Design and Analysis of Lightweight Block Ciphers: 
A Focus on the Linear Layer

Christof Beierle

Doctoral Dissertation
Faculty of Mathematics
Ruhr-Universität Bochum

December 2017
Design and Analysis of Lightweight Block Ciphers:  
A Focus on the Linear Layer

vorgelegt von  
Christof Beierle

Dissertation  
zur Erlangung des Doktorgrades  
der Naturwissenschaften  
an der Fakultät für Mathematik  
der Ruhr-Universität Bochum  

Dezember 2017
First reviewer: Prof. Dr. Gregor Leander
Second reviewer: Prof. Dr. Alexander May
Date of oral examination: February 9, 2018
Abstract

Lots of cryptographic schemes are based on block ciphers. Formally, a block cipher can be defined as a family of permutations on a finite binary vector space. A majority of modern constructions is based on the alternation of a nonlinear and a linear operation. The scope of this work is to study the linear operation with regard to optimized efficiency and necessary security requirements. Our main topics are

- the problem of efficiently implementing multiplication with fixed elements in finite fields of characteristic two.
- a method for finding optimal alternatives for the ShiftRows operation in AES-like ciphers.
- the tweakable block ciphers Skinny and Mantis.
- the effect of the choice of the linear operation and the round constants with regard to the resistance against invariant attacks.
- the derivation of a security argument for the block cipher Simon that does not rely on computer-aided methods.
Zusammenfassung


- das Problem der effizienten Implementierung der Multiplikation mit festen Elementen in endlichen Körpern der Charakteristik zwei.
- eine Methode für das Finden optimaler Alternativen für die ShiftRows Operation in AES-ähnlichen Chiffren.
- die veränderbaren Blockchiffren Skinny und Mantis.
- die Auswirkung der Wahl der linearen Abbildung und der Rundenkonstanten auf die Resistenz gegen Invariantangriffe.
- das Herleiten eines nicht auf Computerberechnungen beruhenden Sicherheitsargumentes für die Blockchiffre Simon.
Acknowledgements

First and foremost I would like to thank my advisor Gregor Leander for accepting me as his student and for giving me the opportunity to work in his group during the last three years. He offered plenty of time for discussions and pointing to interesting research questions. I would also like to thank Alexander May for agreeing to be the second reviewer of this thesis.

My work was funded by DFG Research Training Group GRK 1817. Special thanks also to them.

Further, I am very grateful to all of my co-authors with whom I collaborated during the three years of my Ph.D. studies and to all other people I worked with. I would especially like to thank Roberto Avanzi, Anne Canteaut, Gottfried Herold, Takanori Isobe and Thomas Peyrin for some valuable discussions and comments on particular topics discussed in this thesis.

I would like to thank Anne Canteaut for welcoming me for seven weeks in her group at INRIA, Paris and Christian Rechberger for inviting me to visit his group at DTU, Copenhagen for one week.

I would also like to express my gratitude to the Embedded Security Group of the Faculty of Electrical Engineering and Information Technology for hosting me for the first months as a Ph.D. student and for several weeks during my interdisciplinary project.

Further thanks go to my colleagues Thorsten Kranz and Friedrich Wiemer with whom I shared an office. Especially thanks to Friedrich for helping me several times with Linux issues and using the C3 cluster, and for proofreading parts of this thesis.

Special thanks also to Irmgard and Marion for helping with lots of administrative tasks.

I would like to express my deepest gratitude to all of my family for their support and guidance. Especially, I thank my parents Marion and Klaus and my grandparents Helga and Hermann, from whom I have learned so much. Finally, I thank Anja for all her love and support.

Bochum, February 2018
Contents

List of Figures xi
List of Tables xv
Notations xvii

1 Introduction 1

2 State of the Art in Block Cipher Design 5
  2.1 Security Notions ........................................ 6
  2.2 Block Cipher Constructions .............................. 11
    2.2.1 Key-Alternating Ciphers ............................ 13
    2.2.2 Substitution-Permutation Ciphers ................. 14
    2.2.3 Feistel and ARX .................................... 15
  2.3 Cryptanalytic Attacks .................................. 18
    2.3.1 Differential Cryptanalysis ......................... 18
    2.3.2 Linear Cryptanalysis ................................ 24
  2.4 The Wide-Trail Strategy and AES-like Ciphers ....... 30
    2.4.1 The Branch Number and a Link to Coding Theory . 32
    2.4.2 AES-like Ciphers .................................. 34
    2.4.3 Computing Active S-boxes with Automatic Tools .. 39
  2.5 Lightweight Cryptography .............................. 41
    2.5.1 Lightweight Metrics ................................ 42
    2.5.2 Characteristics of Lightweight Block Ciphers .... 43
    2.5.3 Midori ............................................. 45

I Design of Lightweight Linear Layers 49

3 Lightweight Linear Layers based on Finite Field Multiplications 51
  3.1 Introduction ............................................ 51
  3.2 Preliminaries ........................................... 56
    3.2.1 The XOR-Count and the Cycle Normal Form ....... 57
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>A product cipher</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>A key-alternating cipher</td>
<td>13</td>
</tr>
<tr>
<td>2.3</td>
<td>A substitution-permutation (SP) cipher</td>
<td>14</td>
</tr>
<tr>
<td>2.4</td>
<td>One round $F^i$ of a Feistel cipher</td>
<td>15</td>
</tr>
<tr>
<td>2.5</td>
<td>One round $F^j$ of a key-alternating Feistel cipher</td>
<td>15</td>
</tr>
<tr>
<td>2.6</td>
<td>The structure of a key-alternating cipher with an AES-like round</td>
<td>35</td>
</tr>
<tr>
<td>4.1</td>
<td>An AES-like cipher for equivalent permutations</td>
<td>83</td>
</tr>
<tr>
<td>4.2</td>
<td>The possible transition patterns of the MixColumn matrix in Midori</td>
<td>90</td>
</tr>
<tr>
<td>5.1</td>
<td>Illustration of Mantis$_6$</td>
<td>119</td>
</tr>
<tr>
<td>6.1</td>
<td>The highest possible dimensions of $W_L(c_1,\ldots,c_t)$ for Skinny-64, Prince and Mantis</td>
<td>134</td>
</tr>
<tr>
<td>6.2</td>
<td>The probability that $W_L(c_1,\ldots,c_t) = F^n_3$ for uniformly random constants $c_i$ for several lightweight ciphers</td>
<td>134</td>
</tr>
<tr>
<td>7.1</td>
<td>Comparison of the experimental bounds for Simon-32 and Simon-48 from the literature and our provable bounds</td>
<td>164</td>
</tr>
<tr>
<td>7.2</td>
<td>Illustration of the Simon round function and the generalized Simon-like round function</td>
<td>167</td>
</tr>
<tr>
<td>7.3</td>
<td>Propagation of the Hamming weight for differences $(\alpha,0)$ with $w_1(\alpha) \in {1,2,3}$</td>
<td>171</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The S-box $S_{AES}$ used in the AES</td>
<td>36</td>
</tr>
<tr>
<td>2.2</td>
<td>The 4-bit S-box $S_{\text{Mid64}}$ used in Midori-64</td>
<td>46</td>
</tr>
<tr>
<td>3.1</td>
<td>Optimal instantiations of the generic MDS matrices for $2 \leq n \leq 8$.</td>
<td>71</td>
</tr>
<tr>
<td>3.2</td>
<td>Comparison of our results with $\mathbb{F}_2^n$-linear MDS matrices from the literature by average overhead per row</td>
<td>72</td>
</tr>
<tr>
<td>3.3</td>
<td>Minimal XOR-counts for all elements in $\mathbb{F}_2^4$.</td>
<td>74</td>
</tr>
<tr>
<td>3.4</td>
<td>Minimal XOR-counts for all elements in $\mathbb{F}_2^5$.</td>
<td>75</td>
</tr>
<tr>
<td>3.5</td>
<td>Minimal XOR-counts for all elements in $\mathbb{F}_2^6$.</td>
<td>76</td>
</tr>
<tr>
<td>3.6</td>
<td>Minimal XOR-counts for all elements in $\mathbb{F}_2^7$.</td>
<td>77</td>
</tr>
<tr>
<td>3.7</td>
<td>Minimal XOR-counts for all elements in $\mathbb{F}_2^8$.</td>
<td>78</td>
</tr>
<tr>
<td>3.8</td>
<td>Matrices of the form $C_{x^2+1} + E_{[i_1,j_1]} + E_{[i_2,j_2]}$ with irreducible characteristic pentanomial</td>
<td>78</td>
</tr>
<tr>
<td>4.1</td>
<td>Some classes of permutations that, under the MixM operation of Midori, lead to optimal bounds on the number of active S-boxes</td>
<td>90</td>
</tr>
<tr>
<td>5.1</td>
<td>Number of rounds $t$ for Skinny-$n$-$\kappa$</td>
<td>103</td>
</tr>
<tr>
<td>5.2</td>
<td>The 4-bit S-box $S_{b_4}$ employed in the Skinny-64 versions</td>
<td>103</td>
</tr>
<tr>
<td>5.3</td>
<td>The S-box $S_{b_8}$ used in the Skinny-128 versions</td>
<td>104</td>
</tr>
<tr>
<td>5.4</td>
<td>The Round Constants used in Skinny</td>
<td>107</td>
</tr>
<tr>
<td>5.5</td>
<td>Round-based ASIC implementations of the Skinny-64 and the Skinny-128 versions and comparison to Simon</td>
<td>110</td>
</tr>
<tr>
<td>5.6</td>
<td>Lower bounds on the minimum number of active S-boxes in Skinny</td>
<td>113</td>
</tr>
<tr>
<td>5.7</td>
<td>Number of rounds of Skinny that are broken by the best key-recovery attacks so far</td>
<td>118</td>
</tr>
<tr>
<td>5.8</td>
<td>Lower bounds on the number of active S-boxes in the single-key model and in the related-tweak model for Mantis</td>
<td>124</td>
</tr>
<tr>
<td>5.9</td>
<td>Unrolled implementations of several Mantis versions constrained for the smallest area</td>
<td>125</td>
</tr>
<tr>
<td>5.10</td>
<td>Unrolled implementations of Mantis constrained for shortest delay</td>
<td>125</td>
</tr>
<tr>
<td>5.11</td>
<td>Number of rounds $t$ needed for bounding the value of $P(C)$ below $2^{-n}$ for all versions of Simon and Simeck</td>
<td>175</td>
</tr>
</tbody>
</table>
Notations

\( N \) \quad The set of natural numbers, i.e., \( \{1, 2, \ldots\} \)

\( N_{\leq l} \) \quad The set \( \{1, \ldots, l\} \subset \mathbb{N} \)

\( N^0_{\leq l} \) \quad The set \( \{0, \ldots, l - 1\} \subset \mathbb{Z} \)

\( \mathbb{F}_2 \) \quad The finite field with two elements, i.e., \( \{0, 1\} \)

\( \mathbb{F}_2^s \) \quad The binary extension field with \( 2^s \) elements

\( \text{id}, \text{id}_m \) \quad The identity function (resp. the identity from \( \mathbb{F}_2^m \to \mathbb{F}_2^m \))

\( K^* \) \quad The multiplicative group of a field \( K \)

\( K[X] \) \quad The polynomial ring in \( X \) over the field \( K \)

\( \text{Mat}_m(K), \text{Mat}_m(R) \) \quad The ring of \( m \times m \) matrices with coefficients in the field \( K \), resp. ring with unity \( R \)

\( \text{GL}_m(K) \) \quad The general linear group of degree \( m \) over the field \( K \)

\( M^T \) \quad The transpose of the matrix \( M \)

\( 0_m \) \quad The \( m \times m \) zero matrix

\( I_m \) \quad The \( m \times m \) identity matrix

\( M_{i,j} \) \quad The coefficient of the matrix \( M \) in the \( i \)-th row of the \( j \)-th column

\( E[i,j] \) \quad A matrix having the single non-zero coefficient \( E_{i,j} = 1 \)

\( C_Q \) \quad The companion matrix of a polynomial \( Q \)

\( m_\alpha, m_A, m_L \) \quad The minimal polynomial of a finite field element \( \alpha \) or the minimal polynomial of an invertible matrix \( A \) (resp. linear bijection \( L \))

\( \bigoplus_{i=1}^d A_i \) \quad The block-diagonal matrix consisting of the \( d \) matrix blocks \( A_i \)

\( \bigoplus_{i=1}^d V_i \) \quad The direct sum of vector spaces \( V_i \)

\( \text{dim } V \) \quad The dimension of the vector space \( V \)

\( \text{lcm}(Q_1, \ldots, Q_l) \) \quad The least common multiple of \( Q_1, \ldots, Q_l \in K[X] \)

\( \text{gcd}(Q_1, \ldots, Q_l) \) \quad The greatest common divisor of \( Q_1, \ldots, Q_l \in K[X] \)
**det(A)**  The determinant of a matrix $A$

**w(A)**  The number of non-zero coefficients of a matrix $A$ or the number of non-zero coefficients of a polynomial $A$

**Quot($\mathbb{F}_2[X]$)**  The fraction field of the polynomial ring $\mathbb{F}_2[X]$

**$S_m$**  The symmetric group of degree $m$, i.e., all permutations on \{1, \ldots, m\}

**$B_m$**  The set of all Boolean functions of $m$ variables, i.e., all functions $f: \mathbb{F}_2^m \rightarrow \mathbb{F}_2$

**0, 1**  The constant Boolean functions $x \mapsto 0$ (resp. $x \mapsto 1$)

**deg $g$, deg $Q$**  The algebraic degree of a Boolean function $g$ or the degree of a polynomial $Q$

**$\Delta_\alpha f$**  The derivative of the Boolean function $f$ in direction $\alpha$, i.e., $x \mapsto f(x + \alpha) + f(x)$

**span\{ $x_i$\}_{i \in I}**  The linear span of the vectors $x_i$ for $i \in I$

**$\langle \alpha, x \rangle$**  The dot product of vectors $\alpha, x \in \mathbb{F}_2^m$, i.e., $\sum_{i=1}^m \alpha_i x_i$

**$E$**  A block cipher

**$E_k$**  A keyed instance of the block cipher $E$

**$R_i$**  The $i$-th round of a product cipher

**$R_i$**  The $i$-th unkeyed round function of a key-alternating cipher

**$\cdot \land \cdot$**  The bit-wise AND operation in $\mathbb{F}_2^m$, i.e.,

$$(x_1, \ldots, x_m) \land (y_1, \ldots, y_m) := (x_1 y_1, \ldots, x_m y_m)$$

**$\cdot \lor \cdot$**  The bit-wise OR operation in $\mathbb{F}_2^m$, i.e.,

$$(x_1, \ldots, x_m) \lor (y_1, \ldots, y_m) := (x_1, \ldots, x_m) \land (y_1, \ldots, y_m)$$

**$\lll r$**  A cyclic $r$-bit rotation to the left, i.e., for $(x_1, \ldots, x_m) \in \mathbb{F}_2^m$,

$$(x_1, \ldots, x_m) \lll r := (x_{r+1}, \ldots, x_m, x_1, \ldots, x_r)$$

**$\ggg r$**  A cyclic $r$-bit rotation to the right, i.e., for $(x_1, \ldots, x_m) \in \mathbb{F}_2^m$,

$$(x_1, \ldots, x_m) \ggg r := (x_{m-r+1}, \ldots, x_m, x_1, \ldots, x_{m-r})$$

**$\gg g r$**  An $r$-bit shift to the right, i.e., for $(x_1, \ldots, x_m) \in \mathbb{F}_2^m$,

$$(x_1, \ldots, x_m) \gg r := (0, \ldots, 0, x_1, \ldots, x_{m-r})$$

**$\cdot||\cdot$**  Concatenation of vectors, i.e.,

$$(x_1, \ldots, x_i) || (y_1, \ldots, y_m) := (x_1, \ldots, x_i, y_1, \ldots, y_m)$$

**$w_1(x)$**  The Hamming weight of a vector $x = (x_1, \ldots, x_d) \in \mathbb{F}_2^d$, i.e., the number of non-zero coefficients $x_i$

**$w_s(x)$**  The weight with respect to $s$-bit words of $x = (x_1, \ldots, x_d) \in (\mathbb{F}_2^s)^d$, i.e., the number of non-zero coefficients $x_i$

**0, \ldots, 9, A, \ldots, F**  Digits in base-16 representation (denoted in typewriter font), also used for representing binary vectors, e.g., $C := (1, 1, 0, 0)$
Chapter 1

Introduction

Due to the increasing deployment of connected devices, a huge amount of data is circulating in the “Internet of Things” (IoT). Lots of this data contains sensitive, personal information. Consequently, it has to be protected from the access to third parties. Many of the desired security goals, including confidentiality and authenticity as among the most important, can be realized by symmetric cryptographic algorithms, i.e., block ciphers, stream ciphers, hash functions, and message authentication codes. In the focus of this thesis are block ciphers which can be seen as the cornerstones of symmetric cryptography. Indeed, many other cryptographic primitives and even more complex protocols can be realized with them. For instance, if we already have a secure block cipher at hand, there are well-known constructions for building secure hash functions or message authentication codes. Consequently, one needs a secure cipher to start with.

Formally, a block cipher is defined as a family of permutations on a finite message space parametrized by a key from a finite key space. Ideally, each of those permutations should be indistinguishable from a permutation chosen uniformly at random from the set of all permutations on the message space. As this ideal security goal is quite hard to capture, a rather practical “security notion” has been established, i.e., a cryptographic primitive is assumed to be secure if no significant weaknesses have been found over a sufficient long period of times (e.g., a few years). Nowadays, we have quite efficient and versatile block ciphers of that kind. The most important one is the Advanced Encryption Standard (AES), standardized in 2001 [PUB01]. It can certainly be considered to be the best-understood construction today and since its publication, no significant weaknesses have been found. Its rather simple and elegant design and its versatility for many applications makes it the state-of-the-art cipher nowadays.

However, as the variety of possible applications increases and connected devices become smaller and cheaper, there might be situations in which a cryptographic solution tailored for meeting extremely constrained requirements on performance and efficiency is needed. In this context, one can think of the requirements of hav-
ing extremely low response time, low energy consumption, occupying few chip area, or having compact code size when implemented in software. Those requirements stand even more in focus in cases where offering security is not the main purpose of the application. Therefore, one might need new kind of designs, optimized with regard to various lightweight metrics. Over the years, plenty of those lightweight primitives appeared, see e.g., BP17 for a comprehensive list. Many of those block cipher designs follow the structure of a key-alternating substitution-permutation (SP) cipher. In those structures, the primitive is designed as an iteration of pre-defined round functions which are interleaved by the addition of round keys. The round functions are defined in a simple way, consisting of a non-linear operation and a linear operation, also called linear layer. It is a well-studied design paradigm to which also the AES belongs. In fact, in many designs, a well-known algorithm (e.g., the AES) is taken and its components are modified to meet the required lightweight requirements.

The purpose of this thesis is to broaden the knowledge of lightweight block cipher design. In particular, we study the design of the linear layer in key-alternating ciphers from the viewpoint of optimizing efficiency without sacrificing necessary security properties. When we talk about security, we mean security against well-known attack methods, the most important being differential and linear attacks. Also, as lightweight designs tend to employ sparser components and sometimes rather innovative design considerations, another important part is to avoid the applicability of new, dedicated attack methods.

At the core of our considerations is the well-known wide-trail strategy, introduced in Dae95. It suggests that spending some amount of computational effort into a well-chosen linear layer helps for obtaining strong arguments on the security of the primitive. In this light, there still exist various possibilities for optimization.

Outline of this Thesis

In Chapter 2 we start with explaining the state of the art in block cipher design, focussing on the most important topics needed for our considerations. We first give an intuitive (and then more formal) notion of security and then focus on the most common constructions for block ciphers today. After that, the two most important attack strategies, i.e., differential and linear attacks, are explained in more detail. We then continue by explaining the wide-trail design strategy, a powerful method for deriving strong security arguments on the resistance against those attacks. The AES, which is designed according to the wide-trail strategy, is presented in this context. We conclude the chapter by explaining various lightweight metrics and typical characteristics of existing lightweight designs. We emphasize that this chapter is merely for introductory purposes and we do not claim any new results. When omitting some of the details, we point to the important literature.

The rest of this thesis is arranged in two parts. The first part sets the focus slightly more on the design of linear layers, whereas the second part is more dedicated to the study of certain cryptanalytic attacks and the analysis of existing
lightweight designs. The first part consists of Chapters 3–5.

Chapter 3 studies the problem of efficiently implementing multiplication with fixed elements in finite fields of characteristic two. In particular, we study how choosing an appropriate \( \mathbb{F}_2 \)-basis can reduce the number of XOR operations (i.e., additions in \( \mathbb{F}_2 \)) needed for computing the multiplication. Those results are then used for constructing MDS matrices which need a low number of XOR operations for their implementation. As the linear layer of a block cipher is defined for vector spaces over a finite field of characteristic two, our results shed some light on how a particular choice of field representation can influence the efficiency of the linear layer, and hence the efficiency of the primitive as a whole. This chapter is based on joint work with Thorsten Kranz and Gregor Leander, published in the proceedings of CRYPTO 2016 [BKL16].

Chapter 4 focusses on the linear layer in lightweight AES-like ciphers, i.e., lightweight ciphers that resemble the round function employed in the AES. More precisely, we study how the search for an optimal word permutation, to be used as an alternative for the AES ShiftRows operation, can be conducted in an exhaustive manner, i.e., without relying on a reduction of the search space. We give an algorithm that enumerates all word permutations up to some reasonable notion of equivalence. The lightweight block cipher Midori then serves as a case study for finding the best word permutation in terms of resistance against differential and linear attacks. This cipher is particularly interesting as the designers based their search on heuristic conditions. Indeed, there exists an alternative word permutation that slightly outperforms the original choice. This chapter is based on joint work with Gianira Alfarano, Stefan Kölbl and Gregor Leander.

Chapter 5 presents the two families of lightweight tweakable block ciphers Skinny and Mantis. They were designed for having competitive performance in various environments while at the same time providing strong arguments on the resistance against related-key attacks. Those attacks, which were already introduced in the ’90s, assume a more powerful adversary that is able to encrypt (resp. decrypt) messages under several keys that are related to the key under usage. As a tweakable block cipher can be defined as a block cipher for which parts of the key are assumed to be public, this adversary model is of significant importance. Before we turn into describing the particular designs, we recall the formal notion of the related-key adversary model and tweakable block ciphers. Along with describing the design of Skinny and Mantis and their design rationale, we then explain in more detail how the search for the choice of the linear layer was conducted and how its security was evaluated using Mixed-Integer Linear Programming. The results in this chapter are based on joint work with Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich and Siang Meng Sim, published in the proceedings of CRYPTO 2016 [BJK+16a].

The last two chapters build the second part of this thesis. In Chapter 6, we focus on invariant attacks on lightweight ciphers. Those attacks were introduced in [LAAZ11] in the context of invariant subspaces, and in [TLS16] as the more general nonlinear invariant attack. Those attacks allow to easily distinguish the
cipher from a random permutation for a particular class of weak keys, by exploiting approximations by (nonlinear) Boolean functions. After explaining the attack, we present a link to linear cryptanalysis. In particular, in some cases, the existence of an invariant attack for a particular weak key implies the existence of a high-biased linear approximation over the keyed instance. Then, for SP ciphers with a simple derivation of the round keys, the question we study is how a particular choice of the linear layer, together with a particular choice of round constants, affects the applicability of the attack. We show that the lightweight block ciphers Skinny-64-64, Prince and Mantis\textsubscript{7} (with tweak zero) are resistant against a large class of invariant attacks, including all the practical attacks we are aware of from the literature. In this context, we also explain a straightforward approach how a designer can protect against those class of invariant attacks by carefully choosing the round constants along with the linear layer. Large parts of this chapter are based on joint work with Anne Canteaut, Gregor Leander and Yann Rotella, published in the proceedings of CRYPTO 2017 \cite{BCLR17}.

Chapter \textit{7} focuses on the Simon family of lightweight block ciphers, an innovative design presented by the US National Security Agency. Almost all of the security analysis was conducted and published by third-party researchers, in most cases under the usage of computer-aided tools. We outline a more theoretic security argument against differential attacks that is verifiable by hand. For that, the round function of Simon (and more general Simon-like designs) is considered as a separation into a non-linear and a linear operation. The applicability of our security argument can then be formulated as a property of this linear operation. The chapter is based on the author’s publication that appeared in the proceedings of SCN 2016 \cite{Bei16}.

We conclude each of the Chapters 3–7 by pointing to possible considerations for future work.

\textbf{On the Author’s Contribution}

Large parts of this thesis are based on joint work with other co-authors. Therefore, we start each chapter that contains joint work by mentioning the particular author’s contribution to the results.
Chapter 2

State of the Art in Block Cipher Design

We start with the formal definition of a block cipher as a family of permutations on a finite message space \( \mathcal{M} \) indexed by a key from a finite key space \( \mathcal{K} \). As modern ciphers are implemented on a computer, we will without loss of generality encode messages and keys by binary vectors of fixed length \( n \) and \( \kappa \), respectively.

**Definition 2.1** (Block Cipher). Let \( n, \kappa \in \mathbb{N} \). An \( (n, \kappa) \)-block cipher is a function

\[
E : \mathbb{F}_2^n \times \mathbb{F}_2^\kappa \to \mathbb{F}_2^n
\]

with the property that, for each \( k \in \mathbb{F}_2^\kappa \), the projection \( E_k := E(\cdot, k) \) is a permutation on \( \mathbb{F}_2^n \). Thereby, \( k \) is called the key of the keyed instance \( E_k \).

Given an \( (n, \kappa) \)-block cipher \( E \), we refer to the parameter \( n \) as its block length and to \( \kappa \) as its key length. Whenever the parameters are clear from the context, we simply refer to \( E \) as a block cipher. Given the values \( x \in \mathbb{F}_2^n \) and \( y = E_k(x) \), we call \( x \) the message or plaintext and \( y \) the ciphertext corresponding to encryption under \( E_k \). A ciphertext \( y \in \mathbb{F}_2^n \) can be decrypted to the corresponding plaintext \( x \) via \( x = \overline{E_k}^{-1}(y) \).

Typical values for the block length in modern ciphers are 64, 128 or 256. To encrypt messages of arbitrary length, one can employ the block cipher in a so-called mode of operation. This allows to extend the block cipher \( E \) to a family of permutations \( \tilde{E} \) on \( \mathbb{F}_2^n \ast \), i.e., the set of binary strings of any finite length that is a multiple of \( n \). As an example, we mention the counter mode (CTR mode), which is included in the list of recommendations from the US National Institute of Standards and Technology (NIST) [Dwo01]. In the CTR mode, a message \( (x_1, \ldots, x_m) \in \mathbb{F}_2^n \ast \), with \( x_i \in \mathbb{F}_2^n \), is encrypted under the key \( k \in \mathbb{F}_2^\kappa \) via

\[
(x_1, \ldots, x_m) \xrightarrow{\tilde{E}_k} (x_1 + E_k(c_1), x_2 + E_k(c_2), \ldots, x_m + E_k(c_m)) \quad (2.1)
\]
for some pre-defined counter values \( c_i \in \mathbb{F}_2^n \). For instance, \( c_1 \) can be chosen as the encoding of a natural number \( l_1 < 2^n \) and, for \( i > 1 \), \( c_i \) as the encoding of \( l_{i-1} + 1 \mod 2^n \). In this thesis, we only study block ciphers and do not focus on their operation modes. Therefore, we do not go into further detail here. More examples on the possible usage of block ciphers can be found, e.g., in [KR11] Section 4.

It is worth remarking that it must be possible to evaluate the block cipher \( E \) in an efficient way. An efficient evaluation is crucial as the cipher actually has to be computed in practice. As an example, one could imagine a random \((128,128)\)-block cipher which is simply defined by its look-up table. However, this look-up table would contain so much information (i.e., its description would be \( 2^{128} \cdot 2^{128} \cdot 128 \) bits) that it is not possible to store it and thus to evaluate the cipher in practice. Therefore, one has to rely on simpler constructions of block ciphers. Typical constructions of modern block ciphers are shown in Section 2.2.

First, we explain what we require from a secure cipher.

### 2.1 Security Notions

One of the main purposes of a block cipher is to protect confidential data from the eyes of a third party, called the adversary. Therefore, it is necessary to first define some notion of security and to precise against what kind of adversaries one has to protect. Before we actually provide a formal definition, we give an informal intuition in the following.

Without explaining what security actually means, we require that the security of a block cipher \( E \) (or better said the security of a keyed instance \( E_k \)) should solely rely on the secrecy of the key \( k \) and not on the secrecy of the definition of \( E \). In particular, we assume that the description of \( E \) is publicly available and, given \( k \) and \( x \), everyone can efficiently evaluate \( E_k(x) \) or its inverse \( E_k^{-1}(x) \). This goes back to one of the postulates of Auguste Kerkhoff from 1883, which is nowadays better known as Kerkhoff’s principle.\(^1\)

---

**Kerkhoff’s Principle** [Ker83]. The security of a cryptosystem should not rely on its secrecy. It should stay secure, even if it is in the hands of the adversary.

---

One of the main benefits of sticking to this principle is that, due to the public knowledge of \( E \), the security of the block cipher can be analyzed by experts from all over the world and possible weaknesses are more likely to be revealed. Moreover, (short) keys are easier to keep secret than whole algorithms.

