_7y_8 + y_2 + y_2y_7 + y_2y_7y_8 + y_2y_6y_7y_8 + y_2y_5 \\ & + y_2y_5y_8 + y_2y_5y_7y_8 + y_2y_5y_6 + y_2y_5y_6y_7 + y_2y_5y_6y_7y_8 + y_2y_4 + y_2y_4y_7y_8 + y_2y_4y_6y_7 + y_2y_4y_5y_7 \\ & + y_2y_4y_5y_6y_7y_8 + y_2y_3 + y_2y_3y_7 + y_2y_3y_7y_8 + y_2y_3y_6y_7 + y_2y_3y_6y_7y_8 + y_2y_3y_5 + y_2y_3y_5y_8 + y_2y_3y_5y_7y_8 \\ & + y_2y_3y_5y_6 + y_2y_3y_5y_6y_8 + y_2y_3y_5y_6y_7y_8 + y_2y_3y_4 + y_2y_3y_4y_8 + y_2y_3y_4y_7 + y_2y_3y_4y_6 + y_2y_3y_4y_6y_8 \\ & + y_2y_3y_4y_5y_7y_8 + y_2y_3y_4y_5y_6y_7 + y_1y_7 + y_1y_7y_8 + y_1y_6y_8 + y_1y_5y_8 + y_1y_5y_7 + y_1y_5y_7y_8 + y_1y_5y_6y_8 \\ & + y_1y_5y_6y_7 + y_1y_4y_6y_7 + y_1y_4y_6y_7y_8 + y_1y_4y_5y_7 + y_1y_3y_7 + y_1y_3y_6y_8 + y_1y_3y_5 + y_1y_3y_5y_8 + y_1y_3y_5y_6y_8 \\ & + y_1y_3y_4y_8 + y_1y_3y_4y_7 + y_1y_3y_4y_6y_8 + y_1y_3y_4y_6y_7 + y_1y_3y_4y_6y_7y_8 + y_1y_3y_4y_5 + y_1y_3y_4y_5y_6 \\ & + y_1y_3y_4y_5y_6y_7 + y_1y_2y_7 + y_1y_2y_7y_8 + y_1y_2y_6 + y_1y_2y_6y_8 + y_1y_2y_6y_7 + y_1y_2y_6y_7y_8 + y_1y_2y_5 \\ & + y_1y_2y_5y_7y_8 + y_1y_2y_5y_6y_7y_8 + y_1y_2y_4 + y_1y_2y_4y_8 + y_1y_2y_4y_7y_8 + y_1y_2y_4y_6 + y_1y_2y_4y_6y_8 + y_1y_2y_4y_6y_7 \\ & + y_1y_2y_4y_6y_7y_8 + y_1y_2y_4y_5 + y_1y_2y_4y_5y_6 + y_1y_2y_4y_5y_6y_7 + y_1y_2y_3y_6y_7 + y_1y_2y_3y_6y_7y_8 + y_1y_2y_3y_5 \\ & + y_1y_2y_3y_5y_8 + y_1y_2y_3y_5y_6 + y_1y_2y_3y_5y_6y_8 + y_1y_2y_3y_5y_6y_7y_8 + y_1y_2y_3y_4 + y_1y_2y_3y_4y_6 + y_1y_2y_3y_4y_6y_7 \end{aligned} \tag{1.57}$$

$$\begin{aligned}
x_2 = {} & y_6 + y_6y_7 + y_6y_7y_8 + y_5y_7y_8 + y_5y_6y_8 + y_4 + y_4y_8 + y_4y_7y_8 + y_4y_6 + y_4y_5 + y_4y_5y_7 + y_4y_5y_6 \\
& + y_4y_5y_6y_7 + y_4y_5y_6y_7y_8 + y_3 + y_3y_6 + y_3y_6y_8 + y_3y_6y_7y_8 + y_3y_5y_8 + y_3y_5y_7 + y_3y_5y_7y_8 + y_3y_5y_6y_7 \\
& + y_3y_5y_6y_7y_8 + y_3y_4 + y_3y_4y_8 + y_3y_4y_7 + y_3y_4y_6y_8 + y_3y_4y_6y_7 + y_3y_4y_6y_7y_8 + y_3y_4y_5y_7y_8 + y_3y_4y_5y_6 \\
& + y_3y_4y_5y_6y_7y_8 + y_2y_6 + y_2y_6y_7 + y_2y_6y_7y_8 + y_2y_5y_7 + y_2y_5y_7y_8 + y_2y_5y_6 + y_2y_5y_6y_8 + y_2y_5y_6y_7 \\
& + y_2y_4 + y_2y_4y_8 + y_2y_4y_7 + y_2y_4y_6y_8 + y_2y_4y_6y_7 + y_2y_4y_5y_8 + y_2y_4y_5y_6 + y_2y_4y_5y_6y_8 + y_2y_4y_5y_6y_7y_8 \\
& + y_2y_3y_7 + y_2y_3y_7y_8 + y_2y_3y_6y_8 + y_2y_3y_6y_7y_8 + y_2y_3y_5y_8 + y_2y_3y_5y_7 + y_2y_3y_5y_7y_8 + y_2y_3y_5y_6y_8 \\
& + y_2y_3y_5y_6y_7 + y_2y_3y_5y_6y_7y_8 + y_2y_3y_4y_7 + y_2y_3y_4y_6y_7 + y_2y_3y_4y_6y_7y_8 + y_2y_3y_4y_5 + y_2y_3y_4y_5y_8 \\
& + y_2y_3y_4y_5y_6 + y_2y_3y_4y_5y_6y_8 + y_2y_3y_4y_5y_6y_7y_8 + y_1 + y_1y_7y_8 + y_1y_6 + y_1y_5y_8 + y_1y_5y_7 + y_1y_5y_6y_8 \\
& + y_1y_5y_6y_7y_8 + y_1y_4 + y_1y_4y_7 + y_1y_4y_7y_8 + y_1y_4y_5 + y_1y_4y_5y_7y_8 + y_1y_4y_5y_6 + y_1y_4y_5y_6y_7y_8 + y_1y_3 \\
& + y_1y_3y_7y_8 + y_1y_3y_6y_7 + y_1y_3y_6y_7y_8 + y_1y_3y_5 + y_1y_3y_5y_8 + y_1y_3y_5y_7 + y_1y_3y_5y_7y_8 + y_1y_3y_5y_6 \\
& + y_1y_3y_5y_6y_8 + y_1y_3y_4y_7 + y_1y_3y_4y_6 + y_1y_3y_4y_5 + y_1y_3y_4y_5y_8 + y_1y_3y_4y_5y_7 + y_1y_3y_4y_5y_7y_8 \\
& + y_1y_3y_4y_5y_6 + y_1y_3y_4y_5y_6y_8 + y_1y_3y_4y_5y_6y_7 + y_1y_3y_4y_5y_6y_7y_8 + y_1y_2 + y_1y_2y_7y_8 + y_1y_2y_6y_7 \\
& + y_1y_2y_6y_7y_8 + y_1y_2y_5 + y_1y_2y_5y_7y_8 + y_1y_2y_5y_6 + y_1y_2y_5y_6y_7 + y_1y_2y_4y_7 + y_1y_2y_4y_6 + y_1y_2y_4y_6y_7y_8 \\
& + y_1y_2y_4y_5y_8 + y_1y_2y_4y_5y_7 + y_1y_2y_4y_5y_6y_8 + y_1y_2y_4y_5y_6y_7 + y_1y_2y_4y_5y_6y_7y_8 + y_1y_2y_3 + y_1y_2y_3y_6y_7y_8 \\
& + y_1y_2y_3y_5y_8 + y_1y_2y_3y_5y_6y_8 + y_1y_2y_3y_5y_6y_7 + y_1y_2y_3y_5y_6y_7y_8 + y_1y_2y_3y_4 + y_1y_2y_3y_4y_6 + y_1y_2y_3y_4y_6y_7
\end{aligned} \tag{1.58}$$

$$\begin{aligned}
x_3 = {} & y_7 + y_7y_8 + y_6 + y_6y_7 + y_6y_7y_8 + y_5y_7 + y_5y_6y_7 + y_4y_8 + y_4y_7 + y_4y_7y_8 + y_4y_6 + y_4y_6y_8 + y_4y_6y_7 \\
& + y_4y_5y_8 + y_4y_5y_7y_8 + y_4y_5y_6y_8 + y_4y_5y_6y_7 + y_4y_5y_6y_7y_8 + y_3 + y_3y_7 + y_3y_7y_8 + y_3y_6 + y_3y_6y_7y_8 \\
& + y_3y_5y_7 + y_3y_5y_7y_8 + y_3y_5y_6 + y_3y_5y_6y_7 + y_3y_5y_6y_7y_8 + y_3y_4 + y_3y_4y_7 + y_3y_4y_6 + y_3y_4y_6y_8 + y_3y_4y_5y_8 \\
& + y_3y_4y_5y_7 + y_3y_4y_5y_6 + y_3y_4y_5y_6y_7 + y_2 + y_2y_7y_8 + y_2y_5y_8 + y_2y_5y_7y_8 + y_2y_5y_6y_7 + y_2y_4y_8 \\
& + y_2y_4y_7 + y_2y_4y_7y_8 + y_2y_4y_6y_8 + y_2y_4y_6y_7 + y_2y_4y_5y_8 + y_2y_4y_5y_7y_8 + y_2y_4y_5y_6 + y_2y_4y_5y_6y_7y_8 \\
& + y_2y_3 + y_2y_3y_8 + y_2y_3y_6y_8 + y_2y_3y_6y_7 + y_2y_3y_5 + y_2y_3y_5y_8 + y_2y_3y_5y_6y_8 + y_2y_3y_4 + y_2y_3y_4y_7y_8 \\
& + y_2y_3y_4y_5y_7y_8 + y_2y_3y_4y_5y_6y_8 + y_2y_3y_4y_5y_6y_7 + y_1y_7 + y_1y_6 + y_1y_6y_8 + y_1y_4y_6 + y_1y_4y_6y_7 + y_1y_4y_5 \\
& + y_1y_4y_5y_8 + y_1y_4y_5y_7 + y_1y_4y_5y_7y_8 + y_1y_3 + y_1y_3y_8 + y_1y_3y_7 + y_1y_3y_7y_8 + y_1y_3y_6 + y_1y_3y_6y_7y_8 \\
& + y_1y_3y_4y_7 + y_1y_3y_4y_7y_8 + y_1y_3y_4y_6 + y_1y_3y_4y_6y_7y_8 + y_1y_3y_4y_5y_6y_8 + y_1y_2y_7 + y_1y_2y_7y_8 + y_1y_2y_6 \\
& + y_1y_2y_6y_8 + y_1y_2y_6y_7 + y_1y_2y_5y_7 + y_1y_2y_5y_7y_8 + y_1y_2y_5y_6 + y_1y_2y_5y_6y_7y_8 + y_1y_2y_4y_8 + y_1y_2y_4y_7 \\
& + y_1y_2y_4y_6y_8 + y_1y_2y_4y_6y_7 + y_1y_2y_4y_6y_7y_8 + y_1y_2y_4y_5 + y_1y_2y_4y_5y_8 + y_1y_2y_4y_5y_6 + y_1y_2y_4y_5y_6y_7y_8 \\
& + y_1y_2y_3 + y_1y_2y_3y_6y_8 + y_1y_2y_3y_6y_7y_8 + y_1y_2y_3y_5y_7 + y_1y_2y_3y_5y_7y_8 + y_1y_2y_3y_5y_6 + y_1y_2y_3y_5y_6y_8 \\
& + y_1y_2y_3y_4y_6 + y_1y_2y_3y_4y_6y_8 + y_1y_2y_3y_4y_6y_7 + y_1y_2y_3y_4y_5y_7 + y_1y_2y_3y_4y_5y_7y_8 + y_1y_2y_3y_4y_5y_6
\end{aligned} \tag{1.59}$$

$$\begin{aligned}
x_4 = {} & y_7 + y_7y_8 + y_6y_7y_8 + y_5 + y_5y_7 + y_5y_6 + y_5y_6y_8 + y_4y_8 + y_4y_7 + y_4y_6y_8 + y_4y_6y_7 + y_4y_5y_8 + y_4y_5y_6 \\
& + y_4y_5y_6y_8 + y_4y_5y_6y_7 + y_4y_5y_6y_7y_8 + y_3 + y_3y_8 + y_3y_6y_8 + y_3y_6y_7 + y_3y_6y_7y_8 + y_3y_5 + y_3y_5y_7y_8 \\
& + y_3y_5y_6y_8 + y_3y_5y_6y_7 + y_3y_4 + y_3y_4y_8 + y_3y_4y_6 + y_3y_4y_6y_7 + y_3y_4y_6y_7y_8 + y_3y_4y_5y_7y_8 + y_3y_4y_5y_6y_8 \\
& + y_2y_6 + y_2y_6y_8 + y_2y_6y_7 + y_2y_6y_7y_8 + y_2y_5 + y_2y_5y_7y_8 + y_2y_5y_6y_7y_8 + y_2y_4y_8 + y_2y_4y_7 + y_2y_4y_6 \\
& + y_2y_4y_6y_7y_8 + y_2y_4y_5 + y_2y_4y_5y_8 + y_2y_4y_5y_6y_8 + y_2y_4y_5y_6y_7 + y_2y_3y_8 + y_2y_3y_6 + y_2y_3y_6y_8 + y_2y_3y_6y_7 \\
& + y_2y_3y_6y_7y_8 + y_2y_3y_5y_8 + y_2y_3y_5y_7 + y_2y_3y_5y_6y_7y_8 + y_2y_3y_4 + y_2y_3y_4y_8 + y_2y_3y_4y_7 + y_2y_3y_4y_6 \\
& + y_2y_3y_4y_6y_7y_8 + y_2y_3y_4y_5 + y_2y_3y_4y_5y_8 + y_2y_3y_4y_5y_7y_8 + y_2y_3y_4y_5y_6 + y_2y_3y_4y_5y_6y_8 + y_2y_3y_4y_5y_6y_7 \\
& + y_2y_3y_4y_5y_6y_7y_8 + y_1 + y_1y_7 + y_1y_7y_8 + y_1y_6 + y_1y_6y_7 + y_1y_5y_6y_8 + y_1y_4y_8 + y_1y_4y_7 + y_1y_4y_7y_8 \\
& + y_1y_4y_6 + y_1y_4y_6y_8 + y_1y_4y_5 + y_1y_4y_5y_7 + y_1y_4y_5y_6 + y_1y_4y_5y_6y_8 + y_1y_4y_5y_6y_7y_8 + y_1y_3 + y_1y_3y_8 \\
& + y_1y_3y_6y_7 + y_1y_3y_6y_7y_8 + y_1y_3y_5y_7 + y_1y_3y_5y_7y_8 + y_1y_3y_5y_6 + y_1y_3y_4y_8 + y_1y_3y_4y_6 + y_1y_3y_4y_6y_8 \\
& + y_1y_3y_4y_6y_7 + y_1y_3y_4y_6y_7y_8 + y_1y_3y_4y_5 + y_1y_3y_4y_5y_8 + y_1y_3y_4y_5y_6y_8 + y_1y_3y_4y_5y_6y_7 + y_1y_3y_4y_5y_6y_7y_8 \\
& + y_1y_2 + y_1y_2y_7y_8 + y_1y_2y_6 + y_1y_2y_6y_8 + y_1y_2y_6y_7 + y_1y_2y_5y_7y_8 + y_1y_2y_5y_6y_8 + y_1y_2y_5y_6y_7y_8 + y_1y_2y_4 \\
& + y_1y_2y_4y_7y_8 + y_1y_2y_4y_6y_8 + y_1y_2y_4y_6y_7 + y_1y_2y_4y_6y_7y_8 + y_1y_2y_4y_5 + y_1y_2y_4y_5y_7 + y_1y_2y_4y_5y_7y_8 \\
& + y_1y_2y_4y_5y_6 + y_1y_2y_3y_6y_7 + y_1y_2y_3y_5y_7 + y_1y_2y_3y_5y_7y_8 + y_1y_2y_3y_5y_6 + y_1y_2y_3y_4y_7y_8 + y_1y_2y_3y_4y_6
\end{aligned} \tag{1.60}$$

$$\begin{aligned}
x_5 = {} & y_7y_8 + y_6y_8 + y_6y_7 + y_5y_7 + y_5y_6 + y_5y_6y_8 + y_5y_6y_7y_8 + y_4y_8 + y_4y_6 + y_4y_6y_8 + y_4y_6y_7y_8 + y_4y_5 \\
& + y_4y_5y_8 + y_4y_5y_7 + y_4y_5y_7y_8 + y_4y_5y_6y_8 + y_4y_5y_6y_7 + y_4y_5y_6y_7y_8 + y_3y_8 + y_3y_7 + y_3y_6y_7y_8 + y_3y_5 \\
& + y_3y_5y_8 + y_3y_5y_7 + y_3y_5y_7y_8 + y_3y_5y_6y_7 + y_3y_5y_6y_7y_8 + y_3y_4 + y_3y_4y_7 + y_3y_4y_6 + y_3y_4y_6y_8 + y_3y_4y_6y_7 \\
& + y_3y_4y_6y_7y_8 + y_3y_4y_5y_7 + y_3y_4y_5y_6 + y_3y_4y_5y_6y_8 + y_3y_4y_5y_6y_7 + y_2y_8 + y_2y_7 + y_2y_7y_8 + y_2y_6 + y_2y_6y_8 \\
& + y_2y_6y_7y_8 + y_2y_5y_7y_8 + y_2y_5y_6y_7 + y_2y_5y_6y_7y_8 + y_2y_4 + y_2y_4y_8 + y_2y_4y_7 + y_2y_4y_6 + y_2y_4y_6y_7 \\
& + y_2y_4y_5y_6 + y_2y_4y_5y_6y_8 + y_2y_4y_5y_6y_7 + y_2y_4y_5y_6y_7y_8 + y_2y_3 + y_2y_3y_7y_8 + y_2y_3y_6 + y_2y_3y_6y_8 + y_2y_3y_5 \\
& + y_2y_3y_5y_7 + y_2y_3y_5y_6 + y_2y_3y_5y_6y_7 + y_2y_3y_4y_8 + y_2y_3y_4y_6 + y_2y_3y_4y_6y_7y_8 + y_2y_3y_4y_5 + y_2y_3y_4y_5y_8 \\
& + y_2y_3y_4y_5y_7 + y_2y_3y_4y_5y_7y_8 + y_2y_3y_4y_5y_6 + y_1y_7 + y_1y_7y_8 + y_1y_6 + y_1y_6y_7 + y_1y_5 + y_1y_5y_7 + y_1y_5y_6y_7 \\
& + y_1y_5y_6y_7y_8 + y_1y_4y_6y_7 + y_1y_4y_5y_8 + y_1y_4y_5y_6 + y_1y_4y_5y_6y_7 + y_1y_4y_5y_6y_7y_8 + y_1y_3 + y_1y_3y_8 + y_1y_3y_7 \\
& + y_1y_3y_6 + y_1y_3y_6y_8 + y_1y_3y_6y_7y_8 + y_1y_3y_5 + y_1y_3y_5y_8 + y_1y_3y_5y_7y_8 + y_1y_3y_5y_6 + y_1y_3y_5y_6y_8 \\
& + y_1y_3y_4y_7y_8 + y_1y_3y_4y_6y_8 + y_1y_3y_4y_6y_7y_8 + y_1y_3y_4y_5 + y_1y_3y_4y_5y_7 + y_1y_3y_4y_5y_7y_8 + y_1y_3y_4y_5y_6 \\
& + y_1y_3y_4y_5y_6y_8 + y_1y_3y_4y_5y_6y_7 + y_1y_3y_4y_5y_6y_7y_8 + y_1y_2 + y_1y_2y_7y_8 + y_1y_2y_6y_8 + y_1y_2y_6y_7 + y_1y_2y_5 \\
& + y_1y_2y_5y_8 + y_1y_2y_5y_7 + y_1y_2y_5y_6y_8 + y_1y_2y_5y_6y_7y_8 + y_1y_2y_4 + y_1y_2y_4y_8 + y_1y_2y_4y_7y_8 + y_1y_2y_4y_6 \\
& + y_1y_2y_4y_6y_7y_8 + y_1y_2y_4y_5 + y_1y_2y_4y_5y_8 + y_1y_2y_4y_5y_7y_8 + y_1y_2y_4y_5y_6 + y_1y_2y_3y_7 + y_1y_2y_3y_7y_8 \\
& + y_1y_2y_3y_6y_7y_8 + y_1y_2y_3y_5 + y_1y_2y_3y_5y_6y_8 + y_1y_2y_3y_5y_6y_7 + y_1y_2y_3y_5y_6y_7y_8 + y_1y_2y_3y_4 + y_1y_2y_3y_4y_7 \\
& + y_1y_2y_3y_4y_7y_8 + y_1y_2y_3y_4y_6 + y_1y_2y_3y_4y_6y_8 + y_1y_2y_3y_4y_6y_7 + y_1y_2y_3y_4y_6y_7y_8 + y_1y_2y_3y_4y_5
\end{aligned} \tag{1.61}$$

$$\begin{aligned}
x_6 = {} & y_6y_8 + y_5y_7 + y_5y_6y_8 + y_5y_6y_7y_8 + y_4y_7y_8 + y_4y_6 + y_4y_6y_8 + y_4y_6y_7y_8 + y_4y_5y_8 + y_4y_5y_7 + y_4y_5y_6y_7 \\
& + y_4y_5y_6y_7y_8 + y_3y_8 + y_3y_6y_7 + y_3y_6y_7y_8 + y_3y_5 + y_3y_5y_7 + y_3y_5y_7y_8 + y_3y_5y_6y_7 + y_3y_4y_8 + y_3y_4y_7 \\
& + y_3y_4y_6 + y_3y_4y_6y_7y_8 + y_3y_4y_5y_8 + y_3y_4y_5y_6 + y_3y_4y_5y_6y_8 + y_3y_4y_5y_6y_7 + y_2y_7 + y_2y_5y_8 + y_2y_5y_6 \\
& + y_2y_5y_6y_7 + y_2y_5y_6y_7y_8 + y_2y_4 + y_2y_4y_6 + y_2y_4y_6y_7 + y_2y_4y_5y_6 + y_2y_3y_7 + y_2y_3y_6 + y_2y_3y_6y_8 \\
& + y_2y_3y_5 + y_2y_3y_5y_8 + y_2y_3y_5y_7y_8 + y_2y_3y_5y_6y_7 + y_2y_3y_4y_8 + y_2y_3y_4y_7 + y_2y_3y_4y_7y_8 + y_2y_3y_4y_6y_7y_8 \\
& + y_2y_3y_4y_5 + y_2y_3y_4y_5y_8 + y_2y_3y_4y_5y_7 + y_2y_3y_4y_5y_7y_8 + y_2y_3y_4y_5y_6 + y_1y_7y_8 + y_1y_6 + y_1y_6y_7y_8 \\
& + y_1y_5y_7y_8 + y_1y_5y_6y_7y_8 + y_1y_4y_8 + y_1y_4y_7 + y_1y_4y_6y_8 + y_1y_4y_6y_7y_8 + y_1y_4y_5 + y_1y_4y_5y_8 + y_1y_4y_5y_6 \\
& + y_1y_4y_5y_6y_7 + y_1y_4y_5y_6y_7y_8 + y_1y_3 + y_1y_3y_6y_8 + y_1y_3y_6y_7y_8 + y_1y_3y_5 + y_1y_3y_5y_8 + y_1y_3y_5y_7y_8 \\
& + y_1y_3y_5y_6 + y_1y_3y_5y_6y_8 + y_1y_3y_4y_8 + y_1y_3y_4y_7y_8 + y_1y_3y_4y_6y_8 + y_1y_3y_4y_6y_7y_8 + y_1y_3y_4y_5 + y_1y_3y_4y_5y_8 \\
& + y_1y_2y_6 + y_1y_2y_6y_8 + y_1y_2y_5 + y_1y_2y_5y_8 + y_1y_2y_5y_7 + y_1y_2y_5y_6y_8 + y_1y_2y_5y_6y_7y_8 + y_1y_2y_4 + y_1y_2y_4y_8 \\
& + y_1y_2y_4y_7 + y_1y_2y_4y_6y_8 + y_1y_2y_4y_6y_7 + y_1y_2y_4y_5y_8 + y_1y_2y_4y_5y_7y_8 + y_1y_2y_4y_5y_6 + y_1y_2y_3y_7 + y_1y_2y_3y_6 \\
& + y_1y_2y_3y_6y_8 + y_1y_2y_3y_6y_7 + y_1y_2y_3y_5y_6y_8 + y_1y_2y_3y_5y_6y_7 + y_1y_2y_3y_5y_6y_7y_8 + y_1y_2y_3y_4 + y_1y_2y_3y_4y_7 \\
& + y_1y_2y_3y_4y_7y_8 + y_1y_2y_3y_4y_6 + y_1y_2y_3y_4y_6y_8 + y_1y_2y_3y_4y_6y_7 + y_1y_2y_3y_4y_6y_7y_8 + y_1y_2y_3y_4y_5
\end{aligned} \tag{1.62}$$

$$\begin{aligned}
x_7 = {} & y_6y_8 + y_6y_7y_8 + y_5y_8 + y_5y_7 + y_5y_6y_8 + y_5y_6y_7 + y_4y_7 + y_4y_7y_8 + y_4y_6 + y_4y_6y_8 + y_4y_5y_7 + y_4y_5y_6 \\
& + y_4y_5y_6y_8 + y_3y_6 + y_3y_6y_8 + y_3y_6y_7 + y_3y_5 + y_3y_5y_8 + y_3y_5y_7 + y_3y_5y_6y_8 + y_3y_5y_6y_7y_8 + y_3y_4y_8 \\
& + y_3y_4y_7y_8 + y_3y_4y_6 + y_3y_4y_6y_8 + y_3y_4y_5 + y_3y_4y_5y_7 + y_3y_4y_5y_7y_8 + y_3y_4y_5y_6y_8 + y_2y_6y_8 \\
& + y_2y_5 + y_2y_5y_7 + y_2y_5y_7y_8 + y_2y_5y_6 + y_2y_4 + y_2y_4y_8 + y_2y_4y_7 + y_2y_4y_6 + y_2y_4y_5y_8 + y_2y_4y_5y_7 \\
& + y_2y_4y_5y_6y_8 + y_2y_4y_5y_6y_7 + y_2y_3y_8 + y_2y_3y_7 + y_2y_3y_7y_8 + y_2y_3y_6y_7 + y_2y_3y_5 + y_2y_3y_5y_7 \\
& + y_2y_3y_5y_6y_7y_8 + y_2y_3y_4 + y_2y_3y_4y_6 + y_2y_3y_4y_6y_8 + y_2y_3y_4y_6y_7 + y_2y_3y_4y_5y_7 + y_1y_7y_8 + y_1y_5y_8 \\
& + y_1y_5y_7 + y_1y_5y_6y_8 + y_1y_5y_6y_7y_8 + y_1y_4 + y_1y_4y_8 + y_1y_4y_6 + y_1y_4y_6y_8 + y_1y_4y_6y_7 + y_1y_4y_5 \\
& + y_1y_4y_5y_7y_8 + y_1y_4y_5y_6y_8 + y_1y_4y_5y_6y_7y_8 + y_1y_3 + y_1y_3y_8 + y_1y_3y_7 + y_1y_3y_6 + y_1y_3y_5 + y_1y_3y_5y_8 \\
& + y_1y_3y_4y_7 + y_1y_3y_4y_6 + y_1y_3y_4y_5y_7 + y_1y_3y_4y_5y_7y_8 + y_1y_3y_4y_5y_6 + y_1y_2y_7 + y_1y_2y_7y_8 + y_1y_2y_6 \\
& + y_1y_2y_6y_8 + y_1y_2y_6y_7 + y_1y_2y_5y_8 + y_1y_2y_5y_6 + y_1y_2y_5y_6y_8 + y_1y_2y_5y_6y_7y_8 + y_1y_2y_4 + y_1y_2y_4y_8 \\
& + y_1y_2y_4y_6 + y_1y_2y_4y_5y_8 + y_1y_2y_4y_5y_6y_8 + y_1y_2y_4y_5y_6y_7 + y_1y_2y_4y_5y_6y_7y_8 + y_1y_2y_3 + y_1y_2y_3y_6y_8 \\
& + y_1y_2y_3y_6y_7y_8 + y_1y_2y_3y_5 + y_1y_2y_3y_5y_7 + y_1y_2y_3y_5y_7y_8 + y_1y_2y_3y_5y_6 + y_1y_2y_3y_4y_7y_8 + y_1y_2y_3y_4y_6
\end{aligned} \tag{1.63}$$

$$
\begin{aligned}
x_8 = {} & y_7y_8 + y_6y_8 + y_6y_7 + y_5y_7 + y_5y_6 + y_5y_6y_7y_8 + y_4y_6 + y_4y_5 + y_4y_5y_8 + y_4y_5y_7y_8 + y_4y_5y_6y_8 \\
& + y_4y_5y_6y_7 + y_3y_8 + y_3y_6y_8 + y_3y_5 + y_3y_5y_8 + y_3y_5y_6y_8 + y_3y_4 + y_3y_4y_7 + y_3y_4y_7y_8 + y_3y_4y_6y_7 \\
& + y_3y_4y_6y_7y_8 + y_3y_4y_5y_7 + y_3y_4y_5y_6 + y_3y_4y_5y_6y_8 + y_2y_8 + y_2y_7 + y_2y_7y_8 + y_2y_6y_8 + y_2y_5y_8 + y_2y_5y_7 \\
& + y_2y_4 + y_2y_4y_7 + y_2y_4y_7y_8 + y_2y_4y_5y_8 + y_2y_4y_5y_7 + y_2y_4y_5y_6y_7y_8 + y_2y_3 + y_2y_3y_6 + y_2y_3y_6y_7 \\
& + y_2y_3y_6y_7y_8 + y_2y_3y_5y_8 + y_2y_3y_5y_7y_8 + y_2y_3y_5y_6 + y_2y_3y_5y_6y_8 + y_2y_3y_5y_6y_7 + y_2y_3y_4y_8 + y_2y_3y_4y_6 \\
& + y_2y_3y_4y_6y_7y_8 + y_2y_3y_4y_5 + y_2y_3y_4y_5y_7 + y_1y_7 + y_1y_7y_8 + y_1y_6 + y_1y_6y_7 + y_1y_6y_7y_8 + y_1y_5y_7 \\
& + y_1y_5y_7y_8 + y_1y_5y_6y_8 + y_1y_4y_8 + y_1y_4y_7 + y_1y_4y_6 + y_1y_4y_6y_7y_8 + y_1y_4y_5y_8 + y_1y_4y_5y_7y_8 + y_1y_4y_5y_6y_8 \\
& + y_1y_4y_5y_6y_7y_8 + y_1y_3 + y_1y_3y_8 + y_1y_3y_7y_8 + y_1y_3y_6 + y_1y_3y_6y_8 + y_1y_3y_6y_7 + y_1y_3y_6y_7y_8 + y_1y_3y_5y_8 \\
& + y_1y_3y_4y_7 + y_1y_3y_4y_6 + y_1y_3y_4y_5y_8 + y_1y_3y_4y_5y_6y_8 + y_1y_3y_4y_5y_6y_7 + y_1y_3y_4y_5y_6y_7y_8 + y_1y_2 \\
& + y_1y_2y_7y_8 + y_1y_2y_6y_8 + y_1y_2y_6y_7y_8 + y_1y_2y_5 + y_1y_2y_5y_8 + y_1y_2y_5y_6 + y_1y_2y_5y_6y_7 + y_1y_2y_4y_8 \\
& + y_1y_2y_4y_7 + y_1y_2y_4y_7y_8 + y_1y_2y_4y_6y_8 + y_1y_2y_4y_6y_7 + y_1y_2y_4y_6y_7y_8 + y_1y_2y_4y_5 + y_1y_2y_4y_5y_7 \\
& + y_1y_2y_4y_5y_7y_8 + y_1y_2y_4y_5y_6 + y_1y_2y_3y_7 + y_1y_2y_3y_6y_8 + y_1y_2y_3y_6y_7y_8 + y_1y_2y_3y_5 + y_1y_2y_3y_5y_8 \\
& + y_1y_2y_3y_5y_6y_8 + y_1y_2y_3y_5y_6y_7 + y_1y_2y_3y_5y_6y_7y_8 + y_1y_2y_3y_4 + y_1y_2y_3y_4y_7y_8 + y_1y_2y_3y_4y_6
\end{aligned}
\tag{1.64}
$$

1.8 Comparing the ANFs

Tables 1.12 and 1.13 compare the S-Boxes here discussed with respect to their ANFs, presenting the total number of monomials and the degree of the Algebraic Normal Form of each bit.

For AES, all ANFs, both for the S-Box and its inverse, have degree 7. However, for SAFER, the expressions for bits y_7 and y_8 have degree 6, as well as a shorter amount of monomials. For DES, all ANFs have degree equal to 5.

AES			**SAFER**		
Output bit	**Monomials**	**Degree**	**Output bit**	**Monomials**	**Degree**
y_1	110	7	y_1	160	7
x_1	110	7	x_1	122	7
y_2	112	7	y_2	138	7
x_2	129	7	x_2	126	7
y_3	114	7	y_3	121	7
x_3	122	7	x_3	113	7
y_4	131	7	y_4	130	7
x_4	137	7	x_4	123	7
y_5	136	7	y_5	119	7
x_5	117	7	x_5	138	7
y_6	145	7	y_6	108	7
x_6	143	7	x_6	110	7
y_7	133	7	y_7	46	6
x_7	132	7	x_7	104	7
y_8	132	7	y_8	56	6
x_8	115	7	x_8	110	7

Table 1.12: Comparing AES and SAFER S-Box Algebraic Normal Forms

DES S_1			**DES S_3**		
Output bit	**Monomials**	**Degree**	**Output bit**	**Monomials**	**Degree**
y_1	27	5	y_1	31	5
y_2	33	5	y_2	29	5
y_3	38	5	y_3	40	5
y_4	29	5	y_4	20	5
DES S_2			**DES S_4**		
Output bit	**Monomials**	**Degree**	**Output bit**	**Monomials**	**Degree**
y_1	21	5	y_1	33	5
y_2	16	5	y_2	29	5
y_3	30	5	y_3	35	5
y_4	25	5	y_4	32	5
DES S_5			**DES S_7**		
Output bit	**Monomials**	**Degree**	**Output bit**	**Monomials**	**Degree**
y_1	30	5	y_1	27	5
y_2	22	5	y_2	19	5
y_3	41	5	y_3	28	5
y_4	35	5	y_4	18	5
DES S_6			**DES S_8**		
Output bit	**Monomials**	**Degree**	**Output bit**	**Monomials**	**Degree**
y_1	29	5	y_1	31	5
y_2	23	5	y_2	24	5
y_3	20	5	y_3	23	5
y_4	25	5	y_4	29	5

Table 1.13: Number of monomials and degrees for DES S-Boxes Algebraic Normal Forms

1.9 Extending the analysis to the whole cipher

In theory, we could represent e.g DES as a vectorial Boolean function from $\mathbb{F}_2^{64}$ to $\mathbb{F}_2^{64}$, and the ANFs of each output bit would directly reveal the relationship between ciphertext and plaintext, if we assume the key is set to zero in order to study only the plaintext diffusion. The same could be done for the other ciphers. In the case of AES, we would have a vectorial Boolean function of 128 variables. However, the time complexity for extracting the ANF of a single output bit is $O(n \cdot 2^n)$. Hence, for large n such as 64 and 128, the runtime becomes not practical considering current technological resources. For this reason, different approaches are needed to expand the diffusion analysis to the whole cipher.

1.10 Conclusions

In this work, we have analyzed the S-Boxes of three ciphers (DES, AES and SAFER), with focus on the diffusion property, obtaining the Algebraic Normal Forms of each output bit in order to assess whether they achieve complete diffusion or not. We have found out that DES and AES S-Boxes achieve complete diffusion, while the SAFER S-Box does not. Also, the ANFs for the seventh and the eighth bit in SAFER (y_7 and y_8) have degree 6, while the others have degree 7. Furthermore, the eighth bit of the SAFER S-Box depends only linearly on the first input bit, which does not affect security with respect to diffusion, but could be a sign of weak confusion. For the inverse SAFER S-Box, however, complete diffusion is achieved, the degrees are all equal to 7, and there is no linear dependence, which indicates the inverse S-Box is stronger with respect to confusion and diffusion.

Chapter 2

Analysis of diffusion in the DES cipher

In this chapter, we analyze complete diffusion in the DES cipher [58] with regards to plaintext and key. Both plaintext and key diffusion are complete after the 5-th round. The analysis is performed using a graph (see Section 2.3) to model the dependencies between plaintext/key bits and round output (state) bits.

We assume the reader is familiar with:

- the DES S-Box diffusion analysis conducted in Chapter 1,
- the DES cipher (expansion E, S-boxes, bit permutation P, Feistel Network structure) [58] (see Appendix A).

2.1 Notation

- G: a graph
- V: set of vertices of a graph
- E: set of edges of a graph
- $|V|, |E|$: cardinality (number of elements of) the sets V and E, respectively
- $A[i][j]$: element at row i, column j of the matrix A
- $\mathcal{A}$: adjacency-list representation of a graph
- $\mathcal{A}[v]$: adjacency list for the vertex v of a graph
- x_i^j: i-th state bit after round j of DES (note that, if $j = 0$, it is a plaintext bit)
- L_j: left half of DES state after j-th round, comprising bits from x_1^j to x_{32}^j
- R_j: right half of DES state after j-th round, comprising bits from x_{33}^j to x_{64}^j
- F: application of the DES F function
- K_j: j-th subkey of DES, used at the j-th round, containing 48 bits
- E: the expansion function E of DES
- P: the bit permutation P of DES
- $P[i]$: result of applying the DES permutation P to i (i ranges from 1 to 64)
- i_t: t-th input bit of an S-Box
- e_t: t-th input bit of the expansion function E

- S_1, S_2, ..., S_8: DES S-Boxes
- s_i^j: output bit of an S-Box, i is the bit position with respect to the DES state (from 1 to 64), j is the round (from 1 to 16)
- $PC1$, $PC2$: permutations used in the DES key schedule
- C_i, D_i, B: intermediate variables in the DES key scheduling algorithm
- $K_{j,z}$: z-th bit of the j-th round subkey of DES
- k_p: p-th bit of the DES master key

2.2 Acronyms

- DES: Data Encryption Standard
- ANF: Algebraic Normal Form
- S-Box: Substitution Box

2.3 Preliminaries

(Graph) *A graph $G = (V, E)$ is comprised of a set of vertices (also called nodes) V together with a set of edges E that connect vertices. Each edge is a tuple (v_1, v_2), meaning that vertices v_1 and v_2 are connected.*

(Adjacency matrix representation of a graph) *In an adjacency matrix representation, we assume the vertices of the graph $G = (V, E)$ are numbered, for instance, from 1 to $|V|$. The $|V|\times|V|$ matrix A such that $A[v_1][v_2] = 1$ if $(v_1, v_2) \in E$ (0 otherwise) is the adjacency matrix of the graph. In other words, it is a matrix where 1 means the vertices given by the row and the column are connected and 0 means they are not connected.*

(Adjacency-list representation of a graph) *The adjacency-list representation consists of an array of $\mathcal{A}$ of $|V|$ lists, one for each vertex. For each $v \in V$, the list $\mathcal{A}[v]$ contains all vertices u such that $(v, u) \in E$, i.e all the vertices to which v is connected.*

When two vertices are connected, we say they are *adjacent*, or that they are *neighbors*. For the plaintext diffusion analysis, we take advantage of the adjacency matrix representation. For the key diffusion analysis, we use adjacency lists.

Furthermore, two vertices v_1 and v_2 might not be directly connected to each other by an (v_1, v_2) edge, but there might exist a *path* from one to the other. There is a path from v_1 to v_2 if it is possible to reach v_2 from v_1. For example, if we have edges (v_1, v_3) and (v_3, v_2), v_1 and v_2 are not directly connected by an edge, but we can reach v_2 by going from v_1 to v_3 and then from v_3 to v_2. Therefore, there is a path from v_1 to v_2.

For the purposes of the analysis conducted in this chapter, these concepts are sufficient. However, for more details about graphs, algorithms and problems related to this data structure, the reader can refer to [16].

The core idea in order to analyze diffusion for the DES cipher is to represent each state bit at each round as a node (vertex) in a graph and use edges (and paths) to represent dependence between them and plaintext (or key) bits.

2.4 Plaintext diffusion

Let x_i^j denote the i-th state bit after round j of DES. At any given round j, we have $x_1^j, x_2^j, ..., x_{64}^j$, which is input to round $j + 1$. For each state bit, a node is created on a graph, and edges track dependence between them. In this way, we can track how many plaintext bits, i.e bits with $j = 0$, influence other bits throughout the rounds of DES.

The DES transformations occur as follows.

$$L_j = R_{j-1},$$

$$R_j = L_{j-1} \oplus F(R_{j-1}, K_j).$$

For iteration 1, for instance,

$$L_1 = R_0,$$

$$R_1 = L_0 \oplus F(R_0, K_0).$$

Before any of the transformations take place, the plaintext is $x_1^0, x_2^0, x_3^0, ..., x_{64}^0$, where the left half ($L_0$) comprises the bits from x_1^0 to x_{32}^0 and the right half (R_0) starts at x_{33}^0 and ends at x_{64}^0.

For round 1, the right half of round 0 is copied to become the left half ($L_1 = R_0$), thus we create edges between nodes x_1^1 and x_{33}^0, between x_2^1 and x_{34}^0, and so forth, respectively, with the edge between x_{32}^1 and x_{64}^0 being the last one created to represent the dependence incurred by the $L_1 = R_0$ step. Figure 2.1 shows the nodes and edges related to this process. White nodes are plaintext nodes, red nodes are round 1 nodes.

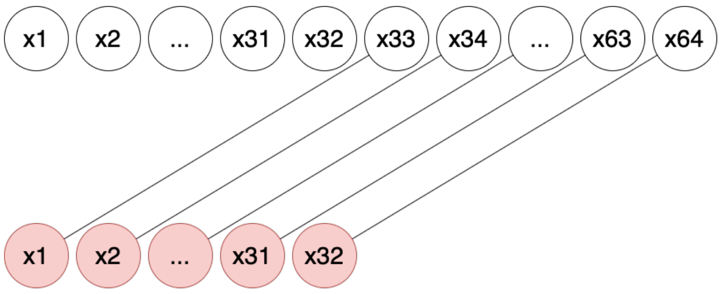


Figure 2.1: Diffusion incurred by $L_1 = R_0$. Each bit of the left half depends on a corresponding bit on the right half of the plaintext.

Regarding $R_1 = L_0 \oplus F(R_0, K_0)$, there is dependence between bits of R_1 and bits of L_0, thus we need edges between x_{33}^1 and x_1^0, between x_{34}^1 and x_2^0, and so forth, respectively, with the edge between x_{64}^1 and x_{32}^0 being the last. Figure 2.2 illustrates this.

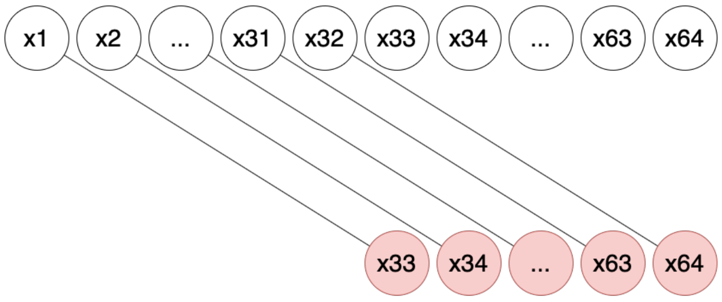


Figure 2.2: Diffusion incurred by the XOR with L_0 when $R_1 = L_0 \oplus F(R_0, K_0)$. Each bit of the right half depends on the corresponding bit on the left half of the plaintext.

These edges keep track of the dependence caused by the XOR function and the Feistel structure only (see Figure 2.3). In order to analyze how the F function provides diffusion, we must consider each step of it: the expansion function E, the S-boxes, and the permutation P. Furthermore, for the moment, we ignore the key, assuming each $K_i = 0$, so that we can focus only on plaintext diffusion. Key diffusion is addressed in Section 2.5.

2.4.1 The S-boxes

From the S-Box analysis conducted with the ANFs in Chapter 1, we know that each S-box takes 6 input bits and completely diffuses them over the 4 output bits. Therefore, if $i_1, i_2, i_3, i_4, i_5, i_6$ are the inputs to an S-box and y_1, y_2, y_3, y_4 are the outputs, each y_j depends on all of the i_t, for $1 \leq j \leq 4$ and $1 \leq t \leq 6$, since the S-box achieves complete diffusion.

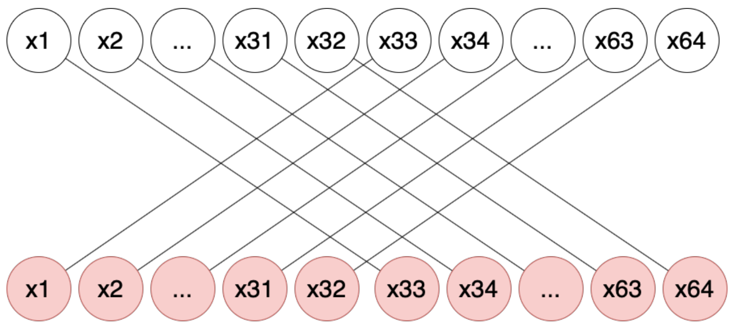


Figure 2.3: Diffusion incurred by $R_1 = L_0$ and by the XOR with L_0 when $R_1 = L_0 \oplus F(R_0, K_0)$, shown together

2.4.2 The E expansion function

Given 32 input bits $e_1, e_2, e_3, ..., e_{32}$, the E function expands them to 48 bits. This is done by appending repeated bits to the left and to the right of each 4-bit slice, according to Table 2.1. For the first slice, from e_1 to e_4, e_{32} is appended to the left and e_5 is appended to the right. For the second slice, e_4 is appended to the left and e_9 is appended to the right, and so forth. For the last slice, e_{28} is appended to the left and e_1 is appended to the right. The output is thus $e_{32}, e_1, e_2, e_3, e_4, e_5, e_4, e_5, e_6, e_7, e_8, e_9, e_8, e_9, ..., e_{30}, e_{31}, e_{32}, e_1$.

In general, for each 4-bit slice of the original input starting at e_k and ending at e_p, e_{p+1} is appended to the right, and e_{k-1} is appended to the left. If $p + 1 = 33$, then e_1 is appended. If $k - 1 = 0$, then e_{32} is appended.

E					
32	**1**	**2**	**3**	**4**	5
4	**5**	**6**	**7**	**8**	9
8	**9**	**10**	**11**	**12**	13
12	**13**	**14**	**15**	**16**	17
16	**17**	**18**	**19**	**20**	21
20	**21**	**22**	**23**	**24**	25
24	**25**	**26**	**27**	**28**	29
28	**29**	**30**	**31**	**32**	1

Table 2.1: The DES expansion function

For the first round of DES, R_0 is the input to the expansion function E, thus bits from x_{33} to x_{64} are expanded.

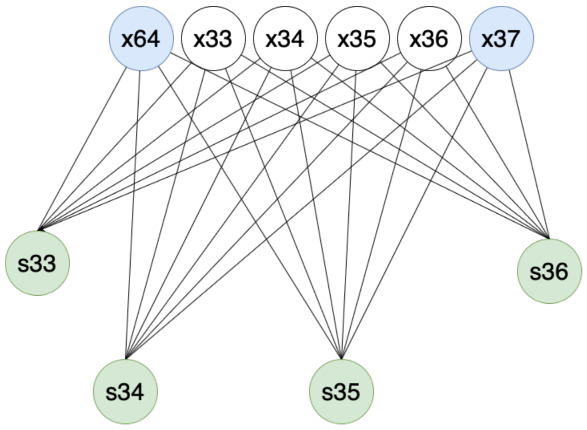
2.4.3 Expansion function and S-box

After the expansion, these bits are fed into the S-boxes. S-box S_1 gets $x^0_{33}, x^0_{34}, x^0_{35}$ and x^0_{36} as its input, i.e, the first 4-bit slice of the right half, and also x^0_{64} and x^0_{37}, according to the expansion E. S_2 gets $x^0_{37}, x^0_{38}, x^0_{39}, x^0_{40}$ (the second slice), and also x_{36} and x_{41}, because of the expansion function. In general, each S-box gets the current 4-bit slice bits as input plus the two extension bits.

If we denote S-box output bits by s^j_i, where i is the bit position at the state and j is the round, we can use the ANFs to track dependencies between s^1_i and plaintext bits. If a bit shows up at the ANF, an edge should be created for it in the graph. There should thus be edges between s^1_{33} and $x^0_{33}, x^0_{34}, x^0_{35}, x^0_{36}, x^0_{64}$ and x^0_{37}. There should be edges between s_{34} and $x^0_{33}, x^0_{34}, x^0_{35}, x^0_{36}, x^0_{64}$ and x^0_{37}, and so forth. For each s^1_i, for i between 33 and 64, similar analysis applies, since the ANFs completely diffuse the input bits. Figures 2.4 to 2.7 illustrate this.

2.4.4 The permutation P

After the S-boxes, the permutation P is performed and this completes the F function of DES. We thus need edges between s^j_i bits (S-box results) and x^1_i bits (the bits after the F function has finally been completely applied to R_0 and the results have been incorporated to the round 1 output).



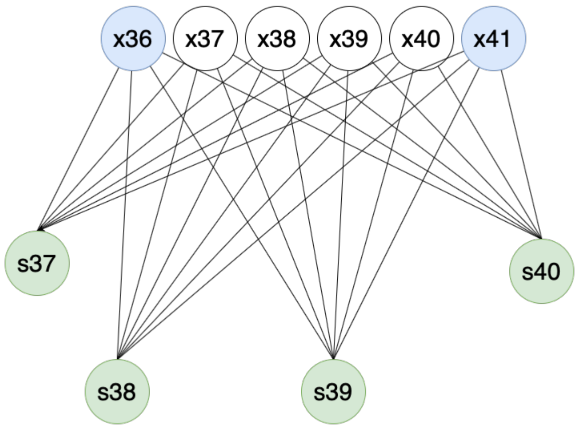
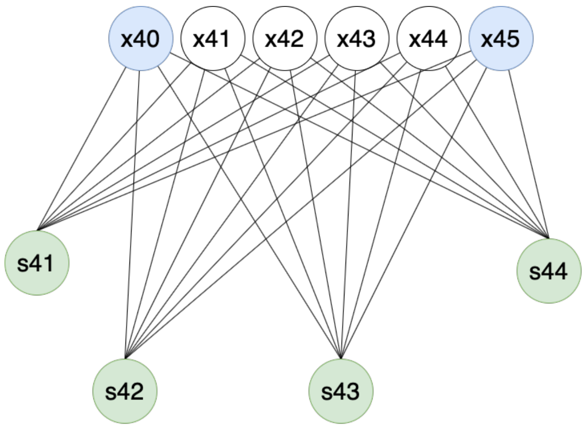


Figure 2.4: Graph edges for expansion function and S-box (S_1 and S_2) analysis. Each S-box output depends on all the six input bits, which are the plaintext bits of the current slice being considered plus the expansion function bits (blue).



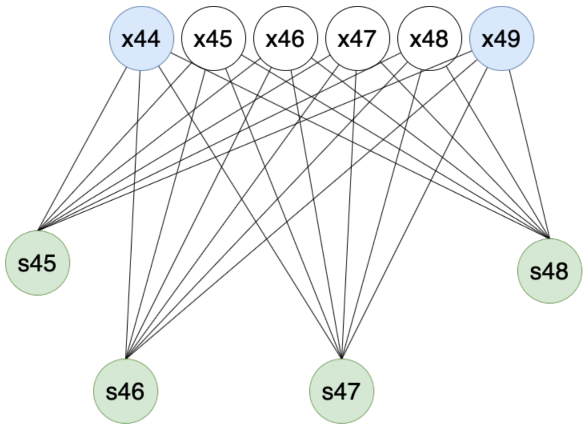
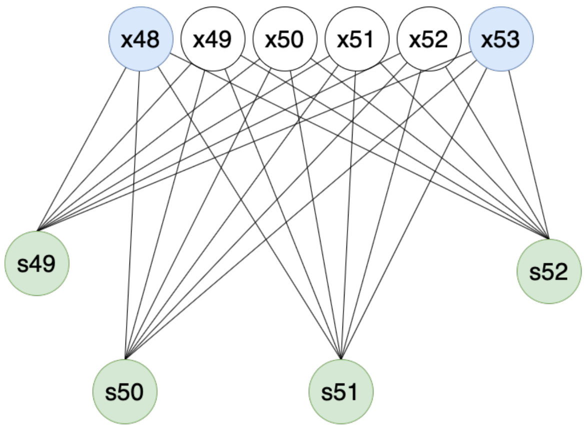


Figure 2.5: Graph edges for expansion function and S-box (S_3 and S_4) analysis, part 2



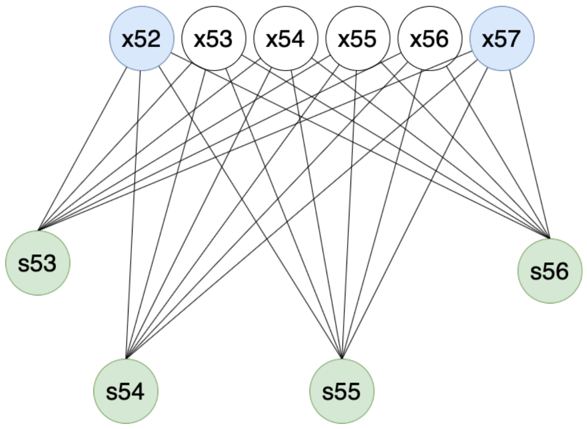


Figure 2.6: Graph edges for expansion function and S-box (S_5 and S_6) analysis, part 3

What happens is, thus, that x^1_{33} will actually depend on the sixteenth bit of the S-boxes output, which is s^1_{48}, due to the permutation P (see Table 2.2). In general, x^j_i depends on $s^j_{P[i-32]+32}$. Figures 2.8 to 2.11 illustrate this.

Once edges between x^1_i nodes and s^1_i nodes have been created, the effect of F has been completely tracked with regards to diffusion on the first round of DES. A state bit x^1_i is dependent on a plaintext bit x^0_k if there is a path from x^1_i to x^0_k in the graph.

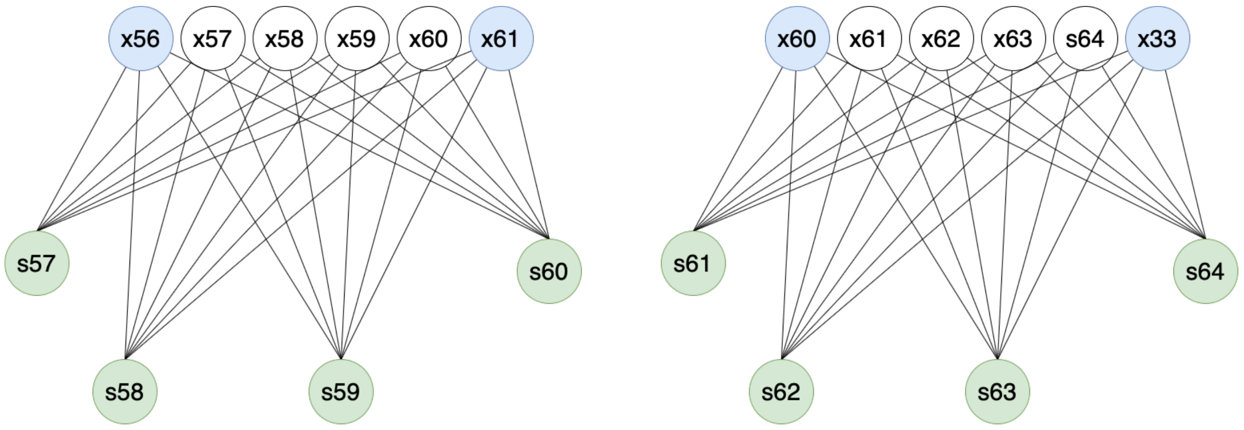


Figure 2.7: Graph edges for expansion function and S-box (S_7 and S_8) analysis, part 4

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Table 2.2: The P permutation

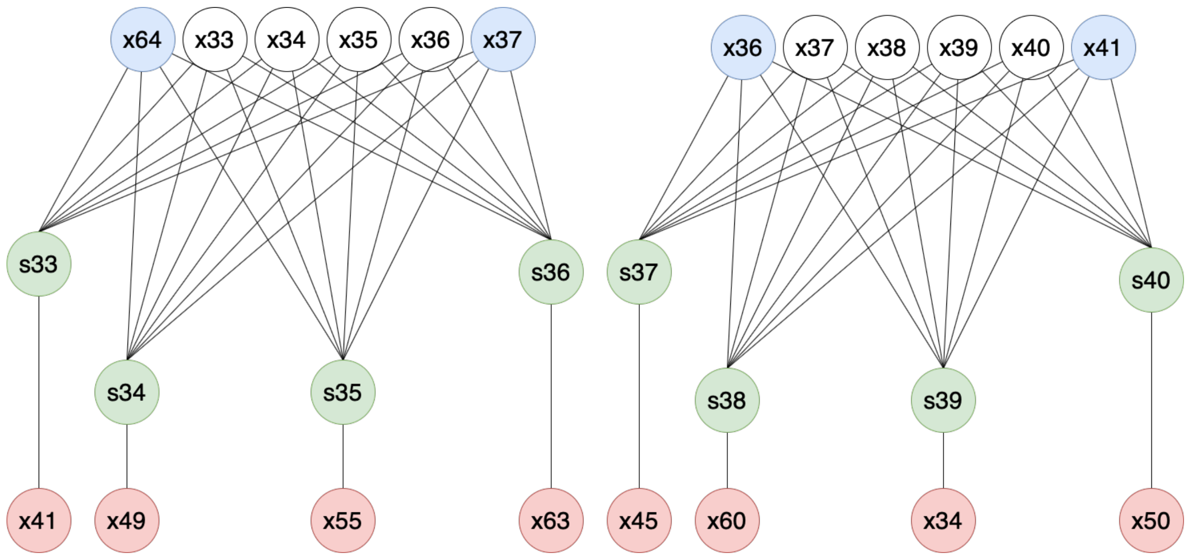


Figure 2.8: Graph edges for the P permutation, part 1

2.4.5 Facilitating path checks

Examples with all the paths for x_1^1 and x_{33}^1 are shown by Figures 2.12 and 2.13, respectively. Note that x_1^1 depends only on x_{33}^0, because it has only been copied on the $L_1 = R_0$ step — F has not been applied to it yet — while x_{33}^1 depends on 7 plaintext bits: x_1 due to the logical XOR with L_0, and the other six due to the application of F.

The intermediate nodes s_i^j, which represent S-Box outputs (green nodes in the figures) are not truly necessary for the implementation, since we are interested only in monitoring plaintext diffusion, although they help us explain the process. We can thus, when implementing this approach, create edges in the graph in the following way: *if there exists a path from x_i^1 to x_k^0, create an edge directly between them,*

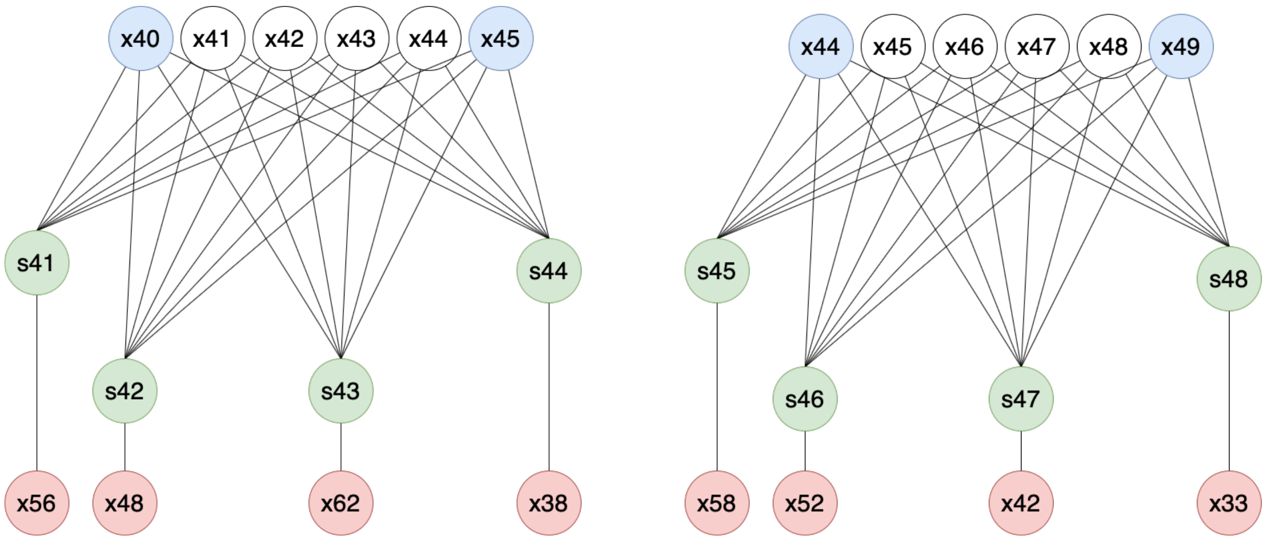


Figure 2.9: Graph edges for the P permutation, part 2

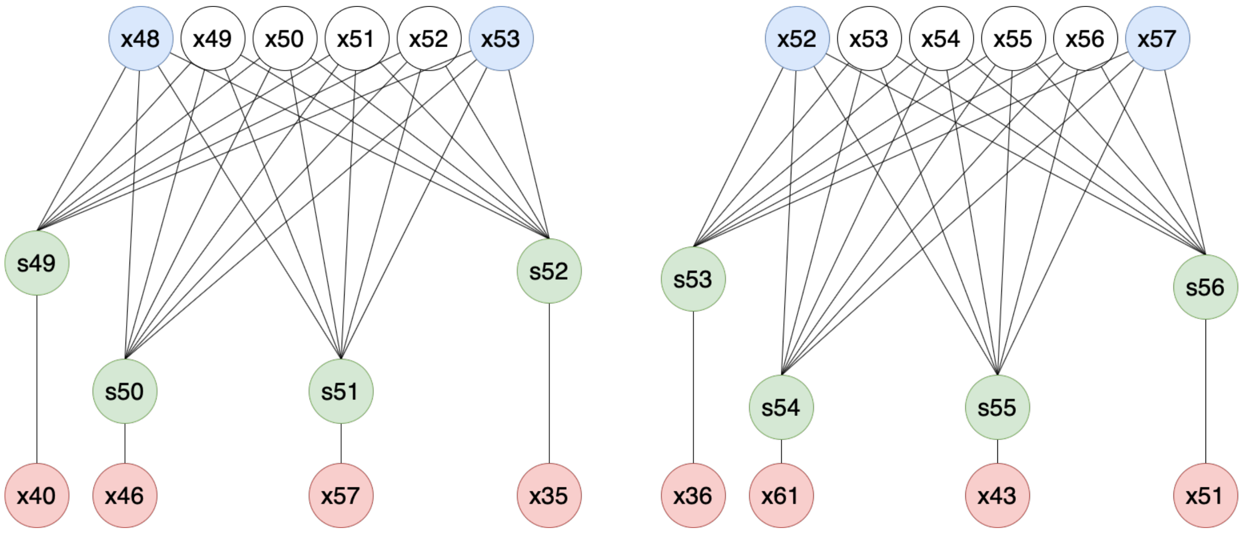


Figure 2.10: Graph edges for the P permutation, part 3

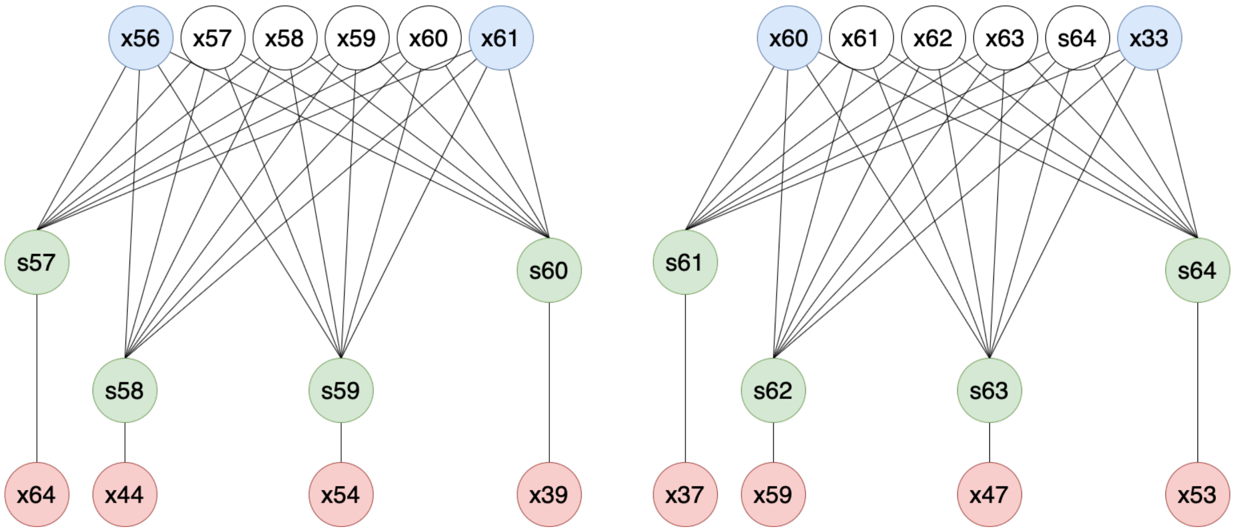


Figure 2.11: Graph edges for the P permutation, part 4

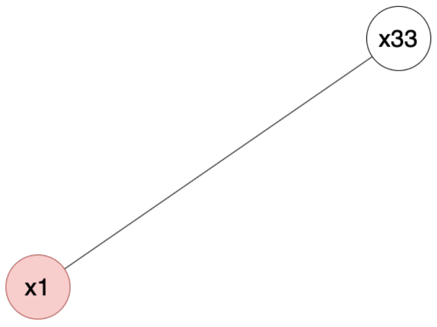


Figure 2.12: Graph paths for x_1^1

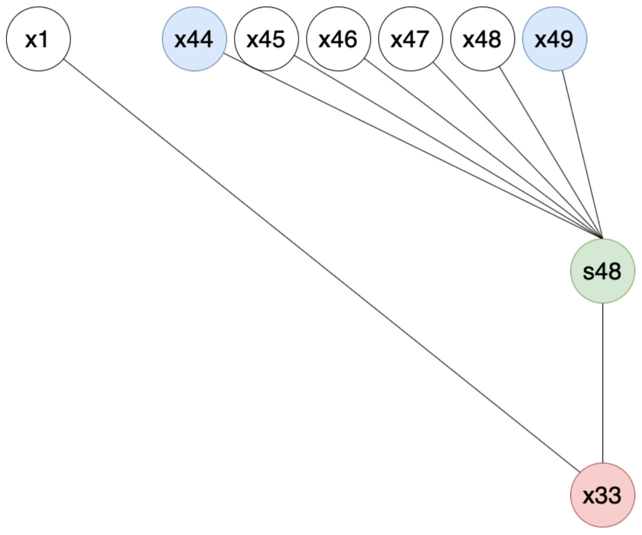


Figure 2.13: Graph paths for x_{33}^1

because, with an adjacency matrix representation, we can check if there is an edge between two nodes in $O(1)$ *time* — and thus we can immediately assess whether x_i^1 depends on x_k^0.

2.4.6 Expanding the analysis to all rounds with adjacency matrices

Graphs can be represented through adjacency lists or through an adjacency matrix. With an adjacency matrix representation, it is possible to make use of matrix exponentiation to expand the analysis towards all 16 rounds of DES, having computed the adjacency matrix which represents the diffusion for round 1.

We start by representing the dependencies of the first round with an adjacency matrix. A 64×64 adjacency matrix A can be built such that $A[r][c] = 1$ if the c-th bit after round 1 depends on the r-th bit of plaintext, and 0 otherwise. A^1 represents diffusion at round 1, A^2 represents diffusion at round 2, and so forth. When all elements of A^p are nonzero, it means that complete plaintext diffusion is achieved at round p of DES.

2.4.7 Plaintext diffusion results

With this method, we observed that A^5 does not contain zeroes, thus DES achieves complete diffusion by the end of round 5, since each state bit depends on all the plaintext bits.

It is relevant to note that elements greater than 1 will appear in A^p when $p > 1$, because of the sums that occur in the product. Therefore, in our implementation, elements greater than zero (not necessarily equal to 1) indicate dependence between bits, while elements equal to zero indicate no dependence between the corresponding bits.

If one wishes matrices with only zeroes and ones, the addition of elements can be replaced by a logical OR operation — if $A^p[i][j] = 1$, $A^q[i][j]$ should be 1 for all $q > p$, because a dependence between plaintext and state bits should not disappear once it has appeared in a certain round.

Unfortunately, the obtained 64x64 matrices are too large to be completely displayed in this document with good image quality. We make them available as `.csv` files at our Git repository [66], together with our code. Figures 2.14 to 2.18 show a graphic representation of them. Black dots indicate state bits which depend on plaintext bits, and white regions are regions in which diffusion has not yet occurred.

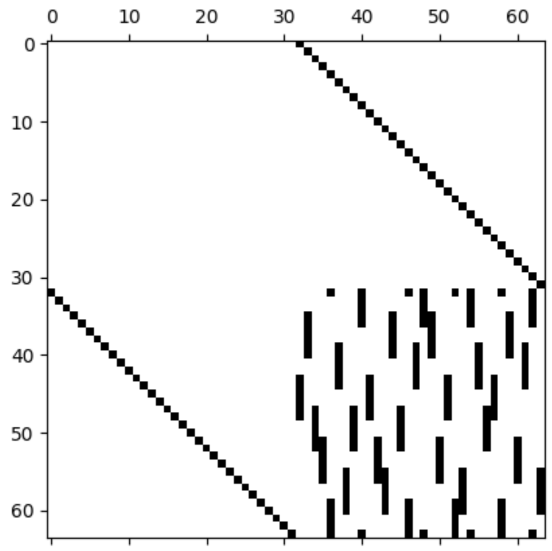


Figure 2.14: DES diffusion at round 1

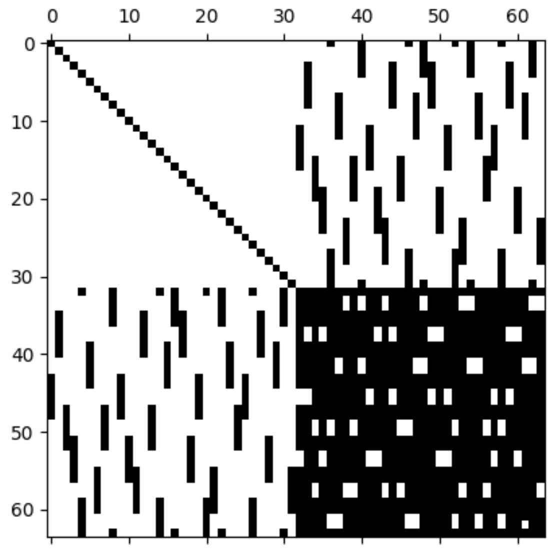


Figure 2.15: DES diffusion at round 2

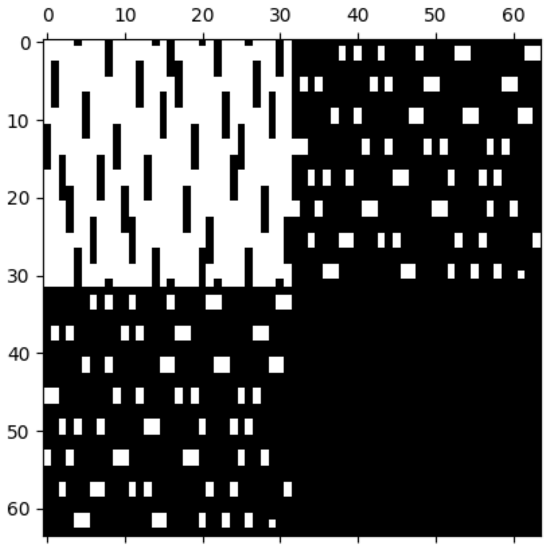


Figure 2.16: DES diffusion at round 3

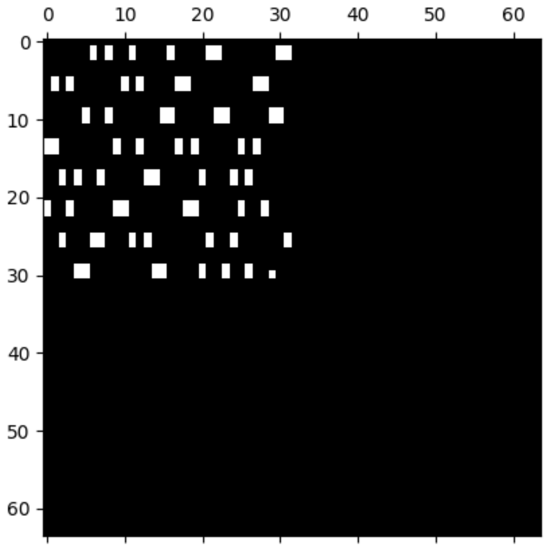


Figure 2.17: DES diffusion at round 4

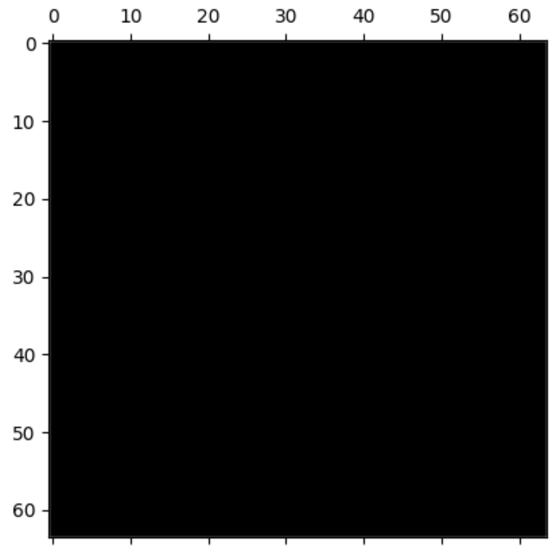


Figure 2.18: DES diffusion at round 5 (finally complete!)

2.5 Key diffusion

In order to analyse key diffusion within DES, the process is analogous to the one employed for the plaintext. We model the problem as a graph, create nodes to represent the variables, and add edges to track dependence between them. However, at each round a different subkey is used. Furthermore, the subkeys are smaller than the state (they have 48 bits whilst the state contains 64 bits), and they are input to DES differently. Because of that, we cannot use matrix exponentiation to expand the analysis to all rounds once we have a matrix to represent the dependencies related to round 1. Therefore, for the key diffusion analysis, we have implemented the graph as an adjacency list.

Also, since each subkey is derived from the master key through the key scheduling process, a thorough understanding of the key scheduling routine is required. Algorithm 2 describes the key scheduling process, also known as *key expansion* or *key derivation.*

Algorithm 2 DES key schedule

```
Require: A 64-bit key $K = k_1k_2...k_{63}k_{64}$
Ensure: Sixteen 48-bit subkeys $K_1, K_2, ..., K_{16}$
1: Define $v_i, 1 \leq i \leq 16$, as follows: $v_i = 1$ for $i \in \{1, 2, 9, 16\}$, $v_i = 2$ otherwise
2: $T \leftarrow PC1(K)$                                          ▷ PC1 is a permutation
3: Represent $T$ as two 28-bit halves $C_0$ and $D_0$, such that $C_0 = k_{57}k_{49}...k_{36}$ and $D_0 = k_{63}k_{55}...k_4$
4: for $i = 1, ..., 16$ do
5:     $C_i \leftarrow C_{i-1} << v_i$                    ▷ Here $<<$ denotes a left circular shift
6:     $D_i \leftarrow D_{i-1} << v_i$
7:     Represent the concatenation of $C_i$ and $D_i$ as $B = b_1b_2...b_{56}$
8:     $K_i \leftarrow PC2(B)$                                    ▷ PC2 is a permutation
9: end for
```

The permutations $PC1$ and $PC2$ mentioned on Algorithm 2 are shown by Tables 2.3 and 2.4, respectively.

PC1						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
above for C_i, below for D_i						
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Table 2.3: The PC1 permutation

PC2						
14	17	11	24	1	5	9
3	28	15	6	21	10	18
23	19	12	4	26	8	27
16	7	27	20	13	2	36
41	52	31	37	47	55	15
30	40	51	45	33	48	22
44	49	39	56	34	53	29
46	42	50	36	29	32	4

Table 2.4: The PC2 permutation

The input to the algorithm is the original key, i.e master key, which comprises 64 bits. The algorithm outputs sixteen 48-bit subkeys, which are to be used in each of the DES sixteen rounds, respectively. K_1 is used at round 1, K_2 at round 2, and so forth. Subkeys must not be repeated in different rounds — once they have been used in their corresponding round, they are no longer used.

The first step is to permute the master key, using $PC1$. Then, the result of the permutation is split into 28-bit halves, here denoted as C_0 and D_0. After that, the C_i and D_i for each round i, $1 \leq i \leq 16$, are generated by means of circular left shifts. The left shift depends on the values of the v_i (1 for rounds $1, 2, 9$ and 16, 2 otherwise). Once C_i and D_i have been generated, their concatenation, denoted by B, is permuted, this time with permutation $PC2$, and the result of this permutation is the subkey K_i of round i.

2.5.1 Modeling key diffusion as a graph

In order to track the diffusion of the key, intermediate nodes are helpful once again. Let x_l^r be a node that represents the l-th bit of the state at round r, s_j^r be a node that represents the j-th bit of an S-box output for round r, and $K_{r,z}$ be a node that represents the z-th bit of a subkey used in round r.

First, consider the j-th bit of the output of an S-box of the first round, hence a node s_j^1. In our previous analysis, which considered only plaintext, we create and edge from s_j^1 to x_l^0. However, in truth, the input to an S-box is not composed by state bits only — it is *the logical XOR of six state bits and six round key bits*. Therefore, if a round key bit $K_{1,z}$ is XORed to x_l^0 in DES, there should also be an edge between s_j^1 and $K_{1,z}$ to track this dependence. Thus a state bit of round 1, e.g x_w^1 which happens to depend on the S-box output s_j^1, will have a path towards $K_{1,z}$ as well.

For the other rounds, such as round 2, the same logic is applied. The intermediate nodes which represent S-box outputs should have edges towards the nodes of the form $K_{2,v}$, which represent bits of the second subkey, i.e the key used for round 2.

In order to track the dependence directly to the master key bits, the nodes of the form $K_{r,z}$ have edges towards nodes of the form k_p, where k_p represents the p-th bit of the master key. A state bit thus depends on a master key bit if there is a path between them. Figure 2.19 illustrates the graph edges for the state bit x_{33}^1.

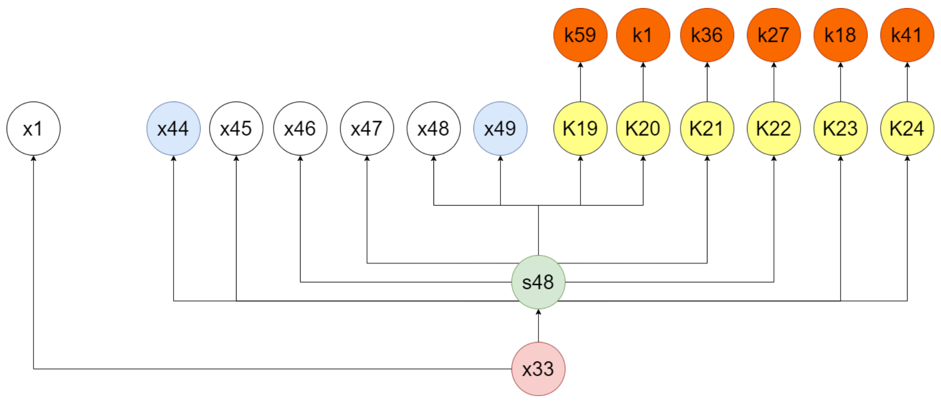


Figure 2.19: An example of key diffusion dependence tracking. White nodes and blue nodes refer to plaintext bits, the green node refers to the S-box output, yellow nodes refer to the subkey, orange nodes refer to the master key, and the red node refers to a round 1 state bit.

To obtain the edges between subkey nodes and master key nodes, we follow the key derivation process and make use of intermediate nodes. In our implementation, we created nodes for each step of the key derivation: nodes for bits of T, bits of C_i and D_i, of B and, finally, of the master key k. And, to easily find the dependence, when there is a path from a node X to a node Y, we add an edge directly from X to Y, as described in Section 2.4.5. In this way, a round key bit of the type K_i is directly mapped to a master key bit of the type k and assessing whether the dependency holds no longer requires checking the existence of a path in the graph, only the existence of an edge.

2.5.2 Key diffusion results

With this method, we could determine that DES also achieves complete key diffusion *by the end of the fifth round*. Another fact is that, if we start, e.g., from round 2, complete key diffusion will be achieved by the end of round 6. If we start from round 3, it will be achieved by the end of round 7, and so forth. In other words, it does not matter which round we start from, the key will have been completely diffused after 5 rounds.

It is relevant to note that, from the 64 bits of the master key, 8 are deemed parity bits, and they are not used, i.e diffused. Thus the complete diffusion is achieved when each state bit depends on 56 master key bits. Table 2.5 shows the key diffusion for the first 5 rounds of DES, with minimum key diffusion referring to a state bit which depends on the smallest amount of master key bits and maximum key diffusion referring to a state bit which depends on the largest amount of master key bits.

Round	**Minimum key diffusion**	**Maximum key diffusion**
1	6	6
2	6	39
3	36	56
4	52	56
5	56	56

Table 2.5: Key diffusion along the first five DES rounds

At round 5, the minimum key diffusion is 56, thus all the key bits have spread their influence to all the state bits, and complete diffusion has been achieved.

2.6 Implementation remarks and computational complexity

For the plaintext diffusion analysis, the computational complexity entails the construction of a 64×64 matrix A representing the dependencies and, then, matrix exponentiation in order to obtain A^5. Due to the small size of the matrix and the small exponent, the computational cost of this method is negligible.

For the key diffusion analysis, the adjacency-list construction can also be done with negligible computational cost, since only a constant amount of edges must be created due to the structure of the DES cipher. Section 2.6.1 generalizes the analysis and provides upper bounds for the computational complexity of our approach.

2.6.1 Asymptotic complexity and an upper bound

Upper bound for plaintext diffusion analysis

In general, constructing an adjacency matrix to track diffusion from the plaintext towards the first round of a block cipher of block size n_b, not necessarily a Feistel Network, and not necessarily similar to DES, would require $O(n_b)$ time. For each round 1 output bit, we need to create the correct edges (assuming edges can be created in constant time). In the worst case, if all round 1 bits have edges towards all plaintext bits, n_b^2 edges would have to be created. Therefore, the worst case run time is $O(n_b^2)$ and the average case run time is $O(n_b)$ to construct the matrix.

A naive matrix multiplication algorithm requires $O(n^3)$ time to multiply two $n \times n$ matrices. If we were to apply this method to a cipher of block size n_b and n_r rounds, assuming we wish to obtain all matrices for all rounds, the complexity would be $O(n_r \cdot n_b^3)$, with a naive exponentiation algorithm that required n_r multiplications to obtain A^{n_r}. Matrix multiplication/exponentiation can be optimized e.g by means of Strassen's algorithm (see [16]) and many other approaches, and optimizing the implementation would be relevant when analysing ciphers with large n_b and/or n_r that made a naive implementation not practical, but it is not the case for DES (and it would probably not be for other ciphers, since n_r and n_b are usually fixed and small).

Since constructing the matrix requires $O(n_b^2)$ time and the exponentiation requires $O(n_r \cdot n_b^3)$ time, the asymptotic complexity is $O(n_r \cdot n_b^3)$. However, since n_r and n_b are small and fixed, the practical run time is actually constant.

Upper bound for key diffusion analysis

For an n_r-round block cipher with n_{sk}-bit subkeys and block size n_b, the adequate edges must be created between each of the n_{sk} bits and the n_b round output bits. In the worst case, $n_b \cdot n_{sk}$ edges must be created at each round, if each round output bit depends on all the subkey bits. Therefore, the run time for this step is $O(n_r \cdot n_b \cdot n_{sk})$.

However, there must be edges mapping from the subkey bits to the n_k master key bits, and directly from the round output bits as well, to facilitate the dependency check. Therefore, $2 \cdot n_k \cdot n_{sk}$ more edges would be required, for each round, in the worst case (worst case = each bit of the subkey depends on all bits of the master key).

The asymptotic run time, considering edges can be created in constant time is thus $O(n_r \cdot n_k \cdot n_{sk} + n_r \cdot n_b \cdot n_{sk})$. However, as is the case of n_b and n_r, n_k and n_{sk} are also fixed and small for most ciphers, which leads to constant time in practice.

2.7 Conclusions and future work

In this chapter, we analyzed the DES cipher with respect to plaintext diffusion and key diffusion. For this purpose, we model the DES cipher as a graph, with vertices for each state, plaintext and key bit, among other intermediate vertices to facilitate the comprehension of the process.

We show that DES achieves both complete plaintext and key diffusion by the end of the 5-th round. For the plaintext diffusion analysis, we use the adjacency matrix representation of the graph and take advantage of matrix exponentiation. For the key diffusion analysis, we used the adjacency-list representation of the graph.

The approaches presented in this chapter have negligible computational cost, as argued in Section 2.6, and thus they might provide an interesting method for diffusion analysis of other ciphers and cryptographic primitives. Furthermore, this chapter is limited to analysing the DES cipher. Other ciphers could be analyzed with this method, as well as hash functions and other cryptographic algorithms.

In Section 2.4.7, we mention that values greater than 1 appear in the matrices because of the sums that occur in the exponentiation process. It could be interesting to assess whether they provide further information about the DES cipher diffusion.

Finally, the graph shows in detail all the dependencies between plaintext/key and round output bits. Attempts at different visual representations of the DES graph other than the one provided in Section 2.4.7 would be interesting.

It is also worth investigating the application of other graph related concepts and algorithms, and whether they would allow us to obtain significant information regarding cryptographic algorithms modeled as graphs. In this chapter, we restrict to the construction of adjacency matrices and lists, but graphs have several other concepts that could be explored (e.g shortest paths, edge weights, among others).

Chapter 3

Analysis of diffusion in AES

From Chapter 1 and the AES specification [59], we know that the AES S-Box achieves complete diffusion, i.e each output bit depends on all input bits, and that an AES round is composed of the `SubBytes`, `ShiftRows`, `MixColumns` and `AddRoundKey` steps. Furthermore, before the first round, an `AddRoundKey` is performed and, in the last round, `MixColumns` is not present. We also know that there are 3 supported key sizes n_k for AES (128 bits, 192 bits and 256 bits), and that the number of rounds n_r varies according to them. For $n_k = 128$, $n_r = 10$. For $n_k = 192$, $n_r = 12$ and, finally, for $n_k = 256$, $n_r = 14$.

In this chapter, we analyze plaintext and key diffusion in the AES cipher [59]. We show that complete plaintext diffusion is achieved after 2 rounds for all parameter sets, whilst complete key diffusion is achieved after 2 rounds for $n_k = 128$. For $n_k = 192$ and $n_k = 256$, it is achieved after 3 rounds.

We assume the reader is already familiar with:

- the AES cipher [59] and its round transformations (available on Appendix A)
- the AES key schedule and how the master key is distributed between k_0 and k_1 for different parameter sets
- the fact that the AES S-Box achieves complete diffusion of its 8 input bits (see Chapter 1 for an S-Box diffusion analysis)
- GF(2^8) arithmetic, more specifically the fact that adding two elements of GF(2^8) is equivalent to performing their bitwise XOR
- the XOR Boolean function

3.1 Notation

- x: AES plaintext
- x_i: i-th byte of AES plaintext
- x'_i: i-th byte of AES round 1 output
- x''_i: i-th byte of AES round 2 output
- MC: the matrix used in the `MixColumns` transformation
- $x_i \rightarrow x_z$: x_i depends on x_z
- $x_i \rightarrow x_p, x_q, ..., x_z$: x_i depends on variables $x_p, x_q, ..., x_z$
- k_r: AES subkey used at round r
- $x^{r,SB}$: AES state after the `SubBytes` transformation of round r has been performed, $r \geq 1$
- $x^{r,SR}$: AES state after the `ShiftRows` transformation of round r has been performed, $r \geq 1$
- $x^{r,MC}$: AES state after the `MixColumns` transformation of round r has been performed, $r \geq 1$

- $x^{r,AK}$: AES state after the `AddRoundKey` transformation of round r has been performed, $r \geq 0$ (because, before the iterations start, there is one `AddRoundKey` transformation)
- $+$: addition in GF(2^8)
- $S(x_i)$: AES `SubBytes` step applied to a byte x_i

3.2 Acronyms

- AES: Advanced Encryption Standard
- S-Box: Substitution Box

3.3 AES

For convenience, AES encryption is presented in Algorithm 3.

Algorithm 3 AES Encryption

```
Require: $x$ (plaintext), $k_0, k_1, \ldots, k_{n_r}$ (subkeys)
Ensure: $y$ (ciphertext)
  $x \leftarrow$ AddRoundKey($x, k_0$)
  for $r$ from 1 to $n_r - 1$ do
    $x \leftarrow$ SubBytes($x$)
    $x \leftarrow$ ShiftRows($x$)
    $x \leftarrow$ MixColumns($x$)
    $x \leftarrow$ AddRoundKey($x, k_r$)
  end for
  $x \leftarrow$ SubBytes($x$)
  $x \leftarrow$ ShiftRows($x$)
  $x \leftarrow$ AddRoundKey($x, k_{n_r}$)
  $y \leftarrow x$
  return $y$
```

3.4 Plaintext diffusion

The AES state is commonly represented as a 4×4 matrix of bytes

$$x = \begin{bmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{bmatrix}.$$

For the moment, we restrict our analysis to plaintext diffusion. We assume $k_r = 0$ for all rounds of AES, which implies $x^{0,AK} = x$, since $x_i + 0 = x_i$. In other words, the `AddRoundKey` step is ignored because we are interested in studying plaintext diffusion.

3.4.1 The first `SubBytes`

After the `SubBytes` transformation of the first round, the state is

$$x^{1,SB} = \begin{bmatrix} S(x_0) & S(x_4) & S(x_8) & S(x_{12}) \\ S(x_1) & S(x_5) & S(x_9) & S(x_{13}) \\ S(x_2) & S(x_6) & S(x_{10}) & S(x_{14}) \\ S(x_3) & S(x_7) & S(x_{11}) & S(x_{15}) \end{bmatrix}.$$

From Chapter 1, we know that the application of the S-Box ensures complete diffusion on the bit level (each bit of $S(x_i)$ depends on all bits of x_i). However, it does not ensure dependence between different bytes, i.e $S(x_1)$ does not depend on x_0. Therefore, $S(x_i) \to x_i$.

3.4.2 The first `ShiftRows`

The `ShiftRows` step performs a cyclic permutation in each row of the matrix, thus resulting in

$$x^{1,SR} = \begin{bmatrix} S(x_0) & S(x_4) & S(x_8) & S(x_{12}) \\ S(x_5) & S(x_9) & S(x_{13}) & S(x_1) \\ S(x_{10}) & S(x_{14}) & S(x_2) & S(x_6) \\ S(x_{15}) & S(x_3) & S(x_7) & S(x_{11}) \end{bmatrix}.$$

3.4.3 The first `MixColumns`

The `MixColumns` step is the multiplication of each column of the state by

$$MC = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}.$$

The first column of the new state is given by

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} S(x_0) \\ S(x_5) \\ S(x_{10}) \\ S(x_{15}) \end{bmatrix} = \begin{bmatrix} 2S(x_0) + 3S(x_5) + S(x_{10}) + S(x_{15}) \\ S(x_0) + 2S(x_5) + 3S(x_{10}) + S(x_{15}) \\ S(x_0) + S(x_5) + 2S(x_{10}) + 3S(x_{15}) \\ 3S(x_0) + S(x_5) + S(x_{10}) + 2S(x_{15}) \end{bmatrix}.$$

Note that, after MC multiplies a column, each element of the resulting column now depends linearly on the others. This applies to the second, third and fourth new columns as well. Therefore,

$$x^{1,MC} = \begin{bmatrix} x'_0 & x'_4 & x'_8 & x'_{12} \\ x'_1 & x'_5 & x'_9 & x'_{13} \\ x'_2 & x'_6 & x'_{10} & x'_{14} \\ x'_3 & x'_7 & x'_{11} & x'_{15} \end{bmatrix},$$

and the following dependencies between bytes now hold:

- each $x'_i \to x_0, x_5, x_{10}, x_{15}$, for $0 \le i \le 3$,
- each $x'_j \to x_4, x_9, x_{14}, x_3$, for $4 \le j \le 7$,
- each $x'_l \to x_8, x_{13}, x_2, x_7$, for $8 \le l \le 11$,
- each $x'_m \to x_{12}, x_1, x_6, x_{11}$, for $12 \le m \le 15$.

The next transformation would be `AddRoundKey`, but $x^{1,AK} = x^{1,MC}$ because all the subkeys are assumed to be zero. We thus move to the analysis of the second round.

3.4.4 The second `SubBytes`

Similarly to the first `SubBytes`, the application of S-Boxes ensures that all bits of $S(x'_i)$ depend on all bits of x'_i, but does not incur dependence between x'_i and x'_j for $i \neq j$.

3.4.5 The second `ShiftRows`

The rows of the state matrix are cyclically shifted, resulting in

$$x^{2,SR} = \begin{bmatrix} S(x'_0) & S(x'_4) & S(x'_8) & S(x'_{12}) \\ S(x'_5) & S(x'_9) & S(x'_{13}) & S(x'_1) \\ S(x'_{10}) & S(x'_{14}) & S(x'_2) & S(x'_6) \\ S(x'_{15}) & S(x'_3) & S(x'_7) & S(x'_{11}) \end{bmatrix}.$$

3.4.6 The second `MixColumns`

Analogously to the first `MixColumns`, the multiplication by MC causes bytes of the same column to spread their dependence. Therefore,

$$x^{2,MC} = \begin{bmatrix} x''_0 & x''_4 & x''_8 & x''_{12} \\ x''_1 & x''_5 & x''_9 & x''_{13} \\ x''_2 & x''_6 & x''_{10} & x''_{14} \\ x''_3 & x''_7 & x''_{11} & x''_{15} \end{bmatrix},$$

and

- each $x''_i \to x'_0, x'_5, x'_{10}, x'_{15}$, for $0 \le i \le 3$,
- each $x''_j \to x'_4, x'_9, x'_{14}, x'_3$, for $4 \le j \le 7$,
- each $x''_l \to x'_8, x'_{13}, x'_2, x'_7$, for $8 \le l \le 11$,
- each $x''_m \to x'_{12}, x'_1, x'_6, x'_{11}$, for $12 \le m \le 15$.

Note that, from Section 3.4.3,

$$\begin{aligned} x'_0 &\to x_0, x_5, x_{10}, x_{15}, \\ x'_5 &\to x_4, x_9, x_{14}, x_3, \\ x'_{10} &\to x_8, x_{13}, x_2, x_7, \\ x'_{15} &\to x'_{12}, x'_1, x'_6, x'_{11}. \end{aligned}$$

Therefore, $x''_0 \to x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}$, i.e x''_0 depends on all plaintext bytes. The same holds for all the other bytes x''_i of the second round output of AES. Thus plaintext diffusion on the byte level is complete after 2 rounds.

3.4.7 Plaintext diffusion on the bit level

`ShiftRows` and `MixColumns` ensure complete diffusion by the end of round 2, on the byte level. We know that each bit of $S(x_i)$ depends on all bits of x_i, due to the complete diffusion achieved by the AES S-Box (see Chapter 1).

After the first round is completed, we have

$$x'_0 = 2S(x_0) + 3S(x_5) + S(x_{10}) + S(x_{15}),$$

due to `ShiftRows` and `MixColumns`. The b-th bit of x'_0 depends on the b-th bits of $S(x_0), S(x_5), S(x_{10})$ and $S(x_{15})$, because addition of two elements in GF(2^8) is equivalent to a bitwise XOR. However, the b-th bit of $S(x_5)$ depends on each bit of x_5, due to the S-Box complete diffusion property. Therefore, each b-th bit of x'_0 depends on all bits of x_0, all bits of x_5, all bits of x_{10} and all bits of x_{15}. The same logic applies to other round 2 outputs, hence each bit of x'_i depends on 32 bits of plaintext.

After the second round is completed, we know that

$$x''_0 = 2S(x'_0) + 3S(x'_5) + S(x'_{10}) + S(x'_{15}).$$

Analogously, the b-th bit of x''_0 depends on the b-th bits of $S(x'_0), S(x'_5), S(x'_{10})$ and $S(x'_{15})$. The b-th bit of $S(x'_0)$ depends on 32 bits of plaintext. The b-th bit of $S(x'_{10})$ depends on other 32 bits of plaintext, and so forth. Therefore, each b-th bit of x''_0 depends on all 128 bits of the AES plaintext, and complete diffusion, on the bit level, is also achieved by the end of the second round of AES.

3.5 Key diffusion

3.5.1 128 key bits ($n_k = 128$)

For $n_k = 128$, k_0 is the master key and it is used in the `AddRoundKey` transformation that precedes the first round of AES.

Let $k_{r,i}$ denote the i-th bit of the AES subkey k_r. We no longer assume $k_r = 0$, therefore,

$$x^{0,AK} = \begin{bmatrix} x_0 + k_{0,0} & x_4 + k_{0,4} & x_8 + k_{0,8} & x_{12} + k_{0,12} \\ x_1 + k_{0,1} & x_5 + k_{0,5} & x_9 + k_{0,9} & x_{13} + k_{0,13} \\ x_2 + k_{0,2} & x_6 + k_{0,6} & x_{10} + k_{0,10} & x_{14} + k_{0,14} \\ x_3 + k_{0,3} & x_7 + k_{0,7} & x_{11} + k_{0,11} & x_{15} + k_{0,15} \end{bmatrix}. \tag{3.1}$$

It suffices to consider that the input to the first `SubBytes` is no longer plaintext only and argue that, if `SubBytes` ensures that $S(x_i)$ depends on all bits of x_i, it ensures that $S(x_i + k_{0,i})$ depends on all bits of $k_{0,i}$.

Furthermore, the new first column of $x^{1,MC}$ is actually

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} S(x_0 + k_{0,0}) \\ S(x_5 + k_{0,5}) \\ S(x_{10} + k_{0,10}) \\ S(x_{15} + k_{0,15}) \end{bmatrix} = \begin{bmatrix} 2S(x_0 + k_{0,0}) + 3S(x_5 + k_{0,5}) + S(x_{10} + k_{0,10}) + S(x_{15} + k_{0,15}) \\ S(x_0 + k_{0,0}) + 2S(x_5 + k_{0,5}) + 3S(x_{10} + k_{0,10}) + S(x_{15} + k_{0,15}) \\ S(x_0 + k_{0,0}) + S(x_5 + k_{0,5}) + 2S(x_{10} + k_{0,10}) + 3S(x_{15} + k_{0,15}) \\ 3S(x_0 + k_{0,0}) + S(x_5 + k_{0,5}) + S(x_{10} + k_{0,10}) + 2S(x_{15} + k_{0,15}) \end{bmatrix}. \tag{3.2}$$

Therefore, the following dependencies between round 1 bytes and key bytes hold:

- each $x'_i \to k_{0,0}, k_{0,5}, k_{0,10}, k_{0,15}$, for $0 \le i \le 3$,
- each $x'_j \to k_{0,4}, k_{0,9}, k_{0,14}, k_{0,3}$, for $4 \le j \le 7$,
- each $x'_l \to k_{0,8}, k_{0,13}, k_{0,2}, k_{0,7}$, for $8 \le l \le 11$,
- each $x'_m \to k_{0,12}, k_{0,1}, k_{0,6}, k_{0,11}$, for $12 \le m \le 15$.

As done in Section 3.4.6, it is possible to show that all round 2 output bytes depend on all k_0 bytes, and thus key diffusion is, too, achieved by the end of the second round, on the byte level. In order to show that it is also achieved on the bit level, the same logic of Section 3.4.7 applies, given that the actual inputs to the S-Box are not only plaintext bytes but the bitwise XOR of them to the key bytes. Therefore, key diffusion is achieved, on the bit level, by the end of round 2 as well.

3.5.2 192 key bits ($n_k = 192$)

For $n_k = 192$, k_0 is the master key and it is used in the `AddRoundKey` transformation that precedes the first round of AES. However, the other 64 master key bits are not present in k_0: they are present in k_1 only.

Each k_r requires two successive applications of the `SubBytes`, `ShiftRows` `MixColumns` sequence of transformations to be completely diffused. Since k_0 has been mixed to the plaintext prior to the start of the AES iterations, two iterations are sufficient. However, k_1 is only mixed to the state by the end of the first iteration. Therefore, it will have been completely diffused by iteration 3.

3.5.3 256 key bits ($n_k = 256$)

For $n_k = 256$, k_0 contains the first 128 bits of the master key and k_1 contains the other 128 bits. The same logic applied for $n_k = 192$ applies here: since 128 key bits require 2 iterations for complete diffusion after they have been mixed to the state, and the bits of k_1 are only mixed by the end of round 1, complete key diffusion is achieved by the end of round 3 for $n_k = 256$.

3.6 Visualizing the transformations

Figure 3.1 illustrates diffusion in AES. Although the state is most commonly represented as a 4×4 matrix, it is also possible to represent it as an ordered tuple of bytes $X = (X_0, ..., X_{15})$.

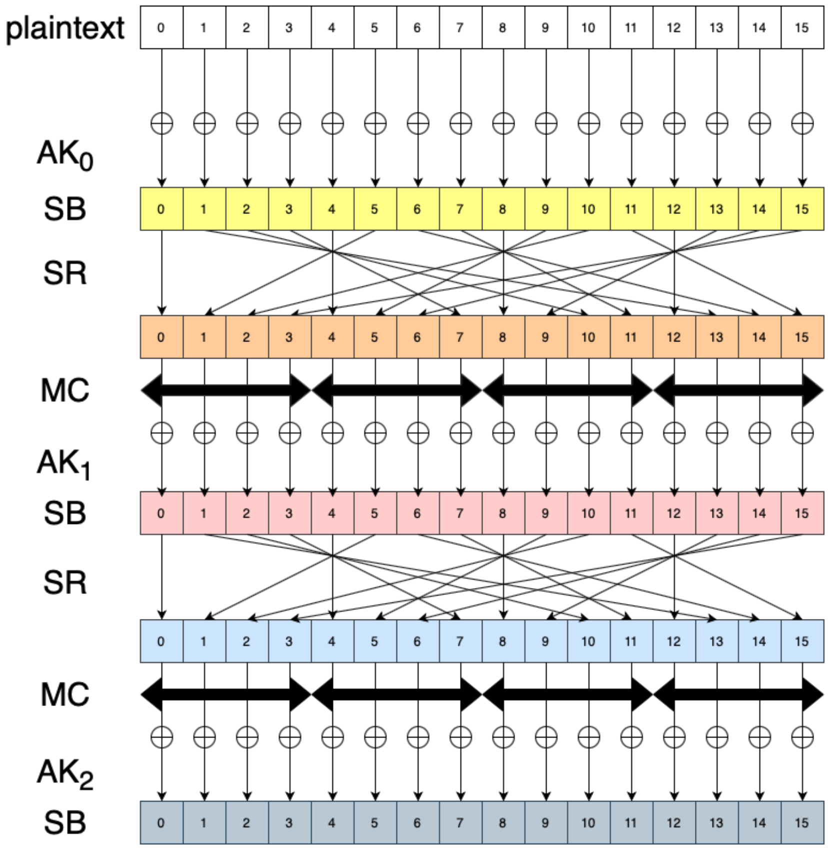


Figure 3.1: Illustrating the AES round transformations. The X symbol is omitted for simplicity. White refers to plaintext, yellow to $x^{1,SB}$, orange to $x^{1,SR}$, pink to $x^{2,SB}$, blue to $x^{2,SR}$ and grey to $x^{3,SB}$.

3.7 Conclusions

In this chapter, we analyzed the AES cipher with respect to the complete diffusion property, and showed that complete plaintext diffusion is achieved by the end of the second iteration, for all parameter sets. We also showed that complete key diffusion is achieved by the end of the second iteration for a key size of 128 bits and, for the key sizes 192 and 256, it is achieved by the end of the third iteration. More about the design of AES can be found in [24].

From the conducted analysis, the role of each AES component towards achieving complete plaintext and key diffusion is clear. `SubBytes` ensures diffusion on the bit level, since all the other steps have bytes as their basic unit. `MixColumns` and `ShiftRows`, together, ensure that complete diffusion is achieved by the second round of the cipher. Without `MixColumns`, the elements within each AES state column would not be diffused. Without `ShiftRows`, elements across different columns would not spread their influence to the state. Therefore, the joint usage of the matrix and the byte permutation is extremely relevant. And, finally, without `AddRoundKey`, the key would not be diffused.

Chapter 4

Obtaining Difference Distribution Tables of AES and DES

Differential Cryptanalysis was discovered by Eli Biham and Adi Shamir [10]. In [10] and [11], they describe how to apply this method to the DES cipher. They show it is possible to break a reduced variant of DES (an 8-round DES), and how to break other reduced variants (with up to 15 rounds) in less than 2^{56} steps. They are also able to break the full 16-round DES, although the cost is greater than the 2^{64} of an exhaustive key search.

The method analyses differences in plaintext and ciphertext pairs. Such differences can be used to assign probabilities to possible keys and, further on, to identify the most probable key.

In this chapter, we focus on the *Difference Distribution Table (DDT)*, which is a core component for a Differential Cryptanalysis (DC) attack. First, we show how to obtain them for the DES cipher [58], and discuss their relationship with the DES design strategies [14] against DC attacks. Then, we apply the same method to obtain the DDT for the AES S-Box and its inverse.

We assume the reader is familiar with:

- The AES cipher structure (available at Appendix A)
- The DES cipher structure (available at Appendix A)
- S-Boxes (see Chapter 1)
- Bitwise XOR

4.1 Notation

- M, M^*: plaintexts
- M': plaintext difference of M and M^*
- X, X^*: intermediate variables during encryption
- X': difference of X and X^*
- Y, Y^*: S-Box outputs
- $S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8$: DES S-Boxes
- IP: the initial permutation of DES
- IP^{-1}: the inverse of the initial permutation of DES
- P: the DES bit permutation P
- E: the DES expansion function E
- K: key
- $\mathcal{D}$: a Difference Distribution Table (DDT)

- k_i: key bits shared between S-Box inputs
- n_s: input size (in bits) of an S-Box
- n_j: the size (in bits) of the joint input of neighbor S-Boxes
- sh_k: number of shared key bits between neighbor S-Boxes

4.2 Acronyms

- DES: Data Encryption Standard
- AES: Advanced Encryption Standard
- DC: Differential Cryptanalysis
- S-Box: Substitution Box
- DDT: Difference Distribution Table

4.3 Fundamentals

4.3.1 Differences (XORs)

(Plaintext difference (XOR)) *Let M and M^* denote two plaintexts. The plaintext difference, also called* plaintext XOR*, is*

$$M' = M \oplus M^*.$$

(Intermediate variable difference (XOR)) *Let X and X^* denote any intermediate variable during the encryption of M and M^*, respectively. The difference (XOR) of these intermediate variables is*

$$X' = X \oplus X^*.$$

S-Box inputs and outputs are intermediate variables and, therefore, the concept of difference can be applied to them.

(Input and output difference of an S-Box) *Let X and X^* be inputs to an S-Box S, and Y and Y^* be the respective outputs. The input difference is*

$$X' = X \oplus X^*,$$

and the output difference is

$$Y' = Y \oplus Y^* = S(X) \oplus S(X^*).$$

In [10], the authors mount a DC attack with a bottom-up approach. They start by considering S-boxes input and output differences. Then they expand the concept to the whole F function. Then, to a complete round of the cipher and, finally, they push the analysis forward towards an arbitrary number of rounds N.

It is worth noting that the S-Boxes, in the case of DES, are the only non-linear component. Other components (e.g IP, P, IP^{-1} and E) are linear. Their output differences are linearly related, whilst S-Box output differences are not. Therefore, the DES S-Boxes require an analysis with *Difference Distribution Tables (DDTs)* (see Definition 6). They are highly relevant for the attack development.

Furthermore, the usage of the XOR to compute the difference is due the fact that the key is combined to the cipher state by means of an XOR. Consequently, the usage of differences (instead of any other individual intermediate value of the cipher state) allows the attacker to track the behavior of the DES components throughout the cipher, which would not be possible using individual values, since the key is unknown. For example, an S-Box input XOR does not depend on the key. However, the output XOR does. Therefore, choosing input differences and observing output differences, it is possible to exploit probabilistic relationships between the key and intermediate states of the cipher. If the key was combined by means of another operator, it would be necessary to define input and output differences in another way. In a more general way, differences can be defined as follows.

(Plaintext difference) *Let M and M^* denote two plaintexts, $\bullet$ be the operator which combines the key to the cipher state, and $M^{*^{-1}}$ be the inverse of M^* with respect to $\bullet$. The plaintext difference is*

$$M' = M \bullet M^{*^{-1}}.$$

Note that, when $\bullet$ is the XOR operator, $M^{*^{-1}} = M^*$.

(Intermediate variable difference) *Let X and X^* denote any intermediate variable during the encryption of M and M^*, respectively, $\bullet$ be the operator which combines the key to the cipher state, and $X^{*^{-1}}$ be the inverse of X^* with respect to $\bullet$. The difference of these intermediate variables is*

$$X' = X \bullet X^{*^{-1}}.$$

Note that, when $\bullet$ is the XOR operator, $X^{*^{-1}} = X^*$.

(Key information revealed by the output difference) *Let X and X^* be plaintext bits and K be key bits, and let $V = X \oplus K$ and $V^* = X^* \oplus K$ be inputs to an S-Box. The input difference is*

$$V' = V \oplus V^* = (X \oplus K) \oplus (X^* \oplus K).$$

The key dependence "disappears" because K is XORed twice:

$$(X \oplus K) \oplus (X^* \oplus K) = X \oplus X^*.$$

Therefore, $V' = X \oplus X^*$, and the input difference is not key-dependent. *Let Y and Y^* be the respective outputs. Then*

$$Y = S(V) = S(X \oplus K),$$

$$Y^* = S(V) = S(X^* \oplus K),$$

and

$$Y' = Y \oplus Y^* = S(X \oplus K) \oplus S(X^* \oplus K).$$

Due to the non-linearity of the S-Box, $S(X \oplus K) \neq S(X) \oplus K$, and $Y' \neq S(X) \oplus K \oplus S(X^*) \oplus K$. The key term does not disappear, and thus the output difference reveals probabilistic information about the unknown key.

4.3.2 Difference Distribution Table

Definition 6 (Difference Distribution Table of an S-Box) *The Difference Distribution Table (DDT) of an S-box is a table that shows the distribution of input XORs and output XORs for an S-box [10].*

In other words, for a given input XOR X', the table shows how many times an output XOR Y' occurs — and thus the probability of a given (X', Y') pair. Formally, the Difference Distribution Table (DDT) of an S-box S: $\mathbb{Z}_2^n \to \mathbb{Z}_2^m$ is a $2^n \times 2^m$ table $\mathcal{D}$ such that

$$\mathcal{D}[X'][Y'] = \#\{(X, X^*) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n : S(X) \oplus S(X^*) = S(X) \oplus S(X \oplus X') = Y'\},$$

where $X' \in \mathbb{Z}_2^n$ is the input difference, $Y' \in \mathbb{Z}_2^m$ is the output difference, and $\oplus$ denotes bitwise exclusive-or. Algorithm 4 shows how to compute the DDT of an S-Box.

Algorithm 4 Obtaining the DDT of an S-box

```
1: Initialize $\mathcal{D}$ with zeroes in all entries
2: for $X = 0, ..., 2^n - 1$ do
3:     for $X^* = 0, ..., 2^n - 1$ do
4:         $Y \leftarrow S(X)$
5:         $Y^* \leftarrow S(X^*)$
6:         $X' \leftarrow X \oplus X^*$
7:         $Y' \leftarrow Y \oplus Y^*$
8:         $\mathcal{D}[X'][Y'] \leftarrow \mathcal{D}[X'][Y'] + 1$
9:     end for
10: end for
11: return $\mathcal{D}$
```

4.3.3 Relevant properties of the DDT

The [0][0] entry

For $X' = 0$, all the output XORs are also zero. That happens because, when $X' = 0$, $X \oplus X^* = 0$, hence $X = X^*$. When $X = X^*$, then $Y = S(X) = Y^* = S(X^*)$, and thus $Y' = 0$, too.

Since all the output XORs associated to $X' = 0$ are zero, the first DDT entry, $\mathcal{D}[0][0]$, contains the count of all such zeroes. Furthermore, the other entries at row 0 are also always be zero, because when $X' = 0$ there are no non-zero Y' values.

The count of zeroes stored at $\mathcal{D}[0][0]$ is always 2^n, because there are 2^n cases for which $X' = 0$. X starts at 0 and ends at $2^n - 1$ in the for loop, and so does X^* in the innermost loop. For each X, there will be one X^* which is exactly the inverse of X (i.e X with all bits flipped), resulting in $X \oplus X^* = 0$, so the number of output differences equal to zero is 2^n. In the case of DES, $\mathcal{D}[0][0] = 2^6 = 64$.

High values and differential uniformity

For a Differential Cryptanalysis attack, an attacker looks for the highest counts in the DDT, since they mean highest probability.

(Differential uniformity [60]) *The differential uniformity of an S-Box S is the maximum value in its Difference Distribution Table, disconsidering the* [0][0] *entry.*

According to [60], an S-box is said *differentially x-uniform* if, in its DDT, x is the differential uniformity. The larger x is, the more non-uniform is the distribution of differences in the S-Box. Some attacks could exploit S-Boxes with high differential uniformity. The differential uniformity of all the 8 DES S-Boxes is 16. In the case of DES, though, Biham and Shamir do not use the maximum DDT entries for their attack.

4.3.4 Expanding the concept to the F function

It is also relevant to understand how the probabilities computed for S-box input and output XORs are extended to input and output XORs of the F function.

(Probability of an input difference causing an output difference) *We say that X' may cause Y' by an S-box S if there is a pair in which the input difference of the S-box equals X' and the output difference of the S-box equals Y'. Furthermore, X' may cause Y' with probability p if, for a fraction p of the pairs with input XOR X', the output XOR is equal to Y'. Hence X' may cause Y' with probability $\mathcal{D}[X'][Y']/2^n$ where n_s is the input size of the S-Box in bits.*

In the case of DES, X' is a 6-bit S-Box input difference and Y' is a 4-bit S-Box output difference. Therefore, X' may cause Y' with probability $\mathcal{D}[X'][Y']/64$.

In the F function of DES ($F : \mathbb{Z}_2^{32} \to \mathbb{Z}_2^{32}$), the input is expanded by E, resulting in 48 bits. Then, each 6-bit slice is fed to an S-Box, which outputs 4 bits. The 4-bit outputs are concatenated and sent to the permutation P, which concludes the application of F. Formally, $F = P(S(E(x \oplus k)))$, where x is the cipher state and k is a subkey.

Let X_i be inputs to each S-box, $1 \leq i \leq 8$, Y_i be outputs of each S-box, p_i be the probability that X_i may cause Y_i by the i-th S-box, $X = X_1||X_2||X_3||X_4||X_5||X_6||X_7||X_8$ be the output of E, and $Y = Y_1||Y_2||Y_3||Y_4||Y_5||Y_6||Y_7||Y_8$ be the output after all S-Boxes are applied to the 6-bit slices.

The probability that X may cause Y by the F function is the product of the p_i, hence $p_1 \cdot p_2 \cdot p_3 \cdot p_4 \cdot p_5 \cdot p_6 \cdot p_7 \cdot p_8$, because, for X to cause Y, X_1 must cause Y_1, X_2 must cause Y_2, and so forth, and these probabilities are independent (that is, X_i causing Y_i is independent of X_j causing Y_j for $i \neq j$).

4.3.5 Characteristics

The concept of *characteristic* allows us to push the analysis to a round of the cipher and further to N rounds.

(Characteristic) *Associated with any pair of encryptions are:*

- *the XOR of the two plaintexts;*
- *the XOR of the two ciphertexts;*

- *the XORs of the inputs of each round, in the two executions;*
- *the XORs of the outputs of each round in the two executions.*

These XOR values form an N-round characteristic*, which has a probability. The probability of a characteristic is that a random pair with the chosen plaintext XOR has the round and ciphertext XORs specified.*

In [10], the authors show examples of 1-round, 3-round and 5-round characteristics. However, as the number of rounds increases, the probabilities decrease. Thus it is necessary to find characteristics such that concatenating them allows us to reach further rounds of the cipher whilst keeping probabilities that are sufficiently high for key recovery by the end of the attack, if possible, maximum probabilities. A relevant type of characteristic is *the iterative characteristic*, which is a characteristic that can be concatenated to itself. In [10], they find an iterative characteristic with probability approximately equal to 1/234.

4.3.6 The difficulties incurred by the DES design

(Active S-Box) *An* active *S-box is one for which the* input XOR is not zero*.*

In a Differential Cryptanalysis attack, we try to maximize probabilities. Therefore, it is relevant to be able to activate as few S-boxes as possible. If too many S-boxes are activated at once, the probabilities decrease drastically. The data complexity required for a successful attack is inversely proportional to the probability. If more data than available in the codebook is required, then the attack becomes infeasible. Activating only one S-box per DES iteration would be the ideal scenario, but DES' design does not allow that, as explained in [14]. The solution found by [10] is to activate three S-boxes.

Due to the expansion function E of DES, bits are shared between S-boxes, and thus 6-bit input differences which activate only one S-box are restricted to X' of the form $00ab00$, a, b non-zero, i.e X' with all bits zero except the middle two bits, which enter only the S-box we would like to activate.

For each single S-box of DES, from S_1 to S_8, the entries of $\mathcal{D}[X'][0]$ happen to be *zero*, for *all X' of this form*, thus rendering an attack which activates only one S-box infeasible. This design criteria of DES, as explained by [14], serves especially to thwart Differential Cryptanalysis. In order to find an input difference X' with adequate form such that $\mathcal{D}[X'][0] \neq 0$, it is necessary to analyse more than one S-box at a time, for instance, 2 or 3 S-boxes at a time.

When activating more than one S-Box at a time, we would also like to take *non-zero input XORs which lead to zero output XORs.* Because of that, the column 0 of the DDT is very important — when the output XOR is zero, it is easier to keep maximum probability through the rounds of the cipher.

In [10], the authors are able to break DES using an iterative characteristic with approximately 1/234 probability and other selected characteristics with probabilities that allow key recovery by the end of the attack. Furthermore, the authors describe how variants of DES (e.g DES without the permutation P, DES with a different S-box order) would be much more vulnerable to the attack than the original design.

4.4 Joint difference distribution tables

A *joint difference distribution table* eases the process of finding characteristics by providing counts of input and output differences, i.e probabilities, associated to adjacent S-boxes. One could build joint DDTs for two adjacent S-boxes, three, four, five, or any number of S-boxes. However, the DDTs get bigger and bigger, and so does the computational cost to obtain them. In this section, we discuss joint DDTs for two S-boxes and for 3 S-boxes.

The first thing about joint DDTs is that *they are key dependent*, because the key bits change from one S-box to another. For a single DES S-box, as explained in Section 4.3.1, it is not necessary to take the key into account when calculating the DDT, because the input difference is key independent.

However, this is not true for *joint DDTs of different S-boxes* of DES, because the expansion E causes state bits to be shared between different S-boxes and the key bits which are input to two neighbor S-Boxes are different. They are not cancelled by the XOR when computing X'. For two S-boxes, it is necessary to take four shared key bits into account and, for three, it is necessary to consider eight shared key bits. Thus in reality there are 16 joint DDTs for each S-box pair, and 256 DDTs for each tuple of three S-boxes.

Note that joint DDTs refer only to *neighbor S-Boxes*, not to any chosen tuple of S-Boxes. Neighbor S-Boxes are S-Boxes which share bits due to the expansion E. The possible pairs are:

- S_1, S_2
- S_2, S_3
- S_3, S_4
- S_4, S_5
- S_5, S_6
- S_6, S_7
- S_7, S_8
- S_8, S_1

For the joint DDTs of 3 neighbor S-Box, the possible tuples are:

- S_1, S_2, S_3
- S_2, S_3, S_4
- S_3, S_4, S_5
- S_4, S_5, S_6
- S_5, S_6, S_7
- S_6, S_7, S_8
- S_7, S_8, S_1
- S_8, S_1, S_2

4.4.1 DDTs for 2 simultaneous S-boxes

As an example, take S-boxes S_1 and S_2 of DES. Algorithm 5 shows how their joint DDT can be computed. Because of the expansion function of DES, bits are shared between consecutive S-boxes, hence if $x_1x_2x_3x_4x_5x_6$ is input to S_1, then the bits x_5 and x_6 are shared with S_2. Thus the input of S_2 can be described as $x_5x_6x_7x_8x_9x_{10}$, and we can limit the loop to $2^{10} - 1$, obtaining the inputs of each S-box through bit manipulation on X and X^*. Furthermore, we can take into account only 4 key bits, k_1 and k_2, which are XORed to x_5 and x_6 on input to S_1, and k_3 and k_4, which are XORed to x_5 and x_6 on input to S_2. There will be 16 DDTs, one for each possible combination of shared key bits.

Algorithm 5 Obtaining Joint DDT of S_1 and S_2 of DES

```
Require: $S$-boxes $S_1$ and $S_2$, 4 shared key bits $k_1, k_2, k_3, k_4$
Ensure: A $2^{10} \times 2^8$ table, the joint DDT of $S_1$ and $S_2$
 1: Initialize $\mathcal{D}$ with zero in all entries
 2: for $X = 0, ..., 2^{10} - 1$ do
 3:     for $X^* = 0, ..., 2^{10} - 1$ do
 4:         $X_1 \leftarrow$ the first 6 bits of $X$
 5:         $X_2 \leftarrow$ the last 6 bits of $X$
 6:         $X_1^* \leftarrow$ the first 6 bits of $X^*$
 7:         $X_2^* \leftarrow$ the last 6 bits of $X^*$
 8:         $X_1 \leftarrow X1 \oplus 0000||k_1||k_2$
 9:         $X_2 \leftarrow X_2 \oplus k_3||k_4||0000$
10:         $Y_1 \leftarrow S_1(X_1)$
11:         $Y_1^* \leftarrow S_1(X_1^*)$
12:         $Y_2 \leftarrow S_2(X_2)$
13:         $Y_2^* \leftarrow S_2(X_2^*)$
14:         $X' \leftarrow X \oplus X^*$
15:         $Y' \leftarrow (Y_1 \oplus Y_1^*)||(Y_2 \oplus Y_2^*)$
16:         $\mathcal{D}[X'][Y'] \leftarrow \mathcal{D}[X'][Y'] + 1$
17:     end for
18: end for
19: return $\mathcal{D}$
```

The [0,0] entry contains $2^{10} = 1024$, since it accumulates all the 2^{10} cases for which $X' = 0$ and thus $Y' = 0$. As for inputs of the form $00abcdef00$ in the 0 column, which activate only S_1 and S_2 and would be attractive for a DC attack, the entries are *also zero*, hence the need for calculating joint DDTs of 3 S-boxes. As explained by Coppersmith in [14], this is intentional in the design of DES, in an attempt to render DC attacks difficult. However, Biham and Shamir [10] succeed by activating 3 adjacent S-Boxes.

4.4.2 3 simultaneous S-boxes

For 3 S-boxes, we must loop from 0 to $2^{14} - 1$, so $X = x_1x_2x_3x_4x_5x_6x_7x_8x_9x_{10}x_{11}x_{12}x_{13}x_{14}$, with x_5 and x_6 being shared between the first two S-boxes, and x_9 and x_{10} being shared between the last two S-boxes. There are 8 key bits to be taken into account, and therefore 256 DDTs for each tuple of 3 neighbor S-boxes.

The [0,0] entry contains $2^{14} = 16384$. The inputs which activate only the 3 S-boxes are of the form $X = 00abcdefghij00$ and, this time, input XORs in this form contain non-zero entries in the column 0, with highest values alternating from 112 and 28, depending on the value of the key. In [43], it is explained that the 1/234 probability found by [10] is an average ($112 + 28 = 140$, and $140/2^{14} \approx 1/234$), because for some S-boxes it will be $112/2^{14}$ and, for others, $28/2^{14}$. These values, 112 and 28, appear only for the $(S1, S2, S3)$ tuple. For the other tuples, the values are smaller.

4.5 DDTs for DES

4.5.1 Single DDTs

Tables 4.1 to 4.8 shows the DDTs for S-Boxes from S_1 to S_8 of DES, respectively. The columns from 0_x to F_x are output differences, the rows from 0_x to 63_x are input differences. Input and output differences are presented in hexadecimal notation, but the entries are decimal values. It is possible to observe that the differential uniformity is 16, in all the S-Boxes.

	0$_x$	**1**$_x$	**2**$_x$	**3**$_x$	**4**$_x$	**5**$_x$	**6**$_x$	**7**$_x$	**8**$_x$	**9**$_x$	**A**$_x$	**B**$_x$	**C**$_x$	**D**$_x$	**E**$_x$	**F**$_x$
0$_x$	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1$_x$	0	0	0	6	0	2	4	4	0	10	12	4	10	6	2	4
2$_x$	0	0	0	8	0	4	4	4	0	6	8	6	12	6	4	2
3$_x$	14	4	2	2	10	6	4	2	6	4	4	0	2	2	2	0
4$_x$	0	0	0	6	0	10	10	6	0	4	6	4	2	8	6	2
5$_x$	4	8	6	2	2	4	4	2	0	4	4	0	12	2	4	6
6$_x$	0	4	2	4	8	2	6	2	8	4	4	2	4	2	0	12
7$_x$	2	4	10	4	0	4	8	4	2	4	8	2	2	2	4	4
8$_x$	0	0	0	12	0	8	8	4	0	6	2	8	8	2	2	4
9$_x$	10	2	4	0	2	4	6	0	2	2	8	0	10	0	2	12
A$_x$	0	8	6	2	2	8	6	0	6	4	6	0	4	0	2	10
B$_x$	2	4	0	10	2	2	4	0	2	6	2	6	6	4	2	12
C$_x$	0	0	0	8	0	6	6	0	0	6	6	4	6	6	14	2
D$_x$	6	6	4	8	4	8	2	6	0	6	4	6	0	2	0	2
E$_x$	0	4	8	8	6	6	4	0	6	6	4	0	0	4	0	8
F$_x$	2	0	2	4	4	6	4	2	4	8	2	2	2	6	8	8
10$_x$	0	0	0	0	0	0	2	14	0	6	6	12	4	6	8	6
11$_x$	6	8	2	4	6	4	8	6	4	0	6	6	0	4	0	0
12$_x$	0	8	4	2	6	6	4	6	6	4	2	6	6	0	4	0
13$_x$	2	4	4	6	2	0	4	6	2	0	6	8	4	6	4	6
14$_x$	0	8	8	0	10	0	4	2	8	2	2	4	4	8	4	0
15$_x$	0	4	6	4	2	2	4	10	6	2	0	10	0	4	6	4
16$_x$	0	8	10	8	0	2	2	6	10	2	0	2	0	6	2	6
17$_x$	4	4	6	0	10	6	0	2	4	4	4	6	6	6	2	0
18$_x$	0	6	6	0	8	4	2	2	2	4	6	8	6	6	2	2
19$_x$	2	6	2	4	0	8	4	6	10	4	0	4	2	8	4	0
1A$_x$	0	6	4	0	4	6	6	6	6	2	2	0	4	4	6	8
1B$_x$	4	4	2	4	10	6	6	4	6	2	2	4	2	2	4	2

1C$_x$	0	10	10	6	6	0	0	12	6	4	0	0	2	4	4	0
1D$_x$	4	2	4	0	8	0	0	2	10	0	2	6	6	6	14	0
1E$_x$	0	2	6	0	14	2	0	0	6	4	10	8	2	2	6	2
1F$_x$	2	4	10	6	2	2	2	8	6	8	0	0	0	4	6	4
20$_x$	0	0	0	10	0	12	8	2	0	6	4	4	4	2	0	12
21$_x$	0	4	2	4	4	8	10	0	4	4	10	0	4	0	2	8
22$_x$	10	4	6	2	2	8	2	2	2	2	6	0	4	0	4	10
23$_x$	0	4	4	8	0	2	6	0	6	6	2	10	2	4	0	10
24$_x$	12	0	0	2	2	2	2	0	14	14	2	0	2	6	2	4
25$_x$	6	4	4	12	4	4	4	10	2	2	2	0	4	2	2	2
26$_x$	0	0	4	10	10	10	2	4	0	4	6	4	4	4	2	0
27$_x$	10	4	2	0	2	4	2	0	4	8	0	4	8	8	4	4
28$_x$	12	2	2	8	2	6	12	0	0	2	6	0	4	0	6	2
29$_x$	4	2	2	10	0	2	4	0	0	14	10	2	4	6	0	4
2A$_x$	4	2	4	6	0	2	8	2	2	14	2	6	2	6	2	2
2B$_x$	12	2	2	2	4	6	6	2	0	2	6	2	6	0	8	4
2C$_x$	4	2	2	4	0	2	10	4	2	2	4	8	8	4	2	6
2D$_x$	6	2	6	2	8	4	4	4	2	4	6	0	8	2	0	6
2E$_x$	6	6	2	2	0	2	4	6	4	0	6	2	12	2	6	4
2F$_x$	2	2	2	2	2	6	8	8	2	4	4	6	8	2	4	2
30$_x$	0	4	6	0	12	6	2	2	8	2	4	4	6	2	2	4
31$_x$	4	8	2	10	2	2	2	2	6	0	0	2	2	4	10	8
32$_x$	4	2	6	4	4	2	2	4	6	6	4	8	2	2	8	0
33$_x$	4	4	6	2	10	8	4	2	4	0	2	2	4	6	2	4
34$_x$	0	8	**16**	6	2	0	0	12	6	0	0	0	0	8	0	6
35$_x$	2	2	4	0	8	0	0	0	14	4	6	8	0	2	14	0
36$_x$	2	6	2	2	8	0	2	2	4	2	6	8	6	4	10	0
37$_x$	2	2	12	4	2	4	4	10	4	4	2	6	0	2	2	4
38$_x$	0	6	2	2	2	0	2	2	4	6	4	4	4	6	10	10
39$_x$	6	2	2	4	12	6	4	8	4	0	2	4	2	4	4	0
3A$_x$	6	4	6	4	6	8	0	6	2	2	6	2	2	6	4	0
3B$_x$	2	6	4	0	0	2	4	6	4	6	8	6	4	4	6	2
3C$_x$	0	10	4	0	12	0	4	2	6	0	4	12	4	4	2	0
3D$_x$	0	8	6	2	2	6	0	8	4	4	0	4	0	12	4	4
3E$_x$	4	8	2	2	2	4	4	14	4	2	0	2	0	8	4	4
3F$_x$	4	8	4	2	4	0	2	4	4	2	4	8	8	6	2	2

Table 4.1: DDT for S_1 of DES

	0$_x$	**1**$_x$	**2**$_x$	**3**$_x$	**4**$_x$	**5**$_x$	**6**$_x$	**7**$_x$	**8**$_x$	**9**$_x$	**A**$_x$	**B**$_x$	**C**$_x$	**D**$_x$	**E**$_x$	**F**$_x$
0$_x$	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1$_x$	0	0	0	4	0	2	6	4	0	14	8	6	8	4	6	2
2$_x$	0	0	0	2	0	4	6	4	0	0	4	6	10	10	12	6
3$_x$	4	8	4	8	4	6	4	2	4	2	2	4	6	2	0	4
4$_x$	0	0	0	0	0	6	0	14	0	6	10	4	10	6	4	4
5$_x$	2	0	4	8	2	4	6	6	2	0	8	4	2	4	10	2
6$_x$	0	12	6	4	6	4	6	2	2	10	2	8	2	0	0	0
7$_x$	4	6	6	4	2	4	4	2	6	4	2	4	4	6	0	6
8$_x$	0	0	0	4	0	4	0	8	0	10	**16**	6	6	0	6	4
9$_x$	14	2	4	10	2	8	2	6	2	4	0	0	2	2	2	4
A$_x$	0	6	6	2	10	4	10	2	6	2	2	4	2	2	4	2
B$_x$	6	2	2	0	2	4	6	2	10	2	0	6	6	4	4	8
C$_x$	0	0	0	4	0	14	0	10	0	6	2	4	4	8	6	6
D$_x$	6	2	6	2	10	2	0	4	0	10	4	2	8	2	2	4
E$_x$	0	6	12	8	0	4	2	0	8	2	4	4	6	2	0	6

$\mathbf{F}_x$	0	8	2	0	6	6	8	2	4	4	4	6	8	0	4	2
$\mathbf{10}_x$	0	0	0	8	0	4	10	2	0	2	8	10	0	10	6	4
$\mathbf{11}_x$	6	6	4	6	4	0	6	4	8	2	10	2	2	4	0	0
$\mathbf{12}_x$	0	6	2	6	2	4	12	4	6	4	0	4	4	6	2	2
$\mathbf{13}_x$	4	0	4	0	8	6	6	0	0	2	0	6	4	8	2	14
$\mathbf{14}_x$	0	6	6	4	10	0	2	12	6	2	2	2	4	4	2	2
$\mathbf{15}_x$	6	8	2	0	8	2	0	2	2	2	2	2	2	14	10	2
$\mathbf{16}_x$	0	8	6	4	2	2	4	2	6	4	6	2	6	0	6	6
$\mathbf{17}_x$	6	4	8	6	4	4	0	4	6	2	4	4	4	2	4	2
$\mathbf{18}_x$	0	6	4	6	10	4	0	2	4	8	0	0	4	8	2	6
$\mathbf{19}_x$	2	4	6	4	4	2	4	2	6	4	6	8	0	6	4	2
$\mathbf{1A}_x$	0	6	8	4	2	4	2	2	8	2	2	6	2	4	4	8
$\mathbf{1B}_x$	0	6	4	4	0	12	6	4	2	2	2	4	4	2	10	2
$\mathbf{1C}_x$	0	4	6	6	12	0	4	0	10	2	6	2	0	0	10	2
$\mathbf{1D}_x$	0	6	2	2	6	0	4	**16**	4	4	2	0	0	4	6	8
$\mathbf{1E}_x$	0	4	8	2	10	6	6	0	8	4	0	2	4	4	0	6
$\mathbf{1F}_x$	4	2	6	6	2	2	2	4	8	6	10	6	4	0	0	2
$\mathbf{20}_x$	0	0	0	2	0	12	10	4	0	0	0	2	14	2	8	10
$\mathbf{21}_x$	0	4	6	8	2	10	4	2	2	6	4	2	6	2	0	6
$\mathbf{22}_x$	4	12	8	4	2	2	0	0	2	8	8	6	0	6	0	2
$\mathbf{23}_x$	8	2	0	2	8	4	2	6	4	8	2	2	6	4	2	4
$\mathbf{24}_x$	10	4	0	0	0	4	0	2	6	8	6	10	8	0	2	4
$\mathbf{25}_x$	6	0	12	2	8	6	10	0	0	8	2	6	0	0	2	2
$\mathbf{26}_x$	2	2	4	4	2	2	10	14	2	0	4	2	2	4	6	4
$\mathbf{27}_x$	6	0	0	2	6	4	2	4	4	4	8	4	8	0	6	6
$\mathbf{28}_x$	8	0	8	2	4	12	2	0	2	6	2	0	6	2	0	10
$\mathbf{29}_x$	0	2	4	10	2	8	6	4	0	10	0	2	10	0	2	4
$\mathbf{2A}_x$	4	0	4	8	6	2	4	4	6	6	2	6	2	2	4	4
$\mathbf{2B}_x$	2	2	6	4	0	2	2	6	2	8	8	4	4	4	8	2
$\mathbf{2C}_x$	10	6	8	6	0	6	4	4	4	2	4	4	0	0	2	4
$\mathbf{2D}_x$	2	2	2	4	0	0	0	2	8	4	4	6	10	2	14	4
$\mathbf{2E}_x$	2	4	0	2	10	4	2	0	2	2	6	2	8	8	10	2
$\mathbf{2F}_x$	12	4	6	8	2	6	2	8	0	4	0	2	0	8	2	0
$\mathbf{30}_x$	0	4	0	2	4	4	8	6	10	6	2	12	0	0	0	6
$\mathbf{31}_x$	0	10	2	0	6	2	10	2	6	0	2	0	6	6	4	8
$\mathbf{32}_x$	8	4	6	0	6	4	4	8	4	6	8	0	2	2	2	0
$\mathbf{33}_x$	2	2	6	10	2	0	0	6	4	4	12	8	4	2	2	0
$\mathbf{34}_x$	0	12	6	4	6	0	4	4	4	0	4	6	4	2	4	4
$\mathbf{35}_x$	0	12	4	6	2	4	4	0	10	0	0	8	0	8	0	6
$\mathbf{36}_x$	8	2	4	0	4	0	4	2	0	8	4	2	6	**16**	2	2
$\mathbf{37}_x$	6	2	2	2	6	6	4	8	2	2	6	2	2	2	4	8
$\mathbf{38}_x$	0	8	8	10	6	2	2	0	4	0	4	2	4	0	4	10
$\mathbf{39}_x$	0	2	0	0	8	0	10	4	10	0	8	4	4	4	4	6
$\mathbf{3A}_x$	4	0	2	8	4	2	2	2	4	8	2	0	4	10	10	2
$\mathbf{3B}_x$	**16**	4	4	2	8	2	2	6	4	4	4	2	0	2	2	2
$\mathbf{3C}_x$	0	2	6	2	8	4	6	0	10	2	2	4	4	10	4	0
$\mathbf{3D}_x$	0	**16**	10	2	4	2	4	2	8	0	0	8	0	6	2	0
$\mathbf{3E}_x$	4	4	0	10	2	4	2	14	4	2	6	6	0	0	6	0
$\mathbf{3F}_x$	4	0	0	2	0	8	2	4	0	2	4	4	4	14	10	6

Table 4.2: DDT for S_2 of DES

	$\mathbf{0}_x$	$\mathbf{1}_x$	$\mathbf{2}_x$	$\mathbf{3}_x$	$\mathbf{4}_x$	$\mathbf{5}_x$	$\mathbf{6}_x$	$\mathbf{7}_x$	$\mathbf{8}_x$	$\mathbf{9}_x$	$\mathbf{A}_x$	$\mathbf{B}_x$	$\mathbf{C}_x$	$\mathbf{D}_x$	$\mathbf{E}_x$	$\mathbf{F}_x$
$\mathbf{0}_x$	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\mathbf{1}_x$	0	0	0	2	0	4	2	12	0	14	0	4	8	2	6	10

2$_x$	0	0	0	2	0	2	0	8	0	4	12	10	4	6	8	8
3$_x$	8	6	10	4	8	6	0	6	4	4	0	0	0	4	2	2
4$_x$	0	0	0	4	0	2	4	2	0	12	8	4	6	8	10	4
5$_x$	6	2	4	8	6	10	6	2	2	8	2	0	2	0	4	2
6$_x$	0	10	6	6	10	0	4	12	2	4	0	0	6	4	0	0
7$_x$	2	0	0	4	4	4	4	2	10	4	4	8	4	4	4	6
8$_x$	0	0	0	10	0	4	4	6	0	6	6	6	6	0	8	8
9$_x$	10	2	0	2	10	4	6	2	0	6	0	4	6	2	4	6
A$_x$	0	10	6	0	14	6	4	0	4	6	6	0	4	0	2	2
B$_x$	2	6	2	10	2	2	4	0	4	2	6	0	2	8	14	0
C$_x$	0	0	0	8	0	12	12	4	0	8	0	4	2	10	2	2
D$_x$	8	2	8	0	0	4	2	0	2	8	14	2	6	2	4	2
E$_x$	0	4	4	2	4	2	4	4	10	4	4	4	4	4	2	8
F$_x$	4	6	4	6	2	2	4	8	6	2	6	2	0	6	2	4
10$_x$	0	0	0	4	0	12	4	8	0	4	2	6	2	14	0	8
11$_x$	8	2	2	6	4	0	2	0	8	4	12	2	10	0	2	2
12$_x$	0	2	8	2	4	8	0	8	8	0	2	2	4	2	14	0
13$_x$	4	4	12	0	2	2	2	10	2	2	2	2	4	4	4	8
14$_x$	0	6	4	4	6	4	6	2	8	6	6	2	2	0	0	8
15$_x$	4	8	2	8	2	4	8	0	4	2	2	2	2	6	8	2
16$_x$	0	6	10	2	8	4	2	0	2	2	2	8	4	6	4	4
17$_x$	0	6	6	0	6	2	4	4	6	2	2	10	6	8	2	0
18$_x$	0	8	4	6	6	0	6	2	4	0	4	2	10	0	6	6
19$_x$	4	2	4	8	4	2	10	2	2	2	6	8	2	6	0	2
1A$_x$	0	8	6	4	4	0	6	4	4	8	0	10	2	2	2	4
1B$_x$	4	10	2	0	2	4	2	4	8	2	2	8	4	2	8	2
1C$_x$	0	6	8	8	4	2	8	0	12	0	10	0	4	0	2	0
1D$_x$	0	2	0	6	2	8	4	6	2	0	4	2	4	10	0	14
1E$_x$	0	4	8	2	4	6	0	4	10	0	2	6	4	8	4	2
1F$_x$	0	6	8	0	10	6	4	6	4	2	2	10	4	0	0	2
20$_x$	0	0	0	0	0	4	4	8	0	2	2	4	10	**16**	12	2
21$_x$	10	8	8	0	8	4	2	4	0	6	6	6	0	0	2	0
22$_x$	12	6	4	4	2	4	10	2	0	4	4	2	4	4	0	2
23$_x$	2	2	0	6	0	2	4	0	4	12	4	2	6	4	8	8
24$_x$	4	8	2	12	6	4	2	10	2	2	2	4	2	0	4	0
25$_x$	6	0	2	0	8	2	0	2	8	8	2	2	4	4	10	6
26$_x$	6	2	0	4	4	0	4	0	4	2	14	0	8	10	0	6
27$_x$	0	2	4	**16**	8	6	6	6	0	2	4	4	0	2	2	2
28$_x$	6	2	10	0	6	4	0	4	4	2	4	8	2	2	8	2
29$_x$	0	2	8	4	0	4	0	6	4	10	4	8	4	4	4	2
2A$_x$	2	6	0	4	2	4	4	6	4	8	4	4	4	2	4	6
2B$_x$	10	2	6	6	4	4	8	0	4	2	2	0	2	4	4	6
2C$_x$	10	4	6	2	4	2	2	2	4	10	4	4	0	2	6	2
2D$_x$	4	2	4	4	4	2	4	**16**	2	0	0	4	4	2	6	6
2E$_x$	4	0	2	10	0	6	10	4	2	6	6	2	2	0	2	8
2F$_x$	8	2	0	0	4	4	4	2	6	4	6	2	4	8	4	6
30$_x$	0	10	8	6	2	0	4	2	10	4	4	6	2	0	6	0
31$_x$	2	6	2	0	4	2	8	8	2	2	2	0	2	12	6	6
32$_x$	2	0	4	8	2	8	4	4	8	4	2	8	6	2	0	2
33$_x$	4	4	6	8	6	6	0	2	2	2	6	4	12	0	0	2
34$_x$	0	6	2	2	**16**	2	2	2	12	2	4	0	4	2	0	8
35$_x$	4	6	0	10	8	0	2	2	6	0	0	6	2	10	2	6
36$_x$	4	4	4	4	0	6	6	4	4	4	4	4	0	6	2	8
37$_x$	4	8	2	4	2	2	6	0	2	4	8	4	10	0	6	2
38$_x$	0	8	12	0	2	2	6	6	2	10	2	2	0	8	0	4

39$_x$	2	6	4	0	6	4	6	4	8	0	4	4	2	4	8	2
3A$_x$	6	0	2	2	4	6	4	4	4	2	2	6	12	2	6	2
3B$_x$	2	2	6	0	0	10	4	8	4	2	4	8	4	4	0	6
3C$_x$	0	2	4	2	12	2	0	6	2	0	2	8	4	6	4	10
3D$_x$	4	6	8	6	2	2	2	2	10	2	6	6	2	4	2	0
3E$_x$	8	6	4	4	2	10	2	0	2	2	4	2	4	2	10	2
3F$_x$	2	6	4	0	0	10	8	2	2	8	6	4	6	2	0	4

Table 4.3: DDT for S_3 of DES

	0$_x$	**1**$_x$	**2**$_x$	**3**$_x$	**4**$_x$	**5**$_x$	**6**$_x$	**7**$_x$	**8**$_x$	**9**$_x$	**A**$_x$	**B**$_x$	**C**$_x$	**D**$_x$	**E**$_x$	**F**$_x$
0$_x$	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1$_x$	0	0	0	0	0	**16**	**16**	0	0	**16**	**16**	0	0	0	0	0
2$_x$	0	0	0	8	0	4	4	8	0	4	4	8	8	8	8	0
3$_x$	8	6	2	0	2	4	8	2	6	0	4	6	0	6	2	8
4$_x$	0	0	0	8	0	0	12	4	0	12	0	4	8	4	4	8
5$_x$	4	2	2	8	2	12	0	2	2	0	12	2	8	2	2	4
6$_x$	0	8	8	4	8	8	0	0	8	0	8	0	4	0	0	8
7$_x$	4	2	6	4	6	0	**16**	6	2	0	0	2	4	2	6	4
8$_x$	0	0	0	4	0	8	4	8	0	4	8	8	4	8	8	0
9$_x$	8	4	4	4	4	0	8	4	4	0	0	4	4	4	4	8
A$_x$	0	6	6	0	6	4	4	6	6	4	4	6	0	6	6	0
B$_x$	0	12	0	8	0	0	0	0	12	0	0	12	8	12	0	0
C$_x$	0	0	0	4	0	8	4	8	0	4	8	8	4	8	8	0
D$_x$	8	4	4	4	4	0	0	4	4	8	0	4	4	4	4	8
E$_x$	0	6	6	4	6	0	4	6	6	4	0	6	4	6	6	0
F$_x$	0	6	6	4	6	4	0	6	6	0	4	6	4	6	6	0
10$_x$	0	0	0	0	0	8	12	4	0	12	8	4	0	4	4	8
11$_x$	4	2	2	**16**	2	4	0	2	2	0	4	2	**16**	2	2	4
12$_x$	0	0	0	8	0	4	4	8	0	4	4	8	8	8	8	0
13$_x$	8	2	6	0	6	4	0	6	2	8	4	2	0	2	6	8
14$_x$	0	8	8	0	8	0	8	0	8	8	0	0	0	0	0	**16**
15$_x$	8	4	4	0	4	8	0	4	4	0	8	4	0	4	4	8
16$_x$	0	8	8	4	8	8	0	0	8	0	8	0	4	0	0	8
17$_x$	4	6	2	4	2	0	0	2	6	**16**	0	6	4	6	2	4
18$_x$	0	8	8	8	8	4	0	0	8	0	4	0	8	0	0	8
19$_x$	4	4	4	0	4	4	**16**	4	4	0	4	4	0	4	4	4
1A$_x$	0	6	6	4	6	0	4	6	6	4	0	6	4	6	6	0
1B$_x$	0	6	6	4	6	4	0	6	6	0	4	6	4	6	6	0
1C$_x$	0	8	8	8	8	4	0	0	8	0	4	0	8	0	0	8
1D$_x$	4	4	4	0	4	4	0	4	4	**16**	4	4	0	4	4	4
1E$_x$	0	6	6	0	6	4	4	6	6	4	4	6	0	6	6	0
1F$_x$	0	0	12	8	12	0	0	12	0	0	0	0	8	0	12	0
20$_x$	0	0	0	8	0	0	0	12	0	0	0	12	8	12	12	0
21$_x$	0	4	8	0	8	4	8	8	4	0	4	4	0	4	8	0
22$_x$	8	2	2	0	2	4	8	6	2	8	4	6	0	6	6	0
23$_x$	4	6	2	8	2	4	0	2	6	0	4	6	8	6	2	4
24$_x$	0	6	6	4	6	4	0	6	6	0	4	6	4	6	6	0
25$_x$	0	8	4	4	4	0	0	4	8	8	0	8	4	8	4	0
26$_x$	0	6	6	0	6	4	8	2	6	8	4	2	0	2	2	8
27$_x$	4	6	2	8	2	4	0	2	6	0	4	6	8	6	2	4
28$_x$	**16**	4	4	0	4	4	4	4	4	4	4	4	0	4	4	0
29$_x$	0	6	2	8	2	4	0	2	6	8	4	6	8	6	2	0
2A$_x$	0	2	2	**16**	2	4	4	2	2	4	4	2	**16**	2	2	0
2B$_x$	8	0	4	0	4	8	**16**	4	0	0	8	0	0	0	4	8

2C$_x$	8	4	4	4	4	0	8	4	4	8	0	4	4	4	4	0
2D$_x$	4	2	6	4	6	8	0	6	2	0	8	2	4	2	6	4
2E$_x$	**16**	0	0	0	0	**16**	0	0	0	0	**16**	0	0	0	0	**16**
2F$_x$	**16**	0	0	0	0	0	**16**	0	0	**16**	0	0	0	0	0	**16**
30$_x$	0	6	6	4	6	4	0	6	6	0	4	6	4	6	6	0
31$_x$	0	8	4	4	4	0	0	4	8	8	0	8	4	8	4	0
32$_x$	**16**	6	6	4	6	0	4	2	6	4	0	2	4	2	2	0
33$_x$	0	2	6	4	6	8	8	6	2	0	8	2	4	2	6	0
34$_x$	0	12	12	8	12	0	0	0	12	0	0	0	8	0	0	0
35$_x$	0	4	8	0	8	4	8	8	4	0	4	4	0	4	8	0
36$_x$	0	2	2	4	2	0	4	6	2	4	0	6	4	6	6	**16**
37$_x$	0	2	6	4	6	8	8	6	2	0	8	2	4	2	6	0
38$_x$	0	4	4	0	4	4	4	4	4	4	4	4	0	4	4	**16**
39$_x$	0	6	2	8	2	4	0	2	6	8	4	6	8	6	2	0
3A$_x$	0	4	4	0	4	8	8	4	4	8	8	4	0	4	4	0
3B$_x$	**16**	4	4	0	4	0	0	4	4	0	0	4	0	4	4	**16**
3C$_x$	0	4	4	4	4	0	8	4	4	8	0	4	4	4	4	8
3D$_x$	4	2	6	4	6	8	0	6	2	0	8	2	4	2	6	4
3E$_x$	0	2	2	8	2	12	4	2	2	4	12	2	8	2	2	0
3F$_x$	8	4	0	8	0	0	0	0	4	**16**	0	4	8	4	0	8

Table 4.4: DDT for S_4 of DES

	0$_x$	**1**$_x$	**2**$_x$	**3**$_x$	**4**$_x$	**5**$_x$	**6**$_x$	**7**$_x$	**8**$_x$	**9**$_x$	**A**$_x$	**B**$_x$	**C**$_x$	**D**$_x$	**E**$_x$	**F**$_x$
0$_x$	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1$_x$	0	0	0	4	0	10	8	6	0	4	2	2	12	10	2	4
2$_x$	0	0	0	4	0	10	6	4	0	6	4	2	4	8	10	6
3$_x$	8	2	4	6	4	4	2	2	6	8	6	4	4	0	2	2
4$_x$	0	0	0	8	0	4	10	6	0	6	6	4	8	6	0	6
5$_x$	12	2	0	4	0	4	8	2	4	0	**16**	2	0	2	0	8
6$_x$	0	8	4	6	4	6	2	2	4	4	6	0	6	0	2	10
7$_x$	2	0	4	8	4	2	6	6	2	8	6	2	2	0	6	6
8$_x$	0	0	0	2	0	8	10	4	0	4	10	4	8	4	4	6
9$_x$	8	6	0	4	0	6	6	2	2	10	2	8	6	2	0	2
A$_x$	0	6	8	6	0	8	0	0	8	10	4	2	8	0	0	4
B$_x$	4	2	2	4	8	10	6	4	2	6	2	2	6	2	2	2
C$_x$	0	0	0	10	0	2	10	2	0	6	10	6	6	6	2	4
D$_x$	10	4	2	2	0	6	**16**	0	0	2	10	2	2	4	0	4
E$_x$	0	6	4	8	4	6	10	2	4	4	4	2	4	0	2	4
F$_x$	4	4	0	8	0	2	0	2	8	2	4	2	8	4	4	12
10$_x$	0	0	0	0	0	4	4	12	0	2	8	10	4	6	12	2
11$_x$	6	6	10	10	4	0	2	6	2	4	0	6	2	4	2	0
12$_x$	0	2	4	2	10	4	0	10	8	6	0	6	0	6	6	0
13$_x$	0	0	6	2	8	0	0	4	4	6	2	8	2	8	10	4
14$_x$	0	12	2	6	4	0	4	4	8	4	4	4	6	2	4	0
15$_x$	4	8	0	2	8	0	2	4	2	2	4	2	4	8	8	6
16$_x$	0	6	10	2	14	0	2	2	4	4	0	6	0	4	6	4
17$_x$	0	6	8	4	8	4	0	2	8	4	0	2	2	8	6	2
18$_x$	0	10	8	0	6	4	0	4	4	4	6	4	4	4	0	6
19$_x$	0	4	6	2	4	4	2	6	4	2	2	4	12	2	10	0
1A$_x$	0	2	**16**	2	12	2	0	6	4	0	0	4	0	4	4	8
1B$_x$	2	8	12	0	0	2	2	6	8	4	0	6	0	0	8	6
1C$_x$	0	10	2	6	6	6	6	4	8	2	0	4	4	4	2	0
1D$_x$	4	6	2	0	8	2	4	6	6	0	8	6	2	4	2	4
1E$_x$	0	2	6	2	4	0	0	2	12	2	2	6	2	10	10	4

1F$_x$	0	6	8	4	8	8	0	6	6	2	0	6	0	6	2	2
20$_x$	0	0	0	8	0	8	2	6	0	4	4	4	6	6	8	8
21$_x$	0	0	0	6	6	2	6	4	6	10	14	4	0	0	4	2
22$_x$	14	4	0	10	0	2	12	2	2	2	10	2	0	0	2	2
23$_x$	2	0	0	4	2	2	10	4	0	8	8	2	6	8	0	8
24$_x$	6	2	8	4	4	4	6	2	2	6	6	2	6	2	2	2
25$_x$	6	0	0	8	2	8	2	6	6	4	2	2	4	2	6	6
26$_x$	12	0	0	4	0	4	4	4	0	8	4	0	12	8	0	4
27$_x$	12	2	0	2	0	12	2	2	4	4	8	4	8	2	2	0
28$_x$	2	8	4	6	2	4	6	0	6	6	4	0	2	2	2	10
29$_x$	6	4	6	8	8	4	6	2	0	0	2	2	10	0	2	4
2A$_x$	4	4	0	2	2	4	6	2	0	0	6	4	10	4	4	12
2B$_x$	4	6	2	6	0	0	12	2	0	4	12	2	6	4	0	4
2C$_x$	8	6	2	6	4	8	6	0	4	4	0	2	6	0	6	2
2D$_x$	4	4	0	4	0	6	4	2	4	12	0	4	4	6	4	6
2E$_x$	6	0	2	4	0	6	6	4	2	10	6	10	6	2	0	0
2F$_x$	10	4	0	2	2	6	10	2	0	2	2	4	6	2	2	10
30$_x$	0	4	8	4	6	4	0	6	10	4	2	4	2	6	4	0
31$_x$	0	6	6	4	10	2	0	0	4	4	0	0	4	6	12	6
32$_x$	4	6	0	2	6	4	6	0	6	0	4	6	4	10	6	0
33$_x$	8	10	0	14	8	0	0	8	2	0	2	4	0	4	4	0
34$_x$	0	4	4	2	14	4	0	8	6	8	2	2	0	4	6	0
35$_x$	0	4	**16**	0	8	4	0	4	4	4	0	8	0	4	4	4
36$_x$	4	4	4	6	2	2	2	12	2	4	4	8	2	4	4	0
37$_x$	4	2	2	2	4	2	0	8	2	2	2	12	6	2	8	6
38$_x$	0	4	8	4	12	0	0	8	10	2	0	0	0	4	2	10
39$_x$	0	8	12	0	2	2	2	2	12	4	0	8	0	4	4	4
3A$_x$	0	14	4	0	4	6	0	0	6	2	10	8	0	0	4	6
3B$_x$	0	2	2	2	4	4	8	6	8	2	2	2	6	14	2	0
3C$_x$	0	0	10	2	6	0	0	2	6	2	2	10	2	4	10	8
3D$_x$	0	6	12	2	4	8	0	8	8	2	2	0	2	2	4	4
3E$_x$	4	4	10	0	2	4	8	8	2	2	0	2	6	8	4	0
3F$_x$	8	6	6	0	4	2	2	4	4	2	8	6	2	4	6	0

Table 4.5: DDT for S_5 of DES

	0$_x$	**1**$_x$	**2**$_x$	**3**$_x$	**4**$_x$	**5**$_x$	**6**$_x$	**7**$_x$	**8**$_x$	**9**$_x$	**A**$_x$	**B**$_x$	**C**$_x$	**D**$_x$	**E**$_x$	**F**$_x$
0$_x$	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1$_x$	0	0	0	6	0	2	6	2	0	4	2	4	6	**16**	14	2
2$_x$	0	0	0	2	0	10	6	10	0	2	4	8	6	6	8	2
3$_x$	0	8	0	8	0	6	4	6	4	4	4	12	2	4	2	0
4$_x$	0	0	0	8	0	0	8	0	0	6	8	10	2	4	10	8
5$_x$	10	2	4	4	4	8	8	4	2	2	0	4	0	8	0	4
6$_x$	0	8	4	4	8	4	2	2	12	0	2	6	6	2	2	2
7$_x$	6	6	4	0	2	10	2	2	2	2	6	6	8	0	6	2
8$_x$	0	0	0	6	0	2	**16**	4	0	2	6	2	4	12	6	4
9$_x$	10	4	2	6	0	2	6	2	4	0	8	6	4	4	2	4
A$_x$	0	14	4	4	0	2	2	2	10	4	4	4	6	4	2	2
B$_x$	4	6	2	0	2	2	12	8	2	2	2	6	8	2	0	6
C$_x$	0	0	0	12	0	10	4	6	0	8	4	4	2	12	2	0
D$_x$	12	0	2	10	6	4	4	2	4	2	6	0	2	6	0	4
E$_x$	0	6	4	0	4	4	10	8	6	2	4	6	2	0	6	2
F$_x$	2	2	2	2	6	2	6	2	10	4	8	2	6	4	4	2
10$_x$	0	0	0	8	0	8	0	12	0	4	2	6	8	4	6	6
11$_x$	6	2	6	4	6	2	6	4	6	6	4	2	4	0	6	0

$\mathbf{12}_x$	0	8	4	2	0	4	2	0	4	10	6	2	8	6	4	4
$\mathbf{13}_x$	6	6	12	0	12	2	0	6	6	2	0	4	0	2	4	2
$\mathbf{14}_x$	0	4	6	2	8	6	0	2	6	10	4	0	2	4	6	4
$\mathbf{15}_x$	2	2	6	6	4	4	2	6	2	6	8	4	4	0	4	4
$\mathbf{16}_x$	0	4	14	6	8	4	2	6	2	0	2	0	4	2	0	10
$\mathbf{17}_x$	2	6	8	0	0	2	0	2	2	6	0	8	8	2	12	6
$\mathbf{18}_x$	0	4	6	6	8	4	2	2	6	4	6	4	2	4	2	4
$\mathbf{19}_x$	2	6	0	2	4	4	4	6	4	8	6	4	2	2	6	4
$\mathbf{1A}_x$	0	6	6	0	8	2	4	6	4	2	4	6	2	0	4	10
$\mathbf{1B}_x$	0	4	10	2	4	4	2	6	6	6	2	2	6	6	2	2
$\mathbf{1C}_x$	0	0	8	2	12	2	6	2	8	6	6	2	4	0	4	2
$\mathbf{1D}_x$	2	4	0	6	8	6	0	2	6	8	6	0	2	4	0	10
$\mathbf{1E}_x$	0	10	8	2	8	2	0	2	6	4	2	4	6	4	2	4
$\mathbf{1F}_x$	0	6	6	8	6	4	2	4	4	2	2	0	2	4	2	12
$\mathbf{20}_x$	0	0	0	0	0	6	6	4	0	4	8	8	4	6	10	8
$\mathbf{21}_x$	2	8	6	8	4	4	6	6	8	4	0	4	0	2	2	0
$\mathbf{22}_x$	**16**	2	4	6	2	4	2	0	6	4	8	2	0	2	2	4
$\mathbf{23}_x$	0	4	0	4	4	6	10	4	2	2	6	2	4	6	6	4
$\mathbf{24}_x$	10	8	0	6	12	6	10	4	8	0	0	0	0	0	0	0
$\mathbf{25}_x$	0	2	4	2	0	4	4	0	4	0	10	10	4	10	6	4
$\mathbf{26}_x$	2	2	0	12	2	2	6	2	4	4	8	0	6	6	8	0
$\mathbf{27}_x$	8	4	0	8	2	4	2	4	0	6	2	4	4	8	2	6
$\mathbf{28}_x$	6	8	4	6	0	4	2	2	4	8	2	6	4	2	2	4
$\mathbf{29}_x$	2	4	4	0	8	8	6	8	6	4	0	4	4	4	2	0
$\mathbf{2A}_x$	6	0	0	6	6	4	6	8	2	4	0	2	2	4	6	8
$\mathbf{2B}_x$	12	0	4	0	0	4	2	2	2	6	10	6	10	2	4	0
$\mathbf{2C}_x$	4	2	6	0	0	6	8	6	4	2	2	8	4	6	4	2
$\mathbf{2D}_x$	6	2	2	6	6	4	4	2	6	2	4	8	4	2	4	2
$\mathbf{2E}_x$	4	6	2	4	2	4	4	2	4	2	4	6	4	10	4	2
$\mathbf{2F}_x$	10	0	4	8	0	6	6	2	0	4	4	2	6	2	2	8
$\mathbf{30}_x$	0	12	8	2	0	6	0	0	6	6	0	2	8	2	6	6
$\mathbf{31}_x$	2	6	10	4	2	2	2	4	6	0	2	6	0	2	4	12
$\mathbf{32}_x$	4	2	2	8	10	8	8	6	0	2	2	4	4	2	2	0
$\mathbf{33}_x$	4	2	2	2	6	0	4	0	10	6	6	4	0	4	8	6
$\mathbf{34}_x$	0	4	4	2	6	4	0	4	6	2	6	4	2	8	0	12
$\mathbf{35}_x$	6	12	4	2	4	2	2	4	8	2	2	0	6	4	4	2
$\mathbf{36}_x$	0	2	2	4	4	4	4	0	2	10	12	4	0	10	4	2
$\mathbf{37}_x$	10	2	2	6	14	2	2	6	2	0	4	6	2	0	4	2
$\mathbf{38}_x$	0	4	14	0	8	2	0	4	4	4	2	0	8	2	4	8
$\mathbf{39}_x$	2	4	8	0	6	2	0	6	2	6	4	2	8	6	2	6
$\mathbf{3A}_x$	8	4	0	4	6	2	0	4	6	8	6	0	6	0	4	6
$\mathbf{3B}_x$	0	4	6	6	2	2	2	14	0	12	0	4	2	2	8	0
$\mathbf{3C}_x$	0	6	**16**	0	2	2	2	8	4	2	0	12	6	2	2	0
$\mathbf{3D}_x$	0	6	2	2	2	6	8	2	4	2	6	2	6	2	4	10
$\mathbf{3E}_x$	4	2	2	4	4	0	6	10	4	2	4	6	6	2	6	2
$\mathbf{3F}_x$	0	4	6	6	4	8	4	0	4	8	4	0	4	8	2	2

Table 4.6: DDT for S_6 of DES

	$\mathbf{0}_x$	$\mathbf{1}_x$	$\mathbf{2}_x$	$\mathbf{3}_x$	$\mathbf{4}_x$	$\mathbf{5}_x$	$\mathbf{6}_x$	$\mathbf{7}_x$	$\mathbf{8}_x$	$\mathbf{9}_x$	$\mathbf{A}_x$	$\mathbf{B}_x$	$\mathbf{C}_x$	$\mathbf{D}_x$	$\mathbf{E}_x$	$\mathbf{F}_x$
$\mathbf{0}_x$	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\mathbf{1}_x$	0	0	0	2	0	4	4	14	0	12	4	6	2	6	6	4
$\mathbf{2}_x$	0	0	0	0	0	12	2	2	0	4	0	4	8	12	6	14
$\mathbf{3}_x$	8	2	12	2	6	8	6	0	6	4	4	2	2	0	0	2
$\mathbf{4}_x$	0	0	0	8	0	4	4	8	0	8	8	12	2	6	2	2

5$_x$	6	0	0	2	8	0	8	4	0	2	6	0	10	6	6	6
6$_x$	0	2	12	0	8	4	8	2	4	4	4	2	6	0	6	2
7$_x$	4	6	4	12	0	4	2	0	0	14	2	6	4	0	0	6
8$_x$	0	0	0	8	0	0	6	10	0	4	12	4	6	6	0	8
9$_x$	10	8	4	8	6	2	2	0	2	6	8	2	0	6	0	0
A$_x$	0	10	6	2	12	2	4	0	4	4	6	4	4	0	0	6
B$_x$	0	2	2	2	4	8	6	4	4	0	4	2	6	4	2	14
C$_x$	0	0	0	4	0	4	8	4	0	2	6	0	14	12	8	2
D$_x$	6	6	2	4	2	6	4	6	6	4	8	8	0	2	0	0
E$_x$	0	12	10	10	0	2	4	2	8	6	4	2	0	0	2	2
F$_x$	2	0	0	0	6	8	8	0	6	2	4	6	8	0	6	8
10$_x$	0	0	0	4	0	2	8	6	0	6	4	10	8	4	8	4
11$_x$	6	10	10	4	4	2	0	4	4	0	2	8	4	2	2	2
12$_x$	0	0	8	8	2	8	2	8	6	4	2	8	0	0	8	0
13$_x$	4	4	2	2	8	6	0	2	2	2	0	4	6	8	14	0
14$_x$	0	8	6	2	8	8	2	6	4	2	0	2	8	6	0	2
15$_x$	4	4	8	2	4	0	4	10	8	2	4	4	4	2	0	4
16$_x$	0	6	10	2	2	2	2	4	10	8	2	2	0	4	10	0
17$_x$	8	2	4	2	6	4	0	6	4	4	2	2	0	4	8	8
18$_x$	0	**16**	2	2	6	0	6	0	6	2	8	0	6	0	2	8
19$_x$	0	8	0	2	4	4	10	4	8	0	6	4	2	6	2	4
1A$_x$	0	2	4	8	12	4	0	6	4	4	0	2	0	6	4	8
1B$_x$	0	6	2	6	4	2	4	4	6	4	8	4	2	0	10	2
1C$_x$	0	8	4	4	2	6	6	6	6	4	6	8	0	2	0	2
1D$_x$	4	4	4	0	0	2	4	2	4	2	2	4	10	10	8	4
1E$_x$	0	0	2	2	12	6	2	0	12	2	2	4	2	6	8	4
1F$_x$	2	2	10	14	2	4	2	4	4	6	0	2	4	8	0	0
20$_x$	0	0	0	14	0	8	4	2	0	4	2	8	2	6	0	14
21$_x$	4	2	6	2	12	2	4	0	6	4	10	2	4	2	2	2
22$_x$	10	6	0	2	4	4	10	0	4	0	12	2	8	0	0	2
23$_x$	0	6	2	2	2	4	6	10	0	4	8	2	2	6	0	10
24$_x$	4	2	0	6	8	2	6	0	8	2	2	0	8	2	12	2
25$_x$	2	0	2	**16**	2	4	6	4	6	8	2	4	0	6	0	2
26$_x$	6	10	0	10	0	6	4	4	2	2	4	6	2	4	2	2
27$_x$	4	0	2	0	2	2	14	0	4	6	6	2	12	2	4	4
28$_x$	14	4	6	4	4	6	2	0	6	6	2	2	4	0	2	2
29$_x$	2	2	0	2	0	8	4	2	4	6	4	4	6	4	12	4
2A$_x$	2	4	0	0	0	2	8	12	0	8	2	4	8	4	4	6
2B$_x$	**16**	6	2	4	6	10	2	2	2	2	2	2	4	2	2	0
2C$_x$	2	6	6	8	2	2	0	6	0	8	4	2	2	6	8	2
2D$_x$	6	2	4	2	8	8	2	8	2	4	4	0	2	0	8	4
2E$_x$	2	4	8	0	2	2	2	4	0	2	8	4	14	6	0	6
2F$_x$	2	2	2	8	0	2	2	6	4	6	8	8	6	2	0	6
30$_x$	0	6	8	2	8	4	4	0	10	4	4	6	0	0	2	6
31$_x$	0	8	4	0	6	2	2	6	6	0	0	2	6	4	8	10
32$_x$	2	4	0	0	6	4	10	6	6	4	6	2	4	6	2	2
33$_x$	0	**16**	6	8	2	0	2	2	4	2	8	4	0	4	6	0
34$_x$	0	4	14	8	2	2	2	4	**16**	2	2	2	0	2	0	4
35$_x$	0	6	0	0	10	8	2	2	6	0	0	8	6	4	4	8
36$_x$	2	0	2	2	4	6	4	4	2	2	4	2	4	**16**	10	0
37$_x$	6	6	6	8	4	2	4	4	4	0	6	8	2	4	0	0
38$_x$	0	2	2	2	8	8	0	2	2	2	0	6	6	4	10	10
39$_x$	4	4	**16**	8	0	6	4	2	4	4	2	6	0	2	2	0
3A$_x$	**16**	6	4	0	2	0	2	6	0	4	8	10	0	0	4	2
3B$_x$	2	0	0	2	0	4	4	4	2	6	2	6	6	12	12	2

3C$_x$	0	0	8	0	12	8	2	6	6	4	0	2	2	4	6	4
3D$_x$	2	4	12	2	2	2	0	4	6	10	2	6	4	2	0	6
3E$_x$	4	6	6	6	2	0	4	8	2	10	4	6	0	4	2	0
3F$_x$	14	0	0	0	8	0	6	8	4	2	0	0	4	8	4	6

Table 4.7: DDT for S_7 of DES

	0$_x$	**1**$_x$	**2**$_x$	**3**$_x$	**4**$_x$	**5**$_x$	**6**$_x$	**7**$_x$	**8**$_x$	**9**$_x$	**A**$_x$	**B**$_x$	**C**$_x$	**D**$_x$	**E**$_x$	**F**$_x$
0$_x$	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1$_x$	0	0	0	6	0	16	10	0	0	0	6	0	14	6	2	4
2$_x$	0	0	0	8	0	10	4	2	0	10	2	4	8	8	6	2
3$_x$	6	0	2	8	2	6	4	0	6	6	6	2	2	0	8	6
4$_x$	0	0	0	2	0	4	6	12	0	6	8	4	10	4	8	0
5$_x$	4	10	6	0	0	2	6	0	4	10	4	6	8	2	0	2
6$_x$	0	0	10	4	6	4	4	8	2	6	4	2	4	2	2	6
7$_x$	6	2	8	2	8	10	6	6	4	2	0	4	0	0	0	6
8$_x$	0	0	0	4	0	6	4	2	0	8	6	10	8	2	2	12
9$_x$	8	4	0	6	0	4	4	6	2	4	6	2	12	2	0	4
A$_x$	0	0	16	4	6	6	4	0	4	6	4	2	2	0	0	10
B$_x$	2	8	0	6	2	6	0	4	4	10	0	2	10	2	6	2
C$_x$	0	0	0	2	0	10	10	6	0	6	6	6	2	6	10	0
D$_x$	6	0	4	10	2	0	8	6	2	2	6	10	2	2	2	2
E$_x$	0	0	6	8	4	8	0	2	10	6	2	4	6	2	4	2
F$_x$	8	0	4	2	2	4	2	2	2	6	4	6	0	2	14	6
10$_x$	0	0	0	4	0	0	8	12	0	0	8	8	2	10	6	6
11$_x$	0	6	4	6	2	2	6	6	4	6	4	6	0	4	4	4
12$_x$	0	4	0	8	6	2	8	4	2	4	4	6	2	4	10	0
13$_x$	4	2	2	6	8	6	2	2	14	2	2	4	2	2	2	4
14$_x$	0	16	4	2	6	0	2	6	4	0	4	6	4	6	4	0
15$_x$	0	10	6	0	6	0	2	8	2	2	0	8	2	6	6	6
16$_x$	0	12	6	4	6	0	0	0	8	6	6	2	2	6	4	2
17$_x$	0	6	8	0	6	2	4	6	6	0	2	6	4	4	2	8
18$_x$	0	12	2	2	8	0	8	0	10	4	4	2	4	2	0	6
19$_x$	6	4	8	0	8	0	4	2	0	0	12	2	4	6	2	6
1A$_x$	0	4	6	2	8	8	0	4	8	0	0	0	6	2	0	16
1B$_x$	2	4	8	10	2	4	2	8	2	4	8	2	0	2	4	2
1C$_x$	0	12	6	4	6	4	2	2	6	0	4	4	2	10	2	0
1D$_x$	8	6	0	0	10	0	0	8	10	4	2	2	2	8	4	0
1E$_x$	0	4	8	6	8	2	4	4	10	2	2	4	2	0	6	2
1F$_x$	4	2	4	2	6	2	4	0	2	6	2	2	2	16	8	2
20$_x$	0	0	0	16	0	4	0	0	0	14	6	4	2	0	4	14
21$_x$	0	0	2	10	2	8	10	0	0	6	6	0	10	2	2	6
22$_x$	8	0	6	0	6	4	10	2	0	6	8	0	4	4	2	4
23$_x$	4	8	0	6	0	4	8	6	2	2	10	4	8	0	0	2
24$_x$	4	0	4	8	4	6	2	4	8	6	2	0	0	4	4	8
25$_x$	0	4	6	8	2	8	8	0	4	2	4	4	2	2	6	4
26$_x$	2	6	0	6	4	4	4	6	6	0	4	4	10	4	2	2
27$_x$	6	6	0	0	2	2	6	2	4	4	6	10	2	6	2	6
28$_x$	10	2	6	2	4	12	12	0	2	2	4	0	0	0	2	6
29$_x$	4	0	0	14	2	10	4	2	8	6	4	0	4	2	2	2
2A$_x$	8	8	0	2	0	2	4	0	2	6	8	14	2	8	0	0
2B$_x$	2	2	0	0	4	2	10	4	6	2	4	0	6	4	8	10
2C$_x$	2	6	6	2	4	6	2	0	2	6	4	0	6	4	10	4
2D$_x$	8	0	4	4	6	2	0	0	6	8	2	4	6	4	4	6
2E$_x$	6	2	2	4	2	2	6	12	4	0	4	2	8	8	0	2

2F$_x$	8	12	4	6	6	4	2	2	2	2	4	2	2	4	0	4
30$_x$	0	4	6	2	10	2	2	2	4	8	0	0	8	4	6	6
31$_x$	4	6	8	0	4	6	0	4	4	6	10	2	2	4	4	0
32$_x$	6	6	6	2	4	6	0	2	0	6	8	2	2	6	6	2
33$_x$	6	6	4	2	4	0	0	10	2	2	0	6	8	4	0	10
34$_x$	0	2	12	4	10	4	0	4	12	0	2	4	2	2	2	4
35$_x$	6	4	4	0	10	0	0	4	10	0	0	4	2	8	8	4
36$_x$	4	6	2	2	2	2	6	8	6	4	2	6	0	4	10	0
37$_x$	2	2	8	2	4	4	4	2	6	2	0	10	6	10	2	0
38$_x$	0	4	8	4	2	6	6	2	4	2	2	4	6	4	4	6
39$_x$	4	4	4	8	0	6	0	6	4	8	2	2	2	4	8	2
3A$_x$	8	8	0	4	2	0	10	4	0	0	0	4	8	6	8	2
3B$_x$	8	2	6	4	4	4	4	0	6	4	4	6	4	4	4	0
3C$_x$	0	6	6	6	6	0	0	8	8	2	4	8	4	2	4	0
3D$_x$	2	2	8	0	10	0	2	12	0	4	0	8	0	2	6	8
3E$_x$	6	4	0	0	4	4	0	10	6	2	6	12	2	4	0	4
3F$_x$	0	6	6	0	4	4	6	10	0	6	8	2	0	4	8	0

Table 4.8: DDT for S_8 of DES

4.5.2 Joint DDTs of two S-Boxes

The joint DDTs are unfortunately too large to be displayed in this report with good visualization quality. For this reason, we make them available as `.csv` files, together with the code used to obtain them, in our Git repository [66].

4.5.3 Joint DDTs of three S-Boxes

The complete DDTs are available at our Git repository [66]. Below we show, as an example, the relevant entries (the ones with output difference = 0) of the joint DDT of (S_1, S_2, S_3) when the combination of shared key bits is 0. The relevant entries are the ones for which the output difference is zero because they can help constructing characteristics with high probabilities throughout the rounds of DES (see Section 4.3.6).

Input difference = $328_x \to 84$
Input difference = $32c_x \to 28$
Input difference = $364_x \to 56$
Input difference = $36c_x \to 112$
Input difference = $3a4_x \to 40$
Input difference = $3ac_x \to 80$
Input difference = $3e4_x \to 16$
Input difference = $3ec_x \to 32$
Input difference = $728_x \to 12$
Input difference = $72c_x \to 4$
Input difference = $764_x \to 8$
Input difference = $76c_x \to 16$
Input difference = $7e4_x \to 8$
Input difference = $7ec_x \to 16$
Input difference = $b28_x \to 12$
Input difference = $b2c_x \to 4$
Input difference = $b64_x \to 8$
Input difference = $b6c_x \to 16$
Input difference = $be4_x \to 8$
Input difference = $bec_x \to 16$
Input difference = $f28_x \to 12$
Input difference = $f2c_x \to 4$

Input difference = $f64_x \rightarrow 8$
Input difference = $f6c_x \rightarrow 16$
Input difference = $fa4_x \rightarrow 8$
Input difference = $fac_x \rightarrow 16$

4.6 DDTs for AES

The same process (see Algorithm 4) can be applied to the AES S-Box and its inverse. Whilst the DES S-Boxes have differential uniformity equal to 16, both the AES S-Box and the inverse have differential uniformity equal to 4. The AES DDTs are also too large and are therefore available at our Git repository [66].

4.7 Computational complexity

4.7.1 Asymptotic complexity for a single n_s-bit input S-Box

As can be seen in Algorithm 4, for an n_s-bit input S-Box, it is necessary to compute all the possible X and, for each possible X, it is necessary to compute all the possible X^*. Therefore, since X and X^* range from 0 to $2^n - 1$, the computational complexity is $O(2^{(2n)})$.

4.7.2 S-Box total lookups

In terms of S-Box lookups, in total 2^{13} lookups are performed, since each DDT entry requires 2 S-Box lookups to be incremented. Since DES possesses 8 S-Boxes, extracting DDTs for all S-Boxes requires 8×2^{13} S-Box lookups. For AES, there is a single S-Box, $n_s = 8$ and 2 S-Box lookups are required to increment the DDT count, therefore 2^{17} lookups in total.

4.7.3 S-Box total lookups for joint DDTs

For the joint DDT of two neighbor S-Boxes of DES (see Algorithm 5), $2^{20} \times 4$ lookups are necessary, since S_1 is looked up twice and S_2 is also looked up twice. However, there is a joint DDT for each possible value of the shared key bits. For DES, 4 key bits are shared, therefore there are 16 joint DDTs, totalizing $16 \times 2^{20} \times 4 = 2^{26}$ S-Box lookups to obtain the DDTs of two neighbor S-Boxes for all possible shared key bits.

For the joint DDT of 3 neighboring S-Boxes of DES, $2^{28} \times 2^{8} \times 6$ lookups are necessary in total, since each DDT entry requires 6 lookups to be incremented and the joint input size is 14 bits.

4.7.4 Asymptotic complexity for joint DDTs

If sh_k key bits are shared, the asymptotic complexity to obtain all DDTs of t joint S-Boxes is $O(2^{2n_j} \times 2^{sh_k} \times 2t)$, with n_j being the size (in bits) of the joint input and sh_k being the amount of shared key bits, and S-Box lookups considered constant time.

4.7.5 Practical run time remarks

Obtaining single DDTs for DES took less than 1 minute in a personal computer. Extracting the joint DDTs for 2 neighbor S-Boxes of DES (for one of the possible combinations of shared key bits) took, in a personal computer, less than 5 minutes, and extracting all the DDTs for all possible shared key bits took less than 80 minutes. However, obtaining all the joint DDTs for 3 S-Boxes took approximately 72 hours. It is relevant to note that, as n_j and sh_k increase, the run time increases exponentially. Therefore, in this work, we restrict to obtaining joint DDTs for maximum 3 S-Boxes of DES. An attempt at extracting joint DDTs for larger tuples of neighbor S-Boxes is an interesting avenue for future work.

4.8 Conclusions

In this chapter, we explained what are Difference Distribution Tables and pointed out their relevant properties with respect to Differential Cryptanalysis attacks. We showed how to obtain them for an S-Box, and discussed the DDTs for the DES cipher and the AES cipher. We have also obtained their differential uniformities (16 for all eight DES S-Boxes and 4 for the AES S-Box and its inverse). Furthermore, we presented the concept of *joint difference distribution table*, which is relevant when activating a single S-Box is not possible for a DC attack (as is the case of DES). We leave a detailed discussion of the steps involved in a DC attack and how security against them is achieved by AES as a future work. Table 4.9 summarizes the linear uniformities and dimensions of all S-Boxes studied in this chapter.

S-Box	**DES** S_1	**DES** S_2	**DES** S_4	**DES** S_4	**DES** S_5	**DES** S_6	**DES** S_7	**DES** S_8	**AES** S	**AES** S **inv**
Differential uniformity	16	16	16	16	16	16	16	16	4	4
Input size	6	6	6	6	6	6	6	6	8	8
Output size	4	4	4	4	4	4	4	4	8	8

Table 4.9: Summary of AES and DES S-Boxes properties

Chapter 5

Obtaining Linear Approximation Tables of AES and DES

Linear Cryptanalysis (LC) was discovered by Matsui [54]. He describes how to apply it to the DES [58] cipher, showing it is capable of breaking 8-round DES with 2^{21} known-plaintexts and 16-round DES with 2^{47} known-plaintexts. LAT – Linear Approximation Tables — of each DES S-Box are a core component for the attack. In this chapter, we focus on discussing how to obtain LATs for DES and AES [59] and on exposing preliminary concepts related to LC.

We assume the reader is familiar with:

- The DES cipher structure, available in [58] and Appendix A
- The AES cipher structure, available in [59] and Appendix A
- The Differential Cryptanalysis, DDTs and joint DDTs overview, available in Chapter 4

5.1 Notation

- X_i: a random variable
- p_i: the probability of X_i being equal to 0 ($Pr[X_i = 0]$)
- ϵ_i: bias of a random variable X_i
- π_S: a hypothetical S-Box
- n: input size of an S-Box
- m: output size of an S-Box
- $X = (x_1, ..., x_n)$: n-tuple containing the inputs of π_S
- $Y = (y_1, ..., y_m)$: m-tuple containing the outputs of π_s
- K: cipher key
- γ, α, β: bit masks
- $X \cdot \gamma$: dot product of X and γ, i.e application of the bit mask γ to the variable X

5.2 Acronyms

- DES: Data Encryption Standard
- AES: Advanced Encryption Standard
- LC: Linear Cryptanalysis
- DC: Differential Cryptanalysis

- DDT: Difference Distribution Table
- LAT: Linear Approximation Table

5.3 Preliminaries

5.3.1 Linear Cryptanalysis overview

The goal of an LC attack is to find a probabilistic linear relationship between a subset of plaintext bits and a subset of state bits that immediately precede the substitutions performed in the last round of the cipher. In other words, *the attacker seeks a subset of bits whose XOR behaves non-randomly*, i.e, taking value 0 with probability bounded away from $\frac{1}{2}$.

Assuming the attacker has a large number of plaintext and ciphertext pairs, all encrypted with the same key K, the attack works by decrypting each ciphertext, using all possible candidate keys for the last round of the iterated cipher. For each of these candidate keys, if there is a probabilistic linear relationship as aforementioned, the candidate key's counter is incremented by 1. At the end of the process, the correct key should have a frequency count which allows us to distinguish it from the other candidates.

Let $X_1, X_2, ...$ be independent random values taking on values from $\{0,1\}$, and $p_1, p_2, ...$ be the probabilities of each of them being equal to zero, i.e, $Pr[X_1 = 0] = p_1, Pr[X_2 = 0] = p_2$ and so forth.

(Bias [54]) *The* bias *of a random variable taking on the values* 0 *and* 1 *is defined as* $\epsilon_i = p_i - \frac{1}{2}$. *Consequently,* $-1/2 \leq \epsilon_i \leq 1/2$, *since* $0 \leq p_i \leq 1$.

(Piling-Up Lemma [54]) *The bias of a random variable* $X_{i_1} \oplus X_{i_2} \oplus ... \oplus X_{i_k}$ *is* $\epsilon_{i_1,i_2,...,i_k} = 2^{k-1} \prod_{j=1}^{k} \epsilon_{i_j}$.

The Linear Cryptanalysis attack exploits sums of random variables, e.g S-boxes inputs and outputs, their biases, and, by extending the concept to the whole iterated cipher, extracts the key.

Considering an example S-box $\pi_S : \{0,1\}^n \to \{0,1\}^m$, if an input n-tuple $X = (x_1, ..., x_n)$ is chosen randomly, each coordinate x_i is a random variable X_i taking values from $\{0,1\}$. These variables are independent. A randomly chosen output tuple $Y = (y_1, ..., y_m)$, similarly, defines a random variable Y_j, also taking values from $\{0,1\}$. However, they are not necessarily independent on each other, nor they are independent of the X_i variables.

It is possible to compute the bias of a random variable of the form $X_{i_1} \oplus \cdots \oplus X_{i_z} \oplus Y_{j_1} \oplus \cdots \oplus Y_{j_l}$, from the Piling-up Lemma. When such bias is bounded away from $\frac{1}{2}$, a Linear Cryptanalysis attack can be mounted, if the amount of required plaintexts for the attack (which is inversely proportional to the bias) does not surpass the cipher's codebook, i.e if the final bias value is not too small.

5.3.2 Bit masks and Linear Approximation Tables

(Bit mask [54]) *A bit mask* γ *selects fixed specific bits of a variable* X. *For example,* $\gamma = [i]$ *refers to selecting only the i-th bit,* $\gamma = [i, j, k]$ *refers to selecting the i-th, j-th and k-th bits, and so forth, and XORing them. Formally, the result of the mask selection for a variable* X *can be obtained by means of the* dot product $X \cdot \gamma$.

For example, let $X = 1111_2$ and $\gamma = [1,2] = 0110_2$. Then

$$X \cdot \gamma = (0 \cdot 1) \oplus (1 \cdot 1) \oplus (1 \cdot 1) \oplus (0 \cdot 1) = 0_2,$$

because γ selects bits 1 and 2, considering that the starting bit, 0, is the least significant bit, and XORs them. Bits 0 and 3 are not selected. Bit masks are used by Matsui [54] to represent linear relationships (or approximations) between subsets of bits of the cipher. As another example, we show the usage of bit masks to represent linear relationship in the logical XOR of two variables.