But what do we actually require from a secure cipher? Obviously, we do not want that, given some plaintext/ciphertext data \((x_i, E_k(x_i))_i\), an adversary is able to reveal the secret key \( k \). If the key would be in the hands of the adversary, it

\(^1\)The postulate was originally stated in French. This is a loose translation.
could simply decrypt all encrypted messages it intercepts. Therefore, protection against those so-called \textit{key-recovery attacks} is a necessary requirement. However, this is not sufficient. Just consider the trivial block cipher $E$ for which $E_k = \text{id}$ for all keys $k$. Clearly, it is not possible to reveal any information on the actual key as it is simply not used for encryption. Obviously, such a cipher should not be considered secure as one immediately obtains information on the plaintext (in this example, one even obtains the whole plaintext). What we rather want to achieve is that any keyed instance is \textit{indistinguishable} from a permutation that is chosen uniformly at random. This has become the state-of-the-art notion of security and we mainly aim for protecting against those so-called \textit{distinguishing attacks}.

\textbf{The Power of the Adversary}

We further have to think about what kind of adversaries we want to protect against. In particular, one can make different assumptions on the kind of data that is in the hands of the adversary.

In a \textit{Ciphertext-Only Attack (COA)}, the adversary has only knowledge of a single (or multiple) ciphertext(s) $(E_k(x_i))_i$. Its aim is to gain some information on the message $x$ or the secret key $k$.

In a \textit{Known-Plaintext Attack (KPA)}, the adversary knows some pre-defined message/ciphertext pairs $(x_i, E_k(x_i))_i$.

In a \textit{Chosen-Plaintext Attack (CPA)}, the adversary is able to choose messages $x_i$ on its own and has knowledge of $(x_i, E_k(x_i))_i$, i.e., it is able to encrypt any chosen message.

In a \textit{Chosen-Ciphertext Attack (CCA)}, the adversary has knowledge of tuples $(x_i, E_k(x_i))_i$ for either chosen plaintexts $x_i$ or chosen ciphertexts $E_k(x_i)$, i.e., it is able to encrypt and decrypt messages of its choice.

As a designer of a cipher, it is reasonable to be rather conservative. In particular, one usually aims for protecting against the last two scenarios (CPA and CCA) as they assume the most powerful adversaries.

\textbf{A Formal Security Definition}

We have already given some intuition on the requirements of a secure block cipher. In this section, we would like to outline – in a more formal way – what it actually means for a block cipher to be secure against CPA and CCA attacks. For this, we use the notion of a \textit{pseudorandom permutation} as described in, e.g., [BR05, Section 4]. Note that in the whole of this chapter, we only consider the \textit{single-key} adversary model. The more involved \textit{related-key} model is considered later in Chapter 5.

As we want to protect against distinguishing attacks, we require from a secure cipher that once the key is fixed and kept secret, an adversary cannot \textit{efficiently distinguish} the keyed instance from a permutation that is chosen uniformly at random. Here, the restriction to efficient adversaries is crucial. As an example for a non-efficient adversary, one could consider an algorithm that simply tries
all possible keys $k$ and computes $E_k(x)$ for all $x$. Of course, the adversary could distinguish a keyed instance from a random permutation as it simply computed the complete look-up table for $E_k$ (see also Example 2.1). Therefore, we naturally have to restrict the computational resources of the adversary and we only consider computationally-bounded algorithms as possible adversaries.

Formally, a $(q,t)$-adversary is defined as a (probabilistic) algorithm $A_{q,t}$ with running time at most $t$ that is allowed to make at most $q$ oracle queries to given functions $O_1, \ldots, O_l$. Thereby, $q$ indicates the total number of allowed queries. After termination, the algorithm outputs a bit $b \in \{0, 1\}$. We write $A^O_{q,t} \Rightarrow b$ to denote the event that the adversary $A_{q,t}$ outputs $b$, given oracle access to $O_1, \ldots, O_l$. In this context, oracle access to some function $O$ means that the adversary has no further knowledge about how the function is defined, except that it learns the output $O(x)$ whenever it requests the function value of $x$.

Now, suppose we are given a block cipher $E: \mathbb{F}_2^n \times \mathbb{F}_2^\kappa \rightarrow \mathbb{F}_2^n$. We consider an adversary $A_{q,t}$ whose goal is to distinguish a keyed instance $E_k$ from a permutation chosen uniformly at random from the class of all permutations on $\mathbb{F}_2^n$. The adversary will either interact with the keyed instance $E_k$ or with a random permutation (as oracles) and has to decide (by outputting the bit $b$) with which particular permutation it interacted. In this model, the (CPA)-advantage of the adversary $A_{q,t}$ is formally defined as

$$\text{Adv}_{E}^{\text{PRP-CPA}}(A_{q,t}) := \text{Prob}_{k \overset{\$}{\leftarrow} \mathbb{F}_2^\kappa} (A_{q,t}^{E_k} \Rightarrow 1) - \text{Prob}_{\pi \overset{\$}{\leftarrow} \text{Perm}_{n}} (A_{q,t}^{\pi} \Rightarrow 1).$$

Thereby, we denote by $k \overset{\$}{\leftarrow} \mathbb{F}_2^\kappa$ that the key $k$ is chosen uniformly at random from the set $\mathbb{F}_2^\kappa$ of possible keys, and by $\pi \overset{\$}{\leftarrow} \text{Perm}_{n}$ that the permutation $\pi$ is chosen uniformly at random from the set of all permutations on $\mathbb{F}_2^n$. Thus, the probabilities are defined over the uniform choices of $k$ and $\pi$ and over the random choices that the probabilistic adversary $A_{q,t}$ makes.

One can further allow oracle access to the inverse permutation as well in order to equip the adversary with more power. Then, the (CCA)-advantage of $A_{q,t}$ can be formally defined as

$$\text{Adv}_{E}^{\text{PRP-CCA}}(A_{q,t}) := \text{Prob}_{k \overset{\$}{\leftarrow} \mathbb{F}_2^\kappa} (A_{q,t}^{E_k,E_k^{-1}} \Rightarrow 1) - \text{Prob}_{\pi \overset{\$}{\leftarrow} \text{Perm}_{n}} (A_{q,t}^{\pi,\pi^{-1}} \Rightarrow 1).$$

These notions can be considered as a measurement on how well a particular adversary is able to “break” the cipher. In the following, we give an example of an adversary that adheres to the very generic brute-force strategy.

**Example 2.1 (Brute-Force Attack).** We consider the family of adversaries $BF_{q,t}$, that is given oracle access to an $n$-bit permutation $O$, as described in Algorithm 2.1. The constant $T_E$ denotes the time to evaluate one block cipher call $E_k(x)$ depending on the machine on which the algorithm runs. For simplicity, we neglect the time for the other operations executed in Algorithm 2.1 and thus basically measure the
time with regard to the number of block cipher calls. One can give a lower bound on the advantage of the adversary $BF_{k,t}$. In particular, we have

$$\text{Prob}_{\mathbf{k} \sim F_2^n} \left( BF_{E_k}^{B\mathcal{F}_{q,t}} \Rightarrow 1 \right) \geq \frac{\lfloor \frac{t}{q T} \rfloor}{2^\kappa}$$

and

$$\text{Prob}_{\mathbf{\pi} \sim \text{Perm}_n} \left( BF_{\pi}^{B\mathcal{F}_{q,t}} \Rightarrow 1 \right) \leq \text{Prob} (\exists k' : \forall i \in \{1, \ldots, q\} : y_i = E_{k'}(x_i)) \leq \frac{2^\kappa}{2^n (2^n - 1) \ldots (2^n - q + 1)}.$$

This immediately leads to a lower bound for the advantage as

$$\text{Adv}_{E}^{\text{PRP-CPA}}(B\mathcal{F}_{q,t}) \geq \left\lfloor \frac{t}{q T} \right\rfloor 2^{-\kappa} - \frac{2^\kappa}{2^n (2^n - 1) \ldots (2^n - q + 1)}.$$

The second term is the only one which is dependent on the block length $n$ and exponentially tends to zero as the number of oracle queries $q$ increases. Thus, a very small value for $q$ suffices in order to approximate this term by zero. For instance, if $E$ is a (128, 64)-block cipher, we could choose $q = 1$ and $t$ such that $\left\lfloor \frac{t}{T} \right\rfloor = 2^{63}$ and obtain

$$\text{Adv}_{E}^{\text{PRP-CPA}}(B\mathcal{F}_{q,t}) \geq 49.9999\%.$$

The algorithm would therefore be at least about 50% more successful in distinguishing the cipher from a random permutation than an algorithm which simple guesses the value $b \sim \{0,1\}$.

When the block length is not larger than the key length, one would choose a slightly higher value for $q$. For instance, for a (64, 64)-block cipher, $q = 2$ would be reasonable.

The brute-force attack described above is very generic and does not exploit any particular structure of the block cipher. Moreover, it suggests that a sufficiently large key length should be used. Although the execution of $2^{63}$ block cipher calls is fairly huge, it is not completely out of reach, especially with a large supercomputer. In comparison, the three versions of the Advanced Encryption Standard (AES), which were standardized by NIST in 2001 [PUB01], provide key length of 128, 192 and 256, respectively. In its report on lightweight cryptography [MBTM17], NIST indicates that new ciphers to be standardized should provide a key length of at least $\kappa = 112$.

From a secure block cipher, we would require that there exists no adversary with a significant high advantage for reasonable restrictions on $q$ and $t$. This leads to the notion of a (strong) pseudorandom permutation as defined in the following.

---

3We do not measure the time for the oracle queries as $q$ in $B\mathcal{F}_{q,t}$ would usually be fairly low, i.e., $q \leq 3$
Algorithm 2.1 $BF_{q,t}$

1: Choose $q$ pairwise different $x_1, \ldots, x_q \overset{\$}{\leftarrow} \mathbb{F}_2^n$
2: Compute $(y_1, \ldots, y_q) \leftarrow (O(x_1), \ldots, O(x_q))$
3: $L \leftarrow \{\}$
4: for $i \in \{1, \ldots, \lfloor \frac{t}{qT_E} \rfloor\}$ do
5: \hspace{1cm} $k' \overset{\$}{\leftarrow} \mathbb{F}_2^n \setminus L$
6: \hspace{1cm} if $\forall j \in \{1, \ldots, q\}: y_j = E_{k'}(x_j)$ then
7: \hspace{2cm} return 1
8: \hspace{1cm} end if
9: \hspace{1cm} $L \leftarrow L \cup \{k'\}$
10: end for
11: return 0

Definition 2.2 ((Strong) Pseudorandom Permutation, see e.g., [BR05]). We say that the block cipher $E$ is a $(q, t, \epsilon)$-PRP (pseudorandom permutation) if
\[
\max_{\mathcal{A}_{q,t}} \text{Adv}_{E}^{\text{PRP-CPA}}(\mathcal{A}_{q,t}) \leq \epsilon.
\]
Analogously, we say $E$ is a $(q, t, \epsilon)$-SPRP (strong pseudorandom permutation) if
\[
\max_{\mathcal{A}_{q,t}} \text{Adv}_{E}^{\text{PRP-CCA}}(\mathcal{A}_{q,t}) \leq \epsilon.
\]

These notions of a pseudorandom permutation and a strong pseudorandom permutation formalize the intuition of security against chosen-plaintext and chosen-ciphertext attacks, respectively. However, these notions are only capable of defining security up to a certain threshold ($\epsilon$) and only against adversaries with bounded resources ($q, t$). Therefore, in practice one always has to evaluate what realistic assumptions on the adversary are and what the particular level of security one wants to achieve is. This of course highly depends on the particular application. As a guidance, one may achieve that there is basically no better attack than brute force.

Provable Security and Block Cipher Design

The benefit of the above formal security definition is that it provides a framework for proving security of (more complex) constructions like cryptographic protocols, modes of operations, or other other (S)PRPs by reducing to the security of the underlying (S)PRPs. In other words, under the assumption that $E$ is a (strong) pseudorandom permutation with parameters $(q, t, \epsilon)$, one tries to build other constructions that can be proven secure for certain parameters $(q', t', \epsilon')$ depending on $(q, t, \epsilon)$. Lots of research is done that mainly focuses on this kind of provable security. However, by the conceptual method of reduction, the security of complex cryptographic systems relies on the security of the underlying building blocks.
(here the underlying block cipher). Consequently, one needs a secure cipher to start with.

The scope of block cipher design – as we focus on in this thesis – can be phrased as building efficient algorithms to be used as block ciphers and providing sound arguments that justify the assumption for such an algorithm being a (strong) pseudorandom permutation. In fact, we are not aware of any block cipher that is proven to be an (S)PRP for reasonable parameters \( q, t, \epsilon \) without any reduction argument. The problem is simply that, given parameters \( q, t \), one has to consider all possible \((q,t)\)-adversaries. Rather, the state of the art is to propose a block cipher design and then analyze the resistance of the block cipher against certain, well-studied, kind of attacks. In a nutshell, one studies adversaries that adhere to specific strategies, the most important being differential and linear attacks. They are described in detail in Section 2.3.1 and Section 2.3.2, respectively.

The designers of new block ciphers usually claim security of their cipher up to certain parameters and directly provide sound arguments on the resistance against well-studied attack strategies. After the publication of the design, the scientific community further analyzes possible methods for attacking the cipher. If no significant weaknesses have been found over a certain period of time (usually a few years), it may be safe to conjecture the cipher being an (S)PRP up to certain parameters \((q,t,\epsilon)\). Then, the block cipher may be used as a building block in provable secure constructions.

As the most prominent example of a block cipher, we refer to the AES. Its specification is publicly available for over 15 years now and it can certainly be considered as the best-analyzed block cipher to date. No significant weaknesses have been spotted. The cipher will be explained in detail in Section 2.4.2.

### 2.2 Block Cipher Constructions

We have already seen an example of an obviously bad block cipher, a cipher in which the key has no influence on the encryption at all. Before we explain common constructions for block ciphers today, we give another example of a weak block cipher. For that, we consider a cipher \( E \) that splits into two (or more) smaller
ciphers $E'$ and $E''$, i.e., a message $x = (x_1, x_2)$ is encrypted as
$$E_k(x_1, x_2) = (E'_{k'}(x_1), E''_{k''}(x_2)) .$$

Such a construction should not be considered as a secure block cipher as changing only one part of the message ($x_1$ or $x_2$) will only change one part of the ciphertext. This may allow for statistical attacks in which the knowledge of the distribution of possible messages is exploited. In the seminal work of Claude Shannon from 1949 [Sha49], the author provides several design principles that a modern cipher should adhere to in order to avoid such statistical attacks. These principles were called confusion and diffusion and, indeed, they are still of major importance in modern designs.

According to Shannon, confusion means “to make the relation between the simple statistics of (the ciphertext) and the simple description of (the key) a very complex and involved one”. Further, to stick to his wording, “in the method of diffusion the statistical structure of (the message) which leads to its redundancy is ‘dissipated’ into long range statistics – i.e., into statistical structure involving long combinations of letters in the cryptogram” [Sha49, Section 23].

Obviously, we further want to avoid that a cipher is linear. If that would be the case, an adversary would just need the encryptions of a basis to be able to encrypt or decrypt arbitrary messages. Therefore, every secure block cipher needs a non-linear component. Most modern designs today realize diffusion by the application of a linear transformation, whereas a non-linear operation provides confusion. The non-linear operation is often realized as a substitution operation on parts of the message space (see Section 2.2.2).

Before we go into more detail of modern block cipher constructions, we explain the notion of a product cipher. This concept also goes back to [Sha49] and today, almost all practical block ciphers can be defined as a product cipher. The high-level structure is depicted in Figure 2.1. Thereby, the cipher $E$ is built from multiple iterations of other (more simple) block ciphers $R_i$, so-called rounds. Each round $R_i$ operates under the influence of a specific round key that is derived from the initial key for $E$ by the so-called key-scheduling algorithm. In other words, each keyed instance $E_k : F_2^n \rightarrow F_2^n$ is defined as
$$E_k = R_{t_k} \circ \cdots \circ R_{1k_1} \circ R_{0k_0},$$
for previously-defined block ciphers $R_i : F_2^n \times F_2^{k_i} \rightarrow F_2^n$ and round keys $k_i$ which are derived by a function $\text{KeySchedule} : F_2^n \rightarrow F_2^{k_0} \times \cdots \times F_2^{k_t}$ as $(k_0, k_1, \ldots, k_t) = \text{KeySchedule}(k)$.

There are two main benefits of building block ciphers as product ciphers. Firstly, they tend to be more efficient to implement. If, for instance, the same round is iterated several times, it only has to be implemented once. Secondly, a product cipher can often be analyzed more easily. In fact, modern block ciphers iterate quite simple round functions several times and the security analysis is most often conducted under the assumption of independent round keys\footnote{For that reason, we omit the discussion on how to design a key schedule.} (see
2.2.1 Key-Alternating Ciphers

A special type of product cipher is the key-alternating cipher to which a majority of modern designs belong. This type of product cipher exactly describes the way the particular round keys are introduced within the rounds. The structure is depicted in Figure 2.2. The key-scheduling function has to generate round keys in $\mathbb{F}_{2^n}$, i.e., $\text{KeySchedule} : \mathbb{F}_2^n \rightarrow \mathbb{F}_{2^n+1}$, and each round $R_i$ is defined as

$$ R_i : \mathbb{F}_2^n \times \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n $$

$$(x, k_i) \mapsto \text{Add}_{k_i}(R_i(x)),$$

where $R_i$ is a bijection on $\mathbb{F}_2^n$ and $\text{Add}_{k_i}$ is defined as

$$ \text{Add}_{k_i} : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n, \quad x \mapsto x + k_i. $$

We then have that any keyed instance of $E$ can be written as

$$ E_k = \text{Add}_{k_t} \circ R_t \circ \cdots \circ \text{Add}_{k_1} \circ R_1 \circ \text{Add}_{k_0} \circ R_0. $$

Note that, according to Kerkhoff’s principle, we assume that all of the unkeyed round functions $R_i$ are publicly known. In particular, any known plaintext $x$ can be transformed by $R_0$ to $R_0(x)$ and thus, the application of $R_0$ in the beginning adds no security to the cipher at all. In practice, one therefore usually starts with the addition of $k_0$, i.e., $R_0 = \text{id}$. Without loss of generality, we can assume $R_0$ to be the identity and refer to the construction depicted in Figure 2.2 (with $R_0 = \text{id}$) as a $t$-round key-alternating cipher.

The benefit of the key-alternating structure is that, on the one hand, it allows for an easier analysis of the cipher under the assumption of independent round keys and, on the other hand, it allows for a quite efficient implementation of the

---

\footnote{Note that, in the literature, the vector addition $+$ in $\mathbb{F}_2^n$ is also often denoted by $\oplus$ as it corresponds to the bit-wise exclusive-or (XOR) operation.}
cipher as changing the particular keyed instance can be realized by simply adding other round keys. Moreover, the unkeyed round functions $R_i$ are often (almost) identical and therefore only a single round has to be implemented.

A majority of common key-alternating block ciphers follow the notion of a substitution-permutation cipher, which we explain in the following.

### 2.2.2 Substitution-Permutation Ciphers

A substitution-permutation cipher (also called SP cipher or SP network) defines a special structure of the round functions in a product cipher. Thereby, the rounds consist of the application of a non-linear operation $S$ (also called substitution layer or $S$-box layer), which is realized as a parallel application of smaller functions (so-called $S$-boxes), and the application of a linear transformation $L$ (also called linear layer). Originally, the structure of a substitution-permutation network was introduced in [FNS75], where the linear layer $L$ was defined as a permutation of bits, i.e., a to $L$ associated matrix is a permutation matrix over $\mathbb{F}_2$. Nowadays, most SP ciphers follow the notion of a key-alternating cipher as explained above and the unkeyed round functions can be decomposed into the (invertible) non-linear layer $S$ and the (invertible) linear layer $L$. The structure is depicted in Figure 2.3. Whenever we refer to SP ciphers in the remainder of this thesis, we are talking about this particular structure.

Formally, the substitution layer $S: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is defined as an $n_s$-times parallel application of an invertible S-box $S_b: \mathbb{F}_2^s \rightarrow \mathbb{F}_2^s$ such that $n = n_s \cdot s$. In other words,\footnote{In principle, one could use a different S-box at each position. However, for easier implementation, usually the same S-box is applied in parallel.} \footnote{We interchangeable identify $\mathbb{F}_2^{n_s}$ with $(\mathbb{F}_2^s)^{n_s}$.}

$$S: \mathbb{F}_2^n \cong \mathbb{F}_2^s \times \cdots \times \mathbb{F}_2^s \rightarrow \mathbb{F}_2^s \times \cdots \times \mathbb{F}_2^s \cong \mathbb{F}_2^n$$

$$(x_1, x_2, \ldots, x_{n_s}) \mapsto (S_b(x_1), S_b(x_2), \ldots, S_b(x_{n_s})).$$

Usually, $s$ is chosen to be rather small. In particular, common choices for $s$ are $s = 4$ or $s = 8$ and therefore, those S-boxes could be simply implemented by storing...
their look-up tables. Sometimes, algebraic constructions for $S_b$ are used. In that case, the S-boxes may also be computed during execution. Research has been done that focusses on building cryptographically strong S-boxes and on optimizing their efficiency, see for instance \[\text{Nyb93, Nyb94, Can05, CDL16, Sto16}\].

The linear layer $L$ can be defined by a matrix $M \in \text{GL}_n(\mathbb{F}_2)$ by fixing a particular choice of basis. Then, the application of $L$ corresponds to (left-) multiplication with $M$. Nowadays, in an SP cipher, one allows $L$ to be any invertible linear transformation that is not necessarily a permutation of bits. It was to a large extent the wide-trail strategy \[\text{Dae95}\] that pioneered the usage of a general linear layer instead of a bit permutation. Although a bit permutation may allow for an easier implementation, the wide-trail strategy suggested that the usage of a slightly more complex linear layer may allow for better trade-offs between security and efficiency. We explain the wide-trail strategy in more detail in Section \[2.4\].

To return to the principles of Shannon, the S-boxes are to a large extend responsible for obtaining (local) confusion within the cipher. Then, the linear layer should diffuse the information over the whole $n$-bit state. It is exactly this linear layer we focus on in this thesis. In particular, we study the questions of how to design the linear layer in order to improve security of the cipher and efficiency of its implementation with regard to certain lightweight metrics (as explained in Section \[2.5\]).

2.2.3 Feistel and ARX

Another important block cipher construction is the so-called Feistel cipher (or Feistel network), which was first introduced in the design of Lucifer (see \[\text{Sor84}\]), a predecessor of the Data Encryption Standard (DES). The DES was developed
in the ’70s an was published as a US FIPS standard in 1977 [PUB77]. A detailed description of the cipher can be found, e.g., in [KR11]. In particular, a Feistel cipher is another special form of product cipher in which all the rounds $R_i$ are of the form $F^i$, which we define below. To define a Feistel round, the requirement is that the block length $n$ is an even number. Let $f: \mathbb{F}_2^n \times \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be any keyed function from $\mathbb{F}_2^n \times \mathbb{F}_2^n$ to $\mathbb{F}_2^n$. It is not required that $f$ fulfills the definition of a block cipher, i.e., a keyed instance $f_k = f(\cdot, k)$ has not necessarily to be a bijection. One also calls $f$ the Feistel function. The Feistel round $F^f$ of an $(n, \kappa)$-block cipher is then defined as

$$F^f: \mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n \times \mathbb{F}_2^n$$

$$(x_l, x_r, k) \mapsto (f_k(x_l) + x_r, x_l) .$$

In other words, any input $x \in \mathbb{F}_2^n$ is split into two halves $x_l$ and $x_r$, the left half $x_l$ will be copied to the right half of the output and the left half of the output consists of the right input half $x_r$, which is XORed with $f_k(x_l)$. The Feistel round is illustrated in Figure 2.4. It is easy to see that $F^f$ fulfills the definition of an $(n, \kappa)$-block cipher. In particular, the inverse of any keyed instance $F_k^f: (x_l, x_r) \mapsto (f_k(x_l) + x_r, x_l)$ can be given as

$$F_k^{-1}^f: (x_l, x_r) \mapsto (x_r, f_k(x_r) + x_l) .$$

This already illustrates the advantages of Feistel ciphers; firstly that the Feistel function $f$ can be any keyed function and, secondly, that decryption is almost identical to encryption, resulting in a low implementation overhead.

In its general form (and also for the particular structure of the DES), Feistel ciphers are structurally different from key-alternating ciphers. However, when the Feistel function $f$ is of the special form

$$f: \mathbb{F}_2^n \times \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$$

$$(x, k) \mapsto \text{Add}_k(f(x))$$

for a public, unkeyed function $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, the resulting Feistel cipher can be considered as a key-alternating cipher in which the round keys are only added to the right halves. In particular, any keyed instance of one Feistel round can then be written as

$$F_k^f: (x_l, x_r) \mapsto (f(x_l) + x_r + k, x_l) .$$

The structure is depicted in Figure 2.5. An example of a key-alternating Feistel cipher is the lightweight cipher Simon [BSS+13].

---

8Originally, instead of copying the left half of the input and processing it through the Feistel function, the right half was copied and processed by the Feistel function. Without loss of generality, we stick to our notion of a Feistel round.
Example 2.2 (Simon). Simon is a block cipher family that was designed by the National Security Agency (NSA) and was published in 2013. It was designed for achieving exceptional good performance when implemented on a variety of platforms. It employs a very simple function \( f \) in the key-alternating Feistel setting. The cipher comes in different versions, supporting block lengths of \( n \in \{32, 48, 64, 96, 128\} \), and \( f \) can be defined for all corresponding values of \( n/2 \) as

\[
 f : \mathbb{F}_2^n \to \mathbb{F}_2^n \\
 x \mapsto (\vartheta_1(x) \land \vartheta_2(x)) + \theta(x),
\]

where \( \land \) denotes the component-wise \( \mathbb{F}_2 \)-multiplication of binary vectors and \( \vartheta_1, \vartheta_2 \) and \( \theta \) are cyclic rotations to the left by eight, one and two bits, respectively. Formally,

\[
\begin{align*}
\vartheta_1 : (x_1, x_2, \ldots, x_{n/2}) &\mapsto (x_9, x_{10}, \ldots, x_{n/2}, x_1, x_2, \ldots, x_8), \\
\vartheta_2 : (x_1, x_2, \ldots, x_{n/2}) &\mapsto (x_2, x_3, \ldots, x_{n/2}, x_1), \\
\theta : (x_1, x_2, \ldots, x_{n/2}) &\mapsto (x_3, x_4, \ldots, x_{n/2}, x_1, x_2).
\end{align*}
\]

Simon can be considered as a rather innovative design because of the simplicity of the Feistel function and because of its key schedule. Most of the security analysis of the cipher was done using computer-aided methods. We consider Simon and Simon-like designs in more detail in Chapter 7, putting the focus on deriving a more theoretical security argument.

Several generalizations of Feistel networks have been introduced, for instance unbalanced Feistel networks [SK96], i.e., Feistel constructions in which the two input parts \( x_l, x_r \) do not have to be of the same length, or generalized Feistel constructions which allow to split the input into more than two parts [ZMI90, Nyb96].

Add-Rotation-XOR (ARX) Ciphers

Some ciphers avoid the usage of smaller S-boxes for their non-linear operation. Instead, they rely on arithmetic operations applied on the state. This in particular avoids the implementation of table look-ups. For efficiency reasons, in many of such designs, three basic arithmetic operations are employed, i.e.,

(i) Addition modulo \( 2^d \),

(ii) Cyclic bit-wise rotations, and

(iii) XOR operations (component-wise \( \mathbb{F}_2 \)-addition in \( \mathbb{F}_2^d \)).

We have already seen Simon as an example of a cipher that applies cyclic rotations and XOR operations. Those two operations are \( \mathbb{F}_2 \)-linear. The non-linear operation is the addition modulo \( 2^d \). There, the \( d \)-bit vectors are regarded as integers smaller than \( 2^d \) and the addition modulo \( 2^d \) is the group operation in \( (\mathbb{Z}_{2^d}, +) \).
The benefit of modular addition is its efficiency in software implementations. Indeed, ARX ciphers can often be implemented using only few lines of code. Block ciphers belonging to the class of ARX designs include for example FEAL [SM88], TEA [WN95], Speck [BSS+13], or SPARX and LAX [DPU+16].

While the wide-trail strategy offers a powerful method for evaluating the security of S-box based ciphers, deriving general arguments on the resistance of ARX designs against the most important attack methods (differential and linear cryptanalysis) has been a long-standing open problem. Recently, Dinu et al. presented a strategy for proving resistance against differential and linear attacks by design [DPU+16].

2.3 Cryptanalytic Attacks

In this section, we explain the most important cryptanalytic attacks on block ciphers, i.e., differential and linear attacks. They have gained so much importance that every new cipher should come along with strong arguments for the resistance against them. All in all, differential and linear attacks can be seen as the cornerstones of cryptanalysis of modern block ciphers. They aim for exploiting specific properties of the cipher that allow to distinguish it from a permutation chosen uniformly at random. Especially for product ciphers, distinguishing attacks can also be seen in the context of key recovery. In particular, the adversary may guess the last round key(s) of the cipher and decrypt ciphertexts over the last round(s). If there exists a distinguishing attack on the reduced-round cipher, the adversary may now be able to validate its key guess. Thereby, it is assumed that a wrong key guess randomizes the intermediate values corresponding to the round-reduced ciphertexts that are obtained by the partial decryption under the wrong key.