Let $A = (..., a_1, a_0)$, $B = (..., b_1, b_0)$ and $C = A \oplus B$. Each i-th bit of C depends linearly (and individually) on the i-th bits of A and B, due to the bitwise XOR. In other words, $c_i = a_i \oplus b_i$. Equivalently, using bit mask notation, $C \cdot \gamma = A \cdot \gamma \oplus B \cdot \gamma$, with $\gamma = [i]$. Note that, when dealing with bit masks, we have switched the indexing of bits: we start from left to right, instead of from right to left. Also, the bit indices start at 0. We do this to follow the notation presented by Matsui [54].

(Linear Approximation Table of an S-Box) *For an n-bit input S-Box S, the LAT of S is the table* $\mathcal{L}$ *such that* $\mathcal{L}[\alpha][\beta] = \#\{x \in \mathbb{Z}_2^n : \alpha \cdot x = \beta \cdot S(x)\} - 2^{n-1}$, *where* α *and* β *are masks. In other words, a LAT entry* $[\alpha, \beta]$ *counts how far the parity of* $\alpha \cdot x \oplus \beta \cdot S(x)$ *deviates (or not) from* 2^{n-1}.

(Trivial and non-trivial linear relations) *A trivial linear relation possesses bias equal to zero. A non-trivial linear relation possesses non-zero bias. In Linear Cryptanalysis, we seek non-trivial linear relations.*

(Trivial and non-trivial masks) *We say that $\alpha = 0$ is a trivial mask, whilst $\alpha \neq 0$ is a non-trivial mask. In LC, we are interested in LAT entries with non-trivial input masks.*

Each entry $\mathcal{L}[\alpha][\beta]$ indicates whether there is a non-trivial linear relation between α input bits and β output bits of S.

(Linear Uniformity) *The linear uniformity of an S-Box is*

$$\gamma_{max} = max_{\alpha \neq 0}|LAT(\alpha, \beta)|,$$

where $|x|$ denotes the absolute value of x. It shows the largest entry in absolute value of the LAT of a given S-Box.

5.4 Computing a LAT

Let S be an n-bit input and m-bit output S-Box. In order to obtain its LAT $\mathcal{L}$, Algorithm 6 can be used. $2^n \times 2^m \times 2^n$ S-Box lookups are necessary, and $2 \times 2^n \times 2^m \times 2^n$ mask selection steps. Therefore, the computational complexity is $O(2^{2n+m})$ S-Box lookups, considering S-Box lookups and mask selection operations to be constant time.

For a DES S-Box, $n = 6$ and $m = 4$, resulting in 2^{16} run time complexity. For an AES S-Box, $n = m = 8$, resulting in 2^{24} complexity. In the case of AES and DES, since we are dealing with integer values, the mask selection can be performed with logical AND, XOR and bit shift operators.

Algorithm 6 Obtaining the LAT of an S-box

```
1: Initialize $\mathcal{L}$ with $2^{n-1}$ (the bias) in all entries
2: for $\alpha = 0, ..., 2^n - 1$ do
3:     for $\beta = 0, ..., 2^m - 1$ do
4:         for $x = 0, ..., 2^n - 1$ do
5:             $l \leftarrow \alpha \cdot x$                ▷ Select bits according to the input mask
6:             $r \leftarrow \beta \cdot S(x)$             ▷ Select bits according to the output mask
7:             if $l = r$ then
8:                 $\mathcal{L}[\alpha][\beta] \leftarrow \mathcal{L}[\alpha][\beta] + 1$
9:             end if
10:        end for
11:    end for
12: end for
13: return $\mathcal{L}$
```

It is also possible to compute LATs using the Walsh-Hadamard Transform of a Boolean Function, as explained in [13].

5.4.1 Practical run time remarks

In a personal computer with a 2.6 GHz Intel Core i7 6-Core processor, a LAT for a DES S-Box could be obtained in less than 2 seconds, and a LAT for one AES S-Box could be obtained in less than 2 minutes.

5.5 LATs for DES

Table 5.1 shows the linear uniformitiy of each DES S-Box. Note that S_5 presents the largest linear uniformity, which signals it contains the most effective linear relationship to be exploited in an LC attack. For this reason, Matsui [54] exploits S_5 to mount his attack. Tables 5.2 to 5.9 show the LATs for S-Boxes S_1 to S_8, respectively. They are also available as `.csv` files at our Git repository [66], together with the code used to obtain them.

S-Box	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
Linear uniformity	18	16	16	16	20	14	18	16

Table 5.1: Linear Uniformities of each DES S-Box

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	-2	-2	-4	-2	0	-4	6	2	0	0	6	4	-2	-6	4
3	0	-2	-2	-4	-2	0	-4	6	2	8	0	-2	4	6	-6	-4
4	0	2	-2	-4	-2	0	-4	-6	-2	4	8	2	0	-2	-6	12
5	0	-2	-2	0	-2	-4	-4	-2	2	-4	-4	2	4	-10	-2	-4
6	0	0	0	4	0	4	0	0	0	-4	4	4	0	0	-4	-8
7	0	-4	0	8	0	0	0	4	4	-4	-8	-4	4	0	0	0
8	0	4	-2	6	-6	-6	0	-4	-4	-4	2	-2	2	-2	0	0
9	0	0	6	-6	-2	-6	4	-4	0	-4	-2	6	2	-6	0	-4
10	0	-2	0	2	0	6	8	2	-2	0	-2	4	-2	0	-2	4
11	0	2	-8	-2	-4	-10	4	2	-6	8	2	4	-2	-4	-2	0
12	0	-2	0	6	0	2	0	2	2	0	6	-4	2	-4	6	0
13	0	6	0	6	4	-2	-4	-2	2	0	6	4	-2	8	-6	-4
14	0	0	-2	-2	2	2	0	0	4	4	6	-2	2	2	-4	4
15	0	0	-2	6	-2	-2	4	-4	-4	-4	-2	-2	-2	-2	0	0
16	0	2	2	0	-2	0	4	-6	0	6	2	-4	6	-4	-4	**-18**
17	0	2	-2	-4	2	-4	-4	10	-4	2	2	-4	-2	-4	0	-6
18	0	4	0	0	-4	4	0	4	-6	2	2	6	2	6	6	-10
19	0	4	-4	-4	0	0	-8	-12	-2	-2	-6	6	2	6	2	2
20	0	4	0	4	-8	-4	4	0	2	6	-2	2	6	2	-2	2
21	0	0	4	-4	-4	4	4	-4	10	2	2	2	-6	2	6	-2
22	0	6	2	0	2	-4	0	2	4	2	2	0	-2	0	0	2
23	0	2	6	-8	6	4	0	-2	-12	-2	-2	0	-6	0	0	-2
24	0	2	8	2	0	6	4	2	4	-2	4	6	0	-2	-4	2
25	0	-2	4	-6	0	-6	0	2	4	-6	8	6	0	2	0	-6
26	0	0	-6	2	-2	-2	4	4	-2	-2	0	0	-4	4	2	2
27	0	4	6	2	-10	2	-8	4	-2	-6	4	0	4	0	-2	2
28	0	-4	2	2	2	-6	0	-4	-2	-2	4	0	0	4	2	2
29	0	4	-2	-2	2	-6	-4	0	2	2	-4	0	-12	0	-6	-6
30	0	2	0	-2	4	-2	0	-2	0	6	-4	-2	0	-2	0	2
31	0	2	-4	2	-4	-2	4	2	4	-6	4	-2	-4	2	0	2
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	2	-2	0	2	0	0	6	-2	0	-4	6	4	10	10	0
35	0	2	-2	0	2	0	0	6	6	0	4	-10	-4	-6	2	0
36	0	2	-6	-8	2	4	4	2	2	0	0	2	0	6	-10	0
37	0	-2	2	4	2	0	-4	-2	-2	0	4	2	-4	6	-6	0
38	0	4	4	-4	8	-8	4	0	0	-8	0	-4	0	4	0	0
39	0	0	-4	-8	-8	4	-4	-4	4	-8	-4	-4	4	4	-4	0
40	0	4	-2	-2	-2	-2	-4	0	4	4	2	6	-2	-6	12	4
41	0	0	-2	-6	2	-2	-8	8	0	-4	-2	-2	6	-2	-4	0
42	0	2	0	-2	0	2	0	-2	2	-8	-6	-4	2	0	2	-4
43	0	-10	0	2	-4	2	4	6	-2	0	-10	4	2	-4	-6	0
44	0	6	-4	2	8	2	4	6	-2	-4	-2	12	-2	-8	-2	0
45	0	-2	-4	2	-4	-2	0	2	-2	-4	-2	4	-6	4	2	-4
46	0	-4	2	6	6	-6	-8	4	-4	0	2	6	6	2	4	0
47	0	-4	2	-2	2	-10	12	0	-4	0	2	-2	10	6	0	4
48	0	-2	-2	0	-2	4	0	2	0	2	6	4	6	0	0	-2

49	0	-2	2	4	2	0	0	-6	-4	-2	-2	-4	-2	0	-4	2
50	0	-4	-4	-4	0	0	0	4	-2	-2	-6	-2	-6	6	2	2
51	0	-4	0	0	4	-4	0	-4	-6	2	2	-2	2	-2	-2	-2
52	0	8	-8	8	4	4	0	0	-2	-2	2	-6	6	6	-2	-2
53	0	4	-4	0	-8	-4	0	-4	-2	2	-2	2	2	-2	-2	2
54	0	6	2	-8	2	-4	8	2	4	-6	2	0	6	0	0	2
55	0	2	-10	0	6	4	8	-2	4	6	-2	0	2	0	0	-2
56	0	-10	4	2	-4	-2	4	-2	4	2	0	6	-4	6	-4	-2
57	0	2	0	-6	-4	2	0	-2	-4	6	-4	-2	4	2	8	-2
58	0	0	6	-2	6	6	0	0	2	2	0	0	8	0	2	2
59	0	4	2	-2	-2	10	4	0	-14	-2	4	0	0	-4	-2	2
60	0	0	10	-2	-6	-2	0	8	-6	6	0	-8	-4	4	-2	2
61	0	8	-2	2	-6	-2	4	4	-2	-6	0	0	0	0	-2	2
62	0	2	0	-2	-8	2	4	2	0	-2	4	-2	-4	2	4	-2
63	0	-14	-12	-6	0	2	0	-2	-4	-6	12	-2	0	-2	4	-2

Table 5.2: LAT for S_1 of DES

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	4	0	0	-4	0	0	0	-4	0	0	0	0	4
3	0	0	4	-8	0	8	0	-4	0	0	-8	-4	0	8	-4	8
4	0	-2	2	4	2	0	4	6	0	6	-2	0	2	0	0	10
5	0	2	2	0	2	-4	4	-6	0	-6	6	4	-6	4	0	-2
6	0	-2	-2	-4	-2	-4	0	-2	-4	2	2	0	2	0	4	10
7	0	2	2	-4	-2	0	4	-2	-4	6	6	0	-6	-4	0	2
8	0	0	2	2	-2	2	4	0	-2	-6	-4	0	0	0	10	-6
9	0	0	-2	-2	-2	2	8	4	-2	2	0	-4	0	8	-10	-2
10	0	-4	2	2	2	2	-4	-8	2	2	4	0	0	4	-6	2
11	0	4	2	10	2	2	4	0	2	2	4	0	0	-4	2	2
12	0	-2	4	-2	0	2	0	6	-2	0	2	0	2	0	-6	-4
13	0	-6	0	-2	0	6	4	-10	6	-4	6	0	2	-4	-2	4
14	0	-6	0	2	0	-2	-8	6	-2	4	2	0	2	4	-2	0
15	0	-2	0	-2	0	2	0	10	6	8	2	4	2	0	-2	4
16	0	0	-4	4	0	0	0	0	0	0	4	4	4	-12	-4	-12
17	0	0	0	0	0	-8	4	4	0	0	8	0	-4	4	8	0
18	0	0	0	4	0	8	0	4	0	0	-4	8	-4	-12	0	12
19	0	0	-8	4	0	8	8	4	0	0	-4	0	4	-4	8	-4
20	0	-2	-6	-4	-2	4	-4	-2	-4	2	2	-4	6	-4	0	2
21	0	10	-2	-4	-2	0	0	-2	4	6	6	-4	-2	0	4	2
22	0	-2	-6	0	2	0	4	2	8	-2	2	0	6	4	0	-2
23	0	10	2	4	2	4	-4	-2	0	2	2	-4	-2	0	0	2
24	0	-4	2	-2	2	2	4	4	-2	-2	4	4	0	4	-2	2
25	0	4	-6	6	2	2	4	-4	-2	-2	4	-4	8	4	-2	2
26	0	8	-2	2	-2	2	0	0	2	6	0	0	0	0	2	-2
27	0	-8	10	6	-2	2	4	-4	2	-2	-4	-4	-8	0	-2	-6
28	0	2	8	-2	0	-2	0	2	2	0	-6	4	10	-4	2	0
29	0	-2	0	2	0	-6	0	-2	2	4	2	0	10	0	2	4
30	0	-2	0	6	0	2	4	-2	2	4	-2	0	2	0	2	0
31	0	2	-4	6	0	-2	-8	-2	-14	0	2	0	2	4	-2	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	4	0	-8	4	0	0	0	-4	**-16**	0	-8	-8	4
35	0	0	-4	0	0	0	0	4	0	0	0	4	0	0	-4	0

36	0	-2	-2	0	-2	-4	4	-10	0	6	-6	-4	6	4	8	2
37	0	-6	-2	4	-2	0	4	2	0	2	2	-8	-2	0	8	-2
38	0	6	-6	0	-14	0	0	-2	-4	-6	-2	4	-2	-4	-4	2
39	0	2	6	0	2	-4	-4	-2	-4	6	-6	4	6	0	0	-6
40	0	-4	2	-2	2	2	0	-8	2	2	0	8	8	-4	-6	-2
41	0	-4	-2	-6	2	2	4	-4	-6	2	-4	-4	0	-4	-2	-6
42	0	8	2	-2	6	-6	0	0	-2	2	0	0	0	0	-6	-2
43	0	0	-6	-2	6	10	0	0	-10	10	0	0	-8	0	2	-2
44	0	-6	0	6	0	-2	-4	-2	2	-8	2	4	-2	0	2	8
45	0	-2	-4	-2	0	-6	0	-10	2	4	-2	4	-10	-4	-2	0
46	0	-2	-4	2	8	2	4	-2	-6	-4	-6	4	-2	4	-2	4
47	0	-6	4	-2	-8	-2	4	2	-6	0	10	0	6	0	-2	0
48	0	0	-4	4	-4	-4	4	4	0	0	-4	-4	0	0	-8	0
49	0	0	0	0	4	-4	0	0	0	0	0	-8	0	-8	-4	4
50	0	0	0	4	4	4	4	0	0	0	4	0	0	0	-4	0
51	0	0	0	-4	-4	-4	-4	0	0	0	-4	0	0	0	4	0
52	0	-10	-2	-8	6	4	-8	2	4	2	6	-8	-2	-4	4	-2
53	0	-6	2	0	-2	0	4	2	-4	-2	-6	0	-2	0	0	-2
54	0	-2	-2	4	-6	0	8	-2	0	6	6	4	-2	4	-4	2
55	0	2	-2	-8	-14	4	0	2	8	2	-2	0	-2	0	-4	-2
56	0	0	2	2	-6	-2	-4	0	2	-2	0	-4	-4	-4	2	2
57	0	8	10	-6	2	6	4	0	-6	-10	8	-4	4	-4	2	2
58	0	-4	-2	6	-2	-2	-8	4	-2	-2	4	0	-4	0	-2	-2
59	0	-4	2	2	-10	6	-4	0	-10	-2	0	-4	4	0	2	2
60	0	-2	-4	-2	4	-2	0	-2	-2	0	2	0	2	-8	-2	0
61	0	2	4	10	-4	10	-8	-6	6	4	2	-4	2	4	-2	-4
62	0	2	-12	-2	4	2	-4	2	6	-12	-2	-4	2	4	-2	0
63	0	-2	-8	-2	-4	-2	0	-6	-2	0	2	4	2	0	2	0

Table 5.3: LAT for S_2 of DES

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	2	-2	-4	0	-2	-6	4	0	6	-2	-8	4	-2	-2	-4
3	0	-2	-2	0	4	-2	-2	-4	-4	-2	2	0	-4	2	-2	0
4	0	-4	0	0	2	-2	-6	2	0	4	4	-4	-6	-10	6	-2
5	0	0	0	-4	2	2	2	6	0	0	4	0	2	-6	6	2
6	0	-2	2	0	2	-4	0	2	0	-6	-2	0	-2	4	8	-2
7	0	-2	2	0	-2	8	4	6	-4	-2	2	-4	-10	4	0	-2
8	0	0	6	-6	0	0	2	-2	-2	2	0	8	6	2	4	-4
9	0	0	2	-2	0	-8	-2	-6	-2	2	-4	-4	-2	2	8	0
10	0	-2	8	-2	0	2	-8	-6	2	0	6	4	-2	0	2	-4
11	0	2	-4	6	4	2	0	-2	-2	0	-2	0	-2	-4	-2	4
12	0	0	-2	-2	2	10	4	-4	2	-2	0	-4	4	-8	6	-6
13	0	-4	2	-2	2	-2	0	12	2	2	4	4	4	4	2	2
14	0	-2	-4	-2	2	-4	-2	-4	-2	4	2	4	4	-2	-8	-2
15	0	-2	-8	2	-2	0	-2	-4	-6	-8	2	4	4	-2	4	2
16	0	0	2	-2	2	2	8	4	2	6	-4	12	-4	-8	2	10
17	0	0	-2	2	-2	-2	0	4	-2	2	4	-4	4	0	-10	6
18	0	-2	4	2	-2	0	2	4	-2	-4	-6	0	0	-14	4	-2
19	0	2	0	2	-2	4	-2	-12	-2	0	-2	8	-8	-2	0	-2
20	0	4	-6	-2	-4	0	2	-2	2	-6	-8	0	6	-2	0	0
21	0	8	-2	6	0	8	2	2	-2	2	-8	-4	-2	2	4	0
22	0	2	0	-10	0	6	0	2	-2	0	2	0	2	0	-2	0

23	0	10	4	-6	0	-2	-4	-2	-2	0	-2	4	2	0	2	-4
24	0	0	4	-4	2	2	-2	-2	-4	4	4	4	-2	6	6	-2
25	0	8	-4	-4	-2	-2	2	2	0	0	0	8	-10	-2	-10	-2
26	0	-6	2	0	-2	4	-4	-2	-4	2	2	4	6	0	0	-2
27	0	-2	2	-4	-2	0	-4	2	4	-2	2	0	-2	4	0	2
28	0	0	-4	0	-4	4	-8	4	0	-8	4	0	-4	4	8	4
29	0	4	4	4	0	4	-4	-4	4	8	0	0	4	0	0	8
30	0	-6	6	0	0	-2	2	0	-8	-2	-2	-4	4	-2	-2	0
31	0	-6	-10	0	0	6	-6	0	0	-2	-2	4	4	-2	-2	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	-2	-2	0	0	2	2	-8	0	2	-2	**-4**	4	2	6	**16**
35	0	2	-2	-4	4	-6	6	-8	-4	2	2	-4	-4	-2	6	-4
36	0	4	0	8	2	-2	-6	2	4	0	-8	8	6	2	2	-6
37	0	0	0	-4	2	-6	2	-2	4	4	8	4	-2	-2	2	6
38	0	2	-6	-12	2	0	0	-2	-4	-6	2	0	2	-4	-4	-2
39	0	2	-6	4	-2	-4	4	2	8	-2	6	-4	-6	-4	4	-2
40	0	0	2	-2	0	8	6	2	-2	-6	-4	4	-2	10	0	0
41	0	0	-2	2	0	0	2	-2	-2	-6	8	8	6	-6	4	4
42	0	2	4	-2	8	-2	-4	2	2	-4	2	-4	-2	-4	-2	4
43	0	-2	8	-2	-4	-10	4	-2	-2	-12	-6	0	-2	0	-6	4
44	0	0	2	-6	2	10	0	0	-2	2	0	-4	0	-4	-2	2
45	0	4	6	2	2	6	-4	-8	-2	-2	4	-4	0	0	-6	2
46	0	2	-8	-2	-6	0	2	-4	2	4	2	0	0	-2	0	-6
47	0	2	4	2	6	4	2	-4	14	-8	2	0	0	-2	-4	-2
48	0	0	-6	-2	-2	-2	-4	0	2	-2	4	4	0	4	-2	6
49	0	0	-2	-6	2	2	4	0	-2	-6	4	-4	0	4	2	2
50	0	2	-4	-2	2	0	6	4	-2	8	2	4	-4	2	0	-2
51	0	-2	0	-2	-6	4	2	-4	-2	4	-2	-4	-4	-2	-4	6
52	0	-4	2	6	0	4	-2	2	6	-2	4	4	-2	-2	0	0
53	0	8	-2	-2	12	-4	-2	-2	2	-2	-4	0	-2	2	4	8
54	0	-2	0	2	-4	-2	-12	2	-6	0	-2	0	-6	-4	-2	4
55	0	6	-4	-2	-12	-2	0	-2	10	0	2	-4	2	4	2	0
56	0	0	0	8	6	-2	6	-2	-4	4	0	0	2	2	-2	-2
57	0	8	0	0	10	2	-6	2	0	0	4	-4	2	2	-2	-2
58	0	6	-2	0	2	4	-4	2	-4	-2	-2	4	-6	0	0	2
59	0	-14	-10	-4	10	0	-4	-2	4	2	-10	0	-6	4	0	-2
60	0	0	0	4	-8	0	0	-4	-4	4	4	0	-4	4	4	0
61	0	-4	0	8	4	0	4	-4	0	-4	8	0	-4	0	-4	-4
62	0	6	2	0	-4	-2	-6	4	-4	-2	-2	0	-4	-6	2	0
63	0	6	-6	8	4	-2	2	4	-12	-2	6	0	4	2	2	0

Table 5.4: LAT for S_3 of DES

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	4	0	4	0	0	0	0	0	0	4	0	4	0	0
3	0	-4	0	8	0	0	8	-4	-4	-8	0	0	-8	0	-4	0
4	0	-2	-2	0	-2	0	0	-10	2	0	0	-6	0	-6	10	0
5	0	2	-2	-4	2	0	-4	6	2	4	0	-10	4	10	6	0
6	0	-2	2	0	-2	-4	-4	2	-2	-4	4	2	0	-2	2	8
7	0	-2	-2	4	-2	12	0	-2	2	0	12	2	4	2	2	0
8	0	-4	0	0	0	0	0	4	4	0	0	0	0	0	-4	0
9	0	0	-4	-8	4	0	8	0	0	-8	0	4	8	-4	0	0

10	0	-4	-4	0	4	0	0	-4	-4	0	0	4	0	-4	-4	0
11	0	4	-4	0	-4	0	0	-4	-4	0	0	-4	0	-4	4	0
12	0	-2	2	0	-2	4	4	-6	-2	4	-4	10	0	-10	-6	-8
13	0	-2	-2	-4	-2	4	0	-10	2	0	4	-6	-4	-6	10	0
14	0	-2	-2	0	-2	0	0	-2	2	0	0	2	0	2	2	0
15	0	2	-2	4	2	0	4	-2	2	-4	0	-2	-4	2	-2	0
16	0	-2	-2	0	2	4	4	10	-2	4	-4	6	0	-6	10	8
17	0	2	-2	-4	-2	-4	0	-6	-2	0	-4	10	-4	10	6	0
18	0	2	-2	0	-2	0	0	-6	-2	0	0	10	0	10	6	0
19	0	2	2	-4	-2	0	4	-10	2	-4	0	-6	4	6	-10	0
20	0	4	-4	0	-4	0	0	4	-4	0	0	-4	0	-4	-4	0
21	0	4	4	8	-4	0	0	4	4	0	0	4	-8	-4	4	0
22	0	8	-4	0	4	8	-8	0	8	-8	-8	4	0	-4	0	0
23	0	4	8	8	8	0	0	-4	-4	0	0	0	8	0	4	0
24	0	2	-2	0	-2	0	0	2	-2	0	0	2	0	2	-2	0
25	0	2	2	4	-2	0	-4	-2	2	4	0	2	-4	-2	-2	0
26	0	-2	-2	0	2	-4	-4	2	-2	-4	4	-2	0	2	2	-8
27	0	2	-2	-12	-2	4	0	2	-2	0	4	2	-12	2	-2	0
28	0	-4	0	0	0	8	-8	4	-4	-8	-8	0	0	0	4	0
29	0	0	-4	-8	-4	0	0	0	0	0	0	4	-8	4	0	0
30	0	-8	0	0	0	0	0	0	8	0	0	0	0	0	0	0
31	0	0	-8	8	8	0	0	0	0	0	0	0	-8	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	4	0	0	0	-8	8	4	4	8	8	0	0	0	4	**-16**
35	0	0	4	0	4	0	0	0	0	0	0	4	0	4	0	0
36	0	2	2	0	-2	4	4	6	2	4	-4	2	0	-2	6	-8
37	0	-2	2	4	2	-4	0	-2	2	0	-4	6	4	6	2	0
38	0	-10	2	0	2	0	0	-2	10	0	0	-2	0	-2	2	0
39	0	-2	-10	4	10	0	4	2	-2	-4	0	-2	-4	2	2	0
40	0	0	4	0	-4	8	-8	0	0	-8	-8	-4	0	4	0	**-16**
41	0	-4	0	0	0	0	0	4	4	0	0	0	0	0	-4	0
42	0	4	-4	0	-4	0	0	-4	-4	0	0	-4	0	-4	4	0
43	0	4	4	0	-4	0	**16**	4	4	**-16**	0	-4	0	4	4	0
44	0	-2	2	0	2	0	0	-2	2	0	0	6	0	6	2	0
45	0	-2	-2	-4	2	0	-4	-6	-2	4	0	-2	4	2	-6	0
46	0	2	10	0	-10	-4	-4	-2	2	-4	4	2	0	-2	-2	8
47	0	-10	2	-4	2	-12	0	-2	10	0	-12	-2	-4	-2	2	0
48	0	-2	-2	0	-2	0	0	6	2	0	0	2	0	2	-6	0
49	0	2	-2	-4	2	0	4	-2	2	-4	0	6	4	-6	-2	0
50	0	-2	2	0	-2	4	4	2	-2	4	-4	-6	0	6	2	8
51	0	-2	-2	4	-2	4	0	6	2	0	4	2	4	2	-6	0
52	0	0	8	0	-8	8	8	0	0	8	-8	0	0	0	0	0
53	0	-8	0	0	0	8	0	0	8	0	8	0	0	0	0	0
54	0	0	-4	0	-4	0	0	0	0	0	0	4	0	4	0	0
55	0	4	0	8	0	0	0	-4	4	0	0	0	-8	0	-4	0
56	0	-10	2	0	-2	-4	-4	2	-10	-4	4	2	0	-2	2	-8
57	0	-2	-10	12	-10	-4	0	-2	2	0	-4	2	12	2	2	0
58	0	-2	-10	0	-10	0	0	-2	2	0	0	2	0	2	2	0
59	0	10	-2	4	2	0	-4	-2	10	4	0	-2	-4	2	-2	0
60	0	4	8	0	8	0	0	-4	-4	0	0	0	0	0	4	0
61	0	-8	4	8	-4	0	0	0	-8	0	0	-4	-8	4	0	0
62	0	-4	-4	0	4	8	8	-4	-4	8	-8	-4	0	4	-4	0
63	0	4	-4	0	-4	-8	0	4	-4	0	-8	-4	0	-4	-4	0

Table 5.5: LAT for S_4 of DES

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	4	-2	2	-2	2	-4	0	4	0	2	-2	2	-2	0	-4
3	0	0	-2	6	-2	-2	4	-4	0	0	-2	6	-2	-2	4	-4
4	0	2	-2	0	0	2	-2	0	0	2	2	4	-4	-2	-2	0
5	0	2	2	-4	0	10	-6	-4	0	2	-10	0	4	-2	2	4
6	0	-2	-4	-6	-2	-4	2	0	0	-2	0	-2	-6	-8	2	0
7	0	2	0	2	-2	8	6	0	-4	6	0	-6	-2	0	-6	-4
8	0	0	2	6	0	0	-2	-6	-2	2	4	-12	2	6	-4	4
9	0	-4	6	-2	0	-4	-6	-6	6	-2	0	-4	2	-6	-8	-4
10	0	4	0	0	-2	-6	2	2	2	2	-2	2	4	-4	-4	0
11	0	4	4	4	6	2	-2	-2	-2	-2	-2	2	0	-8	-4	0
12	0	2	0	-2	0	2	4	10	-2	4	-2	-8	-2	4	-6	-4
13	0	6	0	2	0	-2	4	-10	-2	0	-2	4	-2	8	-6	0
14	0	-2	-2	0	-2	4	0	2	-2	0	4	2	-4	6	-2	-4
15	0	-2	-2	8	6	4	0	2	2	4	8	-2	8	-6	2	0
16	0	2	-2	0	0	-2	-6	-8	0	-2	-2	-4	0	2	10	**-20**
17	0	2	-2	0	4	2	-2	-4	4	2	2	0	-8	-6	2	4
18	0	-2	0	-2	2	-4	-2	-8	4	6	4	6	-2	4	-6	0
19	0	-6	0	2	-2	4	2	0	4	-6	4	2	-6	4	-2	0
20	0	4	-4	0	0	0	0	0	-4	-4	4	4	0	4	-4	0
21	0	4	0	-4	-4	4	-8	-8	0	0	-4	4	8	4	0	4
22	0	0	6	6	2	-2	4	0	4	0	6	2	2	2	0	0
23	0	4	-6	-2	6	-2	-4	4	4	-4	-6	2	-2	2	0	4
24	0	6	0	2	4	-10	-4	2	2	0	-2	0	2	4	-2	-4
25	0	2	4	-6	0	-2	4	-2	6	8	6	4	10	0	2	-4
26	0	2	2	-8	-2	4	0	2	-2	0	4	2	0	-2	-2	0
27	0	2	6	-4	-6	0	0	2	6	8	0	-2	-4	-6	-2	0
28	0	0	-2	2	4	0	-6	2	-2	6	-4	0	2	-2	0	0
29	0	4	-2	6	-8	0	-2	2	10	-2	-8	-8	2	2	0	4
30	0	-4	-8	0	-2	-2	-2	2	-2	2	-2	6	4	4	4	0
31	0	-4	8	-8	2	-6	-6	-2	-2	2	-2	-2	-8	0	0	-4
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	-4	-2	2	-2	2	-4	8	-4	0	-6	6	2	-2	-16	-12
35	0	0	-2	-2	6	-2	-4	4	0	0	-2	-2	-2	6	4	-4
36	0	-2	6	4	0	6	-2	4	4	-6	-2	4	0	14	2	0
37	0	6	2	0	0	6	2	0	-4	-6	2	-8	0	-2	6	-4
38	0	2	4	-2	-2	0	2	-4	4	-2	-4	-2	6	0	-2	0
39	0	-10	0	-2	6	4	6	-4	0	6	-12	2	2	0	6	-4
40	0	4	-2	-2	0	4	-6	2	2	-6	4	0	6	-2	-4	0
41	0	0	2	6	0	0	6	2	2	-2	-8	0	-2	-6	0	0
42	0	0	-4	-8	6	6	6	-6	6	2	-2	-2	-8	4	-4	4
43	0	8	0	4	6	-2	-6	6	2	6	-2	6	-4	0	4	4
44	0	2	4	-6	0	-6	0	6	-2	-4	2	-4	-2	4	6	0
45	0	-2	-4	-2	0	-2	-8	2	-2	0	-6	-8	-2	0	-2	4
46	0	6	2	-4	6	4	4	-2	-10	-8	0	-2	4	-2	2	0
47	0	6	-6	-4	6	-4	4	-2	2	4	4	-6	0	2	-2	-4
48	0	2	-2	0	-4	-6	-2	-4	4	2	2	0	0	2	2	4
49	0	2	-2	0	0	-2	2	0	0	-2	-2	-4	0	2	2	4
50	0	6	0	-2	-2	8	2	4	0	10	0	2	-2	4	2	0
51	0	-6	0	10	2	0	-2	-4	0	6	0	-10	2	4	-2	0
52	0	0	-12	4	-4	0	4	-8	-4	0	-4	0	-4	-4	0	0
53	0	-8	0	0	8	-4	4	0	0	-4	-4	0	4	4	-4	4
54	0	4	-2	-6	-2	-2	8	0	4	-4	-2	-2	6	2	-4	0

55	0	-8	-6	-6	-6	6	0	4	12	0	2	-2	2	2	4	-4
56	0	2	4	-6	0	-2	4	-2	-6	4	-6	0	6	4	-2	0
57	0	-2	8	2	-4	6	-4	-6	-2	-4	2	4	-2	0	2	0
58	0	6	-10	0	2	4	0	-2	6	-4	0	2	4	-2	-2	-4
59	0	-2	-6	-4	-10	0	-8	-2	-10	4	4	-2	0	2	-2	4
60	0	-8	-6	-2	0	-4	2	2	-6	2	4	0	10	-2	4	4
61	0	4	2	2	4	4	-2	2	-2	10	0	0	2	2	4	0
62	0	-4	4	-4	2	2	-2	2	2	-2	-2	-2	4	-4	0	4
63	0	-4	-4	-4	14	6	-6	-2	2	-2	6	-2	0	0	-4	0

Table 5.6: LAT for S_5 of DES

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	2	-6	2	-6	2	6	-2	2	-4	0	0	4
3	0	0	0	0	-2	-2	-2	-2	-2	-6	2	-2	4	8	0	4
4	0	0	2	-2	2	-2	8	0	2	2	0	4	4	8	6	-2
5	0	0	6	-6	2	6	4	-4	-2	-2	-8	4	8	4	-2	-10
6	0	4	-2	-2	0	0	2	-2	0	0	-2	2	-8	4	2	2
7	0	-4	2	2	4	-4	2	-2	0	0	2	-2	4	0	-6	2
8	0	2	-2	0	-4	-2	-2	-8	2	0	8	6	6	-4	0	-2
9	0	-2	6	4	0	6	2	0	-2	0	4	6	-2	0	0	10
10	0	2	2	4	2	4	-8	2	0	2	-2	0	-6	-4	4	-2
11	0	-2	-6	8	2	0	8	-2	0	-2	-10	4	2	0	-4	2
12	0	-2	0	2	2	-4	2	8	-4	-2	0	-2	-2	-4	2	4
13	0	2	4	2	-2	-4	2	4	4	2	-4	-2	2	12	10	0
14	0	2	0	6	-4	2	-4	-2	-2	0	-2	-4	-6	8	2	4
15	0	-2	4	-2	-4	6	0	-2	2	0	-2	0	-2	0	2	0
16	0	2	0	-2	0	2	-4	**-14**	-4	-2	-4	-6	0	2	-12	10
17	0	2	0	-2	4	-2	8	6	-4	-2	-4	-6	4	-2	0	-2
18	0	2	0	-2	-2	0	2	-8	-6	8	-2	8	-4	2	4	-2
19	0	2	0	-2	-2	0	-6	0	-2	4	-6	-4	0	-2	8	10
20	0	-2	2	0	2	-4	-4	-2	-2	-4	-4	2	-4	-2	-6	-4
21	0	6	-2	-4	6	-8	-4	-2	2	8	4	-6	-4	6	-2	0
22	0	-6	6	0	4	2	-6	0	0	-2	-2	4	0	2	-2	0
23	0	-6	2	4	-4	2	-2	4	0	6	-6	0	0	2	-6	4
24	0	0	-2	2	0	0	-2	2	-2	2	4	-4	2	-2	0	0
25	0	-12	-2	-10	-8	-4	6	-2	-6	-6	8	-4	-2	6	4	0
26	0	8	2	-2	2	2	-4	0	0	-8	6	2	-2	-2	-4	0
27	0	-4	2	2	6	2	0	0	-8	4	-2	-2	2	-2	0	0
28	0	0	0	0	-2	-6	2	-2	0	-4	4	8	2	-6	2	2
29	0	4	4	0	-2	-10	-2	-2	0	-8	-8	0	2	-2	-2	-6
30	0	4	8	-4	-4	-4	0	0	-2	2	-2	-6	6	-2	2	2
31	0	0	-4	4	0	-4	0	4	2	2	-2	-2	-2	2	-2	2
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	4	0	-4	-2	2	-2	10	2	-6	-2	**14**	0	0	4	12
35	0	4	0	-4	2	-2	2	6	-2	-2	2	-6	0	0	-4	4
36	0	0	-2	2	-2	2	0	-8	-2	-2	0	-4	4	0	10	2
37	0	0	2	-2	-2	-6	-4	4	2	2	0	4	0	4	-6	2
38	0	8	2	6	0	4	6	6	-4	0	6	-2	4	4	-6	-2
39	0	0	6	10	-4	-8	-2	-2	-12	8	2	2	0	0	2	-2
40	0	2	2	-4	0	2	-2	0	-2	4	0	-2	6	-12	4	2
41	0	-2	2	8	4	-6	-6	0	2	-4	-4	-2	6	0	4	-2

42	0	6	-2	4	2	0	-4	2	-4	-6	-2	-4	-2	4	4	2
43	0	2	-2	0	10	4	-4	-2	4	-2	6	0	6	8	4	-2
44	0	-2	8	-6	2	4	2	0	4	-2	0	-10	-2	-4	2	4
45	0	2	4	2	-2	4	-6	4	-4	2	4	-2	2	-4	2	-8
46	0	6	-8	-6	0	-6	0	-2	6	4	-2	0	6	0	-2	4
47	0	2	4	10	-8	6	4	-2	10	4	6	-4	2	0	-2	0
48	0	2	0	-2	0	2	4	-6	0	2	0	-2	-4	-2	8	-2
49	0	2	0	-2	4	-2	0	-2	0	2	0	-2	0	-6	4	2
50	0	-2	0	2	2	-8	-2	0	-2	-8	2	0	-12	-2	4	-6
51	0	-2	0	2	-6	0	-2	0	2	4	-2	4	0	2	0	-2
52	0	-2	-2	4	-2	0	-4	-2	-2	-4	0	-2	8	2	2	4
53	0	-10	10	0	2	-4	-4	-2	10	0	0	-2	0	2	-2	0
54	0	-10	-6	0	-4	6	-2	0	0	-6	-6	-4	0	-2	2	0
55	0	6	6	4	-4	-2	10	-4	-8	-6	-2	0	0	-2	-2	4
56	0	0	-6	6	4	4	-2	-6	-2	-6	8	0	-2	2	0	0
57	0	4	2	2	-4	0	-2	-2	2	-6	-4	0	2	2	4	0
58	0	4	6	-2	-6	-2	-8	0	0	-4	2	2	6	2	0	0
59	0	8	-2	-6	-10	6	-4	0	0	0	-6	-2	-6	2	-4	-8
60	0	0	0	0	-2	2	2	6	-4	0	0	-4	-2	6	-2	-2
61	0	4	-4	8	-2	-2	6	-2	12	-4	-4	-4	-2	-6	2	-2
62	0	0	-8	0	8	4	-4	0	-6	2	-6	2	6	2	2	-2
63	0	-4	-12	0	-12	-4	-4	4	-2	2	2	-2	6	-2	-2	-2

Table 5.7: LAT for S_6 of DES

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	2	-6	4	-4	2	2	2	2	0	0	-2	-2	0	0
3	0	0	-2	6	0	0	2	-6	2	2	-4	-4	-6	2	0	8
4	0	0	2	2	0	-4	-6	-2	4	4	10	2	4	0	-6	6
5	0	0	2	10	0	4	-6	-2	0	-8	6	-2	0	-4	6	10
6	0	4	0	-4	-8	0	4	-4	-2	-6	-2	2	6	-2	2	-6
7	0	-4	4	0	4	4	-4	4	2	-2	6	2	6	-2	-2	-2
8	0	-4	0	4	-2	-2	-2	6	-4	4	0	0	-2	2	2	-2
9	0	4	-4	0	2	-6	-2	-10	0	0	0	0	-2	-6	-2	-6
10	0	0	-2	-10	2	-2	-4	0	2	6	0	4	0	0	2	2
11	0	0	-2	-2	10	-2	-4	0	6	-6	4	0	4	-4	-2	-2
12	0	0	-2	-2	2	-6	0	0	0	-4	-2	2	-2	-6	4	0
13	0	0	2	2	-2	-2	0	-8	0	4	2	-2	2	-2	-4	-8
14	0	0	0	0	2	2	6	-2	-2	-6	-6	6	-4	-8	-4	0
15	0	0	0	0	2	2	-2	6	6	-6	2	6	-4	0	4	0
16	0	-2	2	4	0	-2	-2	0	-2	4	4	6	-2	-4	8	-14
17	0	2	-2	4	-4	-2	-2	4	-2	0	0	-2	10	4	-8	-2
18	0	-2	0	2	-4	2	-4	-10	0	-2	0	-14	4	-6	4	-2
19	0	2	0	6	4	6	4	-6	0	-6	0	-2	-4	6	-4	-6
20	0	2	8	6	0	-10	4	-2	6	8	-2	-4	-2	-4	2	4
21	0	-2	-4	-2	4	-2	4	-6	2	0	-2	0	-2	0	-2	-4
22	0	-2	2	-4	-8	2	2	0	0	-2	-2	0	0	-6	-2	4
23	0	2	2	-8	-8	-10	2	-4	-12	6	2	0	4	2	2	4
24	0	2	-2	-4	2	0	-4	6	2	8	0	-6	0	2	2	-8
25	0	-2	6	-8	2	-4	-4	-6	6	0	12	2	-4	2	-2	0
26	0	-2	0	2	-2	0	-2	4	0	2	-4	2	2	0	-2	0
27	0	2	4	2	2	0	6	0	4	2	4	-2	2	4	2	0
28	0	2	0	-2	-2	8	2	0	2	0	6	0	-4	2	4	-2

29	0	-2	8	2	-2	-4	2	4	2	-4	-2	4	-12	-2	-4	-6
30	0	2	6	-4	-2	0	4	2	8	-2	-2	0	2	0	0	2
31	0	-2	2	4	2	0	4	-2	0	2	2	0	6	0	0	-2
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	2	2	-4	4	-6	2	2	2	0	-8	-2	-2	-16	-8
35	0	0	-2	-2	0	0	2	2	2	2	-4	4	2	-6	-8	8
36	0	0	-2	6	0	4	-2	2	-4	4	-2	6	4	0	-2	2
37	0	0	-2	-2	0	-4	-2	2	0	0	2	-6	-8	4	2	-2
38	0	-4	-4	0	-8	0	-8	0	6	2	2	6	-2	-2	-2	-2
39	0	-12	0	4	-4	-4	8	0	2	-2	2	-2	-2	-2	-6	2
40	0	0	-4	4	-2	-6	-6	-2	4	-8	-4	8	6	14	-2	-2
41	0	0	0	0	2	-2	2	-2	0	4	-4	0	-2	6	-6	2
42	0	-4	2	-2	2	2	0	0	-6	2	4	4	0	-4	-2	2
43	0	4	10	6	2	2	0	-8	6	6	-8	8	4	0	2	-2
44	0	4	6	2	2	-2	0	12	0	0	-2	-2	6	-10	4	-4
45	0	-4	2	6	-2	-6	-8	4	0	0	-6	-6	-6	2	4	4
46	0	4	0	4	-6	-2	6	2	-2	-2	2	2	4	4	4	-4
47	0	12	-8	4	2	-2	-2	2	6	6	2	2	-4	-4	-4	4
48	0	2	2	0	4	6	2	0	-2	8	4	2	2	4	-4	2
49	0	6	-2	0	0	6	2	4	-2	4	0	-6	-2	-4	-4	-2
50	0	2	0	6	0	-6	0	-2	0	2	0	6	0	-6	0	-2
51	0	6	0	-6	0	6	0	-6	0	-2	0	2	0	-2	0	2
52	0	-2	4	-2	-4	6	4	2	-2	4	2	4	-6	4	2	0
53	0	-6	-8	6	0	-2	4	-2	2	4	10	0	2	0	6	0
54	0	2	-2	4	-4	2	-6	-4	-8	2	2	8	-4	-6	-2	0
55	0	6	-2	0	4	-2	2	0	4	2	-2	0	0	2	2	0
56	0	2	-6	0	-2	4	4	-2	2	0	4	-2	-4	-2	2	0
57	0	6	-6	-4	-2	-8	-4	2	-2	-8	0	-2	0	-2	-2	0
58	0	6	4	-2	2	4	-10	-4	0	2	-8	-2	-2	4	6	0
59	0	2	0	-2	**-18**	4	-2	0	12	2	0	2	-2	0	2	0
60	0	-6	8	-2	2	4	-10	-4	-6	0	-2	0	0	-2	0	2
61	0	-2	-8	2	2	0	-2	0	-6	4	-2	4	-8	2	0	-2
62	0	-6	6	-4	2	-4	0	-2	0	-2	-2	0	-2	4	-4	-2
63	0	14	10	4	-2	-4	0	2	-8	-6	10	0	-6	4	-4	2

Table 5.8: LAT for S_7 of DES

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	2	0	-2	-2	0	2	0	-2	0	-2	4	-4	-2	0	6
3	0	-2	0	2	-2	4	-6	4	-2	-4	6	0	-4	2	0	2
4	0	-2	-2	0	0	2	-2	-4	2	0	4	-2	6	0	0	14
5	0	-2	-2	0	0	2	6	4	-2	12	-8	2	-6	4	4	2
6	0	0	2	2	2	6	0	-4	4	4	2	2	2	-2	4	-8
7	0	-4	2	-10	-6	2	8	0	0	-4	-2	2	-2	-2	0	0
8	0	0	2	-2	0	0	2	-2	-2	2	0	0	2	-2	4	-4
9	0	4	-2	-2	4	0	-6	2	2	2	0	12	-6	-6	0	-4
10	0	2	-2	0	2	4	-4	-2	0	-2	-2	4	6	-4	0	-2
11	0	2	2	12	6	8	4	-2	4	-6	-2	4	-2	-4	4	2
12	0	-2	0	-2	0	2	0	-6	0	2	-4	-10	0	6	4	-6
13	0	2	-4	-2	4	2	0	6	0	-2	0	6	-4	6	4	-2
14	0	0	0	4	-2	2	2	2	-2	-6	2	2	4	4	4	0
15	0	0	4	0	-6	-2	-6	2	-2	2	6	6	8	0	4	0

16	0	0	0	0	0	0	0	-8	0	0	0	0	0	-8	0	**-16**
17	0	0	0	0	0	-8	0	0	0	8	0	-8	0	8	0	0
18	0	2	0	-2	-2	0	2	8	-2	0	-2	4	-4	6	0	-10
19	0	-2	0	2	-2	-4	10	4	-2	4	-10	8	-4	-6	0	2
20	0	-2	-6	-4	4	-2	-2	-4	-2	4	-4	-2	-2	0	4	2
21	0	-2	2	4	-12	6	-2	4	-6	-8	-8	2	2	4	0	-2
22	0	0	6	-10	-2	-6	0	-4	8	0	-6	2	-6	-2	0	4
23	0	-4	-2	2	6	-2	0	0	4	0	-2	2	6	-2	4	4
24	0	-4	2	2	0	4	2	2	6	-2	0	-4	2	2	-4	-8
25	0	0	6	10	-4	4	-6	-2	2	-2	0	0	-6	-2	0	0
26	0	-2	6	-4	-6	0	-4	2	0	2	-2	0	6	0	0	2
27	0	-2	2	0	-10	4	4	-6	-4	-2	-2	-8	-2	0	-4	-2
28	0	-6	-4	-2	4	2	0	6	4	2	4	2	8	2	0	-6
29	0	-2	-8	-2	0	2	-8	2	-4	-2	0	2	4	2	0	-2
30	0	-4	-4	4	2	2	2	-2	2	-6	-6	-2	-4	0	0	0
31	0	-4	0	0	-10	-2	2	-2	-6	2	6	2	0	-4	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	-2	4	-2	2	0	-6	4	-2	-4	-6	-4	0	-2	**16**	2
35	0	2	-4	2	2	-4	-6	8	-2	0	-6	-8	0	-6	-8	-2
36	0	2	2	-8	0	6	2	4	-2	0	4	2	2	0	0	2
37	0	2	2	-8	0	6	-6	-4	2	4	0	-2	-2	-4	-4	-2
38	0	8	2	-6	-2	-6	-4	0	0	-8	6	-2	-6	-2	4	0
39	0	-4	-6	-2	-10	-2	-4	4	4	0	2	6	-2	-2	0	0
40	0	0	2	-2	0	0	-6	6	-6	-2	-4	-4	-2	-6	-8	0
41	0	4	-2	-2	4	0	2	-6	-2	-2	-4	8	6	6	-12	0
42	0	-2	2	0	-2	-4	4	2	4	-2	-2	0	6	-8	4	-2
43	0	-10	-2	-4	2	8	4	2	-8	2	6	0	-2	0	0	2
44	0	-6	4	-2	0	-2	-4	-14	0	-2	0	6	0	2	0	2
45	0	-2	0	-2	4	-2	-4	-2	-8	2	-4	-2	4	-6	8	-2
46	0	0	0	4	2	-10	-2	-2	-10	2	2	2	0	0	0	-4
47	0	8	-4	0	-2	10	-2	-2	-2	10	6	-2	-4	-4	0	4
48	0	-4	0	4	0	4	0	4	0	4	0	-4	0	-12	0	4
49	0	4	0	-4	0	4	0	4	0	4	0	-4	0	-4	0	-4
50	0	-6	-4	-6	2	4	2	0	-2	0	2	0	0	2	8	-2
51	0	6	4	6	2	0	2	4	-2	4	2	-4	0	6	0	2
52	0	-2	-10	0	-4	-2	2	0	2	0	-4	-2	10	-4	-4	2
53	0	6	-2	0	-4	-2	2	0	6	4	0	2	6	0	0	-2
54	0	4	6	2	2	-6	4	4	-4	0	6	2	2	-6	0	0
55	0	0	6	6	-6	-2	-4	0	0	8	-6	2	6	2	4	0
56	0	8	10	-2	0	8	2	-2	-6	6	-4	4	6	2	0	0
57	0	4	-2	-2	-4	0	-6	2	6	-2	-4	0	-2	6	4	0
58	0	-10	10	0	6	-4	4	2	-4	-2	6	0	-2	0	-4	-2
59	0	6	-2	-4	2	0	4	2	-8	-6	-2	0	6	0	0	2
60	0	2	0	2	-4	-6	4	2	4	2	8	-2	0	2	4	-2
61	0	-2	12	-6	8	2	-4	6	4	-2	-4	-2	4	2	-4	2
62	0	-8	4	0	-2	2	-2	6	10	6	2	2	0	0	-4	0
63	0	-8	0	4	2	-2	-10	-2	-6	6	-2	6	-4	4	-4	0

Table 5.9: LAT for S_8 of DES

5.6 LATs for AES

For the AES S-Box (and its inverse), the linear uniformity is 16. The LATs for AES are too large to be displayed in this document with good visualization quality, therefore, we make them available as `.csv`

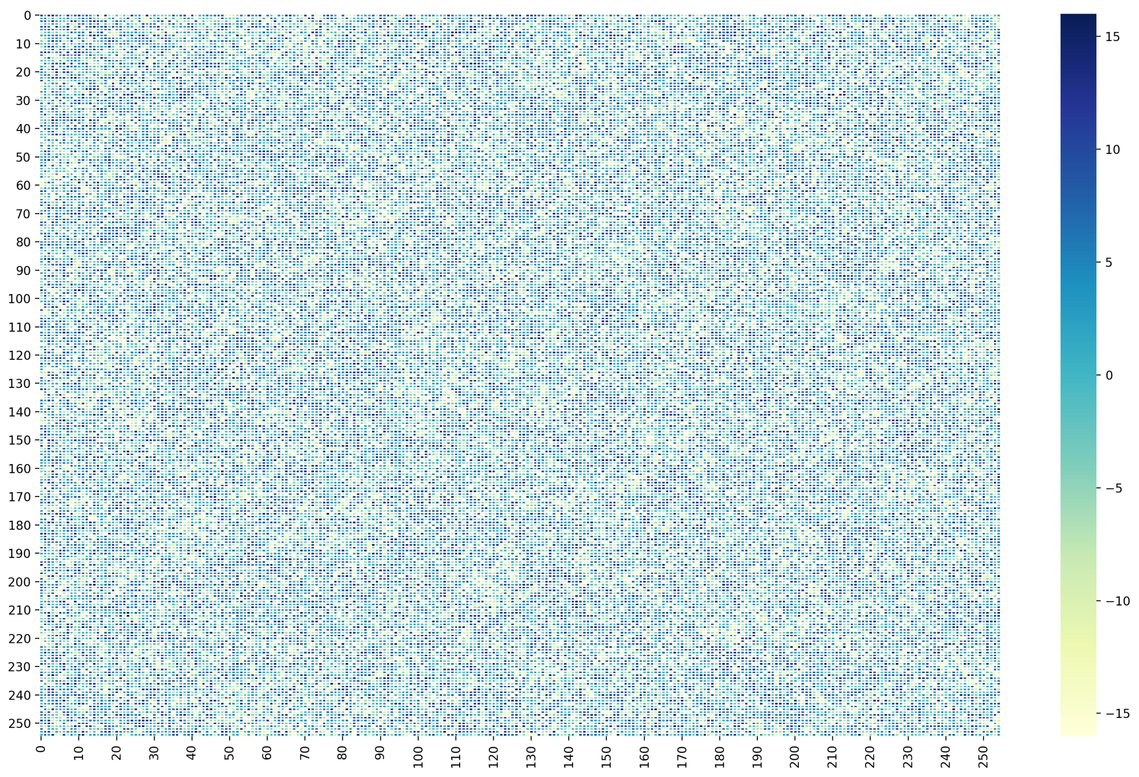


Figure 5.1: LAT heatmap for AES S-Box

files at our Git repository [66], together with the code used to obtain them. Figures 5.1 and 5.2 show one possible visual representation (heatmap) of the AES LATs.

5.7 LC versus DC

Linear Cryptanalysis is more efficient in breaking DES than Differential Cryptanalysis [10], since it is capable of breaking the full 16-round DES in less steps. Furthermore, note that, in LC, we do not need *joint LATs* or other means of restricting S-Box activation, because the usage of bit masks allows the attacker a better control of the propagation of the linear approximations.

This suggests, as mentioned in [47], that the DES designers took DC in account, but not LC. Also, Matsui [54] is able to mount an attack taking advantage of the linear uniformity, while Biham and Shamir [10] cannot take advantage of the differential uniformity of the DES S-Boxes and must find a suitable characteristic for their attack by other means, due to the difficulties incurred by the DES design against DC. See [14] and Chapter 4 of this work for more about DC on DES.

Table 5.10 briefly compares the core concepts of DC and LC. Figures 5.3 and 5.4 show how differences (respectively linear relations) propagate through the DES F function and the S-Boxes.

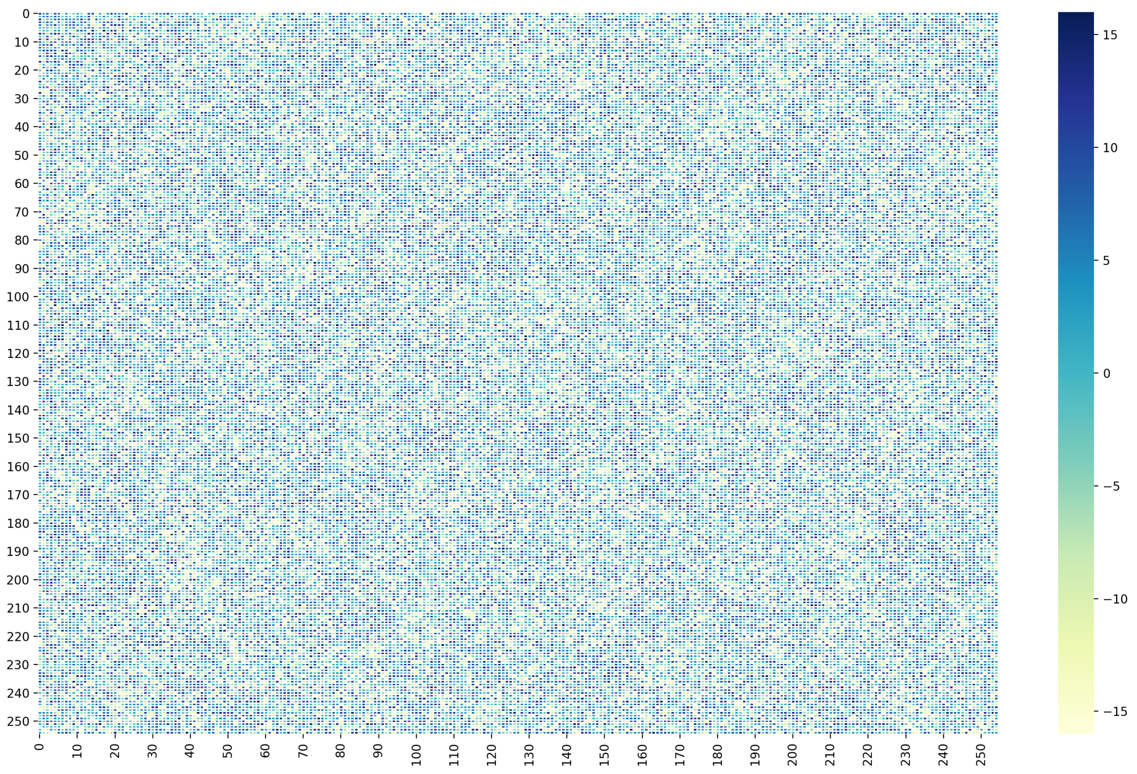


Figure 5.2: LAT heatmap for AES inverse S-Box

Differential Cryptanalysis Concepts	**Linear Cryptanalysis Concepts**
Chosen-plaintext attack (CPA)	Known-plaintext attack (KPA)
Invented by Biham and Shamir [10]	Invented by Matsui [54]
Difference: $X' = X \oplus X^*$	Bitmask: $X \cdot \gamma$
DDT (Difference Distribution Table)	LAT (Linear Approximation Table)
Active S-Box: input difference $\neq 0$	Active S-Box: output mask $\neq 0$
Distinguisher: differential characteristic	Distinguisher: linear relation (or approximation)
Probability of an r-round differential characteristic: $p = \prod_{i=1}^{r} p_i$	Bias of an r-round linear relation: $\epsilon = 2^{r-1} \cdot \prod_{i=1}^{r} \epsilon_i$, from the Piling-up Lemma
Chosen-plaintexts for a characteristic with probability p: approximately $1/p$	Known-plaintexts for a linear relation with bias ϵ: approximately $1/\epsilon^2$
Differential trail (follow non-zero differences)	Linear trail (follow non-zero masks)

Table 5.10: Comparing DC and LC concepts

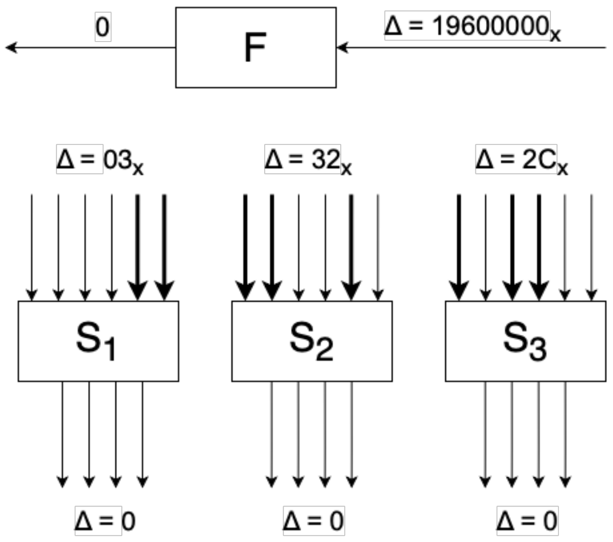


Figure 5.3: Propagation of differences in DC (differential trails)

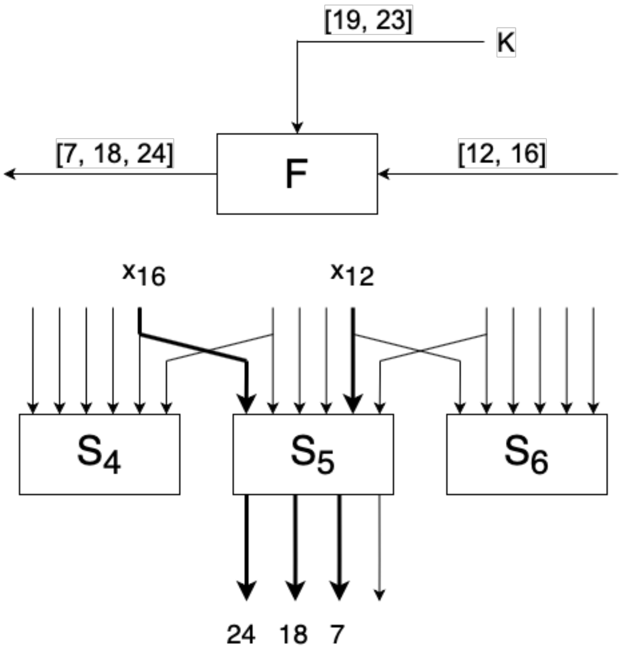


Figure 5.4: Propagation of linear relations in LC (linear trails)

5.8 Conclusions

In this chapter, we gave a brief overview of Linear Cryptanalysis and explained how to obtain LATs — Linear Approximation Tables. We also presented LATs for DES S-Boxes and AES S-Boxes, as well as their linear uniformities. We leave the application of an actual LC attack as a future work.

Chapter 6

Comparing AES and DES

In this chapter, we provide a comparison of two classical ciphers, AES [59] and DES [58]. They are both NIST standards, with AES being the chosen to replace DES. We compare them with respect to their general structure, their diffusion process, their S-Boxes and their vulnerability against DC [11], LC [54] and exhaustive key search [31] attacks. We assume the reader is familiar with:

- The AES cipher structure, available in [59] and in Appendix A
- The DES cipher structure, available in [58] and in Appendix A
- Block cipher, Feistel Network and Substitution Permutation Network definitions, available in Appendix A
- The ANF analysis conducted in Chapter 1
- The diffusion analysis of DES conducted in Chapter 2
- The diffusion analysis of AES conducted in Chapter 3
- The Differential Cryptanalysis overview conducted in Chapter 4
- The Linear Cryptanalysis overview conducted in Chapter 5

6.1 Acronyms

- AES: Advanced Encryption Standard
- DES: Data Encryption Standard
- NIST: National Institute of Standards and Technology
- S-Box: Substitution Box
- ANF: Algebraic Normal Form
- DC: Differential Cryptanalysis
- LC: Linear Cryptanalysis
- LAT: Linear Approximation Table
- DDT: Differential Distribution Table

6.2 Comparative chart

From references [59] and [58] and Chapters 1 to 5 we know the general structures of AES and DES, the fact that they are both iterated block ciphers, AES being an SPN and DES being a Feistel Network. We also know their parameter sets (block size, key size), their most common state representations, how many rounds are required in order to achieve complete plaintext and key diffusion, the ANFs of their S-Boxes, and their differential and linear uniformities. These pieces of information allow us to devise Table 6.1, which compares the two ciphers.

Cipher	**AES**	**DES**
Reference	[59]	[58]
Standardization process	open contest	secret design
Standardization time	2001	1977
Environment	hardware and software	hardware
General structure		
Type	SPN	Feistel Network
Iterated	yes	yes
Block size (bits)	128	64
Key size(s) (bits)	128, 192, 256	64
Number of rounds	10, 12, 14	16
State representation	4×4 matrix of bytes	1-dimensional bit vector
Basic data unit	byte	bit
Key schedule	nonlinear	linear
Algebraic structure	GF(2^8) (polynomials)	GF(2) (bits only)
Diffusion		
Rounds until full plaintext diffusion	2	5
Rounds until full key diffusion	2, 3, 3	5
Diffusion components	`ShiftRows`, `MixColumns`, S-Boxes (bit level)	P, S-Boxes
MDS matrix usage	yes	no
S-Boxes		
Quantity	1	8
Invertible	yes	no
Input size(s)	8	6
Output size(s)	8	4
ANF(s) degree(s)	7	5
Differential uniformity	4	16
Linear uniformity	16	14, 16, 18, 20
Attacks		
Differential Cryptanalysis	resistant	vulnerable
Linear Cryptanalysis	resistant	vulnerable
Exhaustive key search	resistant	vulnerable

Table 6.1: Comparing AES and DES

6.3 Remarks

With respect to the DC attack, it is worth noting that the DES design attempted at thwarting it by sharing plaintext bits between different S-Boxes by means of the extension function E, as outlined by [14]. That led Biham and Shamir [11] to seek alternatives to successfully attack DES other than the exploitation of the high differential uniformity of the S-Boxes. In order to successfully attack DES, they carefully constructed characteristics with maximum probability along the cipher's iterations, activating 3 S-Boxes (S_1, S_2 and S_3) simultaneously with a characteristic of probability approximately equal to 1/234. The fact that plaintext bits are shared between adjacent S-Boxes whilst key bits are not lead us to the need of computing joint DDTs for thorough analysis of the DES S-Boxes, i.e we need the joint DDTs to find non-zero entries which could be useful when constructing characteristics for DC attacks.

However, with respect to LC, Matsui [54] is able to exploit the high linear uniformity (equal to 20) of the S_5 S-Box of DES to mount a successful LC attack.

In the case of AES, both attacks (DC and LC) were taken into account ([47], [24] and [23] provide further detail on this). The diffusion process in AES, through the usage of the MDS matrix, ensures that *the number of active S-Boxes increases exponentially along the cipher iterations.* Consequently, the probability of a characteristic (in the case of DC) and the bias of a linear relation (in the case of LC) decrease drastically. These small probabilities increase the required amount of plaintexts (chosen, for DC, and known, for LC) for the attacks, rendering DC and LC infeasible — the required amount of plaintexts surpasses the size of the AES codebook. In DES, however, the diffusion through the P permutation does not ensure a sufficient increase in the number of active S-Boxes, thus it is possible to find characteristics (and linear relations) with suitable probability, requiring an amount of plaintexts that does not surpass DES's codebook, and therefore making the attacks feasible (although computationally expensive).

In conclusion, the diffusion process of AES is able to thwart DC and LC attacks, whilst the diffusion process of DES is not, and the usage of an MDS matrix in the `MixColumns` transformation plays a central role in that (see Section 6.3.1). Tables 6.2 and 6.3 outline AES's resistance against DC and LC, respectively. Since the AES block size is 128 bits, its codebook comprises all possible values of 128 bits, thus 2^{128} elements. In Tables 6.2 and 6.3, it is possible to see that, at iteration 4, the amount of required plaintexts surpasses 2^{128} (and therefore DC and LC attacks cannot be mounted).

Iterations	**Minimum number of active S-Boxes**	**Total of active S-Boxes**	**Probability of a characteristic**	**Required chosen plaintexts**
1	1	1	$\leq 2^{-6}$	2^{6}
2	$4^1 = 4$	5	$\leq (2^{-6})^5 = 2^{-30}$	2^{30}
3	$4^2 = 16$	21	$\leq (2^{-6})^{21} = 2^{-126}$	2^{126}
4	≥ 4	≥ 25	$\leq (2^{-6})^{25} = 2^{-150}$	2^{150}

Table 6.2: AES's resistance against DC

The differential uniformity (see Chapter 4) of the AES S-Box is 4, and the total of possible entries of its DDT is $2^8 = 256$. Therefore, if one AES S-Box is active, the maximum probability of a characteristic is $4/256 = 2^{-6}$. If 5 S-Boxes are active, the maximum probability is $(2^{-6})^5$, and so forth. The required amount of chosen plaintexts is approximately the inverse of the probability, and thus, on iteration 4, since more than 25 S-Boxes will be active, a DC attack becomes infeasible (the amount of required plaintexts surpasses the codebook).

Iterations	**Mininum number of active S-Boxes**	**Total of active S-Boxes**	**Bias of a linear relation**	**Required known plaintexts**
1	1	1	$\leq (2^{-4})^1 \times 2^0 = 2^{-4}$	2^{8}
2	$4^1 = 4$	5	$\leq (2^{-4})^5 \times 2^4 = 2^{-16}$	2^{32}
3	$4^2 = 16$	21	$\leq (2^{-4})^{21} \times 2^{20} = 2^{-64}$	2^{128}
4	≥ 4	≥ 25	$\leq (2^{-4})^{25} \times 2^{24} = 2^{-76}$	2^{152}

Table 6.3: AES's resistance against LC

For LC, similar analysis applies. The linear uniformity of the AES S-Box is 16, and the total of possible entries of its LAT is $2^8 = 256$. The bias of a random variable (linear relation) $X_{i_1} \oplus X_{i_2} \oplus ... \oplus X_{i_k}$ is $\epsilon_{i_1,i_2,...,i_k} = 2^{k-1} \prod_{j=1}^{k} \epsilon_{i_j}$ (see Chapter 5). In the case of a single active S-Box, $k = 1$, therefore the bias of a linear relation is 2^{-4}. For 4 active S-Boxes, the maximum bias is $(2^{-4})^5 \times 2^4 = 2^{-16}$, and so forth. The necessary known plaintexts are approximately $1/\epsilon^2$, where ϵ is the bias, therefore an LC attack becomes infeasible at iteration 4.

6.3.1 `MixColumns` and the number of active S-Boxes throughout the AES iterations

In [22], the AES (Rijndael) authors introduce the *branch number*, a measure of diffusion for the AES transformations and, particularly, for the `MixColumns` transformation.

(Byte weight of a vector [22]) *The byte weight $W(a)$ of a (byte) vector a is the number of non-zero bytes in a. A non-zero byte is an* active *byte.*

(Branch number of a linear transformation [22]) *The* branch number *of a linear transformation T is $min_{a \neq 0}(W(a) + W(T(a)))$.*

The MDS matrix used in the `MixColumns` step of AES has branch number equal to 5. That means the sum of the non-zero input bits and the non-zero output bits of `MixColumns` is always at least 5. Therefore, a non-zero difference propagates through the AES iterations as illustrated in Figure 6.1, resulting in the total active S-Box counts shown in Tables 6.2 and 6.3. A smaller branch number, e.g 4, would lead to less active S-Boxes on the earlier rounds of the cipher, resulting in a need for more iterations until resistance against DC and LC could be achieved.

In conclusion, when choosing matrices, we should look for high branch numbers, so that the number of active S-Boxes increases rapidly and, as a consequence, resistance against DC and LC can be achieved in early rounds. Figures 6.2 and 6.3 compare the number of active S-Boxes in AES and DES for DC and LC, illustrating this concept. We can observe that the number of active S-Boxes increases more rapidly on AES in both cases, thus making it more secure against the attacks.

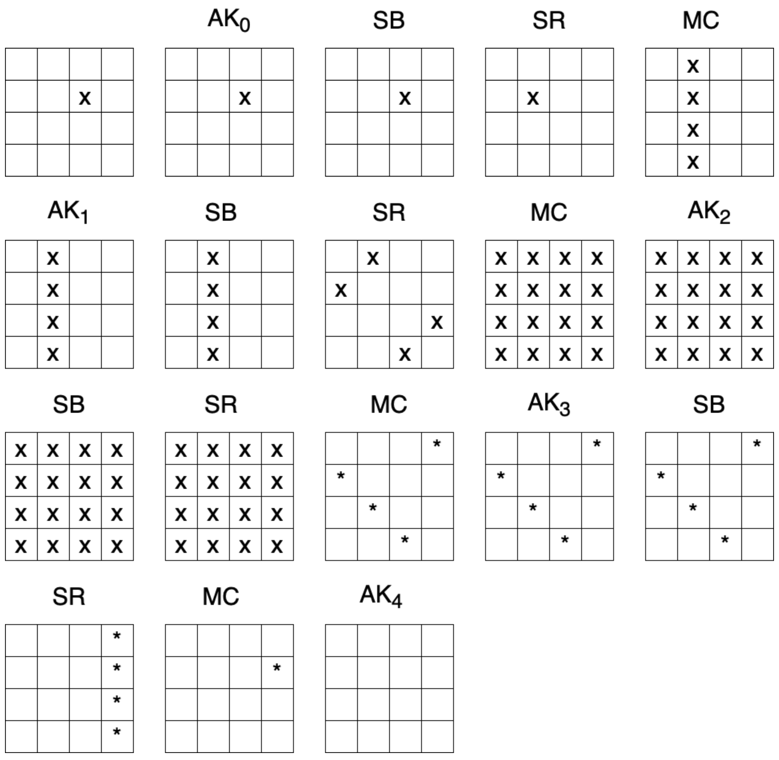


Figure 6.1: Non-zero byte propagation example through AES. AK_i denotes `AddRoundKey` of round i, SB denotes `SubBytes`, SR denotes `ShiftRows` and MC denotes `MixColumns`. Note that the non-zero byte could be located anywhere, and the minimum number of active S-Boxes after 4 iterations would still be 25.

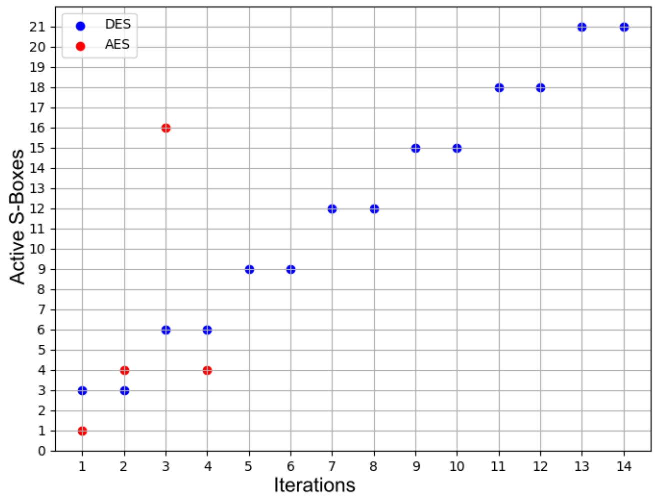


Figure 6.2: Comparing number of active S-Boxes (Differential Cryptanalysis)

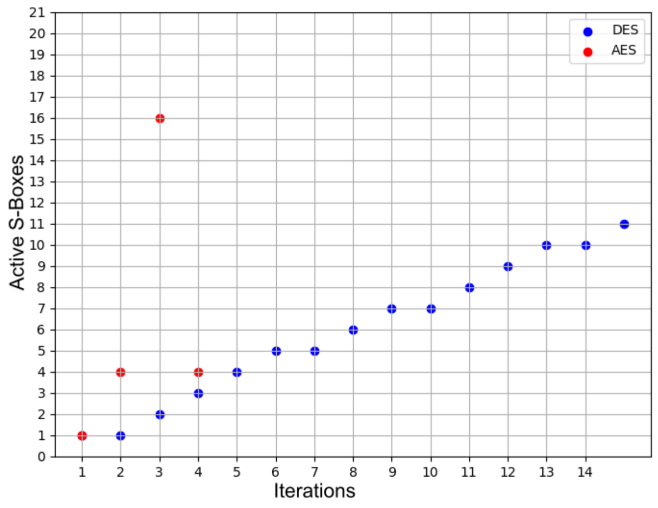


Figure 6.3: Comparing number of active S-Boxes (Linear Cryptanalysis)

Chapter 7

History of the application of MDS matrices in cryptography

MDS matrices have been widely used in the construction of diffusion layers for block ciphers such as SHARK [62], SQUARE [20], BKSQ [21], KHAZAD [4], ANUBIS [3], Hierocrypt-3 [17], Rijndael (AES) [25] and Curupira [5]. They have also been applied in the design of hash functions (e.g Whirlwind [2] and Grøstl [32]). The choice is due to the fact that MDS codes provide transformations with optimal linear and differential branch numbers (see e.g [20] or [62]), thus contributing to security against Differential and Linear Cryptanalysis attacks.

When a matrix is MDS, optimal *branch number* (a measure of diffusion power) is ensured, therefore, from the theoretical security perspective, any two distinct $n \times n$ MDS matrices present equal contribution to a cipher's design in terms of diffusion power. However, their computational cost, which is a relevant practical implementation criterion, *is not* necessarily the same. This motivates not only the search for MDS matrices, but the search for *MDS matrices with low computational cost.* In this work, the computational cost of a matrix A is measured by the amount of **xor** and **xtime** operations required when multiplying a cipher's state column vector by A.

Due to the computational cost of matrix multiplication, there is an interest in finding MDS matrices with coefficients as small as possible, in order to minimize the required amount of **xor** and **xtime** operations required by the implementations. However, the complexity of finding MDS matrices through random search increases proportionally to the dimension, which led to the investigation of systematic methods to construct (or find) MDS matrices. One possible avenue is trying to find direct mathematical constructions which ensure the MDS property, and another is to impose restrictions to limit the random search space (e.g imposing the matrix should be circulant, as was done by the authors of [20]). Furthermore, there is an interest in finding involutory MDS matrices (as pointed by [4] and [3]), so that the encryption and the decryption computational cost are the same.

It is also worth noting that, although MDS matrices are widely used in cryptographic algorithms, there are designs which prefer not to make use of them. The block ciphers Serpent [9], IDEA [46] and PRESENT [12], for instance, do not include MDS matrices in their design. The hash function Keccak [8], which was later selected by NIST to become the SHA-3 standard, also does not use MDS matrices. The computational cost may be related to this choice, since algorithms without matrix multiplications can be more lightweight than those with matrix multiplications.

In this chapter, we aim at providing a history of the application of MDS matrices in cryptography, listing the matrices, the ciphers in which they have been applied, the respective Finite Fields (order and irreducible polynomial), and their cost (amount of **xor** and **xtime** operations).

7.1 Notation

- $det(A)$: determinant of the matrix A
- A^{-1}: inverse matrix of A
- n, k, d: parameters of a code
- $\mathcal{C}$: a code

- G: generator matrix of a code
- I_n: the $n \times n$ identity matrix
- $[I_nB]$: matrix obtained by placing the $n \times n$ matrix B to the right of the $n \times n$ identity matrix I_n. For example, for $B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $[I_nB] = \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 3 & 4 \end{bmatrix}$
- $a \oplus b$: xor (exclusive or) between a and b
- $a+b$: addition between a and b. When a and b are integers, integer addition. However, when a and b are elements of a finite field, it refers to addition within the finite field. When adding in a finite field of characteristic 2 ($GF(2^m)$), addition is equivalent to exclusive or and therefore $+$ means the same as $\oplus$.
- $\mathcal{B}_d(A)$: differential branch number of matrix A
- $\mathcal{B}_l(A)$: linear branch number of matrix A
- $\mathcal{B}(A)$: linear and differential branch number of matrix A, when they are equal. When they are equal, we refer to it as "branch number" only since it is not necessary to specify
- $circ(...)$: circulant matrix
- $rcirc(...)$: right circulant matrix
- $lcirc(...)$: left circulant matrix
- $vand(...)$: Vandermonde matrix
- $Hank(...)$: Hankel matrix
- $Toep(...)$: Toeplitz matrix
- $had(...)$: Hadamard matrix
- $hc(...)$: Hadamard-Cauchy matrix
- $Companion(...)$: companion matrix

7.2 Acronyms

- MDS: Maximum Distance Separable
- AES: Advanced Encryption Standard (Rijndael's name after being chosen by NIST)
- **xor**: bitwise XOR between two bit strings
- **xtime**: refers to the multiplication by the monomial x in the finite field GF(2^m) for an integer m

7.3 Relevant Background

In this section, we present the main relevant concepts related to MDS matrices, core for a complete understanding on the topic and the current state of the art. We start with a review of abstract algebra (groups, fields, rings) in Section 7.3.1. Then, in Section 7.3.2, we introduce finite fields, which are a special kind of field widely used in cryptography, i.e all the matrices are made of elements of finite fields (although implemented with integers) and therefore finite field arithmetic is relevant background. Section 7.3.3 introduces the main concepts and definitions regarding linear codes. Section 7.3.4 defines the MDS (Maximum Distance Separable) property studied in this work, relates it to linear codes and presents theorems which can be used to evaluate if a given matrix is (or not) MDS. Finally, Section 7.3.5 presents the types of matrices studied in the literature for construction of MDS matrices and related theorems.

7.3.1 Abstract algebra — Groups, Rings and Fields

Definition 7 (Abelian Group [25]) *An Abelian group $\langle G, +\rangle$ consists of a set G and an operation defined on its elements, here denoted by $+$:*

$$+ : G \times G \to G : (a, b) \to a + b.$$

In order to qualify as an Abelian group, the operation has to fulfill the following conditions:

1. ***Closure:*** $\forall a, b \in G$, $a + b \in G$
2. ***Associativity:*** $\forall a, b, c \in G$, $(a + b) + c = a + (b + c)$
3. ***Commutativity:*** $\forall a, b \in G$, $a + b = b + a$
4. ***Neutral Element:*** $\exists 0 \in G$, *such that* $\forall a \in G$, $a + 0 = a$
5. ***Inverse Elements:*** $\forall a \in G$, $\exists b \in G$, *such that* $a + b = 0$

The best-known example of an Abelian group is $\langle \mathbb{Z}, +\rangle$: the set of integers, with the operation 'addition'. Since the addition of integers is the best-known example of a group, usually the symbol '+' is used to denote an arbitrary group operation.

Definition 8 (Ring [25]) *A ring $\langle R, +, \cdot\rangle$ consists of a set R with two operations defined on its elements, here denoted by '+' and '$\cdot$'. In order to qualify as a ring, the operations have to fulfill the following conditions:*

1. *The structure $\langle R, +\rangle$ is an Abelian group.*
2. *The operation '$\cdot$' is closed and associative over R. There is a neutral element for '$\cdot$' in R.*
3. *The two operations '+' and '$\cdot$' are related by the law of distributivity: $\forall a, b, c \in R$, $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$*

The neutral element for '$\cdot$' is usually denoted by 1. A ring $\langle R, +, \cdot\rangle$ is called a commutative ring if the operation '$\cdot$' is commutative.

The best-known example of a ring is $\langle \mathbb{Z}, +, \cdot\rangle$: the set of integers, with the operations 'addition' and 'multiplication'. This ring is a commutative ring. The set of matrices with n rows and n columns, with the operations 'matrix addition' and 'matrix multiplication', is a ring, but not a commutative ring (if $n > 1$).

Definition 9 (Field [25]) *A structure $\langle F, +, \cdot\rangle$ is a field if the following two conditions are satisfied:*

1. *$\langle F, +, \cdot\rangle$ is a commutative ring.*
2. *For all elements of F, there is an inverse element in F with respect to the operation '$\cdot$', except for the element 0, the neutral element of $\langle F, +\rangle$.*

A structure $\langle F, +, \cdot\rangle$ is a field if and only if both $\langle F, +\rangle$ and $\langle F \setminus \{0\}, \cdot\rangle$ are Abelian groups and the law of distributivity applies. The neutral element of $\langle F \setminus \{0\}, \cdot\rangle$ is called the unit element of the field.

The best-known example of a field is the set of real numbers, with the operations 'addition' and 'multiplication'. Other examples are the set of complex numbers and the set of rational numbers, with the same operations. Note that for these examples, the number of elements is infinite.

7.3.2 Finite fields — GF(2^m)

(Finite field [25]) *A* finite field *is a field with a finite number of elements. The number of elements in the set is called the* order *of the field.*

(Characteristic and order [25]) *A field with order r exists if and only if r is a prime power, i.e $r = p^m$ for some integer m, where p is a prime integer. p is called the* characteristic *of the field. For each prime power there is exactly one finite field, denoted by GF(p^m).*

(Representing finite fields with prime order [25]) *Elements of a finite field GF(p) can be represented by the integers* $0, 1, ..., p-1$*, and the field operations are integer addition modulo p and integer multiplication modulo p.*

(Representing finite fields with non-prime order [25]) *For finite fields with an order* $r = p^m$ *that is not prime, addition and multiplication cannot be represented by addition and multiplication modulo r, and instead other representations must be used. One of the possible representations for GF(*p^m*) is by means of* polynomials over GF(p) with degree at most $m-1$. *Addition and multiplication are then defined modulo an* irreducible polynomial of degree m.

(Polynomial [25]) *A polynomial over a field* $\mathbb{F}$ *is an expression of the form*

$$b(x) = b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + ... + b_2x^2 + b_1x + b_0,$$

where x is the indeterminate *and* $b_i \in \mathbb{F}$ *are the coefficients. The* degree *of the polynomial equals l if* $b_j = 0$ *for all* $j > l$ *and l is the smallest number with this property. When* $b(x)$ *is the zero polynomial, the degree is* -1 *by convention.*

Note that, when using polynomials over GF(p) to represent the GF(p^m) field, the degree is at most $m-1$.

In this chapter, we focus particularly on fields with characteristic $p = 2$, due to their wide application in cryptography.

Addition and multiplication are defined on polynomials as follows.

(Polynomial addition [25]) *Summing two polynomials* $a(x)$ *and* $b(x)$ *consists of summing the coefficients with equal powers of x, with the sum occuring in the underlying field* $\mathbb{F}$*. The neutral element is 0 (the polynomial with all coefficients equal to 0). The inverse element can be found by replacing each coefficient by its inverse element in* $\mathbb{F}$*. The degree of* $a(x) + b(x)$ *is at most the maximum of the degrees of* $a(x)$ *and* $b(x)$*, therefore addition is closed.*

For polynomials over GF(2) stored as integers in a cryptographic software implementation, addition can be implemented with a bitwise XOR instruction.

(Polynomial multiplication [25]) *In order to make multiplication closed, we select a polynomial* $p(x)$ *of degree l, called the* reduction polynomial. *Multiplication of* $a(x)$ *and* $b(x)$ *is then defined as the algebraic product of the polynomials modulo the reduction polynomial* $p(x)$.

The neutral element is 1 (the polynomial of degree 0 and with coefficient of x^0 *equal to 1). The inverse element of* $a(x)$ *is* $a^{-1}(x)$ *such that* $a(x) \times a^{-1}(x) = 1 \bmod p(x)$*. Note that* $a^{-1}(x)$ *exists only when* $a(x) \neq 0$.

For polynomials over GF(2) stored as integers in a cryptographic software implementation, multiplication by x can be implemented as a logical bit shift followed by conditional XOR (i.e subtraction) of the reduction polynomial (the **xtime** operation). Multiplication by other polynomials can be implemented as a series of **xtime**.

The reduction polynomial is chosen as an irreducible polynomial.

(Irreducible polynomial [25]) *A polynomial* $d(x)$ *is irreducible over the field GF(p) if and only if there exist no two polynomials* $a(x)$ *and* $b(x)$ *with coefficients in GF(p) such that* $d(x) = a(x) \times b(x)$*, where* $a(x)$ *and* $b(x)$ *are of degree greater than 0.*

For further reference on abstract algebra and Finite Fields, the reader may refer to [53], [56] and [55].

7.3.3 Linear codes

Definition 10 (Hamming weight [51]) *The Hamming weight* $w(x)$ *of a vector* x *is the number of nonzero components of the vector* x.

Definition 11 (Hamming distance [51]) *The Hamming distance between two vectors* x *and* y *is* $w(x-y)$*, which is equal to the Hamming weight of the difference of the two vectors.*

Definition 12 (Linear code [51]) *A linear $[n, k, d]$ code over a field $\mathbb{F}$ is a k-dimensional subspace of the vector space $\mathbb{F}^n$, where any two different vectors of the subspace have a Hamming distance of at least d, and d is the largest number with this property.*

The distance d of a linear code equals the minimum weight of a non-zero codeword in the code. A linear code can be described by generator and/or parity-check matrices.

Definition 13 (Generator matrix [51]) *A generator matrix G for an $[n, k, d]$ code C is a $k \times n$ matrix whose rows form a vector space basis for C. The choice of a basis in a vector space is not unique, thus a code has many different generator matrices which can be reduced to one another by performing elementary row operations.*

Definition 14 (Parity-check matrix [51]) *A parity-check matrix H for an $[n, k, d]$ code C is an $(n - k) \times k$ matrix with the property that a vector x is a codeword of C if and only if $Hx^T = 0$.*

Theorem 1 (Singleton Bound [51]) *If C is an $[n, k, d]$ code, then $d \leq n - k + 1$.*

7.3.4 MDS (Maximum Distance Separable) property

Definition 15 (MDS (Maximum Distance Separable) code [51]) *An* MDS code *is a* linear code *that meets the* Singleton bound, *i.e a linear code with $d = n - k + 1$.*

Definition 16 (Differential branch number [25]) *The differential branch number $\mathcal{B}_d(M)$ of a matrix M of order n over a finite field $\mathbb{F}_{2^r}$ is defined as the minimum number of nonzero components in the input vector x and the output vector $M \cdot x$ as we range over all nonzero $x \in (\mathbb{F}_{2^r})^n$, i.e*

$$\mathcal{B}_d(M) = min_{x \neq 0}\{W(x) + W(M \cdot x)\}$$

where $W(x)$ denotes the weight of the vector x i.e number of nonzero components of the vector x.

Note that the differential branch number of a matrix M is exactly the distance of the linear code generated by the matrix $[I_n M]$.

Definition 17 (Linear branch number [25]) *The linear branch number $\mathcal{B}_l(M)$ of a matrix M of order n over a finite field $\mathbb{F}_{2^r}$ is defined as the minimum number of nonzero components in the input vector x and the output vector $M^T \cdot x$ as we range over all nonzero $x \in (\mathbb{F}_{2^r})^n$, i.e*

$$\mathcal{B}_d(M) = min_{x \neq 0}\{W(x) + W(M^T \cdot x)\}$$

where $W(x)$ denotes the weight of the vector x i.e number of nonzero components of the vector x.

Definition 18 (Branch number [25]) *When the differential branch number and the linear branch number are equal, we call them just "branch number". Furthermore, given a matrix M, associated to an $[n, k, d]$ linear code, the linear mapping defined by $\theta(a) = M \cdot a$ has branch number $\mathcal{B}(\theta) = d$. When the linear code is MDS, i.e when M is MDS, the mapping is an optimal diffusion mapping.*

An MDS matrix associated to an $[n, k, d]$ code has *branch number* equal to d, which is the maximum possible branch number, thus providing optimal diffusion.

Definition 19 (Singular matrix) *A square matrix A is singular if and only if $det(A) = 0$.*

Definition 20 (Non-singular matrix) *A is non-singular if and only if $det(A) \neq 0$.*

Definition 21 (Submatrix) *Given a matrix M, a submatrix of M is the matrix obtained after removing z rows and columns of M, $z \geq 1$, provided that there are sufficient rows (and columns) to be removed.*

Theorem 2 (MDS codes [51]) *An $[n, k, d]$-code $\mathcal{C}$ with generator matrix $G = [I_n B]$, where B is a $k \times (n - k)$ matrix, is MDS if and only if every square submatrix of B is non-singular.*

Definition 22 (MDS matrix) *When we say* ***MDS matrix****, we are referring to the matrix B of a generator matrix $G = [I_n B]$ of a MDS code $\mathcal{C}$, i.e a code meeting the conditions of Theorem 2.*

Note that Definition 15 establishes the conditions for a code to be MDS, therefore Theorem 2 is an alternative way of evaluating a code, or a matrix, with respect to the MDS property. For further detail on matrices, determinants and linear algebra, the reader may refer to [48].

Another characterization of MDS matrices is given by [70] (see Proposition 1 below).

Proposition 1 ([70]) *A $k \times k$ matrix M is an MDS matrix if and only if the standard form generator matrix $[I_k M]$ generates a $(2k, k, k+1)$-MDS code.*

Fact 1 ([35]) *All square submatrices of an MDS matrix are MDS.*

Corollary 1 ([35]) *If A is an MDS matrix, then A^T is also an MDS matrix.*

Corollary 2 ([35]) *The inverse of an MDS matrix is MDS.*

Fact 2 *[35] A square matrix A of order n is MDS if and only if $\mathcal{B}_d(A) = \mathcal{B}_l(A) = n+1$.*

7.3.5 Matrix types

Definition 23 (Transpose matrix) *The transpose of a matrix A, denoted by A^T, is the matrix such that $A^T[i][j] = A[j][i]$. In other words, it is the matrix obtained by writing the rows of A as columns. Note that, if A is an $m \times n$ matrix, then A^T will be $n \times m$.*

Definition 24 (Orthogonal matrix [35]) *A square matrix M is called orthogonal matrix if $M \times M^T = I$, where I is the identity matrix.*

Definition 25 (Involutory matrix) *An $n \times n$ square matrix A is involutory if $A \times A = I_n$, where I_n is the identity matrix. In other words, A is involutory when $A = A^{-1}$.*

Definition 26 (Circulant matrix) *An $n \times n$ matrix A is circulant if each row i is formed by a cyclical shift of i positions of the same set of elements $\{a_0, a_1, a_2, ..., a_{n-1}\}$.*

Lemma 1 ([35]) *The product of two circulant matrices is a circulant matrix. Also the inverse and transpose of a circulant matrix are circulant.*

Definition 27 (Left circulant matrix) *A circulant matrix in which the shift is a cyclical shift to the left, i.e*

$$A = \begin{bmatrix} a_0 & a_1 & ... & ... & a_{n-1} \\ a_1 & a_2 & ... & a_{n-1} & a_0 \\ ... & ... & ... & ... & ... \\ a_{n-1} & a_0 & ... & ... & a_{n-2} \end{bmatrix}.$$

In [49], there is a more formal definition.

Definition 28 ([49]) *A left-circulant matrix L of order k is a matrix where each subsequent row is a left-rotation of the previous row, denoted by $lcirc(c_0, c_1, ..., c_{k-1})$, where c_i's are the entries of the first row of the matrix. The (i,j)-entry of L can be expresed as $L[i,j] = c_{(i+j) \ mod \ k}$.*

Definition 29 (Right circulant matrix) *A circulant matrix in which the shift is a cyclical shift to the right, i.e*

$$A = \begin{bmatrix} a_0 & a_1 & ... & ... & a_{n-1} \\ a_{n-1} & a_0 & a_1 & ... & a_{n-2} \\ ... & ... & ... & ... & ... \\ a_1 & ... & a_{n-2} & a_{n-1} & a_0 \end{bmatrix}.$$

Note that circulant matrices can be defined by just one row, since all the other rows are cyclical shifts of the first. Therefore, they can be denoted as $circ(a_0, a_1, ..., a_{n-1})$. In the case of left circulant and right circulant matrices, respectively, $lcirc(a_0, ..., a_{n-1})$ and $rcirc(a_0, ..., a_{n-1})$. For example, matrices (7.56) and its inverse (7.57), used in the Rijndael cipher, can be denoted as $rcirc(02_x, 03_x, 01_x, 01_x)$ and $rcirc(0e_x, 0b_x, 0d_x, 09_x)$. Since a circulant matrix can be defined by one row only, only n elements are required, and this optimizes the process of searching for (and finding) such matrices. However, *there are no guarantees about the MDS property.* One must check whether Theorem 2 holds to ensure the obtained matrix is MDS. As an example, the circulant matrix initially employed in Whirlpool-0 [69] is not MDS, and was further replaced by an actual MDS matrix found by Shirai in [68].

Theorem 3 ([49]) *For order $k \leq 8$, there are at most 5 types of MDS circulant matrices, namely circulant matrices whose first row has:*

Type 0: *k distinct entries;*
Type 1: *1 pair of repeated entries;*
Type 2: *2 pairs of repeated entries;*
Type 3: *3 pairs of repeated entries;*
Type 4: *or 3 repeated entries.*

Definition 30 ([49]) *A cyclic matrix C_ρ of order k is a matrix where each subsequent row is some permutation ρ of the previous row, where ρ is a cycle of length k. We denote the matrix as $cyc_\rho(c_0, c_1, ..., c_{k-1})$, where c_i's are the entries of the first row of the matrix. The (i, j) entry of C_ρ can be expressed as $C_\rho[i][j] = c_{\rho^i(j)}$.*

Note that circulant matrices are a special case of cyclic matrices.

Corollary 3 ([49]) *For order $k \leq 8$, there are at most 5 types of MDS cyclic matrices, namely cyclic matrices whose first row has:*

Type 0: *k distinct entries;*
Type 1: *1 pair of repeated entries;*
Type 2: *2 pairs of repeated entries;*
Type 3: *3 pairs of repeated entries;*
Type 4: *or 3 repeated entries.*

Definition 31 (Cauchy matrix) *Given $x_0, ..., x_{n-1}$ and $y_0, ..., y_{n-1}$, the matrix A where $A[i][j] = \frac{1}{x_i + y_j}$ is called a Cauchy matrix. According to [73], provided that $x_i \neq x_j$ for $0 \leq i, j \leq n-1$, that $y_i \neq y_j$ for $0 \leq i, j \leq n-1$ and that $x_i + y_j \neq 0$ for all i, j, any square submatrix of a Cauchy matrix is nonsingular over any field.*

It is worth noting that a Cauchy matrix construction directly ensures the MDS property, as can be seen in Definition 31, and it requires only $2n$ choices of elements, since the x_i and the y_i must be defined.

In [19], a specific type of Cauchy matrices is introduced — compact Cauchy matrices — as well as a method to obtain involutory matrices of this kind.

Definition 32 (Compact Cauchy matrix [19]) *Let A_X be an $n \times n$ Cauchy matrix. If A_X precisely has n distinct entries, we call A_X a* compact Cauchy matrix.

Theorem 4 (Building involutory compact Cauchy matrices [19]) *Let $A = (a_{i,j})_{2^n \times 2^n}$ be a compact Cauchy matrix over $GF(2^m)$ and $\mu = (\oplus_{k=0}^{2^n-1} a_{0,k})^{-1}$. Then μA is an involution compact Cauchy matrix.*

In other words, it is possible to multiply a compact Cauchy matrix A by a constant μ to make it involutory, where μ is the inverse of the sum of the entries of a row of A.

Definition 33 (Hadamard matrix [6]) *Given n elements $a_0, a_1, ..., a_{n-1}$, n being a power of 2, the matrix H such that $H[i][j] = a_{i \oplus j}$ is a Hadamard matrix.*

In Whirlwind[2], the authors make use of *dyadic* matrices, defining them as a matrix S such that $S[i][j] = s_{i \oplus j}$ given a set $s_0, s_1, ..., s_{n-1}$ of n elements. This definition matches Definition 33, which is used in Anubis and Khazad, but referred as Hadamard instead of dyadic. Therefore, for dyadic matrices, the same situation described for Hadamard applies: a dyadic construction allows us to reduce the scope of the element choice, i.e only n elements must be chosen, but there is no guarantee of MDS property. In [36], they call this type of matrices FFHadamard, but throughout this work we refer to them as Hadamard.

Note that a Hadamard construction, similarly to a circulant construction, requires only n elements to be defined, but it does not guarantee MDS property either. To the best of our knowledge, no proofs about Hadamard matrices being MDS have been presented in the literature at the time of writing. Furthermore, in this work, we present an example of a Hadamard non-MDS matrix. See Section 7.10 for further detail on this.

In [70], the authors explain how to obtain Hadamard involutory matrices. If $H = had(h_0, h_1, \ldots, h_{k-1})$ is a Hadamard matrix, then $H \times H = c^2 \cdot I$, with $c^2 = h_0^2 + h_1^2 + h_2^2 + \ldots + h_{k-1}^2$. In other words, the product of a Hadamard matrix with itself is a multiple of an identity matrix, where the multiple c^2 is the sum of the square of the elements from the first row.

Therefore, a Hadamard matrix is involutory if the sum of the elements of the first row is equal to 1. Note that if we deal with a Hadamard matrix for which the sum of the first row is nonzero, we can make it involutory by dividing it by the sum of its first row.

Definition 34 (Vandermonde matrix [37]) *Given z elements $a_0, a_1, ..., a_{z-1}$, the $z \times n$ matrix V such that*

$$V = \begin{bmatrix} 1 & a_0 & a_0^2 & ... & a_0^{n-1} \\ 1 & a_1 & a_1^2 & ... & a_1^{n-1} \\ ... & ... & ... & ... & ... \\ 1 & a_{z-1} & a_{z-1}^2 & ... & a_{z-1}^{n-1} \end{bmatrix} \tag{7.1}$$

is a Vandermonde matrix. We denote Vandermonde matrices as $vand(a_0, ..., a_{z-1})$.

Similarly to circulant and Hadamard constructions, in the case of the Vandermonde matrix, z elements are required and then we must choose the number of columns n. Not all Vandermonde matrices are square matrices, but the notion of MDS matrix applies only to square matrices. Therefore, z must be equal to n if we wish to construct a Vandermonde MDS matrix. However, again there are no guarantees about the MDS property with a Vandermonde construction. To the best of our knowledge, at the time of writing, there are no such guarantees. The matrices must be constructed and then evaluated with respect to the MDS property, e.g using Theorem 2.

In the PHOTON hash function [33] and the LED block cipher [34], *serial* MDS matrices are used. Serial matrices are especially useful for hardware implementations. In [35], they refer to serial matrices as *recursive* matrices, and show how to obtain them from companion matrices (see Theorem 5).

Definition 35 (Companion matrix [35]) *Let $g(x) = a_0 + a_1x + ... + a_{k-1}x^{k-1} + x^k$ be a monic polynomial over $\mathbb{F}_q$ of degree k. The* companion matrix C_g *associated to the polynomial g is given by*

$$C_g = \begin{bmatrix} 0 & 1 & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & ... & 1 \\ -a_0 & -a_1 & ... & ... & -a_{k-1} \end{bmatrix}.$$

In other words, it is a matrix with the negative coefficients of the polynomial in the last row, and a diagonal of ones above the main diagonal.

Note that, in the case of matrix composed by elements of $GF(2^4)$ and $GF(2^8)$, as is the case of the studied companion matrices in this work, we can ignore the negative sign. We denote by $Companion(a_0, a_1, ..., a_{k-1})$ the companion matrix with $a_0, a_1, ..., a_{k-1}$ in the last row.

In [35], a formula for the inverse of a companion matrix is given. If $a_0 \neq 0$, a companion matrix C_g is nonsingular and its inverse is given by

$$C_g^{-}1 = \begin{bmatrix} \frac{-a_1}{a_0} & \frac{-a_2}{a_0} & ... & \frac{-a_{k-1}}{a_0} & \frac{-1}{a_0} \\ 1 & 0 & ... & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & ... & 1 & 0 \end{bmatrix}.$$

A recursive MDS matrix, according to [35], is an MDS matrix which can be computed as a power of a simple companion matrix, i.e, an MDS matrix $M = C_g^k$ for some companion matrix corresponding to a polynomial $g(x) \in \mathbb{F}_q[x]$ of degree k. We then say that the polynomial $g(x)$ yields a recursive MDS matrix.

Theorem 5 (Obtaining a serial MDS matrix from a companion matrix [35]) *Let $g(x) \in \mathbb{F}_q[x]$ be a polynomial of degree k. Then the matrix $M = C_g^k$ is MDS if and only if the polynomial $g(x)$ has no multiple with weight $\leq k$ and degree $\leq 2k - 1$.*

In [36], the authors introduce a way of obtaining an MDS matrix from two $n \times n$ Vandermonde matrices A and B.

Lemma 2 (Building MDS matrix from two Vandermonde matrices [36]) *For distinct x_0, x_1, $\ldots$, x_{n-1} and y_0, y_1, $\ldots$, y_{n-1} such that $x_i + y_j \neq 0$, the matrix AB^{-1} is an MDS matrix, where $A = vand(x_0, \ldots, x_{n-1})$ and $B = vand(y_0, \ldots, y_{n-1})$.*

The construction of Lemma 2 is also mentioned in [35], but as a theorem.

Theorem 6 ([35]) *Let $V_1 = vand(a_0, a_1, ..., a_{n-1})$ and $V_2 = vand(b_0, b_1, ..., b_{n-1})$ be two Vandermonde matrices such that a_i, b_j are $2n$ distinct elements from some field. Then the matrices $V_1^{-1}V_2$ and $V_2^{-1}V_1$ are such that any square submatrix of them is nonsingular and hence MDS matrices.*

Corollary 4 ([35]) *If $V_1 = vand(a_0, a_1, ..., a_{n-1})$ and $V_2 = vand(b_0, b_1, ..., b_{n-1})$ are two invertible Vandermonde matrices in the field $\mathbb{F}_{2^r}$, satisfying the two properties $a_i = l + b_i$ and $a_i \neq b_j$, for $0 \leq i, j \leq n-1$, where l is a non-zero arbitrary element in $\mathbb{F}_{2^r}$, then $V_1^{-1}V_2$ is an involutory MDS matrix.*

Lemma 3 (Hadamard-Cauchy construction [36], [70]) *Let $G = (x_0, x_1, \ldots, x_{d-1})$ be an additive subgroup of $\mathbb{F}_{2^m}$. Let us consider the coset $r + G, r \notin G$ of G having elements $y_j = r + x_j, j = 0, \ldots, d-1$. Then the $d \times d$ matrix $A = ((a_{i,j}))$, where $a_{i,j} = \frac{1}{x_i + y_j}$, for all $0 \leq i, j \leq d-1$ is an MDS matrix.*

Corollary 5 ([36]) *The matrix A of Lemma 3 is symmetric.*

In [70], the construction from Lemma 3 is called Hadamard-Cauchy, whilst [36] presents it but does not give it a specific name. Note that, for this kind of matrix, a matrix of order d depends on d elements to be defined (like a Hadamard matrix), but the coefficients are computed in the same way we would for a Cauchy construction. Furthermore, it is symmetric ($h_{i,j} = h_{j,i}$), and it depends only on the first row $(x_0, x_1, \ldots, x_{d-1})$ with elements characterized by $h_{i,j} = x_{i \oplus j}$. Hadamard-Cauchy matrices are denoted by $hc(x_0, x_1, \ldots, x_{d-1})$. Also, note that Lemma 4 and Corollary 6 give us a way of building involutory Hadamard-Cauchy matrices.

Lemma 4 ([36]) *Let $A = ((a_{i,j}))$ be the $d \times d$ matrix formed by Lemma 3. Then $A^2 = c^2 I$, where $c = \sum_{k=0}^{d-1} \frac{1}{r + x_k}$.*

Corollary 6 ([36]) *The matrix A of Lemma 3 is involutory if the sum of the elements of any row is 1.*

Definition 36 ([35]) *The $n \times n$ matrix*

$$H = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{n-2} & a_{n-1} \\ a_1 & a_2 & a_3 & \cdots & a_{n-1} & a_n \\ a_2 & a_3 & a_4 & \cdots & a_n & a_{n+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n-1} & a_n & a_{n+1} & \cdots & a_{2n-3} & a_{2n-2} \end{bmatrix}$$

is called a Hankel matrix.

Note that a Hankel matrix is defined by its first row and last column, i.e the elements a_0, a_1, a_2, ..., a_{n-1}, a_n, a_{n+1}, ..., a_{2n-2}. Also, $H[i][j] = a_{i+j}$ and, since a Hankel matrix is symmetric, an involutory (orthogonal) Hankel matrix is orthogonal (involutory). We denote Hankel matrices as $Hank(a_0, ..., a_{2n-2})$.

Theorem 7 ([35]) *For $n \geq 2$, if H is a $2^n \times 2^n$ Hankel MDS matrix over $\mathbb{F}_{2^r}$, then H is not involutory (orthogonal).*

Although the $2^n \times 2^n$ Hankel MDS matrices are not involutory, the Hankel MDS matrices of other orders may be involutory. For example [35], over $\mathbb{F}_{2^8}$ with irreducible polynomial $x^8 + x^6 + x^5 + x^2 + 1$, the 5×5 matrix $Hank(1, \alpha, \alpha^7 + \alpha^5 + \alpha^4 + \alpha + 1, \alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha + 1, \alpha^3 + \alpha, 1, \alpha, \alpha^7 + \alpha^5 + \alpha^4 + \alpha + 1, \alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha + 1)$ is involutory and 6×6 matrix $Hank(1, 1, \alpha^7 + \alpha^5 + \alpha^4 + \alpha + 1, \alpha^5 + \alpha^3 + \alpha^2, \alpha^2, \alpha^7 + \alpha^4 + \alpha^3 + \alpha, 1, 1, \alpha^7 + \alpha^5 + \alpha^4 + \alpha + 1, \alpha^5 + \alpha^3 + \alpha^2, \alpha^2)$ is involutory.

Theorem 8 ([35]) *Suppose $q = 2^r$ and γ is an arbitrary primitive element of the field $\mathbb{F}_q$. Let S_q be a triangular array whose coefficients are constants along skew diagonal in a Hankel matrix fashion defined as*

$$S_q = \begin{matrix} a_1 & a_2 & a_3 & \dots & a_{q-3} & a_{q-2} \\ a_2 & a_3 & a_4 & \dots & a_{q-2} & \\ a_3 & a_4 & \dots & a_{q-2} & & \\ a_4 & \dots & & & & \\ \vdots & \vdots & & & & \\ a_{q-3} & a_{q-2} & & & & \\ a_{q-2} & & & & & \end{matrix}$$

where $a_i = (1-\gamma^i)^{-1}$ for $1 \leq i \leq q-2$. Then every square submatrix of S_q is nonsingular and hence MDS.

Lemma 5 ([35]) *Circulant involutory matrices of order $n \geq 3$ over $\mathbb{F}_{2^r}$ are not MDS.*

Theorem 9 ([35], [50]) *For $n \geq 2$, if L s a $2^n \times 2^n$ left-circulant MDS matrix over $\mathbb{F}_{2^r}$, then L is not involutory (orthogonal).*

It is relevant to note that, although left-circulant matrices can not be involutory if their order is a power of two, left-circulant matrices of other orders may be involutory. In [35], they show left-circulant involutory MDS matrices of order 5 and 6 over $\mathbb{F}_{2^8}$ with irreducible polynomial $x^8 + x^6 + x^5 + x^2 + 1$. The 5×5 matrix is $lcirc(1, \alpha, \alpha^7 + \alpha^5 + \alpha^4 + \alpha + 1, \alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha + 1, \alpha^3 + \alpha)$. The 6×6 matrix is $lcirc(1, 1, \alpha^7 + \alpha^5 + \alpha^4 + \alpha + 1, \alpha^5 + \alpha^3 + \alpha^2, \alpha^2, \alpha^7 + \alpha^4 + \alpha^3 + \alpha)$.

Definition 37 ([35]) *The $n \times n$ matrix*

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & \dots & a_{n-2} & a_{n-1} \\ a_{-1} & a_0 & a_1 & \dots & a_{n-3} & a_{n-2} \\ a_{-2} & a_{-1} & a_0 & \dots & a_{n-4} & a_{n-3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{-(n-1)} & a_{-(n-2)} & a_{-(n-3)} & \dots & a_{-1} & a_0 \end{bmatrix}$$

is called a Toeplitz matrix of order n. In [63], they define a Toeplitz matrix as a matrix where every descending diagonal from left to right is constant. Their definition is analogous to this one.

In [63], it is mentioned that Toeplitz matrices cannot be involutory. However, in a Toeplitz matrix, several submatrices repeat, and one can take advantage of this redundancy while checking the MDS property, making matrix evaluation faster.

A Toeplitz matrix is defined by its first row and first column, for instance, (a_0, a_1, ..., a_{n-1}, a_{-1}, a_{-2}, ..., $a_{-(n-1)}$). The (i,j)-th entry of A can be expressed as $A[i][j] = a_{(j-i)}$. We denote Toeplitz matrices as $Toep(a_0, a_1, ..., a_{n-1}, a_{-1}, ..., a_{-(n-1)})$.

Theorem 10 ([35], [63]) *Toeplitz matrices of order $n \geq 3$ over $\mathbb{F}_{2^r}$ cannot be both MDS and involutory.*

Theorem 11 ([35], [63]) *For $n \geq 2$, any $2^n \times 2^n$ Toeplitz orthogonal matrix over $\mathbb{F}_{2^r}$ is not MDS.*

It is relevant to note that, although the $2^n \times 2^n$ Toeplitz matrices are not orthogonal, orthogonal Toeplitz matrices of other orders, i.e orders that are not powers of two, may exist. In [35], they show examples with $\mathbb{F}_{2^8}$ and the constructing polynomial $x^8 + x^4 + x^3 + x + 1$. The 3×3 Toeplitz matrix $Toep(\alpha, 1 + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^6, \alpha + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^6, \alpha + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^6, 1 + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^6)$ is orthogonal and MDS, as well as the 6×6 matrix $Toep(1, 1, \alpha, 1 + \alpha^2 + \alpha^3 + \alpha^5 + \alpha^6 + \alpha^7, \alpha + \alpha^5, \alpha^2 + \alpha^3 + \alpha^6 + \alpha^7, \alpha^2 + \alpha^3 + \alpha^6 + \alpha^7, \alpha + \alpha^5, 1 + \alpha^2 + \alpha^3 + \alpha^5 + \alpha^6 + \alpha^7, \alpha, 1)$.

A summary on the types of matrices

In summary, we mentioned the following matrix types: circulant, Cauchy, compact-Cauchy, Hadamard, Hadamard-Cauchy, Vandermonde, serial, Hankel and Toeplitz. All these types have the advantage of reducing the number of elements that must be selected when constructing a matrix. An $n \times n$ matrix possesses n^2 elements and thus, when constructing one e.g by selecting random elements, n^2 choices must be made. However, the presented matrix constructions allow us to lower the number of elements we need to select. For instance, for circulant and Hadamard we would only need to choose the first row, comprised of n elements.

The Cauchy construction ensures MDS property, the serial construction also ensures MDS property if Theorem 5 is satisfied, whilst the other constructions do not and e.g Theorem 2 must be used to evaluate MDS property. Furthermore, it is relevant to note that, albeit most matrices in this work have their dimension n be a power of two such as 4 or 8, this is not a requirement for the construction. One can construct $n \times n$ circulant (or Cauchy, or Vandermonde, or serial) matrices for any arbitrary n. The dimension n must only be a power of two for the Hadamard (dyadic) construction.

There is a great interest in building involutory MDS matrices such as e.g the ones employed in ANUBIS [3] and KHAZAD [4]. However, although their constructions are Hadamard, it is relevant to note that a Hadamard construction does not imply involutory property. As an example, the matrices M_0 and M_1 used in Whirlwind [2] are Hadamard (dyadic), but not involutory (see Table 7.4). Furthermore, [73] has shown how to obtain an involutory MDS matrix from a Cauchy construction. However, a Cauchy matrix itself is not necessarily involutory. In order to obtain an involutory matrix, additional manipulation on the coefficients is necessary. For further detail the reader may refer to [73]. Other constructions besides the Cauchy can be used to obtain involutions, too. Table 7.1 summarizes the matrix types and their advantages.

Matrix type	**Element selection requeriments**	**MDS property guarantee**	**Involutory property guarantee**	**Other aspects**
Circulant	n	no	no	—
Hadamard (a.k.a dyadic, FFHadamard)	n	no	yes, as explained in [70]	—
Vandermonde	z, but for square matrices, $z = n$, so, n	yes, with the product of two Vandermonde matrices (see Lemma 2)	no	—
Cauchy	$2n$	yes	can be made involutory with additional coefficient manipulation, as in [73]	—
serial (a.k.a recursive)	n, last row contains the k coefficients of the companion matrix polynomial ($k = n$)	yes, with Theorem 5	no	good for hardware implementation due to representation as powers of companion matrices
compact-Cauchy	$2n$ (Cauchy construction requirements)	guaranteed by Cauchy construction	yes, with Theorem 4	—
Hadamard-Cauchy	$2n$ (Cauchy construction requirements)	guaranteed by Cauchy construction	yes, with Corollary 6	is also symmetric
Hankel	$2n - 1$ (first row and first column)	no	no	—
Toeplitz	$2n - 1$ (first row and first column)	no	no	repetition of submatrices can speed up MDS property evaluation
Random matrix without any special construction	n^2	no	no	—

Table 7.1: Matrix classification summary

7.4 Computing xtime and xor of the matrices

7.4.1 Computational cost unit

Computational cost of multiplication in GF(2^8)

Consider T a state byte, which we multiply by the polynomial $2e_x = 00101110_2 = x^5 + x^3 + x^2 + x$ in GF(2^8). Note that

$$T \cdot 2e_x = T \cdot x^5 + T \cdot x^3 + T \cdot x^2 + T \cdot x = T \cdot x \cdot x \cdot x \cdot x \cdot x + T \cdot x \cdot x \cdot x + T \cdot x \cdot x + T \cdot x,$$

where $\cdot$ denotes multiplication and $+$ denotes addition (which, in GF(2^8), is equivalent to a bitwise XOR). Multiplication by the x polynomial is performed by **xtime**, and addition is performed by **xor**.

Let $T \cdot x = Y$. Then $T \cdot 2e_x = Y + Y \cdot x + Y \cdot x \cdot x + Y \cdot x \cdot x \cdot x \cdot x$.

Let $Y \cdot x = W$. Then $T \cdot 2e_x = Y + W + W \cdot x + W \cdot x \cdot x \cdot x$.

Let $W \cdot x = Z$. Then $T \cdot 2e_x = Y + W + Z + Z \cdot x \cdot x$.

The total number of **xtime** operations in this process is 5 (1 to obtain Y from T, 1 to obtain W from Y, 1 to obtain Z from W, 2 to compute $Z \cdot x \cdot x$), since we can reuse intermediate **xtime** calls. The total number of **xor** operations is 3. For multiplication in GF(2^8), in the worst case, 7 **xtime** would be necessary, since the maximum degree of polynomials in GF(2^8) is 7.

Computational cost of a matrix

The computational cost of an $n \times n$ matrix A is given by the necessary **xor** and **xtime** operations when multiplying a $n \times 1$ column vector by A. As an example, we calculate the cost of matrix (7.49), used in the SQUARE [20] cipher.

A row of matrix (7.49) contains the elements $01_x = 1, 02_x = x$ and $03_x = x + 1$ only. Multiplying by 01_x does not require **xtime** or **xor**, since $01_x \cdot T = T$. Computing $02_x \cdot T = x \cdot T$ requires 1 **xtime**. Computing $03_x \cdot T = (x + 1) \cdot T = T \cdot x + T$ requires 1 **xtime** and 1 **xor**. Furthermore, adding the row multiplication results costs 3 **xor**. Therefore, the cost of a row is 2 **xtime** and 4 **xor**. Equation 7.2 illustrates this, with t_1, t_2, t_3 and t_4 being bytes of the state column vector.

$$\begin{bmatrix} 02_x & 01_x & 01_x & 03_x \end{bmatrix} \cdot \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{bmatrix} = 02_x \cdot t_1 + 01_x \cdot t_2 + 01_x \cdot t_3 + 03_x \cdot t_4. \tag{7.2}$$

Note that matrix (7.49) contains 4 rows, yielding a total cost of 8 **xtime** and 16 **xor**. The computational cost for matrix (7.47), used in the SQUARE cipher, was explained in Section 7.4.1. Here we present an example for (7.50), to illustrate how the cost is computed in this work.

For matrix (7.50), used in SQUARE's decryption process, each row contains elements from $\{0e_x, 09_x, 0d_x, 0b_x\}$.

$0e_x = 00001110_2 = x^3 + x^2 + x$ requires 3 **xtime** and 2 **xor**

$09_x = 00001001_2 = x^3 + 1$ requires 3 **xtime** and 1 **xor**

$0d_x = 00001101_2 = x^3 + x^2 + 1$ requires 3 **xtime** and 2 **xor**

$0b_x = 00001011_2 = x^3 + x + 1$ requires 3 **xtime** and 2 **xor**

There are 3 **xor** to add the intermediate row multiplication results, totalizing 12 **xtime** and 10 **xor** per row. There are 4 rows, hence 48 **xtime** and 40 **xor**. In this case, since all rows are composed of the same elements due to the circulant property of SQUARE's matrix, we can simply multiply the cost of a single row by the amount of rows. However, for e.g SHARK's matrix, we have to compute the cost of each row separately. The code in Listings 1 and 2 were used to compute the costs of each matrix present in the catalogue of Table 7.4.

7.5 MDS Matrix Catalogue

Table 7.4 is an extensive catalogue of MDS matrices found in the literature, showing the year of their discovery, the dimensions (**Dim.** column), whether they are involutory of not (**Invol.** column), their application in e.g block ciphers or hash functions, their bibliographic reference (**Ref.**), finite field (**FF**), irreducible polynomial adopted for the finite field (**Irred. Poly**), count of **xor** and **xtime** required for multiplication (**#xor** and **#xtime** columns), count of **xor** and **xtime** for their inverses (**Inv. #xor** and **Inv.#xtime** columns) and, finally, references to the matrix and its inverse themselves in this document (columns **Mat.** and **Inv.**).

Other remarks about the catalogue:

– All finite fields of Table 7.4 have characteristic $p = 2$.

– See Section 7.9 for further detail on the MDS matrices from C_0 to C_{13} found by Shirai and Shibutani and their relationship with the Whirlpool hash function.

– Since it is relevant to know which ones are the least expensive matrices, we present in Table 7.2 the cheapest matrices per dimension, among all the matrices studied in this work.

In Table 7.5, we present data on the matrices of works [7] by Beierle and [28] by Duwal. These works go towards finding **generic** MDS matrices, i.e MDS matrices that remain MDS for multiple finite fields satisfying some restrictions. For example, Beierle's matrices are parameterized with a value α and the finite field power m. In Duwal's work, they do not specify the irreducible polynomials, thus resulting in multiple possibilities for the inverses of the matrices they list. For this reason we do not have **xtime** and **xor** metrics for Beierle's matrices, since they will depend on how the matrix is instanced with α, and, for Duwal's matrices, we do not have **xtime** and **xor** of the inverses, since there are multiple possible inverses.

We did instantiate Beierle's and Duwal's matrices to extract cost metrics of actual instances as an experiment, but also because we would need to take them into account when computing what are the least computationally expensive matrices found so far in the literature. The metrics for our instances are in Tables 7.13 and 7.14, and the cheapest instances of Beierle's matrices are listed in Table 7.3. With regards to Duwal's matrices, the cheapest for $n = 3$ is (7.188) and, for $n = 4$, (7.194), when considering only encryption cost.

Furthermore, about Duwal's matrices, analyzing Table 7.14, we can see that the costs of the inverses remain invariant with respect to the chosen polynomial within a finite field of the same order. For example, all inverses when the finite field is $GF(2^4)$ have the same costs despite the different choices for the irreducible polynomial, and the same happens for $GF(2^8)$ and $GF(2^2)$. Due to that, the cheapest matrix overall, when considering decryption cost too (for decryption, the costs of the inverse are taken into account), will be the one in the smallest finite field. Matrices in $GF(2^2)$ will have cheaper inverses than the ones in $GF(2^4)$, and matrices in $GF(2^4)$ will have cheaper inverses than the ones in $GF(2^8)$. For example, matrix 7.188 has invariant encryption cost, independently of the finite field, but has its cheapest inverse when working in $GF(2^2)$, and the same occurs for $n = 4$ (matrix 7.194) and $GF(2^4)$. This happens because the coefficient values are limited to 3 for $GF(2^2)$, 15 for $GF(2^4)$ and 255 for $GF(2^8)$, leading to smaller **xtime** counts for the smaller fields.

Dim.	**Mat.**	**#xtime**	**#xor**	**Inv. #xtime**	**Inv. #xor**	**3:1**	**Total #xtime**	**Inv. 3:1**	**Total 3:1**
2	(7.200)	2	2	10	8	8	12	38	46
3	(7.145)	3	6	57	45	15	60	216	231
4	(7.49)	8	16	48	40	40	56	184	224
5	(7.167) $\star$	30	30	30	30	120	60	120	240
6	(7.81)	59	59	74	74	236	133	296	532
7	(7.83)	96	96	104	104	384	200	416	800
8	(7.212)	72	80	360	224	296	432	1304	1600
16	(7.98) $\star$	1248	800	1248	800	4544	2496	4544	9088
32	(7.99) $\star$	5440	3712	5440	3712	20032	10880	20032	40064

Table 7.2: Matrices with smallest computational cost for each dimension. **3:1** refers to the weighted sum $3 \cdot \text{xtime} + \text{xor}$, and **Total** refers to sum for the matrix and the inverse, showing what would be the total cost for encryption and decryption in an algorithm that employed the matrix in its diffusion layer. The $\star$ symbol means the matrix is involutory.

Dim.	**Mat.**	**#xtime**	**#xor**	**Inv. #xtime**	**Inv. #xor**	**3:1**	**Total #xtime**	**Inv. 3:1**	**Total 3:1**
2	(7.200) with $\alpha = 2$	2	2	10	8	8	12	38	46
3	(7.202) with $\alpha = 2$	3	6	21	21	15	24	84	99
4	(7.204) with $\alpha = 2$	16	20	24	24	68	40	96	164
5	(7.205) with $\alpha = 2$	40	35	140	100	155	180	520	675
6	(7.206) with $\alpha = 71$	102	66	216	132	372	318	780	1152
7	(7.207) with $\alpha = 2$	112	84	182	147	420	294	693	1113
8	(7.208) with $\alpha = 30$	248	184	336	216	928	584	1224	2152

Table 7.3: Matrices with smallest computational cost for each dimension, from the Beierle instances. Each columns has the same meaning of Table 7.2.

Year	Dim.	Type	Invol.	Usage	Ref.	FF	Irred. Poly.	#xor	Inv. #xor	#xtime	Inv. #xtime	Mat.	Inv.
1996	8	—	no	SHARK	[62]	$GF(2^8)$	$x^8+x^7+x^6+x^5+x^4+x^2+1$	235	223	369	393	(7.47)	(7.48)
1997	4	right circulant	no	SQUARE	[20]	$GF(2^8)$	$x^8+x^7+x^6+x^5+x^4+x^2+1$	16	40	8	48	(7.49)	(7.50)
1997	8	Cauchy	yes	—	[73]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	240	—	344	—	(7.51)	—
1998	3	right circulant	no	BKSQ	[21]	$GF(2^8)$	$x^8+x^7+x^6+x^5+x^4+x^2+1$	9	39	9	63	(7.58)	(7.59)
1999	4	right circulant	no	Rijndael (AES)	[25]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	16	40	8	48	(7.56)	(7.57)
2000	8	Hadamard	yes	KHAZAD	[4]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	112	—	120	—	(7.52)	—
2000	4	Hadamard	yes	ANUBIS	[3]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	16	—	20	—	(7.53)	—
2000	4	Vandermonde	no	ANUBIS key schedule	[3]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	20	69	32	101	(7.54)	(7.55)
2000	4	right circulant	no	Hierocrypt-3 and Hierocrypt-L1	[17], [18]	$GF(2^8)$	$x^8+x^6+x^5+x+1$	52	52	108	104	(7.60)	(7.61)
2000	4	right circulant	no	Hierocrypt-3	[17]	$GF(2^4)$	x^4+x+1	32	40	40	44	(7.62)	(7.63)
2000	2	—	no	Hierocrypt-L1	[18]	$GF(2^4)$	x^4+x+1	8	7	10	11	(7.64)	(7.65)
2003	8	right circulant	no	Shirai C_0	[68]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	72	240	88	400	(7.209)	(7.223)
2003	8	right circulant	no	Shirai C_1	[68]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	80	288	80	424	(7.210)	(7.224)
2003	8	right circulant	no	Shirai C_2	[68]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	72	224	88	360	(7.211)	(7.225)
2003	8	right circulant	no	Shirai C_3	[68]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	80	224	72	360	(7.212)	(7.226)
2003	8	right circulant	no	Shirai C_4	[68]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	80	240	88	424	(7.213)	(7.227)
2003	8	right circulant	no	Shirai C_5	[68]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	88	224	80	424	(7.214)	(7.228)
2003	8	right circulant	no	Shirai C_6	[68]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	80	256	88	416	(7.215)	(7.229)
2003	8	right circulant	no	Shirai C_7 (adopted for Whirlpool standard ISO/IEC 10118-3)	[68]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	72	224	88	360	(7.216)	(7.230)
2003	8	right circulant	no	Shirai C_8	[68]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	80	224	72	360	(7.217)	(7.231)
2003	8	right circulant	no	Shirai C_9	[68]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	80	256	88	416	(7.218)	(7.232)
2003	8	right circulant	no	Shirai C_{10}	[68]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	88	224	80	424	(7.219)	(7.233)
2003	8	right circulant	no	Shirai C_{11}	[68]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	80	240	88	424	(7.220)	(7.234)
2003	8	right circulant	no	Shirai C_{12}	[68]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	80	288	80	424	(7.221)	(7.235)
2003	8	right circulant	no	Shirai C_{13}	[68]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	72	240	88	400	(7.222)	(7.236)
2004	4	—	no	FOX	[41]	$GF(2^8)$	$x^8+x^7+x^6+x^5+x^4+x^3+1$	30	72	25	106	(7.66)	(7.68)

Year	Dim.	Type	Invol.	Usage	Ref.	FF	Irred. Poly.	#xor	Inv. #xor	#xtime	Inv. #xtime	Mat.	Inv.
2004	8	—	no	FOX	[41]	$GF(2^8)$	$x^8+x^7+x^6+x^5+x^4+x^3+1$	141	284	169	392	(7.67)	(7.69)
2007	3	—	yes	Curupira	[5]	$GF(2^8)$	$x^8+x^6+x^3+x^2+1$	12	—	15	—	(7.70)	—
2007	3	right circulant	no	Curupira key schedule	[5]	$GF(2^8)$	$x^8+x^6+x^3+x^2+1$	27	30	36	36	(7.71)	(7.72)
2009	8	right circulant	no	Grostl	[32]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	104	232	96	376	(7.73)	(7.74)
2011	4	serial	no	PHOTON	[33]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	28	55	27	39	(7.77)	(7.78)
2011	5	serial	no	PHOTON	[33]	$GF(2^4)$	x^4+x+1	44	44	52	52	(7.79)	(7.80)
2011	6	serial	no	PHOTON	[33]	$GF(2^4)$	x^4+x+1	59	74	59	74	(7.81)	(7.82)
2011	7	serial	no	PHOTON	[33]	$GF(2^4)$	x^4+x+1	96	104	96	104	(7.83)	(7.84)
2011	8	serial	no	PHOTON	[33]	$GF(2^4)$	x^4+x+1	125	136	143	143	(7.85)	(7.86)
2011	6	serial	no	PHOTON	[33]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	108	163	126	207	(7.87)	(7.88)
2012	4	serial	no	LED	[34]	$GF(2^4)$	x^4+x+1	26	29	33	39	(7.75)	(7.76)
2013	4	Hadamard-Cauchy	no	—	[36]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	24	64	36	100	(7.91)	(7.92)
2013	4	Hadamard-Cauchy	yes	—	[36]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	56	—	80	—	(7.93)	—
2013	3	Hadamard-Cauchy	no	—	[36]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	32	34	46	53	(7.94)	(7.95)
2013	4	Hadamard-Cauchy	yes	—	[36]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	56	—	60	—	(7.96)	—
2013	8	Hadamard-Cauchy	yes	—	[36]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	176	—	288	—	(7.97)	—
2013	16	Hadamard-Cauchy	yes	—	[36]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	800	—	1248	—	(7.98)	—
2013	32	Hadamard-Cauchy	yes	—	[36]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	3712	—	5440	—	(7.99)	—
2014	4	Hadamard	yes	Prost	[42], [70]	$GF(2^4)$	x^4+x+1	16	—	20	—	(7.118)	—
2015	4	Hadamard	yes	Joltik	[39], [70]	$GF(2^4)$	x^4+x+1	24	—	32	—	(7.89)	—
2015	4	compact Cauchy	yes	—	[19]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	24	—	40	—	(7.90)	—
2015	8	Hadamard-Cauchy	yes	—	[70]	$GF(2^4)$	x^4+x+1	112	—	144	—	(7.100)	—
2015	4	Hadamard	yes	—	[70]	$GF(2^8)$	$x^8+x^6+x^5+x^2+1$	32	—	60	—	(7.101)	—
2015	8	Hadamard	yes	—	[70]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	128	—	208	—	(7.102)	—
2015	8	Hadamard	yes	—	[70]	$GF(2^4)$	x^4+x+1	112	—	144	—	(7.103)	—
2015	16	Hadamard-Cauchy	yes	—	[70]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	832	—	1408	—	(7.104)	—
2015	32	Hadamard-Cauchy	yes	—	[70]	$GF(2^8)$	$x^8+x^6+x^5+x^2+1$	3264	—	5728	—	(7.105)	—
2015	4	Hadamard	no	—	[70]	$GF(2^4)$	x^4+x+1	16	28	36	40	(7.106)	(7.107)
2015	4	Hadamard	no	—	[70]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	20	64	40	96	(7.108)	(7.109)
2015	8	Hadamard	no	—	[70]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	128	216	224	352	(7.110)	(7.111)
2015	8	Hadamard	no	—	[70]	$GF(2^4)$	x^4+x+1	104	144	144	144	(7.112)	(7.113)
2015	16	Hadamard-Cauchy	no	—	[70]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	784	1232	1456	1600	(7.114)	(7.115)
2015	32	Hadamard-Cauchy	no	—	[70]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	3648	4608	5696	6336	(7.116)	(7.117)
2015	3	Cauchy	no	—	[35]	$GF(2^4)$	x^4+x+1	19	17	21	21	(7.119)	(7.120)
2015	3	Cauchy	no	—	[35]	$GF(2^4)$	x^4+x+1	10	14	12	16	(7.121)	(7.122)
2015	4	compact Cauchy	no	—	[35]	$GF(2^4)$	x^4+x+1	36	24	36	32	(7.123)	(7.124)
2015	4	compact Cauchy	yes	—	[35]	$GF(2^4)$	x^4+x+1	24	—	32	—	(7.125)	—
2015	3	product of Vandermonde	no	—	[35]	$GF(2^4)$	x^4+x+1	14	18	16	24	(7.126)	(7.127)
2015	3	product of Vandermonde	yes	—	[35]	$GF(2^4)$	x^4+x+1	20	—	24	—	(7.128)	—
2015	8	circulant	no	—	[35]	$GF(2^16)$	$x^{16}+x^5+x^3+x^2+1$	72	456	88	824	(7.129)	(7.130)
2015	2	circulant	orthogonal	—	[35]	$GF(2^4)$	x^4+x+1	4	4	4	4	(7.131)	(7.132)

Year	Dim.	Type	Invol.	Usage	Ref.	FF	Irred. Poly.	#xor	Inv. #xor	#xtime	Inv. #xtime	Mat.	Inv.
2015	3	circulant	orthogonal	—	[35]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	30	30	39	39	(7.133)	(7.134)
2015	6	circulant	orthogonal	—	[35]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	84	84	120	120	(7.135)	(7.136)
2015	5	left circulant	yes	—	[35]	$GF(2^8)$	$x^8+x^6+x^5+x^2+1$	70	—	90	—	(7.137)	—
2015	6	left circulant	yes	—	[35]	$GF(2^8)$	$x^8+x^6+x^5+x^2+1$	94	—	126	—	(7.138)	—
2015	3	Toeplitz	orthogonal	—	[35]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	30	—	39	—	(7.139)	—
2015	6	Toeplitz	orthogonal	—	[35]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	84	—	120	—	(7.141)	—
2015	5	Hankel	yes	—	[35]	$GF(2^8)$	$x^8+x^6+x^5+x^2+1$	70	—	90	—	(7.143)	—
2015	6	Hankel	yes	—	[35]	$GF(2^8)$	$x^8+x^6+x^5+x^2+1$	84	—	126	—	(7.144)	—
2016	3	left circulant	no	—	[50]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	6	45	3	57	(7.145)	(7.168)
2016	4	left circulant	no	—	[50]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	20	52	32	96	(7.146)	(7.169)
2016	5	left circulant	no	—	[50]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	30	50	45	110	(7.147)	(7.170)
2016	6	left circulant	no	—	[50]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	60	90	108	198	(7.148)	(7.171)
2016	7	left circulant	no	—	[50]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	70	182	126	273	(7.149)	(7.172)
2016	8	left circulant	no	—	[50]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	120	232	200	408	(7.150)	(7.173)
2016	3	left circulant	no	—	[50]	$GF(2^4)$	x^4+x+1	6	21	3	21	(7.151)	(7.174)
2016	4	left circulant	no	—	[50]	$GF(2^4)$	x^4+x+1	16	32	20	40	(7.152)	(7.175)
2016	5	left circulant	no	—	[50]	$GF(2^4)$	x^4+x+1	30	40	40	50	(7.153)	(7.176)
2016	6	left circulant	no	—	[50]	$GF(2^4)$	x^4+x+1	54	72	60	84	(7.154)	(7.177)
2016	8	left circulant	no	—	[50]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	96	320	176	432	(7.155)	(7.178)
2016	8	left circulant	no	—	[50]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	104	232	152	376	(7.156)	(7.179)
2016	8	left circulant	no	—	[50]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	120	192	208	384	(7.157)	(7.180)
2016	8	left circulant	no	—	[50]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	112	248	184	360	(7.158)	(7.181)
2016	8	left circulant	no	—	[50]	$GF(2^8)$	$x^8+x^4+x^3+x+1$	120	216	176	392	(7.159)	(7.182)
2016	8	left circulant	no	—	[50]	$GF(2^8)$	$x^8+x^4+x^3+x^2+1$	120	296	192	432	(7.160)	(7.183)
2016	3	left circulant	yes	—	[50]	$GF(2^8)$	$x^8+x^6+x^5+x^3+1$	24	—	45	—	(7.161)	—
2016	5	left circulant	yes	—	[50]	$GF(2^8)$	$x^8+x^6+x^5+x^2+1$	70	—	90	—	(7.162)	—
2016	6	left circulant	yes	—	[50]	$GF(2^8)$	$x^8+x^6+x^5+x^2+1$	84	—	126	—	(7.163)	—
2016	7	left circulant	yes	—	[50]	$GF(2^8)$	$x^8+x^6+x^5+x^2+1$	112	—	217	—	(7.164)	—
2016	7	left circulant	yes	—	[50]	$GF(2^8)$	$x^8+x^5+x^4+x^3+1$	98	—	189	—	(7.165)	—
2016	3	left circulant	yes	—	[50]	$GF(2^4)$	$x^4+x^3+x^2+x+1$	18	—	21	—	(7.166)	—
2016	5	left circulant	yes	—	[50]	$GF(2^4)$	x^4+x+1	30	—	30	—	(7.167)	—
2017	8	Toeplitz	no	—	[64]	$GF(2^4)$	x^4+x+1	94	120	120	134	(7.184)	(7.186)
2017	8	Toeplitz	no	—	[64]	$GF(2^8)$	$x^8+x^7+x^6+x+1$	122	227	198	359	(7.185)	(7.187)

Year	Dim.	Type	Invol.	Usage	Ref.	FF	Irred. Poly.	#xor	Inv. #xor	#xtime	Inv. #xtime	Mat.	Inv.

Table 7.4: MDS matrices: parameters, usage and cost

Year	Dim.	Type	Ref.	FF	#xor	Inv. #xor	#xtime	Inv. #xtime	Mat.	Inv.
2016	2	circulant	[7]	any $GF(2^m)$ for m and α satisfying conditions	will depend on the instance	will depend on the instance, field and chosen poly	will depend on the instance	will depend on the instance, field and chosen poly	(7.200)	(7.201)
2016	3	circulant	[7]	△	△	△	△	△	(7.202)	(7.203)
2016	4	circulant	[7]	△	△	△	△	△	(7.204)	—
2016	5	circulant	[7]	△	△	△	△	△	(7.205)	—
2016	6	circulant	[7]	△	△	△	△	△	(7.206)	—
2016	7	circulant	[7]	△	△	△	△	△	(7.207)	—
2016	8	circulant	[7]	△	△	△	△	△	(7.208)	—
2018	3	—	[28]	at least $GF(2^4)$ and $GF(2^8)$ according to our experiments	6	will depend on the chosen field and poly	3	will depend on the chosen field and poly	(7.188)	—
2018	3	—	[28]	△	9	△	9	△	(7.189)	—
2018	3	—	[28]	△	8	△	4	△	(7.190)	(7.191)
2018	3	—	[28]	△	8	△	4	△	(7.192)	—
2018	3	—	[28]	△	6	△	3	△	(7.193)	—
2018	4	—	[28]	△	18	△	16	△	(7.194)	—
2018	4	—	[28]	△	18	△	16	△	(7.195)	—
2018	4	—	[28]	△	22	△	18	△	(7.196)	—
2018	4	—	[28]	△	22	△	16	△	(7.197)	—
2018	4	—	[28]	△	20	△	14	△	(7.199)	—

Table 7.5: Beierle and Duwal's MDS matrices. The △ symbol means "same as above".

7.6 Our Cost Metric

In this work, we made the decision to evaluate the matrices using both **xtime** and **xor** as separate cost units, although some other authors, such as Khoo, Duwal, and Beierle [70, 28, 7], evaluate matrices based on the number of XOR gates required for hardware implementation.

Our inspiration for considering **xtime** and **xor** as separate units came from Rijndael's design proposal [22], where these operations are treated independently. We found it valuable to maintain that separation in our work. By treating **xtime** and **xor** as distinct cost units, we can capture the specific contributions and complexities of each operation.

It's important to note that the metric used by Khoo, Duwal, and Beierle [70, 28, 7] is primarily targeted at hardware implementations. In contrast, our metric is more general-purpose, encompassing a wider range of applications and scenarios. By considering both **xtime** and **xor** as separate units, we aim to provide a more versatile evaluation that can be applied to various contexts.

Furthermore, our choice of using separate cost units is particularly beneficial when working with distinct finite fields and different operations within those fields. Since **xtime** and **xor** operations can vary depending on the finite field being used, it becomes useful to consider them separately. This approach allows us to account for the unique characteristics and complexities of each operation within the specific finite field context.

When we compare two matrices and decide which one is best regarding computational cost, we have two additional approaches: consider the weighted sum of **xtime** and **xor**, with weight 3 for **xtime** and weight 1 for **xor** (`3:1-cost` metric), or take just the **xtime** since it is the most expensive operation (`xt-only-cost` metric). Thus when comparing two matrices M_1 and M_2, we choose the one with the smallest cost. Tables 7.3 and 7.2 are an example of this strategy, showing both cost metrics for the matrices. Furthermore, encryption costs consider only the given matrix A's **xtime** and **xor** counts, but, for total encryption + decryption cost, we sum the counts of A^{-1}.

With these strategies, the cost of encryption with a matrix A can be

$$\texttt{3:1-cost}(A) = 3 \times A\texttt{.xtime} + A\texttt{.xor}$$

or

$$\texttt{xt-only-cost}(A) = A\texttt{.xtime}.$$

The cost of decryption is

$$\texttt{3:1-cost}(A^{-1}) = 3 \times A^{-1}\texttt{.xtime} + A^{-}1\texttt{.xor}$$

or

$$\texttt{xt-only-cost}(A^{-1}) = A^{-1}\texttt{.xtime}.$$

And, the total cost, if we are to use A and A^{-1} in a block cipher for encryption and decryption, is

$$\texttt{3:1-cost}(A) + \texttt{3:1-cost}(A^{-1})$$

or

$$\texttt{xt-only-cost}(A) + \texttt{xt-only-cost}(A^{-1}).$$

The idea of assigning weight 3 to the **xtime** count is inspired by work [65].

7.7 Remarks on MDS property challenge in finite fields

7.7.1 Search Space Size

A relevant aspect of searching for matrices that satisfy a given property is certainly how many matrices from the entire search space can actually satisfy the property. In our case, the question becomes about the fraction of Maximum Distance Separable (MDS) matrices of order n within the entire space of matrices of the same order in $GF(2^m)$. We hypothesize that the fraction of MDS matrices in this context is exceedingly small. This can be attributed to the rarity of MDS matrices, as their construction necessitates the absence of singular square submatrices. We try to estimate the size of the search space in the analysis that follows. Here, we consider two variables: n and m.

Let us consider the cases of $m = 4$ and $m = 8$ for $GF(2^m)$ as examples. In general, the set encompassing all matrices of order n in $GF(2^m)$ has a cardinality of $(2^m)^{(n^2)} = 2^{(mn^2)}$. This is due to the fact that each of the n^2 elements in the matrix can be chosen from the 2^m elements available in $GF(2^m)$. To determine if a matrix of order n in $GF(2^m)$ is MDS, we may confine our analysis to matrices that contain non-zero elements exclusively. This reduction in the search space leads to a size of $(2^m - 1)^{(n^2)}$. As we examine the increasing values of n, we observe the following sizes for the search spaces in $GF(2^4)$ and $GF(2^8)$.

n	$(2^4 - 1)^{n^2}$	$(2^8 - 1)^{n^2}$
2	$15^4 = 50625$	$255^4 \approx 2^{31.97}$
3	$15^9 = 38443359375$	$255^9 \approx 2^{71.94}$
4	$15^{16} \approx 2^{62.51}$	$255^{16} \approx 2^{127.90}$ $\triangleleft$ beyond reach with current computational resources

Table 7.6: Search Spaces in GF(2^4) and GF(2^8)

For small values of n and m, such as $n = 2$ and $m = 4$ (and possibly for $m = 8$), an exhaustive search for MDS matrices appears feasible. However, as n increases, the computational complexity of the search becomes exponentially challenging.

7.7.2 MDS Property Evaluation

We choose to analyze the computational cost of evaluating a matrix for MDS property in terms of number of submatrices to be evaluated i.e number of determinants to be computed and checked for non-singularity and in total of multiplications performed between the matrix coefficients, i.e multiplications in the finite field $GF(2^m)$ of the matrix. Computing the determinants involves addition, subtraction and multiplication, but multiplication in $GF(2^m)$ is the most expensive among these operations and therefore the most relevant to estimate the cost of a MDS property evaluation.

Multiplications required to compute a determinant

The number of coefficient multiplications required to compute the determinant of a $z \times z$ matrix depends on the method used for computation.

For a general $z \times z$ matrix, the exact number of coefficient multiplications can be determined using the Laplace expansion or cofactor expansion method. The determinant of a $z \times z$ matrix can be calculated as the sum of z terms, each of which involves the product of one coefficient from each row and column.

The formula for the number of coefficient multiplications in the determinant calculation is given by:

$$\textbf{Number of coefficient multiplications} = z! \times (z-1)! \times (z-2)! \times \ldots \times 1!$$

This formula takes into account the number of terms in the expansion and the number of coefficient multiplications required for each term.

For example, for a 3×3 matrix ($z = 3$), the number of coefficient multiplications is:

$$\textbf{Number of coefficient multiplications} = 3! \times 2! \times 1! = 6 \times 2 \times 1 = 12$$

So, computing the determinant of a 3×3 matrix requires exactly 12 coefficient multiplications.

Similarly, for a 4×4 matrix ($z = 4$), the number of coefficient multiplications is:

$$\textbf{Number of coefficient multiplications} = 4! \times 3! \times 2! \times 1! = 24 \times 6 \times 2 \times 1 = 288$$

Hence, computing the determinant of a 4×4 matrix requires exactly 288 coefficient multiplications.

In general, for a $z \times z$ matrix, the number of coefficient multiplications is $z!$ times the product of factorials from $(z - 1)$ down to 1. It is important to note that this count represents the exact number of coefficient multiplications and does not take into account any potential optimizations or algorithms that exploit the matrix's properties or structure.

Amount of submatrices to be evaluated

The process of evaluating an $n \times n$ matrix matrix for MDS property through Theorem 2 involves computing the determinants of its submatrices. If we are to pick k rows from the n rows to be removed, the number of possibilities is $C(n,k) = n!\,/(k!\times(n-k)!\,)$.

Similarly, if we are to pick k columns, there are $C(n,k) = n!\,/(k!\times(n-k)!\,)$ possibilities as well. Since rows and columns are chosen independently, for each choice of rows all the choices of columns can be made, thus resulting in a total of $C(n,k)^2 = (n!\,/(k!\times(n-k)!\,)^2$ submatrices that require evaluation.

For MDS property evaluation, k ranges from 0, which corresponds to not removing any rows or columns at all, leaving the matrix as is, to $n-1$, where only single coefficients must be checked to be non-zero. So the final amount of submatrices to be evaluated is $\sum_{k=0}^{n-1} C(n,k)^2$. Tables 7.7 and 7.8 shows numeric values for these formulas for dimensions 2 to 10.

n	**Total Submatrices**	**Total Multiplications in $GF(2^m)$**	**Closest Power of 2**
2	5	2	2^1
3	19	14	2^4
4	69	302	2^8
5	251	34862	2^{15}
6	923	24918062	2^{25}
7	3431	125436246062	2^{37}
8	12869	5056710181206062	2^{52}
9	48619	1834938528961266006062	2^{71}
10	184755	6658608419043265483506006062	2^{92}
16	601080389	—	2^{300}
32	1832624140942590533	—	2^{1611}

Table 7.7: Numeric values for MDS property evaluation complexity

n	**# of submatrices per dimension**									
—	0	1	2	3	4	5	6	7	8	9
2	1	4	—	—	—	—	—	—	—	—
3	1	9	9	—	—	—	—	—	—	—
4	1	16	36	16	—	—	—	—	—	—
5	1	25	100	100	25	—	—	—	—	—
6	1	36	225	400	225	36	—	—	—	—
7	1	49	441	1225	1225	441	49	—	—	—
8	1	64	784	3136	4900	3136	784	64	—	—
9	1	81	1296	7056	15876	15876	7056	1296	81	—
10	1	100	2025	14400	44100	63504	44100	14400	2025	100

Table 7.8: Amount of submatrices obtained at each step of MDS evaluation

7.8 A note on Hierocrypt 3 and Hierocrypt-L1

In the Hierocrypt 3 and Hierocrypt-L1 ciphers, there are two different diffusion layers, referred as low level (mds_l) and high level (MDS_H) by the authors, since these ciphers follow a nested SPN structure (for more details please refer to [17] and [18]). Their high level diffusion layer is called MDS_H and is based on multiplication by matrices (7.62) (for Hierocrypt 3) and (7.64) (for Hierocrypt-L1). Supposedly, they are both MDS, as stated in the design rationale section of the ciphers' specification documents (see [17] and [18]). However, for the implementation, the MDS_H transformation can be equivalently expressed as multiplication by a 16×16 binary matrix (in Hierocrypt-3) which is *not MDS*, but yields the same result. For Hierocrypt-L1, a 8×8 non-MDS matrix is used. In [17], they state

> "$MDS(32,4)$ consists of eight parallel $MDS(4,4)$. When all $MDS(4,4)$ are the same, MDS_H is nothing but the combination of byte-wise XOR's and is expressed as 16×16 matrix."

In [18], they present a similar statement about the 8×8 matrix.

Matrix (7.3) is the binary matrix used for MDS_H in Hierocrypt 3. We can see that it is not MDS, since it has a zero element (see Theorem 2).

$$\begin{bmatrix} 1010101011011111 \\ 1101110111100111 \\ 1110111011110011 \\ 0101010110101110 \\ 1111101010101101 \\ 0111110111011110 \\ 0011111011101111 \\ 1110010101011010 \\ 1101111110101010 \\ 1110011111011101 \\ 1111001111101110 \\ 1010111001010101 \\ 1010110111111010 \\ 1101111001111101 \\ 1110111100111110 \\ 0101101011100101 \end{bmatrix} \tag{7.3}$$

Matrix (7.4) is also not MDS, since it has a zero element. It is used for MDS_H in Hierocrypt L1.

$$\begin{bmatrix} 10101110 \\ 11011111 \\ 11100111 \\ 01011101 \\ 11010101 \\ 11101010 \\ 11111101 \\ 10101011 \end{bmatrix} \tag{7.4}$$

7.9 A note on Whirlpool-0

Matrix (7.5) is used in the Whirlpool-0 hash function. The irreducible polynomial of the field is $p(x) = x^8 + x^4 + x^3 + x^2 + 1$, and the inverse matrix is (7.6).

$$\begin{bmatrix} 01_x & 01_x & 03_x & 01_x & 05_x & 08_x & 09_x & 05_x \\ 05_x & 01_x & 01_x & 03_x & 01_x & 05_x & 08_x & 09_x \\ 09_x & 05_x & 01_x & 01_x & 03_x & 01_x & 05_x & 08_x \\ 08_x & 09_x & 05_x & 01_x & 01_x & 03_x & 01_x & 05_x \\ 05_x & 08_x & 09_x & 05_x & 01_x & 01_x & 03_x & 01_x \\ 01_x & 05_x & 08_x & 09_x & 05_x & 01_x & 01_x & 03_x \\ 03_x & 01_x & 05_x & 08_x & 09_x & 05_x & 01_x & 01_x \\ 01_x & 03_x & 01_x & 05_x & 08_x & 09_x & 05_x & 01_x \end{bmatrix} \tag{7.5}$$

$$\begin{bmatrix} a5_x & 07_x & 95_x & 9e_x & 0c_x & a7_x & 01_x & ea_x \\ d1_x & b3_x & e8_x & 5a_x & 01_x & ab_x & 3d_x & 6c_x \\ f8_x & 3c_x & 8a_x & 12_x & 2e_x & 8b_x & cc_x & 5e_x \\ 73_x & 27_x & 4e_x & e0_x & bf_x & c0_x & 24_x & 4a_x \\ 85_x & 17_x & 1c_x & 47_x & 0d_x & 4e_x & 5b_x & aa_x \\ 17_x & b0_x & d6_x & 2d_x & 6c_x & 26_x & ef_x & cb_x \\ dd_x & ce_x & 15_x & da_x & 6c_x & 17_x & 03_x & fc_x \\ ec_x & 78_x & d8_x & ea_x & da_x & 21_x & 17_x & b1_x \end{bmatrix} \tag{7.6}$$

In [69], it is claimed that (7.5) is MDS. However, in [68], Shirai and Shibutani find singular submatrices, which implies that the matrix is, in fact, not MDS. They also present alternative circulant matrices which are MDS.

One of the singular submatrices found in [68] is matrix (7.7). Other singular submatrices were found, for further detail we refer the reader to [68].

$$\begin{bmatrix} 01_x & 05_x \\ 01_x & 05_x \end{bmatrix} \tag{7.7}$$

Furthermore, Shirai and Shibutani obtain 224 MDS matrices satisfying the desired conditions for Whirlpool:

- Branch number equal to 9;
- As many 1-elements as possible (namely, 3);
- Hamming weight of any element is at most 2.

Matrices from (7.209) to (7.222) are the ones obtained by Shirai. They point out that, reversing elements and applying rotations to each of the C_i, it is possible to obtain additional MDS matrices, totalizing thus 224 MDS matrices. For further detail please refer to [68]. The authors of Whirlpool decided to replace the non-MDS matrix by Shirai's C_7 matrix, and therefore Whirlpool-0 refers to the version with the non-MDS matrix, prior to the correction, and Whirlpool refers to the corrected version, with the MDS matrix, which was submitted (and accepted) as a standard at ISO/IEC 10118-3. For further detail on the history and the updates of Whirlpool, the reader may refer to the Whirlpool web page [1].

$$C_0 = rcirc(01_x, 01_x, 02_x, 01_x, 05_x, 08_x, 09_x, 04_x) \tag{7.8}$$
$$C_1 = rcirc(01_x, 01_x, 02_x, 01_x, 06_x, 09_x, 08_x, 03_x) \tag{7.9}$$
$$C_2 = rcirc(01_x, 01_x, 02_x, 01_x, 08_x, 09_x, 04_x, 05_x) \tag{7.10}$$
$$C_3 = rcirc(01_x, 01_x, 02_x, 01_x, 09_x, 06_x, 04_x, 03_x) \tag{7.11}$$
$$C_4 = rcirc(01_x, 01_x, 02_x, 06_x, 05_x, 09_x, 01_x, 08_x) \tag{7.12}$$
$$C_5 = rcirc(01_x, 01_x, 03_x, 01_x, 04_x, 09_x, 05_x, 06_x) \tag{7.13}$$
$$C_6 = rcirc(01_x, 01_x, 03_x, 01_x, 08_x, 04_x, 09_x, 06_x) \tag{7.14}$$
$$C_7 = rcirc(01_x, 01_x, 04_x, 01_x, 08_x, 05_x, 02_x, 09_x) \tag{7.15}$$
$$C_8 = rcirc(01_x, 01_x, 04_x, 01_x, 09_x, 03_x, 02_x, 06_x) \tag{7.16}$$
$$C_9 = rcirc(01_x, 01_x, 04_x, 03_x, 06_x, 08_x, 01_x, 09_x) \tag{7.17}$$
$$C_{10} = rcirc(01_x, 01_x, 05_x, 01_x, 04_x, 06_x, 03_x, 09_x) \tag{7.18}$$
$$C_{11} = rcirc(01_x, 01_x, 05_x, 08_x, 02_x, 09_x, 01_x, 06_x) \tag{7.19}$$
$$C_{12} = rcirc(01_x, 01_x, 08_x, 01_x, 06_x, 03_x, 02_x, 09_x) \tag{7.20}$$
$$C_{13} = rcirc(01_x, 01_x, 08_x, 02_x, 04_x, 05_x, 01_x, 09_x) \tag{7.21}$$

The inverses of Shirai's matrices from C_0 to C_{13} are the following.

$$C_0^{-1} = rcirc(b5_x, 98_x, 23_x, fa_x, 23_x, a5_x, b6_x, 30_x) \tag{7.22}$$
$$C_1^{-1} = rcirc(bb_x, de_x, a0_x, df_x, 4a_x, 55_x, 7a_x, c5_x) \tag{7.23}$$
$$C_2^{-1} = rcirc(04_x, a4_x, cb_x, af_x, c2_x, 3e_x, 0e_x, c2_x) \tag{7.24}$$
$$C_3^{-1} = rcirc(4f_x, aa_x, 2c_x, 0c_x, 84_x, 76_x, 14_x, bb_x) \tag{7.25}$$
$$C_4^{-1} = rcirc(5b_x, e8_x, ed_x, e2_x, 33_x, 98_x, 82_x, 94_x) \tag{7.26}$$
$$C_5^{-1} = rcirc(87_x, d4_x, 76_x, 80_x, 9d_x, e4_x, 24_x, c5_x) \tag{7.27}$$
$$C_6^{-1} = rcirc(ad_x, ef_x, 44_x, 71_x, a8_x, e2_x, 42_x, 7e_x) \tag{7.28}$$
$$C_7^{-1} = rcirc(04_x, af_x, 0e_x, a4_x, c2_x, c2_x, cb_x, 3e_x) \tag{7.29}$$
$$C_8^{-1} = rcirc(4f_x, 0c_x, 14_x, aa_x, 84_x, bb_x, 2c_x, 76_x) \tag{7.30}$$
$$C_9^{-1} = rcirc(e2_x, 44_x, 7e_x, a8_x, ef_x, 42_x, 71_x, ad_x) \tag{7.31}$$
$$C_{10}^{-1} = rcirc(87_x, 80_x, 24_x, d4_x, 9d_x, c5_x, 76_x, e4_x) \tag{7.32}$$
$$C_{11}^{-1} = rcirc(ed_x, 98_x, 5b_x, e2_x, 82_x, e8_x, 33_x, 94_x) \tag{7.33}$$
$$C_{12}^{-1} = rcirc(bb_x, df_x, 7a_x, de_x, 4a_x, c5_x, a0_x, 55_x) \tag{7.34}$$
$$C_{13}^{-1} = rcirc(a5_x, 23_x, 30_x, 23_x, 98_x, b6_x, fa_x, b5_x) \tag{7.35}$$

7.10 About Whirlwind and its (possibly) non-MDS matrix

Matrices (7.38) and (7.40) are used in the Whirlwind hash function, with the irreducible polynomial $p(x) = x^4 + x + 1$. The inverses are, respectively, (7.39) and (7.41). In [2], it is claimed that the matrix is MDS. However, in our work, we have found singular submatrices (see Table 7.9), which leads us to believe, according to Theorem 2, that it is not MDS.

Singular submatrix dimension	8×8	7×7	6×6	5×5	4×4	3×3	2×2	1×1	**Total**
M_0	0	0	48	176	152	176	48	0	600
M_1	0	0	48	256	448	256	48	0	1056

Table 7.9: Amount of singular submatrices found in Whirlwind

For example, the smallest singular submatrix we have found (matrix (7.36)) for (7.38) is 2×2. It is clear that the determinant is $(x+1)(x^2+x) - x(x^2+1) = x^3+x^2+x^2+x-(x^3+x) = x^3+x-x^3-x = 0$.

$$\begin{bmatrix} 3_x & 5_x \\ 2_x & 6_x \end{bmatrix} = \begin{bmatrix} x+1 & x^2+1 \\ x & x^2+x \end{bmatrix} \tag{7.36}$$

There are too many singular submatrices to list, therefore, we list which rows and columns should be removed from the original matrix in order to obtain them, in Table 7.16. Our row/column index starts at 0 and ends at 7.

For (7.40), an example of singular submatrix is (7.37), and the list of rows and columns to be removed to yield all singular submatrices is given by Table 7.17.

$$\begin{bmatrix} 8_x & 7_x \\ 3_x & e_x \end{bmatrix} = \begin{bmatrix} x^3 & x^2+x+1 \\ x+1 & x^3+x^2+x \end{bmatrix} \tag{7.37}$$

The fact that we have found singular submatrices for Whirlwind's M_0 and M_1 matrices shows they are not MDS in $GF(2^4)$. However, it is relevant to note that, in [2], the authors mention the usage of the $GF(2^{16})$ field with *decompositions* to $GF(2^4)$, as well as the usage of a *normal basis* representation for the finite fields. It is possible that, despite not MDS in $GF(2^4)$ with $p(x) = x^4 + x + 1$ as the irreducible polynomial and a polynomial basis $\{1, x, x^2, x^3, x^4\}$, matrices (7.38) and (7.40) be MDS for the normal basis representation. However, we leave the evaluation of the MDS property for different basis as a future work.

Year	**Ord**	**Type**	**Inv**	**Use**	**Bib**	$GF(2)[x]/(p(x))$	**#xor**	**#xtime**	**Matrices**
2010	8	dyadic	no	Whirlwind	[2]	x^4+x+1	104 128	136 136	(7.38) (7.39)
2010	8	dyadic	no	Whirlwind	[2]	x^4+x+1	128 128	128 128	(7.40) (7.41)

Table 7.10: Discovered non-MDS matrices in Whirlwind: parameters, usage and cost

$$\begin{bmatrix} 5_x & 4_x & a_x & 6_x & 2_x & d_x & 8_x & 3_x \\ 4_x & 5_x & 6_x & a_x & d_x & 2_x & 3_x & 8_x \\ a_x & 6_x & 5_x & 4_x & 8_x & 3_x & 2_x & d_x \\ 6_x & a_x & 4_x & 5_x & 3_x & 8_x & d_x & 2_x \\ 2_x & d_x & 8_x & 3_x & 5_x & 4_x & a_x & 6_x \\ d_x & 2_x & 3_x & 8_x & 4_x & 5_x & 6_x & a_x \\ 8_x & 3_x & 2_x & d_x & a_x & 6_x & 5_x & 4_x \\ 3_x & 8_x & d_x & 2_x & 6_x & a_x & 4_x & 5_x \end{bmatrix} \tag{7.38}$$

$$\begin{bmatrix} 7_x & 3_x & e_x & b_x & 8_x & 1_x & 6_x & c_x \\ 3_x & 7_x & b_x & e_x & 1_x & 8_x & c_x & 6_x \\ e_x & b_x & 7_x & 3_x & 6_x & c_x & 8_x & 1_x \\ b_x & e_x & 3_x & 7_x & c_x & 6_x & 1_x & 8_x \\ 8_x & 1_x & 6_x & c_x & 7_x & 3_x & e_x & b_x \\ 1_x & 8_x & c_x & 6_x & 3_x & 7_x & b_x & e_x \\ 6_x & c_x & 8_x & 1_x & e_x & b_x & 7_x & 3_x \\ c_x & 6_x & 1_x & 8_x & b_x & e_x & 3_x & 7_x \end{bmatrix} \tag{7.39}$$

$$\begin{bmatrix} 5_x & e_x & 4_x & 7_x & 1_x & 3_x & f_x & 8_x \\ e_x & 5_x & 7_x & 4_x & 3_x & 1_x & 8_x & f_x \\ 4_x & 7_x & 5_x & e_x & f_x & 8_x & 1_x & 3_x \\ 7_x & 4_x & e_x & 5_x & 8_x & f_x & 3_x & 1_x \\ 1_x & 3_x & f_x & 8_x & 5_x & e_x & 4_x & 7_x \\ 3_x & 1_x & 8_x & f_x & e_x & 5_x & 7_x & 4_x \\ f_x & 8_x & 1_x & 3_x & 4_x & 7_x & 5_x & e_x \\ 8_x & f_x & 3_x & 1_x & 7_x & 4_x & e_x & 5_x \end{bmatrix} \tag{7.40}$$

$$\begin{bmatrix} f_x & 1_x & c_x & 9_x & 3_x & 5_x & 2_x & b_x \\ 1_x & f_x & 9_x & c_x & 5_x & 3_x & b_x & 2_x \\ c_x & 9_x & f_x & 1_x & 2_x & b_x & 3_x & 5_x \\ 9_x & c_x & 1_x & f_x & b_x & 2_x & 5_x & 3_x \\ 3_x & 5_x & 2_x & b_x & f_x & 1_x & c_x & 9_x \\ 5_x & 3_x & b_x & 2_x & 1_x & f_x & 9_x & c_x \\ 2_x & b_x & 3_x & 5_x & c_x & 9_x & f_x & 1_x \\ b_x & 2_x & 5_x & 3_x & 9_x & c_x & 1_x & f_x \end{bmatrix} \tag{7.41}$$

7.11 About one of Duwal's matrices

Matrix (7.198) is presented in [28] as an MDS matrix. However, our experiments (see Listing 7.11) pointed it out as a non-MDS matrix. If we remove rows 1 and 3, and columns 1 and 2, we are left with the submatrix

$$\begin{bmatrix} 3 & 3 \\ 4 & 4 \end{bmatrix} \tag{7.42}$$

which is singular because it's determinant will be $3 \times 4 - 3 \times 4 = 0$, where 3 and 4 are the integer representations of the respective polynomials and 0 is the zero polynomial.

Listing 7.1: Experiment output snippet showing detected singular submatrix

```
['0x3', '0x2', '0x1', '0x3']
['0x2', '0x3', '0x1', '0x1']
['0x4', '0x3', '0x6', '0x4']
['0x1', '0x1', '0x4', '0x6']
non mds matrix detected
submat [[3, 3],
 [4, 4]]
rows to remove (1, 3)
cols to remove (1, 2)
mds False
xor 18
xtime 16
```

7.12 Non-MDS matrix catalogue

Table 7.11 presents matrices which were reported in the literature for usage in symmetric block ciphers (or hash functions) but are not MDS. It shows their **xor** and **xtime** costs, as well as the respective branch numbers. Column **Ord** refers to the matrix dimensions, **Bib** contains the bibliographic reference, **#xor** and **#xtime** refer to the required amount of **xor** and **xtime** operations, i.e the costs, and $\mathcal{B}(\theta)$ presents the upper bound for the branch number. According to the analysis conducted in [68], Whirlpool's matrix has $\mathcal{B}(\theta) \leq 8$. We present now a brief analysis for Hierocrypt-L1 and Hierocrypt-3.

For convenience, we recall that the branch number (see Definition 18) $\mathcal{B}$ of a linear mapping θ is $\mathcal{B}(\theta) = \min_{a \neq 0}\{w(a) + w(\theta(a))\}$, where $w(a)$ is the Hamming weight. The Hamming weight $w(a)$ is the number of non-zero componentes in the vector a. In order to compute the branch number of a transformation θ, we must test non-zero vectors. For a given Hamming weight z, there are $\binom{n}{z}$ possible vectors, assuming we have a fixed symbolic non-zero value d and that we must choose whether a component is zero or equal to d. Since z ranges from 1 to n, the total amount of vectors to be tested with this approach is $\sum_{z=1}^{z=n} \binom{n}{z} = 2^n - 1$.

For Hierocrypt-L1, θ consists of left multiplication by an 8×8 matrix, therefore, we must test non-zero vectors a with $n = 8$ components. For $a = (d, 0, 0, 0, 0, 0, 0, 0)$, $w(a) = 1$, $\theta(a) = (1, 1, 1, 0, 1, 1, 1, 1)$ and $w(\theta(a)) = 7$, which tells us $\mathcal{B}(\theta) \leq 1 + 7 = 8$. For $a = (d, d, 0, 0, 0, 0, 0, 0)$, on the other hand, $w(a) = 2$, $\theta(a) = (1, 0, 0, 1, 0, 0, 0, 1)$ and $w(\theta(a)) = 3$. Thus we know that actually $\mathcal{B}(\theta) \leq 2 + 3 = 5$.

For Hierocrypt-3, the matrix is 16×16, thus we must test non-zero vectors with $n = 16$ components. For $a = (0, 0, 0, d, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, we have $w(a) = 1$, $\theta(a) = (0,d,0,d,d,d,d,0,d,0,d,0,0,d,0,d)$ and $w(\theta(a)) = 9$, leading to $\mathcal{B}(\theta) \leq 1 + 9 = 10$.

Year	**Dim**	**Bib**	**Type**	**Use**	$\mathcal{B}(\theta)$	**#xor**	**#xtime**	**Matrix**
2000	16	[17]	binary	Hierocrypt-3	≤ 10	160	0	(7.3)
2000	8	[18]	binary	Hierocrypt-L1	≤ 5	37	0	(7.4)
2003	8	[69]	right circulant	Whirlpool-0	≤ 8	89 247	87 366	(7.5) (7.6)
2009	16	[57]	—	suggested for AES	—	512	624	(7.44)
2018	16	[28]	—	—	—	18	16	(7.198)

Table 7.11: Non-MDS matrices: parameters, usage and costs.

In [57], they propose a 16×16 matrix (see matrix (7.44)) which could replace AES's matrix and result in complete diffusion in a single round of the cipher. However, although claimed MDS, it is not MDS. As an example, removing rows $(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14)$ and columns $(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 15)$ results in a 2×2 singular submatrix (see matrix (7.43)). They claim to use a Cauchy construction, but a Cauchy construction would ensure MDS property. Therefore, we believe it is not a Cauchy matrix. We have found other singular submatrices for different dimensions, but here we show a 2×2 example for simplicity. We have also checked it for involutory property, and it is involutory although not MDS.

$$\begin{bmatrix} 09_x & 05_x \\ 07_x & 03_x \end{bmatrix} = \begin{bmatrix} x^3 + 1 & x^2 + 1 \\ x^2 + x + 1 & x + 1 \end{bmatrix} \tag{7.43}$$

$$\begin{bmatrix}
01_x03_x04_x05_x06_x07_x08_x09_x0a_x0b_x0c_x0d_x0e_x10_x02_x1e_x \\
03_x01_x05_x04_x07_x06_x09_x08_x0b_x0a_x0d_x0c_x10_x0d_x1d_x02_x \\
04_x05_x01_x03_x08_x09_x06_x07_x0c_x0d_x0a_x0b_x02_x1e_x0e_x10_x \\
05_x04_x03_x01_x09_x08_x07_x06_x0d_x0c_x0b_x0a_x1e_x02_x10_x0e_x \\
06_x07_x08_x09_x01_x03_x04_x05_x0e_x10_x02_x1e_x0a_x0b_x0c_x0d_x \\
07_x06_x09_x08_x03_x01_x05_x04_x10_x0e_x1e_x02_x0b_x0a_x0d_x0c_x \\
08_x09_x06_x07_x04_x05_x01_x03_x02_x1e_x0e_x10_x0c_x0d_x0a_x0b_x \\
09_x08_x07_x06_x05_x04_x03_x01_x1e_x02_x10_x0e_x0d_x0c_x0b_x0a_x \\
0a_x0b_x0c_x0d_x0e_x10_x02_x1e_x01_x03_x04_x05_x06_x07_x08_x09_x \\
0b_x0a_x0d_x0c_x10_x0e_x1e_x02_x03_x01_x05_x04_x07_x06_x09_x08_x \\
0c_x0d_x0a_x0b_x02_x1e_x0e_x10_x04_x05_x01_x03_x08_x09_x06_x07_x \\
0d_x0c_x0b_x0a_x1e_x02_x10_x0e_x05_x04_x03_x01_x09_x08_x07_x06_x \\
0e_x10_x02_x1e_x0a_x0b_x0c_x0d_x06_x07_x08_x09_x01_x03_x04_x05_x \\
10_x0e_x1e_x02_x0b_x0a_x0d_x0c_x07_x06_x09_x08_x03_x01_x05_x04_x \\
02_x1e_x0e_x10_x0c_x0d_x0a_x0b_x08_x09_x06_x07_x04_x05_x01_x03_x \\
1e_x02_x10_x0e_x0d_x0c_x0b_x0a_x09_x08_x07_x06_x05_x04_x03_x01_x
\end{bmatrix} \tag{7.44}$$

7.13 About the irreducible polynomials

It is worth noting that, when changing the irreducible polynomial, the finite field representation changes, and that may cause the inverse matrices to change, resulting in inverse matrices with different computational costs. In this work, we briefly explore this. There are 30 irreducible polynomials of degree 8, which can be used to build $GF(2^8)$, and 3 irreducible polynomials of degree 4, which can respectively be used to build $GF(2^4)$. It is possible to obtain them e.g using the SAGE tool [27] or the Galois Python package [38] for finite field arithmetic.

Table 7.15 shows, for each matrix studied in this work (column A) and each possible irreducible polynomial (column $p(x)$), the determinant (column $|A|$), the multiplicative inverse of the determinant ($\frac{1}{|A|}$), the **xor** cost (xr), the **xtime** cost (xt), whether A is involutory (column Iv), whether the MDS property holds (column MDS) on the finite field, the determinant of the inverse matrix A^{-1} (column $|A|_i$), **xor** and **xtime** costs (xr$_i$ and xt$_i$, respectively), involutory property for A^{-1} (column Iv$_i$) and, finally, MDS property for A^{-1} (column MDS$_i$).

It is worth noting that, for some matrices and fields, the MDS property does not hold. It is the case of SHARK, where the MDS property only holds for the $p(x)$ chosen by the cipher's authors. However, for others, such as Rijndael, KHAZAD and Anubis, the MDS and involutory properties always hold, no matter the field, the determinant does not change, nor do the **xor** and **xtime** costs of the inverse matrix. This suggests that it is possible to choose the best irreducible polynomial when designing a diffusion layer — the polynomial which keeps MDS property and yields inverse matrices with the lowest cost. A brief analysis of this phenomenon follows.

Let A be a matrix. A^{-1} can be obtained by e.g Gauss-Jordan elimination, but another way of finding A^{-1} is to use the adjugate matrix, here denoted A^*.

$$A^{-1} = \frac{1}{|A|}A^* \tag{7.45}$$

$$A^* = (-1)^{i+j}|M_{ji}| \tag{7.46}$$

where M_{ji} is a submatrix obtained by removing j-th row and i-th column of A. Recall that $|A|$ depends on co-factors of A, i.e products of elements of submatrices of A, and, therefore, $|A|$ depends on $p(x)$ — since products are performed modulo $p(x)$ in finite fields.

In Table 7.15, this phenomenon can be observed e.g for the SHARK cipher. However, for some ciphers, the inverse matrix remains the same. Note that, for these matrices (for example, the Rijndael matrix), $|A|= 1$ and thus $\frac{1}{|A|} = 1$ as well, leading to an invariant adjugate matrix $A^* = A^{-1}$, independent of $p(x)$.

For Rijndael specifically, A is a 4×4 matrix with elements always 1_x, 2_x or 3_x, and $|A|$ is computed through co-factors (determinants of 3×3 submatrices). In each of the co-factors, we have sums of products of 3 elements of A, and the irreducible polynomial has degree 8.

Note that the products of any 3 elements of A have maximum degree of 3. For example, the maximum product would be $3_x \times 3_x \times 3_x = (x + 1)^3$. This shows us that the modular reduction is never performed when computing the determinant — hence the inverse matrix does not depend on $p(x)$ and will always be the same, regardless of which $p(x)$ (of degree greater than 3) we choose.

In conclusion, for the general case, the elements of A^{-1} depend on $\frac{1}{|A|}$, which depends on $|A|$. And $|A|$ depends on $p(x)$, since products of matrix elements are performed modulo $p(x)$ and the determinant of a matrix depends on products of its elements. Therefore, changing $p(x)$ may change $|A|$, which consequently may change A^{-1}, but, if $|A|= 1$, the inverse matrix will not depend on $p(x)$. Furthermore, note that $|A|$ may or may not change across different choices of $p(x)$. For example, SHARK's $|A|$ changes, whilst for BKSQ's matrix the determinant is always 3, but, still, the inverse of 3 changes for each $p(x)$, leading to variant inverse matrices for BKSQ. The only cases where the inverse matrix was invariant on $p(x)$ in our experiments were for $|A|= 1$, since the inverse of 1 is always 1 itself in any finite field.

Table 7.12 summarizes this for all the studied matrices.

Matrix identifier	**Determinant**	**MDS property**	**Inverse changes**	**Involutory**
SHARK	changes	for one (or more) of the $p(x)$	yes	never
SQUARE	always 1	always	no	never
BKSQ	always 3	always	yes	never
Tavares	always 1	for one or more of the $p(x)$	no	always
KHAZAD	always 1	for one or more of the $p(x)$	no	always
ANUBIS	always 1	always	no	always
ANUBIS (KE)	changes	always	yes	never
Grostl	changes	for one or more of the $p(x)$	yes	never
Curupira	always 1	always	no	always
Curupira (KE)	always 1D	always	yes	never
Rijndael	always 1	always	no	never
Hierocrypt low	changes	for one or more of the $p(x)$	yes	never

Matrix identifier	Determinant	MDS property	Inverse changes	Involutory
Hierocrypt L1	changes	for one or more of the $p(x)$	yes	never
FOX mu4	changes	for one or more of the $p(x)$	yes	never
FOX mu8	changes	for one or more of the $p(x)$	yes	never
Whirlpool-0	changes	never	yes	never

Table 7.12: Summarizing the effect of changing the irreducible polynomial

7.14 Conclusions

In this chapter we presented relevant background regarding MDS (Maximum Distance Separable) property, such as abstract algebra, finite fields and linear codes. Then, we showed the main matrix types being studied in the literature for MDS constructions (circulant, Hadamard, Cauchy, serial, among others) and ways to attain involutory property.

We presented an extensive catalogue of MDS matrices reported in the literature along with their **xor** and **xtime** costs, as well as non-MDS matrices which are yet thought to be MDS and their singular submatrices that show they are not actually MDS.

We also showed an experiment regarding how the irreducible polynomials lead to changes in the inverse matrix for some of the matrices of our catalogue, and concrete instances of generic matrices of Beierle's and Duwal's works.

Furthermore, we analyzed both mathematically and empirically the complexity of evaluating a matrix for MDS property as the dimension matrix n increases, as well as the size of the search space.