The standard approach of a designer of a new (product) cipher is to prove, mostly under simplifying assumptions, that it is not possible to efficiently distinguish $t$ rounds of the proposed product cipher from a random permutation using differential and linear attacks. Then, the final cipher will be specified as a $t + t_m$-round version of the product cipher, where $t_m$ denotes a reasonable security margin. Therefore, already with explaining the attacks in the following, we mention the designer’s standard security arguments for obtaining confidence in the resistance against them.

2.3.1 Differential Cryptanalysis

The technique of differential cryptanalysis was introduced by Biham and Shamir in 1990 as an attack on round-reduced versions of DES [BS91a, BS91b]. The general idea can be phrased as analyzing how differences in the input of the cipher propagate through differences in the output. It turned out to be a powerful cryptanalytic technique and nowadays, the designers of new ciphers are expected to provide strong arguments for the resistance against differential cryptanalysis. We explain the details of this attack method in the following. Thereby, we keep the
focus on product ciphers and, especially, key-alternating ciphers as it is also done in, e.g., [DR02].

**Definition 2.3 (Differential Probability).** For a vectorial function $F: \mathbb{F}_2^k \to \mathbb{F}_2^l$ and vectors $\alpha \in \mathbb{F}_2^k$, $\beta \in \mathbb{F}_2^l$, we define

$$
\Delta F(\alpha, \beta) := \{x \in \mathbb{F}_2^k \mid F(x) + F(x + \alpha) = \beta\}
$$

The pair $(\alpha, \beta)$ is said to be a differential over $F$ (also denoted $\alpha \xrightarrow{F} \beta$) and the differential probability is defined as

$$
\text{Prob}(\alpha \xrightarrow{F} \beta) := \frac{|\Delta F(\alpha, \beta)|}{2^k}.
$$

Thus, $\text{Prob}(\alpha \xrightarrow{F} \beta)$ describes exactly the probability that $F(x) + F(x + \alpha) = \beta$ over a uniformly chosen $x \in \mathbb{F}_2^k$.

The simple idea of an adversary that adheres to differential cryptanalysis is that it is in possession of a differential $(\alpha, \beta)$ over $E_k$ that holds with high probability, i.e., $\text{Prob}(\alpha \xrightarrow{E_k} \beta) > 2^{-n}$. The adversary can now distinguish the keyed instance from a random permutation by querying the oracle $O$ for many randomly chosen input pairs $(x, x + \alpha)$ and checking whether the output difference $O(x) + O(x + \alpha)$ equals $\beta$ as often as one would expect by the differential probability. Thus, differential cryptanalysis is an example of a chosen-plaintext attack.

As the block cipher is a family of permutations parametrized by a key, we have to differentiate between the fixed-key probability of a differential $(\alpha, \beta)$ and the expected differential probability when averaged over all possible keys. Formally, for an $(n, \kappa)$-block cipher $E$, the expected differential probability of a differential $(\alpha, \beta)$ is defined as

$$
\text{EDP}_E(\alpha, \beta) := \frac{1}{2^\kappa} \sum_{k \in \mathbb{F}_2^\kappa} \text{Prob}(\alpha \xrightarrow{E_k} \beta).
$$

A priori, the adversary has no knowledge of the actual key $k$ of the keyed instance $E_k$ it wants to distinguish from a random permutation. Moreover, it may actually want to exploit a differential that holds with a high expected differential probability. Thus, for practical reasons, we implicitly assume that the fixed-key probability of a differential is to a large extent independent of the actual key used. This assumption was first formulated in [LM91] as the Hypothesis of Stochastic Equivalence.

**Assumption 2.1 (Hypothesis of Stochastic Equivalence).** Given a block cipher $E$ and a differential $(\alpha, \beta)$, then

$$
\text{EDP}_E(\alpha, \beta) \approx \text{Prob}(\alpha \xrightarrow{E_k} \beta)
$$

for "almost" all keys $k$. 

19
Thus, a designer would like to guarantee a low upper bound on the expected differential probability of any non-zero differential. In the following, we elaborate more on differential cryptanalysis of product ciphers and key-alternating cipher in particular.

**Differential Cryptanalysis of Product Ciphers**

Any keyed instance \( E_k \) of a product cipher can be written as iterations of bijective round functions as

\[
E_k = R_{t_k} \circ \cdots \circ R_{1_k} \circ R_{0_k}.
\]

For an iterative function, besides of just considering a differential over the function itself, one can moreover consider all the intermediate differences after each iteration simultaneously. This leads to the definition of a *differential trail* as follows.

**Definition 2.4** (Differential Trail, see, e.g., p. 117 in [DR02]). Let \( F: \mathbb{F}_2^n \to \mathbb{F}_2^n \) be an iterative function of the form \( F = F_t \circ \cdots \circ F_2 \circ F_1 \) with \( F_i: \mathbb{F}_2^n \to \mathbb{F}_2^n \). Given \( t + 1 \) vectors \( \alpha_0, \ldots, \alpha_t \in \mathbb{F}_2^n \), we define

\[
\Delta_{F_1, \ldots, F_t}(\alpha_0, \ldots, \alpha_t) := \{ x \in \mathbb{F}_2^n | \forall 1 \leq i \leq t: F_i \ldots F_1(x) + F_i \ldots F_1(x + \alpha_0) = \alpha_i \}.
\]

The tuple \( (\alpha_0, \ldots, \alpha_t) \in (\mathbb{F}_2^n)^{t+1} \) is said to be a differential trail over \( F \) (also denoted \( \alpha_0 \xrightarrow{F_1} \alpha_1 \xrightarrow{F_2} \ldots \xrightarrow{F_t} \alpha_t \)) and the probability of the differential trail is defined as

\[
\text{Prob}(\alpha_0 \xrightarrow{F_1} \alpha_1 \xrightarrow{F_2} \ldots \xrightarrow{F_t} \alpha_t) := \frac{|\Delta_{F_1, \ldots, F_t}(\alpha_0, \ldots, \alpha_t)|}{2^n}.
\]

Thus, \( \text{Prob}(\alpha_0 \xrightarrow{F_1} \alpha_1 \xrightarrow{F_2} \ldots \xrightarrow{F_t} \alpha_t) \) describes exactly the probability that, over a uniformly chosen \( x \in \mathbb{F}_2^n \), the intermediate difference \( F_i \ldots F_1(x) + F_i \ldots F_1(x + \alpha_0) \) is equal to \( \alpha_i \), for all \( i \in \{1, \ldots, t\} \).

It follows that for an iterative function \( F: \mathbb{F}_2^n \to \mathbb{F}_2^n \), one obtains the probability of a differential \( \alpha_0 \xrightarrow{F} \alpha_t \) as the sum of the probabilities of all its containing differential trails, i.e.,

\[
\text{Prob}(\alpha_0 \xrightarrow{F} \alpha_t) = \sum_{\alpha_1, \ldots, \alpha_{t-1} \in \mathbb{F}_2^n} \text{Prob}(\alpha_0 \xrightarrow{F_1} \alpha_1 \xrightarrow{F_2} \ldots \xrightarrow{F_{t-1}} \alpha_{t-1} \xrightarrow{F_t} \alpha_t).
\]

There remains the question of how to efficiently compute the probability of a differential trail, especially for larger values of \( n \). Indeed, this is a rather difficult task. However, if we assume that the probabilities of the differentials over \( F_i \) are independent, one can compute the probability of the differential trail as the product of the probabilities of the differentials over the \( F_i \). This assumption is likely to make the computation much easier, especially if the definitions of the \( F_i \) are quite simple. We outline this property for the case of product ciphers in the following.
Let $E$ be a product cipher with rounds $R_0, \ldots, R_t$. If all the rounds $R_i$ are such that, for each differential $(\alpha, \beta)$, the expected differential probability $\text{EDP}_{R_i}(\alpha, \beta)$ is independent of the choice of the actual input value $x$, i.e., if

$$\forall x \in \mathbb{F}_2^n : \sum_{k \in \mathbb{F}_2^n} \text{Prob}(\alpha \xrightarrow{R_k} \beta) = |\{k \in \mathbb{F}_2^n \mid R_{i_k}(x) + R_{i_k}(x + \alpha) = \beta\}|,$$

(2.2)

it can be shown that the average probability of any differential trail when averaged over all possible round keys can be computed as the product of the average probabilities of its single-round differentials when averaged over all round keys [LM91]. In other words,

$$\sum_{k_0, \ldots, k_t \in \mathbb{F}_2^n} \text{Prob}(\alpha_0 \xrightarrow{R_{k_0}} \ldots \xrightarrow{R_{k_t}} \alpha_{t+1}) = \sum_{k_0, \ldots, k_t \in \mathbb{F}_2^n} \prod_{i=0}^t \text{Prob}(\alpha_i \xrightarrow{R_{k_i}} \alpha_{i+1}).$$

Product ciphers for which the rounds fulfill Equation 2.2 were defined as Markov ciphers in [LM91]. Key-alternating ciphers are a special case of product ciphers for which the following, well-known connection can be shown.

**Theorem 2.1.** Let $E$ be an $(n, (t + 1)n)$-block cipher that is defined as a $t$-round key-alternating cipher (as depicted in Figure 2.2 with $R_0 = \text{id}$). Let further $\text{KeySchedule}: \mathbb{F}_2^{(t+1)n} \rightarrow (\mathbb{F}_2^n)^{t+1}$ bijectively map the $(t+1)n$-bit initial key to $t+1$ round keys of $n$-bit. Then, the expected differential probability of a differential $(\alpha_0, \alpha_t)$ can be given as

$$\text{EDP}_E(\alpha_0, \alpha_t) = \sum_{\alpha_1, \ldots, \alpha_{t-1} \in \mathbb{F}_2^n} \prod_{i=1}^t \text{Prob}(\alpha_{i-1} \xrightarrow{R_i} \alpha_i).$$

(2.3)

**Proof.** One can see that the rounds for decryption, i.e.,

$$R_i^{-1}: \mathbb{F}_2^n \times \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$$

$$(x, k) \mapsto R_i^{-1}(x + k),$$

fulfill the properties of a Markov cipher as in Equation 2.2. In particular, for all $\tilde{x} \in \mathbb{F}_2^n$, it is

$$\sum_{k \in \mathbb{F}_2^n} \text{Prob}(\alpha \xrightarrow{R_k^{-1}} \beta) = \sum_{k \in \mathbb{F}_2^n} \frac{|\{x \in \mathbb{F}_2^n \mid R_i^{-1}(x + k) + R_i^{-1}(x + k + \alpha) = \beta\}|}{2^n}$$

$$= |\{x \in \mathbb{F}_2^n \mid R_i^{-1}(x) + R_i^{-1}(x + \alpha) = \beta\}|$$

$$= |\{k \in \mathbb{F}_2^n \mid R_k^{-1}(\tilde{x} + k) + R_k^{-1}(\tilde{x} + k + \alpha) = \beta\}|$$

$$= |\{k \in \mathbb{F}_2^n \mid R_k^{-1}(\tilde{x}) + R_k^{-1}(\tilde{x} + \alpha) = \beta\}|.$$

---

9In [LM91], the theory was developed for a more general notion of “difference” that can be defined for any group operation on the message space. However, as we are focusing on key-alternating ciphers, we only consider XOR differences in this thesis.
Moreover, for all keys \( k \in \mathbb{F}_2^n \), we have \( \text{Prob}(\alpha \xrightarrow{R_k^{-1}} \beta) = \text{Prob}(\alpha \xrightarrow{R_{-1}} \beta) \). For the \( t \)-round key-alternating cipher \( E \), one can now deduce the following on the expected differential probability of a differential \((\alpha_{-1}, \alpha_t)\):

\[
\text{EDP}_E(\alpha_{-1}, \alpha_t) = \frac{1}{2(t+1)^n} \sum_{k \in \mathbb{F}_2^{(t+1)n}} \text{Prob}(\alpha_{-1} \xrightarrow{E_k} \alpha_t)
\]

\[
= \frac{1}{2(t+1)^n} \sum_{k \in \mathbb{F}_2^{(t+1)n}} \text{Prob}(\alpha_t \xrightarrow{E_k^{-1}} \alpha_{-1})
\]

\[
= \frac{1}{2(t+1)^n} \sum_{k_0 \ldots k_t \in \mathbb{F}_2^n} \sum_{\alpha_0, \ldots, \alpha_{t-1} \in \mathbb{F}_2^n} \prod_{i=0}^{t} \text{Prob}(\alpha_t \xrightarrow{R_{k_i}^{-1}} \alpha_{i-1})
\]

\[
= \sum_{\alpha_0, \ldots, \alpha_{t-1} \in \mathbb{F}_2^n} \prod_{i=0}^{t} \text{Prob}(\alpha_i \xrightarrow{R_{i}^{-1}} \alpha_{i-1})
\]

\[
= \sum_{\alpha_0, \ldots, \alpha_{t-1} \in \mathbb{F}_2^n} \prod_{i=0}^{t} \text{Prob}(\alpha_{i-1} \xrightarrow{R_i} \alpha_i)
\]

\[
= \sum_{\alpha_0 \in \mathbb{F}_2^n} \text{Prob}(\alpha_{-1} \xrightarrow{R_0} \alpha_0) \sum_{\alpha_1, \ldots, \alpha_{t-1} \in \mathbb{F}_2^n} \prod_{i=1}^{t} \text{Prob}(\alpha_{i-1} \xrightarrow{R_i} \alpha_i)
\]

Now, by the definition of the \( t \)-round key-alternating cipher, we have \( R_0 = \text{id} \) and thus, \( \text{Prob}(\alpha_{-1} \xrightarrow{R_0} \alpha_0) = 1 \) if and only if \( \alpha_{-1} = \alpha_0 \). Otherwise, the probability is zero. By substituting \( \alpha_{-1} \) with \( \alpha_0 \) and vice versa, we finally obtain

\[
\text{EDP}_E(\alpha_0, \alpha_t) = \sum_{\alpha_1, \ldots, \alpha_{t-1} \in \mathbb{F}_2^n} \prod_{i=1}^{t} \text{Prob}(\alpha_{i-1} \xrightarrow{R_i} \alpha_i)
\]

The above theorem is only valid if there is no real key-scheduling algorithm, i.e., if the initial key already consists of all the round keys. In other words, Equation 2.3 only holds true under the assumption of independent round keys. When analyzing a cipher in practice, we will implicitly assume that the round keys are independent and that Equation 2.3 holds true.

**Assumption 2.2 (Independent Round Keys).** We assume that the round keys of a \( t \)-round key-alternating cipher are independent. Then, according to Theorem 2.1, the expected differential probability of a differential \((\alpha_0, \alpha_t)\) can be computed as given in Equation 2.3.
Differentials over the Building Blocks of SP Ciphers

As explained in Section 2.2.2, the rounds of SP ciphers have a special structure, i.e., they consist of a parallel application of smaller invertible S-boxes followed by the application of an invertible linear transformation. For those layers, it is straightforward to derive the following, well-known properties on the differential probability. We therefore state the following proposition without proof.

Proposition 2.1 (Differential Probability over the Building Blocks of SP Ciphers). One can give the differential probability over an S-box layer and a linear layer as follows:

(i) Let $S: (F_2^n)^{n_s} \rightarrow (F_2^n)^{n_s}$ be the $n_s$-times parallel application of a function $S_b: F_2^n \rightarrow F_2^n$. Then,

$$\text{Prob}\left((\alpha_1, \ldots, \alpha_{n_s}) S \rightarrow (\beta_1, \ldots, \beta_{n_s})\right) = \prod_{i=1}^{n_s} \text{Prob}(\alpha_i \xrightarrow{S_b} \beta_i).$$

(ii) Let $L: F_2^n \rightarrow F_2^n$ be a linear transformation. Then,

$$\text{Prob}(\alpha \xrightarrow{L} \beta) = \begin{cases} 1 & \text{if } \beta = L(\alpha) \\ 0 & \text{else} \end{cases}.$$ 

The Standard Argument on the Resistance against Differential Attacks

The standard argument of a designer is to guarantee a low upper bound on the expected differential probability (over a round-reduced version of the cipher) of any non-zero differential; based on a single differential trail. In other words, the designer shows that the product of the differential probabilities in any non-zero differential trail over the round-reduced version is below $2^{-n}$. Then, under the simplifying assumption that the expected differential probability as given in Equation 2.3 is dominated by a single product, the expected differential probability would be too low in order to be able to distinguish the round-reduced version of the cipher from a random permutation, thus rendering the cipher secure against differential cryptanalysis. Note that this approach for proving resistance against differential attacks also implicitly assumes the hypothesis of stochastic equivalence and the assumption of independent round keys (i.e., Assumption 2.1 and Assumption 2.2).

Extensions and Generalizations

Several variants and generalizations of differential attacks have been proposed. For instance, impossible differential attacks which utilize differentials with probability exactly zero (see, e.g., [Knu98, Proposition 1], [BBS05]). As other generalizations, we refer to truncated differentials and higher order differential attacks [Lai94, Knu95].
2.3.2 Linear Cryptanalysis

The general idea of linear cryptanalysis, which was discovered by Matsui in 1993 [Mat94], is to approximate a linear Boolean function of the cipher’s output by a linear Boolean function of the input. In particular, to distinguish a keyed instance \( E_k \) of an \((n, \kappa)\)-block cipher \( E \) from a random permutation, the adversary would exploit linear functions \( l_{\alpha}, l_{\beta} : \mathbb{F}_2^n \rightarrow \mathbb{F}_2 \) for which the approximation

\[
l_{\alpha}(x) = l_{\beta}(E_k(x))
\]

holds with a high absolute bias, i.e., for many values \( x \) or only for few values \( x \).

We explain the most important concepts of linear cryptanalysis in the following. For a comprehensive study, we refer to [DR02]. A more recent systematization of knowledge on linear cryptanalysis is given in [KLW17]. We basically follow the lines of the literature.

It is worth remarking that, for a fixed \( n \), the linear Boolean functions in \( B_n \) form a binary vector space of dimension \( n \) which is isomorphic to \( \mathbb{F}_2^n \) via

\[
\alpha \in \mathbb{F}_2^n \mapsto (l_{\alpha} : x \mapsto \langle \alpha, x \rangle).
\]

Here, \( \langle \alpha, x \rangle \) denotes the canonical inner product, which is defined as \( \sum_i \alpha_i x_i \) in \( \mathbb{F}_2 \). Note that, whenever \( \alpha \neq 0 \), the corresponding linear function \( l_{\alpha} \) is balanced, i.e., the outputs 0 and 1 are taken equally often. Equivalently, this can be stated (see [Car07, Lemma 1]) as

\[
\sum_{x \in \mathbb{F}_2^n} (-1)^{\langle \alpha, x \rangle} = \begin{cases} 2^n & \text{if } \alpha = 0 \\ 0 & \text{if } \alpha \neq 0 \end{cases}.
\tag{2.4}
\]

In the context of linear cryptanalysis, a linear approximation is usually defined in terms of vectors (also called masks) \( \alpha, \beta \).

**Definition 2.5 (Linear Approximation).** For a vectorial function \( F : \mathbb{F}_2^k \rightarrow \mathbb{F}_2^l \), a linear approximation is defined as a tuple \((\alpha, \beta)\) with \( \alpha \in \mathbb{F}_2^k, \beta \in \mathbb{F}_2^l \). The bias of the linear approximation is defined as

\[
\epsilon_F(\alpha, \beta) := \text{Prob}_x(\langle \alpha, x \rangle = \langle \beta, F(x) \rangle) - \frac{1}{2} = \frac{|\{x \in \mathbb{F}_2^k \mid \langle \alpha, x \rangle = \langle \beta, F(x) \rangle\}|}{2^k} - \frac{1}{2}
\]

and its correlation is defined as

\[
\text{cor}_F(\alpha, \beta) := 2 \cdot \epsilon_F(\alpha, \beta) = 2 \cdot \text{Prob}_x (\langle \alpha, x \rangle = \langle \beta, F(x) \rangle) - 1.
\]

Thus, the correlation of a linear approximation \((\alpha, \beta)\) can take values between −1 and 1. In many of the literature, the correlation is equivalently represented in terms of the Fourier transform as

\[
\text{cor}_F(\alpha, \beta) = \frac{1}{2^k} \sum_{x \in \mathbb{F}_2^k} (-1)^{\langle \alpha, x \rangle + \langle \beta, F(x) \rangle}.
\tag{2.5}
\]
This representation, together with the fundamental fact given in Equation 2.4, is usually employed for proving the results presented in this section.

The simple idea of an adversary that adheres to linear cryptanalysis is that it is in possession of a linear approximation \((\alpha, \beta)\) over the keyed instance \(E_k\) that holds with high absolute correlation, i.e., \(|\text{cor}_{E_k}(\alpha, \beta)| > 2^{-\frac{n}{2}}\). The adversary can now distinguish the keyed instance from a random permutation by querying the oracle \(O\) for many inputs \(x\) and checking whether \(\langle \alpha, x \rangle = \langle \beta, O(x) \rangle\) holds as often as one would expect by the correlation of the linear approximation.\(^{10}\) In contrast to differential cryptanalysis, in which the adversary has to choose pairs of plaintexts that fulfill a certain input difference, linear cryptanalysis is an example of a known-plaintext attack.

However, if the adversary wants to exploit a particular linear approximation \((\alpha, \beta)\), its correlation over the keyed instance \(E\) is highly dependent on the actual key \(k\). In particular, this key-dependency can be stated in terms of the so-called linear hull, which was introduced by Nyberg in 1994 \([\text{Nyb95}]\). As it is explained in detail in \([\text{KLW17}]\), the correlation of the linear approximation over \(E_k\) can be given as a signed sum of correlations of linear approximations over \(E\), i.e.,

\[
\text{cor}_{E_k}(\alpha, \beta) = \sum_{\gamma \in \mathbb{F}_2^n} (-1)^{\langle \gamma, k \rangle} \text{cor}_E((\alpha, \gamma), \beta).
\]

Therefore, the suitability of a linear approximation for attacking the cipher actually depends on the distribution of the correlations for all possible keys. In other words, the attack might work for "almost" all keys or only for (few) particular keys, depending on this distribution. The keys for which the linear attack work are usually referred to as weak keys. Thus, for a thorough understanding of the security of a cipher, one has to study this distribution. As the description of the linear hull given above is the most general description of the key dependency for arbitrary block ciphers, we will see later in Corollary 2.1 that the linear hull in the case where \(E\) is a \(t\)-round key-alternating cipher actually reduces to a much concreter and simpler expression.

Moreover, under the assumption of independent round keys (i.e., Assumption 2.2), one can derive the mean and variance of the distribution of correlations over all keys (see Corollary 2.2 and Corollary 2.3). Of course, the actual distribution depends on the key-scheduling algorithm of the cipher and the assumption of independent round keys is only for simplifying the analysis. In fact, not much theory on how the key-schedule affects this distribution is known. For more details, we refer to e.g., \([\text{KLW17}]\), where the focus is on linear key schedules. We now elaborate more on linear cryptanalysis of key-alternating ciphers in particular.

\(^{10}\)[Mat94] indicates that one needs about \(c \cdot |\text{cor}_{E_k}(\alpha, \beta)|^2\) known plaintexts for a distinguisher with a reasonable high advantage, where \(c\) is some small constant. Thus, if the absolute correlation would be smaller than \(2^{-\frac{n}{2}}\), there would not be enough plaintexts available.
Linear Cryptanalysis of Key-Alternating Ciphers

When having a linear approximation \((\alpha_0, \alpha_t)\) over an iterative function, similar to the notion of a differential trail, one can consider a chain of approximations over the particular iterations. This leads to the notion of a linear trail as follows.

**Definition 2.6** (Linear Trail (see [DGV95])). Let \(F : \mathbb{F}_2^n \to \mathbb{F}_2^n\) be an iterated function of the form \(F = F_t \circ \cdots \circ F_2 \circ F_1\) with \(F_i : \mathbb{F}_2^n \to \mathbb{F}_2^n\). Given \(t + 1\) vectors \(\alpha_0, \ldots, \alpha_t \in \mathbb{F}_2^n\), the tuple \((\alpha_0, \ldots, \alpha_t)\) is said to be a linear trail over \(F\) and its correlation is defined as

\[
\text{cor}_{F_1, \ldots, F_t}(\alpha_0, \alpha_1, \ldots, \alpha_t) := \prod_{i=1}^{t} \text{cor}_{F_i}(\alpha_{i-1}, \alpha_i).
\]

This leads to the following important theorem for iterative functions, which was first stated in [DGV95].

**Theorem 2.2** (Theorem of Linear Trail Composition (Theorem 7.8.1 in [DR02])). Let \(F : \mathbb{F}_2^n \to \mathbb{F}_2^n\) be an iterated function of the form \(F = F_t \circ \cdots \circ F_2 \circ F_1\) with \(F_i : \mathbb{F}_2^n \to \mathbb{F}_2^n\). Then, the correlation of the linear approximation \((\alpha_0, \alpha_t)\) over \(F\) can be given as the sum of the correlations of all its containing linear trails, i.e.,

\[
\text{cor}_F(\alpha_0, \alpha_t) = \sum_{\alpha_1, \ldots, \alpha_{t-1} \in \mathbb{F}_2^n} \text{cor}_{F_1, \ldots, F_t}(\alpha_0, \alpha_1, \ldots, \alpha_{t-1}, \alpha_t).
\]

**Proof.** Without loss of generality, we show the theorem for \(F = F_2 \circ F_1\), where \(F_1, F_2 : \mathbb{F}_2^n \to \mathbb{F}_2^n\). The general case then follows by induction. We use the representation given in Equation 2.5 and the fundamental fact given in Equation 2.4. In particular, for fixed \(\alpha_0, \alpha_2 \in \mathbb{F}_2^n\), it is

\[
\sum_{\alpha_1 \in \mathbb{F}_2^n} \text{cor}_{F_1}(\alpha_0, \alpha_1) \text{cor}_{F_2}(\alpha_1, \alpha_2)
= \sum_{\alpha_1 \in \mathbb{F}_2^n} \frac{1}{2^n} \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{(\langle \alpha_0, x \rangle + \langle \alpha_1, F_1(x) \rangle \sum_{x' \in \mathbb{F}_2^n} (-1)^{(\langle \alpha_1, x' \rangle + \langle \alpha_2, F_2(x') \rangle)}
= \frac{1}{2^n} \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} \sum_{x' \in \mathbb{F}_2^n} (-1)^{(\langle \alpha_0, x \rangle + \langle \alpha_2, F_2(x') \rangle \sum_{\alpha_1 \in \mathbb{F}_2^n} (-1)^{(\langle \alpha_1, F_1(x) \rangle + x')}}
= \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{(\langle \alpha_0, x \rangle + \langle \alpha_2, F_2(F_1(x)) \rangle = \text{cor}_F(\alpha_0, \alpha_2).}
\]

We now focus on linear approximations over key-alternating ciphers. Let \(E_k\) be a keyed-instance of a \(t\)-round key alternating cipher. In particular,

\[
E_k = R_{t_k} \circ \cdots \circ R_{1_{k_1}} \circ R_{0_{k_0}}
\]
with $R_{k_i} = \text{Add}_{k_i} \circ R_i$ and $R_0 = \text{id}$. Thereby, $k_0, \ldots, k_t$ denote the round keys that are derived from $k$ by the key-scheduling algorithm. The correlation of a linear approximation $(\alpha, \beta)$ over a key addition $\text{Add}_k$ can be given as

$$\text{cor}_{\text{Add}_k}(\alpha, \beta) = \begin{cases} (-1)^{\langle \beta, k \rangle} & \text{if } \alpha = \beta \\ 0 & \text{else} \end{cases}.$$  

This can be deduced from the fact that

$$\langle \alpha, x \rangle = \langle \beta, x + k \rangle$$

$$\iff \langle \alpha, x \rangle = \langle \beta, x \rangle + \langle \beta, k \rangle$$

$$\iff \langle \alpha + \beta, x \rangle = \langle \beta, k \rangle$$

and $x \mapsto \langle \alpha + \beta, x \rangle$ is balanced (i.e., the correlation $\text{cor}_{\text{Add}_k}(\alpha, \beta)$ is zero) if and only if $\alpha \neq \beta$. Otherwise, $x \mapsto \langle \alpha + \beta, x \rangle$ is equal to the zero function.

This implies that the absolute correlation of a linear trail over $R_{k_0}, \ldots, R_{k_t}$ is independent of the actual round keys $k_0, \ldots, k_t$. In particular,

$$\text{cor}_{R_{k_0}, \ldots, R_{k_t}}(\alpha_{-1}, \alpha_0, \ldots, \alpha_t) = \prod_{i=0}^{t} \text{cor}_{R_{k_i}}(\alpha_{i-1}, \alpha_i)$$

$$= \prod_{i=0}^{t} \sum_{\gamma} \text{cor}_{R_i}(\alpha_{i-1}, \gamma) \text{cor}_{\text{Add}_{k_i}}(\gamma, \alpha_i)$$

$$= \prod_{i=0}^{t} (-1)^{\langle \alpha_i, k_i \rangle} \text{cor}_{R_i}(\alpha_{i-1}, \alpha_i)$$

$$= (-1)^{\langle \alpha_0, k_0 \rangle + \ldots + \langle \alpha_t, k_t \rangle} \prod_{i=0}^{t} \text{cor}_{R_i}(\alpha_{i-1}, \alpha_i).$$

From this formula and Theorem 2.2, one obtains the following, well-known connections as straightforward implications (see [DR02, pp. 103–108]).