7.15 Appendix 7A: Instancing Generic Matrices

Year	Dim.	m	#xor	Inv. #xor	#xtime	Inv. #xtime	α	Instanced Matrix
2016	2	4	2	8	2	10	2	(7.200)
2016	2	8	2	16	2	26	2	(7.200)
2016	3	4	6	21	3	21	2	(7.202)
2016	3	8	6	39	3	57	2	(7.202)
2016	2	4	8	2	10	2	2	(7.201)
2016	2	4	6	4	10	2	3	(7.201)
2016	2	4	4	2	8	4	4	(7.201)
2016	2	4	8	4	8	4	5	(7.201)
2016	2	4	6	4	4	4	6	(7.201)
2016	2	4	4	6	4	4	7	(7.201)
2016	2	4	4	2	8	6	8	(7.201)
2016	2	4	6	4	10	6	9	(7.201)
2016	2	4	6	4	6	6	10	(7.201)
2016	2	4	10	6	8	6	11	(7.201)
2016	2	4	8	4	6	6	12	(7.201)
2016	2	4	2	6	8	6	13	(7.201)
2016	2	4	4	6	10	6	14	(7.201)
2016	2	4	6	8	8	6	15	(7.201)
2016	2	8	16	2	26	2	2	(7.201)
2016	2	8	14	4	26	2	3	(7.201)
2016	2	8	16	2	26	4	4	(7.201)
2016	2	8	18	4	28	4	5	(7.201)
2016	2	8	12	4	24	4	6	(7.201)
2016	2	8	12	6	28	4	7	(7.201)
2016	2	8	16	2	26	6	8	(7.201)
2016	2	8	16	4	24	6	9	(7.201)
2016	2	8	16	4	26	6	10	(7.201)
2016	2	8	14	6	26	6	11	(7.201)
2016	2	8	10	4	24	6	12	(7.201)
2016	2	8	8	6	14	6	13	(7.201)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	2	8	10	6	22	6	14	(7.201)
2016	2	8	12	8	28	6	15	(7.201)
2016	2	8	10	2	28	8	16	(7.201)
2016	2	8	14	4	26	8	17	(7.201)
2016	2	8	8	4	28	8	18	(7.201)
2016	2	8	18	6	22	8	19	(7.201)
2016	2	8	12	4	12	8	20	(7.201)
2016	2	8	12	6	26	8	21	(7.201)
2016	2	8	18	6	28	8	22	(7.201)
2016	2	8	16	8	28	8	23	(7.201)
2016	2	8	14	4	26	8	24	(7.201)
2016	2	8	14	6	20	8	25	(7.201)
2016	2	8	12	6	22	8	26	(7.201)
2016	2	8	10	8	28	8	27	(7.201)
2016	2	8	12	6	22	8	28	(7.201)
2016	2	8	18	8	26	8	29	(7.201)
2016	2	8	8	8	18	8	30	(7.201)
2016	2	8	10	10	28	8	31	(7.201)
2016	2	8	18	2	22	10	32	(7.201)
2016	2	8	14	4	28	10	33	(7.201)
2016	2	8	8	4	28	10	34	(7.201)
2016	2	8	6	6	18	10	35	(7.201)
2016	2	8	10	4	28	10	36	(7.201)
2016	2	8	18	6	28	10	37	(7.201)
2016	2	8	20	6	26	10	38	(7.201)
2016	2	8	4	8	16	10	39	(7.201)
2016	2	8	12	4	22	10	40	(7.201)
2016	2	8	12	6	28	10	41	(7.201)
2016	2	8	12	6	28	10	42	(7.201)
2016	2	8	12	8	24	10	43	(7.201)
2016	2	8	18	6	26	10	44	(7.201)
2016	2	8	8	8	24	10	45	(7.201)
2016	2	8	16	8	28	10	46	(7.201)
2016	2	8	18	10	24	10	47	(7.201)
2016	2	8	16	4	26	10	48	(7.201)
2016	2	8	14	6	28	10	49	(7.201)
2016	2	8	6	6	16	10	50	(7.201)
2016	2	8	14	8	28	10	51	(7.201)
2016	2	8	14	6	22	10	52	(7.201)
2016	2	8	20	8	26	10	53	(7.201)
2016	2	8	12	8	28	10	54	(7.201)
2016	2	8	16	10	28	10	55	(7.201)
2016	2	8	10	6	24	10	56	(7.201)
2016	2	8	6	8	28	10	57	(7.201)
2016	2	8	14	8	24	10	58	(7.201)
2016	2	8	10	10	26	10	59	(7.201)
2016	2	8	10	8	24	10	60	(7.201)
2016	2	8	8	10	24	10	61	(7.201)
2016	2	8	14	10	26	10	62	(7.201)
2016	2	8	8	12	14	10	63	(7.201)
2016	2	8	14	2	28	12	64	(7.201)
2016	2	8	18	4	24	12	65	(7.201)
2016	2	8	12	4	18	12	66	(7.201)
2016	2	8	20	6	20	12	67	(7.201)
2016	2	8	18	4	28	12	68	(7.201)

Year	Dim.	m	#xor	Inv. #xor	#xtime	Inv. #xtime	α	Instanced Matrix
2016	2	8	8	6	20	12	69	(7.201)
2016	2	8	4	6	16	12	70	(7.201)
2016	2	8	18	8	26	12	71	(7.201)
2016	2	8	18	4	26	12	72	(7.201)
2016	2	8	14	6	24	12	73	(7.201)
2016	2	8	14	6	24	12	74	(7.201)
2016	2	8	10	8	18	12	75	(7.201)
2016	2	8	16	6	28	12	76	(7.201)
2016	2	8	12	8	14	12	77	(7.201)
2016	2	8	10	8	20	12	78	(7.201)
2016	2	8	16	10	28	12	79	(7.201)
2016	2	8	12	4	24	12	80	(7.201)
2016	2	8	8	6	18	12	81	(7.201)
2016	2	8	16	6	26	12	82	(7.201)
2016	2	8	16	8	28	12	83	(7.201)
2016	2	8	12	6	24	12	84	(7.201)
2016	2	8	10	8	22	12	85	(7.201)
2016	2	8	16	8	28	12	86	(7.201)
2016	2	8	8	10	24	12	87	(7.201)
2016	2	8	20	6	22	12	88	(7.201)
2016	2	8	8	8	14	12	89	(7.201)
2016	2	8	22	8	28	12	90	(7.201)
2016	2	8	12	10	26	12	91	(7.201)
2016	2	8	12	8	24	12	92	(7.201)
2016	2	8	12	10	22	12	93	(7.201)
2016	2	8	14	10	28	12	94	(7.201)
2016	2	8	16	12	24	12	95	(7.201)
2016	2	8	12	4	28	12	96	(7.201)
2016	2	8	6	6	24	12	97	(7.201)
2016	2	8	16	6	28	12	98	(7.201)
2016	2	8	10	8	26	12	99	(7.201)
2016	2	8	12	6	20	12	100	(7.201)
2016	2	8	12	8	24	12	101	(7.201)
2016	2	8	14	8	28	12	102	(7.201)
2016	2	8	10	10	14	12	103	(7.201)
2016	2	8	14	6	26	12	104	(7.201)
2016	2	8	12	8	28	12	105	(7.201)
2016	2	8	20	8	26	12	106	(7.201)
2016	2	8	16	10	22	12	107	(7.201)
2016	2	8	16	8	22	12	108	(7.201)
2016	2	8	12	10	24	12	109	(7.201)
2016	2	8	20	10	24	12	110	(7.201)
2016	2	8	8	12	16	12	111	(7.201)
2016	2	8	10	6	22	12	112	(7.201)
2016	2	8	10	8	18	12	113	(7.201)
2016	2	8	12	8	26	12	114	(7.201)
2016	2	8	10	10	14	12	115	(7.201)
2016	2	8	12	8	20	12	116	(7.201)
2016	2	8	12	10	16	12	117	(7.201)
2016	2	8	16	10	28	12	118	(7.201)
2016	2	8	16	12	26	12	119	(7.201)
2016	2	8	14	8	28	12	120	(7.201)
2016	2	8	14	10	26	12	121	(7.201)
2016	2	8	14	10	24	12	122	(7.201)
2016	2	8	6	12	16	12	123	(7.201)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	2	8	12	10	26	12	124	(7.201)
2016	2	8	10	12	16	12	125	(7.201)
2016	2	8	18	12	22	12	126	(7.201)
2016	2	8	16	14	22	12	127	(7.201)
2016	2	8	8	2	28	14	128	(7.201)
2016	2	8	10	4	24	14	129	(7.201)
2016	2	8	10	4	24	14	130	(7.201)
2016	2	8	16	6	26	14	131	(7.201)
2016	2	8	10	4	26	14	132	(7.201)
2016	2	8	12	6	28	14	133	(7.201)
2016	2	8	16	6	26	14	134	(7.201)
2016	2	8	18	8	26	14	135	(7.201)
2016	2	8	12	4	24	14	136	(7.201)
2016	2	8	12	6	28	14	137	(7.201)
2016	2	8	6	6	16	14	138	(7.201)
2016	2	8	10	8	24	14	139	(7.201)
2016	2	8	14	6	28	14	140	(7.201)
2016	2	8	14	8	28	14	141	(7.201)
2016	2	8	14	8	26	14	142	(7.201)
2016	2	8	4	10	8	14	143	(7.201)
2016	2	8	12	4	26	14	144	(7.201)
2016	2	8	14	6	24	14	145	(7.201)
2016	2	8	16	6	28	14	146	(7.201)
2016	2	8	18	8	28	14	147	(7.201)
2016	2	8	12	6	24	14	148	(7.201)
2016	2	8	12	8	26	14	149	(7.201)
2016	2	8	10	8	28	14	150	(7.201)
2016	2	8	14	10	24	14	151	(7.201)
2016	2	8	16	6	26	14	152	(7.201)
2016	2	8	12	8	24	14	153	(7.201)
2016	2	8	12	8	22	14	154	(7.201)
2016	2	8	14	10	24	14	155	(7.201)
2016	2	8	6	8	24	14	156	(7.201)
2016	2	8	18	10	24	14	157	(7.201)
2016	2	8	12	10	26	14	158	(7.201)
2016	2	8	12	12	20	14	159	(7.201)
2016	2	8	14	4	22	14	160	(7.201)
2016	2	8	12	6	24	14	161	(7.201)
2016	2	8	10	6	20	14	162	(7.201)
2016	2	8	14	8	26	14	163	(7.201)
2016	2	8	12	6	22	14	164	(7.201)
2016	2	8	16	8	24	14	165	(7.201)
2016	2	8	6	8	16	14	166	(7.201)
2016	2	8	20	10	28	14	167	(7.201)
2016	2	8	12	6	18	14	168	(7.201)
2016	2	8	8	8	26	14	169	(7.201)
2016	2	8	10	8	14	14	170	(7.201)
2016	2	8	12	10	24	14	171	(7.201)
2016	2	8	4	8	24	14	172	(7.201)
2016	2	8	14	10	26	14	173	(7.201)
2016	2	8	18	10	26	14	174	(7.201)
2016	2	8	10	12	24	14	175	(7.201)
2016	2	8	20	6	26	14	176	(7.201)
2016	2	8	6	8	18	14	177	(7.201)
2016	2	8	20	8	28	14	178	(7.201)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	2	8	14	10	22	14	179	(7.201)
2016	2	8	12	8	26	14	180	(7.201)
2016	2	8	14	10	18	14	181	(7.201)
2016	2	8	14	10	24	14	182	(7.201)
2016	2	8	12	12	24	14	183	(7.201)
2016	2	8	16	8	28	14	184	(7.201)
2016	2	8	14	10	20	14	185	(7.201)
2016	2	8	14	10	28	14	186	(7.201)
2016	2	8	8	12	16	14	187	(7.201)
2016	2	8	10	10	26	14	188	(7.201)
2016	2	8	14	12	22	14	189	(7.201)
2016	2	8	16	12	26	14	190	(7.201)
2016	2	8	16	14	24	14	191	(7.201)
2016	2	8	10	4	28	14	192	(7.201)
2016	2	8	10	6	14	14	193	(7.201)
2016	2	8	14	6	24	14	194	(7.201)
2016	2	8	18	8	26	14	195	(7.201)
2016	2	8	10	6	24	14	196	(7.201)
2016	2	8	14	8	28	14	197	(7.201)
2016	2	8	12	8	26	14	198	(7.201)
2016	2	8	10	10	26	14	199	(7.201)
2016	2	8	12	6	26	14	200	(7.201)
2016	2	8	14	8	22	14	201	(7.201)
2016	2	8	14	8	28	14	202	(7.201)
2016	2	8	24	10	28	14	203	(7.201)
2016	2	8	14	8	28	14	204	(7.201)
2016	2	8	8	10	16	14	205	(7.201)
2016	2	8	14	10	18	14	206	(7.201)
2016	2	8	16	12	24	14	207	(7.201)
2016	2	8	14	6	28	14	208	(7.201)
2016	2	8	12	8	18	14	209	(7.201)
2016	2	8	14	8	26	14	210	(7.201)
2016	2	8	16	10	26	14	211	(7.201)
2016	2	8	12	8	28	14	212	(7.201)
2016	2	8	18	10	20	14	213	(7.201)
2016	2	8	12	10	14	14	214	(7.201)
2016	2	8	10	12	14	14	215	(7.201)
2016	2	8	12	8	28	14	216	(7.201)
2016	2	8	12	10	14	14	217	(7.201)
2016	2	8	14	10	28	14	218	(7.201)
2016	2	8	12	12	28	14	219	(7.201)
2016	2	8	10	10	24	14	220	(7.201)
2016	2	8	18	12	26	14	221	(7.201)
2016	2	8	16	12	20	14	222	(7.201)
2016	2	8	12	14	24	14	223	(7.201)
2016	2	8	10	6	12	14	224	(7.201)
2016	2	8	10	8	14	14	225	(7.201)
2016	2	8	18	8	26	14	226	(7.201)
2016	2	8	16	10	26	14	227	(7.201)
2016	2	8	18	8	26	14	228	(7.201)
2016	2	8	8	10	18	14	229	(7.201)
2016	2	8	16	10	26	14	230	(7.201)
2016	2	8	14	12	26	14	231	(7.201)
2016	2	8	10	8	28	14	232	(7.201)
2016	2	8	14	10	20	14	233	(7.201)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	2	8	18	10	26	14	234	(7.201)
2016	2	8	10	12	26	14	235	(7.201)
2016	2	8	14	10	24	14	236	(7.201)
2016	2	8	12	12	28	14	237	(7.201)
2016	2	8	22	12	28	14	238	(7.201)
2016	2	8	18	14	26	14	239	(7.201)
2016	2	8	18	8	26	14	240	(7.201)
2016	2	8	16	10	26	14	241	(7.201)
2016	2	8	12	10	18	14	242	(7.201)
2016	2	8	20	12	26	14	243	(7.201)
2016	2	8	12	10	26	14	244	(7.201)
2016	2	8	6	12	8	14	245	(7.201)
2016	2	8	18	12	26	14	246	(7.201)
2016	2	8	12	14	22	14	247	(7.201)
2016	2	8	14	10	18	14	248	(7.201)
2016	2	8	14	12	28	14	249	(7.201)
2016	2	8	12	12	28	14	250	(7.201)
2016	2	8	10	14	16	14	251	(7.201)
2016	2	8	14	12	22	14	252	(7.201)
2016	2	8	16	14	28	14	253	(7.201)
2016	2	8	16	14	22	14	254	(7.201)
2016	2	8	14	16	28	14	255	(7.201)
2016	3	4	21	6	21	3	2	(7.203)
2016	3	4	24	9	21	3	3	(7.203)
2016	3	4	18	6	21	6	4	(7.203)
2016	3	4	18	9	21	6	5	(7.203)
2016	3	4	12	9	6	6	6	(7.203)
2016	3	4	9	12	6	6	7	(7.203)
2016	3	4	27	6	27	9	8	(7.203)
2016	3	4	18	9	21	9	9	(7.203)
2016	3	4	15	9	27	9	10	(7.203)
2016	3	4	15	12	24	9	11	(7.203)
2016	3	4	21	9	27	9	12	(7.203)
2016	3	4	18	12	24	9	13	(7.203)
2016	3	4	21	12	21	9	14	(7.203)
2016	3	4	18	15	27	9	15	(7.203)
2016	3	8	39	6	57	3	2	(7.203)
2016	3	8	42	9	57	3	3	(7.203)
2016	3	8	27	6	60	6	4	(7.203)
2016	3	8	30	9	63	6	5	(7.203)
2016	3	8	24	9	60	6	6	(7.203)
2016	3	8	24	12	63	6	7	(7.203)
2016	3	8	24	6	51	9	8	(7.203)
2016	3	8	24	9	51	9	9	(7.203)
2016	3	8	33	9	57	9	10	(7.203)
2016	3	8	24	12	57	9	11	(7.203)
2016	3	8	42	9	57	9	12	(7.203)
2016	3	8	39	12	63	9	13	(7.203)
2016	3	8	42	12	60	9	14	(7.203)
2016	3	8	39	15	63	9	15	(7.203)
2016	3	8	33	6	63	12	16	(7.203)
2016	3	8	33	9	60	12	17	(7.203)
2016	3	8	21	9	63	12	18	(7.203)
2016	3	8	24	12	60	12	19	(7.203)
2016	3	8	42	9	63	12	20	(7.203)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	3	8	48	12	57	12	21	(7.203)
2016	3	8	24	12	48	12	22	(7.203)
2016	3	8	27	15	48	12	23	(7.203)
2016	3	8	27	9	57	12	24	(7.203)
2016	3	8	39	12	57	12	25	(7.203)
2016	3	8	39	12	60	12	26	(7.203)
2016	3	8	30	15	63	12	27	(7.203)
2016	3	8	21	12	51	12	28	(7.203)
2016	3	8	24	15	51	12	29	(7.203)
2016	3	8	27	15	54	12	30	(7.203)
2016	3	8	24	18	51	12	31	(7.203)
2016	3	8	21	6	60	15	32	(7.203)
2016	3	8	27	9	63	15	33	(7.203)
2016	3	8	27	9	51	15	34	(7.203)
2016	3	8	24	12	54	15	35	(7.203)
2016	3	8	27	9	48	15	36	(7.203)
2016	3	8	33	12	48	15	37	(7.203)
2016	3	8	30	12	57	15	38	(7.203)
2016	3	8	30	15	63	15	39	(7.203)
2016	3	8	18	9	60	15	40	(7.203)
2016	3	8	18	12	63	15	41	(7.203)
2016	3	8	36	12	48	15	42	(7.203)
2016	3	8	30	15	54	15	43	(7.203)
2016	3	8	36	12	57	15	44	(7.203)
2016	3	8	39	15	63	15	45	(7.203)
2016	3	8	39	15	51	15	46	(7.203)
2016	3	8	42	18	54	15	47	(7.203)
2016	3	8	21	9	60	15	48	(7.203)
2016	3	8	24	12	63	15	49	(7.203)
2016	3	8	27	12	54	15	50	(7.203)
2016	3	8	21	15	51	15	51	(7.203)
2016	3	8	36	12	57	15	52	(7.203)
2016	3	8	39	15	57	15	53	(7.203)
2016	3	8	24	15	42	15	54	(7.203)
2016	3	8	36	18	42	15	55	(7.203)
2016	3	8	27	12	54	15	56	(7.203)
2016	3	8	27	15	51	15	57	(7.203)
2016	3	8	39	15	57	15	58	(7.203)
2016	3	8	33	18	57	15	59	(7.203)
2016	3	8	42	15	63	15	60	(7.203)
2016	3	8	33	18	51	15	61	(7.203)
2016	3	8	36	18	54	15	62	(7.203)
2016	3	8	39	21	63	15	63	(7.203)
2016	3	8	33	6	51	18	64	(7.203)
2016	3	8	39	9	54	18	65	(7.203)
2016	3	8	36	9	51	18	66	(7.203)
2016	3	8	36	12	51	18	67	(7.203)
2016	3	8	36	9	63	18	68	(7.203)
2016	3	8	39	12	60	18	69	(7.203)
2016	3	8	27	12	63	18	70	(7.203)
2016	3	8	18	15	48	18	71	(7.203)
2016	3	8	27	9	57	18	72	(7.203)
2016	3	8	27	12	63	18	73	(7.203)
2016	3	8	33	12	45	18	74	(7.203)
2016	3	8	33	15	42	18	75	(7.203)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	3	8	30	12	48	18	76	(7.203)
2016	3	8	36	15	54	18	77	(7.203)
2016	3	8	36	15	60	18	78	(7.203)
2016	3	8	39	18	63	18	79	(7.203)
2016	3	8	30	9	39	18	80	(7.203)
2016	3	8	30	12	39	18	81	(7.203)
2016	3	8	30	12	60	18	82	(7.203)
2016	3	8	30	15	63	18	83	(7.203)
2016	3	8	33	12	63	18	84	(7.203)
2016	3	8	36	15	54	18	85	(7.203)
2016	3	8	18	15	51	18	86	(7.203)
2016	3	8	18	18	54	18	87	(7.203)
2016	3	8	30	12	54	18	88	(7.203)
2016	3	8	30	15	63	18	89	(7.203)
2016	3	8	30	15	63	18	90	(7.203)
2016	3	8	27	18	60	18	91	(7.203)
2016	3	8	36	15	63	18	92	(7.203)
2016	3	8	30	18	51	18	93	(7.203)
2016	3	8	42	18	51	18	94	(7.203)
2016	3	8	33	21	54	18	95	(7.203)
2016	3	8	33	9	51	18	96	(7.203)
2016	3	8	42	12	54	18	97	(7.203)
2016	3	8	27	12	63	18	98	(7.203)
2016	3	8	30	15	60	18	99	(7.203)
2016	3	8	48	12	57	18	100	(7.203)
2016	3	8	42	15	57	18	101	(7.203)
2016	3	8	33	15	48	18	102	(7.203)
2016	3	8	33	18	54	18	103	(7.203)
2016	3	8	48	12	60	18	104	(7.203)
2016	3	8	51	15	63	18	105	(7.203)
2016	3	8	42	15	57	18	106	(7.203)
2016	3	8	36	18	57	18	107	(7.203)
2016	3	8	24	15	57	18	108	(7.203)
2016	3	8	36	18	63	18	109	(7.203)
2016	3	8	36	18	39	18	110	(7.203)
2016	3	8	30	21	39	18	111	(7.203)
2016	3	8	39	12	57	18	112	(7.203)
2016	3	8	45	15	63	18	113	(7.203)
2016	3	8	21	15	54	18	114	(7.203)
2016	3	8	30	18	63	18	115	(7.203)
2016	3	8	24	15	45	18	116	(7.203)
2016	3	8	30	18	45	18	117	(7.203)
2016	3	8	36	18	63	18	118	(7.203)
2016	3	8	36	21	60	18	119	(7.203)
2016	3	8	39	15	63	18	120	(7.203)
2016	3	8	39	18	60	18	121	(7.203)
2016	3	8	39	18	48	18	122	(7.203)
2016	3	8	51	21	63	18	123	(7.203)
2016	3	8	21	18	57	18	124	(7.203)
2016	3	8	30	21	57	18	125	(7.203)
2016	3	8	24	21	27	18	126	(7.203)
2016	3	8	21	24	27	18	127	(7.203)
2016	3	8	39	6	48	21	128	(7.203)
2016	3	8	36	9	54	21	129	(7.203)
2016	3	8	36	9	63	21	130	(7.203)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	3	8	39	12	54	21	131	(7.203)
2016	3	8	30	9	60	21	132	(7.203)
2016	3	8	27	12	63	21	133	(7.203)
2016	3	8	27	12	57	21	134	(7.203)
2016	3	8	24	15	57	21	135	(7.203)
2016	3	8	30	9	54	21	136	(7.203)
2016	3	8	30	12	51	21	137	(7.203)
2016	3	8	21	12	33	21	138	(7.203)
2016	3	8	21	15	33	21	139	(7.203)
2016	3	8	33	12	48	21	140	(7.203)
2016	3	8	33	15	48	21	141	(7.203)
2016	3	8	42	15	45	21	142	(7.203)
2016	3	8	57	18	63	21	143	(7.203)
2016	3	8	33	9	54	21	144	(7.203)
2016	3	8	48	12	63	21	145	(7.203)
2016	3	8	21	12	36	21	146	(7.203)
2016	3	8	30	15	36	21	147	(7.203)
2016	3	8	36	12	51	21	148	(7.203)
2016	3	8	36	15	51	21	149	(7.203)
2016	3	8	33	15	45	21	150	(7.203)
2016	3	8	33	18	54	21	151	(7.203)
2016	3	8	42	12	51	21	152	(7.203)
2016	3	8	48	15	63	21	153	(7.203)
2016	3	8	33	15	51	21	154	(7.203)
2016	3	8	24	18	51	21	155	(7.203)
2016	3	8	33	15	63	21	156	(7.203)
2016	3	8	27	18	51	21	157	(7.203)
2016	3	8	36	18	57	21	158	(7.203)
2016	3	8	30	21	57	21	159	(7.203)
2016	3	8	36	9	54	21	160	(7.203)
2016	3	8	33	12	63	21	161	(7.203)
2016	3	8	39	12	57	21	162	(7.203)
2016	3	8	33	15	57	21	163	(7.203)
2016	3	8	42	12	57	21	164	(7.203)
2016	3	8	42	15	63	21	165	(7.203)
2016	3	8	24	15	54	21	166	(7.203)
2016	3	8	21	18	42	21	167	(7.203)
2016	3	8	48	12	63	21	168	(7.203)
2016	3	8	48	15	57	21	169	(7.203)
2016	3	8	24	15	39	21	170	(7.203)
2016	3	8	27	18	45	21	171	(7.203)
2016	3	8	27	15	63	21	172	(7.203)
2016	3	8	18	18	51	21	173	(7.203)
2016	3	8	27	18	39	21	174	(7.203)
2016	3	8	27	21	39	21	175	(7.203)
2016	3	8	27	12	57	21	176	(7.203)
2016	3	8	30	15	57	21	177	(7.203)
2016	3	8	33	15	63	21	178	(7.203)
2016	3	8	30	18	60	21	179	(7.203)
2016	3	8	33	15	45	21	180	(7.203)
2016	3	8	30	18	45	21	181	(7.203)
2016	3	8	24	18	45	21	182	(7.203)
2016	3	8	21	21	45	21	183	(7.203)
2016	3	8	27	15	63	21	184	(7.203)
2016	3	8	24	18	60	21	185	(7.203)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	3	8	36	18	42	21	186	(7.203)
2016	3	8	45	21	54	21	187	(7.203)
2016	3	8	33	18	39	21	188	(7.203)
2016	3	8	33	21	39	21	189	(7.203)
2016	3	8	24	21	45	21	190	(7.203)
2016	3	8	24	24	45	21	191	(7.203)
2016	3	8	27	9	48	21	192	(7.203)
2016	3	8	33	12	54	21	193	(7.203)
2016	3	8	21	12	63	21	194	(7.203)
2016	3	8	21	15	57	21	195	(7.203)
2016	3	8	45	12	60	21	196	(7.203)
2016	3	8	45	15	63	21	197	(7.203)
2016	3	8	39	15	51	21	198	(7.203)
2016	3	8	42	18	51	21	199	(7.203)
2016	3	8	27	12	51	21	200	(7.203)
2016	3	8	36	15	51	21	201	(7.203)
2016	3	8	24	15	48	21	202	(7.203)
2016	3	8	27	18	48	21	203	(7.203)
2016	3	8	36	15	63	21	204	(7.203)
2016	3	8	45	18	60	21	205	(7.203)
2016	3	8	24	18	63	21	206	(7.203)
2016	3	8	27	21	57	21	207	(7.203)
2016	3	8	24	12	51	21	208	(7.203)
2016	3	8	27	15	54	21	209	(7.203)
2016	3	8	36	15	51	21	210	(7.203)
2016	3	8	45	18	51	21	211	(7.203)
2016	3	8	45	15	63	21	212	(7.203)
2016	3	8	36	18	60	21	213	(7.203)
2016	3	8	21	18	21	21	214	(7.203)
2016	3	8	18	21	21	21	215	(7.203)
2016	3	8	24	15	48	21	216	(7.203)
2016	3	8	36	18	54	21	217	(7.203)
2016	3	8	30	18	36	21	218	(7.203)
2016	3	8	27	21	36	21	219	(7.203)
2016	3	8	33	18	63	21	220	(7.203)
2016	3	8	27	21	51	21	221	(7.203)
2016	3	8	18	21	54	21	222	(7.203)
2016	3	8	18	24	63	21	223	(7.203)
2016	3	8	27	12	42	21	224	(7.203)
2016	3	8	39	15	45	21	225	(7.203)
2016	3	8	30	15	45	21	226	(7.203)
2016	3	8	27	18	45	21	227	(7.203)
2016	3	8	36	15	51	21	228	(7.203)
2016	3	8	39	18	51	21	229	(7.203)
2016	3	8	27	18	45	21	230	(7.203)
2016	3	8	30	21	45	21	231	(7.203)
2016	3	8	39	15	63	21	232	(7.203)
2016	3	8	33	18	60	21	233	(7.203)
2016	3	8	39	18	57	21	234	(7.203)
2016	3	8	33	21	57	21	235	(7.203)
2016	3	8	30	18	54	21	236	(7.203)
2016	3	8	21	21	51	21	237	(7.203)
2016	3	8	33	21	63	21	238	(7.203)
2016	3	8	33	24	60	21	239	(7.203)
2016	3	8	18	15	33	21	240	(7.203)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	3	8	21	18	33	21	241	(7.203)
2016	3	8	48	18	57	21	242	(7.203)
2016	3	8	48	21	57	21	243	(7.203)
2016	3	8	21	18	45	21	244	(7.203)
2016	3	8	30	21	63	21	245	(7.203)
2016	3	8	30	21	45	21	246	(7.203)
2016	3	8	27	24	45	21	247	(7.203)
2016	3	8	27	18	60	21	248	(7.203)
2016	3	8	33	21	63	21	249	(7.203)
2016	3	8	39	21	63	21	250	(7.203)
2016	3	8	42	24	60	21	251	(7.203)
2016	3	8	24	21	60	21	252	(7.203)
2016	3	8	27	24	63	21	253	(7.203)
2016	3	8	33	24	60	21	254	(7.203)
2016	3	8	36	27	63	21	255	(7.203)
2016	4	4	20	24	16	24	2	(7.204)
2016	4	4	24	28	16	24	3	(7.204)
2016	4	4	20	28	20	28	4	(7.204)
2016	4	4	20	28	20	28	5	(7.204)
2016	4	4	16	32	20	40	9	(7.204)
2016	4	4	20	24	16	40	11	(7.204)
2016	4	4	24	36	16	40	13	(7.204)
2016	4	4	24	44	20	40	14	(7.204)
2016	4	8	24	60	28	100	2	(7.204)
2016	4	8	32	56	32	84	3	(7.204)
2016	4	8	24	68	36	96	4	(7.204)
2016	4	8	28	48	32	84	5	(7.204)
2016	4	8	24	52	36	84	6	(7.204)
2016	4	8	36	76	28	112	7	(7.204)
2016	4	8	24	72	32	112	8	(7.204)
2016	4	8	36	80	36	112	9	(7.204)
2016	4	8	32	60	36	92	11	(7.204)
2016	4	8	24	56	32	84	12	(7.204)
2016	4	8	28	52	32	96	13	(7.204)
2016	4	8	24	60	40	104	14	(7.204)
2016	4	8	32	64	28	96	15	(7.204)
2016	4	8	20	48	44	100	16	(7.204)
2016	4	8	20	48	44	88	17	(7.204)
2016	4	8	36	64	44	104	18	(7.204)
2016	4	8	24	68	44	104	19	(7.204)
2016	4	8	28	56	44	96	20	(7.204)
2016	4	8	36	72	36	112	21	(7.204)
2016	4	8	36	68	44	104	22	(7.204)
2016	4	8	40	88	44	112	23	(7.204)
2016	4	8	24	72	28	112	24	(7.204)
2016	4	8	36	56	44	100	25	(7.204)
2016	4	8	28	60	44	104	26	(7.204)
2016	4	8	32	56	32	100	27	(7.204)
2016	4	8	36	60	40	88	28	(7.204)
2016	4	8	32	48	32	76	29	(7.204)
2016	4	8	28	52	44	104	30	(7.204)
2016	4	8	32	48	36	92	31	(7.204)
2016	4	8	28	56	48	96	32	(7.204)
2016	4	8	40	56	48	104	33	(7.204)
2016	4	8	20	36	40	72	34	(7.204)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	4	8	32	56	48	108	35	(7.204)
2016	4	8	32	68	48	100	36	(7.204)
2016	4	8	28	60	48	104	37	(7.204)
2016	4	8	24	44	40	84	38	(7.204)
2016	4	8	40	52	44	88	39	(7.204)
2016	4	8	24	52	48	92	40	(7.204)
2016	4	8	32	52	48	84	41	(7.204)
2016	4	8	24	60	44	100	42	(7.204)
2016	4	8	32	44	48	96	43	(7.204)
2016	4	8	24	64	44	112	44	(7.204)
2016	4	8	36	52	44	96	45	(7.204)
2016	4	8	36	64	44	104	46	(7.204)
2016	4	8	48	44	48	96	47	(7.204)
2016	4	8	28	52	48	72	48	(7.204)
2016	4	8	28	64	44	104	49	(7.204)
2016	4	8	36	56	48	112	50	(7.204)
2016	4	8	24	72	40	96	51	(7.204)
2016	4	8	40	56	44	84	52	(7.204)
2016	4	8	44	60	48	96	53	(7.204)
2016	4	8	40	48	48	88	54	(7.204)
2016	4	8	32	64	48	104	55	(7.204)
2016	4	8	28	36	48	92	56	(7.204)
2016	4	8	32	68	44	80	57	(7.204)
2016	4	8	36	80	48	92	58	(7.204)
2016	4	8	36	56	40	100	59	(7.204)
2016	4	8	28	68	40	112	60	(7.204)
2016	4	8	32	48	44	92	61	(7.204)
2016	4	8	32	68	32	80	62	(7.204)
2016	4	8	48	72	48	108	63	(7.204)
2016	4	8	32	60	48	92	64	(7.204)
2016	4	8	24	84	52	112	65	(7.204)
2016	4	8	40	64	52	108	66	(7.204)
2016	4	8	32	64	52	84	67	(7.204)
2016	4	8	32	68	52	108	69	(7.204)
2016	4	8	36	68	48	108	70	(7.204)
2016	4	8	24	64	40	92	71	(7.204)
2016	4	8	28	52	48	108	72	(7.204)
2016	4	8	32	60	48	104	73	(7.204)
2016	4	8	28	76	44	104	74	(7.204)
2016	4	8	32	60	48	104	75	(7.204)
2016	4	8	24	52	36	88	76	(7.204)
2016	4	8	36	68	52	104	77	(7.204)
2016	4	8	40	60	52	80	78	(7.204)
2016	4	8	36	56	36	64	79	(7.204)
2016	4	8	24	68	44	104	80	(7.204)
2016	4	8	32	76	48	100	81	(7.204)
2016	4	8	40	60	52	112	82	(7.204)
2016	4	8	32	60	48	92	83	(7.204)
2016	4	8	24	76	40	96	84	(7.204)
2016	4	8	36	64	48	100	85	(7.204)
2016	4	8	36	68	48	112	86	(7.204)
2016	4	8	40	68	52	84	87	(7.204)
2016	4	8	24	60	40	112	88	(7.204)
2016	4	8	48	76	52	108	89	(7.204)
2016	4	8	40	44	52	104	90	(7.204)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	4	8	44	72	52	108	91	(7.204)
2016	4	8	36	56	40	100	92	(7.204)
2016	4	8	36	80	48	104	93	(7.204)
2016	4	8	44	68	48	104	94	(7.204)
2016	4	8	44	64	52	96	95	(7.204)
2016	4	8	24	44	48	96	96	(7.204)
2016	4	8	32	56	52	100	97	(7.204)
2016	4	8	32	52	52	92	98	(7.204)
2016	4	8	36	64	52	92	99	(7.204)
2016	4	8	36	52	44	96	100	(7.204)
2016	4	8	40	76	52	100	101	(7.204)
2016	4	8	24	72	36	112	102	(7.204)
2016	4	8	40	64	52	92	103	(7.204)
2016	4	8	32	60	52	108	104	(7.204)
2016	4	8	28	48	48	92	105	(7.204)
2016	4	8	52	56	52	96	106	(7.204)
2016	4	8	52	72	48	104	107	(7.204)
2016	4	8	36	76	48	108	108	(7.204)
2016	4	8	48	52	52	80	109	(7.204)
2016	4	8	32	60	44	108	110	(7.204)
2016	4	8	48	60	48	104	111	(7.204)
2016	4	8	28	48	44	92	112	(7.204)
2016	4	8	32	72	52	100	113	(7.204)
2016	4	8	32	64	40	104	114	(7.204)
2016	4	8	32	68	52	96	115	(7.204)
2016	4	8	44	72	52	104	116	(7.204)
2016	4	8	44	56	52	108	117	(7.204)
2016	4	8	40	60	52	88	118	(7.204)
2016	4	8	48	52	52	84	119	(7.204)
2016	4	8	32	60	52	100	120	(7.204)
2016	4	8	40	36	52	96	121	(7.204)
2016	4	8	32	72	40	100	122	(7.204)
2016	4	8	52	56	44	108	123	(7.204)
2016	4	8	32	60	28	108	124	(7.204)
2016	4	8	48	64	52	100	125	(7.204)
2016	4	8	48	52	52	100	126	(7.204)
2016	4	8	56	64	52	112	127	(7.204)
2016	4	8	20	56	52	104	128	(7.204)
2016	4	8	20	68	56	100	129	(7.204)
2016	4	8	28	56	52	108	130	(7.204)
2016	4	8	28	60	52	100	131	(7.204)
2016	4	8	32	60	56	104	132	(7.204)
2016	4	8	32	64	56	108	133	(7.204)
2016	4	8	36	72	56	100	134	(7.204)
2016	4	8	40	60	52	84	135	(7.204)
2016	4	8	24	52	56	72	136	(7.204)
2016	4	8	28	92	52	104	137	(7.204)
2016	4	8	28	68	52	108	138	(7.204)
2016	4	8	32	76	48	108	139	(7.204)
2016	4	8	36	56	56	88	140	(7.204)
2016	4	8	40	72	56	108	141	(7.204)
2016	4	8	24	68	36	104	142	(7.204)
2016	4	8	40	72	56	104	143	(7.204)
2016	4	8	32	52	52	92	144	(7.204)
2016	4	8	32	64	56	104	145	(7.204)

Year	Dim.	m	#xor	Inv. #xor	#xtime	Inv. #xtime	α	Instanced Matrix
2016	4	8	32	56	56	96	147	(7.204)
2016	4	8	32	72	52	108	148	(7.204)
2016	4	8	32	56	48	104	149	(7.204)
2016	4	8	36	60	52	88	150	(7.204)
2016	4	8	28	60	56	92	151	(7.204)
2016	4	8	28	44	52	80	152	(7.204)
2016	4	8	36	52	56	84	153	(7.204)
2016	4	8	36	64	48	100	154	(7.204)
2016	4	8	40	84	44	108	155	(7.204)
2016	4	8	40	84	48	112	156	(7.204)
2016	4	8	32	60	52	100	157	(7.204)
2016	4	8	44	64	56	80	158	(7.204)
2016	4	8	44	64	52	112	159	(7.204)
2016	4	8	28	52	52	112	160	(7.204)
2016	4	8	32	72	44	108	161	(7.204)
2016	4	8	36	60	52	108	162	(7.204)
2016	4	8	36	52	56	80	163	(7.204)
2016	4	8	36	36	52	96	164	(7.204)
2016	4	8	40	52	56	92	165	(7.204)
2016	4	8	40	60	56	104	166	(7.204)
2016	4	8	32	56	44	92	167	(7.204)
2016	4	8	28	52	56	96	168	(7.204)
2016	4	8	32	44	48	76	169	(7.204)
2016	4	8	32	64	52	104	170	(7.204)
2016	4	8	40	36	48	64	171	(7.204)
2016	4	8	40	64	52	88	172	(7.204)
2016	4	8	28	84	52	96	173	(7.204)
2016	4	8	40	68	48	96	174	(7.204)
2016	4	8	52	60	52	104	175	(7.204)
2016	4	8	24	48	36	96	176	(7.204)
2016	4	8	44	60	56	92	177	(7.204)
2016	4	8	48	64	56	80	178	(7.204)
2016	4	8	48	68	56	108	179	(7.204)
2016	4	8	48	60	56	96	180	(7.204)
2016	4	8	44	64	56	104	181	(7.204)
2016	4	8	44	48	48	88	182	(7.204)
2016	4	8	40	68	48	96	183	(7.204)
2016	4	8	44	56	56	96	184	(7.204)
2016	4	8	40	52	56	88	185	(7.204)
2016	4	8	36	76	44	104	186	(7.204)
2016	4	8	44	56	56	96	187	(7.204)
2016	4	8	44	68	44	108	188	(7.204)
2016	4	8	40	52	52	84	189	(7.204)
2016	4	8	44	52	48	104	190	(7.204)
2016	4	8	52	60	52	92	191	(7.204)
2016	4	8	24	64	44	100	192	(7.204)
2016	4	8	28	56	56	100	193	(7.204)
2016	4	8	36	60	52	100	194	(7.204)
2016	4	8	36	68	52	104	195	(7.204)
2016	4	8	28	72	56	112	196	(7.204)
2016	4	8	36	56	48	84	197	(7.204)
2016	4	8	32	64	52	92	198	(7.204)
2016	4	8	40	68	52	84	199	(7.204)
2016	4	8	36	44	56	100	200	(7.204)
2016	4	8	32	72	56	108	201	(7.204)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	4	8	44	64	56	92	202	(7.204)
2016	4	8	36	68	56	96	203	(7.204)
2016	4	8	24	56	32	96	204	(7.204)
2016	4	8	48	68	56	104	205	(7.204)
2016	4	8	40	76	48	100	206	(7.204)
2016	4	8	44	64	48	88	207	(7.204)
2016	4	8	32	76	48	104	208	(7.204)
2016	4	8	36	64	56	104	209	(7.204)
2016	4	8	40	56	56	104	210	(7.204)
2016	4	8	40	76	56	88	211	(7.204)
2016	4	8	44	52	56	84	212	(7.204)
2016	4	8	36	72	56	108	213	(7.204)
2016	4	8	36	44	44	92	216	(7.204)
2016	4	8	36	60	56	88	217	(7.204)
2016	4	8	36	64	56	92	218	(7.204)
2016	4	8	44	52	56	96	219	(7.204)
2016	4	8	44	80	52	112	220	(7.204)
2016	4	8	44	72	52	108	222	(7.204)
2016	4	8	48	64	40	88	223	(7.204)
2016	4	8	28	52	40	100	224	(7.204)
2016	4	8	36	36	48	68	225	(7.204)
2016	4	8	44	56	56	96	226	(7.204)
2016	4	8	48	52	56	92	227	(7.204)
2016	4	8	32	76	36	108	228	(7.204)
2016	4	8	48	48	52	104	229	(7.204)
2016	4	8	44	36	56	88	230	(7.204)
2016	4	8	48	48	56	100	231	(7.204)
2016	4	8	44	48	56	100	232	(7.204)
2016	4	8	40	48	56	88	233	(7.204)
2016	4	8	40	56	56	112	234	(7.204)
2016	4	8	44	56	52	88	235	(7.204)
2016	4	8	40	52	56	80	236	(7.204)
2016	4	8	48	64	52	96	237	(7.204)
2016	4	8	52	68	52	112	238	(7.204)
2016	4	8	60	68	56	100	239	(7.204)
2016	4	8	44	52	56	92	240	(7.204)
2016	4	8	52	64	56	84	241	(7.204)
2016	4	8	44	56	52	92	242	(7.204)
2016	4	8	52	56	56	104	243	(7.204)
2016	4	8	32	52	36	88	244	(7.204)
2016	4	8	40	72	52	112	245	(7.204)
2016	4	8	44	60	56	96	246	(7.204)
2016	4	8	56	80	56	104	247	(7.204)
2016	4	8	40	40	56	104	248	(7.204)
2016	4	8	48	80	56	112	249	(7.204)
2016	4	8	48	36	56	96	250	(7.204)
2016	4	8	52	64	56	100	251	(7.204)
2016	4	8	48	76	56	88	252	(7.204)
2016	4	8	48	48	56	96	253	(7.204)
2016	4	8	48	48	56	92	254	(7.204)
2016	4	8	56	76	56	108	255	(7.204)
2016	5	4	25	55	45	45	8	(7.205)
2016	5	4	45	30	45	45	10	(7.205)
2016	5	4	30	45	45	45	12	(7.205)
2016	5	4	55	25	45	45	15	(7.205)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	5	8	35	100	40	140	2	(7.205)
2016	5	8	50	115	45	150	3	(7.205)
2016	5	8	35	70	55	150	4	(7.205)
2016	5	8	45	65	50	110	5	(7.205)
2016	5	8	40	65	55	120	6	(7.205)
2016	5	8	60	90	45	110	7	(7.205)
2016	5	8	35	100	55	170	8	(7.205)
2016	5	8	55	105	60	150	9	(7.205)
2016	5	8	40	75	65	100	10	(7.205)
2016	5	8	40	105	55	170	12	(7.205)
2016	5	8	50	110	55	170	13	(7.205)
2016	5	8	45	90	65	150	14	(7.205)
2016	5	8	60	125	50	150	15	(7.205)
2016	5	8	30	115	75	130	16	(7.205)
2016	5	8	35	60	75	100	17	(7.205)
2016	5	8	55	80	75	150	18	(7.205)
2016	5	8	45	65	75	140	19	(7.205)
2016	5	8	45	80	75	135	20	(7.205)
2016	5	8	60	75	65	135	21	(7.205)
2016	5	8	60	95	75	145	22	(7.205)
2016	5	8	70	90	75	145	23	(7.205)
2016	5	8	40	90	55	150	24	(7.205)
2016	5	8	60	95	75	160	25	(7.205)
2016	5	8	50	135	75	170	26	(7.205)
2016	5	8	60	80	60	160	27	(7.205)
2016	5	8	60	85	70	120	28	(7.205)
2016	5	8	60	50	60	160	29	(7.205)
2016	5	8	55	70	75	120	30	(7.205)
2016	5	8	65	85	65	110	31	(7.205)
2016	5	8	40	115	85	170	32	(7.205)
2016	5	8	60	90	85	150	33	(7.205)
2016	5	8	35	100	75	150	34	(7.205)
2016	5	8	55	105	85	150	35	(7.205)
2016	5	8	50	80	85	170	36	(7.205)
2016	5	8	50	65	85	140	37	(7.205)
2016	5	8	45	95	75	170	38	(7.205)
2016	5	8	70	50	80	130	39	(7.205)
2016	5	8	40	80	85	145	40	(7.205)
2016	5	8	55	75	85	125	41	(7.205)
2016	5	8	45	85	80	155	42	(7.205)
2016	5	8	60	90	85	155	43	(7.205)
2016	5	8	45	105	80	145	44	(7.205)
2016	5	8	65	80	80	135	45	(7.205)
2016	5	8	65	80	80	165	46	(7.205)
2016	5	8	85	95	85	165	47	(7.205)
2016	5	8	45	75	85	150	48	(7.205)
2016	5	8	50	70	80	140	49	(7.205)
2016	5	8	60	90	85	110	50	(7.205)
2016	5	8	50	55	75	170	51	(7.205)
2016	5	8	65	100	80	160	52	(7.205)
2016	5	8	75	105	85	150	53	(7.205)
2016	5	8	70	85	85	160	54	(7.205)
2016	5	8	65	80	85	150	55	(7.205)
2016	5	8	50	60	85	155	56	(7.205)
2016	5	8	60	125	80	155	57	(7.205)

Year	Dim.	m	#xor	Inv. #xor	#xtime	Inv. #xtime	α	Instanced Matrix
2016	5	8	65	65	85	155	58	(7.205)
2016	5	8	70	70	75	165	59	(7.205)
2016	5	8	55	105	75	165	60	(7.205)
2016	5	8	65	60	80	135	61	(7.205)
2016	5	8	65	110	65	135	62	(7.205)
2016	5	8	90	125	85	155	63	(7.205)
2016	5	8	45	90	90	155	64	(7.205)
2016	5	8	40	85	95	165	65	(7.205)
2016	5	8	60	115	95	165	66	(7.205)
2016	5	8	55	70	95	125	67	(7.205)
2016	5	8	35	65	75	105	68	(7.205)
2016	5	8	60	80	90	165	70	(7.205)
2016	5	8	50	95	80	145	71	(7.205)
2016	5	8	45	85	90	145	72	(7.205)
2016	5	8	55	60	90	145	73	(7.205)
2016	5	8	50	110	85	155	74	(7.205)
2016	5	8	60	95	90	155	75	(7.205)
2016	5	8	45	100	75	145	76	(7.205)
2016	5	8	65	75	95	165	77	(7.205)
2016	5	8	40	130	85	165	80	(7.205)
2016	5	8	55	95	90	155	81	(7.205)
2016	5	8	65	45	95	135	82	(7.205)
2016	5	8	60	110	90	155	83	(7.205)
2016	5	8	45	75	80	145	84	(7.205)
2016	5	8	65	110	90	175	85	(7.205)
2016	5	8	65	80	90	135	86	(7.205)
2016	5	8	75	85	95	165	87	(7.205)
2016	5	8	45	125	80	165	88	(7.205)
2016	5	8	80	130	95	175	89	(7.205)
2016	5	8	70	70	95	175	90	(7.205)
2016	5	8	80	75	95	175	91	(7.205)
2016	5	8	65	70	80	145	92	(7.205)
2016	5	8	70	75	90	175	93	(7.205)
2016	5	8	80	75	90	165	94	(7.205)
2016	5	8	85	60	95	165	95	(7.205)
2016	5	8	40	100	90	155	96	(7.205)
2016	5	8	55	85	95	145	97	(7.205)
2016	5	8	55	105	95	145	98	(7.205)
2016	5	8	65	150	95	175	99	(7.205)
2016	5	8	60	115	85	145	100	(7.205)
2016	5	8	70	90	95	135	101	(7.205)
2016	5	8	50	90	75	175	102	(7.205)
2016	5	8	75	105	95	155	103	(7.205)
2016	5	8	55	115	95	145	104	(7.205)
2016	5	8	55	90	90	125	105	(7.205)
2016	5	8	85	120	95	125	106	(7.205)
2016	5	8	90	105	90	125	107	(7.205)
2016	5	8	65	90	90	105	108	(7.205)
2016	5	8	85	55	95	145	109	(7.205)
2016	5	8	65	105	85	145	110	(7.205)
2016	5	8	90	130	90	155	111	(7.205)
2016	5	8	50	80	85	175	112	(7.205)
2016	5	8	60	125	95	175	113	(7.205)
2016	5	8	60	45	80	125	114	(7.205)
2016	5	8	65	70	95	155	115	(7.205)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	5	8	75	145	95	175	116	(7.205)
2016	5	8	80	110	95	165	117	(7.205)
2016	5	8	75	130	95	175	118	(7.205)
2016	5	8	90	75	95	145	119	(7.205)
2016	5	8	60	75	95	175	120	(7.205)
2016	5	8	75	100	95	165	121	(7.205)
2016	5	8	65	120	80	165	122	(7.205)
2016	5	8	95	65	85	145	123	(7.205)
2016	5	8	65	130	65	175	124	(7.205)
2016	5	8	90	125	95	165	125	(7.205)
2016	5	8	90	65	95	125	126	(7.205)
2016	5	8	105	100	95	165	127	(7.205)
2016	5	8	30	80	100	150	128	(7.205)
2016	5	8	35	105	105	150	129	(7.205)
2016	5	8	45	125	100	160	130	(7.205)
2016	5	8	50	90	100	130	131	(7.205)
2016	5	8	50	85	105	145	132	(7.205)
2016	5	8	55	110	105	145	133	(7.205)
2016	5	8	60	70	105	120	134	(7.205)
2016	5	8	70	95	100	140	135	(7.205)
2016	5	8	40	85	105	130	136	(7.205)
2016	5	8	50	90	100	130	137	(7.205)
2016	5	8	50	90	100	170	138	(7.205)
2016	5	8	60	85	95	130	139	(7.205)
2016	5	8	60	70	105	110	140	(7.205)
2016	5	8	70	95	105	140	141	(7.205)
2016	5	8	50	85	80	150	142	(7.205)
2016	5	8	75	70	105	120	143	(7.205)
2016	5	8	50	110	100	135	144	(7.205)
2016	5	8	55	95	105	155	145	(7.205)
2016	5	8	65	35	105	75	146	(7.205)
2016	5	8	55	75	100	90	148	(7.205)
2016	5	8	60	100	95	100	149	(7.205)
2016	5	8	65	110	100	150	150	(7.205)
2016	5	8	60	95	105	140	151	(7.205)
2016	5	8	65	120	95	150	154	(7.205)
2016	5	8	75	95	90	150	155	(7.205)
2016	5	8	70	100	95	130	156	(7.205)
2016	5	8	65	55	100	120	157	(7.205)
2016	5	8	80	105	105	160	158	(7.205)
2016	5	8	85	130	100	150	159	(7.205)
2016	5	8	45	90	100	160	160	(7.205)
2016	5	8	55	95	90	150	161	(7.205)
2016	5	8	60	145	100	170	162	(7.205)
2016	5	8	65	100	105	140	163	(7.205)
2016	5	8	60	105	100	160	164	(7.205)
2016	5	8	70	150	105	170	165	(7.205)
2016	5	8	70	90	105	140	166	(7.205)
2016	5	8	65	95	90	140	167	(7.205)
2016	5	8	50	35	105	125	168	(7.205)
2016	5	8	60	90	95	155	169	(7.205)
2016	5	8	60	100	100	145	170	(7.205)
2016	5	8	75	85	95	165	171	(7.205)
2016	5	8	70	100	100	145	172	(7.205)
2016	5	8	60	85	100	135	173	(7.205)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	5	8	75	95	95	155	174	(7.205)
2016	5	8	95	70	100	135	175	(7.205)
2016	5	8	45	150	80	170	176	(7.205)
2016	5	8	75	85	105	130	177	(7.205)
2016	5	8	80	95	105	140	178	(7.205)
2016	5	8	85	130	105	140	179	(7.205)
2016	5	8	80	95	105	170	180	(7.205)
2016	5	8	80	110	105	170	181	(7.205)
2016	5	8	80	120	95	170	182	(7.205)
2016	5	8	80	95	95	170	183	(7.205)
2016	5	8	75	115	105	155	184	(7.205)
2016	5	8	75	100	105	145	185	(7.205)
2016	5	8	70	110	90	165	186	(7.205)
2016	5	8	85	135	105	165	187	(7.205)
2016	5	8	80	90	90	135	188	(7.205)
2016	5	8	80	105	100	155	189	(7.205)
2016	5	8	85	75	95	155	190	(7.205)
2016	5	8	100	90	100	155	191	(7.205)
2016	5	8	40	115	90	165	192	(7.205)
2016	5	8	50	110	105	155	193	(7.205)
2016	5	8	60	110	100	165	194	(7.205)
2016	5	8	65	115	100	115	195	(7.205)
2016	5	8	50	100	105	135	196	(7.205)
2016	5	8	65	105	95	175	197	(7.205)
2016	5	8	60	105	100	165	198	(7.205)
2016	5	8	75	130	100	155	199	(7.205)
2016	5	8	60	80	105	165	200	(7.205)
2016	5	8	60	105	105	175	201	(7.205)
2016	5	8	75	145	105	175	202	(7.205)
2016	5	8	70	140	105	155	203	(7.205)
2016	5	8	50	105	75	165	204	(7.205)
2016	5	8	85	60	105	125	205	(7.205)
2016	5	8	75	70	95	165	206	(7.205)
2016	5	8	85	95	95	125	207	(7.205)
2016	5	8	55	115	95	175	208	(7.205)
2016	5	8	65	100	105	125	209	(7.205)
2016	5	8	70	120	105	175	210	(7.205)
2016	5	8	75	105	105	165	211	(7.205)
2016	5	8	75	90	105	145	212	(7.205)
2016	5	8	70	95	105	175	213	(7.205)
2016	5	8	65	90	90	155	216	(7.205)
2016	5	8	70	85	105	165	217	(7.205)
2016	5	8	70	105	105	175	218	(7.205)
2016	5	8	85	60	105	125	219	(7.205)
2016	5	8	75	40	100	65	221	(7.205)
2016	5	8	85	130	100	175	222	(7.205)
2016	5	8	95	85	85	145	223	(7.205)
2016	5	8	50	115	85	175	224	(7.205)
2016	5	8	65	100	95	165	225	(7.205)
2016	5	8	75	90	105	135	226	(7.205)
2016	5	8	85	105	105	145	227	(7.205)
2016	5	8	60	140	80	165	228	(7.205)
2016	5	8	85	145	100	165	229	(7.205)
2016	5	8	80	95	105	125	230	(7.205)
2016	5	8	90	140	105	135	231	(7.205)

Year	Dim.	m	#xor	Inv. #xor	#xtime	Inv. #xtime	α	Instanced Matrix
2016	5	8	75	70	105	105	232	(7.205)
2016	5	8	75	65	105	125	233	(7.205)
2016	5	8	75	85	105	135	234	(7.205)
2016	5	8	85	60	100	165	235	(7.205)
2016	5	8	75	85	105	175	236	(7.205)
2016	5	8	90	110	100	165	237	(7.205)
2016	5	8	95	110	100	165	238	(7.205)
2016	5	8	110	145	105	175	239	(7.205)
2016	5	8	75	95	105	155	240	(7.205)
2016	5	8	90	110	105	165	241	(7.205)
2016	5	8	80	100	100	175	242	(7.205)
2016	5	8	95	75	105	125	243	(7.205)
2016	5	8	65	100	80	175	244	(7.205)
2016	5	8	80	95	100	155	245	(7.205)
2016	5	8	85	75	105	175	246	(7.205)
2016	5	8	105	90	105	175	247	(7.205)
2016	5	8	75	110	105	165	248	(7.205)
2016	5	8	90	125	105	155	249	(7.205)
2016	5	8	90	105	105	105	250	(7.205)
2016	5	8	100	130	105	165	251	(7.205)
2016	5	8	90	115	105	145	252	(7.205)
2016	5	8	95	80	105	125	253	(7.205)
2016	5	8	95	60	105	175	254	(7.205)
2016	5	8	110	115	105	175	255	(7.205)
2016	6	8	102	138	108	210	3	(7.206)
2016	6	8	96	126	126	228	5	(7.206)
2016	6	8	90	144	126	246	6	(7.206)
2016	6	8	114	138	120	234	7	(7.206)
2016	6	8	90	108	120	186	8	(7.206)
2016	6	8	114	174	132	222	9	(7.206)
2016	6	8	102	126	120	222	12	(7.206)
2016	6	8	96	144	132	234	13	(7.206)
2016	6	8	108	156	132	240	14	(7.206)
2016	6	8	84	162	114	234	15	(7.206)
2016	6	8	72	162	144	216	17	(7.206)
2016	6	8	108	156	150	228	18	(7.206)
2016	6	8	84	156	144	210	19	(7.206)
2016	6	8	84	168	150	252	20	(7.206)
2016	6	8	120	150	126	210	21	(7.206)
2016	6	8	96	144	144	228	22	(7.206)
2016	6	8	108	108	144	162	23	(7.206)
2016	6	8	78	138	120	216	24	(7.206)
2016	6	8	120	126	138	222	25	(7.206)
2016	6	8	96	180	144	222	26	(7.206)
2016	6	8	96	132	132	192	27	(7.206)
2016	6	8	108	150	144	216	28	(7.206)
2016	6	8	96	162	132	228	30	(7.206)
2016	6	8	96	120	120	228	31	(7.206)
2016	6	8	84	126	138	204	32	(7.206)
2016	6	8	126	108	150	192	33	(7.206)
2016	6	8	72	120	126	180	34	(7.206)
2016	6	8	84	132	132	204	35	(7.206)
2016	6	8	78	108	150	174	36	(7.206)
2016	6	8	72	150	132	234	38	(7.206)
2016	6	8	84	126	144	192	40	(7.206)

Year	Dim.	m	#xor	Inv. #xor	#xtime	Inv. #xtime	α	Instanced Matrix
2016	6	8	96	126	150	210	41	(7.206)
2016	6	8	96	108	132	210	42	(7.206)
2016	6	8	84	150	150	216	43	(7.206)
2016	6	8	78	102	132	228	44	(7.206)
2016	6	8	102	120	138	180	46	(7.206)
2016	6	8	126	90	150	204	47	(7.206)
2016	6	8	96	150	144	234	48	(7.206)
2016	6	8	84	114	138	216	49	(7.206)
2016	6	8	90	144	120	216	51	(7.206)
2016	6	8	96	126	150	192	52	(7.206)
2016	6	8	126	138	156	210	53	(7.206)
2016	6	8	90	102	150	198	54	(7.206)
2016	6	8	96	168	150	234	55	(7.206)
2016	6	8	84	120	150	186	56	(7.206)
2016	6	8	84	126	138	234	57	(7.206)
2016	6	8	96	174	126	228	59	(7.206)
2016	6	8	108	138	144	204	60	(7.206)
2016	6	8	90	114	126	186	61	(7.206)
2016	6	8	108	144	108	216	62	(7.206)
2016	6	8	132	132	150	198	63	(7.206)
2016	6	8	96	96	132	204	64	(7.206)
2016	6	8	90	114	156	222	65	(7.206)
2016	6	8	132	108	162	186	66	(7.206)
2016	6	8	102	174	150	234	67	(7.206)
2016	6	8	96	126	156	228	70	(7.206)
2016	6	8	66	132	102	216	71	(7.206)
2016	6	8	90	150	144	216	72	(7.206)
2016	6	8	78	156	144	240	73	(7.206)
2016	6	8	96	144	138	234	74	(7.206)
2016	6	8	108	150	156	222	77	(7.206)
2016	6	8	96	138	138	240	80	(7.206)
2016	6	8	96	150	132	240	81	(7.206)
2016	6	8	108	108	150	174	82	(7.206)
2016	6	8	78	156	144	252	84	(7.206)
2016	6	8	90	162	132	240	85	(7.206)
2016	6	8	90	150	132	222	86	(7.206)
2016	6	8	108	120	156	192	87	(7.206)
2016	6	8	84	150	126	222	88	(7.206)
2016	6	8	120	162	150	246	89	(7.206)
2016	6	8	120	90	162	150	90	(7.206)
2016	6	8	120	162	150	234	91	(7.206)
2016	6	8	96	174	132	234	92	(7.206)
2016	6	8	108	156	132	240	93	(7.206)
2016	6	8	120	144	138	222	94	(7.206)
2016	6	8	90	120	144	204	96	(7.206)
2016	6	8	90	138	150	234	98	(7.206)
2016	6	8	126	120	156	228	99	(7.206)
2016	6	8	108	126	126	216	100	(7.206)
2016	6	8	84	126	120	192	102	(7.206)
2016	6	8	108	132	132	240	103	(7.206)
2016	6	8	84	126	150	198	104	(7.206)
2016	6	8	96	120	156	210	105	(7.206)
2016	6	8	150	114	150	186	106	(7.206)
2016	6	8	138	150	156	240	107	(7.206)
2016	6	8	78	132	132	210	108	(7.206)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	6	8	96	132	132	240	110	(7.206)
2016	6	8	108	162	144	240	111	(7.206)
2016	6	8	84	108	138	192	112	(7.206)
2016	6	8	84	108	138	192	113	(7.206)
2016	6	8	84	132	120	216	114	(7.206)
2016	6	8	96	138	162	204	115	(7.206)
2016	6	8	108	144	156	222	116	(7.206)
2016	6	8	126	114	162	186	117	(7.206)
2016	6	8	108	132	162	252	118	(7.206)
2016	6	8	126	132	138	240	119	(7.206)
2016	6	8	126	144	162	222	120	(7.206)
2016	6	8	120	156	150	228	121	(7.206)
2016	6	8	96	132	114	198	122	(7.206)
2016	6	8	120	144	144	222	123	(7.206)
2016	6	8	102	138	126	234	124	(7.206)
2016	6	8	126	114	156	210	125	(7.206)
2016	6	8	150	138	162	228	126	(7.206)
2016	6	8	126	150	156	240	127	(7.206)
2016	6	8	84	144	138	240	128	(7.206)
2016	6	8	84	156	162	246	129	(7.206)
2016	6	8	78	90	156	174	130	(7.206)
2016	6	8	102	144	144	228	131	(7.206)
2016	6	8	96	162	162	234	132	(7.206)
2016	6	8	108	132	168	210	134	(7.206)
2016	6	8	102	150	150	234	135	(7.206)
2016	6	8	78	132	150	246	137	(7.206)
2016	6	8	90	144	150	222	138	(7.206)
2016	6	8	90	162	132	234	139	(7.206)
2016	6	8	96	120	150	222	140	(7.206)
2016	6	8	120	144	162	234	141	(7.206)
2016	6	8	72	126	102	186	142	(7.206)
2016	6	8	120	120	162	216	143	(7.206)
2016	6	8	90	138	144	216	144	(7.206)
2016	6	8	108	174	156	252	145	(7.206)
2016	6	8	108	102	150	198	148	(7.206)
2016	6	8	108	144	156	246	149	(7.206)
2016	6	8	90	162	132	246	150	(7.206)
2016	6	8	102	132	162	210	151	(7.206)
2016	6	8	108	150	156	216	154	(7.206)
2016	6	8	102	120	138	222	156	(7.206)
2016	6	8	120	120	168	150	158	(7.206)
2016	6	8	96	138	150	186	159	(7.206)
2016	6	8	78	138	144	222	160	(7.206)
2016	6	8	90	132	144	234	161	(7.206)
2016	6	8	126	150	162	222	162	(7.206)
2016	6	8	102	132	156	246	163	(7.206)
2016	6	8	84	120	150	228	164	(7.206)
2016	6	8	90	150	162	228	165	(7.206)
2016	6	8	108	150	162	210	166	(7.206)
2016	6	8	78	156	102	228	167	(7.206)
2016	6	8	102	162	138	246	169	(7.206)
2016	6	8	84	126	108	210	170	(7.206)
2016	6	8	114	114	144	162	171	(7.206)
2016	6	8	108	138	156	234	172	(7.206)
2016	6	8	84	108	138	192	173	(7.206)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	6	8	126	132	156	198	174	(7.206)
2016	6	8	84	96	138	174	176	(7.206)
2016	6	8	102	144	144	234	177	(7.206)
2016	6	8	132	138	168	210	178	(7.206)
2016	6	8	108	132	138	216	179	(7.206)
2016	6	8	138	132	168	216	180	(7.206)
2016	6	8	120	126	162	222	181	(7.206)
2016	6	8	138	120	156	204	182	(7.206)
2016	6	8	120	126	150	216	183	(7.206)
2016	6	8	102	144	162	222	185	(7.206)
2016	6	8	102	168	120	222	186	(7.206)
2016	6	8	126	126	150	192	187	(7.206)
2016	6	8	120	114	150	180	188	(7.206)
2016	6	8	108	132	156	240	189	(7.206)
2016	6	8	114	168	150	228	190	(7.206)
2016	6	8	144	138	162	186	191	(7.206)
2016	6	8	78	156	132	198	192	(7.206)
2016	6	8	90	168	162	240	193	(7.206)
2016	6	8	96	168	150	222	194	(7.206)
2016	6	8	90	138	138	204	195	(7.206)
2016	6	8	90	120	150	204	196	(7.206)
2016	6	8	108	102	144	222	197	(7.206)
2016	6	8	96	126	162	228	198	(7.206)
2016	6	8	102	120	162	240	200	(7.206)
2016	6	8	120	108	156	198	201	(7.206)
2016	6	8	108	144	156	234	202	(7.206)
2016	6	8	78	78	126	174	204	(7.206)
2016	6	8	126	126	150	192	205	(7.206)
2016	6	8	102	138	150	210	206	(7.206)
2016	6	8	120	138	132	198	207	(7.206)
2016	6	8	96	162	132	246	208	(7.206)
2016	6	8	108	138	150	210	209	(7.206)
2016	6	8	132	114	168	192	211	(7.206)
2016	6	8	126	114	168	186	212	(7.206)
2016	6	8	96	120	126	186	216	(7.206)
2016	6	8	108	132	150	180	217	(7.206)
2016	6	8	102	138	150	228	218	(7.206)
2016	6	8	120	150	156	198	219	(7.206)
2016	6	8	114	144	138	222	222	(7.206)
2016	6	8	108	150	132	240	223	(7.206)
2016	6	8	72	126	126	216	224	(7.206)
2016	6	8	120	96	144	192	225	(7.206)
2016	6	8	108	144	162	240	226	(7.206)
2016	6	8	96	138	138	228	228	(7.206)
2016	6	8	126	144	162	210	229	(7.206)
2016	6	8	102	144	156	252	230	(7.206)
2016	6	8	138	138	162	234	231	(7.206)
2016	6	8	96	150	150	216	233	(7.206)
2016	6	8	108	168	156	240	235	(7.206)
2016	6	8	120	174	162	252	236	(7.206)
2016	6	8	114	186	150	234	237	(7.206)
2016	6	8	138	186	162	240	239	(7.206)
2016	6	8	120	120	168	222	240	(7.206)
2016	6	8	144	144	168	216	241	(7.206)
2016	6	8	120	162	144	222	242	(7.206)

Year	Dim.	m	#xor	Inv. #xor	#xtime	Inv. #xtime	α	Instanced Matrix
2016	6	8	138	120	162	222	243	(7.206)
2016	6	8	84	168	102	234	244	(7.206)
2016	6	8	108	114	162	240	245	(7.206)
2016	6	8	120	120	156	210	246	(7.206)
2016	6	8	138	114	162	204	247	(7.206)
2016	6	8	102	114	162	192	248	(7.206)
2016	6	8	120	126	168	186	249	(7.206)
2016	6	8	114	150	168	210	250	(7.206)
2016	6	8	120	180	162	234	251	(7.206)
2016	6	8	138	180	150	234	252	(7.206)
2016	6	8	138	156	156	234	253	(7.206)
2016	6	8	144	150	156	246	255	(7.206)
2016	7	8	84	147	112	182	2	(7.207)
2016	7	8	119	182	126	301	3	(7.207)
2016	7	8	84	105	154	189	4	(7.207)
2016	7	8	105	133	140	238	5	(7.207)
2016	7	8	91	168	154	287	6	(7.207)
2016	7	8	140	182	126	259	7	(7.207)
2016	7	8	84	133	154	203	8	(7.207)
2016	7	8	133	273	168	322	9	(7.207)
2016	7	8	91	154	154	245	12	(7.207)
2016	7	8	112	140	154	315	13	(7.207)
2016	7	8	133	168	140	238	15	(7.207)
2016	7	8	91	126	182	217	16	(7.207)
2016	7	8	84	133	182	259	17	(7.207)
2016	7	8	140	203	182	273	18	(7.207)
2016	7	8	91	175	182	343	19	(7.207)
2016	7	8	112	161	175	273	20	(7.207)
2016	7	8	133	182	147	287	21	(7.207)
2016	7	8	133	224	175	301	22	(7.207)
2016	7	8	140	140	175	203	23	(7.207)
2016	7	8	112	210	140	287	24	(7.207)
2016	7	8	147	154	196	273	25	(7.207)
2016	7	8	119	210	196	301	26	(7.207)
2016	7	8	147	168	182	280	28	(7.207)
2016	7	8	126	112	154	224	29	(7.207)
2016	7	8	112	189	196	315	30	(7.207)
2016	7	8	119	161	168	245	31	(7.207)
2016	7	8	168	182	210	287	33	(7.207)
2016	7	8	84	133	182	231	34	(7.207)
2016	7	8	126	238	210	273	36	(7.207)
2016	7	8	119	182	210	301	37	(7.207)
2016	7	8	91	147	182	315	38	(7.207)
2016	7	8	154	280	196	322	39	(7.207)
2016	7	8	98	182	203	259	40	(7.207)
2016	7	8	91	217	189	294	42	(7.207)
2016	7	8	91	175	189	336	44	(7.207)
2016	7	8	140	189	189	308	45	(7.207)
2016	7	8	175	175	203	245	47	(7.207)
2016	7	8	119	189	210	329	48	(7.207)
2016	7	8	154	182	210	287	50	(7.207)
2016	7	8	105	196	182	315	51	(7.207)
2016	7	8	168	189	196	252	52	(7.207)
2016	7	8	175	224	210	343	54	(7.207)
2016	7	8	119	168	189	322	57	(7.207)

Year	Dim.	m	#xor	Inv. #xor	#xtime	Inv. #xtime	α	Instanced Matrix
2016	7	8	147	203	203	315	58	(7.207)
2016	7	8	140	196	175	301	59	(7.207)
2016	7	8	119	175	175	287	60	(7.207)
2016	7	8	189	210	203	315	63	(7.207)
2016	7	8	140	133	217	168	64	(7.207)
2016	7	8	105	154	231	315	65	(7.207)
2016	7	8	161	210	231	315	66	(7.207)
2016	7	8	126	189	231	315	67	(7.207)
2016	7	8	154	182	217	336	70	(7.207)
2016	7	8	105	168	189	210	71	(7.207)
2016	7	8	119	161	217	294	72	(7.207)
2016	7	8	126	189	217	336	73	(7.207)
2016	7	8	119	140	217	336	75	(7.207)
2016	7	8	112	133	175	217	76	(7.207)
2016	7	8	147	161	231	287	77	(7.207)
2016	7	8	98	182	203	273	80	(7.207)
2016	7	8	133	203	217	322	81	(7.207)
2016	7	8	140	182	217	280	83	(7.207)
2016	7	8	140	147	217	280	85	(7.207)
2016	7	8	140	161	217	294	86	(7.207)
2016	7	8	161	203	231	315	87	(7.207)
2016	7	8	182	266	231	329	89	(7.207)
2016	7	8	154	224	231	301	90	(7.207)
2016	7	8	175	168	231	329	91	(7.207)
2016	7	8	126	189	189	266	92	(7.207)
2016	7	8	133	196	217	336	93	(7.207)
2016	7	8	161	182	217	266	94	(7.207)
2016	7	8	168	133	231	231	95	(7.207)
2016	7	8	112	210	217	308	96	(7.207)
2016	7	8	133	245	231	315	97	(7.207)
2016	7	8	154	217	231	315	99	(7.207)
2016	7	8	147	140	203	315	100	(7.207)
2016	7	8	154	224	231	343	101	(7.207)
2016	7	8	112	196	175	315	102	(7.207)
2016	7	8	126	217	217	336	105	(7.207)
2016	7	8	217	182	217	280	107	(7.207)
2016	7	8	154	175	217	336	108	(7.207)
2016	7	8	189	175	231	301	109	(7.207)
2016	7	8	147	161	203	329	110	(7.207)
2016	7	8	196	182	217	266	111	(7.207)
2016	7	8	112	182	203	231	112	(7.207)
2016	7	8	126	147	231	301	115	(7.207)
2016	7	8	182	182	231	329	117	(7.207)
2016	7	8	189	210	231	343	119	(7.207)
2016	7	8	168	231	231	301	121	(7.207)
2016	7	8	203	161	203	301	123	(7.207)
2016	7	8	203	266	231	301	125	(7.207)
2016	7	8	189	210	231	245	126	(7.207)
2016	7	8	224	217	231	315	127	(7.207)
2016	7	8	84	168	210	308	128	(7.207)
2016	7	8	77	175	224	259	129	(7.207)
2016	7	8	119	175	210	336	130	(7.207)
2016	7	8	112	210	210	280	131	(7.207)
2016	7	8	119	182	203	287	132	(7.207)
2016	7	8	112	133	203	175	133	(7.207)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	7	8	140	182	210	273	134	(7.207)
2016	7	8	147	182	196	336	135	(7.207)
2016	7	8	105	168	238	287	136	(7.207)
2016	7	8	126	210	224	322	138	(7.207)
2016	7	8	133	154	210	245	139	(7.207)
2016	7	8	140	196	238	301	140	(7.207)
2016	7	8	147	140	238	245	141	(7.207)
2016	7	8	105	126	168	280	142	(7.207)
2016	7	8	154	175	238	343	143	(7.207)
2016	7	8	133	245	217	343	145	(7.207)
2016	7	8	133	189	210	308	148	(7.207)
2016	7	8	140	154	196	203	149	(7.207)
2016	7	8	140	189	210	266	150	(7.207)
2016	7	8	161	175	196	294	155	(7.207)
2016	7	8	168	140	210	231	156	(7.207)
2016	7	8	147	147	224	210	157	(7.207)
2016	7	8	175	252	238	301	158	(7.207)
2016	7	8	182	217	224	252	159	(7.207)
2016	7	8	133	203	196	294	161	(7.207)
2016	7	8	147	210	224	308	162	(7.207)
2016	7	8	161	182	224	308	164	(7.207)
2016	7	8	168	210	238	329	166	(7.207)
2016	7	8	119	252	231	343	168	(7.207)
2016	7	8	126	224	203	301	169	(7.207)
2016	7	8	147	147	203	189	171	(7.207)
2016	7	8	119	224	217	308	173	(7.207)
2016	7	8	161	189	203	315	174	(7.207)
2016	7	8	196	203	217	294	175	(7.207)
2016	7	8	189	217	238	301	177	(7.207)
2016	7	8	203	196	238	301	178	(7.207)
2016	7	8	210	273	238	343	179	(7.207)
2016	7	8	182	210	238	301	181	(7.207)
2016	7	8	182	196	210	315	182	(7.207)
2016	7	8	175	182	210	301	183	(7.207)
2016	7	8	175	175	231	273	184	(7.207)
2016	7	8	168	203	231	301	185	(7.207)
2016	7	8	189	245	231	329	187	(7.207)
2016	7	8	168	210	189	294	188	(7.207)
2016	7	8	210	154	217	238	191	(7.207)
2016	7	8	140	182	245	329	193	(7.207)
2016	7	8	154	182	231	294	194	(7.207)
2016	7	8	161	231	231	308	195	(7.207)
2016	7	8	126	196	245	329	196	(7.207)
2016	7	8	161	147	217	217	197	(7.207)
2016	7	8	168	175	231	280	199	(7.207)
2016	7	8	154	224	245	315	200	(7.207)
2016	7	8	147	196	245	343	201	(7.207)
2016	7	8	154	154	245	273	203	(7.207)
2016	7	8	105	182	161	273	204	(7.207)
2016	7	8	196	133	245	301	205	(7.207)
2016	7	8	154	189	217	329	206	(7.207)
2016	7	8	175	161	217	259	207	(7.207)
2016	7	8	133	147	217	315	208	(7.207)
2016	7	8	140	217	245	343	209	(7.207)
2016	7	8	168	168	245	273	210	(7.207)

Year	**Dim.**	m	**#xor**	**Inv. #xor**	**#xtime**	**Inv. #xtime**	α	**Instanced Matrix**
2016	7	8	147	154	245	315	213	(7.207)
2016	7	8	140	175	203	252	216	(7.207)
2016	7	8	133	224	245	329	217	(7.207)
2016	7	8	147	210	245	315	218	(7.207)
2016	7	8	168	175	245	301	219	(7.207)
2016	7	8	189	189	189	301	223	(7.207)
2016	7	8	154	147	217	231	225	(7.207)
2016	7	8	182	245	245	329	226	(7.207)
2016	7	8	203	217	245	329	227	(7.207)
2016	7	8	203	217	231	294	229	(7.207)
2016	7	8	210	140	245	343	231	(7.207)
2016	7	8	182	217	245	301	232	(7.207)
2016	7	8	175	175	245	287	233	(7.207)
2016	7	8	175	161	245	343	234	(7.207)
2016	7	8	196	175	231	266	235	(7.207)
2016	7	8	175	147	245	287	236	(7.207)
2016	7	8	210	182	231	266	237	(7.207)
2016	7	8	224	231	231	294	238	(7.207)
2016	7	8	259	182	245	329	239	(7.207)
2016	7	8	189	189	245	287	240	(7.207)
2016	7	8	210	238	245	273	241	(7.207)
2016	7	8	182	168	231	322	242	(7.207)
2016	7	8	203	189	245	259	243	(7.207)
2016	7	8	161	140	231	294	245	(7.207)
2016	7	8	210	245	245	287	247	(7.207)
2016	7	8	182	175	245	315	248	(7.207)
2016	7	8	203	168	245	217	250	(7.207)
2016	7	8	210	182	245	315	251	(7.207)
2016	7	8	203	224	245	315	252	(7.207)
2016	7	8	196	217	245	273	254	(7.207)
2016	7	8	217	238	245	343	255	(7.207)
2016	8	8	184	216	248	336	30	(7.208)
2016	8	8	216	240	312	416	67	(7.208)
2016	8	8	128	248	296	384	73	(7.208)
2016	8	8	216	248	264	376	90	(7.208)
2016	8	8	168	264	304	408	140	(7.208)
2016	8	8	200	208	288	296	148	(7.208)
2016	8	8	208	288	336	432	158	(7.208)
2016	8	8	208	272	320	440	200	(7.208)

Table 7.13: Instancing Beierle's generic MDS matrices with different α and m

Year	Dim.	m	#xor	Inv. # xor	#xtime	Inv. #xtime	Inst. Mat.	Irreducible Poly.
2018	3	2	6	9	3	3	(7.188)	x^2+x+1
2018	3	4	6	21	3	21	(7.188)	x^4+x+1
2018	3	4	6	21	3	21	(7.188)	x^4+x^3+1
2018	3	4	6	21	3	21	(7.188)	$x^4+x^3+x^2+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^4+x^3+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^4+x^3+x^2+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^5+x^3+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^5+x^3+x^2+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^5+x^4+x^3+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^5+x^4+x^3+x^2+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^6+x^3+x^2+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^6+x^4+x^3+x^2+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^6+x^5+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^6+x^5+x^2+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^6+x^5+x^3+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^6+x^5+x^4+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^6+x^5+x^4+x^2+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^6+x^5+x^4+x^3+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^2+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^3+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^3+x^2+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^4+x^3+x^2+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^5+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^5+x^3+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^5+x^4+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^5+x^4+x^3+x^2+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^6+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^6+x^3+x^2+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^6+x^4+x^2+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^6+x^4+x^3+x^2+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^6+x^5+x^2+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^6+x^5+x^4+x+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^6+x^5+x^4+x^2+1$
2018	3	8	6	39	3	57	(7.188)	$x^8+x^7+x^6+x^5+x^4+x^3+1$
2018	3	2	9	12	9	9	(7.189)	x^2+x+1
2018	3	4	9	30	9	27	(7.189)	x^4+x+1
2018	3	4	9	30	9	27	(7.189)	x^4+x^3+1
2018	3	4	9	30	9	27	(7.189)	$x^4+x^3+x^2+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^4+x^3+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^4+x^3+x^2+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^5+x^3+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^5+x^3+x^2+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^5+x^4+x^3+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^5+x^4+x^3+x^2+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^6+x^3+x^2+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^6+x^4+x^3+x^2+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^6+x^5+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^6+x^5+x^2+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^6+x^5+x^3+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^6+x^5+x^4+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^6+x^5+x^4+x^2+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^6+x^5+x^4+x^3+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^2+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^3+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^3+x^2+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^4+x^3+x^2+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^5+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^5+x^3+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^5+x^4+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^5+x^4+x^3+x^2+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^6+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^6+x^3+x^2+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^6+x^4+x^2+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^6+x^4+x^3+x^2+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^6+x^5+x^2+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^6+x^5+x^4+x+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^6+x^5+x^4+x^2+1$
2018	3	8	9	48	9	63	(7.189)	$x^8+x^7+x^6+x^5+x^4+x^3+1$
2018	3	2	8	8	4	4	(7.190)	x^2+x+1
2018	3	4	8	8	4	4	(7.190)	x^4+x+1
2018	3	4	8	8	4	4	(7.190)	x^4+x^3+1
2018	3	4	8	8	4	4	(7.190)	$x^4+x^3+x^2+x+1$

Year	Dim.	m	#xor	Inv. # xor	#xtime	Inv. #xtime	Inst. Mat.	Irreducible Poly.
2018	3	8	8	8	4	4	(7.190)	$x^8+x^4+x^3+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^4+x^3+x^2+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^5+x^3+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^5+x^3+x^2+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^5+x^4+x^3+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^5+x^4+x^3+x^2+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^6+x^3+x^2+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^6+x^4+x^3+x^2+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^6+x^5+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^6+x^5+x^2+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^6+x^5+x^3+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^6+x^5+x^4+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^6+x^5+x^4+x^2+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^6+x^5+x^4+x^3+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^2+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^3+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^3+x^2+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^4+x^3+x^2+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^5+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^5+x^3+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^5+x^4+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^5+x^4+x^3+x^2+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^6+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^6+x^3+x^2+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^6+x^4+x^2+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^6+x^4+x^3+x^2+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^6+x^5+x^2+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^6+x^5+x^4+x+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^6+x^5+x^4+x^2+1$
2018	3	8	8	8	4	4	(7.190)	$x^8+x^7+x^6+x^5+x^4+x^3+1$
2018	3	2	8	8	4	4	(7.191)	x^2+x+1
2018	3	4	8	8	4	4	(7.191)	x^4+x+1
2018	3	4	8	8	4	4	(7.191)	x^4+x^3+1
2018	3	4	8	8	4	4	(7.191)	$x^4+x^3+x^2+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^4+x^3+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^4+x^3+x^2+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^5+x^3+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^5+x^3+x^2+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^5+x^4+x^3+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^5+x^4+x^3+x^2+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^6+x^3+x^2+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^6+x^4+x^3+x^2+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^6+x^5+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^6+x^5+x^2+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^6+x^5+x^3+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^6+x^5+x^4+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^6+x^5+x^4+x^2+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^6+x^5+x^4+x^3+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^2+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^3+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^3+x^2+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^4+x^3+x^2+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^5+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^5+x^3+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^5+x^4+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^5+x^4+x^3+x^2+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^6+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^6+x^3+x^2+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^6+x^4+x^2+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^6+x^4+x^3+x^2+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^6+x^5+x^2+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^6+x^5+x^4+x+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^6+x^5+x^4+x^2+1$
2018	3	8	8	8	4	4	(7.191)	$x^8+x^7+x^6+x^5+x^4+x^3+1$
2018	3	2	8	8	4	4	(7.192)	x^2+x+1
2018	3	4	8	16	4	16	(7.192)	x^4+x+1
2018	3	4	8	16	4	16	(7.192)	x^4+x^3+1
2018	3	4	8	16	4	16	(7.192)	$x^4+x^3+x^2+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^4+x^3+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^4+x^3+x^2+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^5+x^3+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^5+x^3+x^2+1$

Year	**Dim.**	m	**#xor**	**Inv. # xor**	**#xtime**	**Inv. #xtime**	**Inst. Mat.**	**Irreducible Poly.**
2018	3	8	8	28	4	40	(7.192)	$x^8+x^5+x^4+x^3+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^5+x^4+x^3+x^2+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^6+x^3+x^2+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^6+x^4+x^3+x^2+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^6+x^5+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^6+x^5+x^2+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^6+x^5+x^3+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^6+x^5+x^4+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^6+x^5+x^4+x^2+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^6+x^5+x^4+x^3+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^2+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^3+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^3+x^2+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^4+x^3+x^2+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^5+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^5+x^3+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^5+x^4+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^5+x^4+x^3+x^2+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^6+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^6+x^3+x^2+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^6+x^4+x^2+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^6+x^4+x^3+x^2+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^6+x^5+x^2+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^6+x^5+x^4+x+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^6+x^5+x^4+x^2+1$
2018	3	8	8	28	4	40	(7.192)	$x^8+x^7+x^6+x^5+x^4+x^3+1$
2018	3	2	6	9	3	3	(7.193)	x^2+x+1
2018	3	4	6	21	3	21	(7.193)	x^4+x+1
2018	3	4	6	21	3	21	(7.193)	x^4+x^3+1
2018	3	4	6	21	3	21	(7.193)	$x^4+x^3+x^2+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^4+x^3+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^4+x^3+x^2+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^5+x^3+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^5+x^3+x^2+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^5+x^4+x^3+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^5+x^4+x^3+x^2+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^6+x^3+x^2+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^6+x^4+x^3+x^2+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^6+x^5+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^6+x^5+x^2+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^6+x^5+x^3+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^6+x^5+x^4+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^6+x^5+x^4+x^2+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^6+x^5+x^4+x^3+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^2+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^3+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^3+x^2+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^4+x^3+x^2+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^5+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^5+x^3+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^5+x^4+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^5+x^4+x^3+x^2+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^6+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^6+x^3+x^2+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^6+x^4+x^2+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^6+x^4+x^3+x^2+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^6+x^5+x^2+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^6+x^5+x^4+x+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^6+x^5+x^4+x^2+1$
2018	3	8	6	39	3	57	(7.193)	$x^8+x^7+x^6+x^5+x^4+x^3+1$
2018	3	4	18	18	16	22	(7.194)	x^4+x+1
2018	3	4	18	18	16	22	(7.194)	x^4+x^3+1
2018	3	4	18	18	16	22	(7.194)	$x^4+x^3+x^2+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^4+x^3+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^4+x^3+x^2+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^5+x^3+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^5+x^3+x^2+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^5+x^4+x^3+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^5+x^4+x^3+x^2+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^6+x^3+x^2+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^6+x^4+x^3+x^2+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^6+x^5+x+1$

Year	Dim.	m	#xor	Inv. # xor	#xtime	Inv. #xtime	Inst. Mat.	Irreducible Poly.
2018	3	8	18	30	16	38	(7.194)	$x^8+x^6+x^5+x^2+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^6+x^5+x^3+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^6+x^5+x^4+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^6+x^5+x^4+x^2+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^6+x^5+x^4+x^3+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^2+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^3+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^3+x^2+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^4+x^3+x^2+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^5+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^5+x^3+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^5+x^4+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^5+x^4+x^3+x^2+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^6+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^6+x^3+x^2+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^6+x^4+x^2+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^6+x^4+x^3+x^2+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^6+x^5+x^2+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^6+x^5+x^4+x+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^6+x^5+x^4+x^2+1$
2018	3	8	18	30	16	38	(7.194)	$x^8+x^7+x^6+x^5+x^4+x^3+1$
2018	3	4	20	26	14	41	(7.199)	x^4+x+1
2018	3	4	20	26	14	41	(7.199)	x^4+x^3+1
2018	3	4	20	26	14	41	(7.199)	$x^4+x^3+x^2+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^4+x^3+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^4+x^3+x^2+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^5+x^3+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^5+x^3+x^2+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^5+x^4+x^3+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^5+x^4+x^3+x^2+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^6+x^3+x^2+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^6+x^4+x^3+x^2+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^6+x^5+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^6+x^5+x^2+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^6+x^5+x^3+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^6+x^5+x^4+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^6+x^5+x^4+x^2+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^6+x^5+x^4+x^3+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^2+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^3+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^3+x^2+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^4+x^3+x^2+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^5+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^5+x^3+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^5+x^4+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^5+x^4+x^3+x^2+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^6+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^6+x^3+x^2+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^6+x^4+x^2+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^6+x^4+x^3+x^2+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^6+x^5+x^2+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^6+x^5+x^4+x+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^6+x^5+x^4+x^2+1$
2018	3	8	20	38	14	81	(7.199)	$x^8+x^7+x^6+x^5+x^4+x^3+1$
2018	3	4	18	18	16	22	(7.195)	x^4+x+1
2018	3	4	18	18	16	22	(7.195)	x^4+x^3+1
2018	3	4	18	18	16	22	(7.195)	$x^4+x^3+x^2+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^4+x^3+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^4+x^3+x^2+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^5+x^3+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^5+x^3+x^2+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^5+x^4+x^3+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^5+x^4+x^3+x^2+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^6+x^3+x^2+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^6+x^4+x^3+x^2+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^6+x^5+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^6+x^5+x^2+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^6+x^5+x^3+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^6+x^5+x^4+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^6+x^5+x^4+x^2+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^6+x^5+x^4+x^3+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^2+x+1$

Year	Dim.	m	#xor	Inv. # xor	#xtime	Inv. #xtime	Inst. Mat.	Irreducible Poly.
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^3+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^3+x^2+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^4+x^3+x^2+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^5+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^5+x^3+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^5+x^4+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^5+x^4+x^3+x^2+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^6+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^6+x^3+x^2+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^6+x^4+x^2+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^6+x^4+x^3+x^2+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^6+x^5+x^2+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^6+x^5+x^4+x+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^6+x^5+x^4+x^2+1$
2018	3	8	18	30	16	38	(7.195)	$x^8+x^7+x^6+x^5+x^4+x^3+1$
2018	3	4	22	38	18	40	(7.196)	x^4+x+1
2018	3	4	22	38	18	40	(7.196)	x^4+x^3+1
2018	3	4	22	38	18	40	(7.196)	$x^4+x^3+x^2+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^4+x^3+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^4+x^3+x^2+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^5+x^3+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^5+x^3+x^2+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^5+x^4+x^3+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^5+x^4+x^3+x^2+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^6+x^3+x^2+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^6+x^4+x^3+x^2+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^6+x^5+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^6+x^5+x^2+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^6+x^5+x^3+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^6+x^5+x^4+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^6+x^5+x^4+x^2+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^6+x^5+x^4+x^3+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^2+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^3+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^3+x^2+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^4+x^3+x^2+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^5+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^5+x^3+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^5+x^4+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^5+x^4+x^3+x^2+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^6+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^6+x^3+x^2+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^6+x^4+x^2+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^6+x^4+x^3+x^2+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^6+x^5+x^2+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^6+x^5+x^4+x+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^6+x^5+x^4+x^2+1$
2018	3	8	22	64	18	106	(7.196)	$x^8+x^7+x^6+x^5+x^4+x^3+1$
2018	3	4	22	40	16	40	(7.197)	x^4+x+1
2018	3	4	22	40	16	40	(7.197)	x^4+x^3+1
2018	3	4	22	40	16	40	(7.197)	$x^4+x^3+x^2+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^4+x^3+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^4+x^3+x^2+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^5+x^3+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^5+x^3+x^2+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^5+x^4+x^3+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^5+x^4+x^3+x^2+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^6+x^3+x^2+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^6+x^4+x^3+x^2+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^6+x^5+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^6+x^5+x^2+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^6+x^5+x^3+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^6+x^5+x^4+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^6+x^5+x^4+x^2+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^6+x^5+x^4+x^3+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^2+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^3+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^3+x^2+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^4+x^3+x^2+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^5+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^5+x^3+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^5+x^4+1$

Year	**Dim.**	m	**#xor**	**Inv. # xor**	**#xtime**	**Inv. #xtime**	**Inst. Mat.**	**Irreducible Poly.**
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^5+x^4+x^3+x^2+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^6+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^6+x^3+x^2+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^6+x^4+x^2+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^6+x^4+x^3+x^2+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^6+x^5+x^2+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^6+x^5+x^4+x+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^6+x^5+x^4+x^2+1$
2018	3	8	22	64	16	104	(7.197)	$x^8+x^7+x^6+x^5+x^4+x^3+1$

Table 7.14: Instancing Duwal's matrices with different finite fields and irreducible polynomials

7.16 Appendix 7B: Matrix List

Matrix (7.47) and its inverse (7.48) are used in the SHARK [62] cipher.

$$\begin{bmatrix} ce_x & 95_x & 57_x & 82_x & 8a_x & 19_x & b0_x & 01_x \\ e7_x & fe_x & 05_x & d2_x & 52_x & c1_x & 88_x & f1_x \\ b9_x & da_x & 4d_x & d1_x & 9e_x & 17_x & 83_x & 86_x \\ d0_x & 9d_x & 26_x & 2c_x & 5d_x & 9f_x & 6d_x & 75_x \\ 52_x & a9_x & 07_x & 6c_x & b9_x & 8f_x & 70_x & 17_x \\ 87_x & 28_x & 3a_x & 5a_x & f4_x & 33_x & 0b_x & 6c_x \\ 74_x & 51_x & 15_x & cf_x & 09_x & a4_x & 62_x & 09_x \\ 0b_x & 31_x & 7f_x & 86_x & be_x & 05_x & 83_x & 34_x \end{bmatrix} \tag{7.47}$$

$$\begin{bmatrix} e7_x & 30_x & 90_x & 85_x & d0_x & 4b_x & 91_x & 41_x \\ 53_x & 95_x & 9b_x & a5_x & 96_x & bc_x & a1_x & 68_x \\ 02_x & 45_x & f7_x & 65_x & 5c_x & 1f_x & b6_x & 52_x \\ a2_x & ca_x & 22_x & 94_x & 44_x & 63_x & 2a_x & a2_x \\ fc_x & 67_x & 8e_x & 10_x & 29_x & 75_x & 85_x & 71_x \\ 24_x & 45_x & a2_x & cf_x & 2f_x & 22_x & c1_x & 0e_x \\ a1_x & f1_x & 71_x & 40_x & 91_x & 27_x & 18_x & a5_x \\ 56_x & f4_x & af_x & 32_x & d2_x & a4_x & dc_x & 71_x \end{bmatrix} \tag{7.48}$$

Matrix (7.49) and its inverse (7.50) are used in the SQUARE [20] cipher. They are right-circulant.

$$\begin{bmatrix} 02_x & 01_x & 01_x & 03_x \\ 03_x & 02_x & 01_x & 01_x \\ 01_x & 03_x & 02_x & 01_x \\ 01_x & 01_x & 03_x & 02_x \end{bmatrix} \tag{7.49}$$

$$\begin{bmatrix} 0e_x & 09_x & 0d_x & 0b_x \\ 0b_x & 0e_x & 09_x & 0d_x \\ 0d_x & 0b_x & 0e_x & 09_x \\ 09_x & 0d_x & 0b_x & 0e_x \end{bmatrix} \tag{7.50}$$

Matrix (7.51) is involutory, and was obtained by [73] with a Cauchy construction.

$$\begin{bmatrix} 93_x & 13_x & 57_x & da_x & 58_x & 47_x & 0c_x & 1f_x \\ 13_x & 93_x & da_x & 57_x & 47_x & 58_x & 1f_x & 0c_x \\ 57_x & da_x & 93_x & 13_x & 0c_x & 1f_x & 58_x & 47_x \\ da_x & 57_x & 13_x & 93_x & 1f_x & 0c_x & 47_x & 58_x \\ 58_x & 47_x & 0c_x & 1f_x & 93_x & 13_x & 57_x & da_x \\ 47_x & 58_x & 1f_x & 0c_x & 13_x & 93_x & da_x & 57_x \\ 0c_x & 1f_x & 58_x & 47_x & 57_x & da_x & 93_x & 13_x \\ 1f_x & 0c_x & 47_x & 58_x & da_x & 57_x & 13_x & 93_x \end{bmatrix} \tag{7.51}$$

Matrix (7.52) is Hadamard and involutory. It is used in the KHAZAD [4] cipher.

$$\begin{bmatrix} 01_x & 03_x & 04_x & 05_x & 06_x & 08_x & 0b_x & 07_x \\ 03_x & 01_x & 05_x & 04_x & 08_x & 06_x & 07_x & 0b_x \\ 04_x & 05_x & 01_x & 03_x & 0b_x & 07_x & 06_x & 08_x \\ 05_x & 04_x & 03_x & 01_x & 07_x & 0b_x & 08_x & 06_x \\ 06_x & 08_x & 0b_x & 07_x & 01_x & 03_x & 04_x & 05_x \\ 08_x & 06_x & 07_x & 0b_x & 03_x & 01_x & 05_x & 04_x \\ 0b_x & 07_x & 06_x & 08_x & 04_x & 05_x & 01_x & 03_x \\ 07_x & 0b_x & 08_x & 06_x & 05_x & 04_x & 03_x & 01_x \end{bmatrix} \tag{7.52}$$

Matrix (7.53) is Hadamard and involutory. It is used in the ANUBIS [3] cipher.

$$\begin{bmatrix} 01_x & 02_x & 04_x & 06_x \\ 02_x & 01_x & 06_x & 04_x \\ 04_x & 06_x & 01_x & 02_x \\ 06_x & 04_x & 02_x & 01_x \end{bmatrix} \tag{7.53}$$

Still regarding the ANUBIS cipher, while (7.53) is used as its linear transformation layer, (7.54) is used in the key extraction. It is a Vandermonde construction. When $N = 4$, it is an MDS matrix (see Theorem 2). Matrix (7.55) is the inverse of (7.54).

$$\begin{bmatrix} 01_x & 01_x & 01_x & ... & 01_x \\ 01_x & 02_x & 02_x^2 & ... & 02_x^{N-1} \\ 01_x & 06_x & 06_x^2 & ... & 06_x^{N-1} \\ 01_x & 08_x & 08_x^2 & ... & 08_x^{N-1} \end{bmatrix} = \begin{bmatrix} 01_x & 01_x & 01_x & 01_x \\ 01_x & 02_x & 04_x & 08_x \\ 01_x & 06_x & 14_x & 78_x \\ 01_x & 08_x & 40_x & 3a_x \end{bmatrix} \text{for } N = 4 \tag{7.54}$$

$$\begin{bmatrix} 71_x & 53_x & 7c_x & 5f_x \\ 8c_x & 25_x & c3_x & 6a_x \\ a3_x & ad_x & 71_x & 7f_x \\ 5f_x & db_x & ce_x & 4a_x \end{bmatrix} \tag{7.55}$$

Matrix (7.56) and its inverse (7.57) are used in the Rijndael [25] cipher, which was selected to become AES. They are right-circulant. We show the hexadecimal notation and the corresponding polynomials to emphasize that, albeit stored as integers in cryptographic software implementation, all matrix elements are actually polynomials in a Finite Field. This applies not only to the Rijndael cipher's matrices but to all matrices listed in this work.

$$\begin{bmatrix} 02_x & 03_x & 01_x & 01_x \\ 01_x & 02_x & 03_x & 01_x \\ 01_x & 01_x & 02_x & 03_x \\ 03_x & 01_x & 01_x & 02_x \end{bmatrix} = \begin{bmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{bmatrix} \tag{7.56}$$

$$\begin{bmatrix} 0e_x & 0b_x & 0d_x & 09_x \\ 09_x & 0e_x & 0b_x & 0d_x \\ 0d_x & 09_x & 0e_x & 0b_x \\ 0b_x & 0d_x & 09_x & 0e_x \end{bmatrix} = \begin{bmatrix} x^3+x^2+x & x^3+x+1 & x^3+x^2+1 & x^3+1 \\ x^3+1 & x^3+x^2+x & x^3+x+1 & x^3+x^2+1 \\ x^3+x^2+1 & x^3+1 & x^3+x^2+x & x^3+x+1 \\ x^3+x+1 & x^3+x^2+1 & x^3+1 & x^3+x^2+x \end{bmatrix} \tag{7.57}$$

Furthermore, it is interesting to note that Rijndael's matrix (7.56) is the transpose of SQUARE's matrix (7.49) and this also happens to the inverses (matrix (7.50) is the transpose of (7.57)).

Matrices (7.58) and its inverse (7.59) are used in the BKSQ [21] cipher. They are right-circulant.

$$\begin{bmatrix} 03_x & 02_x & 02_x \\ 02_x & 03_x & 02_x \\ 02_x & 02_x & 03_x \end{bmatrix} \tag{7.58}$$

$$\begin{bmatrix} ac_x & ad_x & ad_x \\ ad_x & ac_x & ad_x \\ ad_x & ad_x & ac_x \end{bmatrix} \tag{7.59}$$

The Hierocrypt-3 cipher makes use of two MDS matrices, one for lower level diffusion and another for higher level diffusion on the cipher, which follows a nested Substitution Permutation Network design (for more detail the reader may refer to [17]). Matrix (7.60) and its inverse (7.61) are used for lower level diffusion. (7.62) and (7.63) (the inverse) are used for higher level diffusion. It is worth noting that, for lower level diffusion, the finite field is $GF(2^8)$, whilst, for higher level diffusion, the authors choose $GF(2^4)$.

$$\begin{bmatrix} c4_x & 65_x & c8_x & 8b_x \\ 8b_x & c4_x & 65_x & c8_x \\ c8_x & 8b_x & c4_x & 65_x \\ 65_x & c8_x & 8b_x & c4_x \end{bmatrix} \tag{7.60}$$

$$\begin{bmatrix} 82_x & c4_x & 34_x & f6_x \\ f6_x & 82_x & c4_x & 34_x \\ 34_x & f6_x & 82_x & c4_x \\ c4_x & 34_x & f6_x & 82_x \end{bmatrix} \tag{7.61}$$

$$\begin{bmatrix} 5_x & 5_x & a_x & e_x \\ e_x & 5_x & 5_x & a_x \\ a_x & e_x & 5_x & 5_x \\ 5_x & a_x & e_x & 5_x \end{bmatrix} \tag{7.62}$$

$$\begin{bmatrix} b_x & e_x & e_x & 6_x \\ 6_x & b_x & e_x & e_x \\ e_x & 6_x & b_x & e_x \\ e_x & e_x & 6_x & b_x \end{bmatrix} \tag{7.63}$$

The Hierocrypt-L1 cipher too uses matrix (7.60) and the inverse (7.61) in its lower diffusion layer. However, for the higher layer, (7.64) and (7.65) (inverse) are used. Analogously to Hierocrypt-3, the higher layer uses $GF(2^4)$.

$$\begin{bmatrix} 5_x & 7_x \\ a_x & b_x \end{bmatrix} \tag{7.64}$$

$$\begin{bmatrix} c_x & a_x \\ 5_x & b_x \end{bmatrix} \tag{7.65}$$

Matrices (7.66) (inverse: (7.68)) and (7.67) (inverse: (7.69)) are used in the FOX block cipher family, with $z = x^7+x^6+x^5+x^4+x^3+x^2+1, a = x+1, b = x^7+x, c = x, d = x^2, e = x^7+x^6+x^5+x^4+x^3+x^2$ and $f = x^6 + x^5 + x^4 + x^3 + x^2 + x$.

$$\begin{bmatrix} 1 & 1 & 1 & x \\ 1 & z & x & 1 \\ z & x & 1 & 1 \\ x & 1 & z & 1 \end{bmatrix} = \begin{bmatrix} 01_x & 01_x & 01_x & 02_x \\ 01_x & fd_x & 02_x & 01_x \\ fd_x & 02_x & 01_x & 01_x \\ 02_x & 01_x & fd_x & 01_x \end{bmatrix} \tag{7.66}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & a \\ 1 & a & b & c & d & e & f & 1 \\ a & b & c & d & e & f & 1 & 1 \\ b & c & d & e & f & 1 & a & 1 \\ c & d & e & f & 1 & a & b & 1 \\ d & e & f & 1 & a & b & c & 1 \\ e & f & 1 & a & b & c & d & 1 \\ f & 1 & a & b & c & d & e & 1 \end{bmatrix} = \begin{bmatrix} 01_x & 01_x & 01_x & 01_x & 01_x & 01_x & 01_x & 03_x \\ 01_x & 03_x & 82_x & 02_x & 04_x & fc_x & 7e_x & 01_x \\ 03_x & 82_x & 02_x & 04_x & fc_x & 7e_x & 01_x & 01_x \\ 82_x & 02_x & 04_x & fc_x & 7e_x & 01_x & 03_x & 01_x \\ 02_x & 04_x & fc_x & 7e_x & 01_x & 03_x & 82_x & 01_x \\ 04_x & fc_x & 7e_x & 01_x & 03_x & 82_x & 02_x & 01_x \\ fc_x & 7e_x & 01_x & 03_x & 82_x & 02_x & 04_x & 01_x \\ 7e_x & 01_x & 03_x & 82_x & 02_x & 04_x & fc_x & 01_x \end{bmatrix} \tag{7.67}$$

$$\begin{bmatrix} 7e_x & e1_x & ad_x & b0_x \\ 7e_x & ad_x & b0_x & e1_x \\ 7e_x & b0_x & e1_x & ad_x \\ c3_x & 7e_x & 7e_x & 7e_x \end{bmatrix} \tag{7.68}$$

$$\begin{bmatrix} c6_x & fe_x & 3a_x & 73_x & 6d_x & 0c_x & d2_x & b7_x \\ c6_x & 3a_x & 73_x & 6d_x & 0c_x & d2_x & b7_x & fe_x \\ c6_x & 73_x & 6d_x & 0c_x & d2_x & b7_x & fe_x & 3a_x \\ c6_x & 6d_x & 0c_x & d2_x & b7_x & fe_x & 3a_x & 73_x \\ c6_x & 0c_x & d2_x & b7_x & fe_x & 3a_x & 73_x & 6d_x \\ c6_x & d2_x & b7_x & fe_x & 3a_x & 73_x & 6d_x & 0c_x \\ c6_x & b7_x & fe_x & 3a_x & 73_x & 6d_x & 0c_x & d2_x \\ ea_x & c6_x & c6_x & c6_x & c6_x & c6_x & c6_x & c6_x \end{bmatrix} \tag{7.69}$$

Matrix (7.70) is used in Curupira's diffusion layer. It is involutory.

$$\begin{bmatrix} 03_x & 02_x & 02_x \\ 04_x & 05_x & 04_x \\ 06_x & 06_x & 07_x \end{bmatrix} \tag{7.70}$$

Matrix (7.71) is used in Curupira's key scheduling process, with $c(x) = x^4 + x^3 + x^2$.

$$\begin{bmatrix} 1+c(x) & c(x) & c(x) \\ c(x) & 1+c(x) & c(x) \\ c(x) & c(x) & 1+c(x) \end{bmatrix} = \begin{bmatrix} 1d_x & 1c_x & 1c_x \\ 1c_x & 1d_x & 1c_x \\ 1c_x & 1c_x & 1d_x \end{bmatrix} \tag{7.71}$$

Matrix (7.72) is the inverse of (7.71).

$$\begin{bmatrix} 1c_x & 1d_x & 1d_x \\ 1d_x & 1c_x & 1d_x \\ 1d_x & 1d_x & 1c_x \end{bmatrix} \tag{7.72}$$

Matrix (7.73) is used in the Grøstl hash function. It is right-circulant. Its inverse is (7.74).

$$\begin{bmatrix} 02_x & 02_x & 03_x & 04_x & 05_x & 03_x & 05_x & 07_x \\ 07_x & 02_x & 02_x & 03_x & 04_x & 05_x & 03_x & 05_x \\ 05_x & 07_x & 02_x & 02_x & 03_x & 04_x & 05_x & 03_x \\ 03_x & 05_x & 07_x & 02_x & 02_x & 03_x & 04_x & 05_x \\ 05_x & 03_x & 05_x & 07_x & 02_x & 02_x & 03_x & 04_x \\ 04_x & 05_x & 03_x & 05_x & 07_x & 02_x & 02_x & 03_x \\ 03_x & 04_x & 05_x & 03_x & 05_x & 07_x & 02_x & 02_x \\ 02_x & 03_x & 04_x & 05_x & 03_x & 05_x & 07_x & 02_x \end{bmatrix} \tag{7.73}$$

$$\begin{bmatrix} 13_x & 5a_x & 54_x & 72_x & 50_x & df_x & 45_x & 53_x \\ 53_x & 13_x & 5a_x & 54_x & 72_x & 50_x & df_x & 45_x \\ 45_x & 53_x & 13_x & 5a_x & 54_x & 72_x & 50_x & df_x \\ df_x & 45_x & 53_x & 13_x & 5a_x & 54_x & 72_x & 50_x \\ 50_x & df_x & 45_x & 53_x & 13_x & 5a_x & 54_x & 72_x \\ 72_x & 50_x & df_x & 45_x & 53_x & 13_x & 5a_x & 54_x \\ 54_x & 72_x & 50_x & df_x & 45_x & 53_x & 13_x & 5a_x \\ 5a_x & 54_x & 72_x & 50_x & df_x & 45_x & 53_x & 13_x \end{bmatrix} \tag{7.74}$$

Matrix (7.75) is used in the LED block cipher. Its inverse is matrix (7.76).

$$\begin{bmatrix} 4_x & 1_x & 2_x & 2_x \\ 8_x & 6_x & 5_x & 6_x \\ b_x & e_x & a_x & 9_x \\ 2_x & 2_x & f_x & b_x \end{bmatrix} = \begin{bmatrix} 0_x & 1_x & 0_x & 0_x \\ 0_x & 0_x & 1_x & 0_x \\ 0_x & 0_x & 0_x & 1_x \\ 4_x & 1_x & 2_x & 2_x \end{bmatrix}^4 = Companion(4_x, 1_x, 2_x, 2_x)^4 \tag{7.75}$$

$$\begin{bmatrix} c_x & c_x & d_x & 4_x \\ 3_x & 8_x & 4_x & 5_x \\ 7_x & 6_x & 2_x & e_x \\ d_x & 9_x & 9_x & d_x \end{bmatrix} \tag{7.76}$$

Matrices (7.77) to (7.88) refer to the different matrices employed in different flavors of the PHOTON hash function and the respective inverses.

$$\begin{bmatrix} 01_x & 02_x & 01_x & 04_x \\ 04_x & 09_x & 06_x & 11_x \\ 11_x & 26_x & 18_x & 42_x \\ 42_x & 95_x & 64_x & 0b_x \end{bmatrix} = Companion(01_x, 02_x, 01_x, 04_x) \tag{7.77}$$

$$\begin{bmatrix} 14_x & 1a_x & 35_x & 0c_x \\ 0c_x & 0c_x & 16_x & 05_x \\ 05_x & 06_x & 09_x & 02_x \\ 02_x & 01_x & 04_x & 01_x \end{bmatrix} \tag{7.78}$$

$$\begin{bmatrix} 1_x & 2_x & 9_x & 9_x & 2_x \\ 2_x & 5_x & 3_x & 8_x & d_x \\ d_x & b_x & a_x & c_x & 1_x \\ 1_x & f_x & 2_x & 3_x & e_x \\ e_x & e_x & 8_x & 5_x & c_x \end{bmatrix} = Companion(1_x, 2_x, 9_x, 9_x, 2_x)^5 \tag{7.79}$$

$$\begin{bmatrix} c_x & 5_x & 8_x & e_x & e_x \\ e_x & 3_x & 2_x & f_x & 1_x \\ 1_x & c_x & a_x & b_x & d_x \\ d_x & 8_x & 3_x & 5_x & 2_x \\ 2_x & 9_x & 9_x & 2_x & 1_x \end{bmatrix} \tag{7.80}$$

$$\begin{bmatrix} 1_x & 2_x & 8_x & 5_x & 8_x & 2_x \\ 2_x & 5_x & 1_x & 2_x & 6_x & c_x \\ c_x & 9_x & f_x & 8_x & 8_x & d_x \\ d_x & 5_x & b_x & 3_x & a_x & 1_x \\ 1_x & f_x & d_x & e_x & b_x & 8_x \\ 8_x & 2_x & 3_x & 3_x & 2_x & 8_x \end{bmatrix} = Companion(1_x, 2_x, 8_x, 5_x, 8_x, 2_x)^6 \tag{7.81}$$

$$\begin{bmatrix} 8_x & 2_x & 3_x & 3_x & 2_x & 8_x \\ 8_x & b_x & e_x & d_x & f_x & 1_x \\ 1_x & a_x & 3_x & b_x & 5_x & d_x \\ d_x & 8_x & 8_x & f_x & 9_x & c_x \\ c_x & 6_x & 2_x & 1_x & 5_x & 2_x \\ 2_x & 8_x & 5_x & 8_x & 2_x & 1_x \end{bmatrix} \tag{7.82}$$

$$\begin{bmatrix} 1_x & 4_x & 6_x & 1_x & 1_x & 6_x & 4_x \\ 4_x & 2_x & f_x & 2_x & 5_x & a_x & 5_x \\ 5_x & 3_x & f_x & a_x & 7_x & 8_x & d_x \\ d_x & 4_x & b_x & 2_x & 7_x & f_x & 9_x \\ 9_x & f_x & 7_x & 2_x & b_x & 4_x & d_x \\ d_x & 8_x & 7_x & a_x & f_x & 3_x & 5_x \\ 5_x & a_x & 5_x & 2_x & f_x & 2_x & 4_x \end{bmatrix} = Companion(1_x, 4_x, 6_x, 1_x, 1_x, 6_x, 4_x)^7 \tag{7.83}$$

$$\begin{bmatrix} 4_x & 2_x & f_x & 2_x & 5_x & a_x & 5_x \\ 5_x & 3_x & f_x & a_x & 7_x & 8_x & d_x \\ d_x & 4_x & b_x & 2_x & 7_x & f_x & 9_x \\ 9_x & f_x & 7_x & 2_x & b_x & 4_x & d_x \\ d_x & 8_x & 7_x & a_x & f_x & 3_x & 5_x \\ 5_x & a_x & 5_x & 2_x & f_x & 2_x & 4_x \\ 4_x & 6_x & 1_x & 1_x & 6_x & 4_x & 1_x \end{bmatrix} \tag{7.84}$$

$$\begin{bmatrix} 2_x & 4_x & 2_x & b_x & 2_x & 8_x & 5_x & 6_x \\ c_x & 9_x & 8_x & d_x & 7_x & 7_x & 5_x & 2_x \\ 4_x & 4_x & d_x & d_x & 9_x & 4_x & d_x & 9_x \\ 1_x & 6_x & 5_x & 1_x & c_x & d_x & f_x & e_x \\ f_x & c_x & 9_x & d_x & e_x & 5_x & e_x & d_x \\ 9_x & e_x & 5_x & f_x & 4_x & c_x & 9_x & 6_x \\ c_x & 2_x & 2_x & a_x & 3_x & 1_x & 1_x & e_x \\ f_x & 1_x & d_x & a_x & 5_x & a_x & 2_x & 3_x \end{bmatrix} = Companion(2_x, 4_x, 2_x, 11_x, 2_x, 8_x, 5_x, 6_x)^8 \tag{7.85}$$

$$\begin{bmatrix} 4_x & 7_x & 9_x & a_x & c_x & c_x & 3_x & f_x \\ d_x & d_x & a_x & a_x & 7_x & d_x & a_x & 7_x \\ e_x & 2_x & 3_x & e_x & 4_x & a_x & 5_x & b_x \\ 5_x & 4_x & 7_x & a_x & b_x & 3_x & b_x & a_x \\ 7_x & b_x & 3_x & 5_x & d_x & 4_x & 7_x & 2_x \\ 4_x & f_x & f_x & 6_x & 1_x & e_x & e_x & b_x \\ 5_x & e_x & a_x & 6_x & 3_x & 6_x & f_x & 1_x \\ 2_x & 1_x & c_x & 1_x & 4_x & b_x & 3_x & 9_x \end{bmatrix} \tag{7.86}$$

$$\begin{bmatrix} 02_x & 03_x & 01_x & 02_x & 01_x & 04_x \\ 08_x & 0e_x & 07_x & 09_x & 06_x & 11_x \\ 22_x & 3b_x & 1f_x & 25_x & 18_x & 42_x \\ 84_x & e4_x & 79_x & 9b_x & 67_x & 0b_x \\ 16_x & 99_x & ef_x & 6f_x & 90_x & 4b_x \\ 96_x & cb_x & d2_x & 79_x & 24_x & a7_x \end{bmatrix} = Companion(2_x, 3_x, 1_x, 2_x, 1_x, 4_x)^6 \tag{7.87}$$

$$\begin{bmatrix} 15_x & 50_x & eb_x & 62_x & 79_x & 99_x \\ 29_x & a5_x & c9_x & c2_x & fb_x & 2b_x \\ 56_x & 54_x & 8e_x & 9f_x & e9_x & 57_x \\ ae_x & af_x & 03_x & 20_x & c8_x & ae_x \\ 47_x & 47_x & 01_x & 44_x & 8e_x & 46_x \\ 8c_x & 8d_x & 01_x & 8d_x & 02_x & 8d_x \end{bmatrix} \tag{7.88}$$

Matrix (7.89) is used in the Joltik [39] cipher. It is involutory.

$$\begin{bmatrix} 1_x & 4_x & 9_x & d_x \\ 4_x & 1_x & d_x & 9_x \\ 9_x & d_x & 1_x & 4_x \\ d_x & 9_x & 4_x & 1_x \end{bmatrix} \tag{7.89}$$

Matrix (7.90) is shown in [19] as an example of compact Cauchy involutory MDS matrix.

$$\begin{bmatrix} 01_x & 12_x & 04_x & 16_x \\ 12_x & 01_x & 16_x & 04_x \\ 04_x & 16_x & 01_x & 12_x \\ 16_x & 04_x & 12_x & 01_x \end{bmatrix} \tag{7.90}$$

Matrices from (7.91) to (7.99) were found in [36].

$$\begin{bmatrix} 01_x & 02_x & 03_x & d0_x \\ 02_x & 01_x & d0_x & 03_x \\ 03_x & d0_x & 01_x & 02_x \\ d0_x & 03_x & 02_x & 01_x \end{bmatrix} \tag{7.91}$$

$$\begin{bmatrix} 98_x & 2b_x & b3_x & 7a_x \\ 2b_x & 98_x & 7a_x & b3_x \\ b3_x & 7a_x & 98_x & 2b_x \\ 7a_x & b3_x & 2b_x & 98_x \end{bmatrix} \tag{7.92}$$

$$\begin{bmatrix} 7a_x & f4_x & 8e_x & 01_x \\ f4_x & 7a_x & 01_x & 8e_x \\ 8e_x & 01_x & 7a_x & f4_x \\ 01_x & 8e_x & f4_x & 7a_x \end{bmatrix} \tag{7.93}$$

$$\begin{bmatrix} 7a_x & f4_x & 8e_x \\ f4_x & 7a_x & 01_x \\ 8e_x & 01_x & 7a_x \end{bmatrix} \tag{7.94}$$

$$\begin{bmatrix} aa_x & f7_x & 8c_x \\ f7_x & f3_x & 06_x \\ 8c_x & 06_x & 89_x \end{bmatrix} \tag{7.95}$$

$$\begin{bmatrix} 01_x & 02_x & fc_x & fe_x \\ 02_x & 01_x & fe_x & fc_x \\ fc_x & fe_x & 01_x & 02_x \\ fe_x & fc_x & 02_x & 01_x \end{bmatrix} \tag{7.96}$$

$$hc(01_x, 02_x, 06_x, 8c_x, 30_x, fb_x, 87_x, c4_x) = \begin{bmatrix} 01_x & 02_x & 06_x & 8c_x & 30_x & fb_x & 87_x & c4_x \\ 02_x & 01_x & 8c_x & 06_x & fb_x & 30_x & c4_x & 87_x \\ 06_x & 8c_x & 01_x & 02_x & 87_x & c4_x & 30_x & fb_x \\ 8c_x & 06_x & 02_x & 01_x & c4_x & 87_x & fb_x & 30_x \\ 30_x & fb_x & 87_x & c4_x & 01_x & 02_x & 06_x & 8c_x \\ fb_x & 30_x & c4_x & 87_x & 02_x & 01_x & 8c_x & 06_x \\ 87_x & c4_x & 30_x & fb_x & 06_x & 8c_x & 01_x & 02_x \\ c4_x & 87_x & fb_x & 30_x & 8c_x & 06_x & 02_x & 01_x \end{bmatrix} \tag{7.97}$$

$$\begin{bmatrix} 01_x & 03_x & 08_x & b2_x & 0d_x & 60_x & e8_x & 1c_x & 0f_x & 2c_x & a2_x & 8b_x & c9_x & 7a_x & ac_x & 35_x \\ 03_x & 01_x & b2_x & 08_x & 60_x & 0d_x & 1c_x & e8_x & 2c_x & 0f_x & 8b_x & a2_x & 7a_x & c9_x & 35_x & ac_x \\ 08_x & b2_x & 01_x & 03_x & e8_x & 1c_x & 0d_x & 60_x & a2_x & 8b_x & 0f_x & 2c_x & ac_x & 35_x & c9_x & 7a_x \\ b2_x & 08_x & 03_x & 01_x & 1c_x & e8_x & 60_x & 0d_x & 8b_x & a2_x & 2c_x & 0f_x & 35_x & ac_x & 7a_x & c9_x \\ 0d_x & 60_x & e8_x & 1c_x & 01_x & 03_x & 08_x & b2_x & c9_x & 7a_x & ac_x & 35_x & 0f_x & 2c_x & a2_x & 8b_x \\ 60_x & 0d_x & 1c_x & e8_x & 03_x & 01_x & b2_x & 08_x & 7a_x & c9_x & 35_x & ac_x & 2c_x & 0f_x & 8b_x & a2_x \\ e8_x & 1c_x & 0d_x & 60_x & 08_x & b2_x & 01_x & 03_x & ac_x & 35_x & c9_x & 7a_x & a2_x & 8b_x & 0f_x & 2c_x \\ 1c_x & e8_x & 60_x & 0d_x & b2_x & 08_x & 03_x & 01_x & 35_x & ac_x & 7a_x & c9_x & 8b_x & a2_x & 2c_x & 0f_x \\ 0f_x & 2c_x & a2_x & 8b_x & c9_x & 7a_x & ac_x & 35_x & 01_x & 03_x & 08_x & b2_x & 0d_x & 60_x & e8_x & 1c_x \\ 2c_x & 0f_x & 8b_x & a2_x & 7a_x & c9_x & 35_x & ac_x & 03_x & 01_x & b2_x & 08_x & 60_x & 0d_x & 1c_x & e8_x \\ a2_x & 8b_x & 0f_x & 2c_x & ac_x & 35_x & c9_x & 7a_x & 08_x & b2_x & 01_x & 03_x & e8_x & 1c_x & 0d_x & 60_x \\ 8b_x & a2_x & 2c_x & 0f_x & 35_x & ac_x & 7a_x & c9_x & b2_x & 08_x & 03_x & 01_x & 1c_x & e8_x & 60_x & 0d_x \\ c9_x & 7a_x & ac_x & 35_x & 0f_x & 2c_x & a2_x & 8b_x & 0d_x & 60_x & e8_x & 1c_x & 01_x & 03_x & 08_x & b2_x \\ 7a_x & c9_x & 35_x & ac_x & 2c_x & 0f_x & 8b_x & a2_x & 60_x & 0d_x & 1c_x & e8_x & 03_x & 01_x & b2_x & 08_x \\ ac_x & 35_x & c9_x & 7a_x & a2_x & 8b_x & 0f_x & 2c_x & e8_x & 1c_x & 0d_x & 60_x & 08_x & b2_x & 01_x & 03_x \\ 35_x & ac_x & 7a_x & c9_x & 8b_x & a2_x & 2c_x & 0f_x & 1c_x & e8_x & 60_x & 0d_x & b2_x & 08_x & 03_x & 01_x \end{bmatrix} \tag{7.98}$$

$$\begin{aligned} &hc(01_x, 02_x, 04_x, 69_x, 07_x, ec_x, cc_x, 72_x, 0b_x, 54_x, 29_x, be_x, 74_x, f9_x, c4_x, 87_x, \\ &\quad 0e_x, 47_x, c2_x, c3_x, 39_x, 8e_x, 1c_x, 85_x, 55_x, 26_x, 1e_x, af_x, 68_x, b6_x, 59_x, 1f_x) \end{aligned} \tag{7.99}$$

Matrix (7.100) is a Hadamard-Cauchy involution matrix, presented in [70].

$$hc(0f_x, 02_x, 0c_x, 05_x, 0a_x, 04_x, 03_x, 08_x) \tag{7.100}$$

Matrices from (7.101) to (7.118) are also presented in [70].

$$had(01_x, 02_x, b0_x, b2_x) \tag{7.101}$$

$$had(01_x, 02_x, 03_x, 91_x, 04_x, 70_x, 05_x, e1_x) \tag{7.102}$$

$$had(2_x, 3_x, 4_x, c_x, 5_x, a_x, 8_x, f_x) \tag{7.103}$$

$$hc(08_x, 16_x, 8a_x, 01_x, 70_x, 8d_x, 24_x, 76_x, a8_x, 91_x, ad_x, 48_x, 05_x, b5_x, af_x, f8_x) \tag{7.104}$$

$$\begin{aligned} &hc(d2_x, 06_x, 05_x, 4d_x, 21_x, f8_x, 11_x, 62_x, 08_x, d8_x, e9_x, 28_x, 4b_x, a6_x, 10_x, 2c_x,\\ &\quad a1_x, 49_x, 4c_x, d1_x, 59_x, b2_x, 13_x, a4_x, 03_x, c3_x, 42_x, 79_x, a0_x, 6f_x, ab_x, 41_x) \end{aligned} \tag{7.105}$$

$$had(1_x, 2_x, 8_x, 9_x) = \begin{bmatrix} 1_x & 2_x & 8_x & 9_x \\ 2_x & 1_x & 9_x & 8_x \\ 8_x & 9_x & 1_x & 2_x \\ 9_x & 8_x & 2_x & 1_x \end{bmatrix} \tag{7.106}$$

$$\begin{bmatrix} d_x & 9_x & 2_x & f_x \\ 9_x & d_x & f_x & 2_x \\ 2_x & f_x & d_x & 9_x \\ f_x & 2_x & 9_x & d_x \end{bmatrix} \tag{7.107}$$

$$had(01_x, 02_x, 04_x, 91_x) = \begin{bmatrix} 01_x & 02_x & 04_x & 91_x \\ 02_x & 01_x & 91_x & 04_x \\ 04_x & 91_x & 01_x & 02_x \\ 91_x & 04_x & 02_x & 01_x \end{bmatrix} \tag{7.108}$$

$$\begin{bmatrix} 27_x & 4e_x & 9c_x & 79_x \\ 4e_x & 27_x & 79_x & 9c_x \\ 9c_x & 79_x & 27_x & 4e_x \\ 79_x & 9c_x & 4e_x & 27_x \end{bmatrix} \tag{7.109}$$

$$had(01_x, 02_x, 03_x, 08_x, 04_x, 91_x, e1_x, a9_x) \tag{7.110}$$

$$had(1a_x, 34_x, 2e_x, d0_x, 68_x, e7_x, 0d_x, 92_x) \tag{7.111}$$

$$had(1_x, 2_x, 6_x, 8_x, 9_x, c_x, d_x, a_x) \tag{7.112}$$

$$had(c_x, b_x, e_x, a_x, 6_x, f_x, 3_x, 1_x) \tag{7.113}$$

$$hc(b1_x, 1c_x, 30_x, 09_x, 08_x, 91_x, 18_x, e4_x, 98_x, 12_x, 70_x, b5_x, 97_x, 90_x, a9_x, 5b_x) \tag{7.114}$$

$$hc(4c_x, 8b_x, bf_x, f6_x, 6a_x, 27_x, be_x, e7_x, d1_x, 2f_x, 69_x, 79_x, e9_x, bb_x, f2_x, 0f_x) \tag{7.115}$$

$$\begin{aligned} &hc(b9_x, 7c_x, 93_x, bc_x, bd_x, 26_x, fa_x, a9_x, 32_x, 31_x, 24_x, b5_x, bb_x, 06_x, a0_x, 44_x,\\ &\quad 95_x, b3_x, 0c_x, 1c_x, 07_x, e5_x, a4_x, 2e_x, 56_x, 4c_x, 55_x, 02_x, 66_x, 39_x, 48_x, 08_x) \end{aligned} \tag{7.116}$$

$$hc(a9_x, b7_x, 7f_x, b2_x, b5_x, f2_x, a3_x, d9_x, 9e_x, 97_x, fc_x, 8d_x, a7_x, 12_x, e6_x, 1f_x,$$
$$6d_x, 9f_x, 24_x, 54_x, 15_x, fe_x, fa_x, ca_x, 61_x, 27_x, 68_x, e,_x f1_x, af_x, 3b_x, 38_x) \tag{7.117}$$

Matrix (7.118) is from [42] (Prøst algorithm) and also used in [70] for comparison purposes.

$$had(1_x, 2_x, 6_x, 4_x) \tag{7.118}$$

The following matrices, where x is a primitive element of $GF(2^4)$, have been reported by Gupta and Pandey [35]. The matrices reported in the paper are in polynomial notation, whilst the respective inverses (when they are not involutory) are in hexadecimal notation because we computed them for our research and our Python code outputs them in hexadecimal format.

$$\begin{bmatrix} 1 & x^3+x^2+x+1 & x^2+x+1 \\ x^3+1 & x^2+1 & x^3+x+1 \\ x^3+x^2+1 & x^2 & x^3+x^2+x \end{bmatrix} \tag{7.119}$$

$$\begin{bmatrix} 6_x & 1_x & d_x \\ c_x & d_x & c_x \\ 4_x & 7_x & d_x \end{bmatrix} \tag{7.120}$$

$$\begin{bmatrix} 1 & x^2+x & x^2+1 \\ x^2+x & 1 & x^2 \\ x^2+1 & x^2 & 1 \end{bmatrix} \tag{7.121}$$

$$\begin{bmatrix} c_x & 6_x & 2_x \\ 6_x & a_x & 3_x \\ 2_x & 3_x & 7_x \end{bmatrix} \tag{7.122}$$

$$\begin{bmatrix} x^3+x^2+1 & x^2+x+1 & x^3+x & x+1 \\ x^2+x+1 & x^3+x^2+1 & x+1 & x^3+x \\ x^3+x & x+1 & x^3+x^2+1 & x^2+x+1 \\ x+1 & x^3+x & x^2+x+1 & x^3+x^2+1 \end{bmatrix} \tag{7.123}$$

$$\begin{bmatrix} 6_x & 4_x & 2_x & e_x \\ 4_x & 6_x & e_x & 2_x \\ 2_x & e_x & 6_x & 4_x \\ e_x & 2_x & 4_x & 6_x \end{bmatrix} \tag{7.124}$$

$$\frac{1}{x+1} \cdot \begin{bmatrix} x^3+x^2+1 & x^2+x+1 & x^3+x & x+1 \\ x^2+x+1 & x^3+x^2+1 & x+1 & x^3+x \\ x^3+x & x+1 & x^3+x^2+1 & x^2+x+1 \\ x+1 & x^3+x & x^2+x+1 & x^3+x^2+1 \end{bmatrix} \tag{7.125}$$

$$\begin{bmatrix} x^3+x^2 & x^2+1 & 1 \\ x^2+1 & x^3+x+1 & 1 \\ x^3 & x^3+x^2+x+1 & 1 \end{bmatrix} \tag{7.126}$$

$$\begin{bmatrix} 9_x & c_x & 5_x \\ 7_x & 9_x & e_x \\ f_x & 4_x & a_x \end{bmatrix} \tag{7.127}$$

$$\begin{bmatrix} x^3 & x^3+1 & x^3+1 \\ x^3+x^2+x & x^3+x^2+x+1 & x^3+x^2+x \\ x^2+x+1 & x^2+x+1 & x^2+x \end{bmatrix} \tag{7.128}$$

$$circ(1, 1, x^2, 1, x^3, 1+x^2, x, 1+x^3) \tag{7.129}$$

$$\begin{bmatrix} 01d6_x & e364_x & 02fb_x & 7901_x & fe00_x & 49e2_x & fd2e_x & 2c60_x \\ 2c60_x & 01d6_x & e364_x & 02fb_x & 7901_x & fe00_x & 49e2_x & fd2e_x \\ fd2e_x & 2c60_x & 01d6_x & e364_x & 02fb_x & 7901_x & fe00_x & 49e2_x \\ 49e2_x & fd2e_x & 2c60_x & 01d6_x & e364_x & 02fb_x & 7901_x & fe00_x \\ fe00_x & 49e2_x & fd2e_x & 2c60_x & 01d6_x & e364_x & 02fb_x & 7901_x \\ 7901_x & fe00_x & 49e2_x & fd2e_x & 2c60_x & 01d6_x & e364_x & 02fb_x \\ 02fb_x & 7901_x & fe00_x & 49e2_x & fd2e_x & 2c60_x & 01d6_x & e364_x \\ e364_x & 02fb_x & 7901_x & fe00_x & 49e2_x & fd2e_x & 2c60_x & 01d6_x \end{bmatrix} \tag{7.130}$$

$$\begin{bmatrix} x & 1+x \\ 1+x & x \end{bmatrix} \tag{7.131}$$

$$\begin{bmatrix} 2_x & 3_x \\ 3_x & 2_x \end{bmatrix} \tag{7.132}$$

$$circ(x, 1+x^2+x^3+x^4+x^6, x+x^2+x^3+x^4+x^6) \tag{7.133}$$

$$\begin{bmatrix} 02_x & 5e_x & 5d_x \\ 5d_x & 02_x & 5e_x \\ 5e_x & 5d_x & 02_x \end{bmatrix} \tag{7.134}$$

$$circ(1, 1, x, 1+x^2+x^3+x^5+x^6+x^7, x+x^5, x^2+x^3+x^6+x^7) \tag{7.135}$$

$$\begin{bmatrix} 01_x & cc_x & 22_x & ed_x & 02_x & 01_x \\ 01_x & 01_x & cc_x & 22_x & ed_x & 02_x \\ 02_x & 01_x & 01_x & cc_x & 22_x & ed_x \\ ed_x & 02_x & 01_x & 01_x & cc_x & 22_x \\ 22_x & ed_x & 02_x & 01_x & 01_x & cc_x \\ cc_x & 22_x & ed_x & 02_x & 01_x & 01_x \end{bmatrix} \tag{7.136}$$

$$lcirc(1, \alpha, \alpha^7+\alpha^5+\alpha^4+\alpha+1, \alpha^7+\alpha^5+\alpha^4+\alpha^3+\alpha+1, \alpha^3+\alpha) \tag{7.137}$$

$$lcirc(1, 1, \alpha^7+\alpha^5+\alpha^4+\alpha+1, \alpha^5+\alpha^3+\alpha^2, \alpha^2, \alpha^7+\alpha^4+\alpha^3+\alpha) \tag{7.138}$$

$$Toep(\alpha, 1+\alpha^2+\alpha^3+\alpha^4+\alpha^6, \alpha+\alpha^2+\alpha^3+\alpha^4+\alpha^6, \alpha+\alpha^2+\alpha^3+\alpha^4+\alpha^6, 1+\alpha^2+\alpha^3+\alpha^4+\alpha^6) \tag{7.139}$$

$$\begin{bmatrix} 02_x & 5e_x & 5d_x \\ 5d_x & 02_x & 5e_x \\ 5e_x & 5d_x & 02_x \end{bmatrix} \tag{7.140}$$

$$\begin{aligned} Toep(1, 1, \alpha, 1+\alpha^2+\alpha^3+\alpha^5+\alpha^6+\alpha^7, \alpha+\alpha^5, \alpha^2+\alpha^3+\alpha^6+\alpha^7, \alpha^2+\alpha^3 \\ +\alpha^6+\alpha^7, \alpha+\alpha^5, 1+\alpha^2+\alpha^3+\alpha^5+\alpha^6+\alpha^7, \alpha, 1) \end{aligned} \tag{7.141}$$

$$\begin{bmatrix} 01_x & cc_x & 22_x & ed_x & 02_x & 01_x \\ 01_x & 01_x & cc_x & 22_x & ed_x & 02_x \\ 02_x & 01_x & 01_x & cc_x & 22_x & ed_x \\ ed_x & 02_x & 01_x & 01_x & cc_x & 22_x \\ 22_x & ed_x & 02_x & 01_x & 01_x & cc_x \\ cc_x & 22_x & ed_x & 02_x & 01_x & 01_x \end{bmatrix} \tag{7.142}$$

$$Hank(1, \alpha, \alpha^7+\alpha^5+\alpha^4+\alpha+1, \alpha^7+\alpha^5+\alpha^4+\alpha^3+\alpha+1, \alpha^3+\alpha, 1, \alpha, \alpha^7+\alpha^5+\alpha^4+\alpha+1, \alpha^7+\alpha^5+\alpha^4+\alpha^3+\alpha+1) \tag{7.143}$$

$$Hank(1,1,\alpha^7+\alpha^5+\alpha^4+\alpha+1,\alpha^5+\alpha^3+\alpha^2,\alpha^2,\alpha^7+\alpha^4+\alpha^3+\alpha,1,1,\alpha^7+\alpha^5+\alpha^4+\alpha+1,\alpha^5+\alpha^3+\alpha^2,\alpha^2) \tag{7.144}$$

The following matrices have been reported by Liu and Sim [50].

$$lcirc(01_x,01_x,02_x) \tag{7.145}$$

$$lcirc(01_x,01_x,02_x,91_x) \tag{7.146}$$

$$lcirc(01_x,01_x,02_x,91_x,02_x) \tag{7.147}$$

$$lcirc(01_x,02_x,e1_x,91_x,01_x,08_x) \tag{7.148}$$

$$lcirc(01_x,01_x,91_x,02_x,04_x,02_x,91_x) \tag{7.149}$$

$$lcirc(01_x,01_x,02_x,e1_x,08_x,e0_x,01_x,a9_x) \tag{7.150}$$

$$lcirc(1_x,1_x,2_x) \tag{7.151}$$

$$lcirc(1_x,1_x,9_x,4_x) \tag{7.152}$$

$$lcirc(2_x,2_x,9_x,1_x,9_x) \tag{7.153}$$

$$lcirc(1_x,1_x,9_x,c_x,9_x,3_x) \tag{7.154}$$

$$lcirc(01_x,01_x,91_x,02_x,04_x,02_x,12_x,91_x) \tag{7.155}$$

$$lcirc(01_x,01_x,04_x,02_x,a9_x,91_x,02_x,03_x) \tag{7.156}$$

$$lcirc(01_x,01_x,02_x,e0_x,06_x,e1_x,91_x,04_x) \tag{7.157}$$

$$lcirc(01_x,02_x,91_x,08_x,04_x,06_x,e1_x,03_x) \tag{7.158}$$

$$lcirc(01_x,01_x,02_x,01_x,74_x,8d_x,46_x,04_x) \tag{7.159}$$

$$lcirc(01_x,01_x,02_x,8e_x,47_x,10_x,01_x,46_x) \tag{7.160}$$

$$lcirc(5a_x,0a_x,51_x) \tag{7.161}$$

$$lcirc(01_x,02_x,b3_x,bb_x,0a_x) \tag{7.162}$$

$$lcirc(01_x,01_x,b3_x,2c_x,04_x,9a_x) \tag{7.163}$$

$$lcirc(01_x,02_x,10_x,b2_x,58_x,a4_x,5c_x) \tag{7.164}$$

$$lcirc(01_x,01_x,21_x,08_x,96_x,26_x,98_x) \tag{7.165}$$

$$lcirc(2_x,f_x,c_x) \tag{7.166}$$

$$lcirc(1_x, 2_x, 5_x, 4_x, 3_x) \tag{7.167}$$

These are Liu Sim's matrices' inverses, respectively, for the non-involutory cases.

$$\begin{bmatrix} 5f_x & 5f_x & e1_x \\ 5f_x & e1_x & 5f_x \\ e1_x & 5f_x & 5f_x \end{bmatrix} \tag{7.168}$$

$$\begin{bmatrix} 55_x & 41_x & 71_x & 5a_x \\ 41_x & 71_x & 5a_x & 55_x \\ 71_x & 5a_x & 55_x & 41_x \\ 5a_x & 55_x & 41_x & 71_x \end{bmatrix} \tag{7.169}$$

$$\begin{bmatrix} 09_x & 09_x & c2_x & 04_x & c2_x \\ 09_x & c2_x & 04_x & c2_x & 09_x \\ c2_x & 04_x & c2_x & 09_x & 09_x \\ 04_x & c2_x & 09_x & 09_x & c2_x \\ c2_x & 09_x & 09_x & c2_x & 04_x \end{bmatrix} \tag{7.170}$$

$$\begin{bmatrix} 11_x & 58_x & 90_x & 0f_x & 54_x & c0_x \\ 58_x & 90_x & 0f_x & 54_x & c0_x & 11_x \\ 90_x & 0f_x & 54_x & c0_x & 11_x & 58_x \\ 0f_x & 54_x & c0_x & 11_x & 58_x & 90_x \\ 54_x & c0_x & 11_x & 58_x & 90_x & 0f_x \\ c0_x & 11_x & 58_x & 90_x & 0f_x & 54_x \end{bmatrix} \tag{7.171}$$

$$\begin{bmatrix} e6_x & e6_x & 70_x & 0f_x & 91_x & 0f_x & 70_x \\ e6_x & 70_x & 0f_x & 91_x & 0f_x & 70_x & e6_x \\ 70_x & 0f_x & 91_x & 0f_x & 70_x & e6_x & e6_x \\ 0f_x & 91_x & 0f_x & 70_x & e6_x & e6_x & 70_x \\ 91_x & 0f_x & 70_x & e6_x & e6_x & 70_x & 0f_x \\ 0f_x & 70_x & e6_x & e6_x & 70_x & 0f_x & 91_x \\ 70_x & e6_x & e6_x & 70_x & 0f_x & 91_x & 0f_x \end{bmatrix} \tag{7.172}$$

$$\begin{bmatrix} 5d_x & c4_x & b8_x & 81_x & 5f_x & 5a_x & 22_x & 96_x \\ c4_x & b8_x & 81_x & 5f_x & 5a_x & 22_x & 96_x & 5d_x \\ b8_x & 81_x & 5f_x & 5a_x & 22_x & 96_x & 5d_x & c4_x \\ 81_x & 5f_x & 5a_x & 22_x & 96_x & 5d_x & c4_x & b8_x \\ 5f_x & 5a_x & 22_x & 96_x & 5d_x & c4_x & b8_x & 81_x \\ 5a_x & 22_x & 96_x & 5d_x & c4_x & b8_x & 81_x & 5f_x \\ 22_x & 96_x & 5d_x & c4_x & b8_x & 81_x & 5f_x & 5a_x \\ 96_x & 5d_x & c4_x & b8_x & 81_x & 5f_x & 5a_x & 22_x \end{bmatrix} \tag{7.173}$$

$$\begin{bmatrix} 7_x & 7_x & 9_x \\ 7_x & 9_x & 7_x \\ 9_x & 7_x & 7_x \end{bmatrix} \tag{7.174}$$

$$\begin{bmatrix} c_x & 8_x & 7_x & 7_x \\ 8_x & 7_x & 7_x & c_x \\ 7_x & 7_x & c_x & 8_x \\ 7_x & c_x & 8_x & 7_x \end{bmatrix} \tag{7.175}$$

$$\begin{bmatrix} 5_x & 5_x & 9_x & 1_x & 9_x \\ 5_x & 9_x & 1_x & 9_x & 5_x \\ 9_x & 1_x & 9_x & 5_x & 5_x \\ 1_x & 9_x & 5_x & 5_x & 9_x \\ 9_x & 5_x & 5_x & 9_x & 1_x \end{bmatrix} \tag{7.176}$$

$$\begin{bmatrix} 9_x & c_x & 2_x & 6_x & 7_x & e_x \\ c_x & 2_x & 6_x & 7_x & e_x & 9_x \\ 2_x & 6_x & 7_x & e_x & 9_x & c_x \\ 6_x & 7_x & e_x & 9_x & c_x & 2_x \\ 7_x & e_x & 9_x & c_x & 2_x & 6_x \\ e_x & 9_x & c_x & 2_x & 6_x & 7_x \end{bmatrix} \tag{7.177}$$

$$\begin{bmatrix} d6_x & fa_x & ee_x & f7_x & f8_x & c7_x & f8_x & 21_x \\ fa_x & ee_x & f7_x & f8_x & c7_x & f8_x & 21_x & d6_x \\ ee_x & f7_x & f8_x & c7_x & f8_x & 21_x & d6_x & fa_x \\ f7_x & f8_x & c7_x & f8_x & 21_x & d6_x & fa_x & ee_x \\ f8_x & c7_x & f8_x & 21_x & d6_x & fa_x & ee_x & f7_x \\ c7_x & f8_x & 21_x & d6_x & fa_x & ee_x & f7_x & f8_x \\ f8_x & 21_x & d6_x & fa_x & ee_x & f7_x & f8_x & c7_x \\ 21_x & d6_x & fa_x & ee_x & f7_x & f8_x & c7_x & f8_x \end{bmatrix} \tag{7.178}$$

$$\begin{bmatrix} 8c_x & ec_x & 43_x & 0a_x & 98_x & 36_x & 79_x & 6d_x \\ ec_x & 43_x & 0a_x & 98_x & 36_x & 79_x & 6d_x & 8c_x \\ 43_x & 0a_x & 98_x & 36_x & 79_x & 6d_x & 8c_x & ec_x \\ 0a_x & 98_x & 36_x & 79_x & 6d_x & 8c_x & ec_x & 43_x \\ 98_x & 36_x & 79_x & 6d_x & 8c_x & ec_x & 43_x & 0a_x \\ 36_x & 79_x & 6d_x & 8c_x & ec_x & 43_x & 0a_x & 98_x \\ 79_x & 6d_x & 8c_x & ec_x & 43_x & 0a_x & 98_x & 36_x \\ 6d_x & 8c_x & ec_x & 43_x & 0a_x & 98_x & 36_x & 79_x \end{bmatrix} \tag{7.179}$$

$$\begin{bmatrix} 4b_x & 43_x & 8c_x & 85_x & 65_x & 92_x & 04_x & 99_x \\ 43_x & 8c_x & 85_x & 65_x & 92_x & 04_x & 99_x & 4b_x \\ 8c_x & 85_x & 65_x & 92_x & 04_x & 99_x & 4b_x & 43_x \\ 85_x & 65_x & 92_x & 04_x & 99_x & 4b_x & 43_x & 8c_x \\ 65_x & 92_x & 04_x & 99_x & 4b_x & 43_x & 8c_x & 85_x \\ 92_x & 04_x & 99_x & 4b_x & 43_x & 8c_x & 85_x & 65_x \\ 04_x & 99_x & 4b_x & 43_x & 8c_x & 85_x & 65_x & 92_x \\ 99_x & 4b_x & 43_x & 8c_x & 85_x & 65_x & 92_x & 04_x \end{bmatrix} \tag{7.180}$$

$$\begin{bmatrix} 2a_x & 9d_x & 12_x & 27_x & ba_x & 0f_x & d2_x & a7_x \\ 9d_x & 12_x & 27_x & ba_x & 0f_x & d2_x & a7_x & 2a_x \\ 12_x & 27_x & ba_x & 0f_x & d2_x & a7_x & 2a_x & 9d_x \\ 27_x & ba_x & 0f_x & d2_x & a7_x & 2a_x & 9d_x & 12_x \\ ba_x & 0f_x & d2_x & a7_x & 2a_x & 9d_x & 12_x & 27_x \\ 0f_x & d2_x & a7_x & 2a_x & 9d_x & 12_x & 27_x & ba_x \\ d2_x & a7_x & 2a_x & 9d_x & 12_x & 27_x & ba_x & 0f_x \\ a7_x & 2a_x & 9d_x & 12_x & 27_x & ba_x & 0f_x & d2_x \end{bmatrix} \tag{7.181}$$

$$\begin{bmatrix} 22_x & c4_x & 68_x & e7_x & e9_x & 38_x & 49_x & 54_x \\ c4_x & 68_x & e7_x & e9_x & 38_x & 49_x & 54_x & 22_x \\ 68_x & e7_x & e9_x & 38_x & 49_x & 54_x & 22_x & c4_x \\ e7_x & e9_x & 38_x & 49_x & 54_x & 22_x & c4_x & 68_x \\ e9_x & 38_x & 49_x & 54_x & 22_x & c4_x & 68_x & e7_x \\ 38_x & 49_x & 54_x & 22_x & c4_x & 68_x & e7_x & e9_x \\ 49_x & 54_x & 22_x & c4_x & 68_x & e7_x & e9_x & 38_x \\ 54_x & 22_x & c4_x & 68_x & e7_x & e9_x & 38_x & 49_x \end{bmatrix} \tag{7.182}$$

$$\begin{bmatrix} 25_x & c6_x & c7_x & a9_x & fb_x & f2_x & b3_x & 9b_x \\ c6_x & c7_x & a9_x & fb_x & f2_x & b3_x & 9b_x & 25_x \\ c7_x & a9_x & fb_x & f2_x & b3_x & 9b_x & 25_x & c6_x \\ a9_x & fb_x & f2_x & b3_x & 9b_x & 25_x & c6_x & c7_x \\ fb_x & f2_x & b3_x & 9b_x & 25_x & c6_x & c7_x & a9_x \\ f2_x & b3_x & 9b_x & 25_x & c6_x & c7_x & a9_x & fb_x \\ b3_x & 9b_x & 25_x & c6_x & c7_x & a9_x & fb_x & f2_x \\ 9b_x & 25_x & c6_x & c7_x & a9_x & fb_x & f2_x & b3_x \end{bmatrix} \tag{7.183}$$

The following matrices were reported by Sarkar [64].

$$Toep(x, 1, x^4, 1, x^5, x^{14}, x^7, x^8, x^3, x^6, x^{14}, x^{14}, x^8, x^6, x^3) \tag{7.184}$$

$$Toep(1, 1, x, x^{253}, 1, x^{253}, x^{252}, x^{157}, x^{158}, x^{253}, x^{254}, x, x^{254}, x^2, x) \tag{7.185}$$

These are Sarkar's matrices' inverses, in integer notation.

$$\begin{bmatrix} 6 & 9 & 14 & 3 & 11 & 4 & 9 & 1 \\ 1 & 5 & 3 & 11 & 8 & 3 & 15 & 9 \\ 6 & 5 & 8 & 6 & 14 & 6 & 3 & 4 \\ 1 & 14 & 2 & 4 & 11 & 14 & 8 & 11 \\ 4 & 3 & 4 & 13 & 4 & 6 & 11 & 3 \\ 14 & 10 & 13 & 4 & 2 & 8 & 3 & 14 \\ 7 & 10 & 10 & 3 & 14 & 5 & 5 & 9 \\ 3 & 7 & 14 & 4 & 1 & 6 & 1 & 6 \end{bmatrix} \tag{7.186}$$

$$\begin{bmatrix} 240 & 99 & 21 & 8 & 119 & 12 & 187 & 181 \\ 169 & 240 & 99 & 21 & 8 & 119 & 12 & 187 \\ 181 & 169 & 240 & 99 & 21 & 8 & 119 & 12 \\ 24 & 181 & 169 & 240 & 99 & 21 & 8 & 119 \\ 238 & 24 & 181 & 169 & 240 & 99 & 21 & 8 \\ 16 & 238 & 24 & 181 & 169 & 240 & 99 & 21 \\ 42 & 16 & 238 & 24 & 181 & 169 & 240 & 99 \\ 198 & 42 & 16 & 238 & 24 & 181 & 169 & 240 \end{bmatrix} \tag{7.187}$$

The following matrices have been reported by Duval and Leurent [28].

$$\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{7.188}$$

$$\begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & 2 \\ 2 & 2 & 3 \end{bmatrix} \tag{7.189}$$

$$\begin{bmatrix} 2 & 1 & 3 \\ 1 & 1 & 1 \\ 3 & 1 & 2 \end{bmatrix} \tag{7.190}$$

$$\begin{bmatrix} 3 & 1 & 2 \\ 1 & 1 & 1 \\ 2 & 1 & 3 \end{bmatrix} \tag{7.191}$$

$$\begin{bmatrix} 3 & 1 & 3 \\ 1 & 1 & 2 \\ 2 & 1 & 1 \end{bmatrix} \tag{7.192}$$

$$\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{7.193}$$

$$\begin{bmatrix} 2 & 2 & 3 & 1 \\ 1 & 3 & 6 & 4 \\ 3 & 1 & 4 & 4 \\ 3 & 2 & 1 & 3 \end{bmatrix} \tag{7.194}$$

$$\begin{bmatrix} 2 & 2 & 3 & 1 \\ 1 & 3 & 6 & 4 \\ 3 & 1 & 4 & 4 \\ 3 & 2 & 1 & 3 \end{bmatrix} \tag{7.195}$$

$$\begin{bmatrix} 5 & 7 & 1 & 3 \\ 4 & 6 & 1 & 1 \\ 1 & 3 & 5 & 7 \\ 1 & 1 & 4 & 6 \end{bmatrix} \tag{7.196}$$

$$\begin{bmatrix} 6 & 7 & 1 & 5 \\ 2 & 3 & 1 & 1 \\ 1 & 5 & 6 & 7 \\ 1 & 1 & 2 & 3 \end{bmatrix} \tag{7.197}$$

$$\begin{bmatrix} 3 & 2 & 1 & 3 \\ 2 & 3 & 1 & 1 \\ 4 & 3 & 6 & 4 \\ 1 & 1 & 4 & 6 \end{bmatrix} \tag{7.198}$$

$$\begin{bmatrix} 1 & 2 & 4 & 3 \\ 2 & 3 & 2 & 3 \\ 3 & 3 & 5 & 1 \\ 3 & 1 & 1 & 3 \end{bmatrix} \tag{7.199}$$

The following matrices, found by Beierle [7], are not specific of a finite field, i.e they are MDS for any field $GF(2^m)$, as long as some restrictions regarding m and the α parameter are satisfied.

Matrices (7.200) (inverse: (7.201)) and (7.202) (inverse: (7.203)) are MDS for all $\alpha \neq 0, 1$.

$$circ(1, \alpha) = \begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix} \tag{7.200}$$

$$\frac{1}{1-\alpha^2} \cdot \begin{bmatrix} 1 & -\alpha \\ -\alpha & 1 \end{bmatrix} \tag{7.201}$$

$$circ(1, 1, \alpha) = \begin{bmatrix} 1 & 1 & \alpha \\ \alpha & 1 & 1 \\ 1 & \alpha & 1 \end{bmatrix} \tag{7.202}$$

$$\frac{1}{(1-\alpha)-(\alpha-1)+\alpha(\alpha^2-1)} \cdot \begin{bmatrix} 1-\alpha & \alpha^2-1 & 1-\alpha \\ 1-\alpha & 1-\alpha & \alpha^2-1 \\ \alpha^2-1 & 1-\alpha & 1-\alpha \end{bmatrix} \tag{7.203}$$

Matrix (7.204) is MDS for $m > 3$ and any α that is not a root of the following polynomials:

$$\begin{aligned} x \\ x+1 \\ x^2+x+1 \\ x^3+x+1 \\ x^3+x^2+1 \\ x^4+x^3+x^2+x+1 \\ x^5+x^2+1 \end{aligned}$$

$$circ(1, 1, \alpha, \alpha^{-2}) \tag{7.204}$$

Matrix (7.205) is MDS for $m > 3$ and any α that is not a root of the following polynomials:

$$\begin{aligned} &x \\ &x+1 \\ &x^2+x+1 \\ &x^3+x+1 \\ &x^3+x^2+1 \\ &x^4+x+1 \\ &x^4+x^3+1 \end{aligned}$$

$$circ(1, 1, \alpha, \alpha^{-2}, \alpha) \quad (7.205)$$

Matrix (7.206) is MDS for $m > 5$. For this matrix, we refer the reader to [7], because the list of restricted polynomials is bigger.

$$circ(1, \alpha, \alpha^{-1}, \alpha^{-2}, 1, \alpha^3) \quad (7.206)$$

Matrix (7.207) is MDS for $m > 5$. For this matrix, we refer the reader to [7], because the list of restricted polynomials is bigger.

$$circ(1, 1, \alpha^{-2}, \alpha, \alpha^2, \alpha, \alpha^{-2}) \quad (7.207)$$

Matrix (7.208) is MDS for $m > 7$. For this matrix, we refer the reader to [7], because the list of restricted polynomials is bigger.

$$circ(1, 1, \alpha^{-1}, \alpha, \alpha^{-1}, \alpha^3, \alpha^4, \alpha^{-3}) \quad (7.208)$$

The following matrices were suggested by Shirai in [68] to replace Whirlpool's matrix.

$$\begin{aligned}
C_0 &= rcirc(01_x, 01_x, 02_x, 01_x, 05_x, 08_x, 09_x, 04_x) && (7.209)\\
C_1 &= rcirc(01_x, 01_x, 02_x, 01_x, 06_x, 09_x, 08_x, 03_x) && (7.210)\\
C_2 &= rcirc(01_x, 01_x, 02_x, 01_x, 08_x, 09_x, 04_x, 05_x) && (7.211)\\
C_3 &= rcirc(01_x, 01_x, 02_x, 01_x, 09_x, 06_x, 04_x, 03_x) && (7.212)\\
C_4 &= rcirc(01_x, 01_x, 02_x, 06_x, 05_x, 09_x, 01_x, 08_x) && (7.213)\\
C_5 &= rcirc(01_x, 01_x, 03_x, 01_x, 04_x, 09_x, 05_x, 06_x) && (7.214)\\
C_6 &= rcirc(01_x, 01_x, 03_x, 01_x, 08_x, 04_x, 09_x, 06_x) && (7.215)\\
C_7 &= rcirc(01_x, 01_x, 04_x, 01_x, 08_x, 05_x, 02_x, 09_x) && (7.216)\\
C_8 &= rcirc(01_x, 01_x, 04_x, 01_x, 09_x, 03_x, 02_x, 06_x) && (7.217)\\
C_9 &= rcirc(01_x, 01_x, 04_x, 03_x, 06_x, 08_x, 01_x, 09_x) && (7.218)\\
C_{10} &= rcirc(01_x, 01_x, 05_x, 01_x, 04_x, 06_x, 03_x, 09_x) && (7.219)\\
C_{11} &= rcirc(01_x, 01_x, 05_x, 08_x, 02_x, 09_x, 01_x, 06_x) && (7.220)\\
C_{12} &= rcirc(01_x, 01_x, 08_x, 01_x, 06_x, 03_x, 02_x, 09_x) && (7.221)\\
C_{13} &= rcirc(01_x, 01_x, 08_x, 02_x, 04_x, 05_x, 01_x, 09_x) && (7.222)
\end{aligned}$$

The inverses of Shirai's matrices from C_0 to C_{13} are the following.

$$C_0^{-1} = rcirc(b5_x, 98_x, 23_x, fa_x, 23_x, a5_x, b6_x, 30_x) \tag{7.223}$$
$$C_1^{-1} = rcirc(bb_x, de_x, a0_x, df_x, 4a_x, 55_x, 7a_x, c5_x) \tag{7.224}$$
$$C_2^{-1} = rcirc(04_x, a4_x, cb_x, af_x, c2_x, 3e_x, 0e_x, c2_x) \tag{7.225}$$
$$C_3^{-1} = rcirc(4f_x, aa_x, 2c_x, 0c_x, 84_x, 76_x, 14_x, bb_x) \tag{7.226}$$
$$C_4^{-1} = rcirc(5b_x, e8_x, ed_x, e2_x, 33_x, 98_x, 82_x, 94_x) \tag{7.227}$$
$$C_5^{-1} = rcirc(87_x, d4_x, 76_x, 80_x, 9d_x, e4_x, 24_x, c5_x) \tag{7.228}$$
$$C_6^{-1} = rcirc(ad_x, ef_x, 44_x, 71_x, a8_x, e2_x, 42_x, 7e_x) \tag{7.229}$$
$$C_7^{-1} = rcirc(04_x, af_x, 0e_x, a4_x, c2_x, c2_x, cb_x, 3e_x) \tag{7.230}$$
$$C_8^{-1} = rcirc(4f_x, 0c_x, 14_x, aa_x, 84_x, bb_x, 2c_x, 76_x) \tag{7.231}$$
$$C_9^{-1} = rcirc(e2_x, 44_x, 7e_x, a8_x, ef_x, 42_x, 71_x, ad_x) \tag{7.232}$$
$$C_{10}^{-1} = rcirc(87_x, 80_x, 24_x, d4_x, 9d_x, c5_x, 76_x, e4_x) \tag{7.233}$$
$$C_{11}^{-1} = rcirc(ed_x, 98_x, 5b_x, e2_x, 82_x, e8_x, 33_x, 94_x) \tag{7.234}$$
$$C_{12}^{-1} = rcirc(bb_x, df_x, 7a_x, de_x, 4a_x, c5_x, a0_x, 55_x) \tag{7.235}$$
$$C_{13}^{-1} = rcirc(a5_x, 23_x, 30_x, 23_x, 98_x, b6_x, fa_x, b5_x) \tag{7.236}$$

7.17 Appendix 7C: Polynomials Experiment

A	$p(x)$	$\|A\|$	$\frac{1}{\|A\|}$	**xr**	**xt**	**Iv**	**MDS**	$\|A\|_i$	$\mathbf{xr}_i$	$\mathbf{xt}_i$	$\mathbf{Iv}_i$	$\mathbf{MDS}_i$
SHARK (7.47)	$x^8+x^4+x^3+x+1$	A1	7C	235	369	no	no	7C	261	399	no	no
SHARK (7.47)	$x^8+x^4+x^3+x^2+1$	FE	7E	235	369	no	no	7E	233	372	no	no
SHARK (7.47)	$x^8+x^5+x^3+x+1$	23	E8	235	369	no	no	E8	237	380	no	no
SHARK (7.47)	$x^8+x^5+x^3+x^2+1$	55	89	235	369	no	no	89	235	370	no	no
SHARK (7.47)	$x^8+x^5+x^4+x^3+1$	87	9B	235	369	no	no	9B	255	380	no	no
SHARK (7.47)	$x^8+x^5+x^4+x^3+x^2+x+1$	60	E3	235	369	no	no	E3	258	389	no	no
SHARK (7.47)	$x^8+x^6+x^3+x^2+1$	4F	E3	235	369	no	no	E3	257	398	no	no
SHARK (7.47)	$x^8+x^6+x^4+x^3+x^2+x+1$	C2	EA	235	369	no	no	EA	260	390	no	no
SHARK (7.47)	$x^8+x^6+x^5+x+1$	FD	CB	235	369	no	no	CB	232	384	no	no
SHARK (7.47)	$x^8+x^6+x^5+x^2+1$	4E	B8	235	369	no	no	B8	242	396	no	no
SHARK (7.47)	$x^8+x^6+x^5+x^3+1$	3C	E	235	369	no	no	E	257	375	no	no
SHARK (7.47)	$x^8+x^6+x^5+x^4+1$	5B	95	235	369	no	no	95	225	372	no	no
SHARK (7.47)	$x^8+x^6+x^5+x^4+x^2+x+1$	94	67	235	369	no	no	67	249	396	no	no
SHARK (7.47)	$x^8+x^6+x^5+x^4+x^3+x+1$	91	B3	235	369	no	no	B3	257	392	no	no
SHARK (7.47)	$x^8+x^7+x^2+x+1$	9A	EA	235	369	no	no	EA	240	387	no	no
SHARK (7.47)	$x^8+x^7+x^3+x+1$	1D	F6	235	369	no	no	F6	256	400	no	no
SHARK (7.47)	$x^8+x^7+x^3+x^2+1$	63	4	235	369	no	no	4	247	373	no	no
SHARK (7.47)	$x^8+x^7+x^4+x^3+x^2+x+1$	29	32	235	369	no	no	32	233	377	no	no
SHARK (7.47)	$x^8+x^7+x^5+x+1$	EF	95	235	369	no	no	95	241	380	no	no
SHARK (7.47)	$x^8+x^7+x^5+x^3+1$	76	54	235	369	no	no	54	246	381	no	no
SHARK (7.47)	$x^8+x^7+x^5+x^4+1$	8	36	235	369	no	no	36	230	390	no	no
SHARK (7.47)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	A	AD	235	369	no	no	AD	245	382	no	no
SHARK (7.47)	$x^8+x^7+x^6+x+1$	34	64	235	369	no	no	64	243	394	no	no
SHARK (7.47)	$x^8+x^7+x^6+x^3+x^2+x+1$	F9	78	235	369	no	no	78	274	403	no	no
SHARK (7.47)	$x^8+x^7+x^6+x^4+x^2+x+1$	A3	24	235	369	no	no	24	230	383	no	no
SHARK (7.47)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	86	CE	235	369	no	no	CE	241	379	no	no
SHARK (7.47)	$x^8+x^7+x^6+x^5+x^2+x+1$	37	52	235	369	no	no	52	254	375	no	no
SHARK (7.47)	$x^8+x^7+x^6+x^5+x^4+x+1$	29	40	235	369	no	no	40	254	395	no	no
SHARK (7.47)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	FC	EC	235	369	no	yes	EC	223	393	no	yes
SHARK (7.47)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	BF	1B	235	369	no	no	1B	257	385	no	no
SQUARE (7.49)	$x^8+x^4+x^3+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^4+x^3+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^5+x^3+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^5+x^3+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^5+x^4+x^3+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^5+x^4+x^3+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^6+x^3+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^6+x^4+x^3+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^6+x^5+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^6+x^5+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^6+x^5+x^3+1$	1	1	16	8	no	yes	1	40	48	no	yes

A	$p(x)$	$\lvert A\rvert$	$\frac{1}{\lvert A\rvert}$	**xr**	**xt**	**Iv**	**MDS**	$\lvert A\rvert_i$	**xr**$_i$	**xt**$_i$	**Iv**$_i$	**MDS**$_i$
SQUARE (7.49)	$x^8+x^6+x^5+x^4+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^6+x^5+x^4+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^6+x^5+x^4+x^3+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^3+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^3+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^4+x^3+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^5+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^5+x^3+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^5+x^4+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^6+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^6+x^3+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^6+x^4+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^6+x^5+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^6+x^5+x^4+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
SQUARE (7.49)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	1	1	16	8	no	yes	1	40	48	no	yes
BKSQ (7.58)	$x^8+x^4+x^3+x+1$	3	F6	9	9	no	yes	F6	57	63	no	yes
BKSQ (7.58)	$x^8+x^4+x^3+x^2+1$	3	F4	9	9	no	yes	F4	48	63	no	yes
BKSQ (7.58)	$x^8+x^5+x^3+x+1$	3	E6	9	9	no	yes	E6	48	63	no	yes
BKSQ (7.58)	$x^8+x^5+x^3+x^2+1$	3	E4	9	9	no	yes	E4	39	63	no	yes
BKSQ (7.58)	$x^8+x^5+x^4+x^3+1$	3	E8	9	9	no	yes	E8	39	63	no	yes
BKSQ (7.58)	$x^8+x^5+x^4+x^3+x^2+x+1$	3	EA	9	9	no	yes	EA	48	63	no	yes
BKSQ (7.58)	$x^8+x^6+x^3+x^2+1$	3	C4	9	9	no	yes	C4	30	63	no	yes
BKSQ (7.58)	$x^8+x^6+x^4+x^3+x^2+x+1$	3	CA	9	9	no	yes	CA	39	63	no	yes
BKSQ (7.58)	$x^8+x^6+x^5+x+1$	3	DE	9	9	no	yes	DE	57	63	no	yes
BKSQ (7.58)	$x^8+x^6+x^5+x^2+1$	3	DC	9	9	no	yes	DC	48	63	no	yes
BKSQ (7.58)	$x^8+x^6+x^5+x^3+1$	3	D8	9	9	no	yes	D8	39	63	no	yes
BKSQ (7.58)	$x^8+x^6+x^5+x^4+1$	3	D0	9	9	no	yes	D0	30	63	no	yes
BKSQ (7.58)	$x^8+x^6+x^5+x^4+x^2+x+1$	3	D2	9	9	no	yes	D2	39	63	no	yes
BKSQ (7.58)	$x^8+x^6+x^5+x^4+x^3+x+1$	3	D6	9	9	no	yes	D6	48	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^2+x+1$	3	82	9	9	no	yes	82	21	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^3+x+1$	3	86	9	9	no	yes	86	30	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^3+x^2+1$	3	84	9	9	no	yes	84	21	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^4+x^3+x^2+x+1$	3	8A	9	9	no	yes	8A	30	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^5+x+1$	3	9E	9	9	no	yes	9E	48	63	no	yes

A	$p(x)$	$\lvert A\rvert$	$\frac{1}{\lvert A\rvert}$	**xr**	**xt**	**Iv**	**MDS**	$\lvert A\rvert_i$	**xr**$_i$	**xt**$_i$	**Iv**$_i$	**MDS**$_i$
BKSQ (7.58)	$x^8+x^7+x^5+x^3+1$	3	98	9	9	no	yes	98	30	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^5+x^4+1$	3	90	9	9	no	yes	90	21	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	3	94	9	9	no	yes	94	30	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^6+x+1$	3	BE	9	9	no	yes	BE	57	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^6+x^3+x^2+x+1$	3	BA	9	9	no	yes	BA	48	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^6+x^4+x^2+x+1$	3	B2	9	9	no	yes	B2	39	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	3	B4	9	9	no	yes	B4	39	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^6+x^5+x^2+x+1$	3	A2	9	9	no	yes	A2	30	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^6+x^5+x^4+x+1$	3	AE	9	9	no	yes	AE	48	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	3	AC	9	9	no	yes	AC	39	63	no	yes
BKSQ (7.58)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	3	A8	9	9	no	yes	A8	30	63	no	yes
Tavares (7.51)	$x^8+x^4+x^3+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^4+x^3+x^2+1$	1	1	240	344	yes	yes	1	240	344	yes	yes
Tavares (7.51)	$x^8+x^5+x^3+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^5+x^3+x^2+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^5+x^4+x^3+1$	1	1	240	344	yes	yes	1	240	344	yes	yes
Tavares (7.51)	$x^8+x^5+x^4+x^3+x^2+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^6+x^3+x^2+1$	1	1	240	344	yes	yes	1	240	344	yes	yes
Tavares (7.51)	$x^8+x^6+x^4+x^3+x^2+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^6+x^5+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^6+x^5+x^2+1$	1	1	240	344	yes	yes	1	240	344	yes	yes
Tavares (7.51)	$x^8+x^6+x^5+x^3+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^6+x^5+x^4+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^6+x^5+x^4+x^2+x+1$	1	1	240	344	yes	yes	1	240	344	yes	yes
Tavares (7.51)	$x^8+x^6+x^5+x^4+x^3+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^2+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^3+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^3+x^2+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^4+x^3+x^2+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^5+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^5+x^3+1$	1	1	240	344	yes	yes	1	240	344	yes	yes
Tavares (7.51)	$x^8+x^7+x^5+x^4+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^6+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^6+x^3+x^2+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^6+x^4+x^2+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^6+x^5+x^2+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^6+x^5+x^4+x+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	1	1	240	344	yes	no	1	240	344	yes	no
Tavares (7.51)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	1	1	240	344	yes	no	1	240	344	yes	no

A	$p(x)$	$\|A\|$	$\frac{1}{\|A\|}$	**xr**	**xt**	**Iv**	**MDS**	$\|A\|_i$	**xr**$_i$	**xt**$_i$	**Iv**$_i$	**MDS**$_i$
KHAZAD (7.52)	$x^8+x^4+x^3+x+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^4+x^3+x^2+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^5+x^3+x+1$	1	1	112	120	yes	no	1	112	120	yes	no
KHAZAD (7.52)	$x^8+x^5+x^3+x^2+1$	1	1	112	120	yes	no	1	112	120	yes	no
KHAZAD (7.52)	$x^8+x^5+x^4+x^3+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^5+x^4+x^3+x^2+x+1$	1	1	112	120	yes	no	1	112	120	yes	no
KHAZAD (7.52)	$x^8+x^6+x^3+x^2+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^6+x^4+x^3+x^2+x+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^6+x^5+x+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^6+x^5+x^2+1$	1	1	112	120	yes	no	1	112	120	yes	no
KHAZAD (7.52)	$x^8+x^6+x^5+x^3+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^6+x^5+x^4+1$	1	1	112	120	yes	no	1	112	120	yes	no
KHAZAD (7.52)	$x^8+x^6+x^5+x^4+x^2+x+1$	1	1	112	120	yes	no	1	112	120	yes	no
KHAZAD (7.52)	$x^8+x^6+x^5+x^4+x^3+x+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^7+x^2+x+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^7+x^3+x+1$	1	1	112	120	yes	no	1	112	120	yes	no
KHAZAD (7.52)	$x^8+x^7+x^3+x^2+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^7+x^4+x^3+x^2+x+1$	1	1	112	120	yes	no	1	112	120	yes	no
KHAZAD (7.52)	$x^8+x^7+x^5+x+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^7+x^5+x^3+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^7+x^5+x^4+1$	1	1	112	120	yes	no	1	112	120	yes	no
KHAZAD (7.52)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^7+x^6+x+1$	1	1	112	120	yes	no	1	112	120	yes	no
KHAZAD (7.52)	$x^8+x^7+x^6+x^3+x^2+x+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^7+x^6+x^4+x^2+x+1$	1	1	112	120	yes	no	1	112	120	yes	no
KHAZAD (7.52)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	1	1	112	120	yes	no	1	112	120	yes	no
KHAZAD (7.52)	$x^8+x^7+x^6+x^5+x^2+x+1$	1	1	112	120	yes	no	1	112	120	yes	no
KHAZAD (7.52)	$x^8+x^7+x^6+x^5+x^4+x+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
KHAZAD (7.52)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	1	1	112	120	yes	yes	1	112	120	yes	yes
Anubis (7.53)	$x^8+x^4+x^3+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^4+x^3+x^2+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^5+x^3+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^5+x^3+x^2+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^5+x^4+x^3+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^5+x^4+x^3+x^2+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^6+x^3+x^2+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^6+x^4+x^3+x^2+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^6+x^5+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^6+x^5+x^2+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^6+x^5+x^3+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^6+x^5+x^4+1$	1	1	16	20	yes	yes	1	16	20	yes	yes

A	$p(x)$	$\|A\|$	$\frac{1}{\|A\|}$	**xr**	**xt**	**Iv**	**MDS**	$\|A\|_i$	**xr**$_i$	**xt**$_i$	**Iv**$_i$	**MDS**$_i$
Anubis (7.53)	$x^8+x^6+x^5+x^4+x^2+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^6+x^5+x^4+x^3+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^2+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^3+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^3+x^2+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^4+x^3+x^2+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^5+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^5+x^3+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^5+x^4+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^6+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^6+x^3+x^2+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^6+x^4+x^2+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^6+x^5+x^2+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^6+x^5+x^4+x+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (7.53)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	1	1	16	20	yes	yes	1	16	20	yes	yes
Anubis (KE) (7.54)	$x^8+x^4+x^3+x+1$	63	D3	20	32	no	yes	D3	65	96	no	yes
Anubis (KE) (7.54)	$x^8+x^4+x^3+x^2+1$	A5	C2	20	32	no	yes	C2	69	101	no	yes
Anubis (KE) (7.54)	$x^8+x^5+x^3+x+1$	A9	EF	20	32	no	yes	EF	71	100	no	yes
Anubis (KE) (7.54)	$x^8+x^5+x^3+x^2+1$	7B	B1	20	32	no	yes	B1	55	93	no	yes
Anubis (KE) (7.54)	$x^8+x^5+x^4+x^3+1$	E5	3C	20	32	no	yes	3C	59	97	no	yes
Anubis (KE) (7.54)	$x^8+x^5+x^4+x^3+x^2+x+1$	3B	29	20	32	no	yes	29	61	101	no	yes
Anubis (KE) (7.54)	$x^8+x^6+x^3+x^2+1$	9D	F4	20	32	no	yes	F4	47	81	no	yes
Anubis (KE) (7.54)	$x^8+x^6+x^4+x^3+x^2+x+1$	E1	18	20	32	no	yes	18	55	104	no	yes
Anubis (KE) (7.54)	$x^8+x^6+x^5+x+1$	EC	D0	20	32	no	yes	D0	57	97	no	yes
Anubis (KE) (7.54)	$x^8+x^6+x^5+x^2+1$	4	59	20	32	no	yes	59	57	93	no	yes
Anubis (KE) (7.54)	$x^8+x^6+x^5+x^3+1$	BD	D4	20	32	no	yes	D4	49	81	no	yes
Anubis (KE) (7.54)	$x^8+x^6+x^5+x^4+1$	96	4D	20	32	no	yes	4D	63	101	no	yes
Anubis (KE) (7.54)	$x^8+x^6+x^5+x^4+x^2+x+1$	72	56	20	32	no	yes	56	63	100	no	yes
Anubis (KE) (7.54)	$x^8+x^6+x^5+x^4+x^3+x+1$	C1	F3	20	32	no	yes	F3	67	92	no	yes

A	$p(x)$	$\lvert A\rvert$	$\frac{1}{\lvert A\rvert}$	**xr**	**xt**	**Iv**	**MDS**	$\lvert A\rvert_i$	**xr**$_i$	**xt**$_i$	**Iv**$_i$	**MDS**$_i$
Anubis (KE) (7.54)	$x^8+x^7+x^2+x+1$	AC	B	20	32	no	yes	B	61	98	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^3+x+1$	37	A5	20	32	no	yes	A5	57	93	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^3+x^2+1$	B9	EC	20	32	no	yes	EC	63	100	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^4+x^3+x^2+x+1$	2D	42	20	32	no	yes	42	57	94	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^5+x+1$	F3	B	20	32	no	yes	B	63	93	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^5+x^3+1$	E0	42	20	32	no	yes	42	59	96	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^5+x^4+1$	73	88	20	32	no	yes	88	53	101	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	FA	41	20	32	no	yes	41	61	100	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^6+x+1$	87	5A	20	32	no	yes	5A	67	97	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^6+x^3+x^2+x+1$	34	C5	20	32	no	yes	C5	59	100	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^6+x^4+x^2+x+1$	AD	AA	20	32	no	yes	AA	61	93	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	A6	F5	20	32	no	yes	F5	61	101	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^6+x^5+x^2+x+1$	88	3B	20	32	no	yes	3B	53	102	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^6+x^5+x^4+x+1$	AE	3	20	32	no	yes	3	59	98	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	2	FA	20	32	no	yes	FA	57	95	no	yes
Anubis (KE) (7.54)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	A3	36	20	32	no	yes	36	57	93	no	yes
Whirlwind M_0 (7.38)	x^4+x+1	B	5	104	136	no	no	5	128	136	no	no
Whirlwind M_0 (7.38)	x^4+x^3+1	4	6	104	136	no	no	6	112	160	no	no
Whirlwind M_0 (7.38)	$x^4+x^3+x^2+x+1$	E	B	104	136	no	no	B	128	128	no	no
Whirlwind M_1 (7.40)	x^4+x+1	9	2	128	128	no	no	2	128	128	no	no
Whirlwind M_1 (7.40)	x^4+x^3+1	6	4	128	128	no	no	4	120	144	no	no
Whirlwind M_1 (7.40)	$x^4+x^3+x^2+x+1$	C	D	128	128	no	no	D	120	160	no	no
Grøstl (7.73)	$x^8+x^4+x^3+x+1$	1A	FD	104	96	no	yes	FD	232	376	no	yes
Grøstl (7.73)	$x^8+x^4+x^3+x^2+1$	1C	A0	104	96	no	no	A0	240	384	no	no
Grøstl (7.73)	$x^8+x^5+x^3+x+1$	2A	B9	104	96	no	no	B9	288	376	no	no
Grøstl (7.73)	$x^8+x^5+x^3+x^2+1$	2C	74	104	96	no	no	74	216	352	no	no
Grøstl (7.73)	$x^8+x^5+x^4+x^3+1$	38	C7	104	96	no	no	C7	248	408	no	no
Grøstl (7.73)	$x^8+x^5+x^4+x^3+x^2+x+1$	3E	22	104	96	no	no	22	224	408	no	no

A	$p(x)$	$\lvert A\rvert$	$\frac{1}{\lvert A\rvert}$	**xr**	**xt**	**Iv**	**MDS**	$\lvert A\rvert_i$	**xr**$_i$	**xt**$_i$	**Iv**$_i$	**MDS**$_i$
Grøstl (7.73)	$x^8+x^6+x^3+x^2+1$	4C	3C	104	96	no	no	3C	240	400	no	no
Grøstl (7.73)	$x^8+x^6+x^4+x^3+x^2+x+1$	5E	91	104	96	no	no	91	248	432	no	no
Grøstl (7.73)	$x^8+x^6+x^5+x+1$	62	2B	104	96	no	no	2B	264	416	no	no
Grøstl (7.73)	$x^8+x^6+x^5+x^2+1$	64	1C	104	96	no	no	1C	240	328	no	no
Grøstl (7.73)	$x^8+x^6+x^5+x^3+1$	68	34	104	96	no	yes	34	248	376	no	yes
Grøstl (7.73)	$x^8+x^6+x^5+x^4+1$	70	7A	104	96	no	no	7A	288	384	no	no
Grøstl (7.73)	$x^8+x^6+x^5+x^4+x^2+x+1$	76	63	104	96	no	no	63	232	400	no	no
Grøstl (7.73)	$x^8+x^6+x^5+x^4+x^3+x+1$	7A	3D	104	96	no	no	3D	208	360	no	no
Grøstl (7.73)	$x^8+x^7+x^2+x+1$	86	B8	104	96	no	no	B8	280	376	no	no
Grøstl (7.73)	$x^8+x^7+x^3+x+1$	8A	F7	104	96	no	no	F7	208	376	no	no
Grøstl (7.73)	$x^8+x^7+x^3+x^2+1$	8C	A9	104	96	no	yes	A9	200	384	no	yes
Grøstl (7.73)	$x^8+x^7+x^4+x^3+x^2+x+1$	9E	B1	104	96	no	yes	B1	176	360	no	yes
Grøstl (7.73)	$x^8+x^7+x^5+x+1$	A2	DF	104	96	no	no	DF	224	376	no	no
Grøstl (7.73)	$x^8+x^7+x^5+x^3+1$	A8	3B	104	96	no	no	3B	272	376	no	no
Grøstl (7.73)	$x^8+x^7+x^5+x^4+1$	B0	7F	104	96	no	no	7F	216	344	no	no
Grøstl (7.73)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	BC	79	104	96	no	no	79	272	376	no	no
Grøstl (7.73)	$x^8+x^7+x^6+x+1$	C2	F8	104	96	no	no	F8	248	352	no	no
Grøstl (7.73)	$x^8+x^7+x^6+x^3+x^2+x+1$	CE	C0	104	96	no	no	C0	240	416	no	no
Grøstl (7.73)	$x^8+x^7+x^6+x^4+x^2+x+1$	D6	29	104	96	no	no	29	232	384	no	no
Grøstl (7.73)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	DC	8D	104	96	no	no	8D	200	344	no	no
Grøstl (7.73)	$x^8+x^7+x^6+x^5+x^2+x+1$	E6	E0	104	96	no	no	E0	256	384	no	no
Grøstl (7.73)	$x^8+x^7+x^6+x^5+x^4+x+1$	F2	1B	104	96	no	no	1B	240	376	no	no
Grøstl (7.73)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	F4	13	104	96	no	no	13	200	312	no	no
Grøstl (7.73)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	F8	70	104	96	no	yes	70	256	408	no	yes
Curupira (7.70)	$x^8+x^4+x^3+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^4+x^3+x^2+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^5+x^3+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^5+x^3+x^2+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^5+x^4+x^3+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^5+x^4+x^3+x^2+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^6+x^3+x^2+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^6+x^4+x^3+x^2+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^6+x^5+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^6+x^5+x^2+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^6+x^5+x^3+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^6+x^5+x^4+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^6+x^5+x^4+x^2+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^6+x^5+x^4+x^3+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^2+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes

A	$p(x)$	$\|A\|$	$\frac{1}{\|A\|}$	**xr**	**xt**	**Iv**	**MDS**	$\|A\|_i$	**xr**$_i$	**xt**$_i$	**Iv**$_i$	**MDS**$_i$
Curupira (7.70)	$x^8+x^7+x^3+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^3+x^2+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^4+x^3+x^2+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^5+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^5+x^3+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^5+x^4+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^6+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^6+x^3+x^2+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^6+x^4+x^2+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^6+x^5+x^2+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^6+x^5+x^4+x+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (7.70)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	1	1	12	15	yes	yes	1	12	15	yes	yes
Curupira (KE) (7.71)	$x^8+x^4+x^3+x+1$	1D	40	27	36	no	yes	40	12	54	no	yes
Curupira (KE) (7.71)	$x^8+x^4+x^3+x^2+1$	1D	83	27	36	no	yes	83	18	63	no	yes
Curupira (KE) (7.71)	$x^8+x^5+x^3+x+1$	1D	33	27	36	no	yes	33	27	45	no	yes
Curupira (KE) (7.71)	$x^8+x^5+x^3+x^2+1$	1D	B0	27	36	no	yes	B0	30	63	no	yes
Curupira (KE) (7.71)	$x^8+x^5+x^4+x^3+1$	1D	18	27	36	no	yes	18	21	36	no	yes
Curupira (KE) (7.71)	$x^8+x^5+x^4+x^3+x^2+x+1$	1D	61	27	36	no	yes	61	18	54	no	yes
Curupira (KE) (7.71)	$x^8+x^6+x^3+x^2+1$	1D	1C	27	36	no	yes	1C	30	36	no	yes
Curupira (KE) (7.71)	$x^8+x^6+x^4+x^3+x^2+x+1$	1D	3B	27	36	no	yes	3B	36	45	no	yes
Curupira (KE) (7.71)	$x^8+x^6+x^5+x+1$	1D	9E	27	36	no	yes	9E	48	63	no	yes
Curupira (KE) (7.71)	$x^8+x^6+x^5+x^2+1$	1D	3F	27	36	no	yes	3F	45	45	no	yes
Curupira (KE) (7.71)	$x^8+x^6+x^5+x^3+1$	1D	BB	27	36	no	yes	BB	45	63	no	yes
Curupira (KE) (7.71)	$x^8+x^6+x^5+x^4+1$	1D	79	27	36	no	yes	79	36	54	no	yes
Curupira (KE) (7.71)	$x^8+x^6+x^5+x^4+x^2+x+1$	1D	1E	27	36	no	yes	1E	39	36	no	yes
Curupira (KE) (7.71)	$x^8+x^6+x^5+x^4+x^3+x+1$	1D	F5	27	36	no	yes	F5	45	63	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^2+x+1$	1D	64	27	36	no	yes	64	30	54	no	yes

A	$p(x)$	$\|A\|$	$\frac{1}{\|A\|}$	**xr**	**xt**	**Iv**	**MDS**	$\|A\|_i$	**xr**$_i$	**xt**$_i$	**Iv**$_i$	**MDS**$_i$
Curupira (KE) (7.71)	$x^8+x^7+x^3+x+1$	1D	F6	27	36	no	yes	F6	57	63	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^3+x^2+1$	1D	2F	27	36	no	yes	2F	36	45	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^4+x^3+x^2+x+1$	1D	16	27	36	no	yes	16	30	36	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^5+x+1$	1D	51	27	36	no	yes	51	18	54	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^5+x^3+1$	1D	A5	27	36	no	yes	A5	27	63	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^5+x^4+1$	1D	EC	27	36	no	yes	EC	48	63	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	1D	6A	27	36	no	yes	6A	39	54	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^6+x+1$	1D	23	27	36	no	yes	23	18	45	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^6+x^3+x^2+x+1$	1D	EF	27	36	no	yes	EF	54	63	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^6+x^4+x^2+x+1$	1D	41	27	36	no	yes	41	9	54	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	1D	85	27	36	no	yes	85	18	63	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^6+x^5+x^2+x+1$	1D	95	27	36	no	yes	95	27	63	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^6+x^5+x^4+x+1$	1D	78	27	36	no	yes	78	39	54	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	1D	F3	27	36	no	yes	F3	45	63	no	yes
Curupira (KE) (7.71)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	1D	27	27	36	no	yes	27	27	45	no	yes
Rijndael (7.56)	$x^8+x^4+x^3+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^4+x^3+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^5+x^3+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^5+x^3+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^5+x^4+x^3+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^5+x^4+x^3+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^6+x^3+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^6+x^4+x^3+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^6+x^5+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^6+x^5+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^6+x^5+x^3+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^6+x^5+x^4+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^6+x^5+x^4+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^6+x^5+x^4+x^3+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes

A	$p(x)$	$\lvert A\rvert$	$\frac{1}{\lvert A\rvert}$	**xr**	**xt**	**Iv**	**MDS**	$\lvert A\rvert_i$	**xr**$_i$	**xt**$_i$	**Iv**$_i$	**MDS**$_i$
Rijndael (7.56)	$x^8+x^7+x^3+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^3+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^4+x^3+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^5+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^5+x^3+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^5+x^4+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^6+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^6+x^3+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^6+x^4+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^6+x^5+x^2+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^6+x^5+x^4+x+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	1	1	16	8	no	yes	1	40	48	no	yes
Rijndael (7.56)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	1	1	16	8	no	yes	1	40	48	no	yes
Hierocrypt (low) (7.60)	$x^8+x^4+x^3+x+1$	F1	23	52	108	no	yes	23	64	100	no	yes
Hierocrypt (low) (7.60)	$x^8+x^4+x^3+x^2+1$	33	2E	52	108	no	yes	2E	64	100	no	yes
Hierocrypt (low) (7.60)	$x^8+x^5+x^3+x+1$	C	AC	52	108	no	yes	AC	64	96	no	yes
Hierocrypt (low) (7.60)	$x^8+x^5+x^3+x^2+1$	CA	61	52	108	no	yes	61	56	96	no	yes
Hierocrypt (low) (7.60)	$x^8+x^5+x^4+x^3+1$	69	EF	52	108	no	yes	EF	44	96	no	yes
Hierocrypt (low) (7.60)	$x^8+x^5+x^4+x^3+x^2+x+1$	CD	18	52	108	no	yes	18	64	100	no	yes
Hierocrypt (low) (7.60)	$x^8+x^6+x^3+x^2+1$	AA	66	52	108	no	yes	66	60	92	no	yes
Hierocrypt (low) (7.60)	$x^8+x^6+x^4+x^3+x^2+x+1$	32	E6	52	108	no	yes	E6	80	108	no	yes
Hierocrypt (low) (7.60)	$x^8+x^6+x^5+x+1$	13	85	52	108	no	yes	85	52	104	no	yes
Hierocrypt (low) (7.60)	$x^8+x^6+x^5+x^2+1$	1B	31	52	108	no	yes	31	48	96	no	yes
Hierocrypt (low) (7.60)	$x^8+x^6+x^5+x^3+1$	63	6F	52	108	no	yes	6F	36	76	no	yes
Hierocrypt (low) (7.60)	$x^8+x^6+x^5+x^4+1$	12	EE	52	108	no	yes	EE	40	88	no	yes
Hierocrypt (low) (7.60)	$x^8+x^6+x^5+x^4+x^2+x+1$	C	8F	52	108	no	yes	8F	48	96	no	yes
Hierocrypt (low) (7.60)	$x^8+x^6+x^5+x^4+x^3+x+1$	28	3C	52	108	no	yes	3C	52	96	no	yes
Hierocrypt (low) (7.60)	$x^8+x^7+x^2+x+1$	6	41	52	108	no	yes	41	68	100	no	yes

A	$p(x)$	$\|A\|$	$\frac{1}{\|A\|}$	**xr**	**xt**	**Iv**	**MDS**	$\|A\|_i$	**xr**$_i$	**xt**$_i$	**Iv**$_i$	**MDS**$_i$
Hierocrypt (low) (7.60)	$x^8+x^7+x^3+x+1$	5	FB	52	108	no	yes	FB	32	40	no	yes
Hierocrypt (low) (7.60)	$x^8+x^7+x^3+x^2+1$	FA	31	52	108	no	yes	31	36	68	no	yes
Hierocrypt (low) (7.60)	$x^8+x^7+x^4+x^3+x^2+x+1$	CD	79	52	108	no	yes	79	60	104	no	yes
Hierocrypt (low) (7.60)	$x^8+x^7+x^5+x+1$	D9	6E	52	108	no	yes	6E	64	92	no	yes
Hierocrypt (low) (7.60)	$x^8+x^7+x^5+x^3+1$	46	B	52	108	no	yes	B	76	108	no	yes
Hierocrypt (low) (7.60)	$x^8+x^7+x^5+x^4+1$	95	78	52	108	no	yes	78	76	100	no	yes
Hierocrypt (low) (7.60)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	C5	E2	52	108	no	yes	E2	60	96	no	yes
Hierocrypt (low) (7.60)	$x^8+x^7+x^6+x+1$	A4	86	52	108	no	yes	86	56	108	no	yes
Hierocrypt (low) (7.60)	$x^8+x^7+x^6+x^3+x^2+x+1$	EB	3B	52	108	no	yes	3B	60	100	no	yes
Hierocrypt (low) (7.60)	$x^8+x^7+x^6+x^4+x^2+x+1$	AB	B6	52	108	no	yes	B6	68	108	no	yes
Hierocrypt (low) (7.60)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	DB	AD	52	108	no	yes	AD	68	100	no	yes
Hierocrypt (low) (7.60)	$x^8+x^7+x^6+x^5+x^2+x+1$	AD	94	52	108	no	no	94	56	108	no	no
Hierocrypt (low) (7.60)	$x^8+x^7+x^6+x^5+x^4+x+1$	DC	7	52	108	no	yes	7	40	84	no	yes
Hierocrypt (low) (7.60)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	B2	79	52	108	no	yes	79	48	88	no	yes
Hierocrypt (low) (7.60)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	3D	B6	52	108	no	no	B6	76	104	no	no
Hierocrypt-3 (high) (7.62)	x^4+x+1	5	B	32	40	no	yes	B	40	44	no	yes
Hierocrypt-3 (high) (7.62)	x^4+x^3+1	E	7	32	40	no	no	7	28	32	no	no
Hierocrypt-3 (high) (7.62)	$x^4+x^3+x^2+x+1$	8	4	32	40	no	yes	4	16	28	no	yes
Hierocrypt L1 (high) (7.64)	x^4+x+1	2	9	8	10	no	yes	9	7	11	no	yes
Hierocrypt L1 (high) (7.64)	x^4+x^3+1	8	3	8	10	no	yes	3	8	10	no	yes
Hierocrypt L1 (high) (7.64)	$x^4+x^3+x^2+x+1$	E	B	8	10	no	yes	B	9	10	no	yes
FOX mu4 (7.66)	$x^8+x^4+x^3+x+1$	C4	DA	30	25	no	yes	DA	70	105	no	yes
FOX mu4 (7.66)	$x^8+x^4+x^3+x^2+1$	62	CA	30	25	no	yes	CA	58	91	no	yes
FOX mu4 (7.66)	$x^8+x^5+x^3+x+1$	75	FD	30	25	no	yes	FD	76	111	no	yes
FOX mu4 (7.66)	$x^8+x^5+x^3+x^2+1$	55	89	30	25	no	yes	89	71	103	no	yes
FOX mu4 (7.66)	$x^8+x^5+x^4+x^3+1$	1F	35	30	25	no	yes	35	43	85	no	yes
FOX mu4 (7.66)	$x^8+x^5+x^4+x^3+x^2+x+1$	1C	E2	30	25	no	yes	E2	46	90	no	yes