**Corollary 2.1** (Linear Hull Theorem for Key-Alternating Ciphers). Let $E$ be a $t$-round key-alternating cipher (as depicted in Figure 2.2 with $R_0 = \text{id}$). Then, the correlation of a linear approximation $(\alpha_0, \alpha_t)$ over a keyed instance $E_k$ is given as

$$\text{cor}_{E_k}(\alpha_0, \alpha_t) = \sum_{\alpha_1, \ldots, \alpha_{t-1} \in \mathbb{F}_2} (-1)^{\langle \alpha_0, k_0 \rangle + \ldots + \langle \alpha_t, k_t \rangle} \prod_{i=1}^{t} \text{cor}_{R_i}(\alpha_{i-1}, \alpha_i).$$

Thereby, $(k_0, \ldots, k_t)$ denote the round keys that are derived from $k$ by the key-scheduling algorithm.

For independent round keys, it follows that the average correlation over all keys is equal to zero. One can further give the average square correlation of a
given linear approximation over all keys as the sum of the square correlations of all containing linear trails over the unkeyed rounds. These connections are stated in the following Corollaries 2.2 and 2.3.

**Corollary 2.2.** Let \( E \) be an \((n, (t+1)n)\)-block cipher that is defined as a \(t\)-round key-alternating cipher (as depicted in Figure 2.2 with \( R_0 = \text{id} \)). Let further \( \text{KeySchedule}: F_2^{(t+1)n} \rightarrow (F_2^n)^{t+1} \) bijectively map the \((t+1)n\)-bit initial key to \(t+1\) round keys of \( n\)-bit. Then, the average correlation over all keys of any non-trivial linear approximation \((\alpha_0, \alpha_t)\) equals zero, i.e.,

\[
\forall \alpha_0, \alpha_t \in F_2^n \setminus \{0\}: \frac{1}{2^{(t+1)n}} \sum_{k \in F_2^{(t+1)n}} \text{cor}_{E_k}(\alpha_0, \alpha_t) = 0.
\]

**Proof.** Let \( \alpha_0, \alpha_t \in F_2^n \setminus \{0\} \). Then,

\[
\sum_{k \in F_2^{(t+1)n}} \text{cor}_{E_k}(\alpha_0, \alpha_t) = \sum_{k_0, \ldots, k_t \in F_2^n} \sum_{\alpha_1, \ldots, \alpha_{t-1} \in F_2^n} (-1)^{\langle \alpha_0, k_0 \rangle + \cdots + \langle \alpha_t, k_t \rangle} \prod_{i=1}^t \text{cor}_{R_i}(\alpha_{i-1}, \alpha_i)
\]

\[
= \sum_{\alpha_1, \ldots, \alpha_{t-1} \in F_2^n} \prod_{i=1}^t \text{cor}_{R_i}(\alpha_{i-1}, \alpha_i) \sum_{k_0, \ldots, k_t \in F_2^n} (-1)^{\langle \alpha_0, k_0 \rangle + \cdots + \langle \alpha_t, k_t \rangle}
\]

\[
= \sum_{\alpha_1, \ldots, \alpha_{t-1} \in F_2^n} \prod_{i=1}^t \text{cor}_{R_i}(\alpha_{i-1}, \alpha_i) \cdot 0
\]

\[= 0. \]

\[
\]

**Corollary 2.3** (See Theorem 7.9.1 in [DR02]). Let \( E \) be an \((n, (t+1)n)\)-block cipher that is defined as a \(t\)-round key-alternating cipher (as depicted in Figure 2.2 with \( R_0 = \text{id} \)). Let further \( \text{KeySchedule}: F_2^{(t+1)n} \rightarrow (F_2^n)^{t+1} \) bijectively map the \((t+1)n\)-bit initial key to \(t+1\) round keys of \( n\)-bit. Then, for the average square correlation over all keys of the linear approximation \((\alpha_0, \alpha_t)\), one obtains

\[
\frac{1}{2^{(t+1)n}} \sum_{k \in F_2^{(t+1)n}} \text{cor}_{E_k}(\alpha_0, \alpha_t)^2 = \sum_{\alpha_1, \ldots, \alpha_{t-1}} \prod_{i=1}^t \text{cor}_{R_i}(\alpha_{i-1}, \alpha_i)^2.
\]

**Linear Approximations over the Building Blocks in SP Ciphers**

We have already seen the correlation of a linear approximation over a key addition \( \text{Add}_k \). For the other building blocks of SP ciphers, similar to the case of differentials as described in Proposition 2.1 there are well-known simplified expressions for the correlations of linear approximations. As they are straightforward to obtain, we state them without proof.
Proposition 2.2 (Correlation over the Building Blocks of SP ciphers). One can compute the correlation of a linear approximation over an S-box layer and a linear layer as follows:

(i) Let $S: (\mathbb{F}_2^n)^{n_s} \rightarrow (\mathbb{F}_2^n)^{n_s}$ be the $n_s$-times parallel application of a bijective function $S_b: \mathbb{F}_2^s \rightarrow \mathbb{F}_2^s$. Then,

$$\text{cor}_S ((\alpha_1, \ldots, \alpha_{n_s}), (\beta_1, \ldots, \beta_{n_s})) = \prod_{i=1}^{n_s} \text{cor}_{S_b}(\alpha_i, \beta_i).$$

(ii) Let $L: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, $x \mapsto Mx$ be a linear permutation given by $M \in \text{GL}_n(\mathbb{F}_2)$. Then,

$$\text{cor}_L(\alpha, \beta) = \begin{cases} 1 & \text{if } \alpha = M^\top \beta \\ 0 & \text{else} \end{cases}.$$ 

The Standard Argument on the Resistance against Linear Attacks

Similar to the argument for the resistance against differential cryptanalysis, the standard designer’s argument for the resistance against linear cryptanalysis is based on a single linear trail. In particular, the aim is to guarantee a low upper bound (i.e., $< 2^{-\frac{n}{2}}$) on the absolute correlation of any non-zero linear trail over a reduced-round version of the cipher. Then, under the simplifying assumption that the correlation of the linear approximation as given in Corollary 2.1 is dominated by a correlation of a single linear trail, the absolute correlation of the linear approximation would be too low in order to be able to distinguish the round-reduced version of the cipher from a random permutation.

Extensions and Generalizations

Over the years, several extensions and generalizations of linear cryptanalysis appeared. As an example, we refer to zero-correlation attacks [BR14], in which linear approximations with correlation exactly zero are utilized. In principle, instead of approximating the cipher’s input and output by linear Boolean functions $l_\alpha$ and $l_\beta$, one can think about approximating the input and output by any, not necessarily linear, Boolean functions $g$ and $h$, respectively. Analogous to the correlation of a linear approximation, for a vectorial function $F: \mathbb{F}_2^k \rightarrow \mathbb{F}_2^l$, the correlation of an approximation $(g, h)$, with $g \in B_k, h \in B_l$, is defined as

$$\text{cor}_F(g, h) := 2 \cdot \text{Prob}_x (g(x) = h(F(x))) - 1.$$ 

In the case where $g$ and $h$ are balanced, the distinguisher works similar as in the case of linear cryptanalysis. The usage of nonlinear approximations as a generalization of linear cryptanalysis has first been considered in [HKM95] and [KR96b]. However, for a long time, nonlinear approximations were no real thread for practical block ciphers as there is a complex key dependency and no real possibility of
iteratively joining approximations, as it is done with linear trails. More recently, the usage of nonlinear approximations was rediscovered with the introduction of so-called invariant attacks, i.e., invariant subspace [LAAZ11] and nonlinear invariant attacks [TLS16]. These attacks utilize a nonlinear approximation \((g, h)\) for which \(\text{cor}_{E_k}(g, h) \in \{-1, 1\}\) for a significant fraction of weak keys \(k\). They work especially well for lightweight ciphers with a simple key schedule, see Section 2.5 and Chapter 6.

2.4 The Wide-Trail Strategy and AES-like Ciphers

In this section, we explain the wide-trail strategy as introduced by Daemen in [Dae95]. It proposes a design strategy for key-alternating block ciphers that allows for simple arguments on the resistance against differential and linear attacks. The main starting point is that the (unkeyed) round functions \(R_i\) of the key-alternating cipher are composed as \(R_i = L \circ S\) for an S-box layer \(S\) and a linear layer \(L\). Thereby, \(S\) consists of an \(n_s\)-times parallel application of the bijective \(s\)-bit S-box \(S_b\) and \(L\) can be given by \(M \in \text{GL}_{n_s}(\mathbb{F}_2)\) as \(x \mapsto Mx\). Instead of using an ordinary bit permutation for \(L\), as it was originally done in substitution-permutation networks, the wide-trail strategy explains how the linear layer could be chosen in a more general way to avoid the existence of differential (resp. linear) trails with high probability (resp. absolute correlation).

For a vector \(v = (v_1, \ldots, v_d) \in (\mathbb{F}_2^s)^d\), its weight with respect to \(s\)-bit words, denoted \(w_s(v)\), is defined as the total number of non-zero \(s\)-bit words in \(v\). Formally,

\[
w_s(v) := \sum_{i=1}^{d} \delta_s(v_i), \quad \text{where} \quad \delta_s(v_i) := \begin{cases} 1 \in \mathbb{Z} & \text{if } v_i \neq 0 \\ 0 \in \mathbb{Z} & \text{if } v_i = 0 \end{cases}.
\]

Thereby, the value of \(\delta_s(v_i)\) is said to be the activity of \(v_i\). We further define the function

\[
\vec{\delta}_s: (\mathbb{F}_2^s)^d \rightarrow \mathbb{Z}^d \\
(v_1, \ldots, v_d) \mapsto (\delta_s(v_1), \ldots, \delta_s(v_d))
\]

and we call \(\vec{\delta}_s(v)\) the activity pattern of \(v\). Let now \(C = (\alpha^{(0)}, \alpha^{(1)}, \ldots, \alpha^{(t)})\) be a \(t\)-round (differential or linear) trail,\(^{11}\) where \(\forall r \in \{0, \ldots, t\}, \alpha^{(r)} = (\alpha_1^{(r)}, \ldots, \alpha_{n_s}^{(r)}) \in (\mathbb{F}_2^s)^{n_s}\). The weight of the trail with respect to \(s\)-bit words, denoted \(w_t(C)\), is defined as the total number of non-zero \(s\)-bit words in the first \(t\) difference (resp. mask) patterns for the round inputs, i.e.,

\[
\text{wt}_s(C) := \sum_{r=0}^{t-1} w_s(\alpha^{(r)}) = \sum_{r=0}^{t-1} \sum_{i=1}^{n_s} \delta_s(\alpha_i^{(r)}).
\]

\(^{11}\) We change to superscript notation \(^{(r)}\) for denoting the round \(r\) within a trail.
Note that, as the round function is defined as $L \circ S$ with $S$ splitting into smaller $s$-bit S-boxes as described above, the $s$-bit words in the first $t$ components of the trail correspond exactly to the S-box input differences (resp. masks). Therefore, if $\delta_s(\alpha_i^{(r)}) = 1$, the $i$-th S-box is called active. If $\delta_s(\alpha_i^{(r)}) = 0$, it is called passive.

**The Wide-Trail Security Argument**

According to Proposition 2.1, we have a simple formula for the product of the probabilities of the single-round differentials contained in the differential trail $C$.

**Similarly, if $C$ is a linear trail, we have a simple formula for the absolute correlation of $C$ according to Proposition 2.2. In particular,**

$$\prod_{r=1}^{t} \text{Prob} (\alpha^{(r-1)} \xrightarrow{L \circ S} \alpha^{(r)})$$

and

$$|\text{cor}_{L \circ S, \ldots, L \circ S} (\alpha^{(0)}, \alpha^{(1)}, \ldots, \alpha^{(t)})| = \prod_{r=1}^{t} \prod_{i=1}^{n_s} |\text{cor}_{S_b} (\alpha_i^{(r-1)}, (M^T \alpha^{(r)}))| .$$

We can then bound the product of the probabilities of the single-round differentials contained in a differential trail $C$ as

$$\prod_{r=1}^{t} \text{Prob}(\alpha^{(r-1)} \xrightarrow{L \circ S} \alpha^{(r)}) \leq p_{S_b}^{w(C)} , \text{ where } p_{S_b} := \max_{\alpha \neq \beta} \{ \text{Prob}(\alpha \xrightarrow{S_b} \beta) \} .$$

Analogously, the absolute correlation of a linear trail $C$ can be upper bounded as

$$|\text{cor}_{L \circ S, \ldots, L \circ S}(\alpha^{(0)}, \alpha^{(1)}, \ldots, \alpha^{(t)})| \leq c_{S_b}^{w(C)} , \text{ where } c_{S_b} := \max_{\alpha, \beta \neq 0} \{|\text{cor}_{S_b}(\alpha, \beta)|\} .$$

We say that a $t$-round differential trail $C = (\alpha^{(0)}, \alpha^{(1)}, \ldots, \alpha^{(t)}) \neq 0$ is valid if every differential probability of its containing single-round differentials is non-zero. We denote the set of valid $t$-round differential trails over $R_t \circ \cdots \circ R_2 \circ R_1$ by $V_{\text{diff}}^{R_1, \ldots, R_t}$. Formally,

$$V_{R_1, \ldots, R_t}^{\text{diff}} = \{(\alpha^{(0)}, \alpha^{(1)}, \ldots, \alpha^{(t)}) \neq 0 \mid \forall r \in \{1, \ldots, t\} : \text{Prob}(\alpha^{(r-1)} \xrightarrow{R_r} \alpha^{(r)}) \neq 0 \} .$$

Analogously, we say that a $t$-round linear trail $C = (\alpha^{(0)}, \alpha^{(1)}, \ldots, \alpha^{(t)}) \neq 0$ is valid if its correlation is non-zero. We denote the set of valid $t$-round linear trails over $R_t \circ \cdots \circ R_2 \circ R_1$ by $V_{R_1, \ldots, R_t}^{\text{lin}}$. Formally,

$$V_{R_1, \ldots, R_t}^{\text{lin}} = \{(\alpha^{(0)}, \alpha^{(1)}, \ldots, \alpha^{(t)}) \neq 0 \mid \forall r \in \{1, \ldots, t\} : \text{cor}_{R_r}(\alpha^{(r-1)}, \alpha^{(r)}) \neq 0 \} .$$

The goal of the cipher designer is to guarantee a low upper bound on the product of the single-round differential probabilities of any valid differential trail and on
the absolute correlation of any valid linear trail, respectively. There are basically two approaches the designer can focus on. Firstly, he can choose an S-box $S_b$ for which $p_{S_b}$ and $c_{S_b}$ are low. As upper bounds, one can show that, for any $s$-bit S-box $S_b$, $p_{S_b} \geq 2^{1-s}$ and $c_{S_b} \geq 2^{-2^{s-1}}$, see [NK93, CV95]. In the first place, this suggests that the word length $s$ should be chosen to be large, ideally $s = n$. However, as the value of $s$ gets larger, the size of the description of the S-box as a look-up table grows exponentially. Therefore, one usually aims for $s$ to be fairly low, e.g., $s \leq 8$.

A second approach would be to design a linear layer $L$ that maximizes the minimum weight of any valid differential or linear trail. This suggest the usage of a more complex, carefully-chosen, $L$ and is exactly the approach of the wide-trail strategy. In a nutshell, it should be the responsibility of the linear layer to guarantee high diffusion in differential and linear trails. According to Proposition 2.1 and as explained above, the set

$$\Gamma_{L,s,t}^{\text{diff}} := \{ (\alpha(0), \ldots, \alpha(t)) \neq 0 | \forall r \in \{1, \ldots, t\}: \vec{\delta}_s(\alpha^{(r-1)}) = \vec{\delta}_s(L^{-1}(\alpha^{(r)})) \} \quad (2.6)$$

contains all valid $t$-round differential trails as a subset. Moreover, it is only dependent on the linear layer $L$ and not on the S-box layer $S$. Similarly, according to Proposition 2.2, the set

$$\Gamma_{L,s,t}^{\text{lin}} := \{ (\alpha(0), \ldots, \alpha(t)) \neq 0 | \forall r \in \{1, \ldots, t\}: \vec{\delta}_s(\alpha^{(r-1)}) = \vec{\delta}_s(M^T \alpha^{(r)}) \} \quad (2.7)$$

contains all valid $t$-round linear trails as a subset and is not dependent on the particular S-box layer $S$. The goal of the wide-trail strategy is to define a linear layer

$L: \mathbb{F}_2^{s \cdot n_s} \rightarrow \mathbb{F}_2^{s \cdot n_s}$

that guarantees a high weight for all trails in $\Gamma_{L,s,t}^{\text{diff}}$ and $\Gamma_{L,s,t}^{\text{lin}}$, for some certain value of $t$. One also uses the wording that $L$ should lead to a high number of active S-boxes over $t$ rounds. That value of $t$ for which the minimum number of active S-boxes can be proven to be high enough would be an indicator for the number of rounds specified for the cipher.

It is worth remarking that, whenever the weight of any valid $t$-round trail is at least $w$, the weight of any valid $dt$-round trail must be at least $d \cdot w$.

### 2.4.1 The Branch Number and a Link to Coding Theory

In [Dae95], Daemen introduced the branch number of a linear transformation as a measure for its diffusion. Indeed, it provides a lower bound on the minimum number of active S-boxes in any valid differential (resp. linear) trail over two rounds.

**Definition 2.7** (Differential and linear branch number (see, e.g., pp. 131–132 of [DR02])). Let $L: \mathbb{F}_2^{s \cdot n_s} \rightarrow \mathbb{F}_2^{s \cdot n_s}, x \mapsto Mx$ be an $\mathbb{F}_2$-linear transformation. The differential branch number of $L$ with respect to $s$-bit words is defined as

$$B_s^{\text{diff}}(L) := \min_{\alpha \neq 0} \{ w_s(\alpha) + w_s(L(\alpha)) \}.$$
Analogously, the linear branch number of $L$ with respect to $s$-bit words is defined as

$$B_s^{\text{lin}}(L) := \min_{\alpha \neq 0} \{w_s(\alpha) + w_s(M^T \alpha)\}.$$ 

**Proposition 2.3** (Two-Round Propagation Theorem (Theorem 9.3.1 in [DR02])).

The weight with respect to $s$-bit words of any valid two-round differential (resp. linear) trail over round functions $L \circ S$, where $S$ consists of a parallel application of an $s$-bit S-box, is lower bounded by the differential (resp. linear) branch number of $L$, i.e.,

$$\min_{C \in V^{\text{diff}}_{L,S,S,L}} \{\text{wt}_s(C)\} \geq B_s^{\text{diff}}(L) \quad \text{and} \quad \min_{C \in V^{\text{lin}}_{L,S,S,L}} \{\text{wt}_s(C)\} \geq B_s^{\text{lin}}(L).$$

**Proof.** As the case of linear trails can be proven in a similar way as the case of differential trails, we only show the proof for differential trails here.

We consider trails in $\Gamma^{\text{diff}}_{L,s,t}$. In particular, let $C = (\alpha(0), \alpha(1), \alpha(2))$ be a differential trail in $\Gamma^{\text{diff}}_{L,s,2}$. Then, $\vec{\delta}_s(\alpha(0)) = \vec{\delta}_s(L^{-1}(\alpha(1)))$. Moreover,

$$\text{wt}_s(C) = w_s(\alpha(0)) + w_s(\alpha(1)) = w_s(L^{-1}(\alpha(1))) + w_s(\alpha(1)) \geq B_s^{\text{diff}}(L),$$

as $\alpha(1) \neq 0$. \qed

One immediately obtains

**Corollary 2.4.** The weight with respect to $s$-bit words of any valid $2t$-round differential (resp. linear) trail over round functions $L \circ S$, where $S$ consists of a parallel application of an $s$-bit S-box, is lower bounded by $t \cdot B_s^{\text{diff}}(L)$ (resp. $t \cdot B_s^{\text{lin}}(L)$).

It is obvious that for any $\mathbb{F}_2$-linear transformation on $\mathbb{F}_{2^d}$, its differential and linear branch number with respect to $s$-bit words is smaller or equal to $d + 1$. In [RDP+96], the relation between the branch number of a linear transformation and the minimum distance of a linear code was pointed out. In particular, if $L: \mathbb{F}_{2^d}^d \rightarrow \mathbb{F}_{2^d}^d$ is defined as an $\mathbb{F}_{2^d}$-linear transformation over the $d$-dimensional vector space $\mathbb{F}_{2^d}$, the differential branch number of $L$ is equal to the minimum distance of the $\mathbb{F}_{2^d}$-linear code $C$ of length $2d$ and dimension $d$ generated by $[I_d M]$, where $I_d$ is the $d \times d$ identity matrix and $M \in \text{GL}_d(\mathbb{F}_{2^d})$ the to $L$ associated matrix for some choice of basis. Analogously, the linear branch number of $L$ is equal to the minimum distance of the dual code $C^\perp$, which is generated by $[M^T I_d]$. From the Singleton bound of linear codes, (see [MS77, Chapter 1, Theorem 11]), one immediately obtains that the minimum distance of each of those codes is upper bounded by $d + 1$. As linear codes with highest possible minimum distance are called maximum distance separable (MDS), the following terminology has been established for matrices that lead to an optimal branch number.

**Definition 2.8** (MDS Matrix). A $d \times d$ matrix $M$ with coefficients in $\mathbb{F}_{2^d}$ is called maximum distance separable (MDS), if the $\mathbb{F}_{2^d}$-linear code of length $2d$ and dimension $d$ generated by $[I_d M]$ is MDS.
Since for every MDS code its dual code is also MDS (see \cite[Chapter 11, Theorem 2]{MS77}), it follows that a $d \times d$ MDS matrix $M$ corresponds to a linear transformation on $\mathbb{F}_{2^d}$, with optimal differential and linear branch number of $d+1$. Further, an MDS matrix can be characterized as follows.

**Theorem 2.3** (Theorem 8, p. 321 in \cite{MS77}). Let $M$ be a $d \times d$ matrix with coefficients in $\mathbb{F}_{2^s}$. Then, $M$ is MDS if and only if all its square submatrices are invertible.

It follows that MDS matrices must be invertible. A discussion on a more general MDS property in cases where $L$ is not $\mathbb{F}_{2^s}$-linear, but just $\mathbb{F}_2$-linear, is made in Section 3.5.

In order to guarantee the highest possible number of active S-boxes, and thus to obtain a high resistance against differential and linear attacks, the wide-trail strategy suggests the application of an MDS matrix for the linear layer $L$. However, such an MDS matrix may cause a huge implementation overhead for larger values of $n_s$. In the following, we explain a cipher design that allows for a better trade-off in terms of the minimum number of active S-boxes and implementation efficiency.

### 2.4.2 AES-like Ciphers

We now explain a block cipher structure that was especially designed according to the wide-trail strategy. Originally, the structure was introduced with the block cipher SQUARE \cite{DKR97}, a predecessor of the Rijndael cipher which was adopted as the Advanced Encryption Standard (AES) in 2001 \cite{PUB01}. As the AES is the best-known and best-studied block cipher today, we explain its particular round function in the following. However, as the AES inspired lots of other designs (e.g., ANUBIS \cite{BR00b}, LED \cite{GPPR11}, mCrypton \cite{LK06}, Midori \cite{BBI15}, PHOTON \cite{GPP11}, Prince \cite{BCG12}, QARMA \cite{Ava17}, Skinny and Mantis (Chapter 5), and Whirlpool \cite{BR00a}), we give a more general definition first. We call ciphers designed according to this general notion AES-like ciphers.

An AES-like cipher fulfills the notion of an SP cipher as depicted in Figure 2.3. In particular, it is a key-alternating block cipher operating on a block length of $n = s \cdot n_s$, where $n_s$ is split into two dimensions $n_r, n_c$ such that $n_s = n_r \cdot n_c$. For a better representation, one usually writes the cipher’s input, output and its internal states $x \in \mathbb{F}_{2^{n_r \cdot n_c}}$ as an $n_r \times n_c$-dimensional array with $s$-bit words,\footnote{Note that, in the following, we will represent an $s$-bit word $x_i$ as an element of the finite field with $2^s$ elements, i.e., $x_i \in \mathbb{F}_{2^s}$.} i.e.,

$$x = \begin{bmatrix} x_1 & x_{n_r+1} & \cdots & x_{(n_c-1)n_r+1} \\ x_2 & x_{n_r+2} & \cdots & x_{(n_c-1)n_r+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n_r} & x_{2n_r} & \cdots & x_{n_c n_r} \end{bmatrix}, \quad x_i \in \mathbb{F}_{2^s}.$$
The characteristic of an AES-like cipher is that it adheres to a special kind of round function, as given in Definition 2.9 and depicted in Figure 2.6. After the application of such an (unkeyed) round, a round key \( k^{(r)} \in \mathbb{F}_2^{n_r \times n_c} \) is added to the cipher’s internal state as it is common for a key-alternating cipher. Our notion of an AES-like cipher sets no requirements on the key-scheduling algorithm and, for simplicity, we will ignore it in the following. Anyway, as explained earlier in Sections 2.3.1 and 2.3.2, the standard security argument against differential and linear attacks is done under the assumption of independent round keys.

**Definition 2.9.** An AES-like round is defined as a permutation

\[
R_{Sb,p,M} : \mathbb{F}_2^{n_r \times n_c} \rightarrow \mathbb{F}_2^{n_r \times n_c},
\]

which is parametrized by the word length \( s \in \mathbb{N} \), the state dimension \( n_r, n_c \in \mathbb{N} \), where \( n_r \) denotes the number of rows and \( n_c \) denotes the number of columns of an \( n_r \times n_c \) state, an invertible S-box \( Sb : \mathbb{F}_2^s \rightarrow \mathbb{F}_2^s \), a permutation \( p \in S_{n_r,n_c} \), and a matrix \( M \in \text{GL}_{n_c}(\mathbb{F}_2^s) \). In particular, the round function \( R_{Sb,p,M} \) is composed of the bijective transformations \( S_{Sb} \), \( \text{Permute}_p \) and \( \text{Mix}_M \) operating on an \( n_r \times n_c \) state, such that \( R_{Sb,p,M} = \text{Mix}_M \circ \text{Permute}_p \circ S_{Sb} \):

1. \( S_{Sb} \) is a parallel application of the S-box \( Sb : \mathbb{F}_2^s \rightarrow \mathbb{F}_2^s \) to all \( n_r \cdot n_c \) words of the state.

\[
S_{Sb} : (\mathbb{F}_2^s)^{n_r \times n_c} \rightarrow (\mathbb{F}_2^s)^{n_r \times n_c}
\]

\[
\forall i \in \mathbb{N}_{< n_r}, \forall j \in \mathbb{N}_{< n_c} : x_{n_r i + j} \mapsto Sb(x_{n_r i + j}).
\]

2. \( \text{Permute}_p \) permutes the words of the state according to the permutation \( p \), i.e.,

\[
\text{Permute}_p : (\mathbb{F}_2^s)^{n_r \times n_c} \rightarrow (\mathbb{F}_2^s)^{n_r \times n_c}
\]

\[
\forall i \in \mathbb{N}_{< n_r}, \forall j \in \mathbb{N}_{< n_c} : x_{n_r i + j} \mapsto x_{p(n_r i + j)}.
\]
Table 2.1: The S-box $S_{\text{AES}}$ used in the AES. For each (hexadecimal) value of $x$ and $y$, the table shows $S_{\text{AES}}(x||y)$ as a hexadecimal value. For instance, $S_{\text{AES}}(1B) = AF$.