A	$p(x)$	$\lvert A\rvert$	$\frac{1}{\lvert A\rvert}$	**xr**	**xt**	**Iv**	**MDS**	$\lvert A\rvert_i$	**xr**$_i$	**xt**$_i$	**Iv**$_i$	**MDS**$_i$
FOX mu4 (7.66)	$x^8+x^6+x^3+x^2+1$	79	CE	30	25	no	yes	CE	61	104	no	yes
FOX mu4 (7.66)	$x^8+x^6+x^4+x^3+x^2+x+1$	B6	25	30	25	no	yes	25	58	91	no	yes
FOX mu4 (7.66)	$x^8+x^6+x^5+x+1$	AA	F2	30	25	no	yes	F2	59	87	no	yes
FOX mu4 (7.66)	$x^8+x^6+x^5+x^2+1$	55	6C	30	25	no	yes	6C	59	100	no	yes
FOX mu4 (7.66)	$x^8+x^6+x^5+x^3+1$	60	E8	30	25	no	yes	E8	75	98	no	yes
FOX mu4 (7.66)	$x^8+x^6+x^5+x^4+1$	BC	2B	30	25	no	yes	2B	89	95	no	yes
FOX mu4 (7.66)	$x^8+x^6+x^5+x^4+x^2+x+1$	66	D4	30	25	no	yes	D4	73	112	no	yes
FOX mu4 (7.66)	$x^8+x^6+x^5+x^4+x^3+x+1$	28	3C	30	25	no	yes	3C	76	96	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^2+x+1$	14	DC	30	25	no	yes	DC	66	106	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^3+x+1$	72	18	30	25	no	yes	18	61	95	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^3+x^2+1$	DB	E6	30	25	no	yes	E6	79	91	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^4+x^3+x^2+x+1$	69	7D	30	25	no	yes	7D	65	109	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^5+x+1$	B6	98	30	25	no	yes	98	52	100	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^5+x^3+1$	25	B7	30	25	no	yes	B7	72	97	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^5+x^4+1$	F2	64	30	25	no	yes	64	67	112	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	32	F8	30	25	no	yes	F8	59	100	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^6+x+1$	D3	16	30	25	no	no	16	57	103	no	no
FOX mu4 (7.66)	$x^8+x^7+x^6+x^3+x^2+x+1$	8A	D	30	25	no	yes	D	61	104	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^6+x^4+x^2+x+1$	90	5D	30	25	no	yes	5D	86	108	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	A8	65	30	25	no	yes	65	56	106	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^6+x^5+x^2+x+1$	3	A2	30	25	no	yes	A2	59	84	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^6+x^5+x^4+x+1$	71	70	30	25	no	yes	70	64	88	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	D6	11	30	25	no	yes	11	47	103	no	yes
FOX mu4 (7.66)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	13	26	30	25	no	yes	26	72	106	no	yes
FOX mu8 (7.67)	$x^8+x^4+x^3+x+1$	3	F6	141	169	no	no	F6	283	399	no	no
FOX mu8 (7.67)	$x^8+x^4+x^3+x^2+1$	5A	CB	141	169	no	no	CB	268	371	no	no
FOX mu8 (7.67)	$x^8+x^5+x^3+x+1$	43	44	141	169	no	no	44	268	391	no	no
FOX mu8 (7.67)	$x^8+x^5+x^3+x^2+1$	BD	FA	141	169	no	no	FA	248	392	no	no
FOX mu8 (7.67)	$x^8+x^5+x^4+x^3+1$	A2	11	141	169	no	no	11	229	405	no	no
FOX mu8 (7.67)	$x^8+x^5+x^4+x^3+x^2+x+1$	11	7C	141	169	no	no	7C	194	369	no	no
FOX mu8 (7.67)	$x^8+x^6+x^3+x^2+1$	CD	71	141	169	no	no	71	290	411	no	no
FOX mu8 (7.67)	$x^8+x^6+x^4+x^3+x^2+x+1$	A2	34	141	169	no	no	34	234	350	no	no
FOX mu8 (7.67)	$x^8+x^6+x^5+x+1$	14	A3	141	169	no	no	A3	290	418	no	no
FOX mu8 (7.67)	$x^8+x^6+x^5+x^2+1$	3	DC	141	169	no	no	DC	255	405	no	no
FOX mu8 (7.67)	$x^8+x^6+x^5+x^3+1$	85	DE	141	169	no	no	DE	229	399	no	no
FOX mu8 (7.67)	$x^8+x^6+x^5+x^4+1$	9B	6F	141	169	no	no	6F	246	392	no	no
FOX mu8 (7.67)	$x^8+x^6+x^5+x^4+x^2+x+1$	E5	51	141	169	no	no	51	212	342	no	no
FOX mu8 (7.67)	$x^8+x^6+x^5+x^4+x^3+x+1$	E2	9A	141	169	no	no	9A	251	364	no	no
FOX mu8 (7.67)	$x^8+x^7+x^2+x+1$	31	D7	141	169	no	no	D7	234	397	no	no
FOX mu8 (7.67)	$x^8+x^7+x^3+x+1$	19	CD	141	169	no	no	CD	256	406	no	no
FOX mu8 (7.67)	$x^8+x^7+x^3+x^2+1$	F8	C4	141	169	no	no	C4	257	364	no	no
FOX mu8 (7.67)	$x^8+x^7+x^4+x^3+x^2+x+1$	FA	FB	141	169	no	no	FB	235	371	no	no

A	$p(x)$	$\|A\|$	$\frac{1}{\|A\|}$	**xr**	**xt**	**Iv**	**MDS**	$\|A\|_i$	**xr**$_i$	**xt**$_i$	**Iv**$_i$	**MDS**$_i$
FOX mu8 (7.67)	$x^8+x^7+x^5+x+1$	BA	E1	141	169	no	no	E1	255	427	no	no
FOX mu8 (7.67)	$x^8+x^7+x^5+x^3+1$	5	EF	141	169	no	no	EF	246	404	no	no
FOX mu8 (7.67)	$x^8+x^7+x^5+x^4+1$	A1	E1	141	169	no	no	E1	235	392	no	no
FOX mu8 (7.67)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	73	DC	141	169	no	no	DC	172	363	no	no
FOX mu8 (7.67)	$x^8+x^7+x^6+x+1$	CE	C	141	169	no	no	C	263	371	no	no
FOX mu8 (7.67)	$x^8+x^7+x^6+x^3+x^2+x+1$	49	EC	141	169	no	no	EC	256	391	no	no
FOX mu8 (7.67)	$x^8+x^7+x^6+x^4+x^2+x+1$	1D	41	141	169	no	no	41	230	406	no	no
FOX mu8 (7.67)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	51	BB	141	169	no	no	BB	283	426	no	no
FOX mu8 (7.67)	$x^8+x^7+x^6+x^5+x^2+x+1$	71	EB	141	169	no	no	EB	304	378	no	no
FOX mu8 (7.67)	$x^8+x^7+x^6+x^5+x^4+x+1$	20	52	141	169	no	no	52	269	377	no	no
FOX mu8 (7.67)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	82	1B	141	169	no	no	1B	276	383	no	no
FOX mu8 (7.67)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	F0	99	141	169	no	yes	99	284	392	no	yes
Whirlpool-0 (7.5)	$x^8+x^4+x^3+x+1$	1A	FD	88	88	no	no	FD	248	336	no	no
Whirlpool-0 (7.5)	$x^8+x^4+x^3+x^2+1$	1C	A0	88	88	no	no	A0	240	360	no	no
Whirlpool-0 (7.5)	$x^8+x^5+x^3+x+1$	2A	B9	88	88	no	no	B9	256	416	no	no
Whirlpool-0 (7.5)	$x^8+x^5+x^3+x^2+1$	2C	74	88	88	no	no	74	248	392	no	no
Whirlpool-0 (7.5)	$x^8+x^5+x^4+x^3+1$	38	C7	88	88	no	no	C7	264	440	no	no
Whirlpool-0 (7.5)	$x^8+x^5+x^4+x^3+x^2+x+1$	3E	22	88	88	no	no	22	256	408	no	no
Whirlpool-0 (7.5)	$x^8+x^6+x^3+x^2+1$	4C	3C	88	88	no	no	3C	192	360	no	no
Whirlpool-0 (7.5)	$x^8+x^6+x^4+x^3+x^2+x+1$	5E	91	88	88	no	no	91	280	440	no	no
Whirlpool-0 (7.5)	$x^8+x^6+x^5+x+1$	62	2B	88	88	no	no	2B	264	424	no	no
Whirlpool-0 (7.5)	$x^8+x^6+x^5+x^2+1$	64	1C	88	88	no	no	1C	224	384	no	no
Whirlpool-0 (7.5)	$x^8+x^6+x^5+x^3+1$	68	34	88	88	no	no	34	216	328	no	no
Whirlpool-0 (7.5)	$x^8+x^6+x^5+x^4+1$	70	7A	88	88	no	no	7A	272	376	no	no
Whirlpool-0 (7.5)	$x^8+x^6+x^5+x^4+x^2+x+1$	76	63	88	88	no	no	63	248	408	no	no
Whirlpool-0 (7.5)	$x^8+x^6+x^5+x^4+x^3+x+1$	7A	3D	88	88	no	no	3D	272	408	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^2+x+1$	86	B8	88	88	no	no	B8	184	328	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^3+x+1$	8A	F7	88	88	no	no	F7	208	328	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^3+x^2+1$	8C	A9	88	88	no	no	A9	216	392	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^4+x^3+x^2+x+1$	9E	B1	88	88	no	no	B1	256	400	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^5+x+1$	A2	DF	88	88	no	no	DF	256	376	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^5+x^3+1$	A8	3B	88	88	no	no	3B	256	392	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^5+x^4+1$	B0	7F	88	88	no	no	7F	264	400	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^5+x^4+x^3+x^2+1$	BC	79	88	88	no	no	79	240	400	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^6+x+1$	C2	F8	88	88	no	no	F8	264	400	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^6+x^3+x^2+x+1$	CE	C0	88	88	no	no	C0	240	432	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^6+x^4+x^2+x+1$	D6	29	88	88	no	no	29	184	384	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^6+x^4+x^3+x^2+1$	DC	8D	88	88	no	no	8D	248	336	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^6+x^5+x^2+x+1$	E6	E0	88	88	no	no	E0	240	432	no	no

A	$p(x)$	$\|A\|$	$\frac{1}{\|A\|}$	**xr**	**xt**	**Iv**	**MDS**	$\|A\|_i$	**xr**$_i$	**xt**$_i$	**Iv**$_i$	**MDS**$_i$
Whirlpool-0 (7.5)	$x^8+x^7+x^6+x^5+x^4+x+1$	F2	1B	88	88	no	no	1B	272	392	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^6+x^5+x^4+x^2+1$	F4	13	88	88	no	no	13	200	360	no	no
Whirlpool-0 (7.5)	$x^8+x^7+x^6+x^5+x^4+x^3+1$	F8	70	88	88	no	no	70	272	424	no	no

Table 7.15: Studying different irreducible polynomials for each matrix

7.18 Appendix 7D: Whirlwind non-MDS matrices tables

Rows to be removed	**Columns to be removed**
(0, 3)	(0, 7)
(0, 3)	(3, 4)
(0, 5)	(0, 7)
(0, 5)	(2, 5)
(0, 6)	(2, 5)
(0, 6)	(3, 4)
(0, 7)	(0, 3)
(0, 7)	(0, 5)
(0, 7)	(2, 4)
(0, 7)	(2, 7)
(0, 7)	(3, 5)
(0, 7)	(4, 7)
(1, 2)	(1, 6)
(1, 2)	(2, 5)
(1, 4)	(1, 6)
(1, 4)	(3, 4)
(1, 6)	(1, 2)
(1, 6)	(1, 4)
(1, 6)	(2, 4)
(1, 6)	(3, 5)
(1, 6)	(3, 6)
(1, 6)	(5, 6)
(1, 7)	(2, 5)
(1, 7)	(3, 4)
(2, 4)	(0, 7)
(2, 4)	(1, 6)
(2, 5)	(0, 5)
(2, 5)	(0, 6)
(2, 5)	(1, 2)
(2, 5)	(1, 7)
(2, 5)	(2, 7)
(2, 5)	(5, 6)
(2, 7)	(0, 7)
(2, 7)	(2, 5)
(3, 4)	(0, 3)
(3, 4)	(0, 6)
(3, 4)	(1, 4)
(3, 4)	(1, 7)
(3, 4)	(3, 6)
(3, 4)	(4, 7)
(3, 5)	(0, 7)
(3, 5)	(1, 6)
(3, 6)	(1, 6)
(3, 6)	(3, 4)
(4, 7)	(0, 7)
(4, 7)	(3, 4)
(5, 6)	(1, 6)
(5, 6)	(2, 5)
(0, 1, 2)	(0, 3, 7)
(0, 1, 2)	(3, 4, 6)
(0, 1, 3)	(1, 2, 6)
(0, 1, 3)	(2, 5, 7)
(0, 1, 4)	(1, 3, 7)
(0, 1, 5)	(0, 2, 6)
(0, 1, 6)	(0, 5, 7)
(0, 1, 6)	(1, 2, 4)
(0, 1, 6)	(3, 5, 6)
(0, 1, 7)	(0, 3, 5)
(0, 1, 7)	(1, 4, 6)
(0, 1, 7)	(2, 4, 7)
(0, 2, 3)	(1, 2, 5)
(0, 2, 3)	(1, 4, 6)
(0, 2, 4)	(1, 2, 4)
(0, 2, 4)	(2, 3, 7)
(0, 2, 5)	(0, 5, 6)
(0, 2, 5)	(1, 2, 7)

Rows to be removed	**Columns to be removed**
(0, 2, 5)	(3, 4, 5)
(0, 2, 5)	(4, 6, 7)
(0, 2, 6)	(0, 1, 5)
(0, 2, 6)	(0, 3, 6)
(0, 2, 7)	(0, 3, 5)
(0, 2, 7)	(1, 6, 7)
(0, 2, 7)	(2, 4, 7)
(0, 2, 7)	(4, 5, 6)
(0, 3, 4)	(0, 3, 6)
(0, 3, 4)	(1, 2, 3)
(0, 3, 4)	(1, 4, 7)
(0, 3, 5)	(0, 1, 7)
(0, 3, 5)	(0, 2, 7)
(0, 3, 5)	(0, 3, 7)
(0, 3, 5)	(0, 4, 7)
(0, 3, 5)	(0, 5, 7)
(0, 3, 5)	(0, 6, 7)
(0, 3, 5)	(1, 3, 5)
(0, 3, 6)	(0, 2, 6)
(0, 3, 6)	(0, 3, 4)
(0, 3, 6)	(1, 3, 4)
(0, 3, 6)	(2, 3, 4)
(0, 3, 6)	(3, 4, 5)
(0, 3, 6)	(3, 4, 6)
(0, 3, 6)	(3, 4, 7)
(0, 3, 7)	(0, 1, 2)
(0, 3, 7)	(0, 3, 5)
(0, 3, 7)	(2, 4, 7)
(0, 4, 5)	(3, 5, 7)
(0, 4, 6)	(0, 5, 6)
(0, 4, 6)	(3, 6, 7)
(0, 4, 7)	(0, 3, 5)
(0, 4, 7)	(2, 4, 7)
(0, 4, 7)	(5, 6, 7)
(0, 5, 6)	(0, 2, 5)
(0, 5, 6)	(0, 4, 6)
(0, 5, 6)	(1, 2, 5)
(0, 5, 6)	(2, 3, 5)
(0, 5, 6)	(2, 4, 5)
(0, 5, 6)	(2, 5, 6)
(0, 5, 6)	(2, 5, 7)
(0, 5, 7)	(0, 1, 6)
(0, 5, 7)	(0, 3, 5)
(0, 5, 7)	(1, 2, 3)
(0, 5, 7)	(2, 4, 7)
(0, 6, 7)	(0, 3, 5)
(0, 6, 7)	(1, 3, 6)
(0, 6, 7)	(2, 4, 7)
(1, 2, 3)	(0, 3, 4)
(1, 2, 3)	(0, 5, 7)
(1, 2, 4)	(0, 1, 6)
(1, 2, 4)	(0, 2, 4)
(1, 2, 4)	(1, 2, 6)
(1, 2, 4)	(1, 3, 6)
(1, 2, 4)	(1, 4, 6)
(1, 2, 4)	(1, 5, 6)
(1, 2, 4)	(1, 6, 7)
(1, 2, 5)	(0, 2, 3)
(1, 2, 5)	(0, 5, 6)
(1, 2, 5)	(1, 2, 7)
(1, 2, 6)	(0, 1, 3)
(1, 2, 6)	(1, 2, 4)
(1, 2, 6)	(3, 5, 6)
(1, 2, 7)	(0, 2, 5)
(1, 2, 7)	(1, 2, 5)
(1, 2, 7)	(1, 3, 7)
(1, 2, 7)	(2, 3, 5)
(1, 2, 7)	(2, 4, 5)
(1, 2, 7)	(2, 5, 6)

Rows to be removed	**Columns to be removed**
(1, 2, 7)	(2, 5, 7)
(1, 3, 4)	(0, 3, 6)
(1, 3, 4)	(1, 4, 7)
(1, 3, 4)	(2, 4, 5)
(1, 3, 4)	(5, 6, 7)
(1, 3, 5)	(0, 3, 5)
(1, 3, 5)	(2, 3, 6)
(1, 3, 6)	(0, 6, 7)
(1, 3, 6)	(1, 2, 4)
(1, 3, 6)	(3, 5, 6)
(1, 3, 6)	(4, 5, 7)
(1, 3, 7)	(0, 1, 4)
(1, 3, 7)	(1, 2, 7)
(1, 4, 5)	(2, 4, 6)
(1, 4, 6)	(0, 1, 7)
(1, 4, 6)	(0, 2, 3)
(1, 4, 6)	(1, 2, 4)
(1, 4, 6)	(3, 5, 6)
(1, 4, 7)	(0, 3, 4)
(1, 4, 7)	(1, 3, 4)
(1, 4, 7)	(1, 5, 7)
(1, 4, 7)	(2, 3, 4)
(1, 4, 7)	(3, 4, 5)
(1, 4, 7)	(3, 4, 6)
(1, 4, 7)	(3, 4, 7)
(1, 5, 6)	(1, 2, 4)
(1, 5, 6)	(3, 5, 6)
(1, 5, 6)	(4, 6, 7)
(1, 5, 7)	(1, 4, 7)
(1, 5, 7)	(2, 6, 7)
(1, 6, 7)	(0, 2, 7)
(1, 6, 7)	(1, 2, 4)
(1, 6, 7)	(3, 5, 6)
(2, 3, 4)	(0, 3, 6)
(2, 3, 4)	(1, 4, 7)
(2, 3, 4)	(2, 5, 7)
(2, 3, 5)	(0, 5, 6)
(2, 3, 5)	(1, 2, 7)
(2, 3, 5)	(3, 4, 6)
(2, 3, 6)	(1, 3, 5)
(2, 3, 7)	(0, 2, 4)
(2, 4, 5)	(0, 5, 6)
(2, 4, 5)	(1, 2, 7)
(2, 4, 5)	(1, 3, 4)
(2, 4, 6)	(1, 4, 5)
(2, 4, 6)	(2, 4, 7)
(2, 4, 7)	(0, 1, 7)
(2, 4, 7)	(0, 2, 7)
(2, 4, 7)	(0, 3, 7)
(2, 4, 7)	(0, 4, 7)
(2, 4, 7)	(0, 5, 7)
(2, 4, 7)	(0, 6, 7)
(2, 4, 7)	(2, 4, 6)
(2, 5, 6)	(0, 5, 6)
(2, 5, 6)	(1, 2, 7)
(2, 5, 6)	(4, 5, 7)
(2, 5, 7)	(0, 1, 3)
(2, 5, 7)	(0, 5, 6)
(2, 5, 7)	(1, 2, 7)
(2, 5, 7)	(2, 3, 4)
(2, 6, 7)	(1, 5, 7)
(3, 4, 5)	(0, 2, 5)
(3, 4, 5)	(0, 3, 6)
(3, 4, 5)	(1, 4, 7)
(3, 4, 6)	(0, 1, 2)
(3, 4, 6)	(0, 3, 6)
(3, 4, 6)	(1, 4, 7)
(3, 4, 6)	(2, 3, 5)
(3, 4, 7)	(0, 3, 6)

Rows to be removed	**Columns to be removed**
(3, 4, 7)	(1, 4, 7)
(3, 4, 7)	(4, 5, 6)
(3, 5, 6)	(0, 1, 6)
(3, 5, 6)	(1, 2, 6)
(3, 5, 6)	(1, 3, 6)
(3, 5, 6)	(1, 4, 6)
(3, 5, 6)	(1, 5, 6)
(3, 5, 6)	(1, 6, 7)
(3, 5, 6)	(3, 5, 7)
(3, 5, 7)	(0, 4, 5)
(3, 5, 7)	(3, 5, 6)
(3, 6, 7)	(0, 4, 6)
(4, 5, 6)	(0, 2, 7)
(4, 5, 6)	(3, 4, 7)
(4, 5, 7)	(1, 3, 6)
(4, 5, 7)	(2, 5, 6)
(4, 6, 7)	(0, 2, 5)
(4, 6, 7)	(1, 5, 6)
(5, 6, 7)	(0, 4, 7)
(5, 6, 7)	(1, 3, 4)
(0, 1, 2, 3)	(0, 3, 5, 6)
(0, 1, 2, 3)	(1, 2, 4, 7)
(0, 1, 2, 4)	(0, 4, 6, 7)
(0, 1, 2, 4)	(2, 4, 5, 6)
(0, 1, 2, 5)	(1, 2, 4, 6)
(0, 1, 2, 5)	(1, 3, 5, 6)
(0, 1, 2, 6)	(0, 4, 5, 6)
(0, 1, 2, 6)	(1, 3, 4, 5)
(0, 1, 2, 6)	(2, 3, 5, 7)
(0, 1, 2, 6)	(2, 4, 6, 7)
(0, 1, 3, 4)	(0, 2, 4, 7)
(0, 1, 3, 4)	(0, 3, 5, 7)
(0, 1, 3, 5)	(1, 5, 6, 7)
(0, 1, 3, 5)	(3, 4, 5, 7)
(0, 1, 3, 7)	(0, 2, 4, 5)
(0, 1, 3, 7)	(1, 4, 5, 7)
(0, 1, 3, 7)	(2, 3, 4, 6)
(0, 1, 3, 7)	(3, 5, 6, 7)
(0, 1, 4, 6)	(1, 2, 3, 5)
(0, 1, 4, 6)	(3, 4, 5, 7)
(0, 1, 4, 7)	(0, 2, 5, 6)
(0, 1, 4, 7)	(0, 3, 6, 7)
(0, 1, 4, 7)	(1, 3, 5, 6)
(0, 1, 5, 6)	(0, 2, 4, 7)
(0, 1, 5, 6)	(1, 2, 6, 7)
(0, 1, 5, 6)	(1, 3, 4, 7)
(0, 1, 5, 7)	(0, 2, 3, 4)
(0, 1, 5, 7)	(2, 4, 5, 6)
(0, 2, 3, 4)	(0, 1, 5, 7)
(0, 2, 3, 4)	(0, 4, 5, 6)
(0, 2, 3, 4)	(1, 3, 6, 7)
(0, 2, 3, 4)	(2, 4, 6, 7)
(0, 2, 3, 6)	(0, 4, 6, 7)
(0, 2, 3, 6)	(2, 4, 5, 6)
(0, 2, 3, 7)	(0, 3, 4, 6)
(0, 2, 3, 7)	(1, 3, 4, 7)
(0, 2, 4, 5)	(0, 1, 3, 7)
(0, 2, 4, 5)	(1, 5, 6, 7)
(0, 2, 4, 6)	(0, 3, 4, 7)
(0, 2, 4, 6)	(1, 2, 5, 6)
(0, 2, 4, 7)	(0, 1, 3, 4)
(0, 2, 4, 7)	(0, 1, 5, 6)
(0, 2, 4, 7)	(2, 3, 5, 6)
(0, 2, 4, 7)	(3, 4, 6, 7)
(0, 2, 5, 6)	(0, 1, 4, 7)
(0, 2, 5, 6)	(1, 2, 3, 6)
(0, 2, 5, 6)	(1, 4, 5, 6)
(0, 2, 5, 6)	(2, 3, 4, 7)
(0, 2, 6, 7)	(1, 2, 3, 5)

Rows to be removed	**Columns to be removed**
(0, 2, 6, 7)	(3, 4, 5, 7)
(0, 3, 4, 5)	(1, 2, 4, 6)
(0, 3, 4, 5)	(1, 2, 5, 7)
(0, 3, 4, 5)	(2, 3, 4, 7)
(0, 3, 4, 6)	(0, 2, 3, 7)
(0, 3, 4, 6)	(0, 4, 5, 7)
(0, 3, 4, 6)	(1, 2, 4, 5)
(0, 3, 4, 6)	(1, 2, 6, 7)
(0, 3, 4, 7)	(0, 2, 4, 6)
(0, 3, 4, 7)	(1, 3, 5, 7)
(0, 3, 5, 6)	(0, 1, 2, 3)
(0, 3, 5, 6)	(4, 5, 6, 7)
(0, 3, 5, 7)	(0, 1, 3, 4)
(0, 3, 5, 7)	(1, 2, 4, 5)
(0, 3, 5, 7)	(1, 2, 6, 7)
(0, 3, 5, 7)	(3, 4, 6, 7)
(0, 3, 6, 7)	(0, 1, 4, 7)
(0, 3, 6, 7)	(1, 2, 4, 6)
(0, 3, 6, 7)	(1, 2, 5, 7)
(0, 4, 5, 6)	(0, 1, 2, 6)
(0, 4, 5, 6)	(0, 2, 3, 4)
(0, 4, 5, 7)	(0, 3, 4, 6)
(0, 4, 5, 7)	(1, 3, 4, 7)
(0, 4, 6, 7)	(0, 1, 2, 4)
(0, 4, 6, 7)	(0, 2, 3, 6)
(0, 4, 6, 7)	(1, 3, 4, 5)
(0, 4, 6, 7)	(2, 3, 5, 7)
(1, 2, 3, 5)	(0, 1, 4, 6)
(1, 2, 3, 5)	(0, 2, 6, 7)
(1, 2, 3, 5)	(1, 4, 5, 7)
(1, 2, 3, 5)	(3, 5, 6, 7)
(1, 2, 3, 6)	(0, 2, 5, 6)
(1, 2, 3, 6)	(1, 2, 5, 7)
(1, 2, 3, 7)	(1, 5, 6, 7)
(1, 2, 3, 7)	(3, 4, 5, 7)
(1, 2, 4, 5)	(0, 3, 4, 6)
(1, 2, 4, 5)	(0, 3, 5, 7)
(1, 2, 4, 5)	(2, 3, 5, 6)
(1, 2, 4, 6)	(0, 1, 2, 5)
(1, 2, 4, 6)	(0, 3, 4, 5)
(1, 2, 4, 6)	(0, 3, 6, 7)
(1, 2, 4, 6)	(2, 5, 6, 7)
(1, 2, 4, 7)	(0, 1, 2, 3)
(1, 2, 4, 7)	(4, 5, 6, 7)
(1, 2, 5, 6)	(0, 2, 4, 6)
(1, 2, 5, 6)	(1, 3, 5, 7)
(1, 2, 5, 7)	(0, 3, 4, 5)
(1, 2, 5, 7)	(0, 3, 6, 7)
(1, 2, 5, 7)	(1, 2, 3, 6)
(1, 2, 5, 7)	(1, 4, 5, 6)
(1, 2, 6, 7)	(0, 1, 5, 6)
(1, 2, 6, 7)	(0, 3, 4, 6)
(1, 2, 6, 7)	(0, 3, 5, 7)
(1, 3, 4, 5)	(0, 1, 2, 6)
(1, 3, 4, 5)	(0, 4, 6, 7)
(1, 3, 4, 7)	(0, 1, 5, 6)
(1, 3, 4, 7)	(0, 2, 3, 7)
(1, 3, 4, 7)	(0, 4, 5, 7)
(1, 3, 4, 7)	(2, 3, 5, 6)
(1, 3, 5, 6)	(0, 1, 2, 5)
(1, 3, 5, 6)	(0, 1, 4, 7)
(1, 3, 5, 6)	(2, 3, 4, 7)
(1, 3, 5, 6)	(2, 5, 6, 7)
(1, 3, 5, 7)	(0, 3, 4, 7)
(1, 3, 5, 7)	(1, 2, 5, 6)
(1, 3, 6, 7)	(0, 2, 3, 4)
(1, 3, 6, 7)	(2, 4, 5, 6)
(1, 4, 5, 6)	(0, 2, 5, 6)
(1, 4, 5, 6)	(1, 2, 5, 7)

Rows to be removed	**Columns to be removed**
(1, 4, 5, 7)	(0, 1, 3, 7)
(1, 4, 5, 7)	(1, 2, 3, 5)
(1, 5, 6, 7)	(0, 1, 3, 5)
(1, 5, 6, 7)	(0, 2, 4, 5)
(1, 5, 6, 7)	(1, 2, 3, 7)
(1, 5, 6, 7)	(2, 3, 4, 6)
(2, 3, 4, 6)	(0, 1, 3, 7)
(2, 3, 4, 6)	(1, 5, 6, 7)
(2, 3, 4, 7)	(0, 2, 5, 6)
(2, 3, 4, 7)	(0, 3, 4, 5)
(2, 3, 4, 7)	(1, 3, 5, 6)
(2, 3, 5, 6)	(0, 2, 4, 7)
(2, 3, 5, 6)	(1, 2, 4, 5)
(2, 3, 5, 6)	(1, 3, 4, 7)
(2, 3, 5, 7)	(0, 1, 2, 6)
(2, 3, 5, 7)	(0, 4, 6, 7)
(2, 4, 5, 6)	(0, 1, 2, 4)
(2, 4, 5, 6)	(0, 1, 5, 7)
(2, 4, 5, 6)	(0, 2, 3, 6)
(2, 4, 5, 6)	(1, 3, 6, 7)
(2, 4, 6, 7)	(0, 1, 2, 6)
(2, 4, 6, 7)	(0, 2, 3, 4)
(2, 5, 6, 7)	(1, 2, 4, 6)
(2, 5, 6, 7)	(1, 3, 5, 6)
(3, 4, 5, 7)	(0, 1, 3, 5)
(3, 4, 5, 7)	(0, 1, 4, 6)
(3, 4, 5, 7)	(0, 2, 6, 7)
(3, 4, 5, 7)	(1, 2, 3, 7)
(3, 4, 6, 7)	(0, 2, 4, 7)
(3, 4, 6, 7)	(0, 3, 5, 7)
(3, 5, 6, 7)	(0, 1, 3, 7)
(3, 5, 6, 7)	(1, 2, 3, 5)
(4, 5, 6, 7)	(0, 3, 5, 6)
(4, 5, 6, 7)	(1, 2, 4, 7)
(0, 1, 2, 3, 4)	(0, 2, 5, 6, 7)
(0, 1, 2, 3, 4)	(1, 2, 3, 5, 6)
(0, 1, 2, 3, 5)	(0, 2, 3, 4, 7)
(0, 1, 2, 3, 5)	(1, 3, 4, 6, 7)
(0, 1, 2, 3, 6)	(0, 1, 3, 4, 7)
(0, 1, 2, 3, 6)	(0, 2, 4, 5, 7)
(0, 1, 2, 3, 7)	(0, 1, 2, 5, 6)
(0, 1, 2, 3, 7)	(1, 3, 4, 5, 6)
(0, 1, 2, 4, 5)	(1, 2, 3, 5, 7)
(0, 1, 2, 4, 6)	(0, 1, 2, 4, 7)
(0, 1, 2, 4, 6)	(1, 2, 3, 6, 7)
(0, 1, 2, 4, 7)	(0, 1, 2, 4, 6)
(0, 1, 2, 4, 7)	(0, 2, 3, 4, 5)
(0, 1, 2, 4, 7)	(0, 2, 3, 4, 7)
(0, 1, 2, 4, 7)	(0, 2, 3, 5, 7)
(0, 1, 2, 4, 7)	(0, 2, 4, 5, 7)
(0, 1, 2, 4, 7)	(0, 3, 4, 5, 7)
(0, 1, 2, 4, 7)	(2, 3, 4, 5, 7)
(0, 1, 2, 5, 6)	(0, 1, 2, 3, 7)
(0, 1, 2, 5, 6)	(0, 2, 3, 5, 6)
(0, 1, 2, 5, 6)	(1, 2, 4, 5, 7)
(0, 1, 2, 5, 7)	(0, 1, 4, 6, 7)
(0, 1, 2, 5, 7)	(0, 2, 3, 5, 6)
(0, 1, 2, 5, 7)	(1, 2, 4, 5, 7)
(0, 1, 2, 5, 7)	(3, 4, 5, 6, 7)
(0, 1, 2, 6, 7)	(0, 2, 3, 5, 6)
(0, 1, 2, 6, 7)	(1, 2, 4, 5, 7)
(0, 1, 2, 6, 7)	(1, 3, 4, 6, 7)
(0, 1, 3, 4, 5)	(0, 2, 3, 4, 6)
(0, 1, 3, 4, 6)	(0, 1, 5, 6, 7)
(0, 1, 3, 4, 6)	(0, 3, 4, 5, 6)
(0, 1, 3, 4, 6)	(1, 2, 3, 4, 7)
(0, 1, 3, 4, 6)	(2, 4, 5, 6, 7)
(0, 1, 3, 4, 7)	(0, 1, 2, 3, 6)
(0, 1, 3, 4, 7)	(0, 3, 4, 5, 6)

Rows to be removed	**Columns to be removed**
(0, 1, 3, 4, 7)	(1, 2, 3, 4, 7)
(0, 1, 3, 5, 6)	(0, 1, 3, 5, 7)
(0, 1, 3, 5, 6)	(1, 2, 3, 4, 5)
(0, 1, 3, 5, 6)	(1, 2, 3, 4, 6)
(0, 1, 3, 5, 6)	(1, 2, 3, 5, 6)
(0, 1, 3, 5, 6)	(1, 2, 4, 5, 6)
(0, 1, 3, 5, 6)	(1, 3, 4, 5, 6)
(0, 1, 3, 5, 6)	(2, 3, 4, 5, 6)
(0, 1, 3, 5, 7)	(0, 1, 3, 5, 6)
(0, 1, 3, 5, 7)	(0, 2, 3, 6, 7)
(0, 1, 3, 6, 7)	(0, 2, 5, 6, 7)
(0, 1, 3, 6, 7)	(0, 3, 4, 5, 6)
(0, 1, 3, 6, 7)	(1, 2, 3, 4, 7)
(0, 1, 4, 5, 6)	(1, 3, 5, 6, 7)
(0, 1, 4, 5, 7)	(0, 2, 4, 6, 7)
(0, 1, 4, 6, 7)	(0, 1, 2, 5, 7)
(0, 1, 4, 6, 7)	(0, 3, 4, 5, 6)
(0, 1, 4, 6, 7)	(1, 2, 3, 4, 7)
(0, 1, 5, 6, 7)	(0, 1, 3, 4, 6)
(0, 1, 5, 6, 7)	(0, 2, 3, 5, 6)
(0, 1, 5, 6, 7)	(1, 2, 4, 5, 7)
(0, 2, 3, 4, 5)	(0, 1, 2, 4, 7)
(0, 2, 3, 4, 5)	(0, 3, 5, 6, 7)
(0, 2, 3, 4, 5)	(1, 3, 4, 5, 6)
(0, 2, 3, 4, 6)	(0, 1, 3, 4, 5)
(0, 2, 3, 4, 6)	(0, 2, 3, 5, 6)
(0, 2, 3, 4, 7)	(0, 1, 2, 3, 5)
(0, 2, 3, 4, 7)	(0, 1, 2, 4, 7)
(0, 2, 3, 4, 7)	(0, 3, 5, 6, 7)
(0, 2, 3, 5, 6)	(0, 1, 2, 5, 6)
(0, 2, 3, 5, 6)	(0, 1, 2, 5, 7)
(0, 2, 3, 5, 6)	(0, 1, 2, 6, 7)
(0, 2, 3, 5, 6)	(0, 1, 5, 6, 7)
(0, 2, 3, 5, 6)	(0, 2, 3, 4, 6)
(0, 2, 3, 5, 6)	(0, 2, 5, 6, 7)
(0, 2, 3, 5, 6)	(1, 2, 5, 6, 7)
(0, 2, 3, 5, 7)	(0, 1, 2, 4, 7)
(0, 2, 3, 5, 7)	(0, 3, 5, 6, 7)
(0, 2, 3, 5, 7)	(1, 4, 5, 6, 7)
(0, 2, 3, 5, 7)	(2, 3, 4, 5, 6)
(0, 2, 3, 6, 7)	(0, 1, 3, 5, 7)
(0, 2, 4, 5, 6)	(0, 3, 4, 5, 6)
(0, 2, 4, 5, 6)	(2, 3, 5, 6, 7)
(0, 2, 4, 5, 7)	(0, 1, 2, 3, 6)
(0, 2, 4, 5, 7)	(0, 1, 2, 4, 7)
(0, 2, 4, 5, 7)	(0, 3, 5, 6, 7)
(0, 2, 4, 5, 7)	(1, 2, 3, 4, 5)
(0, 2, 4, 6, 7)	(0, 1, 4, 5, 7)
(0, 2, 4, 6, 7)	(1, 2, 4, 6, 7)
(0, 2, 5, 6, 7)	(0, 1, 2, 3, 4)
(0, 2, 5, 6, 7)	(0, 1, 3, 6, 7)
(0, 2, 5, 6, 7)	(0, 2, 3, 5, 6)
(0, 2, 5, 6, 7)	(1, 2, 4, 5, 7)
(0, 3, 4, 5, 6)	(0, 1, 3, 4, 6)
(0, 3, 4, 5, 6)	(0, 1, 3, 4, 7)
(0, 3, 4, 5, 6)	(0, 1, 3, 6, 7)
(0, 3, 4, 5, 6)	(0, 1, 4, 6, 7)
(0, 3, 4, 5, 6)	(0, 2, 4, 5, 6)
(0, 3, 4, 5, 6)	(0, 3, 4, 6, 7)
(0, 3, 4, 5, 6)	(1, 3, 4, 6, 7)
(0, 3, 4, 5, 7)	(0, 1, 2, 4, 7)
(0, 3, 4, 5, 7)	(0, 3, 5, 6, 7)
(0, 3, 4, 5, 7)	(2, 4, 5, 6, 7)
(0, 3, 4, 6, 7)	(0, 3, 4, 5, 6)
(0, 3, 4, 6, 7)	(1, 2, 3, 4, 7)
(0, 3, 4, 6, 7)	(1, 4, 5, 6, 7)
(0, 3, 5, 6, 7)	(0, 2, 3, 4, 5)
(0, 3, 5, 6, 7)	(0, 2, 3, 4, 7)
(0, 3, 5, 6, 7)	(0, 2, 3, 5, 7)

Rows to be removed	**Columns to be removed**
(0, 3, 5, 6, 7)	(0, 2, 4, 5, 7)
(0, 3, 5, 6, 7)	(0, 3, 4, 5, 7)
(0, 3, 5, 6, 7)	(1, 3, 5, 6, 7)
(0, 3, 5, 6, 7)	(2, 3, 4, 5, 7)
(0, 4, 5, 6, 7)	(1, 2, 3, 4, 6)
(0, 4, 5, 6, 7)	(1, 2, 5, 6, 7)
(1, 2, 3, 4, 5)	(0, 1, 3, 5, 6)
(1, 2, 3, 4, 5)	(0, 2, 4, 5, 7)
(1, 2, 3, 4, 5)	(1, 2, 4, 6, 7)
(1, 2, 3, 4, 6)	(0, 1, 3, 5, 6)
(1, 2, 3, 4, 6)	(0, 4, 5, 6, 7)
(1, 2, 3, 4, 6)	(1, 2, 4, 6, 7)
(1, 2, 3, 4, 6)	(2, 3, 4, 5, 7)
(1, 2, 3, 4, 7)	(0, 1, 3, 4, 6)
(1, 2, 3, 4, 7)	(0, 1, 3, 4, 7)
(1, 2, 3, 4, 7)	(0, 1, 3, 6, 7)
(1, 2, 3, 4, 7)	(0, 1, 4, 6, 7)
(1, 2, 3, 4, 7)	(0, 3, 4, 6, 7)
(1, 2, 3, 4, 7)	(1, 2, 3, 5, 7)
(1, 2, 3, 4, 7)	(1, 3, 4, 6, 7)
(1, 2, 3, 5, 6)	(0, 1, 2, 3, 4)
(1, 2, 3, 5, 6)	(0, 1, 3, 5, 6)
(1, 2, 3, 5, 6)	(1, 2, 4, 6, 7)
(1, 2, 3, 5, 7)	(0, 1, 2, 4, 5)
(1, 2, 3, 5, 7)	(1, 2, 3, 4, 7)
(1, 2, 3, 6, 7)	(0, 1, 2, 4, 6)
(1, 2, 4, 5, 6)	(0, 1, 3, 5, 6)
(1, 2, 4, 5, 6)	(1, 2, 4, 6, 7)
(1, 2, 4, 5, 6)	(3, 4, 5, 6, 7)
(1, 2, 4, 5, 7)	(0, 1, 2, 5, 6)
(1, 2, 4, 5, 7)	(0, 1, 2, 5, 7)
(1, 2, 4, 5, 7)	(0, 1, 2, 6, 7)
(1, 2, 4, 5, 7)	(0, 1, 5, 6, 7)
(1, 2, 4, 5, 7)	(0, 2, 5, 6, 7)
(1, 2, 4, 5, 7)	(1, 2, 5, 6, 7)
(1, 2, 4, 5, 7)	(1, 3, 4, 5, 7)
(1, 2, 4, 6, 7)	(0, 2, 4, 6, 7)
(1, 2, 4, 6, 7)	(1, 2, 3, 4, 5)
(1, 2, 4, 6, 7)	(1, 2, 3, 4, 6)
(1, 2, 4, 6, 7)	(1, 2, 3, 5, 6)
(1, 2, 4, 6, 7)	(1, 2, 4, 5, 6)
(1, 2, 4, 6, 7)	(1, 3, 4, 5, 6)
(1, 2, 4, 6, 7)	(2, 3, 4, 5, 6)
(1, 2, 5, 6, 7)	(0, 2, 3, 5, 6)
(1, 2, 5, 6, 7)	(0, 4, 5, 6, 7)
(1, 2, 5, 6, 7)	(1, 2, 4, 5, 7)
(1, 3, 4, 5, 6)	(0, 1, 2, 3, 7)
(1, 3, 4, 5, 6)	(0, 1, 3, 5, 6)
(1, 3, 4, 5, 6)	(0, 2, 3, 4, 5)
(1, 3, 4, 5, 6)	(1, 2, 4, 6, 7)
(1, 3, 4, 5, 7)	(1, 2, 4, 5, 7)
(1, 3, 4, 5, 7)	(2, 3, 4, 6, 7)
(1, 3, 4, 6, 7)	(0, 1, 2, 3, 5)
(1, 3, 4, 6, 7)	(0, 1, 2, 6, 7)
(1, 3, 4, 6, 7)	(0, 3, 4, 5, 6)
(1, 3, 4, 6, 7)	(1, 2, 3, 4, 7)
(1, 3, 5, 6, 7)	(0, 1, 4, 5, 6)
(1, 3, 5, 6, 7)	(0, 3, 5, 6, 7)
(1, 4, 5, 6, 7)	(0, 2, 3, 5, 7)
(1, 4, 5, 6, 7)	(0, 3, 4, 6, 7)
(2, 3, 4, 5, 6)	(0, 1, 3, 5, 6)
(2, 3, 4, 5, 6)	(0, 2, 3, 5, 7)
(2, 3, 4, 5, 6)	(1, 2, 4, 6, 7)
(2, 3, 4, 5, 7)	(0, 1, 2, 4, 7)
(2, 3, 4, 5, 7)	(0, 3, 5, 6, 7)
(2, 3, 4, 5, 7)	(1, 2, 3, 4, 6)
(2, 3, 4, 6, 7)	(1, 3, 4, 5, 7)
(2, 3, 5, 6, 7)	(0, 2, 4, 5, 6)
(2, 4, 5, 6, 7)	(0, 1, 3, 4, 6)

Rows to be removed	**Columns to be removed**
(2, 4, 5, 6, 7)	(0, 3, 4, 5, 7)
(3, 4, 5, 6, 7)	(0, 1, 2, 5, 7)
(3, 4, 5, 6, 7)	(1, 2, 4, 5, 6)
(0, 1, 2, 3, 4, 7)	(0, 1, 3, 4, 6, 7)
(0, 1, 2, 3, 4, 7)	(0, 2, 3, 4, 5, 7)
(0, 1, 2, 3, 5, 6)	(0, 1, 2, 5, 6, 7)
(0, 1, 2, 3, 5, 6)	(1, 2, 3, 4, 5, 6)
(0, 1, 2, 4, 5, 7)	(0, 1, 2, 5, 6, 7)
(0, 1, 2, 4, 5, 7)	(0, 2, 3, 4, 5, 7)
(0, 1, 2, 4, 6, 7)	(0, 2, 3, 4, 5, 7)
(0, 1, 2, 4, 6, 7)	(1, 2, 3, 4, 5, 6)
(0, 1, 2, 5, 6, 7)	(0, 1, 2, 3, 5, 6)
(0, 1, 2, 5, 6, 7)	(0, 1, 2, 4, 5, 7)
(0, 1, 2, 5, 6, 7)	(0, 2, 3, 4, 5, 6)
(0, 1, 2, 5, 6, 7)	(0, 2, 3, 5, 6, 7)
(0, 1, 2, 5, 6, 7)	(1, 2, 3, 4, 5, 7)
(0, 1, 2, 5, 6, 7)	(1, 2, 4, 5, 6, 7)
(0, 1, 3, 4, 5, 6)	(0, 1, 3, 4, 6, 7)
(0, 1, 3, 4, 5, 6)	(1, 2, 3, 4, 5, 6)
(0, 1, 3, 4, 6, 7)	(0, 1, 2, 3, 4, 7)
(0, 1, 3, 4, 6, 7)	(0, 1, 3, 4, 5, 6)
(0, 1, 3, 4, 6, 7)	(0, 2, 3, 4, 5, 6)
(0, 1, 3, 4, 6, 7)	(0, 3, 4, 5, 6, 7)
(0, 1, 3, 4, 6, 7)	(1, 2, 3, 4, 5, 7)
(0, 1, 3, 4, 6, 7)	(1, 2, 3, 4, 6, 7)
(0, 1, 3, 5, 6, 7)	(0, 2, 3, 4, 5, 7)
(0, 1, 3, 5, 6, 7)	(1, 2, 3, 4, 5, 6)
(0, 2, 3, 4, 5, 6)	(0, 1, 2, 5, 6, 7)
(0, 2, 3, 4, 5, 6)	(0, 1, 3, 4, 6, 7)
(0, 2, 3, 4, 5, 7)	(0, 1, 2, 3, 4, 7)
(0, 2, 3, 4, 5, 7)	(0, 1, 2, 4, 5, 7)
(0, 2, 3, 4, 5, 7)	(0, 1, 2, 4, 6, 7)
(0, 2, 3, 4, 5, 7)	(0, 1, 3, 5, 6, 7)
(0, 2, 3, 4, 5, 7)	(0, 2, 3, 5, 6, 7)
(0, 2, 3, 4, 5, 7)	(0, 3, 4, 5, 6, 7)
(0, 2, 3, 5, 6, 7)	(0, 1, 2, 5, 6, 7)
(0, 2, 3, 5, 6, 7)	(0, 2, 3, 4, 5, 7)
(0, 3, 4, 5, 6, 7)	(0, 1, 3, 4, 6, 7)
(0, 3, 4, 5, 6, 7)	(0, 2, 3, 4, 5, 7)
(1, 2, 3, 4, 5, 6)	(0, 1, 2, 3, 5, 6)
(1, 2, 3, 4, 5, 6)	(0, 1, 2, 4, 6, 7)
(1, 2, 3, 4, 5, 6)	(0, 1, 3, 4, 5, 6)
(1, 2, 3, 4, 5, 6)	(0, 1, 3, 5, 6, 7)
(1, 2, 3, 4, 5, 6)	(1, 2, 3, 4, 6, 7)
(1, 2, 3, 4, 5, 6)	(1, 2, 4, 5, 6, 7)
(1, 2, 3, 4, 5, 7)	(0, 1, 2, 5, 6, 7)
(1, 2, 3, 4, 5, 7)	(0, 1, 3, 4, 6, 7)
(1, 2, 3, 4, 6, 7)	(0, 1, 3, 4, 6, 7)
(1, 2, 3, 4, 6, 7)	(1, 2, 3, 4, 5, 6)
(1, 2, 4, 5, 6, 7)	(0, 1, 2, 5, 6, 7)
(1, 2, 4, 5, 6, 7)	(1, 2, 3, 4, 5, 6)

Table 7.16: Whirlwind M_0 (7.38) singular submatrices

Rows to be removed	**Columns to be removed**
(0, 2)	(1, 5)
(0, 2)	(3, 7)
(0, 4)	(0, 5)
(0, 4)	(0, 7)
(0, 4)	(1, 3)
(0, 4)	(1, 4)
(0, 4)	(3, 4)
(0, 4)	(5, 7)
(0, 5)	(0, 4)
(0, 5)	(1, 5)
(0, 7)	(0, 4)
(0, 7)	(3, 7)
(1, 3)	(0, 4)

Rows to be removed	**Columns to be removed**
(1, 3)	(2, 6)
(1, 4)	(0, 4)
(1, 4)	(1, 5)
(1, 5)	(0, 2)
(1, 5)	(0, 5)
(1, 5)	(1, 4)
(1, 5)	(1, 6)
(1, 5)	(2, 5)
(1, 5)	(4, 6)
(1, 6)	(1, 5)
(1, 6)	(2, 6)
(2, 5)	(1, 5)
(2, 5)	(2, 6)
(2, 6)	(1, 3)
(2, 6)	(1, 6)
(2, 6)	(2, 5)
(2, 6)	(2, 7)
(2, 6)	(3, 6)
(2, 6)	(5, 7)
(2, 7)	(2, 6)
(2, 7)	(3, 7)
(3, 4)	(0, 4)
(3, 4)	(3, 7)
(3, 6)	(2, 6)
(3, 6)	(3, 7)
(3, 7)	(0, 2)
(3, 7)	(0, 7)
(3, 7)	(2, 7)
(3, 7)	(3, 4)
(3, 7)	(3, 6)
(3, 7)	(4, 6)
(4, 6)	(1, 5)
(4, 6)	(3, 7)
(5, 7)	(0, 4)
(5, 7)	(2, 6)
(0, 1, 2)	(0, 3, 5)
(0, 1, 2)	(0, 3, 6)
(0, 1, 2)	(0, 5, 6)
(0, 1, 2)	(1, 2, 4)
(0, 1, 2)	(1, 2, 7)
(0, 1, 2)	(1, 4, 7)
(0, 1, 2)	(2, 4, 7)
(0, 1, 2)	(3, 5, 6)
(0, 1, 3)	(0, 3, 5)
(0, 1, 3)	(0, 3, 6)
(0, 1, 3)	(0, 5, 6)
(0, 1, 3)	(1, 2, 4)
(0, 1, 3)	(1, 2, 7)
(0, 1, 3)	(1, 4, 7)
(0, 1, 3)	(2, 4, 7)
(0, 1, 3)	(3, 5, 6)
(0, 1, 4)	(0, 5, 7)
(0, 1, 4)	(1, 3, 4)
(0, 1, 5)	(0, 2, 5)
(0, 1, 5)	(1, 4, 6)
(0, 1, 6)	(3, 5, 7)
(0, 1, 7)	(2, 4, 6)
(0, 2, 3)	(0, 3, 5)
(0, 2, 3)	(0, 3, 6)
(0, 2, 3)	(0, 5, 6)
(0, 2, 3)	(1, 2, 4)
(0, 2, 3)	(1, 2, 7)
(0, 2, 3)	(1, 4, 7)
(0, 2, 3)	(2, 4, 7)
(0, 2, 3)	(3, 5, 6)
(0, 2, 4)	(0, 5, 7)
(0, 2, 4)	(1, 3, 4)
(0, 2, 4)	(1, 6, 7)
(0, 2, 4)	(2, 5, 6)

Rows to be removed	**Columns to be removed**
(0, 2, 5)	(0, 1, 5)
(0, 2, 5)	(1, 2, 5)
(0, 2, 5)	(1, 3, 5)
(0, 2, 5)	(1, 4, 5)
(0, 2, 5)	(1, 5, 6)
(0, 2, 5)	(1, 5, 7)
(0, 2, 6)	(0, 4, 7)
(0, 2, 6)	(1, 3, 6)
(0, 2, 6)	(2, 5, 7)
(0, 2, 6)	(3, 4, 5)
(0, 2, 7)	(0, 3, 7)
(0, 2, 7)	(1, 3, 7)
(0, 2, 7)	(2, 3, 7)
(0, 2, 7)	(3, 4, 7)
(0, 2, 7)	(3, 5, 7)
(0, 2, 7)	(3, 6, 7)
(0, 3, 4)	(0, 5, 7)
(0, 3, 4)	(1, 3, 4)
(0, 3, 4)	(2, 4, 6)
(0, 3, 5)	(0, 1, 2)
(0, 3, 5)	(0, 1, 3)
(0, 3, 5)	(0, 2, 3)
(0, 3, 5)	(1, 2, 3)
(0, 3, 5)	(4, 5, 6)
(0, 3, 5)	(4, 5, 7)
(0, 3, 5)	(4, 6, 7)
(0, 3, 5)	(5, 6, 7)
(0, 3, 6)	(0, 1, 2)
(0, 3, 6)	(0, 1, 3)
(0, 3, 6)	(0, 2, 3)
(0, 3, 6)	(1, 2, 3)
(0, 3, 6)	(4, 5, 6)
(0, 3, 6)	(4, 5, 7)
(0, 3, 6)	(4, 6, 7)
(0, 3, 6)	(5, 6, 7)
(0, 3, 7)	(0, 2, 7)
(0, 3, 7)	(1, 5, 7)
(0, 3, 7)	(3, 4, 6)
(0, 4, 5)	(0, 5, 7)
(0, 4, 5)	(1, 3, 4)
(0, 4, 6)	(0, 5, 7)
(0, 4, 6)	(1, 2, 6)
(0, 4, 6)	(1, 3, 4)
(0, 4, 6)	(2, 3, 5)
(0, 4, 7)	(0, 2, 6)
(0, 4, 7)	(0, 5, 7)
(0, 4, 7)	(1, 3, 4)
(0, 5, 6)	(0, 1, 2)
(0, 5, 6)	(0, 1, 3)
(0, 5, 6)	(0, 2, 3)
(0, 5, 6)	(1, 2, 3)
(0, 5, 6)	(4, 5, 6)
(0, 5, 6)	(4, 5, 7)
(0, 5, 6)	(4, 6, 7)
(0, 5, 6)	(5, 6, 7)
(0, 5, 7)	(0, 1, 4)
(0, 5, 7)	(0, 2, 4)
(0, 5, 7)	(0, 3, 4)
(0, 5, 7)	(0, 4, 5)
(0, 5, 7)	(0, 4, 6)
(0, 5, 7)	(0, 4, 7)
(0, 6, 7)	(1, 3, 5)
(1, 2, 3)	(0, 3, 5)
(1, 2, 3)	(0, 3, 6)
(1, 2, 3)	(0, 5, 6)
(1, 2, 3)	(1, 2, 4)
(1, 2, 3)	(1, 2, 7)
(1, 2, 3)	(1, 4, 7)
(1, 2, 3)	(2, 4, 7)

Rows to be removed	**Columns to be removed**
(1, 2, 3)	(3, 5, 6)
(1, 2, 4)	(0, 1, 2)
(1, 2, 4)	(0, 1, 3)
(1, 2, 4)	(0, 2, 3)
(1, 2, 4)	(1, 2, 3)
(1, 2, 4)	(4, 5, 6)
(1, 2, 4)	(4, 5, 7)
(1, 2, 4)	(4, 6, 7)
(1, 2, 4)	(5, 6, 7)
(1, 2, 5)	(0, 2, 5)
(1, 2, 5)	(1, 4, 6)
(1, 2, 5)	(3, 5, 7)
(1, 2, 6)	(0, 4, 6)
(1, 2, 6)	(1, 3, 6)
(1, 2, 6)	(2, 5, 7)
(1, 2, 7)	(0, 1, 2)
(1, 2, 7)	(0, 1, 3)
(1, 2, 7)	(0, 2, 3)
(1, 2, 7)	(1, 2, 3)
(1, 2, 7)	(4, 5, 6)
(1, 2, 7)	(4, 5, 7)
(1, 2, 7)	(4, 6, 7)
(1, 2, 7)	(5, 6, 7)
(1, 3, 4)	(0, 1, 4)
(1, 3, 4)	(0, 2, 4)
(1, 3, 4)	(0, 3, 4)
(1, 3, 4)	(0, 4, 5)
(1, 3, 4)	(0, 4, 6)
(1, 3, 4)	(0, 4, 7)
(1, 3, 5)	(0, 2, 5)
(1, 3, 5)	(0, 6, 7)
(1, 3, 5)	(1, 4, 6)
(1, 3, 5)	(3, 4, 7)
(1, 3, 6)	(0, 2, 6)
(1, 3, 6)	(1, 2, 6)
(1, 3, 6)	(2, 3, 6)
(1, 3, 6)	(2, 4, 6)
(1, 3, 6)	(2, 5, 6)
(1, 3, 6)	(2, 6, 7)
(1, 3, 7)	(0, 2, 7)
(1, 3, 7)	(1, 5, 6)
(1, 3, 7)	(2, 4, 5)
(1, 3, 7)	(3, 4, 6)
(1, 4, 5)	(0, 2, 5)
(1, 4, 5)	(1, 4, 6)
(1, 4, 6)	(0, 1, 5)
(1, 4, 6)	(1, 2, 5)
(1, 4, 6)	(1, 3, 5)
(1, 4, 6)	(1, 4, 5)
(1, 4, 6)	(1, 5, 6)
(1, 4, 6)	(1, 5, 7)
(1, 4, 7)	(0, 1, 2)
(1, 4, 7)	(0, 1, 3)
(1, 4, 7)	(0, 2, 3)
(1, 4, 7)	(1, 2, 3)
(1, 4, 7)	(4, 5, 6)
(1, 4, 7)	(4, 5, 7)
(1, 4, 7)	(4, 6, 7)
(1, 4, 7)	(5, 6, 7)
(1, 5, 6)	(0, 2, 5)
(1, 5, 6)	(1, 3, 7)
(1, 5, 6)	(1, 4, 6)
(1, 5, 7)	(0, 2, 5)
(1, 5, 7)	(0, 3, 7)
(1, 5, 7)	(1, 4, 6)
(1, 5, 7)	(2, 3, 4)
(1, 6, 7)	(0, 2, 4)
(2, 3, 4)	(1, 5, 7)
(2, 3, 5)	(0, 4, 6)

Rows to be removed	**Columns to be removed**
(2, 3, 6)	(1, 3, 6)
(2, 3, 6)	(2, 5, 7)
(2, 3, 7)	(0, 2, 7)
(2, 3, 7)	(3, 4, 6)
(2, 4, 5)	(1, 3, 7)
(2, 4, 6)	(0, 1, 7)
(2, 4, 6)	(0, 3, 4)
(2, 4, 6)	(1, 3, 6)
(2, 4, 6)	(2, 5, 7)
(2, 4, 7)	(0, 1, 2)
(2, 4, 7)	(0, 1, 3)
(2, 4, 7)	(0, 2, 3)
(2, 4, 7)	(1, 2, 3)
(2, 4, 7)	(4, 5, 6)
(2, 4, 7)	(4, 5, 7)
(2, 4, 7)	(4, 6, 7)
(2, 4, 7)	(5, 6, 7)
(2, 5, 6)	(0, 2, 4)
(2, 5, 6)	(1, 3, 6)
(2, 5, 6)	(2, 5, 7)
(2, 5, 7)	(0, 2, 6)
(2, 5, 7)	(1, 2, 6)
(2, 5, 7)	(2, 3, 6)
(2, 5, 7)	(2, 4, 6)
(2, 5, 7)	(2, 5, 6)
(2, 5, 7)	(2, 6, 7)
(2, 6, 7)	(1, 3, 6)
(2, 6, 7)	(2, 5, 7)
(3, 4, 5)	(0, 2, 6)
(3, 4, 6)	(0, 3, 7)
(3, 4, 6)	(1, 3, 7)
(3, 4, 6)	(2, 3, 7)
(3, 4, 6)	(3, 4, 7)
(3, 4, 6)	(3, 5, 7)
(3, 4, 6)	(3, 6, 7)
(3, 4, 7)	(0, 2, 7)
(3, 4, 7)	(1, 3, 5)
(3, 4, 7)	(3, 4, 6)
(3, 5, 6)	(0, 1, 2)
(3, 5, 6)	(0, 1, 3)
(3, 5, 6)	(0, 2, 3)
(3, 5, 6)	(1, 2, 3)
(3, 5, 6)	(4, 5, 6)
(3, 5, 6)	(4, 5, 7)
(3, 5, 6)	(4, 6, 7)
(3, 5, 6)	(5, 6, 7)
(3, 5, 7)	(0, 1, 6)
(3, 5, 7)	(0, 2, 7)
(3, 5, 7)	(1, 2, 5)
(3, 5, 7)	(3, 4, 6)
(3, 6, 7)	(0, 2, 7)
(3, 6, 7)	(3, 4, 6)
(4, 5, 6)	(0, 3, 5)
(4, 5, 6)	(0, 3, 6)
(4, 5, 6)	(0, 5, 6)
(4, 5, 6)	(1, 2, 4)
(4, 5, 6)	(1, 2, 7)
(4, 5, 6)	(1, 4, 7)
(4, 5, 6)	(2, 4, 7)
(4, 5, 6)	(3, 5, 6)
(4, 5, 7)	(0, 3, 5)
(4, 5, 7)	(0, 3, 6)
(4, 5, 7)	(0, 5, 6)
(4, 5, 7)	(1, 2, 4)
(4, 5, 7)	(1, 2, 7)
(4, 5, 7)	(1, 4, 7)
(4, 5, 7)	(2, 4, 7)
(4, 5, 7)	(3, 5, 6)
(4, 6, 7)	(0, 3, 5)

Rows to be removed	**Columns to be removed**
(4, 6, 7)	(0, 3, 6)
(4, 6, 7)	(0, 5, 6)
(4, 6, 7)	(1, 2, 4)
(4, 6, 7)	(1, 2, 7)
(4, 6, 7)	(1, 4, 7)
(4, 6, 7)	(2, 4, 7)
(4, 6, 7)	(3, 5, 6)
(5, 6, 7)	(0, 3, 5)
(5, 6, 7)	(0, 3, 6)
(5, 6, 7)	(0, 5, 6)
(5, 6, 7)	(1, 2, 4)
(5, 6, 7)	(1, 2, 7)
(5, 6, 7)	(1, 4, 7)
(5, 6, 7)	(2, 4, 7)
(5, 6, 7)	(3, 5, 6)
(0, 1, 2, 3)	(0, 1, 2, 4)
(0, 1, 2, 3)	(0, 1, 2, 7)
(0, 1, 2, 3)	(0, 1, 3, 5)
(0, 1, 2, 3)	(0, 1, 3, 6)
(0, 1, 2, 3)	(0, 1, 4, 7)
(0, 1, 2, 3)	(0, 1, 5, 6)
(0, 1, 2, 3)	(0, 2, 3, 5)
(0, 1, 2, 3)	(0, 2, 3, 6)
(0, 1, 2, 3)	(0, 2, 4, 7)
(0, 1, 2, 3)	(0, 2, 5, 6)
(0, 1, 2, 3)	(0, 3, 4, 5)
(0, 1, 2, 3)	(0, 3, 4, 6)
(0, 1, 2, 3)	(0, 3, 5, 6)
(0, 1, 2, 3)	(0, 3, 5, 7)
(0, 1, 2, 3)	(0, 3, 6, 7)
(0, 1, 2, 3)	(0, 4, 5, 6)
(0, 1, 2, 3)	(0, 5, 6, 7)
(0, 1, 2, 3)	(1, 2, 3, 4)
(0, 1, 2, 3)	(1, 2, 3, 7)
(0, 1, 2, 3)	(1, 2, 4, 5)
(0, 1, 2, 3)	(1, 2, 4, 6)
(0, 1, 2, 3)	(1, 2, 4, 7)
(0, 1, 2, 3)	(1, 2, 5, 7)
(0, 1, 2, 3)	(1, 2, 6, 7)
(0, 1, 2, 3)	(1, 3, 4, 7)
(0, 1, 2, 3)	(1, 3, 5, 6)
(0, 1, 2, 3)	(1, 4, 5, 7)
(0, 1, 2, 3)	(1, 4, 6, 7)
(0, 1, 2, 3)	(2, 3, 4, 7)
(0, 1, 2, 3)	(2, 3, 5, 6)
(0, 1, 2, 3)	(2, 4, 5, 7)
(0, 1, 2, 3)	(2, 4, 6, 7)
(0, 1, 2, 3)	(3, 4, 5, 6)
(0, 1, 2, 3)	(3, 5, 6, 7)
(0, 1, 2, 4)	(0, 1, 2, 3)
(0, 1, 2, 4)	(0, 1, 4, 5)
(0, 1, 2, 4)	(0, 2, 4, 6)
(0, 1, 2, 4)	(0, 3, 5, 6)
(0, 1, 2, 4)	(1, 2, 4, 7)
(0, 1, 2, 4)	(1, 3, 5, 7)
(0, 1, 2, 4)	(2, 3, 6, 7)
(0, 1, 2, 4)	(4, 5, 6, 7)
(0, 1, 2, 5)	(0, 3, 5, 6)
(0, 1, 2, 5)	(1, 2, 4, 7)
(0, 1, 2, 5)	(2, 5, 6, 7)
(0, 1, 2, 6)	(0, 1, 2, 6)
(0, 1, 2, 6)	(0, 3, 5, 6)
(0, 1, 2, 6)	(1, 2, 4, 7)
(0, 1, 2, 7)	(0, 1, 2, 3)
(0, 1, 2, 7)	(0, 1, 6, 7)
(0, 1, 2, 7)	(0, 2, 5, 7)
(0, 1, 2, 7)	(0, 3, 5, 6)
(0, 1, 2, 7)	(1, 2, 4, 7)
(0, 1, 2, 7)	(1, 3, 4, 6)

Rows to be removed	**Columns to be removed**
(0, 1, 2, 7)	(2, 3, 4, 5)
(0, 1, 2, 7)	(4, 5, 6, 7)
(0, 1, 3, 4)	(0, 3, 5, 6)
(0, 1, 3, 4)	(1, 2, 4, 7)
(0, 1, 3, 4)	(3, 4, 6, 7)
(0, 1, 3, 5)	(0, 1, 2, 3)
(0, 1, 3, 5)	(0, 1, 4, 5)
(0, 1, 3, 5)	(0, 2, 4, 6)
(0, 1, 3, 5)	(0, 3, 5, 6)
(0, 1, 3, 5)	(1, 2, 4, 7)
(0, 1, 3, 5)	(1, 3, 5, 7)
(0, 1, 3, 5)	(2, 3, 6, 7)
(0, 1, 3, 5)	(4, 5, 6, 7)
(0, 1, 3, 6)	(0, 1, 2, 3)
(0, 1, 3, 6)	(0, 1, 6, 7)
(0, 1, 3, 6)	(0, 2, 5, 7)
(0, 1, 3, 6)	(0, 3, 5, 6)
(0, 1, 3, 6)	(1, 2, 4, 7)
(0, 1, 3, 6)	(1, 3, 4, 6)
(0, 1, 3, 6)	(2, 3, 4, 5)
(0, 1, 3, 6)	(4, 5, 6, 7)
(0, 1, 3, 7)	(0, 1, 3, 7)
(0, 1, 3, 7)	(0, 3, 5, 6)
(0, 1, 3, 7)	(1, 2, 4, 7)
(0, 1, 4, 5)	(0, 1, 2, 4)
(0, 1, 4, 5)	(0, 1, 3, 5)
(0, 1, 4, 5)	(0, 1, 6, 7)
(0, 1, 4, 5)	(0, 2, 3, 6)
(0, 1, 4, 5)	(0, 4, 5, 6)
(0, 1, 4, 5)	(1, 2, 3, 7)
(0, 1, 4, 5)	(1, 4, 5, 7)
(0, 1, 4, 5)	(2, 3, 4, 5)
(0, 1, 4, 5)	(2, 4, 6, 7)
(0, 1, 4, 5)	(3, 5, 6, 7)
(0, 1, 4, 6)	(0, 2, 4, 7)
(0, 1, 4, 6)	(0, 2, 5, 6)
(0, 1, 4, 7)	(0, 1, 2, 3)
(0, 1, 4, 7)	(4, 5, 6, 7)
(0, 1, 5, 6)	(0, 1, 2, 3)
(0, 1, 5, 6)	(4, 5, 6, 7)
(0, 1, 5, 7)	(1, 3, 4, 7)
(0, 1, 5, 7)	(1, 3, 5, 6)
(0, 1, 6, 7)	(0, 1, 2, 7)
(0, 1, 6, 7)	(0, 1, 3, 6)
(0, 1, 6, 7)	(0, 1, 4, 5)
(0, 1, 6, 7)	(0, 2, 3, 5)
(0, 1, 6, 7)	(0, 5, 6, 7)
(0, 1, 6, 7)	(1, 2, 3, 4)
(0, 1, 6, 7)	(1, 4, 6, 7)
(0, 1, 6, 7)	(2, 3, 6, 7)
(0, 1, 6, 7)	(2, 4, 5, 7)
(0, 1, 6, 7)	(3, 4, 5, 6)
(0, 2, 3, 4)	(0, 2, 3, 4)
(0, 2, 3, 4)	(0, 3, 5, 6)
(0, 2, 3, 4)	(1, 2, 4, 7)
(0, 2, 3, 5)	(0, 1, 2, 3)
(0, 2, 3, 5)	(0, 1, 6, 7)
(0, 2, 3, 5)	(0, 2, 5, 7)
(0, 2, 3, 5)	(0, 3, 5, 6)
(0, 2, 3, 5)	(1, 2, 4, 7)
(0, 2, 3, 5)	(1, 3, 4, 6)
(0, 2, 3, 5)	(2, 3, 4, 5)
(0, 2, 3, 5)	(4, 5, 6, 7)
(0, 2, 3, 6)	(0, 1, 2, 3)
(0, 2, 3, 6)	(0, 1, 4, 5)
(0, 2, 3, 6)	(0, 2, 4, 6)
(0, 2, 3, 6)	(0, 3, 5, 6)
(0, 2, 3, 6)	(1, 2, 4, 7)
(0, 2, 3, 6)	(1, 3, 5, 7)

Rows to be removed	**Columns to be removed**
(0, 2, 3, 6)	(2, 3, 6, 7)
(0, 2, 3, 6)	(4, 5, 6, 7)
(0, 2, 3, 7)	(0, 3, 5, 6)
(0, 2, 3, 7)	(0, 4, 5, 7)
(0, 2, 3, 7)	(1, 2, 4, 7)
(0, 2, 4, 5)	(0, 3, 4, 6)
(0, 2, 4, 5)	(1, 2, 4, 6)
(0, 2, 4, 6)	(0, 1, 2, 4)
(0, 2, 4, 6)	(0, 1, 3, 5)
(0, 2, 4, 6)	(0, 2, 3, 6)
(0, 2, 4, 6)	(0, 4, 5, 6)
(0, 2, 4, 6)	(1, 2, 3, 7)
(0, 2, 4, 6)	(1, 4, 5, 7)
(0, 2, 4, 6)	(2, 4, 6, 7)
(0, 2, 4, 6)	(3, 5, 6, 7)
(0, 2, 4, 7)	(0, 1, 2, 3)
(0, 2, 4, 7)	(0, 1, 4, 6)
(0, 2, 4, 7)	(2, 3, 4, 6)
(0, 2, 4, 7)	(4, 5, 6, 7)
(0, 2, 5, 6)	(0, 1, 2, 3)
(0, 2, 5, 6)	(0, 1, 4, 6)
(0, 2, 5, 6)	(2, 3, 4, 6)
(0, 2, 5, 6)	(4, 5, 6, 7)
(0, 2, 5, 7)	(0, 1, 2, 7)
(0, 2, 5, 7)	(0, 1, 3, 6)
(0, 2, 5, 7)	(0, 2, 3, 5)
(0, 2, 5, 7)	(0, 5, 6, 7)
(0, 2, 5, 7)	(1, 2, 3, 4)
(0, 2, 5, 7)	(1, 4, 6, 7)
(0, 2, 5, 7)	(2, 4, 5, 7)
(0, 2, 5, 7)	(3, 4, 5, 6)
(0, 2, 6, 7)	(0, 3, 4, 6)
(0, 2, 6, 7)	(1, 2, 4, 6)
(0, 3, 4, 5)	(0, 1, 2, 3)
(0, 3, 4, 5)	(4, 5, 6, 7)
(0, 3, 4, 6)	(0, 1, 2, 3)
(0, 3, 4, 6)	(0, 2, 4, 5)
(0, 3, 4, 6)	(0, 2, 6, 7)
(0, 3, 4, 6)	(4, 5, 6, 7)
(0, 3, 5, 6)	(0, 1, 2, 3)
(0, 3, 5, 6)	(0, 1, 2, 4)
(0, 3, 5, 6)	(0, 1, 2, 5)
(0, 3, 5, 6)	(0, 1, 2, 6)
(0, 3, 5, 6)	(0, 1, 2, 7)
(0, 3, 5, 6)	(0, 1, 3, 4)
(0, 3, 5, 6)	(0, 1, 3, 5)
(0, 3, 5, 6)	(0, 1, 3, 6)
(0, 3, 5, 6)	(0, 1, 3, 7)
(0, 3, 5, 6)	(0, 2, 3, 4)
(0, 3, 5, 6)	(0, 2, 3, 5)
(0, 3, 5, 6)	(0, 2, 3, 6)
(0, 3, 5, 6)	(0, 2, 3, 7)
(0, 3, 5, 6)	(0, 4, 5, 6)
(0, 3, 5, 6)	(0, 4, 5, 7)
(0, 3, 5, 6)	(0, 4, 6, 7)
(0, 3, 5, 6)	(0, 5, 6, 7)
(0, 3, 5, 6)	(1, 2, 3, 4)
(0, 3, 5, 6)	(1, 2, 3, 5)
(0, 3, 5, 6)	(1, 2, 3, 6)
(0, 3, 5, 6)	(1, 2, 3, 7)
(0, 3, 5, 6)	(1, 4, 5, 6)
(0, 3, 5, 6)	(1, 4, 5, 7)
(0, 3, 5, 6)	(1, 4, 6, 7)
(0, 3, 5, 6)	(1, 5, 6, 7)
(0, 3, 5, 6)	(2, 4, 5, 6)
(0, 3, 5, 6)	(2, 4, 5, 7)
(0, 3, 5, 6)	(2, 4, 6, 7)
(0, 3, 5, 6)	(2, 5, 6, 7)
(0, 3, 5, 6)	(3, 4, 5, 6)

Rows to be removed	Columns to be removed
(0, 3, 5, 6)	(3, 4, 5, 7)
(0, 3, 5, 6)	(3, 4, 6, 7)
(0, 3, 5, 6)	(3, 5, 6, 7)
(0, 3, 5, 6)	(4, 5, 6, 7)
(0, 3, 5, 7)	(0, 1, 2, 3)
(0, 3, 5, 7)	(1, 3, 4, 5)
(0, 3, 5, 7)	(1, 3, 6, 7)
(0, 3, 5, 7)	(4, 5, 6, 7)
(0, 3, 6, 7)	(0, 1, 2, 3)
(0, 3, 6, 7)	(4, 5, 6, 7)
(0, 4, 5, 6)	(0, 1, 2, 3)
(0, 4, 5, 6)	(0, 1, 4, 5)
(0, 4, 5, 6)	(0, 2, 4, 6)
(0, 4, 5, 6)	(0, 3, 5, 6)
(0, 4, 5, 6)	(1, 2, 4, 7)
(0, 4, 5, 6)	(1, 3, 5, 7)
(0, 4, 5, 6)	(2, 3, 6, 7)
(0, 4, 5, 6)	(4, 5, 6, 7)
(0, 4, 5, 7)	(0, 2, 3, 7)
(0, 4, 5, 7)	(0, 3, 5, 6)
(0, 4, 5, 7)	(1, 2, 4, 7)
(0, 4, 6, 7)	(0, 3, 5, 6)
(0, 4, 6, 7)	(0, 4, 6, 7)
(0, 4, 6, 7)	(1, 2, 4, 7)
(0, 5, 6, 7)	(0, 1, 2, 3)
(0, 5, 6, 7)	(0, 1, 6, 7)
(0, 5, 6, 7)	(0, 2, 5, 7)
(0, 5, 6, 7)	(0, 3, 5, 6)
(0, 5, 6, 7)	(1, 2, 4, 7)
(0, 5, 6, 7)	(1, 3, 4, 6)
(0, 5, 6, 7)	(2, 3, 4, 5)
(0, 5, 6, 7)	(4, 5, 6, 7)
(1, 2, 3, 4)	(0, 1, 2, 3)
(1, 2, 3, 4)	(0, 1, 6, 7)
(1, 2, 3, 4)	(0, 2, 5, 7)
(1, 2, 3, 4)	(0, 3, 5, 6)
(1, 2, 3, 4)	(1, 2, 4, 7)
(1, 2, 3, 4)	(1, 3, 4, 6)
(1, 2, 3, 4)	(2, 3, 4, 5)
(1, 2, 3, 4)	(4, 5, 6, 7)
(1, 2, 3, 5)	(0, 3, 5, 6)
(1, 2, 3, 5)	(1, 2, 3, 5)
(1, 2, 3, 5)	(1, 2, 4, 7)
(1, 2, 3, 6)	(0, 3, 5, 6)
(1, 2, 3, 6)	(1, 2, 4, 7)
(1, 2, 3, 6)	(1, 4, 5, 6)
(1, 2, 3, 7)	(0, 1, 2, 3)
(1, 2, 3, 7)	(0, 1, 4, 5)
(1, 2, 3, 7)	(0, 2, 4, 6)
(1, 2, 3, 7)	(0, 3, 5, 6)
(1, 2, 3, 7)	(1, 2, 4, 7)
(1, 2, 3, 7)	(1, 3, 5, 7)
(1, 2, 3, 7)	(2, 3, 6, 7)
(1, 2, 3, 7)	(4, 5, 6, 7)
(1, 2, 4, 5)	(0, 1, 2, 3)
(1, 2, 4, 5)	(4, 5, 6, 7)
(1, 2, 4, 6)	(0, 1, 2, 3)
(1, 2, 4, 6)	(0, 2, 4, 5)
(1, 2, 4, 6)	(0, 2, 6, 7)
(1, 2, 4, 6)	(4, 5, 6, 7)
(1, 2, 4, 7)	(0, 1, 2, 3)
(1, 2, 4, 7)	(0, 1, 2, 4)
(1, 2, 4, 7)	(0, 1, 2, 5)
(1, 2, 4, 7)	(0, 1, 2, 6)
(1, 2, 4, 7)	(0, 1, 2, 7)
(1, 2, 4, 7)	(0, 1, 3, 4)
(1, 2, 4, 7)	(0, 1, 3, 5)
(1, 2, 4, 7)	(0, 1, 3, 6)
(1, 2, 4, 7)	(0, 1, 3, 7)

Rows to be removed	**Columns to be removed**
(1, 2, 4, 7)	(0, 2, 3, 4)
(1, 2, 4, 7)	(0, 2, 3, 5)
(1, 2, 4, 7)	(0, 2, 3, 6)
(1, 2, 4, 7)	(0, 2, 3, 7)
(1, 2, 4, 7)	(0, 4, 5, 6)
(1, 2, 4, 7)	(0, 4, 5, 7)
(1, 2, 4, 7)	(0, 4, 6, 7)
(1, 2, 4, 7)	(0, 5, 6, 7)
(1, 2, 4, 7)	(1, 2, 3, 4)
(1, 2, 4, 7)	(1, 2, 3, 5)
(1, 2, 4, 7)	(1, 2, 3, 6)
(1, 2, 4, 7)	(1, 2, 3, 7)
(1, 2, 4, 7)	(1, 4, 5, 6)
(1, 2, 4, 7)	(1, 4, 5, 7)
(1, 2, 4, 7)	(1, 4, 6, 7)
(1, 2, 4, 7)	(1, 5, 6, 7)
(1, 2, 4, 7)	(2, 4, 5, 6)
(1, 2, 4, 7)	(2, 4, 5, 7)
(1, 2, 4, 7)	(2, 4, 6, 7)
(1, 2, 4, 7)	(2, 5, 6, 7)
(1, 2, 4, 7)	(3, 4, 5, 6)
(1, 2, 4, 7)	(3, 4, 5, 7)
(1, 2, 4, 7)	(3, 4, 6, 7)
(1, 2, 4, 7)	(3, 5, 6, 7)
(1, 2, 4, 7)	(4, 5, 6, 7)
(1, 2, 5, 7)	(0, 1, 2, 3)
(1, 2, 5, 7)	(1, 3, 4, 5)
(1, 2, 5, 7)	(1, 3, 6, 7)
(1, 2, 5, 7)	(4, 5, 6, 7)
(1, 2, 6, 7)	(0, 1, 2, 3)
(1, 2, 6, 7)	(4, 5, 6, 7)
(1, 3, 4, 5)	(0, 3, 5, 7)
(1, 3, 4, 5)	(1, 2, 5, 7)
(1, 3, 4, 6)	(0, 1, 2, 7)
(1, 3, 4, 6)	(0, 1, 3, 6)
(1, 3, 4, 6)	(0, 2, 3, 5)
(1, 3, 4, 6)	(0, 5, 6, 7)
(1, 3, 4, 6)	(1, 2, 3, 4)
(1, 3, 4, 6)	(1, 4, 6, 7)
(1, 3, 4, 6)	(2, 4, 5, 7)
(1, 3, 4, 6)	(3, 4, 5, 6)
(1, 3, 4, 7)	(0, 1, 2, 3)
(1, 3, 4, 7)	(0, 1, 5, 7)
(1, 3, 4, 7)	(2, 3, 5, 7)
(1, 3, 4, 7)	(4, 5, 6, 7)
(1, 3, 5, 6)	(0, 1, 2, 3)
(1, 3, 5, 6)	(0, 1, 5, 7)
(1, 3, 5, 6)	(2, 3, 5, 7)
(1, 3, 5, 6)	(4, 5, 6, 7)
(1, 3, 5, 7)	(0, 1, 2, 4)
(1, 3, 5, 7)	(0, 1, 3, 5)
(1, 3, 5, 7)	(0, 2, 3, 6)
(1, 3, 5, 7)	(0, 4, 5, 6)
(1, 3, 5, 7)	(1, 2, 3, 7)
(1, 3, 5, 7)	(1, 4, 5, 7)
(1, 3, 5, 7)	(2, 4, 6, 7)
(1, 3, 5, 7)	(3, 5, 6, 7)
(1, 3, 6, 7)	(0, 3, 5, 7)
(1, 3, 6, 7)	(1, 2, 5, 7)
(1, 4, 5, 6)	(0, 3, 5, 6)
(1, 4, 5, 6)	(1, 2, 3, 6)
(1, 4, 5, 6)	(1, 2, 4, 7)
(1, 4, 5, 7)	(0, 1, 2, 3)
(1, 4, 5, 7)	(0, 1, 4, 5)
(1, 4, 5, 7)	(0, 2, 4, 6)
(1, 4, 5, 7)	(0, 3, 5, 6)
(1, 4, 5, 7)	(1, 2, 4, 7)
(1, 4, 5, 7)	(1, 3, 5, 7)
(1, 4, 5, 7)	(2, 3, 6, 7)

Rows to be removed	**Columns to be removed**
(1, 4, 5, 7)	(4, 5, 6, 7)
(1, 4, 6, 7)	(0, 1, 2, 3)
(1, 4, 6, 7)	(0, 1, 6, 7)
(1, 4, 6, 7)	(0, 2, 5, 7)
(1, 4, 6, 7)	(0, 3, 5, 6)
(1, 4, 6, 7)	(1, 2, 4, 7)
(1, 4, 6, 7)	(1, 3, 4, 6)
(1, 4, 6, 7)	(2, 3, 4, 5)
(1, 4, 6, 7)	(4, 5, 6, 7)
(1, 5, 6, 7)	(0, 3, 5, 6)
(1, 5, 6, 7)	(1, 2, 4, 7)
(1, 5, 6, 7)	(1, 5, 6, 7)
(2, 3, 4, 5)	(0, 1, 2, 7)
(2, 3, 4, 5)	(0, 1, 3, 6)
(2, 3, 4, 5)	(0, 1, 4, 5)
(2, 3, 4, 5)	(0, 2, 3, 5)
(2, 3, 4, 5)	(0, 5, 6, 7)
(2, 3, 4, 5)	(1, 2, 3, 4)
(2, 3, 4, 5)	(1, 4, 6, 7)
(2, 3, 4, 5)	(2, 3, 6, 7)
(2, 3, 4, 5)	(2, 4, 5, 7)
(2, 3, 4, 5)	(3, 4, 5, 6)
(2, 3, 4, 6)	(0, 2, 4, 7)
(2, 3, 4, 6)	(0, 2, 5, 6)
(2, 3, 4, 7)	(0, 1, 2, 3)
(2, 3, 4, 7)	(4, 5, 6, 7)
(2, 3, 5, 6)	(0, 1, 2, 3)
(2, 3, 5, 6)	(4, 5, 6, 7)
(2, 3, 5, 7)	(1, 3, 4, 7)
(2, 3, 5, 7)	(1, 3, 5, 6)
(2, 3, 6, 7)	(0, 1, 2, 4)
(2, 3, 6, 7)	(0, 1, 3, 5)
(2, 3, 6, 7)	(0, 1, 6, 7)
(2, 3, 6, 7)	(0, 2, 3, 6)
(2, 3, 6, 7)	(0, 4, 5, 6)
(2, 3, 6, 7)	(1, 2, 3, 7)
(2, 3, 6, 7)	(1, 4, 5, 7)
(2, 3, 6, 7)	(2, 3, 4, 5)
(2, 3, 6, 7)	(2, 4, 6, 7)
(2, 3, 6, 7)	(3, 5, 6, 7)
(2, 4, 5, 6)	(0, 3, 5, 6)
(2, 4, 5, 6)	(1, 2, 4, 7)
(2, 4, 5, 6)	(2, 4, 5, 6)
(2, 4, 5, 7)	(0, 1, 2, 3)
(2, 4, 5, 7)	(0, 1, 6, 7)
(2, 4, 5, 7)	(0, 2, 5, 7)
(2, 4, 5, 7)	(0, 3, 5, 6)
(2, 4, 5, 7)	(1, 2, 4, 7)
(2, 4, 5, 7)	(1, 3, 4, 6)
(2, 4, 5, 7)	(2, 3, 4, 5)
(2, 4, 5, 7)	(4, 5, 6, 7)
(2, 4, 6, 7)	(0, 1, 2, 3)
(2, 4, 6, 7)	(0, 1, 4, 5)
(2, 4, 6, 7)	(0, 2, 4, 6)
(2, 4, 6, 7)	(0, 3, 5, 6)
(2, 4, 6, 7)	(1, 2, 4, 7)
(2, 4, 6, 7)	(1, 3, 5, 7)
(2, 4, 6, 7)	(2, 3, 6, 7)
(2, 4, 6, 7)	(4, 5, 6, 7)
(2, 5, 6, 7)	(0, 1, 2, 5)
(2, 5, 6, 7)	(0, 3, 5, 6)
(2, 5, 6, 7)	(1, 2, 4, 7)
(3, 4, 5, 6)	(0, 1, 2, 3)
(3, 4, 5, 6)	(0, 1, 6, 7)
(3, 4, 5, 6)	(0, 2, 5, 7)
(3, 4, 5, 6)	(0, 3, 5, 6)
(3, 4, 5, 6)	(1, 2, 4, 7)
(3, 4, 5, 6)	(1, 3, 4, 6)
(3, 4, 5, 6)	(2, 3, 4, 5)

Rows to be removed	**Columns to be removed**
(3, 4, 5, 6)	(4, 5, 6, 7)
(3, 4, 5, 7)	(0, 3, 5, 6)
(3, 4, 5, 7)	(1, 2, 4, 7)
(3, 4, 5, 7)	(3, 4, 5, 7)
(3, 4, 6, 7)	(0, 1, 3, 4)
(3, 4, 6, 7)	(0, 3, 5, 6)
(3, 4, 6, 7)	(1, 2, 4, 7)
(3, 5, 6, 7)	(0, 1, 2, 3)
(3, 5, 6, 7)	(0, 1, 4, 5)
(3, 5, 6, 7)	(0, 2, 4, 6)
(3, 5, 6, 7)	(0, 3, 5, 6)
(3, 5, 6, 7)	(1, 2, 4, 7)
(3, 5, 6, 7)	(1, 3, 5, 7)
(3, 5, 6, 7)	(2, 3, 6, 7)
(3, 5, 6, 7)	(4, 5, 6, 7)
(4, 5, 6, 7)	(0, 1, 2, 4)
(4, 5, 6, 7)	(0, 1, 2, 7)
(4, 5, 6, 7)	(0, 1, 3, 5)
(4, 5, 6, 7)	(0, 1, 3, 6)
(4, 5, 6, 7)	(0, 1, 4, 7)
(4, 5, 6, 7)	(0, 1, 5, 6)
(4, 5, 6, 7)	(0, 2, 3, 5)
(4, 5, 6, 7)	(0, 2, 3, 6)
(4, 5, 6, 7)	(0, 2, 4, 7)
(4, 5, 6, 7)	(0, 2, 5, 6)
(4, 5, 6, 7)	(0, 3, 4, 5)
(4, 5, 6, 7)	(0, 3, 4, 6)
(4, 5, 6, 7)	(0, 3, 5, 6)
(4, 5, 6, 7)	(0, 3, 5, 7)
(4, 5, 6, 7)	(0, 3, 6, 7)
(4, 5, 6, 7)	(0, 4, 5, 6)
(4, 5, 6, 7)	(0, 5, 6, 7)
(4, 5, 6, 7)	(1, 2, 3, 4)
(4, 5, 6, 7)	(1, 2, 3, 7)
(4, 5, 6, 7)	(1, 2, 4, 5)
(4, 5, 6, 7)	(1, 2, 4, 6)
(4, 5, 6, 7)	(1, 2, 4, 7)
(4, 5, 6, 7)	(1, 2, 5, 7)
(4, 5, 6, 7)	(1, 2, 6, 7)
(4, 5, 6, 7)	(1, 3, 4, 7)
(4, 5, 6, 7)	(1, 3, 5, 6)
(4, 5, 6, 7)	(1, 4, 5, 7)
(4, 5, 6, 7)	(1, 4, 6, 7)
(4, 5, 6, 7)	(2, 3, 4, 7)
(4, 5, 6, 7)	(2, 3, 5, 6)
(4, 5, 6, 7)	(2, 4, 5, 7)
(4, 5, 6, 7)	(2, 4, 6, 7)
(4, 5, 6, 7)	(3, 4, 5, 6)
(4, 5, 6, 7)	(3, 5, 6, 7)
(0, 1, 2, 3, 4)	(0, 1, 2, 4, 7)
(0, 1, 2, 3, 4)	(0, 1, 3, 5, 6)
(0, 1, 2, 3, 4)	(0, 2, 3, 5, 6)
(0, 1, 2, 3, 4)	(0, 3, 4, 5, 6)
(0, 1, 2, 3, 4)	(0, 3, 5, 6, 7)
(0, 1, 2, 3, 4)	(1, 2, 3, 4, 7)
(0, 1, 2, 3, 4)	(1, 2, 4, 5, 7)
(0, 1, 2, 3, 4)	(1, 2, 4, 6, 7)
(0, 1, 2, 3, 5)	(0, 1, 2, 4, 7)
(0, 1, 2, 3, 5)	(0, 1, 3, 5, 6)
(0, 1, 2, 3, 5)	(0, 2, 3, 5, 6)
(0, 1, 2, 3, 5)	(0, 3, 4, 5, 6)
(0, 1, 2, 3, 5)	(0, 3, 5, 6, 7)
(0, 1, 2, 3, 5)	(1, 2, 3, 4, 7)
(0, 1, 2, 3, 5)	(1, 2, 4, 5, 7)
(0, 1, 2, 3, 5)	(1, 2, 4, 6, 7)
(0, 1, 2, 3, 6)	(0, 1, 2, 4, 7)
(0, 1, 2, 3, 6)	(0, 1, 3, 5, 6)
(0, 1, 2, 3, 6)	(0, 2, 3, 5, 6)
(0, 1, 2, 3, 6)	(0, 3, 4, 5, 6)