<table>
<thead>
<tr>
<th>$x$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>63</td>
<td>7C</td>
<td>77</td>
<td>7B</td>
<td>F2</td>
<td>6B</td>
<td>6F</td>
<td>C5</td>
<td>30</td>
<td>01</td>
<td>67</td>
<td>2B</td>
<td>FE</td>
<td>D7</td>
<td>AB</td>
<td>76</td>
</tr>
<tr>
<td>1</td>
<td>CA</td>
<td>82</td>
<td>C9</td>
<td>7D</td>
<td>FA</td>
<td>59</td>
<td>47</td>
<td>F0</td>
<td>AD</td>
<td>D4</td>
<td>A2</td>
<td>AF</td>
<td>9C</td>
<td>A4</td>
<td>72</td>
<td>C0</td>
</tr>
<tr>
<td>2</td>
<td>B7</td>
<td>FD</td>
<td>93</td>
<td>26</td>
<td>36</td>
<td>3F</td>
<td>F7</td>
<td>CC</td>
<td>34</td>
<td>A5</td>
<td>E5</td>
<td>F1</td>
<td>71</td>
<td>D8</td>
<td>31</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>04</td>
<td>C7</td>
<td>23</td>
<td>C3</td>
<td>1A</td>
<td>1B</td>
<td>6E</td>
<td>5A</td>
<td>A0</td>
<td>52</td>
<td>B6</td>
<td>B3</td>
<td>29</td>
<td>E3</td>
<td>D4</td>
<td>A2</td>
</tr>
<tr>
<td>4</td>
<td>09</td>
<td>83</td>
<td>2C</td>
<td>1A</td>
<td>1B</td>
<td>6E</td>
<td>5A</td>
<td>A0</td>
<td>52</td>
<td>B6</td>
<td>B3</td>
<td>29</td>
<td>E3</td>
<td>D4</td>
<td>A2</td>
<td>9C</td>
</tr>
<tr>
<td>5</td>
<td>53</td>
<td>D1</td>
<td>00</td>
<td>ED</td>
<td>20</td>
<td>FC</td>
<td>B1</td>
<td>5B</td>
<td>6A</td>
<td>CB</td>
<td>BE</td>
<td>39</td>
<td>4A</td>
<td>4C</td>
<td>58</td>
<td>CF</td>
</tr>
<tr>
<td>6</td>
<td>D0</td>
<td>EF</td>
<td>AA</td>
<td>FB</td>
<td>43</td>
<td>4D</td>
<td>33</td>
<td>85</td>
<td>45</td>
<td>F9</td>
<td>02</td>
<td>7F</td>
<td>50</td>
<td>3C</td>
<td>9F</td>
<td>A8</td>
</tr>
<tr>
<td>7</td>
<td>77</td>
<td>31</td>
<td>A3</td>
<td>40</td>
<td>8F</td>
<td>92</td>
<td>9D</td>
<td>38</td>
<td>F5</td>
<td>BC</td>
<td>B6</td>
<td>DA</td>
<td>21</td>
<td>10</td>
<td>FF</td>
<td>F3</td>
</tr>
<tr>
<td>8</td>
<td>CD</td>
<td>0C</td>
<td>13</td>
<td>EC</td>
<td>5F</td>
<td>97</td>
<td>44</td>
<td>17</td>
<td>C4</td>
<td>A7</td>
<td>7E</td>
<td>3D</td>
<td>64</td>
<td>5D</td>
<td>19</td>
<td>73</td>
</tr>
<tr>
<td>9</td>
<td>60</td>
<td>81</td>
<td>4F</td>
<td>DC</td>
<td>22</td>
<td>2A</td>
<td>90</td>
<td>88</td>
<td>46</td>
<td>EE</td>
<td>B8</td>
<td>14</td>
<td>DE</td>
<td>5E</td>
<td>0B</td>
<td>DB</td>
</tr>
<tr>
<td>A</td>
<td>E0</td>
<td>32</td>
<td>3A</td>
<td>0A</td>
<td>49</td>
<td>06</td>
<td>24</td>
<td>5C</td>
<td>C2</td>
<td>D3</td>
<td>AC</td>
<td>62</td>
<td>91</td>
<td>95</td>
<td>E4</td>
<td>79</td>
</tr>
<tr>
<td>B</td>
<td>E7</td>
<td>C8</td>
<td>37</td>
<td>6D</td>
<td>8D</td>
<td>D5</td>
<td>4E</td>
<td>A9</td>
<td>6C</td>
<td>56</td>
<td>F4</td>
<td>EA</td>
<td>65</td>
<td>7A</td>
<td>AE</td>
<td>08</td>
</tr>
<tr>
<td>C</td>
<td>BA</td>
<td>78</td>
<td>25</td>
<td>2E</td>
<td>1C</td>
<td>A6</td>
<td>B4</td>
<td>C5</td>
<td>E8</td>
<td>DD</td>
<td>74</td>
<td>1F</td>
<td>4B</td>
<td>BD</td>
<td>8B</td>
<td>8A</td>
</tr>
<tr>
<td>D</td>
<td>70</td>
<td>3E</td>
<td>B5</td>
<td>66</td>
<td>48</td>
<td>03</td>
<td>F6</td>
<td>OE</td>
<td>61</td>
<td>35</td>
<td>57</td>
<td>B9</td>
<td>86</td>
<td>C1</td>
<td>1D</td>
<td>9E</td>
</tr>
<tr>
<td>E</td>
<td>E1</td>
<td>F8</td>
<td>98</td>
<td>11</td>
<td>69</td>
<td>D9</td>
<td>8E</td>
<td>94</td>
<td>9B</td>
<td>1E</td>
<td>87</td>
<td>E9</td>
<td>CE</td>
<td>55</td>
<td>28</td>
<td>DF</td>
</tr>
<tr>
<td>F</td>
<td>8C</td>
<td>A1</td>
<td>89</td>
<td>0D</td>
<td>BF</td>
<td>E6</td>
<td>42</td>
<td>68</td>
<td>41</td>
<td>99</td>
<td>2D</td>
<td>0F</td>
<td>B0</td>
<td>54</td>
<td>BB</td>
<td>16</td>
</tr>
</tbody>
</table>

3. $\text{Mix}_M$ applies a left-multiplication by the $n_r \times n_r$ matrix $M$ to all columns of the state, i.e.,

$$\text{Mix}_M : (F_{2^n})^{n_r \times n_r} \to (F_{2^n})^{n_r \times n_r}$$

$$\forall j \in \mathbb{N}_{\leq n}: [x_{n_r,j+1}, \ldots, x_{n_r,j+n_r}]^\top \mapsto M \cdot [x_{n_r,j+1}, \ldots, x_{n_r,j+n_r}]^\top.$$

The Advanced Encryption Standard

The AES comes with three different versions, i.e., AES-128, AES-192 and AES-256. All of them operate on a block length of $n = 128$, but differ in their key length. In particular, AES-128 supports a key length of $\kappa = 128$, AES-192 a key length of $\kappa = 192$, and AES-256 a key length of $\kappa = 256$. The 128-bit state is represented as a $4 \times 4$ array of words of length $s = 8$. As one represents the 8-bit words by elements in the finite field with $2^8$ elements, one has to agree on a particular field representation. In the AES, the field is represented as $F_{2^n} \cong \mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X + 1)$ and thus, each element of the field is given as a polynomial in $\mathbb{F}_2[X]$ of degree lower than 8. In the literature, such a polynomial is usually denoted as a hexadecimal value representing the coefficient vector of the polynomial. For instance, the field element $X + 1$ is denoted by 03. Using this notation, the unkeyed round $R_{S_{\text{AES}}, p, M} : \mathbb{F}_{2^8}^{16} \to \mathbb{F}_{2^8}^{16}$ of all versions of the AES is defined as follows:
S_{\text{SbAES}} (SubBytes). The S-box $S_{\text{AES}}: \mathbb{F}_{2^8} \rightarrow \mathbb{F}_{2^8}$ employed in the S-box layer is given in Table 2.1. It has the algebraic expression\(^{13}\)

$$x \mapsto h(x^{2^8-2}),$$

where $h$ is an affine permutation.

Permute\(_p\) (ShiftRows) operates as a permutation of the words of the state. In particular, it left-rotates the rows of the state by the offset 0, 1, 2 and 3, respectively.

\[
\begin{bmatrix}
  x_1 & x_5 & x_9 & x_{13} \\
  x_2 & x_6 & x_{10} & x_{14} \\
  x_3 & x_7 & x_{11} & x_{15} \\
  x_4 & x_8 & x_{12} & x_{16}
\end{bmatrix}
\mapsto
\begin{bmatrix}
  x_1 & x_5 & x_9 & x_{13} \\
  x_6 & x_{10} & x_{14} & x_2 \\
  x_{11} & x_3 & x_7 & x_{15} \\
  x_{16} & x_4 & x_8 & x_{12}
\end{bmatrix}.
\]

This corresponds to the permutation

$$p = (1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \ 13 \ 14 \ 15 \ 16).$$

Mix\(_M\) (MixColumns). The MDS matrix

$$M = \begin{pmatrix}
  02 & 03 & 01 & 01 \\
  01 & 02 & 03 & 01 \\
  01 & 01 & 02 & 03 \\
  03 & 01 & 01 & 02
\end{pmatrix} \in \text{GL}_4(\mathbb{F}_{2^8})$$

is applied to every of the four columns of the state.

The difference between the three versions of the AES is their particular key-scheduling algorithm and the number of applied rounds. The key-scheduling algorithm of AES-128 takes the 128-bit initial key $k$ and generates eleven 128-bit round keys $k^{(0)}, \ldots, k^{(10)}$. Thus, AES-128 is a 10-round key-alternating cipher. All but the last round of the cipher are the same and as defined above. The only exception is the last round which omits the MixColumns operation. Similarly, the key-scheduling algorithm of AES-192 takes the 192-bit initial key $k$ and generates 13 128-bit round keys $k^{(0)}, \ldots, k^{(12)}$. Thus, AES-192 is a 12-round key-alternating cipher. Again, the MixColumns operation is omitted in the last round. The last version, AES-256, is a 14-round key alternating cipher and the key-scheduling algorithm takes the 256-bit initial key $k$ and generates 15 128-bit round keys $k^{(0)}, \ldots, k^{(14)}$. As for the other versions, the MixColumns operation in the last round is omitted. For more details on the specification of the AES, and particularly for the key-scheduling algorithms, we refer to [PUB01].

\(^{13}\)It was shown in [Nyb94] that this algebraic construction based on inversion in the finite field has strong cryptographic properties. In particular, $p_{S_{\text{SbAES}}} = 2^{-6}$ and $c_{S_{\text{SbAES}}} = 2^{-3}$. It is still an open problem to answer whether there exists a bijective S-box on 8-bit that improves those values.
Active S-boxes in AES-like Ciphers – The Four-Round Propagation Theorem

One of the advantages of the AES-like design is its simple structure. Moreover, if the permutation \( p \) is carefully chosen, one can prove a strong lower bound on the weight of any valid four-round trail. In particular, one obtains that the minimum number of active S-boxes in any valid four-round trail is lower bounded by the square of the branch number of the linear transformation \( x \mapsto Mx \). In the following, we denote the differential branch number of this transformation by \( B_{\text{diff}}^s(M) \) and the linear branch number by \( B_{\text{lin}}^s(M) \), respectively.

**Theorem 2.4** (Four-Round Propagation Theorem (Theorem 9.5.1 in [DR02]). Let \( R_{\text{sb},p,M} \) be an AES-like round with \( n_c \geq n_r \) such that, for each column of the state, \( \text{Permute}_p \) distributes the words of a column to all different columns.

Then, the minimum number of active S-boxes of any valid four-round differential (resp. linear) trail is lower bounded by \( B_{\text{diff}}^s(M)^2 \) (resp. \( B_{\text{lin}}^s(M)^2 \)).

**Proof.** We only show the case of differential trails. Let \( C = (\alpha^{(0)}, \alpha^{(1)}, \alpha^{(2)}, \alpha^{(3)}, \alpha^{(4)}) \in \Gamma_{\text{Mix}_M \circ \text{Permute}_p,s,4}^\text{diff} \). We have to show that \( \text{wt}_s(C) \geq B_{\text{diff}}^s(M)^2 \). As \( \text{Permute}_p \) commutes with \( \text{Sb} \), the four rounds of \( R_{\text{sb},p,M} \) can be written as

\[
\text{Mix}_M \circ \text{Permute}_p \circ \text{Sb} \circ \text{Mix}_M \circ \text{PMP} \circ \text{Sb} \circ \text{Mix}_M \circ \text{Permute}_p \circ \text{Sb} ,
\]

where \( \text{PMP} := \text{Permute}_p \circ \text{Mix}_M \circ \text{Permute}_p \). We thus have for the activity pattern of the trail:

\[
\delta_s(\alpha^{(0)}) = \delta_s(\text{Permute}_p^{-1}(\text{Mix}_M^{-1}(\alpha^{(1)})))
\]

\[
\delta_s(\alpha^{(1)}) = \delta_s(\text{PMP}^{-1}(\text{Permute}_p(\alpha^{(2)})))
\]

\[
\delta_s(\text{Permute}_p(\alpha^{(2)})) = \delta_s(\text{Mix}_M^{-1}(\alpha^{(3)})) .
\]

We can now give a lower bound for the weight of the trail \( C \), by using the branch number of PMP with respect to columns (i.e., \( sn_r \)-bit words), as

\[
\text{wt}_s(C) = w_s(\alpha^{(0)}) + w_s(\alpha^{(1)}) + w_s(\alpha^{(2)}) + w_s(\alpha^{(3)})
\]

\[
= w_s(\alpha^{(0)}) + w_s(\alpha^{(1)}) + w_s(\text{Permute}_p(\alpha^{(2)})) + w_s(\alpha^{(3)})
\]

\[
= w_s(\text{Mix}_M^{-1}(\alpha^{(1)})) + w_s(\alpha^{(1)}) + w_s(\text{Permute}_p(\alpha^{(2)})) + w_s(\alpha^{(3)})
\]

\[
\geq B_{\text{diff}}^s(M) \cdot w_{sn_r}(\alpha^{(1)}) + B_{\text{diff}}^s(M) \cdot w_{sn_r}(\text{Permute}_p(\alpha^{(2)}))
\]

\[
= B_{\text{diff}}^s(M) \cdot (w_{sn_r}(\alpha^{(1)}) + w_{sn_r}(\text{Permute}_p(\alpha^{(2)})))
\]

\[
\geq B_{\text{diff}}^s(M) \cdot B_{\text{diff}}^s(\text{PMP}) .
\]

It is left to show that PMP has a branch number with respect to columns larger or equal to \( B_{\text{diff}}^s(M) \). For this, let \( \alpha \in \mathbb{F}_{2^{n_c}}^{n_c} \neq 0 \). Let further \( \beta = (\beta_1, \ldots, \beta_{n_r,n_c}) = \ldots \)
Permute_p(\alpha), let \gamma = (\gamma_1, \ldots, \gamma_{n_c}) = Mix_M(\beta) and let \delta = Permute_p(\gamma). As \beta \neq 0, there exists an active column in \beta, i.e., there exists a \ j \in \{0, \ldots, n_c - 1\} such that
\[ \delta_{sn_r}(\beta_{n_rj+1}, \ldots, \beta_{n_rj+n_r}) = 1. \]
Because of the definition of the branch number of \ M, we further have
\[ w_s((\beta_{n_rj+1}, \ldots, \beta_{n_rj+n_r})) + w_s((\gamma_{n_rj+1}, \ldots, \gamma_{n_rj+n_r})) \geq B_{s,\text{diff}}(M). \]
Since Permute_p^{-1} distributes the words \beta_{n_rj+1}, \ldots, \beta_{n_rj+n_r} in all different columns and Permute_p distributes the words \gamma_{n_rj+1}, \ldots, \gamma_{n_rj+n_r} in all different columns, we must have at least \ w_s(\beta_{n_rj+1}, \ldots, \beta_{n_rj+n_r}) active columns in \alpha and also at least \ w_s(\gamma_{n_rj+1}, \ldots, \gamma_{n_rj+n_r}) active columns in \delta. In other words,
\[ w_{sn_r}(\alpha) \geq w_s(\beta_{n_rj+1}, \ldots, \beta_{n_rj+n_r}) \text{ and } w_{sn_r}(\delta) \geq w_s(\gamma_{n_rj+1}, \ldots, \gamma_{n_rj+n_r}). \]
Thus, one finally obtains \ w_{sn_r}(\alpha) + w_{sn_r}(\delta) \geq B_s^{\text{diff}}(M). \]

**Corollary 2.5.** Let R_{sb,p,M} be an AES-like round as in Theorem 2.4. Then, the minimum number of active S-boxes of any valid 4t-round differential (resp. linear) trail is lower bounded by \( t \cdot B_s^{\text{diff}}(M)^2 \) (resp. \( t \cdot B_s^{\text{lin}}(M)^2 \)).

### 2.4.3 Computing Active S-boxes with Automatic Tools

The Four-Round Propagation Theorem definitely belongs to the strongest wide-trail arguments we currently know, strengthening confidence in the security of the AES. Its simplicity and elegance inspired several designers to adopt the structure of an AES-like round function. However, in ciphers which do not follow an AES-like design or in cases where the bounds for multiple rounds obtained by iterating the bounds obtained by the Four-Round Propagation Theorem (Corollary 2.5) are not strong enough, the minimum number of active S-boxes has to be analyzed in a different way. In most cases, this is done using computer-aided tools. We explain two common methods, i.e., Matsui’s approach and Mixed-Integer Linear Programming (MILP).

**Matsui’s Approach**

In [Mat95], Section 4, Matsui presented an algorithm for computing the maximum of the product of the single-round differential probabilities over all (non-trivial) differential trails, and the maximum absolute correlation over all (non-trivial) linear trails in DES, respectively. In the context of the wide-trail approach, it is especially useful for computing the minimum weight of all trails in \( \Gamma^{\text{diff}}_{L,s,t} \) or \( \Gamma^{\text{lin}}_{L,s,t} \) for a given number of rounds \( t \). For the case of differential trails, this algorithm is given as Algorithm 2.2 below. Indeed, the designers of the AES-like lightweight cipher Midori utilized this approach for computing the minimum number of active S-boxes in order to find the best choice for the Permute_p operation.
Algorithm 2.2 Matsui’s algorithm for computing $\min_{C \in \Gamma_{L,s,t}^{\text{diff}}} \text{wt}_s(C)$

1: procedure $\text{Matsui}_t((B_0, \ldots, B_{t-1}), \tilde{B}_t)$
2: \hspace{1em} for $\delta^{(0)} \in \{0,1\}^{n_s} \setminus \{0\}$ do
3: \hspace{2em} $\text{MatsuiRecursive}_t((B_0, \ldots, B_{t-1}), 0, \text{wt}_s(\delta^{(0)}), \delta^{(0)})$
4: \hspace{1em} end for
5: $B_t \leftarrow \tilde{B}_t$
6: return $B_t$
7: end procedure
8: 
9: procedure $\text{MatsuiRecursive}_t((B_0, \ldots, B_{t-1}), r, w, \delta^{(r)})$
10: \hspace{1em} if $r = t-1$ then
11: \hspace{2em} $\tilde{B}_t \leftarrow w$
12: \hspace{2em} return
13: \hspace{1em} end if
14: \hspace{1em} for $\delta^{(r+1)}$ s.t. $\exists \alpha^{(r)} \in (\mathbb{F}_2^{s})^{n_s}$ with $\delta^{(r)} = \tilde{\delta}_s(\alpha^{(r)})$ and $\delta^{(r+1)} = \tilde{\delta}_s(L(\alpha^{(r)}))$ do
15: \hspace{2em} if $w + B_{t-r-1} \leq \tilde{B}_t$ then
16: \hspace{3em} $\text{MatsuiRecursive}_t((B_0, \ldots, B_{t-1}), r+1, w + \text{wt}_s(\delta^{(r+1)}), \delta^{(r+1)})$
17: \hspace{2em} end if
18: \hspace{1em} end for
19: end procedure

For $t \in \mathbb{N}$, let

$$B_t = \min_{C \in \Gamma_{L,s,t}^{\text{diff}}} \text{wt}_s(C)$$

denote the minimum number of active S-boxes over all trails in $\Gamma_{L,s,t}^{\text{diff}}$ and let $B_0 := 0$. In order to run $\text{Matsui}_t$ for computing $B_t$, one needs to know $B_{t'}$ for all $t' < t$. In other words, one has to run $\text{Matsui}_{t'}$ for all values of $t'$ smaller than $t$ first. As Matsui stated in his original paper, the values of $B_t$ up to $t = 3$ can often be obtained quite easily, so there might be no necessity to start with computing $B_1$ by Matsui’s algorithm. Further, the algorithm $\text{Matsui}_t$ needs as input an initial value $\tilde{B}_t$ with the requirement that $B_t \geq \tilde{B}_t$. This initial value has to be estimated first. During the execution, the algorithm will dynamically overwrite $\tilde{B}_t$ with tighter estimates for the actual bound $B_t$. It is an example of a branch and bound algorithm. In particular, the activity patterns can be seen to be arranged in a tree, where each node represents a possible activity pattern of the input difference at one particular round and its children are the possible activity patterns for the output differences according to Equation 2.6. The tree is traversed in a depth-first manner and whenever the weight of a (partial) trail, represented by a path in the tree, exceeds $\tilde{B}_t$, the subtree with the current node as a parent can be pruned and has not to be traversed any more. Thus, the preciser the initial guess of $\tilde{B}_t$ is, the more paths of the tree can be pruned in the first place and the more efficient the algorithm gets. Moreover, if the linear layer of
the cipher allows only few possible output differential activity patterns for many of the possible input differences, the algorithm can become very efficient.

**Mixed-Integer Linear Programming (MILP)**

Another way of computing the minimum number of active S-boxes is to model the minimization problem

$$\min_{C \in \Gamma_{\text{diff}}^{L,s,t}} \text{wt}_s(C)$$

as a MILP instance, see, e.g., [MWGP12]. The crucial part is to model the linear constraints that exactly define the branching transitions of the linear layer corresponding to the trails in $\Gamma_{\text{diff}}^{L,s,t}$ (resp. $\Gamma_{\text{lin}}^{L,s,t}$). For efficiency reasons, often a superset of $\Gamma_{\text{diff}}^{L,s,t}$ (resp. $\Gamma_{\text{lin}}^{L,s,t}$) is modelled instead. For instance, in [MWGP12], the authors define the optimization problem by linear constraints that model only the branch number of the linear layer and thus, their bounds might not be tight for the specific linear layer considered. Over the years, lots of progress has been made in order to derive tighter bounds using MILP, especially in the more involved related-key setting (see, e.g., [SHS+13, SHW+14b, SHW+14a]). In the design of Skinny and Mantis, a MILP approach is used in order to derive bounds on the minimum number of active S-boxes, both in the single-key setting and in the related-key setting. Details follow in Chapter 5.

### 2.5 Lightweight Cryptography

Today, the AES can certainly be considered as the most versatile block cipher to be employed in environments with high security standards. By design, it only requires a reasonable implementation overhead for the security it offers. However, there might be situations in which the AES cannot be employed as its operations require too much computational resources. An important field of research that emerged over the last years is lightweight cryptography. Although it is not trivial to give a precise definition of lightweight cryptography, its goal is to establish cryptographic solutions for applications in resource-constrained environments. Especially in the context of the “Internet of Things” (IoT), there is a strong demand for cryptographic algorithms (e.g., block ciphers) that can be implemented on devices with extremely constrained resources, e.g., low chip area or low power supply. Further, especially if the main application of the device is not to provide a cryptographic solution, the cryptographic algorithm should come with negligible additional cost. To give a concrete example from the healthcare sector, consider a pacemaker. Here, the constrained resource is energy as one certainly wants to avoid frequently changing batteries. If a cryptographic algorithm should be implemented in order to prevent manipulating the device from outside, it should require as low energy as possible.

Because of its importance over the last years, NIST is currently running a lightweight cryptography project with the goal of learning more about the needs
from industry and to standardize new lightweight designs. We refer to the report on this project [MBTM17] for further details. We also refer to [BP17] for a comprehensive survey on lightweight cryptography and for a huge list of examples of existing designs. In this thesis, we merely consider lightweight block ciphers, especially their design and analysis.

2.5.1 Lightweight Metrics

A lightweight design is optimized with respect to certain lightweight metrics. Such metrics include chip area, latency, throughput, code size, power or energy consumption. Here, we briefly explain two important lightweight metrics, i.e., chip area and latency, as examples.

Area in Hardware

One of the most considered lightweight metrics is area. If a block cipher should be implemented on a small hardware device, e.g., an RFID tag, only a limited amount of chip area is available. Usually, the hardware area needed for implementation is measured in terms of Gate Equivalents (GE). Thereby, one GE determines the area for implementing a single two-input bitwise NAND gate (i.e., \( x_1 \land x_2 \) for \( x_1, x_2 \in \mathbb{F}_2 \)). One of the first block ciphers that was designed with a focus on reducing area in hardware implementations is Present [BKL+07]. It has a block length of \( n = 64 \) and supports two different key length of \( \kappa = 80 \) and \( \kappa = 128 \), respectively. The design offers very competitive area requirements. In the original paper, the designers were able to implement the 80-bit key length version of their cipher requiring 1570 GE, outperforming several other ciphers including the AES. Present employs a simple bit permutation as its linear layer. The main reason is that a bit permutation can be implemented without any gates, merely as permuting wires between gates.

Whenever the linear layer is designed in a more complex manner, a special focus lies on the number of XOR gates required for its implementation. Thereby, one XOR gate implements the field addition of two elements in \( \mathbb{F}_2 \). Reducing the number of XOR operations is one important design goal that has attracted lots of attention recently. In Chapter 3, we look at this so-called XOR-count in more detail and analyze the efficiency of multiplication in finite fields of characteristic two.

Latency

Another lightweight metric is latency. In case of a block cipher, this defines the amount of time (usually measured in ns) needed for encrypting the message and providing the ciphertext. Low-latency ciphers are especially required in applications where a fast response is crucial. One of the first low-latency block ciphers optimized for hardware implementations is Prince [BCG+12]. Its main feature is the capability of encrypting messages within a single clock cycle requiring only a
reasonable low amount of chip area. For this, the cipher is not implemented in a round-based manner, i.e., when only a single round is implemented and iterated several times. Instead, the rounds are unrolled and the full cipher is implemented at once. For this reason, Prince employs only a small number of rounds. The other beneficial feature of Prince is that decryption can be realized with almost no implementation overhead due to its so-called α-reflection property.

In Section 5.4 we present Mantis, a family of (tweakable) low-latency block ciphers. It is inspired by the design of Prince and the low-energy block cipher Midori, which we explain in Section 2.5.3 below.

2.5.2 Characteristics of Lightweight Block Ciphers

In this section, we briefly explain the characteristics that lightweight block cipher designs usually have in common.

**Sparser Components**

If we consider the AES, it consists of cryptographically strong building blocks. Its S-box is of length \( s = 8 \) bits and has a maximum differential probability and maximum absolute correlation of only \( p_{\text{Sbox}} = 2^{-6} \) and \( c_{\text{Sbox}} = 2^{-3} \), respectively. No cryptographically stronger S-box of eight bits length with respect to those values is known to date. Further, its MixColumns matrix is MDS and its key-scheduling algorithm is quite complex and non-linear. However, those operations may still be too expensive for applications in extremely constrained environments. For instance, the S-box has to be either implemented as an inversion in the finite field \( \mathbb{F}_{2^8} \) or by storing a look-up table of size \( 8 \times 2^8 \) bits. Further, due to its density, the MDS MixColumns matrix might as well be too expensive to be implemented.

Therefore, in lightweight cryptography, designers focus on tailoring the cryptographic components to a minimum. For instance, a four-bit S-box requires less area than an eight-bit S-box, a sparse linear layer can be implemented more efficiently than an MDS matrix. For that reason, in lightweight block ciphers, one often has S-boxes of length \( s = 4 \). In cases where still eight-bit S-boxes are employed, they are usually constructed by smaller building-blocks, e.g., four-bit S-boxes, as done in [CDL16]. In order to minimize XOR operations, the linear layer is often very sparse and therefore only offers a limited amount of diffusion.

**More Rounds**

The 128-bit key length version of the AES only needs ten rounds to strengthen against cryptanalytic attacks. The main reason for that is the employment of cryptographically strong building blocks. One can think about the following extreme: The non-linear operation within a cipher round consists of just a single bitwise AND and the round key is introduced by a single bitwise XOR. In order to obtain a secure cipher, many rounds have to be applied and the actual secu-
rity would come from iterating a very weak function a lot. However, this would significantly increase the latency in a round-based implementation. Therefore, designers of lightweight cryptographic algorithms are usually aiming to find the best trade-off between lightweightness of the rounds and the number of rounds that have to be applied. This, of course, depends on the particular lightweight metric one wants to optimize. For low-latency requirements, the number of rounds has to be quite low.

**Simple Key Schedules**

As the building blocks of lightweight designs are tailored to a minimum, several designs also employ a very simple, often linear, key schedule. One of the most simple kind of key schedules one can think of is the usage of identical round keys in every round. However, in order to avoid slide attacks [BW99], the (keyed) rounds $R_i$ of a block cipher should not be all the same. Therefore, in the simplest practical key schedule, pre-defined and public round constants $c_0, \ldots, c_t \in \mathbb{F}_2$ are added to the round keys. In other words, the key-scheduling algorithm takes the initial key $k$ and derives the round keys of the form

$$k \mapsto (k + c_0, k + c_1, \ldots, k + c_t).$$

Examples for schemes that employ such a kind of key schedule are LED, Midori, Noekeon [DPVAR00] and Prince. Such a simple and linear key schedule may look suspicious at a first sight, especially as we analyze the security of ciphers under the assumption of independent round keys. Kranz, Leander and Wiemer recently analyzed this key schedule in [KLW17] and found out that, fortunately, for every linear approximation, the average variance of the correlation over all possible round constants is the same as the variance for independent round keys as stated in Corollary 2.3. This gives indication for the soundness of employing simple linear key schedules and choosing random round constants.

However, studying the security of key-alternating ciphers with a key-scheduling algorithm of the form $k \mapsto (k + c_0, k + c_1, \ldots, k + c_t)$ remains still of significant importance in lightweight cryptography. In particular, several lightweight block ciphers with such a key-scheduling algorithm have recently been broken by so-called invariant attacks, i.e., the invariant subspace attack [LAAZ11] or the nonlinear invariant attack [TLS16]. Chapter 6 deals with those attacks in more detail.

---

14 Such an example was described by Bogdanov at the summer school on “Design and Security of Cryptographic Functions, Algorithms and Devices” in Albena, Bulgaria in 2013. He claimed that several thousand of those rounds are required for obtaining a secure cipher. The slides can be found at [https://www.cosic.esat.kuleuven.be/summer_school_albena/slides/Andrey_lightweight-bc.pdf](https://www.cosic.esat.kuleuven.be/summer_school_albena/slides/Andrey_lightweight-bc.pdf) (accessed: December 5, 2017).

15 Several variants are possible. For instance, sometimes two different round keys $k'$ and $k''$ are derived from the initial key $k$ and are employed in every second round, i.e., $k \mapsto (k' + c_0, k'' + c_1, k' + c_2, \ldots)$. 