Rows to be removed	Columns to be removed
(0, 1, 2, 3, 6)	(0, 3, 5, 6, 7)
(0, 1, 2, 3, 6)	(1, 2, 3, 4, 7)
(0, 1, 2, 3, 6)	(1, 2, 4, 5, 7)
(0, 1, 2, 3, 6)	(1, 2, 4, 6, 7)
(0, 1, 2, 3, 7)	(0, 1, 2, 4, 7)
(0, 1, 2, 3, 7)	(0, 1, 3, 5, 6)
(0, 1, 2, 3, 7)	(0, 2, 3, 5, 6)
(0, 1, 2, 3, 7)	(0, 3, 4, 5, 6)
(0, 1, 2, 3, 7)	(0, 3, 5, 6, 7)
(0, 1, 2, 3, 7)	(1, 2, 3, 4, 7)
(0, 1, 2, 3, 7)	(1, 2, 4, 5, 7)
(0, 1, 2, 3, 7)	(1, 2, 4, 6, 7)
(0, 1, 2, 4, 5)	(0, 1, 2, 5, 7)
(0, 1, 2, 4, 5)	(1, 3, 4, 5, 6)
(0, 1, 2, 4, 6)	(0, 1, 2, 5, 7)
(0, 1, 2, 4, 6)	(0, 3, 4, 6, 7)
(0, 1, 2, 4, 6)	(1, 3, 4, 5, 6)
(0, 1, 2, 4, 6)	(2, 3, 4, 5, 7)
(0, 1, 2, 4, 7)	(0, 1, 2, 3, 4)
(0, 1, 2, 4, 7)	(0, 1, 2, 3, 5)
(0, 1, 2, 4, 7)	(0, 1, 2, 3, 6)
(0, 1, 2, 4, 7)	(0, 1, 2, 3, 7)
(0, 1, 2, 4, 7)	(0, 4, 5, 6, 7)
(0, 1, 2, 4, 7)	(1, 4, 5, 6, 7)
(0, 1, 2, 4, 7)	(2, 4, 5, 6, 7)
(0, 1, 2, 4, 7)	(3, 4, 5, 6, 7)
(0, 1, 2, 5, 6)	(0, 1, 2, 5, 7)
(0, 1, 2, 5, 6)	(0, 2, 4, 6, 7)
(0, 1, 2, 5, 6)	(1, 3, 4, 5, 6)
(0, 1, 2, 5, 7)	(0, 1, 2, 4, 5)
(0, 1, 2, 5, 7)	(0, 1, 2, 4, 6)
(0, 1, 2, 5, 7)	(0, 1, 2, 5, 6)
(0, 1, 2, 5, 7)	(0, 1, 4, 5, 6)
(0, 1, 2, 5, 7)	(0, 2, 4, 5, 6)
(0, 1, 2, 5, 7)	(1, 2, 4, 5, 6)
(0, 1, 2, 6, 7)	(1, 3, 4, 5, 7)
(0, 1, 3, 4, 5)	(0, 1, 3, 4, 6)
(0, 1, 3, 4, 5)	(0, 2, 4, 5, 7)
(0, 1, 3, 4, 6)	(0, 1, 3, 4, 5)
(0, 1, 3, 4, 6)	(0, 1, 3, 4, 7)
(0, 1, 3, 4, 6)	(0, 1, 3, 5, 7)
(0, 1, 3, 4, 6)	(0, 1, 4, 5, 7)
(0, 1, 3, 4, 6)	(0, 3, 4, 5, 7)
(0, 1, 3, 4, 6)	(1, 3, 4, 5, 7)
(0, 1, 3, 4, 7)	(0, 1, 3, 4, 6)
(0, 1, 3, 4, 7)	(0, 2, 4, 5, 7)
(0, 1, 3, 4, 7)	(1, 3, 5, 6, 7)
(0, 1, 3, 5, 6)	(0, 1, 2, 3, 4)
(0, 1, 3, 5, 6)	(0, 1, 2, 3, 5)
(0, 1, 3, 5, 6)	(0, 1, 2, 3, 6)
(0, 1, 3, 5, 6)	(0, 1, 2, 3, 7)
(0, 1, 3, 5, 6)	(0, 4, 5, 6, 7)
(0, 1, 3, 5, 6)	(1, 4, 5, 6, 7)
(0, 1, 3, 5, 6)	(2, 4, 5, 6, 7)
(0, 1, 3, 5, 6)	(3, 4, 5, 6, 7)
(0, 1, 3, 5, 7)	(0, 1, 3, 4, 6)
(0, 1, 3, 5, 7)	(0, 2, 4, 5, 7)
(0, 1, 3, 5, 7)	(1, 2, 5, 6, 7)
(0, 1, 3, 5, 7)	(2, 3, 4, 5, 6)
(0, 1, 3, 6, 7)	(0, 2, 4, 5, 6)
(0, 1, 4, 5, 6)	(0, 1, 2, 5, 7)
(0, 1, 4, 5, 6)	(1, 3, 4, 5, 6)
(0, 1, 4, 5, 7)	(0, 1, 3, 4, 6)
(0, 1, 4, 5, 7)	(0, 2, 4, 5, 7)
(0, 1, 4, 6, 7)	(1, 2, 3, 5, 7)
(0, 1, 5, 6, 7)	(0, 2, 3, 4, 6)
(0, 2, 3, 4, 5)	(1, 3, 5, 6, 7)
(0, 2, 3, 4, 6)	(0, 1, 5, 6, 7)
(0, 2, 3, 4, 6)	(0, 2, 3, 5, 7)

Rows to be removed	**Columns to be removed**
(0, 2, 3, 4, 6)	(1, 2, 4, 5, 6)
(0, 2, 3, 4, 6)	(1, 3, 4, 6, 7)
(0, 2, 3, 4, 7)	(0, 2, 3, 5, 7)
(0, 2, 3, 4, 7)	(0, 2, 4, 5, 6)
(0, 2, 3, 4, 7)	(1, 3, 4, 6, 7)
(0, 2, 3, 5, 6)	(0, 1, 2, 3, 4)
(0, 2, 3, 5, 6)	(0, 1, 2, 3, 5)
(0, 2, 3, 5, 6)	(0, 1, 2, 3, 6)
(0, 2, 3, 5, 6)	(0, 1, 2, 3, 7)
(0, 2, 3, 5, 6)	(0, 4, 5, 6, 7)
(0, 2, 3, 5, 6)	(1, 4, 5, 6, 7)
(0, 2, 3, 5, 6)	(2, 4, 5, 6, 7)
(0, 2, 3, 5, 6)	(3, 4, 5, 6, 7)
(0, 2, 3, 5, 7)	(0, 2, 3, 4, 6)
(0, 2, 3, 5, 7)	(0, 2, 3, 4, 7)
(0, 2, 3, 5, 7)	(0, 2, 3, 6, 7)
(0, 2, 3, 5, 7)	(0, 2, 4, 6, 7)
(0, 2, 3, 5, 7)	(0, 3, 4, 6, 7)
(0, 2, 3, 5, 7)	(2, 3, 4, 6, 7)
(0, 2, 3, 6, 7)	(0, 2, 3, 5, 7)
(0, 2, 3, 6, 7)	(1, 3, 4, 6, 7)
(0, 2, 4, 5, 6)	(0, 1, 2, 5, 7)
(0, 2, 4, 5, 6)	(0, 1, 3, 6, 7)
(0, 2, 4, 5, 6)	(0, 2, 3, 4, 7)
(0, 2, 4, 5, 6)	(1, 3, 4, 5, 6)
(0, 2, 4, 5, 7)	(0, 1, 3, 4, 5)
(0, 2, 4, 5, 7)	(0, 1, 3, 4, 7)
(0, 2, 4, 5, 7)	(0, 1, 3, 5, 7)
(0, 2, 4, 5, 7)	(0, 1, 4, 5, 7)
(0, 2, 4, 5, 7)	(0, 3, 4, 5, 7)
(0, 2, 4, 5, 7)	(1, 3, 4, 5, 7)
(0, 2, 4, 6, 7)	(0, 1, 2, 5, 6)
(0, 2, 4, 6, 7)	(0, 2, 3, 5, 7)
(0, 2, 4, 6, 7)	(1, 2, 3, 4, 5)
(0, 2, 4, 6, 7)	(1, 3, 4, 6, 7)
(0, 2, 5, 6, 7)	(1, 2, 3, 5, 6)
(0, 2, 5, 6, 7)	(1, 2, 3, 5, 7)
(0, 2, 5, 6, 7)	(1, 2, 3, 6, 7)
(0, 2, 5, 6, 7)	(1, 2, 5, 6, 7)
(0, 2, 5, 6, 7)	(1, 3, 5, 6, 7)
(0, 2, 5, 6, 7)	(2, 3, 5, 6, 7)
(0, 3, 4, 5, 6)	(0, 1, 2, 3, 4)
(0, 3, 4, 5, 6)	(0, 1, 2, 3, 5)
(0, 3, 4, 5, 6)	(0, 1, 2, 3, 6)
(0, 3, 4, 5, 6)	(0, 1, 2, 3, 7)
(0, 3, 4, 5, 6)	(0, 4, 5, 6, 7)
(0, 3, 4, 5, 6)	(1, 4, 5, 6, 7)
(0, 3, 4, 5, 6)	(2, 4, 5, 6, 7)
(0, 3, 4, 5, 6)	(3, 4, 5, 6, 7)
(0, 3, 4, 5, 7)	(0, 1, 3, 4, 6)
(0, 3, 4, 5, 7)	(0, 2, 4, 5, 7)
(0, 3, 4, 5, 7)	(1, 2, 3, 5, 7)
(0, 3, 4, 6, 7)	(0, 1, 2, 4, 6)
(0, 3, 4, 6, 7)	(0, 2, 3, 5, 7)
(0, 3, 4, 6, 7)	(1, 3, 4, 6, 7)
(0, 3, 5, 6, 7)	(0, 1, 2, 3, 4)
(0, 3, 5, 6, 7)	(0, 1, 2, 3, 5)
(0, 3, 5, 6, 7)	(0, 1, 2, 3, 6)
(0, 3, 5, 6, 7)	(0, 1, 2, 3, 7)
(0, 3, 5, 6, 7)	(0, 4, 5, 6, 7)
(0, 3, 5, 6, 7)	(1, 4, 5, 6, 7)
(0, 3, 5, 6, 7)	(2, 4, 5, 6, 7)
(0, 3, 5, 6, 7)	(3, 4, 5, 6, 7)
(0, 4, 5, 6, 7)	(0, 1, 2, 4, 7)
(0, 4, 5, 6, 7)	(0, 1, 3, 5, 6)
(0, 4, 5, 6, 7)	(0, 2, 3, 5, 6)
(0, 4, 5, 6, 7)	(0, 3, 4, 5, 6)
(0, 4, 5, 6, 7)	(0, 3, 5, 6, 7)
(0, 4, 5, 6, 7)	(1, 2, 3, 4, 7)

Rows to be removed	Columns to be removed
(0, 4, 5, 6, 7)	(1, 2, 4, 5, 7)
(0, 4, 5, 6, 7)	(1, 2, 4, 6, 7)
(1, 2, 3, 4, 5)	(0, 2, 4, 6, 7)
(1, 2, 3, 4, 6)	(1, 2, 3, 5, 6)
(1, 2, 3, 4, 6)	(1, 2, 3, 5, 7)
(1, 2, 3, 4, 6)	(1, 2, 3, 6, 7)
(1, 2, 3, 4, 6)	(1, 2, 5, 6, 7)
(1, 2, 3, 4, 6)	(1, 3, 5, 6, 7)
(1, 2, 3, 4, 6)	(2, 3, 5, 6, 7)
(1, 2, 3, 4, 7)	(0, 1, 2, 3, 4)
(1, 2, 3, 4, 7)	(0, 1, 2, 3, 5)
(1, 2, 3, 4, 7)	(0, 1, 2, 3, 6)
(1, 2, 3, 4, 7)	(0, 1, 2, 3, 7)
(1, 2, 3, 4, 7)	(0, 4, 5, 6, 7)
(1, 2, 3, 4, 7)	(1, 4, 5, 6, 7)
(1, 2, 3, 4, 7)	(2, 4, 5, 6, 7)
(1, 2, 3, 4, 7)	(3, 4, 5, 6, 7)
(1, 2, 3, 5, 6)	(0, 2, 5, 6, 7)
(1, 2, 3, 5, 6)	(1, 2, 3, 4, 6)
(1, 2, 3, 5, 6)	(1, 3, 4, 5, 7)
(1, 2, 3, 5, 7)	(0, 1, 4, 6, 7)
(1, 2, 3, 5, 7)	(0, 2, 5, 6, 7)
(1, 2, 3, 5, 7)	(0, 3, 4, 5, 7)
(1, 2, 3, 5, 7)	(1, 2, 3, 4, 6)
(1, 2, 3, 6, 7)	(0, 2, 5, 6, 7)
(1, 2, 3, 6, 7)	(1, 2, 3, 4, 6)
(1, 2, 4, 5, 6)	(0, 1, 2, 5, 7)
(1, 2, 4, 5, 6)	(0, 2, 3, 4, 6)
(1, 2, 4, 5, 6)	(1, 3, 4, 5, 6)
(1, 2, 4, 5, 7)	(0, 1, 2, 3, 4)
(1, 2, 4, 5, 7)	(0, 1, 2, 3, 5)
(1, 2, 4, 5, 7)	(0, 1, 2, 3, 6)
(1, 2, 4, 5, 7)	(0, 1, 2, 3, 7)
(1, 2, 4, 5, 7)	(0, 4, 5, 6, 7)
(1, 2, 4, 5, 7)	(1, 4, 5, 6, 7)
(1, 2, 4, 5, 7)	(2, 4, 5, 6, 7)
(1, 2, 4, 5, 7)	(3, 4, 5, 6, 7)
(1, 2, 4, 6, 7)	(0, 1, 2, 3, 4)
(1, 2, 4, 6, 7)	(0, 1, 2, 3, 5)
(1, 2, 4, 6, 7)	(0, 1, 2, 3, 6)
(1, 2, 4, 6, 7)	(0, 1, 2, 3, 7)
(1, 2, 4, 6, 7)	(0, 4, 5, 6, 7)
(1, 2, 4, 6, 7)	(1, 4, 5, 6, 7)
(1, 2, 4, 6, 7)	(2, 4, 5, 6, 7)
(1, 2, 4, 6, 7)	(3, 4, 5, 6, 7)
(1, 2, 5, 6, 7)	(0, 1, 3, 5, 7)
(1, 2, 5, 6, 7)	(0, 2, 5, 6, 7)
(1, 2, 5, 6, 7)	(1, 2, 3, 4, 6)
(1, 3, 4, 5, 6)	(0, 1, 2, 4, 5)
(1, 3, 4, 5, 6)	(0, 1, 2, 4, 6)
(1, 3, 4, 5, 6)	(0, 1, 2, 5, 6)
(1, 3, 4, 5, 6)	(0, 1, 4, 5, 6)
(1, 3, 4, 5, 6)	(0, 2, 4, 5, 6)
(1, 3, 4, 5, 6)	(1, 2, 4, 5, 6)
(1, 3, 4, 5, 7)	(0, 1, 2, 6, 7)
(1, 3, 4, 5, 7)	(0, 1, 3, 4, 6)
(1, 3, 4, 5, 7)	(0, 2, 4, 5, 7)
(1, 3, 4, 5, 7)	(1, 2, 3, 5, 6)
(1, 3, 4, 6, 7)	(0, 2, 3, 4, 6)
(1, 3, 4, 6, 7)	(0, 2, 3, 4, 7)
(1, 3, 4, 6, 7)	(0, 2, 3, 6, 7)
(1, 3, 4, 6, 7)	(0, 2, 4, 6, 7)
(1, 3, 4, 6, 7)	(0, 3, 4, 6, 7)
(1, 3, 4, 6, 7)	(2, 3, 4, 6, 7)
(1, 3, 5, 6, 7)	(0, 1, 3, 4, 7)
(1, 3, 5, 6, 7)	(0, 2, 3, 4, 5)
(1, 3, 5, 6, 7)	(0, 2, 5, 6, 7)
(1, 3, 5, 6, 7)	(1, 2, 3, 4, 6)
(1, 4, 5, 6, 7)	(0, 1, 2, 4, 7)

Rows to be removed	**Columns to be removed**
(1, 4, 5, 6, 7)	(0, 1, 3, 5, 6)
(1, 4, 5, 6, 7)	(0, 2, 3, 5, 6)
(1, 4, 5, 6, 7)	(0, 3, 4, 5, 6)
(1, 4, 5, 6, 7)	(0, 3, 5, 6, 7)
(1, 4, 5, 6, 7)	(1, 2, 3, 4, 7)
(1, 4, 5, 6, 7)	(1, 2, 4, 5, 7)
(1, 4, 5, 6, 7)	(1, 2, 4, 6, 7)
(2, 3, 4, 5, 6)	(0, 1, 3, 5, 7)
(2, 3, 4, 5, 7)	(0, 1, 2, 4, 6)
(2, 3, 4, 6, 7)	(0, 2, 3, 5, 7)
(2, 3, 4, 6, 7)	(1, 3, 4, 6, 7)
(2, 3, 5, 6, 7)	(0, 2, 5, 6, 7)
(2, 3, 5, 6, 7)	(1, 2, 3, 4, 6)
(2, 4, 5, 6, 7)	(0, 1, 2, 4, 7)
(2, 4, 5, 6, 7)	(0, 1, 3, 5, 6)
(2, 4, 5, 6, 7)	(0, 2, 3, 5, 6)
(2, 4, 5, 6, 7)	(0, 3, 4, 5, 6)
(2, 4, 5, 6, 7)	(0, 3, 5, 6, 7)
(2, 4, 5, 6, 7)	(1, 2, 3, 4, 7)
(2, 4, 5, 6, 7)	(1, 2, 4, 5, 7)
(2, 4, 5, 6, 7)	(1, 2, 4, 6, 7)
(3, 4, 5, 6, 7)	(0, 1, 2, 4, 7)
(3, 4, 5, 6, 7)	(0, 1, 3, 5, 6)
(3, 4, 5, 6, 7)	(0, 2, 3, 5, 6)
(3, 4, 5, 6, 7)	(0, 3, 4, 5, 6)
(3, 4, 5, 6, 7)	(0, 3, 5, 6, 7)
(3, 4, 5, 6, 7)	(1, 2, 3, 4, 7)
(3, 4, 5, 6, 7)	(1, 2, 4, 5, 7)
(3, 4, 5, 6, 7)	(1, 2, 4, 6, 7)
(0, 1, 2, 3, 4, 6)	(0, 1, 3, 4, 5, 7)
(0, 1, 2, 3, 4, 6)	(1, 2, 3, 5, 6, 7)
(0, 1, 2, 3, 5, 7)	(0, 1, 2, 4, 5, 6)
(0, 1, 2, 3, 5, 7)	(0, 2, 3, 4, 6, 7)
(0, 1, 2, 4, 5, 6)	(0, 1, 2, 3, 5, 7)
(0, 1, 2, 4, 5, 6)	(0, 1, 2, 4, 5, 7)
(0, 1, 2, 4, 5, 6)	(0, 1, 2, 5, 6, 7)
(0, 1, 2, 4, 5, 6)	(0, 1, 3, 4, 5, 6)
(0, 1, 2, 4, 5, 6)	(1, 2, 3, 4, 5, 6)
(0, 1, 2, 4, 5, 6)	(1, 3, 4, 5, 6, 7)
(0, 1, 2, 4, 5, 7)	(0, 1, 2, 4, 5, 6)
(0, 1, 2, 4, 5, 7)	(0, 1, 3, 4, 5, 7)
(0, 1, 2, 5, 6, 7)	(0, 1, 2, 4, 5, 6)
(0, 1, 2, 5, 6, 7)	(1, 2, 3, 5, 6, 7)
(0, 1, 3, 4, 5, 6)	(0, 1, 2, 4, 5, 6)
(0, 1, 3, 4, 5, 6)	(0, 1, 3, 4, 5, 7)
(0, 1, 3, 4, 5, 7)	(0, 1, 2, 3, 4, 6)
(0, 1, 3, 4, 5, 7)	(0, 1, 2, 4, 5, 7)
(0, 1, 3, 4, 5, 7)	(0, 1, 3, 4, 5, 6)
(0, 1, 3, 4, 5, 7)	(0, 1, 3, 4, 6, 7)
(0, 1, 3, 4, 5, 7)	(0, 2, 3, 4, 5, 7)
(0, 1, 3, 4, 5, 7)	(0, 2, 4, 5, 6, 7)
(0, 1, 3, 4, 6, 7)	(0, 1, 3, 4, 5, 7)
(0, 1, 3, 4, 6, 7)	(0, 2, 3, 4, 6, 7)
(0, 2, 3, 4, 5, 7)	(0, 1, 3, 4, 5, 7)
(0, 2, 3, 4, 5, 7)	(0, 2, 3, 4, 6, 7)
(0, 2, 3, 4, 6, 7)	(0, 1, 2, 3, 5, 7)
(0, 2, 3, 4, 6, 7)	(0, 1, 3, 4, 6, 7)
(0, 2, 3, 4, 6, 7)	(0, 2, 3, 4, 5, 7)
(0, 2, 3, 4, 6, 7)	(0, 2, 3, 5, 6, 7)
(0, 2, 3, 4, 6, 7)	(1, 2, 3, 4, 6, 7)
(0, 2, 3, 4, 6, 7)	(1, 3, 4, 5, 6, 7)
(0, 2, 3, 5, 6, 7)	(0, 2, 3, 4, 6, 7)
(0, 2, 3, 5, 6, 7)	(1, 2, 3, 5, 6, 7)
(0, 2, 4, 5, 6, 7)	(0, 1, 3, 4, 5, 7)
(0, 2, 4, 5, 6, 7)	(1, 2, 3, 5, 6, 7)
(1, 2, 3, 4, 5, 6)	(0, 1, 2, 4, 5, 6)
(1, 2, 3, 4, 5, 6)	(1, 2, 3, 5, 6, 7)
(1, 2, 3, 4, 6, 7)	(0, 2, 3, 4, 6, 7)
(1, 2, 3, 4, 6, 7)	(1, 2, 3, 5, 6, 7)

Rows to be removed	Columns to be removed
(1, 2, 3, 5, 6, 7)	(0, 1, 2, 3, 4, 6)
(1, 2, 3, 5, 6, 7)	(0, 1, 2, 5, 6, 7)
(1, 2, 3, 5, 6, 7)	(0, 2, 3, 5, 6, 7)
(1, 2, 3, 5, 6, 7)	(0, 2, 4, 5, 6, 7)
(1, 2, 3, 5, 6, 7)	(1, 2, 3, 4, 5, 6)
(1, 2, 3, 5, 6, 7)	(1, 2, 3, 4, 6, 7)
(1, 3, 4, 5, 6, 7)	(0, 1, 2, 4, 5, 6)
(1, 3, 4, 5, 6, 7)	(0, 2, 3, 4, 6, 7)

Table 7.17: Whirlwind M_1 (7.40) singular submatrices

7.19 Appendix 7E: Code Snippets

Listing 1 Python function to compute the **xtime** and **xor** cost of a single matrix coefficient

```
def poly_xtime_cost(poly, ORDER):
    if poly == 0:
        return 0
    degree = ORDER
    degree_mask = 2**ORDER

    while (poly & degree_mask) == 0:
        degree_mask = degree_mask >> 1
        degree -= 1
    return degree

def poly_xor_cost(poly, ORDER):
    mask = 1
    set_bits = 0
    current_bit = 0
    while current_bit < ORDER:
        if (poly & mask) != 0:
            set_bits += 1
        mask = mask << 1
        current_bit += 1
    return set_bits - 1
```

Listing 2 Python function to compute the **xtime** and **xor** cost of a whole matrix

```
def matrix_xtime_cost(mat, ORDER):
    total_cost = 0
    for row in range(len(mat)):
        row_cost = 0
        for col in range(len(mat[row])):
            row_cost += poly_xtime_cost(mat[row][col], ORDER)
        total_cost += row_cost
    return total_cost

def matrix_xor_cost(mat, ORDER):
    total_cost = 0
    for row in range(len(mat)):
        row_cost = len(mat) - 1
        for col in range(len(mat[row])):
            row_cost += poly_xor_cost(mat[row][col], ORDER)
        total_cost += row_cost
    return total_cost
```

Chapter 8

Genetic Algorithms for MDS Matrix Search: Preliminary Attempts

In this chapter we show how to model MDS matrix search for Genetic Algorithms application and outline our practical experiments. We assume the reader is familiar with the MDS matrix-related concepts of Chapter 7, specially the cost metrics outlined in Section 7.6, and the catalogue we built on Table 7.4.

8.1 Acronyms

- GA: Genetic Algorithm(s)

8.2 Genetic Algorithms Concepts

Genetic Algorithms are inspired by the evolutionary process of species. In this process, there is an initial population of individuals, which reproduce in pairs. In the reproduction process, the genetic material of both parents is mixed (*crossover* step), and a *mutation* may or may not occur, introducing new features to the genetic material of the child. After several parent pairs have successfully reproduced, the population is now composed of the parents and the children. Through time, the individuals with most aptitude to the environment survive whilst the ones with least aptitude have less chances, and, after several iterations of this process, the species' features change. Furthermore, good features, i.e features which help survival in that environment, will be passed from parents to children, leading to a species that successfully adapts and lives in that particular environment.

Genetic Algorithms seek to simulate this process, introducing randomness and trying to find optimal solutions to problems in Computer Science — a population is composed of several candidate solutions, each individual (or also referred to as chromosome) being one candidate solution. The aptitude (*fitness*) of an individual is a measure of how good that candidate solution is to the problem. The Genetic Algorithm simulates the evolutionary process until optimal solutions are found or the population is unable to improve.

All the concepts are highlighted below.

Fitness Function. The fitness function tells how apt an individual is to survive on the environment, i.e how good a candidate solution is for our problem. For example, assuming our problem is to find a pair of numbers (a, b) with maximum sum. An individual could be $(1, 2)$ and another individual could be $(3, 4)$. We could define $f(a, b) = a+b$ as our fitness function, and we would have $f(1, 2) = 3$ and $f(3, 4) = 7$, indicating that $(3, 4)$ is better as a candidate solution to our problem than $(1, 2)$, i.e $(3, 4)$ has more aptitude than $(1, 2)$.

Chromosome/Individual and Population. A candidate solution is called individual or chromosome. In our maximum sum problem, the chromosome can be the pair of numbers itself. The population is a set of several different individuals, for example, several different pairs of numbers, each having their fitness.

Selection for reproduction. This term refers to how the Genetic Algorithm will choose which parents of the current population will reproduce and generate offspring. There are several ways of selecting parents to reproduce, for instance, choosing randomly, choosing only the most fit, among others.

Pairing. Pairing refers to how the pairs of parents for reproduction, once they have been selected, are formed. We can pair e.g the most fit individuals together or form random pairs. There are several approaches.

Crossover. In Biology, crossover refers to the mixing of genetic material from parents. In Genetic Algorithms, there are several kinds of crossover to represent this mixing of features between individuals. For instance, for our example problem of finding pairs of numbers with maximum sum, given two parents $x = (a, b)$ and $y = (c, d)$, one way of mixing them would be taking the first element of x and the second of y, resulting in a child (a, d).

Mutation. Without genetic mutations, reproduction would only lead to mixing genetic material from the parents, and new features would not appear in the population. Therefore, a mutation rate is required for actual evolution, with new features, to happen in a population. An example mutation would be, given a chromosome (a, b), changing one of its numbers to a random number, obtaining e.g (a, k).

Replacement. Refers to who will be part of the new population, now that there are new individuals. There are several replacement strategies that could be adopted, for instance, selecting N most fit individuals where N is the desired size for the new population, random selection, complete replacement (where all the parents die and only the children remain), among others.

A Genetic Algorithm performs, at each iteration, selection for reproduction, crossover between parents, mutations (according to the mutation rate), and replacement of the population. Iterations are performed until an optimal solution is found to the problem, or until it is seen that better (most fit) individuals are not being obtained throughout the iterations, i.e it is not possible to find better candidate solutions for the problem. For more details and tutorials on Genetic Algorithms techniques, we refer the reader to references [72] and [15].

8.2.1 Prior Application of GA

There have been prior applications of GA in other matrix related problems, and in other Cryptography and Error Correcting Codes problems, but not for MDS matrix construction to the best of our knowledge so far, thus making our approach novel.

As examples of prior GA applications, work [40] (2012) uses GA to devise an optimized matrix multiplication algorithm, [45] (1999) uses GA to find linear error correcting codes, [71] (2021) is a cipher where encryption and decryption are based in GA concepts and [29] (2016) uses GA to perform cryptanalysis on SDES algorithm.

8.2.2 About our choice of GA for MDS matrix search

Genetic algorithms are well-suited for optimization problems due to their ability to explore large solution spaces, handle complex and non-linear relationships, and find near-optimal solutions in a relatively efficient manner. Here are some reasons why genetic algorithms are a popular choice for such tasks.

1. **Exploration of Solution Space.** Genetic algorithms are excellent at exploring a vast solution space. In the context of searching matrices, genetic algorithms can generate a diverse set of candidate solutions by representing the matrices as individuals in a population. The genetic operators, such as crossover and mutation, allow for the exploration of different combinations of elements within the matrix. This exploration capability enables the algorithm to potentially find promising solutions that may be difficult to discover through other search methods.

2. **Handling Non-linearity.** Many mathematical and optimization problems involve non-linear relationships, where small changes in input values can result in significant changes in output values. Genetic algorithms can handle such non-linear relationships by evaluating and selecting individuals based on their fitness or objective function values. The genetic operators, particularly the crossover operation, facilitate the combination and recombination of genetic material (matrix elements) from different individuals, allowing the algorithm to explore complex relationships in the solution space.

3. **Efficient Parallel Processing.** Genetic algorithms can take advantage of parallel processing capabilities, allowing for efficient exploration of the solution space. This parallelism arises naturally from the population-based nature of genetic algorithms, where multiple individuals can be evaluated

simultaneously. This feature makes genetic algorithms suitable for parallel and distributed computing environments, enabling faster execution and better utilization of computational resources.

4. **Robustness and Global Optimization.** Genetic algorithms are known for their robustness and ability to find near-optimal solutions. They are less prone to getting stuck in local optima compared to other optimization methods. By maintaining diversity in the population through selection, crossover, and mutation, genetic algorithms can escape local optima and continue searching for better solutions. This characteristic makes them well-suited for optimization problems where the objective function may have multiple local optima.

5. **Adaptability and Flexibility.** Genetic algorithms are highly adaptable and can be easily modified to accommodate various problem domains. The encoding scheme used to represent matrices can be tailored to the problem at hand, allowing for efficient representation and manipulation of the solution space. Additionally, various selection strategies, genetic operators, and parameter settings can be customized to enhance the algorithm's performance for specific problems.

While genetic algorithms have several advantages, it is worth noting that they are not universally applicable and may not always be the best choice for every problem. The performance of genetic algorithms can be influenced by factors such as population size, crossover and mutation rates, and the design of the fitness function. Therefore, careful consideration and experimentation are necessary to determine the suitability of genetic algorithms for a specific matrix search or optimization problem.

We chose GA for this work due to the novelty of the approach (no previous usage of GA on MDS matrix search in the literature) and due to the high potential for customization of our optimization process through the fitness function, illustrated by the formula below, where $w_1, ..., w_6$ are customizable weights and $f(A)$ is the fitness of a matrix A.

$$f(A) = w_1 \cdot A.\texttt{xtime} + w_2 \cdot A.\texttt{xor} + w_3 \cdot A^{-1}.\texttt{xtime} + w_4 \cdot A^{-1}.\texttt{xor} + w_5 \cdot A.\texttt{mds} + w_6 \cdot A.\texttt{involutory}$$

A candidate matrix A has six main attributes: **xor** count, **xtime** count, **xtime** count of the inverse A^{-1}, **xor** count of A^{-1}, a flag indicating whether it is MDS, and a flag indicating whether it is involutory. For applicability in e.g hash functions, since there is no decryption, the costs of A^{-1} are irrelevant, and therefore optimizing just A with regards to **xor** and **xtime** counts and MDS property would be sufficient. However, for a block cipher, the decryption cost becomes relevant, and then one can attempt at optimizing all the counts. Furthermore, different weights can be assigned to each attribute, representing different relevances of encryption and decryption costs (e.g both costs being equally relevant, encryption cost being slightly more relevant, or any other weights). Even the relevance of **xtime** versus **xor** counts themselves could be customized if desired, although it is known that **xtime** is more computationally expensive.

8.3 Modeling MDS matrix search with Genetic Algorithms

8.3.1 Chromosome Representation

Our chromosomes are the matrices themselves, in their representation with integer coefficients, together with additional information that will be relevant to evaluate their fitness such as whether they satisfy MDS property and **xor** and **xtime** costs.

8.3.2 Fitness Functions

– `fitness_1`: If the candidate is MDS, the fitness is defined as $2.0/c$, where c is denotes the cost of the candidate matrix. Otherwise, the fitness is $1.0/c$. This fitness function assigns a higher fitness value to MDS candidates, indicating their desirability.

– `fitness_2`: If the candidate is MDS, the fitness is $1.0/c$. Otherwise, the fitness is 0.0. This fitness function assigns a positive fitness value only to MDS candidates, indicating their superiority over non-MDS candidates.

– `fitness_3`: If the candidate's cost is 0 (that can happen if we have e.g a matrix of zeroes), the fitness is -1, indicating we do not wish matrices full of zeroes. If the candidate is MDS, the fitness is $2 + 1.0/c$. Otherwise, the fitness is $1.0/c$. This fitness function assigns a higher fitness value to MDS candidates, with an additional constant term of 2, indicating further their desirability.

8.3.3 Selection Methods

Fitness Proportionate Selection. Fitness proportionate selection assigns selection probabilities to individuals in the population based on their fitness. The probabilities are proportional to the individuals' fitness values relative to the total fitness of the population. In an analogy, this method creates a roulette wheel where each individual occupies a portion of the wheel corresponding to their probability and selection is performed by spinning the wheel multiple times to choose parents. The higher the fitness, the higher the chance of selection.

Tournament Selection. Tournament selection involves randomly selecting a fixed number of individuals (`tournament_size`) from the population to compete in a tournament. In each tournament, the individual with the highest fitness (the winner) is selected as a parent. This process is repeated until the desired number of parents is obtained. Tournament selection allows for selection pressure, as the fitter individuals have a higher chance of being selected as parents.

Rank-based Selection. Rank-based selection assigns selection probabilities to individuals based on their rank in the population, rather than their fitness values. The individuals are sorted based on their fitness, and ranks are assigned accordingly. Selection probabilities are then calculated based on the ranks. Individuals with higher ranks, indicating better fitness, have higher probabilities of being selected as parents. This approach helps maintain diversity in the population by giving a chance for lower-ranking individuals to be selected.

Stochastic Universal Sampling. Stochastic universal sampling is a technique that selects parents by dividing the selection range into equal-sized intervals. A selection pointer is randomly placed within the first interval. Parents are chosen by incrementing the pointer by a fixed step size, which is determined based on the total fitness and the desired number of parents. The individuals whose fitness falls within the selected intervals are chosen as parents. This method ensures that the selection process covers the entire population, preventing biases towards individuals with higher fitness values.

Elitism. Elitism, in the context of selection for reproduction, refers to the preservation of the best individuals from the population. The function selects the top `num_elites` individuals with the highest fitness and designates them as elites. These elites are then directly carried over to the next generation without undergoing any variation or crossover operations. Elitism ensures that the best solutions found so far are preserved and can potentially improve the overall quality of the population in subsequent generations.

8.3.4 Pairing Methods

Random Pairing. In random pairing, selected parents are paired up randomly to form pairs for reproduction. The function randomly selects two distinct parents from the pool of selected parents, removes them from the pool, and adds them as a pair. This process continues until there are no more parents left to form pairs.

Sequential Pairing. Sequential pairing pairs up selected parents in a sequential manner. Each parent is paired with the next parent in the ordered list of selected parents. The last parent is paired with the first parent to ensure all parents are paired. This method guarantees that every parent is paired exactly once.

Randomized Sequential Pairing. Randomized sequential pairing is similar to sequential pairing but with an additional step of shuffling the selected parents randomly before pairing them. This random shuffling introduces variability in the pairing process, which can help to diversify the resulting offspring.

Tournament Selection. In tournament selection, pairs of parents are formed by conducting tournaments among the selected parents. For each pair, a subset of parents (`pairing_tournament_size`) is randomly chosen from the selected parents. The fittest parent in the tournament is selected as the first parent of the pair, and then the second fittest parent is selected as the second parent. The selected parents are removed from the pool of selected parents, and the process is repeated until no more parents can be paired.

Fitness Proportionate Pairing. Fitness proportionate pairing assigns pairing probabilities to the selected parents based on their fitness. The probabilities are proportional to the parents' fitness values relative to the total fitness of the selected parents. Parents are paired by randomly selecting two distinct parents from the pool of selected parents, using their probabilities as weights for the selection. The selected parents are then removed from the pool, and the process continues until there are no more parents left to form pairs. This method gives higher fitness parents a higher chance of being selected for pairing.

8.3.5 Crossover Methods

– `midpoint_crossover1`: Takes two parents, `parent1` and `parent2`, and generates two children, `child1` and `child2`, such that `child1` is composed by the first half of `parent1` concatenated to the second half of `parent2`, and `child2` is composed by the first half of `parent2` concatenated to the second half of `parent1`. In other words, the midpoint of the matrices is calculated, and two children are generated by combining the obtained halves.

– `midpoint_crossover2`: Obtains children by mixing halves as well, but does it for each row instead of the matrix as a whole.

– `alternating_crossover`: Mixes the genetic material by alternating matrix elements, getting one from each parent at a time when constructing the child matrix.

– `random_points_crossover1`: Similar to `midpoint_crossover1`, but instead of using the midpoint, a random point is chosen to split the genetic material.

– `random_points_crossover2`: Similar to `midpoint_crossover2`, but uses a random point to split the genetic material as well.

– `random_points_crossover3`: Computes a random split point for each row instead of keeping it fixed for all rows.

To illustrate, let the following matrices be the parents.

$$\text{parent1}=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix} \text{parent2}=\begin{bmatrix}10 & 11 & 12\\ 13 & 14 & 15\\ 16 & 17 & 18\end{bmatrix}.$$

`midpoint_crossover1` results in

$$\text{child1}=\begin{bmatrix}1 & 2 & 3\\ 4 & 14 & 15\\ 16 & 17 & 18\end{bmatrix} \text{child2}=\begin{bmatrix}10 & 11 & 12\\ 13 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}.$$

`midpoint_crossover2` results in

$$\text{child1}=\begin{bmatrix}1 & 11 & 12\\ 4 & 14 & 15\\ 7 & 17 & 18\end{bmatrix} \text{child2}=\begin{bmatrix}10 & 2 & 3\\ 13 & 5 & 6\\ 16 & 8 & 9\end{bmatrix}.$$

`alternating_crossover` results in

$$\text{child1}=\begin{bmatrix}1 & 11 & 3\\ 13 & 5 & 15\\ 7 & 17 & 9\end{bmatrix} \text{child2}=\begin{bmatrix}10 & 2 & 12\\ 4 & 14 & 6\\ 16 & 8 & 18\end{bmatrix}.$$

8.3.6 Mutation Methods

– `mutation1` selects a random matrix element and changes it to 1.

– `mutation2` generates a random amount q and then proceeds to randomly change q elements to 1.

– `mutation3` selects a random matrix element and changes it to a random value within the finite field limits.

– `mutation4` generates a random amount q and then proceeds to randomly change q elements to random values within the finite field limits.

8.3.7 Replacement Methods

Generational Replacement. This replacement method simply discards the old population and replaces it entirely with the offspring. It does not consider the fitness or any other criteria of the individuals in the old population. The new population consists solely of the offspring generated in the current generation.

Elitism. Elitism is a replacement method that preserves the best individuals from the old population and combines them with the offspring. It aims to maintain the high-quality solutions found so far. The method selects a certain number of elites from the old population based on their fitness (typically the top individuals). The remaining slots in the new population are filled with the offspring, excluding the elites. This ensures that the best solutions survive to the next generation.

Tournament Selection. In tournament selection, a certain number of individuals (`tournament_size`) is randomly selected from the combined population of the old population and offspring. These individuals compete against each other, and the one with the highest fitness (winner) is chosen as a survivor for the new population. This process is repeated until the desired number of survivors (`num_survivors`) is reached. This method allows for selection pressure and introduces randomness in selecting the survivors.

Fitness-based Replacement. Fitness-based replacement involves ranking the individuals in the combined population (old population + offspring) based on their fitness. The individuals are sorted in descending order of fitness, and the top individuals (`num_survivors`) are selected as survivors for the new population. This method favors individuals with higher fitness values, allowing them to pass their genetic material to the next generation while discarding weaker individuals.

These replacement methods are used to manage the population and control the evolution process. Each method has its advantages and trade-offs in terms of exploration, exploitation, and preservation of diversity.

8.4 Experimental Method and Details

8.4.1 Machines

Four machines were utilized in the experiments, referred to as `ifrit`, `termina`, `mac`, and `finger`. Table 8.2 lists the processors of each machine and listings for full specifications. For `ifrit`, `termina` and `finger`, the full specification was obtained through the `lscpu` command since they have Linux OS. For `mac`, `sysctl -a | grep machdep.cpu` was used.

Machine	CPU	Full Specification
`ifrit`	Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz	Listing 8.1
`termina`	Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz	Listing 8.2
`mac`	Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz	Listing 8.3
`finger`	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz	Listing 8.4

Table 8.2: Machine Specifications

8.4.2 Process

We executed four types of experiments, namely:

- Type I. Weighted sum of **xtime** and **xor** as cost (`3:1-cost` metric) + random initialization of the population.
- Type II. Weighted sum of **xtime** and **xor** as cost (`3:1-cost` metric) + initialization with existing solutions i.e existing low cost MDS matrices from our catalogue.
- Type III. Cost defined as **xtime** only (`xt-only-cost` metric) + random initialization of the population.
- Type IV. Cost defined as **xtime** only (`xt-only-cost` metric) + initialization with existing solutions.

With regards to GA specific settings (initial population size, mutation method, crossover method etc), we experimented different combinations according to Table 8.3. For details on the two cost metrics we refer the reader to Section 7.6.

Parameter	**Possible Values**
Field	$GF(2^4)$, $GF(2^8)$
Dimension	2, 3, 4, 5, 6, 7, 8
`num_elites` for replacement (w/a)	10
`num_survivors` for replacement (w/a)	10
`tournament_size` for replacement (w/a)	10
`tournament_size` for pairing (w/a)	10
Initial population size	24, 40
Maximum of iterations	100
Number of parents for selection	12, 20
Mutation probability	0.0, 0.1, 0.2, 0.5, 0.7, 1.0
Fitness function	`fitness_1`, `fitness_2`, `fitness_3`
Selection function	fitness proportionate, tournament, rank-based, stochastic universal sampling,elitism
Pairing function	random, sequential, randomized sequential, tournament, fitness proportionate
Crossover function	`midpoint_crossover1`, `midpoint_crossover2`, `alternating_crossover`, `random_points_crossover1`, `random_points_crossover2`,`random_points_crossover3`
Mutation function	`mutation1`, `mutation2`, `mutation3`, `mutation4`
Replacement function	generational replacement, elitism, tournament, fitness-based

Table 8.3: Values used for GA experiments

Each experiment type explored several combinations of Table 8.3, with a maximum timeout of 2 hours per combination. Each of the experiment types was executed on a different machine so that we could run them simultaneously when desired to save time without one interfering on another (which could happen if they were all executed on the same machine, e.g they could all end up slower due to sharing the CPU resources between processes). Each experiment type ran for ≈ 20 hours before we interrupted the execution to analyse the results.

Results were saved to log files containing the following information for analysis: the respective finite field, the GA settings of Table 8.3, matrix dimension, population of each iteration, offspring of each iteration, list with all MDS matrices worse than the baseline, list with all MDS matrices with equal computational cost to the baseline, list with all MDS matrices with better computational cost than the baseline. Furthermore, since a single matrix can appear in multiple generations of the evolution simulation, we also had sets with the matrices (worst, same, better) to ensure each matrix would be listed and counted a single time. We also, for each matrix, had a flag regarding it being a newly discovered matrix or a matrix that was already present in the catalogue of Chapter 7, and, to ease the identification of MDS matrices better than the baselines amidst all the matrices in the log, the string `"DISCOVERY"` would be printed to the log file in case any such matrix was found.

8.4.3 Baselines

The baselines for our experiments are in Table 8.4. For each matrix dimension, we took the cheapest costs of our existent matrices dataset, i.e the catalogue in Table 7.4. A GA generated matrix is deemed `BETTER`, `WORSE` or `SAME` according to these baselines. `BETTER` refers to cost smaller than the baseline, `SAME` to cost equal and `WORSE` to cost bigger than the baseline. Furthermore, we also flag each matrix as a new discovery or as a matrix that already exists in our dataset throughout the experiments performing lookups on a Python dictionary containing all the existing matrices of our dataset.

8.4.4 Remarks on the experimental process

We do not detail the exact time the experiments ran on each machine, or specific timeout occurrences, because this was a rather exploratory approach with several combinations and experiments occurring at the same time. We were trying to see if genetic algorithms could help us generate MDS matrices, and whether they could optimize their **xtime** and **xor** counts as well given state of the art baselines to try to optimize against, but we did not identify any specific combination of parameters that yields remarkable results. We provide approximate times, explain the process and list possible improvements, since no specific combination was remarkable enough for us to delve deeper into its performance and details.

Matrix Dimension	`3:1-cost` baseline	`xt-only-cost` baseline
2	8	2
3	15	3
4	40	8
5	120	30
6	234	59
7	384	96
8	296	72
16	4544	1248
32	20032	5440

Table 8.4: Matrix cost baselines for optimization attempts

Our experiments were carried out over a duration of approximately 20 hours on four different machines. Each experiment involved multiple combinations of GA parameters and settings, as indicated in Table 8.3. We believe that with further in-depth analysis, significant improvements could be achieved.

8.5 Results Summary

Table 8.5 briefly exemplifies a few of the MDS matrices found by our GA implementations, per different dimension. Several other matrices were found, but, since none of them presented better **xor** and **xtime** counts than the baselines, we do not dive deep into presenting them here, providing a few examples only. We also would like to remark that these example matrices have not been reported in the literature yet, to the best of our knowledge, and are thus **new** MDS matrices, despite having worse computational cost than the baselines.

The most promising of our obtained matrices seems to be matrix (8.6), which has `baseline diff = 0`, meaning it as as good as the baseline for dimension 6 (although not surpassing it).

Dim.	xor	xtime	Baseline Diff.	FF	Matrix
2	3	2	1	$GF(2^4)$	(8.1)
3	24	34	31	$GF(2^8)$	(8.2)
4	26	32	24	$GF(2^4)$	(8.3)
5	89	126	96	$GF(2^8)$	(8.4)
6	84	120	210	$GF(2^8)$	(8.5)
6	54	60	0	$GF(2^4)$	(8.6)
7	96	104	24	$GF(2^4)$	(8.7)
7	112	214	370	$GF(2^8)$	(8.8)

Table 8.5: Examples of matrices found by our Genetic Algorithms implementation, for different dimensions. The **Baseline Diff** column refers to the difference between the cost of the given matrix and the baseline cost, quantifying how much more expensive it is when compared to the state of the art.

$$\begin{bmatrix} 1 & 3 \\ 2 & 1 \end{bmatrix} \tag{8.1}$$

$$\begin{bmatrix} 173 & 6 & 197 \\ 205 & 2 & 60 \\ 9 & 3 & 3 \end{bmatrix} \tag{8.2}$$

$$\begin{bmatrix} 6 & 14 & 12 & 7 \\ 11 & 8 & 2 & 9 \\ 1 & 13 & 1 & 8 \\ 2 & 6 & 13 & 1 \end{bmatrix} \tag{8.3}$$

$$\begin{bmatrix} 19 & 31 & 49 & 10 & 29 \\ 135 & 207 & 55 & 232 & 179 \\ 208 & 13 & 63 & 103 & 28 \\ 10 & 148 & 48 & 52 & 12 \\ 201 & 254 & 14 & 25 & 27 \end{bmatrix} \tag{8.4}$$

$$\begin{bmatrix} 1 & 204 & 34 & 237 & 2 & 1 \\ 1 & 1 & 204 & 34 & 237 & 2 \\ 2 & 1 & 1 & 204 & 34 & 237 \\ 237 & 2 & 1 & 1 & 204 & 34 \\ 34 & 237 & 2 & 1 & 1 & 204 \\ 204 & 34 & 237 & 2 & 1 & 1 \end{bmatrix} \tag{8.5}$$

$$\begin{bmatrix} 1 & 1 & 9 & 12 & 9 & 3 \\ 1 & 9 & 12 & 9 & 3 & 1 \\ 9 & 12 & 9 & 3 & 1 & 1 \\ 12 & 9 & 3 & 1 & 1 & 9 \\ 9 & 3 & 1 & 1 & 9 & 12 \\ 3 & 1 & 1 & 9 & 12 & 9 \end{bmatrix} \tag{8.6}$$

$$\begin{bmatrix} 1 & 4 & 6 & 1 & 1 & 6 & 4 \\ 4 & 2 & 15 & 2 & 5 & 10 & 5 \\ 5 & 3 & 15 & 10 & 7 & 8 & 13 \\ 13 & 4 & 11 & 2 & 7 & 15 & 9 \\ 9 & 15 & 7 & 2 & 11 & 4 & 13 \\ 13 & 8 & 7 & 10 & 15 & 3 & 5 \\ 5 & 10 & 5 & 2 & 15 & 2 & 4 \end{bmatrix} \tag{8.7}$$

$$\begin{bmatrix} 1 & 1 & 16 & 178 & 88 & 164 & 92 \\ 1 & 16 & 178 & 88 & 164 & 92 & 1 \\ 16 & 178 & 88 & 164 & 92 & 1 & 1 \\ 178 & 88 & 164 & 92 & 1 & 2 & 16 \\ 88 & 164 & 92 & 1 & 2 & 16 & 178 \\ 164 & 92 & 1 & 2 & 16 & 178 & 88 \\ 92 & 1 & 2 & 16 & 178 & 88 & 164 \end{bmatrix} \tag{8.8}$$

It is possible to notice that there are 1s distributed in the coefficients of the matrices, which is expected, since one of our mutation functions involves taking random elements and replacing them with 1s.

8.6 Code

The code regarding all the experiments we performed and GA implementation is available on our Git repository [66], however, we do not make all our experiment log files available due to their size ($\approx$ 100GB in total).

8.7 Discussion

We briefly explored applying Genetic Algorithms to MDS matrix construction. In this section, we summarize our process and outline several alternative approaches and fine tuning options that could lead to interesting results, leaving them as future work.

Our experiments used two computational cost metrics, namely `3:1-cost` and `xt-only-cost`. Initially, we adopted the `3:1-cost` metric as it aligns with our original plan and considers both the **xtime** and **xor** operations, along with their computational costs, **xtime** being the more computationally expensive operation. However, upon observing that no MDS matrices surpassing the baseline performance were discovered in the first set of experiments, we relaxed the cost metric to `xt-only-cost` to explore potential outcomes. Unfortunately, this adjustment did not yield any significant improvements either. Additionally, we experimented with initializing the genetic algorithm using pre-existing solutions (the MDS matrices from our dataset) instead of using random initialization. Despite these efforts, we still did not achieve superior results compared to the baseline matrices. We did find matrices that satisfy the MDS property with the GA experiments, but, since none of them surpassed the computational cost baselines, we do not detail them here.

Given the preliminary, exploratory nature and limited scope of our experiments, it is crucial to acknowledge that our findings are inconclusive. The results neither categorize GA as an ineffective approach for low cost MDS matrix search nor validate it as a suitable method. However, we believe that with additional research and thorough refinement of the GA settings beyond what was conducted in this study, it is possible for GA to evolve into a promising approach for low cost MDS matrix construction.

8.7.1 Possible Improvements for the Algorithm

Several improvements in our algorithm can be considered. We list them here as suggestions for future work.

Mutation of Specific Matrices. An intriguing idea is to mutate specific matrices that are already good, such as those from the baselines. By applying successive mutations to these matrices, we can explore the potential for further improvements on the **xor** and **xtime** counts whilst keeping MDS property.

Crossover between Baseline Matrices. Another approach worth exploring is performing crossover between strategically chosen matrices from the baselines. This could involve e.g selecting the smallest coefficients from each matrix or trying all possible crossovers to identify MDS matrices with good cost metrics.

Customized Crossover and Mutation Methods. Consideration should be given to designing customized crossover and mutation methods that ensure the MDS property is maintained throughout the process. This would allow for more controlled exploration of the solution space.

Performance Evaluation against a Single Matrix. Instead of attempting to beat the state of the art, we could assess the performance of the GA when attempting to beat a single matrix. This analysis can provide valuable insights into the algorithm's optimization capabilities, convergence and yield fine tuning ideas that could later be applied to the more complex challenge of beating the state of the art **xor** and **xtime** counts.

Different Fitness Functions. An improvement worth investigating is the design of different fitness functions that consider both the MDS property and the computational cost. Finding better formulas that express the relevance of these factors can enhance the selection process and steer the GA towards more desirable solutions.

Exploration of Special Matrix Constructions. In our experiments, we built the whole matrices, selecting all the required n^2 elements. We did not explore the special constructions present in the literature studied in Chapter 7, such as Toeplitz, Hankel, Cauchy, Vandermonde, among others. Combining GA usage with special constructions, reducing the element selection space, or guaranteeing MDS property and allowing us to focus only on cost optimization, could yield interesting results towards finding more efficient MDS matrices.

Other Sophisticated AI Techniques. Investigating the usage of other forms of artificial intelligence, such as deep learning, recurrent neural networks (RNNs) or reinforcement learning (RL), could provide additional optimization capabilities. RNNs could be of special interest due to their capability of learning complex relationships between data. In our case, the neural network would try to learn the proper relationship between matrix coefficients to lead to MDS property and low computational cost.

Hybrid Approaches. Investigating hybrid approaches that combine GA with other AI, or investigating other variants of GA, can provide complementary strengths and potentially improve the search process as well.

Exploring Other Optimization and Combinatorics Heuristics. Expanding the exploration beyond GA to include other optimization and combinatorics heuristics can help identify alternative approaches and potentially more efficient algorithms.

Refining Mutation Functions. The current mutation functions in our experiments randomly change elements to 1s, which can have unintended consequences. As the number of such mutations increases, the matrix may become filled with multiple 1s, potentially leading to a full 1s matrix or even a singular submatrix (e.g., introducing a 2×2 square of 1s). Although this mutation was intended to reduce the cost, it may jeopardize the MDS property. To address this, it would be more beneficial to decrease the coefficient by 1 unit or by half, as opposed to replacing it with 1. This approach ensures a more gradual loss of MDS property while reducing the cost.

Customized Random Element Mutation. In addition to the mutation functions that change elements to random values, we can introduce restrictions to maintain cost efficiency. For example, we could always select a random element that is smaller than the current coefficient. Alternatively, we could employ a probability-based approach, such as a 50% chance of increasing the coefficient by one unit and a 50% chance of decreasing it. Further variations could favor decreasing with a 49% chance of increasing and a 51% chance of decreasing, or other probabilities that prioritize decreasing (e.g., 1/3 increasing and 2/3 decreasing). These refined mutation strategies ensure that the mutations are less drastic, enabling the preservation of the MDS property while moderately reducing the cost.

Involutory Constructions. Exploring involutory constructions, which possess desirable properties for cryptographic applications due to keeping encryption and decryption cost equal, could be incorporated into our GA framework, for example, in a fitness function that favored involutory constructions over non involutory ones.

Study GA for Error Correcting Codes. Since MDS matrices are related to Error Correcting Codes theory and there have been works using GA to find optimal Error Correcting Codes in the past, perhaps studying these works in further detail could provide insights for the MDS matrix search problem.

Baselines Considering Finite Fields. Recognizing the impact of finite fields on computational cost, establishing baselines not only per dimension but also per dimension combined with the finite field can provide more meaningful comparisons and evaluations. In our experiments, the baselines were per dimension, but not per finite field. Because of that, matrices in $GF(2^4)$, which have their coefficients limited to 15, have a chance of being cheaper than matrices in $GF(2^8)$, since their coefficients are limited to 255, and the bigger the integer value, the bigger the polynomial degree, and thus the bigger the **xtime** count. Therefore, when we are generating matrices in $GF(2^8)$, beating the baseline is more challenging, because the baseline cost is the one of a matrix whose's coefficients are limited to 15. If we have different baselines for different finite fields, we might provide more fair comparisons between our matrices and the baselines.

Gradient Score for MDS Property. Instead of a binary flag indicating whether a matrix is MDS or not, we could introduce a gradient score that quantifies the MDS property. For instance, we could consider the count of non-singular submatrices as a measure of MDS quality. By incorporating this score into the fitness function, the GA would be encouraged to gradually move towards matrices with higher non-singular counts, ultimately converging on the MDS property. This scoring mechanism provides a more nuanced assessment of matrix quality, leading to potentially better results, and better representing the relevance and trade off between MDS property and computational cost of the candidate matrices.

Studying GA for Specific Mathematical Properties. Investigating the application of GA for finding matrices with specific mathematical properties and small numbered coefficients can yield new ideas for enhancing the fitness function. By studying deeper the related literature, we can gain valuable insights into incorporating additional factors that promote the desired properties.

Rethinking the Cost Metrics. Considering the importance of the cost metrics, it is worthwhile to reevaluate its formulation. It is essential to recompute the baselines to ensure fair comparisons and accurate assessments of the cost metrics. Exploring different cost metrics other than `3:1-cost` and `xt-only-cost` can lead to improved performance and better alignment with the desired objectives.

Customizing the other steps of GA. GA comprises other steps other than crossover and mutation, namely selection, pairing, reproduction and replacement. Investigating customized approaches for these steps that could maximize number of MDS matrices and contribute to **xor** and **xtime** optimized counts would also be of interest, e.g different replacement methods, or pruning the population to keep only the best matrices.

8.7.2 Possible Improvements for the Experimental Process

There are improvements that could enhance our experimental suite itself, facilitating a more comprehensive analysis of the results. We list them here as suggestions as well.

Expanded Metrics Log. It is beneficial to include metrics for non-MDS matrices and observe the ratio of MDS versus non-MDS matrices throughout the experiment. By incorporating this information into the metrics log, a comparison can be made to understand the relative difficulty of finding cheap matrices, MDS matrices and matrices that are both cheap and MDS. This comprehensive comparison allows for a deeper understanding of the trade-offs involved in the search for optimal MDS matrices.

Exact Time Measurements. Accurate and precise time measurements are helpful to gain a better understanding of the practical computational requirements and time complexity associated with the MDS matrix search process, instead of restricting the knowledge about the topic to the mathematical formulas and assymptotic complexity.

Detailed Analysis of Logs and Progress. Performing a more deep and detailed analysis of the experiment logs and monitoring the progression of mutations and crossovers within the population can provide valuable insights. By examining the mutation and crossover patterns, it becomes possible to identify trends and patterns that may lead to improved performance and guide future iterations of the algorithm.

Graphical Representation of Average Fitness. Incorporating graphs that display the average fitness of the population over time provides a visual representation of the algorithm's progress. These graphs could allow for easier tracking of performance improvements or plateaus, enabling researchers to gain a clearer understanding of the search dynamics and identify potential areas for refinement.

Automatic Convergence Detector. Instead of solely limiting the maximum iterations, implementing an automatic convergence detection mechanism can provide more precise termination conditions. This detector can be based on specific criteria such as stagnation of fitness improvement, stability of the population, or other convergence indicators. By incorporating such a mechanism, experiments can be efficiently terminated when further iterations no longer yield significant improvements.

Improved Data Visualization. Enhancing the data visualization capabilities of the experimental suite is crucial for better comprehension and analysis of the results. Utilizing graphical representations such as graphs, heatmaps, and other visualizations can facilitate the exploration of large log files and aid in identifying patterns, trends, and anomalies more efficiently.

Parallelism. By employing parallel computing techniques, genetic algorithms can achieve significant speedup and enhanced exploration of the search space. Parallelism enables the execution of multiple genetic algorithm instances simultaneously, each exploring a different portion of the solution space. This approach not only accelerates the overall optimization process but also improves the diversity of solutions generated. By allowing multiple individuals or populations to evolve independently, parallel genetic algorithms can effectively avoid getting trapped in local optima, promoting the discovery of better solutions. Additionally, parallelism enables efficient utilization of modern hardware architectures, such as multi-core processors or distributed computing systems, maximizing computational resources and enabling the handling of larger problem sizes. We did not use parallelism in our Python experiments suite implementation, but we believe it could be especially useful to make the experiments faster.

By incorporating these improvements into the experimental suite, we could can gain deeper insights, conduct more thorough analyses, and obtain a richer understanding of the MDS matrix search process, facilitating decision-making regarding algorithmic refinements and future research directions.

8.8 Appendix A: Full Machine Specifications

Listing 8.1: Full specification for `ifrit`

```
Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 12
On-line CPU(s) list: 0-11
Thread(s) per core: 2
Core(s) per socket: 6
Socket(s): 1
NUMA node(s): 1
Vendor ID: GenuineIntel
CPU family: 6
Model: 158
Model name: Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz
Stepping: 10
CPU MHz: 4469.432
CPU max MHz: 4600.0000
CPU min MHz: 800.0000
BogoMIPS: 6399.96
Virtualization: VT-x
L1d cache: 32K
L1i cache: 32K
L2 cache: 256K
L3 cache: 12288K
```

```
NUMA node0 CPU(s): 0-11
Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
    clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm
    constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid
    aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma
    cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes
    xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single
    pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase
    tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm mpx rdseed adx smap
    clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp
    hwp_notify hwp_act_window hwp_epp md_clear flush_l1d arch_capabilities
```

Listing 8.2: Full specification for `termina`

```
Architecture: x86_64
  CPU op-mode(s): 32-bit, 64-bit
  Address sizes: 39 bits physical, 48 bits virtual
  Byte Order: Little Endian
CPU(s): 12
  On-line CPU(s) list: 0-11
Vendor ID: GenuineIntel
  Model name: Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz
    CPU family: 6
    Model: 158
    Thread(s) per core: 2
    Core(s) per socket: 6
    Socket(s): 1
    Stepping: 10
    CPU(s) scaling MHz: 61%
    CPU max MHz: 4600.0000
    CPU min MHz: 800.0000
    BogoMIPS: 6402.62
    Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mc
                   a cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss
                   ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art
                    arch_perfmon pebs bts rep_good nopl xtopology nonstop_
                   tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cp
                   l vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid ss
                   e4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes
                   xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_f
                   ault invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow
                    vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust
                   bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap cl
                   flushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm
                    ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp
                    md_clear flush_l1d arch_capabilities
Virtualization features:
  Virtualization: VT-x
Caches (sum of all):
  L1d: 192 KiB (6 instances)
  L1i: 192 KiB (6 instances)
  L2: 1.5 MiB (6 instances)
  L3: 12 MiB (1 instance)
NUMA:
  NUMA node(s): 1
  NUMA node0 CPU(s): 0-11
Vulnerabilities:
  Itlb multihit: KVM: Mitigation: VMX disabled
  L1tf: Mitigation; PTE Inversion; VMX conditional cache flushe
```

```
                     s, SMT vulnerable
  Mds: Mitigation; Clear CPU buffers; SMT vulnerable
  Meltdown: Mitigation; PTI
  Mmio stale data: Mitigation; Clear CPU buffers; SMT vulnerable
  Retbleed: Mitigation; IBRS
  Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl
  Spectre v1: Mitigation; usercopy/swapgs barriers and __user pointer
                      sanitization
  Spectre v2: Mitigation; IBRS, IBPB conditional, RSB filling, PBRSB-
                     eIBRS Not affected
  Srbds: Mitigation; Microcode
  Tsx async abort: Mitigation; TSX disabled
```

Listing 8.3: Full specification for `mac`

```
machdep.cpu.mwait.linesize_min: 64
machdep.cpu.mwait.linesize_max: 64
machdep.cpu.mwait.extensions: 3
machdep.cpu.mwait.sub_Cstates: 286531872
machdep.cpu.thermal.sensor: 1
machdep.cpu.thermal.dynamic_acceleration: 1
machdep.cpu.thermal.invariant_APIC_timer: 1
machdep.cpu.thermal.thresholds: 2
machdep.cpu.thermal.ACNT_MCNT: 1
machdep.cpu.thermal.core_power_limits: 1
machdep.cpu.thermal.fine_grain_clock_mod: 1
machdep.cpu.thermal.package_thermal_intr: 1
machdep.cpu.thermal.hardware_feedback: 0
machdep.cpu.thermal.energy_policy: 1
machdep.cpu.xsave.extended_state: 31 832 1088 0
machdep.cpu.xsave.extended_state1: 15 832 256 0
machdep.cpu.arch_perf.version: 4
machdep.cpu.arch_perf.number: 4
machdep.cpu.arch_perf.width: 48
machdep.cpu.arch_perf.events_number: 7
machdep.cpu.arch_perf.events: 0
machdep.cpu.arch_perf.fixed_number: 3
machdep.cpu.arch_perf.fixed_width: 48
machdep.cpu.cache.linesize: 64
machdep.cpu.cache.L2_associativity: 4
machdep.cpu.cache.size: 256
machdep.cpu.tlb.inst.large: 8
machdep.cpu.tlb.data.small: 64
machdep.cpu.tlb.data.small_level1: 64
machdep.cpu.address_bits.physical: 39
machdep.cpu.address_bits.virtual: 48
machdep.cpu.tsc_ccc.numerator: 216
machdep.cpu.tsc_ccc.denominator: 2
machdep.cpu.max_basic: 22
machdep.cpu.max_ext: 2147483656
machdep.cpu.vendor: GenuineIntel
machdep.cpu.brand_string: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
machdep.cpu.family: 6
machdep.cpu.model: 158
machdep.cpu.extmodel: 9
machdep.cpu.extfamily: 0
machdep.cpu.stepping: 10
machdep.cpu.feature_bits: 9221959987971750911
machdep.cpu.leaf7_feature_bits: 43804591 1073741824
machdep.cpu.leaf7_feature_bits_edx: 3154128384
machdep.cpu.extfeature_bits: 1241984796928
machdep.cpu.signature: 591594
machdep.cpu.brand: 0
machdep.cpu.features: FPU VME DE PSE TSC MSR PAE MCE CX8 APIC SEP MTRR PGE MCA CMOV
    PAT PSE36 CLFSH DS ACPI MMX FXSR SSE SSE2 SS HTT TM PBE SSE3 PCLMULQDQ DTES64 MON
    DSCPL VMX EST TM2 SSSE3 FMA CX16 TPR PDCM SSE4.1 SSE4.2 x2APIC MOVBE POPCNT AES
    PCID XSAVE OSXSAVE SEGLIM64 TSCTMR AVX1.0 RDRAND F16C
machdep.cpu.leaf7_features: RDWRFSGS TSC_THREAD_OFFSET SGX BMI1 AVX2 SMEP BMI2 ERMS
    INVPCID FPU_CSDS MPX RDSEED ADX SMAP CLFSOPT IPT SGXLC MDCLEAR TSXFA IBRS STIBP
    L1DF ACAPMSR SSBD
machdep.cpu.extfeatures: SYSCALL XD 1GBPAGE EM64T LAHF LZCNT PREFETCHW RDTSCP TSCI
machdep.cpu.logical_per_package: 16
machdep.cpu.cores_per_package: 8
machdep.cpu.microcode_version: 240
machdep.cpu.processor_flag: 5
machdep.cpu.core_count: 6
machdep.cpu.thread_count: 12
```

Listing 8.4: Full specification for `finger`

```
Architecture: x86_64
  CPU op-mode(s): 32-bit, 64-bit
  Address sizes: 39 bits physical, 48 bits virtual
  Byte Order: Little Endian
CPU(s): 12
  On-line CPU(s) list: 0-11
Vendor ID: GenuineIntel
  Model name: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
    CPU family: 6
    Model: 165
    Thread(s) per core: 2
    Core(s) per socket: 6
    Socket(s): 1
    Stepping: 2
    CPU max MHz: 5000.0000
    CPU min MHz: 800.0000
    BogoMIPS: 5199.98
    Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mc
                     a cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss
                     ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art
                      arch_perfmon pebs bts rep_good nopl xtopology nonstop_
                     tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cp
                     l vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1
                      sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsav
                     e avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault
                      epb invpcid_single ssbd ibrs ibpb stibp ibrs_enhanced
                     tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase t
                     sc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx rdseed a
                     dx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsa
                     ves dtherm ida arat pln pts hwp hwp_notify hwp_act_wind
                     ow hwp_epp pku ospke md_clear flush_l1d arch_capabiliti
                     es
Virtualization features:
  Virtualization: VT-x
Caches (sum of all):
  L1d: 192 KiB (6 instances)
  L1i: 192 KiB (6 instances)
  L2: 1.5 MiB (6 instances)
  L3: 12 MiB (1 instance)
NUMA:
  NUMA node(s): 1
  NUMA node0 CPU(s): 0-11
Vulnerabilities:
  Itlb multihit: KVM: Mitigation: VMX disabled
  L1tf: Not affected
  Mds: Not affected
  Meltdown: Not affected
  Mmio stale data: Mitigation; Clear CPU buffers; SMT vulnerable
  Retbleed: Mitigation; Enhanced IBRS
  Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl
                       and seccomp
  Spectre v1: Mitigation; usercopy/swapgs barriers and __user pointer
                       sanitization
  Spectre v2: Mitigation; Enhanced IBRS, IBPB conditional, RSB fillin
                      g, PBRSB-eIBRS SW sequence
  Srbds: Mitigation; Microcode
  Tsx async abort: Not affected
```

Conclusions

In summary, this thesis contributes to the understanding and analysis of symmetric cryptographic algorithms. The comparative analysis of ANFs, the graph diffusion analysis model, the detailed explanations of DDTs and LATs, and the creation of the MDS matrix catalogue all provide a comprehensive foundation for cryptographic research. The availability of code implementations and suggestions for improvement ensures that this work can be built upon by future researchers, ultimately leading to the development of more secure and robust symmetric cryptographic algorithms. Furthermore, we have also explored the application of Genetic Algorithms (GA) for MDS matrix construction. While the GA experiments did not yield significant results in terms of improving the **xor** and **xtime** counts, we provide valuable insights and suggestions for future improvements to enhance the GA approach. This thesis presents the first catalogue with ≈ 150 MDS matrices from 22 years of cryptography literature along with their computational costs, providing a valuable resource for future cryptographic algorithm designs.

One possible avenue for improvement involves mutating specific matrices that are already considered good, such as those from the baselines. By applying successive mutations to these matrices, researchers can explore the potential for further improvements in terms of both **xor** and **xtime** counts while preserving the MDS property. Additionally, performing crossover between strategically chosen matrices from the baselines could offer promising results, especially when selecting the smallest coefficients or exploring all possible crossovers. Customized crossover and mutation methods could be designed to ensure the maintenance of the MDS property throughout the process, providing more controlled exploration of the solution space. It is also worth considering different fitness functions that take into account both the MDS property and the computational cost, allowing for a more refined selection process. We also suggest exploring other sophisticated AI techniques, such as deep learning, recurrent neural networks (RNNs), or reinforcement learning (RL), to provide additional optimization capabilities. Furthermore, investigating hybrid approaches that combine GA with other AI techniques or exploring different variants of GA can potentially improve the search process.

Bibliography

[1] Paulo Barreto. Whirlpool Web Page. `https://web.archive.org/web/20171129084214/http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html`, 2008. [Online; accessed 01-February-2022].

[2] Paulo S. L. M. Barreto, Ventzislav Nikov, Svetla Nikova, Vincent Rijmen, and Elmar Tischhauser. Whirlwind: a new cryptographic hash function. *Des. Codes Cryptogr.*, 56(2-3):141–162, 2010.

[3] Paulo S. L. M. Barreto and Vincent Rijmen. The ANUBIS block cipher. In *First NESSIE Workshop, Heverlee, Belgium*, 2000.

[4] Paulo S. L. M. Barreto and Vincent Rijmen. The KHAZAD Legacy-Level block cipher. In *First NESSIE Workshop, Heverlee, Belgium*, 2000.

[5] PSLM Barreto and M Simplicio. Curupira, a block cipher for constrained platforms. *Anais do 25o Simpsio Brasileiro de Redes de Computadores e Sistemas Distribudos-SBRC*, 1:61–74, 2007.

[6] K.G. Beauchamp. *Walsh Functions and Their Applications*. Nutrition, Basic and Applied Science. Academic Press, 1975.

[7] Christof Beierle, Thorsten Kranz, and Gregor Leander. Lightweight multiplication in $GF(2^n)$ with applications to MDS matrices. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 625–653. Springer, 2016.

[8] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 313–314. Springer, 2013.

[9] Eli Biham, Ross J. Anderson, and Lars R. Knudsen. Serpent: A new block cipher proposal. In Serge Vaudenay, editor, *Fast Software Encryption, 5th International Workshop, FSE '98, Paris, France, March 23-25, 1998, Proceedings*, volume 1372 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 1998.

[10] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[11] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.

[12] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[13] Anne Canteaut. Lecture Notes on Cryptographic Boolean Functions. `https://www.rocq.inria.fr/secret/Anne.Canteaut/poly.pdf`, 2016. [Online; accessed 07-June-2021].

[14] Don Coppersmith. The data encryption standard (DES) and its strength against attacks. *IBM J. Res. Dev.*, 38(3):243–250, 1994.

[15] Ben Coppin. *Artificial Intelligence Illuminated.* Jones and Bartlett Publishers, Inc., USA, 2004.

[16] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009.

[17] Toshiba Corporation. Specification of Hierocrypt-3. In *First NESSIE Workshop, Heverlee, Belgium*, 2000.

[18] Toshiba Corporation. Specification of Hierocrypt-L1. In *First NESSIE Workshop, Heverlee, Belgium*, 2000.

[19] Ting Cui, Chenhui Jin, and Zhiyin Kong. On compact cauchy matrices for substitution-permutation networks. *IEEE Trans. Computers*, 64(7):2098–2102, 2015.

[20] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher SQUARE. In Eli Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer, 1997.

[21] Joan Daemen and Vincent Rijmen. The block cipher BKSQ. In Jean-Jacques Quisquater and Bruce Schneier, editors, *Smart Card Research and Applications, This International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings*, volume 1820 of *Lecture Notes in Computer Science*, pages 236–245. Springer, 1998.

[22] Joan Daemen and Vincent Rijmen. AES Proposal: Rijndael. 10 1999.

[23] Joan Daemen and Vincent Rijmen. The wide trail design strategy. In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, volume 2260 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 2001.

[24] Joan Daemen and Vincent Rijmen. *The Design of Rijndael.* Springer-Verlag, Berlin, Heidelberg, 2002.

[25] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard.* Information Security and Cryptography. Springer, 2002.

[26] Yoni De Mulder. *White-Box Cryptography: Analysis of White-Box AES Implementations (White-Box Cryptografie: Analyse van White-Box AES implementaties).* PhD thesis, Katholieke Universiteit Leuven, Belgium, 2014.

[27] The Sage Developers, William Stein, David Joyner, David Kohel, John Cremona, and Burçin Eröcal. Sagemath, version 9.0, 2020.

[28] Sébastien Duval and Gaëtan Leurent. MDS matrices with lightweight circuits. *IACR Cryptol. ePrint Arch.*, page 260, 2018.

[29] Kamil Dworak, Jakub Nalepa, Urszula Boryczka, and Michal Kawulok. *Cryptanalysis of SDES Using Genetic and Memetic Algorithms*, pages 3–14. Springer International Publishing, Cham, 2016.

[30] Horst Feistel. Cryptography and computer privacy. *Scientific American*, 228(5):15–23, 1973.

[31] Electronic Frontier Foundation, Mike Loukides, and John Gilmore. *Cracking DES: Secrets of Encryption Research, Wiretap Politics and Chip Design.* O'Reilly & Associates, Inc., USA, 1998.

[32] Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Grøstl - a SHA-3 candidate. In Helena Handschuh, Stefan Lucks, Bart Preneel, and Phillip Rogaway, editors, *Symmetric Cryptography, 11.01. - 16.01.2009*, volume 09031 of *Dagstuhl Seminar Proceedings.* Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2009.

[33] Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 222–239. Springer, 2011.

[34] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED block cipher. *IACR Cryptol. ePrint Arch.*, page 600, 2012.

[35] Kishan Chand Gupta, Sumit Kumar Pandey, Indranil Ghosh Ray, and Susanta Samanta. Cryptographically significant mds matrices over finite fields: A brief survey and some generalized results, 2019.

[36] Kishan Chand Gupta and Indranil Ghosh Ray. On constructions of involutory mds matrices. In *AFRICACRYPT*, 2013.

[37] Kenneth Hoffmann and Ray Kunze. Linear algebra. *Mathematics of Computation*, 15(75):407, 1971.

[38] Matt Hostetter. Galois: A performant NumPy extension for Galois fields, 11 2020.

[39] Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. Joltik v1. 3. *CAESAR Round*, 2, 2015.

[40] András Joó, Anikó Ekárt, and Juan Pablo Neirotti. Genetic algorithms for discovery of matrix multiplication methods. *IEEE Trans. Evol. Comput.*, 16(5):749–751, 2012.

[41] Pascal Junod and Serge Vaudenay. FOX : A new family of block ciphers. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography, 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers*, volume 3357 of *Lecture Notes in Computer Science*, pages 114–129. Springer, 2004.

[42] Elif Bilge Kavun, Martin M. Lauridsen, Gregor Leander, Christian Rechberger, Peter Schwabe, and Tolga Yalçın. Prøst. CAESAR Proposal, 2014. `http://proest.compute.dtu.dk`.

[43] L. Knudsen. Iterative characteristics of des and s^2-des. In *CRYPTO*, 1992.

[44] Alan G. Konheim. *Cryptography, a Primer.* John Wiley & Sons, Inc., USA, 1st edition, 1981.

[45] Jérôme Lacan and Pascal Chatonnay. Search of optimal error correcting codes with genetic algorithms. In Bernd Reusch, editor, *Computational Intelligence*, pages 93–98, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[46] X Lai and J Massey. The idea block cipher. *NESSIE Block Cipher Submissions*, 2000.

[47] Susan Landau. Polynomials in the nation's service: Using algebra to design the advanced encryption standard. *The American Mathematical Monthly*, 111:89–117, 02 2004.

[48] S. Lang. *Linear Algebra.* Springer Undergraduate Texts in Mathematics and Technology. Springer, 1987.

[49] Meicheng Liu and Siang Meng Sim. Lightweight MDS generalized circulant matrices. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 101–120. Springer, 2016.

[50] Meicheng Liu and Siang Meng Sim. Lightweight mds generalized circulant matrices. In Thomas Peyrin, editor, *Fast Software Encryption*, pages 101–120, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[51] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes.* North-holland Publishing Company, 2nd edition, 1978.

[52] James L. Massey. SAFER K-64: A byte-oriented block-ciphering algorithm. In Ross J. Anderson, editor, *Fast Software Encryption, Cambridge Security Workshop, Cambridge, UK, December 9-11, 1993, Proceedings*, volume 809 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 1993.

[53] A.M. Masuda and D. Panario. *Tópicos de corpos finitos com aplicações em criptografia e teoria de códigos.* Publicações matemáticas. IMPA, 2007.

[54] M. Matsui. Linear cryptanalysis method for des cipher. In *EUROCRYPT*, 1993.

[55] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography.* CRC Press, Inc., USA, 1st edition, 1996.

[56] G.L. Mullen and D. Panario. *Handbook of Finite Fields*. Discrete Mathematics and Its Applications. CRC Press, 2013.

[57] Jorge Nakahara and Élcio Abrahão. A new involutory mds matrix for the aes. *Int. J. Netw. Secur.*, 9:109–116, 2009.

[58] National Institute of Standards and Technology. *Data Encryption Standard (DES)*, 1977. Federal Information Processing Standards Publication 46, `https://csrc.nist.gov/CSRC/media/Publications/fips/46/archive/1977-01-15/documents/NBS.FIPS.46.pdf`,.

[59] National Institute of Standards and Technology. *Announcing the Advanced Encryption Standard (AES)*, 2001. Federal Information Processing Standards Publication 197, `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.

[60] K. Nyberg. Differentially uniform mappings for cryptography. In *EUROCRYPT*, 1993.

[61] Christof Paar and Jan Pelzl. *Understanding Cryptography - A Textbook for Students and Practitioners*. Springer, 2010.

[62] Vincent Rijmen, Joan Daemen, Bart Preneel, Antoon Bosselaers, and Erik De Win. The cipher SHARK. In Dieter Gollmann, editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings*, volume 1039 of *Lecture Notes in Computer Science*, pages 99–111. Springer, 1996.

[63] Sumanta Sarkar and Habeeb Syed. Lightweight diffusion layer: Importance of toeplitz matrices. *IACR Trans. Symmetric Cryptol.*, 2016:95–113, 2016.

[64] Sumanta Sarkar and Habeeb Syed. Analysis of Toeplitz MDS matrices. *IACR Cryptol. ePrint Arch.*, page 368, 2017.

[65] Akashi Satoh, Sumio Morioka, Kohji Takano, and Seiji Munetoh. A compact rijndael hardware architecture with s-box optimization. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 239–254, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[66] Ana Clara Zoppi Serpa. My diffusion studies codes and experimental data. `https://github.com/AnaClaraZoppiSerpa/diffusion-studies-supporting-codes`, 2023.

[67] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949.

[68] Taizo Shirai and Kyoji Shibutani. On the diffusion matrix employed in the whirlpool hashing function. 01 2003.

[69] Paulo S.L.M and Vincent Rijmen. The whirlpool hashing function. 2003.

[70] S.M.Sim, K.Khoo, F.Oggier, and T.Peyrin. Lightweight MDS involution matrices (full version). IACR Cryptology ePrint Archive, Report 2015/258, 2015. `https://eprint.iacr.org/2015/258.pdf`.

[71] Muhammad Tahir, Muhammad Sardaraz, Zahid Mehmood, and Shakoor Muhammad. Cryptoga: a cryptosystem based on genetic algorithm for cloud data security. *Cluster Computing*, 24(2):739–752, Jun 2021.

[72] Fernando Von Zuben. Computação evolutiva: Uma abordagem pragmática. 01 2000.

[73] A. M. Youssef, S. Mister, and S. E. Tavares. On the design of linear transformations for substitution permutation encryption networks. In C. Adams and M. Just, editors, *Selected Areas in Cryptography, 11th International Workshop, SAC 1997, Ottawa, Canada, August 11 - 12, 1997, Revised Selected Papers*, pages 40–48. Springer, 1997.

Appendix A

Background on AES, DES, block ciphers, Feistel Networks and Substitution Permutation Networks

A.1 Formal Definitions of Block Ciphers

In [30], Feistel explains how the fact that information (e.g text) can be represented by bits allows the creation of complex structures for encryption, since the usage of bits allows logical operations (e.g AND, OR, XOR, NOT) to be performed and combined. He defines a *block cipher* as an algorithm which encrypts n_b bits of plaintext at a time. He also introduces the concept of substitution boxes (*S-Boxes*) and the idea of combining permutations and *S-Boxes* in order to build strong block ciphers. Feistel's ideas, too, served as the foundation for the design of modern block ciphers.

Other definitions of block ciphers and types of block ciphers also formalize the concept, as follows.

According to [26], a block cipher is a pair of cryptographic algorithms, one to encrypt and another to decrypt, operating on a *block* of fixed n_b bits; n_b is the *block size.* Encryption and decryption are performed using a secret key of n_k bits.

Definition 38 (Block cipher) *A block cipher of n_b bits $\mathcal{E}$ is a deterministic function mapping a block of plaintext of n_b bits to a block of ciphertext of n_b bits, under action of a secret key k of n_k bits:*

$$\mathcal{E} : \mathbb{F}_2^{n_b} \times \mathbb{F}_2^{n_k} \to \mathbb{F}_2^{n_b} : (m, k) \mapsto c = \mathcal{E}_k(m), \tag{A.1}$$

where n_b and n_k are the block size and the key size of the block cipher, respectively. The decryption process maps ciphertexts to their original plaintext, and is denoted by $\mathcal{D}$ or $\mathcal{D}_k$.

The functions $\mathcal{E}_k$ and $\mathcal{D}_k$ must satisfy $\mathcal{D}_k(\mathcal{E}_k(m)) = m$, $\forall m \in \mathbb{F}_2^{n_b}, \forall k \in \mathbb{F}_2^{n_k}$, and one should only be able to decrypt a text which has been encrypted with key k using the same key k itself. These algorithms are used to encrypt and/or decrypt plaintext blocks of n_b bits, thus providing confidentiality.

An iterative block cipher is a block cipher constructed by iterating a key-dependent function several times. Their core concept is that a single iteration may result in a weak block cipher, but several iterations (also known as "rounds") may result in a secure block cipher.

Definition 39 (Iterative Block Cipher) *An n_b-bit block cipher $\mathcal{E}$ is iterative with n_r rounds if, for each key $k \in \mathbb{F}_2^{n_k}$, the mapping $\mathcal{E}_k$ is the iterative application of n_r key-dependent round transformations over the plaintext m, i.e,*

$$\mathcal{E}_k(m) = \mathcal{E}^{(n_r)}_{k^{(n_r)}} \circ \ldots \circ \mathcal{E}^{(2)}_{k^{(2)}} \circ \mathcal{E}^{(1)}_{k^{(1)}}(m) = \mathcal{E}^{(n_r)}_{k^{n_r}}(\mathcal{E}^{(n_r-1)}_{k^{n_r-1}}(...(\mathcal{E}^{(1)}_{k^1}(m))...).$$

Each transformation $\mathcal{E}^{(r)}_{k^{(r)}}, 1 \leq r \leq n_r$, is a permutation in $\mathbb{F}_2^{n_b}$ and $k^{(r)}$ denotes a r-th subkey. Subkeys are derived from k through a key expansion algorithm. Subkeys are also known as round keys, and the key expansion algorithm is also known as key scheduling algorithm.

Two of the main kinds of iterative block ciphers are *Feistel Networks* and *Substitution Permutation Networks* (SPNs) [61].

Definition 40 (Feistel Network.) *A Feistel Network is an n_r-round iterative block cipher constructed in the following way:*

- *Let F be a round function, and let $k_1, k_2, \dots, k_{n_r}$ be the subkeys derived from the master key k.*
- *A plaintext block m is split in two halves L_0 and R_0.*
- *The i-th round output is $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus F(R_{i-1}, k_i)$.*
- *The ciphertext, i.e the n_r-th round output, is $(R_{n_r},\ L_{n_r})$.*

Definition 41 (Substitution Permutation Network Cipher.) *An SPN cipher is an n_r-round iterative block cipher in which each round transformation $\mathcal{E}^{(r)}_{k^{(r)}}, 1 \leq r \leq n_r$, consists of three layers (not necessarily in this order):*

- *A confusion layer, implemented with S-boxes (substitution boxes);*
- *A diffusion layer with D-boxes (diffusion boxes) or transpositions;*
- *A key addition layer.*

Furthermore, these layers must be invertible (bijective).

A.2 The DES block cipher

DES stands for "Data Encryption Standard" [58]. It is an algorithm standardized by NBS (National Bureau of Standards), which later became NIST (National Institute of Standards and Technology). The complete specification is available in [58], but cryptography textbooks such as [55] and [61] also provide descriptions of DES.

DES operates on a 64-bit plaintext. The right 32-bit half of the input plaintext is denoted here as R_0, and the left 32-bit half is L_0, thus the plaintext is $L_0||R_0$, where $||$ indicates the concatenation operation. The key is denoted by k. DES is a 16-round Feistel Network. Each round requires a subkey, as explained in the previous section. Here, the subkeys are denoted by k_i, where i is the current round.

Firstly, an initial permutation, known as *IP*, is applied to the plaintext. Then, the 16 rounds of the cipher, following the Feistel Network structure and using the subkeys. After the 16-th round, *IP* is reversed, and the ciphertext is returned as the cipher's output.

Let L_i be the left 32-bit half of the text *after round i has been performed*, and R_i be the right 32-bit half of the text, also after round i has been performed. Hence the plaintext is $L_0||R_0$, and then after the first round our state is $L_1||R_1$, and then $L_2||R_2$, and so forth, until $L_{16}||R_{16}$ (our final state and thus the ciphertext) is reached. From the Feistel Network structure, at each round i, L_i and R_i are obtained as follows:

$$L_i = R_{i-1}, \tag{A.2}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, k_i), \tag{A.3}$$

where F is the DES core transformation function. One should note that the Feistel structure itself, without a proper F function, would not provide secure encryption, for it performs only XOR operations — and these alone do not add sufficient diffusion/confusion to the cipher. The security of DES (and of other Feistel Networks, too) depends on a careful design of the F function.

The F function consists of: 1) expanding R_{i-1} to obtain 48 bits from its original 32 bits, 2) performing a logical XOR between the round key k_i and the expanded bits, 3) applying S-Boxes to the XOR result and finally, 4) a permutation P on the overall result. Hence $F = P(S(E(R_{i-1}) \oplus k_i))$, where E denotes the expansion, S denotes application of S-Boxes and P denotes permutation.

An S-Box is a *substitution box*, i.e a function, which maps a fixed size input to a fixed size output, usually in a non-linear way. For instance, an S-Box can map an input integer X to an output integer Y. They are usually interpreted and provided in algorithm specification documents (such as [58]) as lookup tables.

In the case of DES, there are 8 S-Boxes. Each of them takes a 6-bit input and maps it to a 4-bit output. So, since S denotes the application of S-Boxes, this means that the 48-bit result of the expansion function E is split into 8 parts, and each of these 8 parts is fed into one of the S-Boxes. They are called $S_1, S_2, S_3, S_4, S_5, S_6, S_7$ and S_8. And, since each of them gives us a 4-bit result, $S(E(R_{i-1}) \oplus k_i)$ will contain 32 bits, which will then be permuted with P and fed to the next DES round.

The DES algorithm achieves complete plaintext diffusion after 5 iterations, according to [44]. In [44], a diffusion table is provided, indicating how many plaintext bits have been spread to the cipher's intermediate state at each round. Complete diffusion occurs when each state bit depends on all the 64 plaintext bits, and that occurs only after the fifth iteration of DES.

Furthermore, the S-boxes are an essential component for the diffusion and the confusion within the DES F function. They were actually carefully designed to spread all the input bits to the output bits: each output bit of an S-box does depend on all of the 6 input bits. Regarding diffusion throughout the DES iterations, the permutation P plays a central role. More about the design criteria of DES can be found in [14].

Details about the permutations IP and P, the expansion function E and the 8 S-boxes are not provided in this text, but they can be found on the algorithm's specification [58] or on Cryptography textbooks such as [55].

A.3 The AES block cipher

The *Advanced Encryption Standard* (AES) [59] is a standard defined by the National Institute of Standards and Technology (NIST) for cryptographic protection of electronic data. It was defined in 2001, after a long open process for substitution of the *Data Encryption Standard* (DES), which dates from the 70s. Unlike DES, AES is not a Feistel Network. It is an SPN (Substitution Permutation Network).

An AES round consists of the successive application of 4 steps: *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey*. The first and the last round are exceptions, since MixColumns is not performed in the last round, and the first round contains an extra *AddRoundKey*. In Algorithm 1 below, r is the current round, n_r is the total number of rounds, and k_i, from k_0 to k_{n_r}, are the round keys.

NIST standardizes quantities n_r of rounds and bit lengths n_k for the key, hence the following AES types exist: AES-128 ($n_k = 128$, $n_r = 10$), AES-192 ($n_k = 192$, $n_r = 12$) and AES-256 ($n_k = 256$, $n_r = 14$). The block size is always $n_b = 128$. Other aspects and details, such as the key derivation algorithm, are covered in the algorithm's specification [59] and Cryptography textbooks, such as [61].

Algorithm 7 AES Encryption

```
Require: $x$ (plaintext), $k_0, k_1, \ldots, k_{n_r}$ (subkeys)
Ensure: $y$ (ciphertext)
  $x \leftarrow \texttt{AddRoundKey}(x, k_0)$
  for $r$ from 1 to $n_r - 1$ do
      $x \leftarrow \texttt{SubBytes}(x)$
      $x \leftarrow \texttt{ShiftRows}(x)$
      $x \leftarrow \texttt{MixColumns}(x)$
      $x \leftarrow \texttt{AddRoundKey}(x, k_r)$
  end for
  $x \leftarrow \texttt{SubBytes}(x)$
  $x \leftarrow \texttt{ShiftRows}(x)$
  $x \leftarrow \texttt{AddRoundKey}(x, k_{n_r})$
  $y \leftarrow x$
  return $y$
```

For a better understanding on these operations, the state is represented as a 4x4 matrix in which each element is a byte. In the following example, a 16 byte state is considered, and these bytes are $x_0, x_1, \ldots, x_{15}$. Such state is represented in 4 columns:

$$\begin{bmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{bmatrix}$$

SubBytes. Replace each state byte according to a non-linear S-Box S ($s_i \leftarrow S(s_i)$). The new state, after SubBytes has been applied, is, therefore:

$$\begin{bmatrix} S(s_0) & S(s_4) & S(s_8) & S(s_{12}) \\ S(s_1) & S(s_5) & S(s_9) & S(s_{13}) \\ S(s_2) & S(s_6) & S(s_{10}) & S(s_{14}) \\ S(s_3) & S(s_7) & S(s_{11}) & S(s_{15}) \end{bmatrix}$$

ShiftRows. The rows of the matrix suffer cyclic shifts, except for the first row. The second is shifted one cell to the left. The third, two cells to the left; and the fourth, three cells to the left, resulting in the following state:

$$\begin{bmatrix} S(s_0) & S(s_4) & S(s_8) & S(s_{12}) \\ S(s_5) & S(s_9) & S(s_{13}) & S(s_1) \\ S(s_{10}) & S(s_{14}) & S(s_2) & S(s_6) \\ S(s_{15}) & S(s_3) & S(s_7) & S(s_{11}) \end{bmatrix}$$

MixColumns. Multiplication of each column of the state by an MC matrix, i.e, each column c_i of the state is replaced by the new column $c'_i = (MC \times c_i)$. This multiplication is actually performed interpreting the state bytes as polynomials of a Finite Field, instead of their values as integer numbers.

The MC matrix is the following:

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

AddRoundKey. Modulo 2 addition — which is equivalent to a logical XOR — of each state bit to each corresponding bit of the current round subkey.

While DES operates on bits as its basic data unit, AES operates on bytes, and on the algebraic structure that arises from interpreting them as polynomials of a Finite Field. While all DES rounds are the same, the first and the last round of AES are different from the intermediate rounds. Both ciphers contain substitution boxes and permutations, but DES does not contain any operation similar to the MixColumns step of AES, which is a relevant component for AES's diffusion property. According to [24], AES can achieve complete plaintext diffusion after 2 rounds. More about the design criteria behind AES can be found in [24] and [47].