44
Innovative Designs

By extremely focussing on efficiency and performance, sometimes designers come up with innovative constructions that deviate from well-known cipher designs. One of the most prominent examples of an innovative block cipher design is the NSA cipher Simon, as mentioned in Example 2.2. It was designed for having competitive performance on a variety of platforms, making it a flexible lightweight cipher for the IoT. Although Simon is a Feistel cipher, it is neither based on S-boxes nor on an ARX construction. Moreover, it employs an innovative key schedule. The drawback of an innovative design is that it may be harder to analyze and may lead to new, dedicated, attacks. Indeed, the NSA did not publish any design rationale of their cipher from a cryptographic viewpoint and only explained their considerations with respect to performance (see also \cite{BSS15}). This, implicitly, left the task of cryptanalysis to third researchers. In fact, several papers that focussed on analyzing Simon appeared since its introduction in 2013. Lots of those papers employed dedicated, computer-aided arguments for analyzing the cipher with respect to standard attacks. However, no serious threats have been found so far and, still, Simon offers a reasonable security margin based on existing cryptanalysis. In Chapter 7 we also take a look at Simon and derive a security argument on the resistance against differential attacks that does not rely on computer-aided tools, i.e., that can be verified by hand.

Chapter 5 explains the block cipher Skinny which was designed with the motivation of having a cipher that offers a competitive level of performance but comes with security arguments against standard cryptanalytic attacks by design, especially in the more involved related-key setting.

2.5.3 Midori

Midori is a lightweight block cipher published by Banik et al. in 2015 \cite{BBI15}. It was especially designed for achieving low energy consumption. It comes in two different versions, i.e., Midori-64 and Midori-128, to support block lengths of $n = 64$ and $n = 128$, respectively. In both versions, the key length is $\kappa = 128$. The overall design is inspired by the AES which makes it belonging to the class of AES-like ciphers as described in Section 2.4.2. In particular, the building blocks of the rounds are much simpler than those of the AES with the purpose of minimizing energy consumption. For instance, the implementation of the multiplicative structure of the finite field $\mathbb{F}_2^*$ is avoided by only using the trivial elements 0 and 1 as coefficients in the MixColumn matrix. Moreover, the cipher employs the typical “lightweight” key-scheduling algorithm of generating basically the same round key and adding pre-defined round constants. As we are going to consider Midori-64 as a case study for some of our results, i.e., finding the best word permutation layers (Chapter 4) and studying invariant attacks (Chapter 6), we briefly explain the design of Midori-64 in the following. Moreover, the low-latency block cipher Mantis, which we explain in Section 5.4, is to a large extend inspired by the design of Midori. For more details and the exact specification of Midori-64 and Midori-128,
we refer to the original design document.

**The Round Function of Midori-64**

The unkeyed round function \( R_{\text{SbMid64},p,M} : \mathbb{F}_2^{16} \to \mathbb{F}_2^{16} \) of Midori-64 fulfills the definition of an AES-like round as given in Definition 2.9. In particular, the state is represented by a \( 4 \times 4 \) array of words of length \( s = 4 \) as

\[
\begin{bmatrix}
  x_1 & x_5 & x_9 & x_{13} \\
  x_2 & x_6 & x_{10} & x_{14} \\
  x_3 & x_7 & x_{11} & x_{15} \\
  x_4 & x_8 & x_{12} & x_{16}
\end{bmatrix}
\]

The round function of Midori-64 is composed of the following consecutive operations. It is iterated 16 times with a round key addition in between.

1. \( S_{\text{SbMid64}} \) (SubCell). The involutory S-box \( \text{SbMid64} : \mathbb{F}_2^4 \to \mathbb{F}_2^4 \) as given in Table 2.2 is employed in the S-box layer. Note that in Midori one does not need the multiplicative structure of the finite field \( \mathbb{F}_2^4 \). Therefore, we simply give the S-box as a mapping \( \mathbb{F}_2^4 \to \mathbb{F}_2^4 \). A four-bit vector is represented in hexadecimal notation, e.g., \((0, 0, 1, 0) = 2\).

Table 2.2: The 4-bit S-box \( \text{SbMid64} \) used in Midori-64.

<table>
<thead>
<tr>
<th>( x )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{SbMid64}(x) )</td>
<td>C</td>
<td>A</td>
<td>D</td>
<td>3</td>
<td>E</td>
<td>B</td>
<td>F</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

2. Permute \( p \) (ShuffleCell) operates as a permutation of the words of the state as follows:

\[
\begin{bmatrix}
  x_1 & x_5 & x_9 & x_{13} \\
  x_2 & x_6 & x_{10} & x_{14} \\
  x_3 & x_7 & x_{11} & x_{15} \\
  x_4 & x_8 & x_{12} & x_{16}
\end{bmatrix}
\]

\[
\mapsto
\begin{bmatrix}
  x_1 & x_{15} & x_{10} & x_{8} \\
  x_{11} & x_5 & x_4 & x_{14} \\
  x_6 & x_{12} & x_{13} & x_3 \\
  x_{16} & x_2 & x_7 & x_9
\end{bmatrix}
\]

This corresponds to the permutation

\[
p = \begin{pmatrix}
  1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
  1 & 11 & 16 & 15 & 5 & 12 & 2 & 10 & 4 & 13 & 7 & 8 & 14 & 3 & 9
\end{pmatrix}
\]

3. Mix \( M \) (MixColumn). The involutory matrix

\[
M = \begin{pmatrix}
  0 & 1 & 1 & 1 \\
  1 & 0 & 1 & 1 \\
  1 & 1 & 0 & 1 \\
  1 & 1 & 1 & 0
\end{pmatrix}
\in \text{GL}_4(\mathbb{F}_2)
\]

is applied to every of the four columns of the state.

---

\(^{16}\)The last round omits the linear layer.
The Key-Scheduling Algorithm of Midori-64

Midori-64 takes the initial 128-bit key $k$ and splits it into two 64-bit keys $\xi_0$ and $\xi_1$, thus $k = \xi_0 \parallel \xi_1$. Then, we have for the whitening keys, $k_0 = k_{16} = \xi_0 + \xi_1$, and for the other round keys, $k_i = \xi_{(i-1) \mod 2} + c_i$, where the $c_i$ are pre-defined round constants. For the precise definition of the round constants, we refer to the original design paper. It is worth remarking that the round constants are all contained in the set $\{(0, 0, 0, 0), (0, 0, 0, 1)\}^k$, i.e., in each word of the state only the least significant bit is affected by the round constant addition.

On the ShuffleCell Permutation

One major contribution of the designers was the observation that the usage of a word permutation $\text{Permute}_p$ that is different from the AES ShiftRows operation, in combination with a non-MDS MixColumns matrix, may lead to a higher number of active S-boxes as one would expect by iterating the bounds obtained by the Four-Round Propagation Theorem (Theorem 2.4). Indeed, the designers were able to increase the minimum number of active S-boxes, e.g., for six rounds from 20 (i.e., $4 \cdot 4$ active S-boxes for four rounds plus 4 active S-boxes for two rounds) to 30, by using the permutation as described above. Those bounds were obtained by automatic search tools and unfortunately, there is not much theoretical understanding why the bounds improve that significantly. In Chapter 4, we analyze possible alternatives for the ShuffleCell permutation in Midori.

Invariant Attacks on Midori-64

Several external cryptanalysis has already been conducted on Midori. One particular threat is that Midori-64 is vulnerable to invariant attacks, see [GJN+16, TLS16]. In particular, [TLS16] showed that there exists a fraction of $2^{-64}$ weak keys for which the keyed instance can be distinguished from a random permutation using only very few plaintext/ciphertext pairs. Chapter 6 studies the invariant attack in more detail.
Part I

Design of Lightweight Linear Layers
Chapter 3

Lightweight Linear Layers based on Finite Field Multiplications

This chapter is based on the publication [BKL16] which is joint work with Thorssten Kranz and Gregor Leander. All authors equally contributed. The main part of the author was Section 3.2 and 3.3 i.e., studying the efficiency of multiplications in finite fields of characteristic two.

3.1 Introduction

Many block cipher designs, including the AES or other AES-like ciphers, build on finite fields as their underlying mathematical structure. In most cases, those ciphers can be designed without having to specify a concrete representation of the finite field in advance. However, when the cipher is finally being implemented in practice, one necessarily has to choose a particular representation of the finite field, basically as binary strings for its elements. For instance, in the case of the AES, the finite field with $2^8$ elements is represented as $\mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X + 1)$, see Section 2.4.2. In general, this choice does not influence the security of the cipher, but might heavily influence the performance of the resulting implementation. In this chapter we focus on this choice of field representations and evaluate how to choose an optimal representation with respect to multiplication with fixed field elements.

When applying an MDS matrix in the linear layer of a block cipher design, the main challenge is to choose an MDS matrix that is most suitable for allowing an
efficient implementation. In particular, as those MDS matrices are usually defined over a finite field with characteristic two, i.e., $\mathbb{F}_2$, one important question is how the choice of a particular $\mathbb{F}_2$-basis of $\mathbb{F}_2$, impacts the implementation efficiency.

From a design point of view, one thus has to choose a linear layer given as a mapping on $\mathbb{F}_2^n$ and an $\mathbb{F}_2$-basis of $\mathbb{F}_2$ to concretely specify the primitive. This is actually a very natural separation of the design of the cipher and its specification (and thus implementation) on bit level. As nicely explained in [DR11] by introducing RINDEL-GF, this separation is probably most obvious for the AES itself, but in principle possible for any cipher. Following [DR11], the choice of basis is to a large extent independent of the design and the security of the cipher. However, the choice of basis might have a significant impact on the efficiency of the cipher on certain platforms.

For software implementations, depending on the details, the choice of basis is either irrelevant (in e.g. a table-based implementation) or hard to capture (in e.g. a bit-sliced implementation) as the efficiency might depend on the exact instructions offered by a given platform. For hardware implementations, one has to distinguish between a serial implementation or a round-based implementation. As the round-based implementation seems most relevant in practice (see [SKO15]), we mainly focus on this use-case here.

For a round-based hardware implementation, the impact of the choice of basis already becomes apparent when focusing on how to implement the multiplication with one given element $\alpha$ in $\mathbb{F}_2$. For different choices of bases, the efficiency of implementations of the resulting $\mathbb{F}_2$-linear mappings differs significantly. Thus, the very fundamental task we study in the first part of this chapter is:

*For a given element $\alpha \in \mathbb{F}_2$, find a basis such that multiplication by $\alpha$ can be implemented most efficiently.*

While the above question is of independent interest, with potentially very different applications, we use our results for designing efficient linear layers. Thus, in the second part of this chapter, we will give several constructions of MDS matrices. Echoing the above, the construction of our MDS matrices are independent of the choice of the basis – actually to a large extent independent of the field size as well.

The combination of the first part, i.e., how to choose a basis that allows for an optimal implementation, and the second part, i.e., the construction of MDS matrices, finally results in implementations of MDS matrices that are very efficient for a large variety of parameters. This application serves as a nice example where an improved understanding on how to choose the field representation immediately leads to improved results. This is even more interesting as the construction of efficient MDS matrices has been an active field of research recently.

**Related Work**

In particular the construction of efficient serial MDS matrices is a well-studied subject. Considering serial implementations of MDS matrices is based on the initial idea of Guo, Peyrin, and Poschmann used in the design of PHOTON [GPPII] and
later in the block cipher LED [GPPR11]. In a nutshell the idea is not to implement an MDS matrix directly, but rather implement a matrix $A$ such that $A^l$ is MDS for some small $l$. When considering a hardware implementation, it reduces the chip area if implementing $A$ is significantly cheaper than $A^l$. The circuit implementing $A$ is then iterated $l$ times, which does not increase its size significantly. This basic idea has been further generalized and improved in a series of subsequent papers. In [SDMS12] and [WWW13] the authors focus on even more efficient choices for $A$ by considering $\mathbb{F}_2$-linear MDS codes. Their approach uses symbolic computations in order to derive general conditions on how to choose the matrix entries independent of the dimension.

In [XZL14], Xu et al. furthermore took into account the cost of implementing the inverse matrix. In [AF15], Augot and Finiasz improved significantly upon the efficiency of the search algorithm of [SDMS12], allowing them to search for MDS matrices of much larger dimension than previously possible.

For the case of round-based implementations, the authors of [SKOP15] focus on MDS matrices that have an efficient implementation (in terms of the number of XOR operations needed) and put special emphasis on involutory MDS matrices, i.e., MDS matrices that are their own inverse. They derive several constructions and rather efficient search methods for MDS matrices meeting their goals. Liu and Sim [LS16] improved upon some of those results by characterizing equivalences in circulant (and circulant-like) MDS matrices and thus further reduced the search space. In both works, in order to improve the efficiency for a given MDS matrix defined over a finite field, the authors considered different representations of the underlying finite fields by running through all possible irreducible polynomials of the given degree. However, in view of the question of how to choose an optimal basis, this corresponds to investigating only a small subset of all possible bases. Work on investigating the XOR-count distribution for other than the polynomial bases has also been done in [SS16a].

Li and Wang constructed circulant involutory $\mathbb{F}_2$-linear MDS matrices [LW16]. While it was already known that circulant MDS matrices over a finite field cannot be involutory [GR14], they have shown their existence in this more general case. Independently, the authors of [LS16] have shown the existence of left-circulant involutory MDS matrices over finite fields.

Recently, Sarkar and Syed pointed out how lightweight linear layers could be constructed from Toeplitz matrices and constructed MDS matrices with optimized XOR-count [SS16b]. As previous work, they only considered field representations given by a polynomial basis. In [JPST17], the authors introduced the s-XOR metric which allows to reuse intermediate results for computing the XOR operations. They investigated the s-XOR count for all elements of the finite fields $\mathbb{F}_{2^4}$ and $\mathbb{F}_{2^8}$ under all possible polynomial bases and were able to derive some of the lightest MDS matrices known to date using an improved search tool. We adopted this s-XOR metric for our purposes of optimizing finite field multiplications with fixed elements.

It is important to remark that all of those results on finding the lightest MDS
matrices – as well as our approach on constructing lightweight MDS matrices – are based on local optimizations. In a matrix-vector multiplication, every coordinate of the result is computed as the sum over multiplications in the finite field. In local optimizations, the XOR operations needed to sum up the results of the multiplications are considered as a fixed part and only the overhead for computing the multiplications in the finite field are optimized. Recently, Kranz et al. focussed on globally optimizing the implementation cost for those locally-optimized matrices [KLSW17]. Using well-known heuristic algorithms for finding the shortest linear straight-line program, they were able to significantly improve the implementation. Thus, their results suggest that future research should focus on globally optimizing the implementation cost of linear layers.

Results of this Chapter

After introducing notation and recalling some basics in Section 3.2 in Section 3.3 we study the question of how to find an optimal implementation of the multiplication by a given field element \( \alpha \). Here, efficiency is measured in terms of the number of binary additions (aka. XOR operations) needed to implement the corresponding binary matrix. Note that this metric corresponds to the s-XOR metric as introduced in [JPST17] and differs from the XOR-count metric used in [KPPY14] and [SKOP15]. In those two (and many other) papers, the XOR-count of an \( s \times s \) matrix \( M \) is defined as the number of ones in \( M \) minus \( s \). However, the number of (additional) ones in a matrix does not necessarily correspond to the number of XOR operations needed for implementation. Thus, while the number of ones in \( M \) is certainly an easier to handle metric, it is more appropriate to consider the actual number of XOR operations as the efficiency metric. For technical reasons, we focus on the number of XOR operations without using temporary registers, i.e., in-place XOR operations. One of our main results in this first part is that for a non-trivial element \( \alpha \), one can find a basis such that the matrix corresponding to multiplication with \( \alpha \) can be implemented with one single XOR operation if and only if the characteristic polynomial of \( \alpha \) is an irreducible trinomial, i.e., an irreducible polynomial with exactly three non-zero coefficients. Note that in our notion, an XOR-count equal to one coincides with the definition of the XOR-count in [KPPY14] and [SKOP15]. The interesting part here is that the condition on the characteristic polynomial is not only sufficient but also necessary. As an immediate consequence, one cannot hope to implement the multiplication by any element \( \alpha \neq 1 \) in \( \mathbb{F}_{2^8}^* \) with one binary addition only. This follows by the above and the well-known fact that there do not exist irreducible trinomials of degree 8 [Swa62].

We further show that, for any given basis, there are at most two (non-trivial) elements \( \alpha \) and \( \beta \) such that the multiplication with those elements can be implemented with one XOR operation. In fact, \( \beta \) is necessarily the multiplicative inverse of \( \alpha \).

While the weight of the (irreducible) characteristic polynomial of an element \( \alpha \) clearly gives an upper bound of the number of XOR operations needed to im-
plement the corresponding multiplication, we show that this bound is in general not tight in the case where the characteristic polynomial is of weight larger than three.

In particular, for all elements $\alpha \in \mathbb{F}_{2^s}^*$ with $4 \leq s \leq 8$ we present an optimal representation such that the multiplication with $\alpha$ can be implemented with a minimal number of XOR operations. For all those elements $\alpha$, that are not contained in a proper subfield of $\mathbb{F}_{2^s}$, the multiplication can be implemented with at most three XOR operations (and often with two only). Those results are given in Tables 3.3 to 3.7 and cover the cases which are most relevant for symmetric cryptography in practice. Interestingly, and maybe counter-intuitive, multiplication with non-trivial elements in a proper subfield turns out to be among the most expensive in all the cases explored here.

Moreover, for all $2 \leq s \leq 2048$ for which no irreducible trinomial of degree $s$ exists, we present one element $\alpha \in \mathbb{F}_{2^s}$ such that multiplication by $\alpha$ requires two XOR operations, see Table 3.8. Those results are proven optimal by the above mentioned necessary and sufficient condition.

In the second part, i.e., Section 3.4, we present several (circulant) matrices. Entries in those matrices are represented as powers of a generic field element $\alpha$. By symbolically computing all minors, i.e., the determinants of all square submatrices, we derive a list of polynomials in $\mathbb{F}_2[\alpha]$. Now, whenever $\alpha$ is chosen such that it is not a root of any of those polynomials, the matrix is MDS. One nice consequence of this approach is that, as the degree of those polynomials is limited, our matrices are MDS for almost all elements in $\mathbb{F}_{2^s}$ as soon as $s$ is large enough, i.e. larger than the maximal degree of those polynomials.

Finally, the first and second part are combined in Section 3.4.2 to result in very efficient MDS matrices in terms of the XOR-count. A summary of our results and comparison with other work is given in Table 3.1 and Table 3.2 respectively. The main observation here is that if multiplication by $\alpha$ can be implemented with $t$ XOR operations, then multiplication by $\alpha^\pm i$ for $i \geq 0$ can be implemented with at most $t \cdot i$ XOR operations. Thus, by simply minimizing the sum of the (absolute) exponents for our circulant MDS matrices, we immediately reduce the XOR-count.

As an interesting side result, we like to point out that the XOR-count per bit actually decreases with increasing field size. For example, our $4 \times 4$ MDS matrices have a per bit XOR-count of $3 + \frac{2}{s}$, or $3 + \frac{6}{s}$ in the case that no irreducible trinomial of degree $s$ exists.

Thus, even though reducing the number of XOR operations has already received considerable attention recently, this part nicely shows that our improved understanding of how to choose an optimal basis allows us to easily improve upon known constructions. Note that such improvements are possible independent from which XOR-count definition is used, i.e., we were able to improve existing results

\footnote{It is exactly this part where considering only in-place XOR operations becomes very helpful, as otherwise multiplication by $\alpha$ and by $\alpha^{-1}$ might differ in their XOR-count.}

\footnote{This is also true for the constructions given in [WWW13], but does not hold for the subfield (or code-interleaving) construction.}
also in the simple XOR-count definition by changing the basis. For example, we found an element in \( \mathbb{F}_{2^8} \) with only two additional non-zero entries in its matrix, which directly improves the results of \cite{SKOP15}.

Finally, in Section 3.5 we give a perspective on \( \mathbb{F}_{2^s} \)-linear MDS matrices. In particular, we point out that while there exists no \( \alpha \in \mathbb{F}_{2^8} \) (resp. \( \mathbb{F}_{2^{13}}, \mathbb{F}_{2^{16}} \)) which can be implemented with only one XOR operation, there does exist an \( 8 \times 8 \) (resp. \( 13 \times 13, 16 \times 16 \)) binary matrix, that can be used in place for the multiplication by \( \alpha \) in the above mentioned \( 4 \times 4 \) matrix to result in an additive MDS matrix with reduced cost.\(^4\) Again, the idea of considering the entries of the matrix as powers of a single field element is beneficial as the conditions for the matrix to be MDS remain basically unchanged.

We then conclude by pointing to some interesting questions for future investigations.

## 3.2 Preliminaries

Although there exists up to isomorphism only one finite field for every possible order, we are interested in the specific representation. For instance, if \( Q \in \mathbb{F}_2[X] \) is an irreducible polynomial of degree \( s \), then \( \mathbb{F}_{2^s} \cong \mathbb{F}_2[X]/(Q) \) where \((Q)\) denotes the ideal generated by \( Q \).

We first recall some basics about finite fields and matrix representations. For more background the reader is referred to, e.g., \cite[Section 2.5]{LN94} and \cite{War94}. Let \( V \cong K^s \) be a finite-dimensional vector space over the field \( K \). Every linear mapping \( f: V \to V \) can be described as \( v \mapsto A_B v \) by a left-multiplication with a matrix \( A_B \in \text{Mat}_s(K) \). This representation is dependent on the choice of the basis \( B \) for \( V \). For instance, if \( B = \{b_1, \ldots, b_s\} \), the \( j \)-th column of \( A_B \) consists of the coefficients \( a_{1,j}, \ldots, a_{s,j} \) of \( f(b_j) = \sum_{i=1}^s a_{i,j} b_i \). Thus, changing the basis from \( B \) to \( B' \) results in a different matrix representation of \( f \). This transformation is called the change of basis transformation, which is simply a conjugation of \( A_B \). Thus, \( A_{B'} = T A_B T^{-1} \) using an invertible matrix \( T \). In this case, \( A_B \) and \( A_{B'} \) are called similar (resp. permutation-similar if \( T \) is a permutation matrix).

There is a natural way of representing the elements in a finite field with characteristic \( p \) as vectors with coefficients in \( \mathbb{F}_p \). In the following, we consider the representation of the multiplication by \( \alpha \) by a matrix as described in the following diagram.

\[ \begin{array}{ccc}
\mathbb{F}_{2^s} & \xrightarrow{-\alpha} & \mathbb{F}_{2^s} \\
\Phi_B & & \Phi_B^{-1} \\
\mathbb{F}_2^s & \xrightarrow{M_{\alpha,B}} & \mathbb{F}_2^s
\end{array} \]

\(^4\)Note that the authors of \cite{LW16} constructed a similar \( 32 \times 32 \) \( \mathbb{F}_2 \)-linear MDS matrix.
The bijection $\Phi_B$ maps elements $\alpha \in \mathbb{F}_2$ to its vectorial representation over $\mathbb{F}_2$ with regard to a basis $B$ (and $\Phi_B^{-1}$ vice versa). $M_{\alpha,B}$ denotes the $s \times s$ matrix representing (left-) multiplication by the element $\alpha$. For different bases $B$ and $B'$, one can obtain $M_{\alpha,B'}$ from $M_{\alpha,B}$ by the change of basis transformation, in particular $M_{\alpha,B'} = TM_{\alpha,B}T^{-1}$ for an invertible $T$. We denote similarity of matrices with the relation symbol $\sim$, (resp. $\sim_\pi$ for permutation-similarity). The characteristic polynomial of a matrix $A$ is defined as $\chi_A := \det(\lambda I - A) \in \mathbb{F}_2[\lambda]$ and the minimal polynomial is denoted by $m_A$. Recall that the minimal polynomial is the (monic) polynomial $P$ of least degree, such that $P(A) = 0$. It is a well-known fact that the minimal polynomial divides the characteristic polynomial, thus $\chi_A(A) = 0$. As the minimal polynomial and the characteristic polynomial are actually properties of the underlying linear mapping, similar matrices have the same characteristic and the same minimal polynomial. A special type of matrix, that will play an important role in the following is the companion matrix of a polynomial.

**Definition 3.1.** For a monic polynomial

$$Q = X^d + \pi_{d-1}X^{d-1} + \cdots + \pi_1X + \pi_0 \in \mathbb{F}_2[X]$$

of degree $d$, the companion matrix of $Q$ is defined as the $d \times d$ matrix

$$C_Q := \begin{pmatrix} 0 & \pi_0 \\ 1 & \pi_1 \\ \vdots & \vdots \\ 1 & \pi_{d-2} \\ \vdots & \vdots \\ 1 & \pi_{d-1} \end{pmatrix}.$$

It is known from linear algebra that the characteristic polynomial and the minimal polynomial of $C_Q$ are equal to $Q$ itself, i.e., $\chi_{C_Q} = m_{C_Q} = Q$. In addition, any matrix $A$ is similar to a companion matrix if and only if its characteristic polynomial coincides with its minimal polynomial. In particular, $C_Q$ is exactly the rational canonical form of $A$ in this case (see Proposition 6.10 in Chapter 6).

### 3.2.1 The XOR-Count and the Cycle Normal Form

The XOR-count of a field element was already studied in [KPPY14] and [SKOP15]. In their formal definition, a matrix $A \in \text{GL}_s(\mathbb{F}_2)$ has an XOR-count of $t$ if and only if $A$ can be written as a permutation matrix with $t$ additional non-zero entries. Formally, $A = P + \sum_{k=1}^t E[i_k,j_k]$ and $w(A) = s + t$. Here, $P$ is a permutation matrix and $E[i_k,j_k]$ denotes a binary $s \times s$ matrix which consists of all zeros, except in the $i_k$-th row of the $j_k$-th column. Although all matrices of that structure can be implemented with at most $t$ XOR operations (not necessarily without temporary registers), the construction does not contain all possible matrices which are realizable with at most $t$ XOR operations. For instance, there are matrices with three additional non-zero entries such that the result of their defining linear
function can be computed with just two additions. As an example, consider
\[
\begin{pmatrix}
1 & 0 & 1 \\
1 & 1 & 1 \\
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
v_1 \\
v_2 \\
v_3
\end{pmatrix}
= \begin{pmatrix}
v_1 + v_3 \\
(v_1 + v_3) + v_2 \\
v_3
\end{pmatrix}.
\]

In the following, we consider an alternative definition which includes the cases described above. Note that this corresponds to the notion of the s-XOR metric, which was first introduced in [JPST17].

**Definition 3.2.** A matrix \( A \in \text{GL}_s(\mathbb{F}_2) \) has an XOR-count of \( t \), denoted \( w_\oplus(A) = t \), if \( t \) is the minimal number such that \( A \) can be written as
\[
A = P \prod_{k=1}^{t} (I + E^{[i_k,j_k]})
\]
for a permutation matrix \( P \) and such that \( i_k \neq j_k \) for all \( k \).

Note that if a matrix can be represented in the form \( P \prod_{k=1}^{t} (I + E^{[i_k,j_k]}) \), the number of factors \( (I + E^{[i_k,j_k]}) \) clearly gives an upper bound on the actual XOR-count. It is worth pointing out that the definition above just counts the number of XOR operations without using temporary registers. Those are technically somewhat easier to handle. However, this restriction does not make a difference for matrices with XOR-count less or equal to two, which we are most concerned about in in the following. In general, allowing temporary registers might well reduce the number of XOR operations needed for an implementation.

This definition of the XOR-count coincides with the one from [KPPY14] for the case that \( t = 1 \), i.e., for matrices of XOR-count 1. For other cases, the number of additional non-zero entries can increase. We will often consider \( t = 2 \) within this chapter. By evaluating the product, it follows that any \( A \) with \( w_\oplus(A) = 2 \) is of the form
\[
A = \begin{cases}
P + P(E^{[i_1,j_1]} + E^{[i_2,j_2]}) & \text{iff } i_2 \neq j_1 \\
P + P(E^{[i_1,j_1]} + E^{[i_2,j_2]} + E^{[i_1,j_2]}) & \text{iff } i_2 = j_1.
\end{cases}
\]
The XOR-count is invariant under permutation-similarity. Moreover, naturally in the setting not allowing temporary registers, the XOR-count is invariant under taking the inverse. This is summarized and formally proven in the following lemma and corollary.

**Lemma 3.1.** If \( A \sim_\pi A' \), then \( w_\oplus(A) = w_\oplus(A') \).

**Proof.** Let \( A' = QAQ^{-1} \) where \( Q \) is the permutation matrix representing the permutation \( \sigma \in S_s \). Let \( I + E^{[i_k,j_k]} \) be a factor in the XOR-count representation of \( A = P \prod_{k=1}^{t} (I + E^{[i_k,j_k]}) \), where \( t = w_\oplus(A) \). Then the following identity holds:
\[
(I + E^{[i_k,j_k]})(Q^{-1} + E^{[i_k,\sigma^{-1}(j_k)]}) = Q^{-1}(I + E^{[\sigma^{-1}(i_k),\sigma^{-1}(j_k)]}).
\]
One is able to commute $Q^{-1}$ to the front before the first factor by proceeding for all of the $t$ factors and finally obtain

$$A' = QPQ^{-1} \prod_{k=1}^{t} (I + E^{[\sigma^{-1}(i_k), \sigma^{-1}(j_k)]}).$$

It follows that $w_\oplus(A') \leq w_\oplus(A)$. By reverting the above steps we obtain $w_\oplus(A) \leq w_\oplus(A')$.

**Corollary 3.1.** If $w_\oplus(A) = t$, then also $w_\oplus(A^{-1}) = t$.

**Proof.** We show that $A^{-1}$ is permutation-similar to a matrix with an XOR-count of $t$.

$$\left(P \prod_{k=1}^{t} (I + E^{[i_k,j_k]})\right)^{-1} = \prod_{k=t}^{1} (I + E^{[i_k,j_k]}) P^{-1} \prod_{k=t}^{1} (I + E^{[i_k,j_k]})$$

Later, we would like to be able to exhaustively search over all matrices with low XOR-count for a given dimension $s$. Since the number of permutation matrices (which is $s!$) rapidly increases with $s$, an exhaustive search will quickly become infeasible if we do not restrict the structure of $P$. By a well-known fact from combinatorics, one is able to assume $P$ to be in a specific form.

**Lemma 3.2.** For any permutation matrix $P$ of dimension $s$, it is

$$P \sim_\pi \bigoplus_{k=1}^{d} C_{X^{m_k+1}}$$

for some $m_k$ with $\sum_{k=1}^{d} m_k = s$ and $m_1 \geq \cdots \geq m_d \geq 1$.

**Proof.** It is well known that two permutations with the same cycle type are conjugate [DF04, Chapter 4.3, Proposition 11]. That is, given the permutations $\sigma, \tau \in S_s$ as

$$\sigma = (o_1, o_2, \ldots, o_{d_1})(o_{d_1+1}, \ldots, o_{d_2}) \cdots (o_{d_{m-1}+1}, \ldots, o_{d_m})$$

$$\tau = (t_1, t_2, \ldots, t_{d_1})(t_{d_1+1}, \ldots, t_{d_2}) \cdots (t_{d_{m-1}+1}, \ldots, t_{d_m})$$

in cycle notation, one can find a $\pi \in S_s$ such that $\pi \sigma \pi^{-1} = \tau$. This $\pi$ operates as a relabeling of indices.

Let $\sigma$ in the form above be the permutation defined by $P$. Now, there exits a permutation $\pi$ such that $\pi \sigma \pi^{-1} = (d_1, 1, 2, \ldots, d_1 - 1)(d_2, d_1 + 1, d_1 + 2, \ldots, d_2 - 1) \cdots (d_m, d_{m-1} + 1, d_{m-1} + 2, \ldots, d_m - 1)$. If $Q$ denotes the permutation matrix defined by $\pi$, one obtains $QPQ^{-1}$ in the desired form. 

59
We say that any permutation matrix of this structure is in cycle normal form. The cycle normal form of $P$ is denoted by $C(P)$. Up to permutation-similarity, we can always assume that the permutation matrix $P$ of a given matrix with XOR-count $t$ is in cycle normal form, as stated in the following corollary.

**Corollary 3.2.**

\[
P \prod_{k=1}^{t} (I + E^{[i_k, j_k]}) \sim \pi \quad C(P) \prod_{k=1}^{t} (I + E^{[\sigma^{-1}(i_k), \sigma^{-1}(j_k)]})
\]

for some permutation $\sigma \in S_s$.

### 3.3 Efficient Multiplication in Finite Fields

In this section, we first present some theoretic results towards understanding the structure of matrices $M_{\alpha, B}$ representing (left-) multiplication by some finite field element $\alpha \in \mathbb{F}_{2^s}$. The parameter $B$ indicates a basis of $\mathbb{F}_{2^s}$ considered as an $s$-dimensional vector space over $\mathbb{F}_2$. The XOR-count of $M_{\alpha, B}$ is indeed depending on the choice of the basis $B$. As described in Corollary 3.2, we can assume a certain normal form for matrices with an XOR-count of $t$.

Not every (invertible) matrix is a representation of a field multiplication. For example, an obvious condition for that is that the multiplicative order of the matrix divides $2^s - 1$. In order to understand exactly which matrices indeed represent multiplication with some field element $\alpha$, Theorem 3.1 below gives a characterization that allows to efficiently decide when a given matrix corresponds to multiplication by a field element. The crucial part is the minimal polynomial of $\alpha$. It is a property of the linear mapping

\[ f_\alpha : \mathbb{F}_{2^s} \to \mathbb{F}_{2^s}, \beta \mapsto \alpha \beta \]

and is invariant under changing the specific representation of $f_\alpha$ to $\beta \mapsto M_{\alpha, B} \beta$.

**Theorem 3.1.** Let $A \in \text{Mat}_s(\mathbb{F}_2) \setminus \{0_s\}$. Then $A = M_{\alpha, B}$ for some element $\alpha \in \mathbb{F}_{2^s}^*$ with respect to some basis $B$ if and only if $m_A$ is irreducible.

**Proof.** As described in [War94], the ring generated by a matrix $A$ defines a field of order $2^s$ if and only if the characteristic polynomial $\chi_A$ is irreducible. This is the case since $\chi_A(A) = 0$ and thus $A$ is the root of an irreducible polynomial of degree $s$. One can see that $\mathbb{F}_2(A) = \{ \sum_{i=0}^{s-1} \alpha_i A^i \mid \alpha_i \in \mathbb{F}_2 \}$ since it must contain all sums of powers of $A$. However, for $\mathbb{F}_2(A)$ being a field it is not necessary that $A$ has an irreducible characteristic polynomial. It can be possible that $A$ generates a subfield $\mathbb{F}_{2^m}$ of $\mathbb{F}_{2^s}$. As we show now, this is the case if and only if the minimal polynomial of $\alpha$ is irreducible and has degree $m$.

If $m_A$ is not irreducible, $\mathbb{F}_2(A)$ is not a field and thus $A$ cannot represent a field multiplication. Let now $m_A$ be irreducible. The characteristic polynomial
\(\chi_A\) is necessarily a power of \(m_A\), since both of these polynomials share the same irreducible factors. So, \(\chi_A = (m_A)^d\) for some positive integer \(d\). Both \(d\) and \(\deg m_A\) divide \(s\). Because of the irreducibility of \(m_A\), the rational canonical form of \(A\) consists of \(d\) blocks of \(C_{m_A}\). Thus, we obtain the similarity

\[A \sim \bigoplus_{k=1}^{d} C_{m_A}.\]

Since \(\chi_{C_{m_A}} = m_A\), the matrix \(A\) defines a multiplication with some element in a subfield of \(F_{2^s}\).

Note that any field element \(\alpha\) is, up to its conjugates \(\alpha, \alpha^2, \alpha^4, \ldots, \alpha^{2^{s-1}}\), uniquely identified by its minimal polynomial. For every field element \(\alpha\), the minimal polynomial \(m_\alpha\) is exactly the minimal polynomial \(m_A\) of a matrix \(A\) representing multiplication with \(\alpha\). Furthermore, two matrices \(A, A' \in \text{Mat}_s(F_2)\) with the same irreducible minimal polynomial are similar. Thus, given a matrix \(A\), identifying the element \(\alpha\) such that \(A = M_\alpha, B\) is equivalent to computing the (irreducible) minimal polynomial of \(A\).

The main question is which field elements can be implemented with a minimal number of XOR operations, or in particular, what is the minimal XOR-count for a given (non-trivial) field element \(\alpha \in F_{2^s}\). Trivially, multiplication with \(\alpha = 1\) can be implemented with zero additions since \(M_{1,B} = I_s\) for all bases \(B\). On the other hand, if the XOR-count is 0, the element is equal to 1. In a first place, we thus aim for an XOR-count of 1 whenever possible. By a simple observation, this optimal result can be realized if the minimal polynomial of \(\alpha\) is a trinomial of degree \(s\).

Example 3.1. Let the field with \(2^s\) elements be represented as \(F_{2^s} = F_2[X]/(Q)\) for an irreducible polynomial \(Q\) of degree \(s\). For the (left-) multiplication with \(X\) in the canonical basis \(B = \{1, X, X^2, \ldots, X^{s-1}\}\), it is \(M_{X,B} = C_Q\). Thus, \(w(M_{X,B}) = w(Q) - 2\) and the XOR-count of \(M_{X,B}\) equals 1 if \(Q\) is a trinomial.

Since our approach is about finding any (non-trivial) element \(\alpha \in F_{2^s}\), such that multiplication with \(\alpha\) can be implemented with minimal additions, this fact implies that we cannot hope to improve upon the implementation costs if there exists an irreducible trinomial of degree \(s\). However, for several \(s\), including the interesting case where \(s\) is a multiple of 8, there does not exist such a trinomial [Swa62]. The question is what happens for these cases. As one of our main results, we show that the condition on the minimal polynomial is not only sufficient but also necessary.

### 3.3.1 Characterizing Elements with Optimal XOR-Count

In this section, we prove the converse of the fact described in Example 3.1, namely the necessary condition on the minimal (resp. characteristic) polynomial of \(\alpha\) resulting in an XOR-count of 1.

Theorem 3.2. Let \(\alpha \in F_{2^s}\). Then there exists a matrix \(A\) with \(w_{\oplus}(A) = 1\) such that \(A = M_{\alpha, B}\) for some basis \(B\) if and only if \(m_\alpha\) is a trinomial of degree \(s\).
Proof. Let $M_{\alpha,B}$ represent multiplication by some element $\alpha \in \mathbb{F}_2^s$ with respect to the basis $B = \{b_1, \ldots, b_s\}$ and let further $w_{\oplus}(M_{\alpha,B}) = 1$. We show that the characteristic polynomial $\chi_{M_{\alpha,B}}$ is a trinomial and coincides with $m_\alpha$. Since the XOR-count is 1, we can assume, w.l.o.g., that $M_{\alpha,B} = P + E[i,j]$ such that $P = \bigoplus_{k=1}^l C X^m + 1$ is in cycle normal form. We first show that $l = 1$. Suppose $l > 1$, then, depending on $E[i,j]$, the matrix $M_{\alpha,B}$ is either in upper or lower block-triangular form consisting of at least two diagonal blocks. Since at least one of them must be of the form $C X^m + 1$, the polynomial $X^m + 1$ must divide the characteristic polynomial $\chi_{M_{\alpha,B}}$. Since further $(X + 1) \mid (X^m + 1)$, the minimal polynomial of $\alpha$ is necessarily a multiple of $X + 1$. This is a contradiction since $\alpha \neq 1$ and $m_\alpha$ must be irreducible. Hence, $M_{\alpha,B}$ is permutation-similar to $C X^s + 1 + E[i,j]$. It is further $i \neq j + 1 \mod s$ since otherwise $M_{\alpha,B}$ would be singular.

We now investigate how $\alpha$ operates on the basis elements $b_k \in B$. Considering the structure of $M_{\alpha,B}$, we obtain the following list of equations.

\[
\begin{align*}
\alpha b_1 &= b_2 \\
\vdots \\
\alpha b_{j-1} &= b_j \\
\alpha b_j &= b_{j+1} + b_i \\
\alpha b_{j+1} &= b_{j+2} \\
\vdots \\
\alpha b_s &= b_1 .
\end{align*}
\]

By defining $\gamma = b_{j+1}$, one can express every basis element $b_k$ as a power of $\alpha$ multiplied by $\gamma$. In particular,

\[
b_{j+k} \mod s = \alpha^{k-1} \gamma
\]

for $k \in \{1, \ldots, s\}$. Combining this observation with the identity $\alpha b_j = b_{j+1} + b_i$, one obtains

\[
\alpha^s \gamma = \gamma + \alpha^t \gamma
\]

for some exponent $t \neq 0$. Since $\gamma \neq 0$, the field element $\alpha$ is a root of the trinomial $Q = X^s + X^t + 1$. It is left to show that $Q$ is exactly the minimal polynomial of $\alpha$. Suppose that $m_\alpha = X^m + \sum_{k=0}^{m-1} c_k X^k$ with coefficients $c_k \in \{0, 1\}$ and $m < s$. By multiplying $m_\alpha(\alpha)$ with $\gamma$, one obtains

\[
\alpha^m \gamma = \sum_{k=0}^{m-1} c_k \alpha^k \gamma
\]

and thus $b_{t_m} = \sum_{k=0}^{m-1} c_k b_{t_k}$ for some basis elements $b_{t_k}$. We are now able to express one basis element $b_{t_k}$ as a sum of other elements from $B$ which is contradictory to
the linear independence of the basis. Hence, \( \deg m_\alpha = s \) and thus \( m_\alpha = Q \) which finally proves the theorem.

Note that the polynomial \( Q \) is exactly the characteristic polynomial of \( M_{\alpha,B} \) since it must be a monic multiple of \( m_\alpha \) having degree \( s \). An alternative way of proving that the characteristic polynomial of a matrix \( C_{X^s+1} + E^{[i,j]} \) is a trinomial is given in Section 3.3.2 below. As a simple corollary one obtains that any \( \alpha \in \mathbb{F}_2^* \) with an XOR-count of 1 cannot be contained in a proper subfield.

**Corollary 3.3.** Let \( \alpha \in \mathbb{F}_2^* \setminus \{1\} \) and let further \( \deg m_\alpha < s \), indicating that \( \alpha \) lies in a proper subfield of \( \mathbb{F}_2^* \). Then, any matrix \( M_{\alpha,B} \) representing multiplication by a field element \( \alpha \) with respect to some basis \( B \) has \( w_\oplus(M_{\alpha,B}) > 1 \).

This result implies that building MDS layers using a block interleaving construction [ADK+14], also called subfield construction in [KPPY14], almost always results in suboptimal implementation costs. Note that specific instances of this construction are also implicitly used in the AES, LS-Designs [GLSV15] and the hash function Whirlwind [BNN+10].

Now let \( \alpha \) be an element with XOR-count 1. From Corollary 3.1 we know that \( \alpha^{-1} \) has the same XOR-count. Next, we show that there do not exist any further elements with an XOR-count equal to 1.

**Theorem 3.3.** For any given basis \( B \) of \( \mathbb{F}_2^* \), there exist at most two field elements \( \alpha \) and \( \alpha^{-1} \) with \( w_\oplus(M_{\alpha,B}) = w_\oplus(M_{\alpha^{-1},B}) = 1 \).

**Proof.** Let \( \alpha \in \mathbb{F}_2^* \) with \( w_\oplus(M_{\alpha,B}) = 1 \) for the basis \( B = \{b_1, \ldots, b_s\} \). We show that, for any \( \beta \in \mathbb{F}_2^* \) with \( w_\oplus(M_{\beta,B}) = 1 \), it holds that \( \beta = \alpha^k \).

Since, w.l.o.g., \( M_{\alpha,B} \) can be assumed to be of the form \( C_{X^s+1} + E^{[i,j]} \), we know that (3.1) and (3.2) hold. We further know that \( M_{\alpha,B} \) is of the form \( P + E^{[i',j']} \) and thus there exist \( l, m \in \{1, \ldots, s\} \) with \( l \neq m \) and \( \beta b_{j+l} \mod s = b_{j+m} \mod s \). Using equation (3.1), we can write \( \beta = \alpha^{m-l} =: \alpha^y \) where \( y \in \{-(s-1), \ldots, s-1\} \). We directly see that \( y \neq 0 \). It remains to show that \( -1 \leq y \leq 1 \).

Assume \( y \geq 2 \). We use equations (3.1) and (3.2) to obtain
\[
\beta b_{j+(s-y+1)} \mod s = \alpha^s \gamma = \gamma + \alpha^t \gamma = b_{j+1} \mod s + b_{j+t+1} \mod s.
\]
Since \( 0 < t < s \), it holds that \( b_{j+1} \mod s \neq b_{j+t+1} \mod s \) and thus the according column contains an additional 1. For the next column, we have
\[
\beta b_{j+(s-y+2)} \mod s = \alpha^{s+1} \gamma = \alpha \gamma + \alpha^{t+1} \gamma
\]
\[
= \begin{cases} 
  b_{j+2} \mod s + b_{j+t+2} \mod s, & \text{for } t < s-1 \\
  b_{j+2} \mod s + b_{j+1} \mod s + b_j \mod s, & \text{for } t = s-1
\end{cases}
\]
Hence, this column also contains at least one additional 1 which is contradictory to the XOR-count of 1.

For \( -y \geq 2 \) we can construct the same contradiction by considering \( \beta^{-1} \). 

63
We now understand the structure of field elements $\alpha$ that can be implemented with a single addition. One might think that also for the other cases, the weight of the minimal polynomial of $\alpha$ strictly lower-bounds the XOR-count as $w(m_\alpha) - 2$. As we will see next, this is not the case.

### 3.3.2 Experimental Search for Optimal XOR-Counts

Surprisingly, we often can improve the XOR-count, compared to using the companion matrix for multiplication, if the weight of the minimal polynomial is greater than 3. For instance, if $m_\alpha$ is an irreducible pentanomial (that is of weight 5) of degree $s$, there often exists a basis $B$ such that $w(\oplus(M_\alpha,B)) = 2$. Indeed, for all $2 \leq s \leq 2048$ for which no irreducible trinomial of degree $s$ exists, we found some element $\alpha \in \mathbb{F}_2^s$ with an XOR-count of 2 for some basis $B$. For every such dimension, we present an example of such a matrix in Table 3.8. Thus, for all practically relevant fields, we are able to identify an element such that multiplication can be implemented with one or two XOR operations. By Theorem 3.2 these results are proven to be optimal.

Moreover, as fields of small size are most interesting for SP ciphers, we investigated those in full detail. For the fields $\mathbb{F}_{2^4}$, $\mathbb{F}_{2^5}$, $\mathbb{F}_{2^6}$, $\mathbb{F}_{2^7}$ and $\mathbb{F}_{2^8}$ we present the optimal XOR-count for each non-trivial element $\alpha$ in Tables 3.3, 3.4, 3.5, 3.6 and Table 3.7 respectively. The main observation is that each element which is not contained in a proper subfield can be implemented with at most 3 additions. Furthermore, whenever an XOR-count of 2 is possible, the minimal polynomial of $\alpha$ is a pentanomial in all those cases. However, a more thorough characterization of elements with non-optimal XOR-count is left as an open problem (see Section 3.6 for more details).

Those results are based on a search. Since we are only interested in matrices up to similarity (due to the change of basis), we just need to consider all matrices in the normal form described in Corollary 3.2. This will exhaust all possibilities of similarity classes for a given XOR-count $t$. In particular, the search space is reduced from $s!(s(s-1))^t$ to only $p(s)(s(s-1))^t$ where $p(s)$ denotes the number of partitions of $s$, which is exactly the number of possible cycle normal forms of dimension $s$. This allows us to exhaustively search over all similarity classes up to $t = 3$ XOR operations for the fields of small size. The key-point here is that, instead of searching for an optimal basis for a given field element, we generated all matrices with small XOR-count and used Theorem 3.1 in order to check which field element (if any) the given matrix corresponds to.

In order to identify a single lightweight element for larger field sizes, we identified conditions in which cases the characteristic polynomial of a matrix with XOR-count 2 has weight 5, see Theorem 3.4 below. During the search, one only has to check for irreducibility. This allows to compute the results presented in Table 3.8 extremely fast, i.e., within a couple of minutes on a standard PC.
Theorem 3.4. Let \( M = C_{X^s+1} + E^{[i_1,j_1]} + E^{[i_2,j_2]} \) such that the following relations hold:

\[
i_1 < j_1 \neq s, \quad i_2 > j_2 + 1, \quad i_1 \leq j_2, \\
i_2 \leq j_1, \quad j_1 - (i_1 - 1) \neq s, \quad s - (j_1 - i_1) \neq i_2 - j_2.
\]

The characteristic polynomial of \( M \) is a pentanomial of degree \( s \). In particular, \( \chi_M = \lambda^s + \lambda^{s+i_1-i_2-j_2-2} + \lambda^{s+i_1-j_1-1} + \lambda^{i_2-j_2-1} + 1 \).

Proof of Theorem 3.4

In the following, we first present an alternative way of proving the fact that the characteristic polynomial of some matrix \( M = C_{X^s+1} + E^{[i,j]} \) with \( w(M) = 1 \) is a trinomial of degree \( s \). This is true in general, even if \( M \) does not represent a multiplication with a field element. This later helps us to prove Theorem 3.4.

Lemma 3.3. For \( M = C_{X^s+1} + E^{[i,j]} \) with \( w(M) = s + 1 \), the characteristic polynomial \( \chi_M \) of \( M \) is a trinomial of degree \( s \).

Proof. It is to compute \( \chi_M = \det(\lambda I_s - M) = \det(\lambda I_s + C_{X^s+1} + E^{[i,j]}) \). If \( j = s \), then \( M = C_{X^s+1}X^{s-1}+1 \) and \( \chi_M = \lambda^s + \lambda^{s-i} + 1 \) is a trinomial of degree \( s \). Thus, w.l.o.g., one can assume \( j < s \). To compute the determinant, we use Laplace’s formula by expanding along the \( s \)-th column. One obtains

\[
\chi_M = \det \left( \begin{array}{c}
\lambda \\
1 \\
\vdots \\
1
\end{array} \right) + E^{[i-1,j]} \]

\[
+ \lambda \det \left( \begin{array}{c}
\lambda \\
1 \\
\vdots \\
1
\end{array} \right) + E^{[i,j]},
\]

where \( E^{[0,j]} := 0 \) and \( E^{[s,j]} := 0 \). Both of these remaining matrices are of dimension \((s-1) \times (s-1)\). We now distinguish three cases:

1. \( i < j \): The additional 1 lies in the upper triangle of \( M \). Now, \( \chi_M \) reduces to \( \chi_M = 1 + \lambda \det(\lambda I_{s-1} + C_{X^{s-1}} + E^{[i,j]}) \). In order to compute the remaining determinant, we keep on expanding along the last column for \( s - 1 - j \) times until the additional 1 is located in the rightmost column. We now obtain the determinant of a companion matrix. Thus, \( \chi_M = 1 + \lambda^{s-j} \det(\lambda I_j + C_{X^{s-j+i-1}}) \]

\[
= 1 + \lambda^{s-j} (\lambda^j + \lambda^{i-1}) = \lambda^s + \lambda^{s-j+i-1} + 1.
\]
2. $i = j$: In this case, the additional 1 lies on the main diagonal of $M$ and

$$\chi_M = 1 + \lambda(\lambda^{s-2}(\lambda + 1)) = \lambda^s + \lambda^{s-1} + 1.$$ 

3. $i > j$: The additional 1 lies in the lower triangle of $M$. Because of the structure of $M$, it is further $i > (j + 1)$. Defining the $m \times m$ matrix $S_m^\lambda$ as

$$S_m^\lambda := \begin{pmatrix}
1 & \lambda & & \\
& 1 & \lambda & \\
& & \ddots & \ddots \\
& & & 1 & \lambda \\
& & & & 1
\end{pmatrix},$$

the characteristic polynomial of $M$ reduces to $\chi_M = \det(S_m^\lambda + E_{i-1,j}) + \lambda^s$. We expand along the last row of $S_m^\lambda + E_{i-1,j}$ for $s - i$ times and get $\chi_M = \det(S_m^\lambda + E_{i-1,j}) + \lambda^s$.

Now, the additional 1 lies in the last row of the remaining $(i - 1) \times (i - 1)$-dimensional matrix. The goal is now to shift this 1 to the first column. This is done by expanding $j - 1$ times along the first column. We now obtain $\chi_M = \det(S_{i-1}^\lambda + E_{i-1,j}) + \lambda^s$ and the additional 1 is in the lower left corner of the matrix. As a last step, we expand along the first column for one more time and finally get

$$\chi_M = \lambda^s + \det(S_{i-1}^\lambda + E_{i-1,j}) = \lambda^s + \det(\lambda I_{i-1} + C_{X_{i-1}} + E_{i-1,j}) + 1$$

$$= \lambda^s + \lambda^{i-j-1} + 1.$$ 

We now present the proof of Theorem 3.4 which makes use of Lemma 3.3.

**Proof of Theorem 3.4** The first two conditions ensure that $M$ has exactly one additional non-zero entry in the upper and one in the lower triangle (not on the main diagonal). Since $j_1, j_2, i_2 \neq s$, we can expand along the last column and obtain

$$\chi_M = \det(S_{s-1}^\lambda + E_{i_1-1,j_1} + E_{i_2-1,j_2}) + \lambda \det(\lambda I_{s-1} + C_{X_{s-1}} + E_{i_1,j_1} + E_{i_2,j_2}).$$

For simplicity, we define $A := S_{s-1}^\lambda + E_{i_1-1,j_1} + E_{i_2-2,j_2}$ and $B := \lambda I_{s-1} + C_{X_{s-1}} + E_{i_1,j_1} + E_{i_2,j_2}$. In order to compute the latter part, we ”push” the additional non-zero entry from the upper triangle to the top-right corner by first expanding $s - 1 - j_1$ times along the last column and then expanding $i_1 - 1$ times along the first row. The condition $i_2 \leq j_1$ ensures that $E_{i_2,j_2}$ will not be eliminated from expanding along the last column and the condition $i_1 \leq j_2$ ensures that $E_{i_2,j_2}$
will not be eliminated from expanding along the first row. Using Lemma 3.3, one obtains

$$\lambda \det(B) = \lambda^{s-i_1-j_1} \lambda^{i_1-1} \det(\lambda \lambda_{j_1-i_1+1} + C_{X_{j_1-i_1+1}} + E^{[i_2-i_1+1,j_2-i_1+1]})$$

$$= \lambda^{s-j_1+i_1} \left( \lambda^{i_1-1} + \lambda^{i_2-i_1+1-j_2+i_1-1} + 1 \right)$$

$$= \lambda^s + \lambda^{s+i_1-j_1+i_2-j_2-2} + \lambda^{s+i_1-j_1-1}.$$

For $\det(A)$, we proceed similar to case (iii) in Lemma 3.3. We first expand $j_2 - 1$ times along the first column in order to get the additional non-zero value from the lower triangle to the leftmost column. Because of the condition $i_1 \leq j_2$, this eliminates $E^{[i_1-1,j_1]}$. Now, one can expand $s - j_2 - (i_2 - j_2)$ times along the last row, until the remaining additional non-zero entry lies in the lower left corner of the remaining matrix. We finally expand along the first column one more time and obtain

$$\det(A) = \det(S^\lambda_{i_1-j_2} + E^{[i_2-j_2,1]}) = \det(S^\lambda_{i_2-j_2} + E^{[i_2-j_2,1]}) = \lambda^{i_2-j_2-1} + 1.$$

The last two relations make sure that all of the five coefficients of $\det(A) + \lambda \det(B)$ are distinct such that $\chi_M$ is indeed a pentanomial.

### 3.4 Constructing Lightweight MDS Matrices

Our goal is now to construct lightweight MDS matrices. We use the results obtained in the previous sections and restrict our search to circulant matrices and entries with low XOR-count. This simplifies checking the MDS property and computing an upper bound of the XOR-count of the whole matrix. The complexity of our algorithm enables us to easily search for MDS matrices up to dimension 8. Our construction is generic and works for all finite fields $\mathbb{F}_{2^s}$ with $s > b$ for a given bound $b$.

More precisely, we construct circulant matrices with entries of the form $\alpha^{\pm i}$ where $\alpha$ is an element in $\mathbb{F}_{2^s}$. Choosing entries of this form enables us to easily upper-bound the XOR-count of the elements since

$$w_{\oplus}(x^{\pm k}) \leq k w_{\oplus}(x).$$

This can be easily seen by using Corollary 3.1 and the fact that $\alpha^k$ can be implemented by $k$ times implementing $\alpha$. We want to keep the size of the finite field over which the matrix is defined generic. Thus, we choose the matrix entries from a subgroup of the field of fractions of the polynomial ring $\mathbb{F}_2[X]$, denoted Quot($\mathbb{F}_2[X]$). That is, every element is of the form

$$\frac{X^d + a_d X^{d-1} + \cdots + a_1 X + a_0}{X^t + b_t X^{t-1} + \cdots + b_1 X + b_0}.$$

More precisely, and as mentioned above, we restrict our search to elements from $\langle X \rangle$ which is the multiplicative subgroup of Quot($\mathbb{F}_2[X]$) generated by $X$. Our
search works by constructing MDS conditions for an $n \times n$ matrix $M$ with entries in $\langle X \rangle$. This approach later allows us to substitute the indeterminate $X$ by any $\alpha \in \mathbb{F}_{2^s}$ that fulfills all of the conditions given below. In this context, we let $M(\alpha) \in \text{Mat}_n(\mathbb{F}_{2^s})$ denote the matrix obtained by substituting $X$ with $\alpha \in \mathbb{F}_{2^s}$.

We define the \textit{weight} of some circulant matrix with entries in $\langle X \rangle$ as the sum of the absolute values of the exponents in its first row, that is, the number of times $\alpha$ has to be applied \textit{per row}. Then, for a given dimension, we are interested in finding the lightest matrix $M$ which can be made MDS for as many finite fields as possible. Note that the higher priority here was to find a lightweight matrix. Thus, there might exist matrices which can be made MDS for even more fields, but with a probably higher cost.

### MDS conditions

Note that a matrix is MDS if and only if all its square submatrices are invertible [MS77, p. 321, Theorem 8]. Thus, given a matrix $M \in \text{Mat}_n(\text{Quot}(\mathbb{F}_2[X]))$, we compute the determinants of all square submatrices (called \textit{minors}) of $M$ in order to check the MDS property. This way one obtains a list of conditions (polynomials in $\mathbb{F}_2$) for a matrix to be MDS. Since the determinant of a matrix with elements from a field is an element of the field itself, all of these determinants can be represented as the fraction of two polynomials. Thus, $M$ is MDS if and only if the numerator of all minors is non-zero. One can decompose the numerators into their irreducible factors and collect all of them in a set $\mathcal{T}$. This set now defines the MDS conditions. In particular, $M(\alpha)$ is MDS if and only if $\alpha$ is not a root of any of these irreducible polynomials in $\mathcal{T}$, i.e., if and only if $m_\alpha \notin \mathcal{T}$. This trivially holds for $s > \max_{P \in \mathcal{T}} \{\deg P\}$ and any $\alpha \in \mathbb{F}_{2^s}$ which is not contained in a proper subfield. In general, if $\alpha$ is not contained in a proper subfield, the necessary and sufficient condition for the existence of an MDS matrix $M(\alpha)$ is that not all irreducible polynomials of degree $s$ are contained in $\mathcal{T}$. We note that there exists a value $b$ which lower bounds the field size for which $M$ can always be made MDS. That is, for all $t > b$, there exists an irreducible polynomial of degree $t$ which is not in $\mathcal{T}$.

#### 3.4.1 Generic Lightweight MDS Matrices

We now present some results obtained by the approach described above. Given the restrictions, these matrices achieve the smallest weight, i.e., the smallest sum of (absolute) exponents of $X$. Later, we will use these generic matrices to build concrete instantiations of $n \times n$ MDS matrices $M(\alpha)$ for $n \in \{2, 3, \ldots, 8\}$ over a finite field $\mathbb{F}_{2^s}$ with $s > b$. We note that the given results are not necessarily the only possible constructions with the smallest weight.

We also present the conditions for the matrix to be MDS, i.e., the irreducible polynomials that must not be equal to $m_\alpha$. However, since the number of conditions rapidly increases with the dimension of the matrix, we refrain from presenting a complete list for dimensions 6 to 8. Instead, we give the SageMath [StSDT16].
Listing 3.1: Sage code for computing the set $\mathcal{T}$.

```python
P.<X> = GF(2)[]
K = FractionField(P)

def mds_equations(M):
    R = [P(X)]
    for i in range(len(M.rows())+1)[1:]:
        L = M.minors(i)
        for l in L:
            if l != 0:
                F = list(l.numerator().factor())
                for f in F:
                    R.append(f[0])
            else:
                return
    return list(set(R))

```

source code that was used to compute the set $\mathcal{T}$ of irreducible polynomials in Listing 3.1.

2 × 2 and 3 × 3 matrices

The matrices

$$\text{circ}(1, \alpha) = \begin{pmatrix} 1 & \alpha \\ \alpha & 1 \end{pmatrix}$$

and

$$\text{circ}(1,1,\alpha) = \begin{pmatrix} 1 & 1 & \alpha \\ \alpha & 1 & 1 \\ 1 & \alpha & 1 \end{pmatrix}$$

are MDS for all $\alpha \neq 0, 1$.

4 × 4 matrices

For $s > 3$, there exists an $\alpha \in \mathbb{F}_{2^s}$ such that the matrix $\text{circ}(1,1,\alpha,\alpha^{-2})$ is MDS. More precisely, the matrix is MDS if and only if $\alpha$ is not a root of any of the
following polynomials:

\[
\begin{align*}
X \\
X + 1 \\
X^2 + X + 1 \\
X^3 + X + 1 \\
X^3 + X^2 + 1 \\
X^4 + X^3 + X^2 + X + 1 \\
X^5 + X^2 + 1
\end{align*}
\]

5 × 5 matrices

For \( s > 3 \), there exists an \( \alpha \in \mathbb{F}_{2^s} \) such that the matrix \( \text{circ}(1, 1, \alpha, \alpha^{-2}, \alpha) \) is MDS. More precisely, the matrix is MDS if and only if \( \alpha \) is not a root of any of the following polynomials:

\[
\begin{align*}
X \\
X + 1 \\
X^2 + X + 1 \\
X^3 + X + 1 \\
X^3 + X^2 + 1 \\
X^4 + X^3 + X^2 + X + 1 \\
X^5 + X^2 + 1
\end{align*}
\]

6 × 6 matrices

For \( s > 5 \), there exists an \( \alpha \in \mathbb{F}_{2^s} \) such that the matrix \( \text{circ}(1, \alpha, \alpha^{-2}, \alpha^{-1}, 1, \alpha^3) \) is MDS.

7 × 7 matrices

For \( s > 5 \), there exists an \( \alpha \in \mathbb{F}_{2^s} \) such that \( \text{circ}(1, 1, \alpha^{-2}, \alpha, \alpha^2, \alpha, \alpha^{-2}) \) is an MDS matrix.

8 × 8 matrices

For \( s > 7 \), there exists an \( \alpha \in \mathbb{F}_{2^s} \) such that \( \text{circ}(1, 1, \alpha^{-1}, \alpha, \alpha^{-1}, \alpha^3, \alpha^4, \alpha^{-3}) \) is an MDS matrix.
Table 3.1: Optimal instantiations of the generic MDS matrices for $2 \leq n \leq 8$. In each cell, the first entry describes the minimal polynomial of $\alpha \in \mathbb{F}_{2^s}$ and the second entry describes the overhead of the instantiated $n \times n$ matrix $M(\alpha)$. The trinomial $X^s + X^a + 1$ is denoted by $(a)$ and the pentanomial $X^s + X^a + X^b + X^c + 1$ is denoted by $(a,b,c)$, respectively.

<table>
<thead>
<tr>
<th>n</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(1), 1</td>
<td>(1), 1</td>
<td>(1), 1</td>
<td>(2), 1</td>
<td>(1), 1</td>
<td>(1), 1</td>
<td>(6,5,1), 2</td>
<td>(1), 1</td>
<td>(3), 1</td>
<td>(2), 1</td>
<td>(3), 1</td>
<td>(10,9,1), 2</td>
</tr>
<tr>
<td>3</td>
<td>(1), 1</td>
<td>(1), 1</td>
<td>(1), 1</td>
<td>(2), 1</td>
<td>(1), 1</td>
<td>(1), 1</td>
<td>(6,5,1), 2</td>
<td>(1), 1</td>
<td>(3), 1</td>
<td>(2), 1</td>
<td>(3), 1</td>
<td>(10,9,1), 2</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>(1), 1</td>
<td>(3), 3</td>
<td>3</td>
<td>(1), 1</td>
<td>(1), 1</td>
<td>(6,5,1), 6</td>
<td>(1), 1</td>
<td>(3), 3</td>
<td>(2), 3</td>
<td>(3), 3</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>(3,2,1), 8</td>
<td>(2), 4</td>
<td>(1), 1</td>
<td>(1), 1</td>
<td>(6,5,1), 8</td>
<td>(1), 1</td>
<td>(3), 4</td>
<td>(2), 4</td>
<td>(3), 4</td>
<td>(10,9,1), 8</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>(1), 8</td>
<td>(1), 1</td>
<td>(6,5,1), 16</td>
<td>(1), 1</td>
<td>(3), 8</td>
<td>(2), 8</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>(6,5,2), 26</td>
<td>(8), 13</td>
<td>(3), 13</td>
<td>(2), 13</td>
<td>(3), 13</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

3.4.2 Instantiating Lightweight MDS Matrices

We now combine the efficient multiplication in finite fields from Section 3.3 with our construction of MDS matrices, i.e., the presented generic MDS matrices are instantiated with elements $\alpha$ with low XOR-count.

In a matrix multiplication every element is computed as the sum over multiplications. The according XOR-count was already discussed in [KPPY14] and [SKOP15]. For our matrices, the total number of XOR operations needed per row is upper bounded by

$$(n - 1)s + w \cdot w_\oplus(\alpha).$$

Here, $(n - 1)s$ XORs are the static part which comes from summing over the multiplication results and $w$ is the weight as defined above. The overhead of $w \cdot w_\oplus(\alpha)$ XORs is needed for multiplying with the single elements. The static part cannot be changed by fast multiplication. Therefore, this overhead is the part that has to be minimized.

The cost per bit for the whole matrix is given by

$$n((n - 1)s + w \cdot w_\oplus(\alpha)) = n - 1 + \frac{w \cdot w_\oplus(\alpha)}{s}.$$ 

One can notice that it decreases for larger field sizes.

For each of the matrices $M$ described in Section 3.4.1, Table 3.1 presents choices for $\alpha$ such that $M(\alpha)$ is MDS. Note that concrete instantiations are only given up to the field size $s = 13$. The reason is that for larger $s$, all possible $C_P$ with $P$ as an irreducible degree-$s$ polynomial of weight 3 are valid choices. If no such trinomial exists, one can choose $M_{\alpha,B}$ as in Table 3.8.

Table 3.2 compares the results presented in this section to the best constructions known to date. It turned out that our construction of the $4 \times 4$ MDS matrix in $\mathbb{F}_{2^4}$ is identical to the $\mathbb{F}_2$-linear matrix constructed in [LW16, LS16]. We stress that our construction leads to among the lightest MDS matrices, improving the results described in [LS16, SKOP15] for $4 \times 4$ MDS matrices over $\mathbb{F}_{2^8}$ and $8 \times 8$ MDS matrices over $\mathbb{F}_{2^8}$, respectively. This is also the case when considering an
Table 3.2: Comparison of our results with the (non-involutory) $\mathbb{F}_2$-linear MDS matrices from \cite[Section 6.2]{SKOP15} and \cite{LS16, LW16, SS16b, JPST17} by average overhead per row. In cases where a product is given, a subfield construction was used. $\dagger$: In these constructions, the XOR-count is given by counting the number of additional 1’s in the corresponding matrix.

<table>
<thead>
<tr>
<th>(n, s)</th>
<th>our</th>
<th>\cite{SKOP15}$\dagger$</th>
<th>\cite{LS16}$\dagger$</th>
<th>\cite{LW16}$\dagger$</th>
<th>\cite{SS16b}$\dagger$</th>
<th>\cite{JPST17}</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4,4)</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td>(4,8)</td>
<td>6</td>
<td>2$\cdot$5</td>
<td>8</td>
<td>–</td>
<td>6.75</td>
<td>2$\cdot$2.5</td>
</tr>
<tr>
<td>(8,8)</td>
<td>26</td>
<td>40</td>
<td>30</td>
<td>–</td>
<td>–</td>
<td>24</td>
</tr>
</tbody>
</table>

unrolled implementation of the serial implementations in \cite{WWW13}. Unrolled variants of their implementations have an XOR-count that is slightly larger than ours. Moreover, and more importantly, the circuit depth is considerably increased due to the optimization with respect to a serial implementation.

Note that our results in Table 3.2 are measured by the XOR-count from Definition 3.2 while the results from \cite{LW16, LS16, SKOP15, SS16b} use the more simple XOR-count definition, i.e., counting the number of 1’s in the matrix. Additionally to those results, our understanding of how to choose an optimal basis can also be used to improve existing results in the simple XOR-count definition. For example, we can represent the $8 \times 8$ MDS matrix in $\mathbb{F}_2^8$ from \cite{LS16} with 28 additional ones instead of 30 by change of basis.

3.5 Generalizing the MDS Property

Here, following, e.g., \cite{WWW13}, we consider a generalization to $\mathbb{F}_2$-linear MDS codes in order to improve efficiency.

There are some dimensions for which no field element with an XOR-count of 1 exists, for instance $s = 8$. However, especially this dimension is very important since lots of block cipher designs, including the AES, are byte oriented. One would like to have some element $\alpha$ with $w_\oplus(\alpha) = 1$. A way of solving this problem is to not restrict to field elements. Instead, $\alpha$ can be chosen to be some other matrix in the ring $R = \text{Mat}_s(\mathbb{F}_2)$. Given an $n \times n$ matrix $M$ with elements in Quot($\mathbb{F}_2[X]$), the substitution $M(\alpha)$ now consists of elements in a commutative ring with unity, which is the subring of $R$ generated by $\alpha$. In general, given a commutative ring with unity $R$, one can define the determinant $\det_R: \text{Mat}_n(R) \to R$ in a similar way than for matrices over fields. As described in \cite[pp. 212 - 215]{Kna07}, any $A \in \text{Mat}_n(R)$ is invertible if and only if $\det_R(A)$ is a unit in $R$. We now define the MDS property for matrices over a commutative ring.

**Definition 3.3.** Let $R$ be a commutative ring with unity. A matrix $M \in \text{Mat}_n(R)$ is MDS if and only if for every $1 \leq d \leq n$, any $d \times d$ submatrix of $M$ is invertible.

For checking the MDS property in our case, we use a well-known fact about block matrices.
**Theorem 3.5** (Theorem 1 in [Si00]). Let $K$ be a field and let $R$ be a commutative subring of $\text{Mat}_s(K)$ for some integer $s$. For any matrix $M \in \text{Mat}_d(R)$, it is

$$\det(M) = \det(\det_R(M)),$$

where $\det(M)$ is the determinant of $M$ considered as $M \in \text{Mat}_{ds}(K)$.

As an implication, $M(\alpha)$ is MDS if and only if $P(\alpha)$ is invertible for all $P \in \mathcal{T}$, if and only if $\det(P(\alpha)) \neq 0$ for all $P \in \mathcal{T}$.

**2 × 2 and 3 × 3 matrices**

Given $M = \text{circ}(1, X)$ (resp. $M = \text{circ}(1, 1, X)$), one has to make sure that both $X$ and $X + 1$ are invertible for $M$ to be MDS. This is the case if $X$ is substituted by the companion matrix $C_{X^s+X+1}$ for $s \geq 2$. Thus, $M(C_{X^s+X+1})$ is MDS and each non-trivial entry has an XOR-count of 1.

**4 × 4 matrices**

The MDS conditions are more complex than above. So, we only present some improvements for $s \in \{8, 13, 16\}$. The matrix $M = \text{circ}(1, 1, \alpha, \alpha^{-2})$ is MDS for

$$\alpha \in \{C_{X^8+X^2+1}, C_{X^{13}+X+1}, C_{X^{16}+X+1}\}.$$

Note that a similar matrix for $s = 8$ was recently constructed in [LW16].

### 3.6 Conclusion and Open Problems

We presented a study of optimal multiplication bases with respect to the XOR-count. When applied to MDS matrices those lead to very efficient round-based implementations. We expect our results to be applied in other domains as well.

Our investigations leave some possibilities for future research. While we have been able to characterize exactly which field elements can be implemented with one XOR operation only, the general case is still open. For small fields of dimension smaller or equal to eight, we were able to compute the optimal bases with the help of an exhaustive computer search. However, for larger dimensions, this approach turns quickly inefficient and more insight would be needed. As a first step, we conjecture the following statement.

**Conjecture 3.1.** If $w_{\oplus}(M_{\alpha,B}) = 2$, then $m_\alpha$ is of weight smaller or equal to 5.

Note that the converse of the conjectured statement is (unlike the case of trinomials) wrong. As can be seen in Table 3.7 there exist a pentanomial of degree 8 which cannot be implemented with two XOR operations only. Beyond that, our intuition is that the larger the weight of the minimal polynomial, the larger the gap between the most efficient multiplication and the efficiency of multiplying by
Table 3.3: Minimal XOR-counts for all elements in \( \mathbb{F}_{2^4}^* \).

<table>
<thead>
<tr>
<th>minimal polynomial ( m_\alpha )</th>
<th>( \min w_\oplus(\alpha) )</th>
<th>matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X + 1 )</td>
<td>0</td>
<td>( I )</td>
</tr>
<tr>
<td>( X^2 + X + 1 )</td>
<td>2</td>
<td>( C_{m_\alpha} \oplus C_{m_\alpha} )</td>
</tr>
<tr>
<td>( X^4 + X + 1 )</td>
<td>1</td>
<td>( C_{m_\alpha} )</td>
</tr>
<tr>
<td>( X^4 + X^3 + 1 )</td>
<td>1</td>
<td>( C_{m_\alpha} )</td>
</tr>
<tr>
<td>( X^4 + X^3 + X^2 + X + 1 )</td>
<td>2</td>
<td>( C_{X^4+1} + E[2,2] + E[3,4] )</td>
</tr>
</tbody>
</table>

means of the companion matrix. Quantifying and demonstrating such a statement is an interesting and challenging open problem. Another interesting question is to get an improved understanding of how to most efficiently multiply with elements in proper subfields. More specifically, as a generalization of Corollary 3.3, one may ask the following question.

**Question 3.1.** Is the most efficient way to multiply with a subfield element given by multiplying in the subfield \( d \) times, where \( d \) is the extension degree of the field when viewed as an extension of the subfield. More precisely, given an \( \alpha \in \mathbb{F}_{2^m}^* \subset \mathbb{F}_{2^s}^* \) in a proper subfield of dimension \( m = \frac{s}{d} \) and let \( M_{\alpha \in \mathbb{F}_{2^m},B'} \) be the multiplication matrix in \( \mathbb{F}_{2^m} \) with an optimal XOR-count. Is \( M_{\alpha \in \mathbb{F}_{2^m},B} = \bigoplus_{k=1}^{d} M_{\alpha \in \mathbb{F}_{2^m},B'} \) a matrix with the lowest possible XOR-count for multiplication with \( \alpha \in \mathbb{F}_{2^s}^* \)? In particular, is \( w_\oplus(M_{\alpha \in \mathbb{F}_{2^s},B}) = dw_\oplus(M_{\alpha \in \mathbb{F}_{2^m},B'}) \)?

Finally, for MDS matrices it should be noted that we locally achieve the optimal solution. What would be needed to finally settle the search for lightweight matrices is a global optimal solution. That is, for a given dimension, find an MDS matrix that can be implemented with the minimal number of XOR operations. Very recently, Kranz et al. tackled this problem by applying well-known heuristic algorithms for finding the shortest linear straight-line program to the most efficient locally-optimized MDS matrices [KLSW17]. They substantially reduced the number of XOR operations needed for implementing the matrices.

Finally, when optimizing for software, similar questions can be phrased and investigating solutions that are valid for more than one specific platform is a challenging research topic.
Table 3.4: Minimal XOR-counts for all elements in $\mathbb{F}_{2^5}^\ast$.

<table>
<thead>
<tr>
<th>minimal polynomial $m_\alpha$</th>
<th>min $w_\oplus(\alpha)$</th>
<th>matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X + 1$</td>
<td>0</td>
<td>$I$</td>
</tr>
<tr>
<td>$X^5 + X^2 + 1$</td>
<td>1</td>
<td>$C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^5 + X^3 + 1$</td>
<td>1</td>
<td>$C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^5 + X^3 + X^2 + X + 1$</td>
<td>2</td>
<td>$C_{X^5 + 1} + E^{[2,4]} + E^{[4,2]}$</td>
</tr>
<tr>
<td>$X^5 + X^4 + X^2 + X + 1$</td>
<td>2</td>
<td>$C_{X^5 + 1} + E^{[2,2]} + E^{[3,5]}$</td>
</tr>
<tr>
<td>$X^5 + X^4 + X^3 + X + 1$</td>
<td>2</td>
<td>$C_{X^5 + 1} + E^{[2,3]} + E^{[3,1]} + E^{[3,3]}$</td>
</tr>
<tr>
<td>$X^5 + X^4 + X^3 + X^2 + 1$</td>
<td>2</td>
<td>$C_{X^5 + 1} + E^{[2,2]} + E^{[3,4]}$</td>
</tr>
</tbody>
</table>

Table 3.5: Minimal XOR-counts for all elements in $\mathbb{F}_{2^6}^\ast$.

<table>
<thead>
<tr>
<th>minimal polynomial $m_\alpha$</th>
<th>min $w_\oplus(\alpha)$</th>
<th>matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X + 1$</td>
<td>0</td>
<td>$I$</td>
</tr>
<tr>
<td>$X^2 + X + 1$</td>
<td>3</td>
<td>$C_{m_\alpha} + C_{m_\alpha} + C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^3 + X + 1$</td>
<td>2</td>
<td>$C_{m_\alpha} + C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^3 + X^2 + 1$</td>
<td>2</td>
<td>$C_{m_\alpha} + C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^6 + X + 1$</td>
<td>1</td>
<td>$C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^6 + X^3 + 1$</td>
<td>1</td>
<td>$C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^6 + X^4 + X^2 + X + 1$</td>
<td>2</td>
<td>$(C_{X^6 + 1} + C_{X^2 + 1})(I + E^{[1,5]} + E^{[5,4]})$</td>
</tr>
<tr>
<td>$X^6 + X^4 + X^3 + X + 1$</td>
<td>2</td>
<td>$C_{X^6 + 1} + E^{[2,3]} + E^{[4,6]}$</td>
</tr>
<tr>
<td>$X^6 + X^5 + 1$</td>
<td>1</td>
<td>$C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^6 + X^5 + X^2 + X + 1$</td>
<td>2</td>
<td>$C_{X^6 + 1} + E^{[2,2]} + E^{[3,6]}$</td>
</tr>
<tr>
<td>$X^6 + X^5 + X^3 + X^2 + 1$</td>
<td>2</td>
<td>$C_{X^6 + 1} + E^{[2,2]} + E^{[3,5]}$</td>
</tr>
<tr>
<td>$X^6 + X^5 + X^4 + X + 1$</td>
<td>2</td>
<td>$C_{X^6 + 1} + E^{[2,3]} + E^{[3,1]} + E^{[3,3]}$</td>
</tr>
<tr>
<td>$X^6 + X^5 + X^4 + X^2 + 1$</td>
<td>2</td>
<td>$(C_{X^6 + 1} + C_{X^2 + 1})(I + E^{[1,5]} + E^{[6,1]} + E^{[6,5]})$</td>
</tr>
</tbody>
</table>

75
Table 3.6: Minimal XOR-counts for all elements in $\mathbb{F}_{3^7}^*$.

<table>
<thead>
<tr>
<th>minimal polynomial $m_\alpha$</th>
<th>$\min w_{\oplus}(\alpha)$</th>
<th>matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X + 1$</td>
<td>0</td>
<td>$I$</td>
</tr>
<tr>
<td>$X^7 + X + 1$</td>
<td>1</td>
<td>$C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^7 + X^3 + 1$</td>
<td>1</td>
<td>$C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^7 + X^3 + X^2 + X + 1$</td>
<td>2</td>
<td>$C_{X^7 + 1} + E^{[2,6]} + E^{[4,2]}$</td>
</tr>
<tr>
<td>$X^7 + X^4 + 1$</td>
<td>1</td>
<td>$C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^7 + X^4 + X^3 + X^2 + 1$</td>
<td>2</td>
<td>$(C_{X^7 + 1} \oplus C_{X^{3+1}})(I + E^{[1,5]} + E^{[5,3]})$</td>
</tr>
<tr>
<td>$X^7 + X^5 + X^2 + X + 1$</td>
<td>2</td>
<td>$(C_{X^7 + 1} \oplus C_{X^{2+1}})(I + E^{[1,6]} + E^{[6,5]})$</td>
</tr>
<tr>
<td>$X^7 + X^5 + X^3 + X + 1$</td>
<td>2</td>
<td>$C_{X^7 + 1} + E^{[2,3]} + E^{[4,7]}$</td>
</tr>
<tr>
<td>$X^7 + X^5 + X^4 + X^3 + 1$</td>
<td>2</td>
<td>$(C_{X^7 + 1} \oplus C_{X^{3+1}})(I + E^{[1,5]} + E^{[7,2]})$</td>
</tr>
<tr>
<td>$X^7 + X^5 + X^4 + X^3 + X^2 + X + 1$</td>
<td>3</td>
<td>$C_{X^7 + 1} + E^{[2,3]} + E^{[4,6]} + E^{[4,7]}$</td>
</tr>
<tr>
<td>$X^7 + X^6 + 1$</td>
<td>1</td>
<td>$C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^7 + X^6 + X^3 + X + 1$</td>
<td>2</td>
<td>$(C_{X^7 + 1} \oplus C_{X^{1+1}})(I + E^{[1,7]} + E^{[7,4]})$</td>
</tr>
<tr>
<td>$X^7 + X^6 + X^4 + X + 1$</td>
<td>2</td>
<td>$(C_{X^7 + 1} \oplus C_{X^{1+1}})(I + E^{[1,7]} + E^{[7,3]})$</td>
</tr>
<tr>
<td>$X^7 + X^6 + X^4 + X^2 + 1$</td>
<td>2</td>
<td>$C_{X^7 + 1} + E^{[2,4]} + E^{[4,1]} + E^{[4,4]}$</td>
</tr>
<tr>
<td>$X^7 + X^6 + X^5 + X^2 + 1$</td>
<td>2</td>
<td>$(C_{X^7 + 1} \oplus C_{X^{2+1}})(I + E^{[1,6]} + E^{[7,1]} + E^{[7,6]})$</td>
</tr>
<tr>
<td>$X^7 + X^6 + X^5 + X^3 + X^2 + X + 1$</td>
<td>3</td>
<td>$C_{X^7 + 1} + E^{[2,2]} + E^{[2,3]} + E^{[4,7]}$</td>
</tr>
<tr>
<td>$X^7 + X^6 + X^5 + X^4 + 1$</td>
<td>2</td>
<td>$C_{X^7 + 1} + E^{[2,2]} + E^{[3,4]}$</td>
</tr>
<tr>
<td>$X^7 + X^6 + X^5 + X^4 + X^2 + X + 1$</td>
<td>3</td>
<td>$C_{X^7 + 1} + E^{[2,2]} + E^{[3,4]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^7 + X^6 + X^5 + X^4 + X^3 + X^2 + 1$</td>
<td>3</td>
<td>$C_{X^7 + 1} + E^{[2,2]} + E^{[2,3]} + E^{[4,6]}$</td>
</tr>
</tbody>
</table>
Table 3.7: Minimal XOR-counts for all elements in $\mathbb{F}_2^8$.

<table>
<thead>
<tr>
<th>minimal polynomial $m_\alpha$</th>
<th>$\min w_{\oplus}(\alpha)$</th>
<th>matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X + 1$</td>
<td>0</td>
<td>$\bigoplus_{k=1}^4 C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^2 + X + 1$</td>
<td>4</td>
<td>$C_{m_\alpha} \oplus C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^4 + X^3 + 1$</td>
<td>2</td>
<td>$C_{m_\alpha} \oplus C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^4 + X^3 + X^2 + X + 1$</td>
<td>4</td>
<td>$\bigoplus_{k=1}^2 (C_{X^4+1} + E^{[2,6]} + E^{[3,4]})$</td>
</tr>
<tr>
<td>$X^8 + X^4 + X^3 + X + 1$</td>
<td>2</td>
<td>$C_{m_\alpha}$</td>
</tr>
<tr>
<td>$X^8 + X^5 + X^3 + X + 1$</td>
<td>2</td>
<td>$C_{X^8+1} + E^{[2,6]} + E^{[5,2]}$</td>
</tr>
<tr>
<td>$X^8 + X^5 + X^3 + X^2 + 1$</td>
<td>2</td>
<td>$(C_{X^8+1} \oplus C_{X^5+1})(I + E^{[1,6]} + E^{[6,5]})$</td>
</tr>
<tr>
<td>$X^8 + X^5 + X^4 + X^3 + 1$</td>
<td>2</td>
<td>$C_{X^8+1} + E^{[2,6]} + E^{[6,2]}$</td>
</tr>
<tr>
<td>$X^8 + X^4 + X^3 + X^2 + X + 1$</td>
<td>4</td>
<td>$(C_{X^8+1} \oplus C_{X^8+1})(I + E^{[1,6]} + E^{[6,2]})$</td>
</tr>
<tr>
<td>$X^8 + X^6 + X^3 + X^2 + 1$</td>
<td>2</td>
<td>$C_{X^8+1} + E^{[2,6]} + E^{[4,2]}$</td>
</tr>
<tr>
<td>$X^8 + X^6 + X^3 + X^2 + X + 1$</td>
<td>3</td>
<td>$(C_{X^8+1} \oplus C_{X^2+1})(I + E^{[1,7]} + E^{[7,2]})$</td>
</tr>
<tr>
<td>$X^8 + X^6 + X^5 + X^3 + X + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,3]} + E^{[4,6]}$</td>
</tr>
<tr>
<td>$X^8 + X^6 + X^5 + X^2 + X + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,3]} + E^{[4,5]}$</td>
</tr>
<tr>
<td>$X^8 + X^6 + X^5 + X^2 + 1$</td>
<td>2</td>
<td>$C_{X^8+1} + E^{[2,4]} + E^{[3,8]}$</td>
</tr>
<tr>
<td>$X^8 + X^6 + X^5 + X^3 + X + 1$</td>
<td>2</td>
<td>$(C_{X^8+1} \oplus C_{X^8+1})(I + E^{[1,8]} + E^{[8,5]})$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^3 + X^2 + X + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^2 + X + 1$</td>
<td>2</td>
<td>$(C_{X^8+1} \oplus C_{X^8+1})(I + E^{[1,9]} + E^{[9,3]})$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^3 + X + 1$</td>
<td>2</td>
<td>$(C_{X^8+1} \oplus C_{X^8+1})(I + E^{[1,0]} + E^{[8,1]} + E^{[8,6]})$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^4 + 1$</td>
<td>2</td>
<td>$C_{X^8+1} + E^{[2,6]} + E^{[3,5]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^4 + 1$</td>
<td>2</td>
<td>$C_{X^8+1} + E^{[2,6]} + E^{[3,5]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^3 + X^2 + X + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^3 + X + 1$</td>
<td>2</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^2 + X + 1$</td>
<td>2</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^2 + 1$</td>
<td>2</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^3 + X^2 + X + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^3 + X + 1$</td>
<td>3</td>
<td>$(C_{X^8+1} \oplus C_{X^2+1})(I + E^{[1,1]} + E^{[7,3]} + E^{[7,8]})$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^2 + X + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^2 + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^3 + X^2 + X + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^3 + X + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^2 + X + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^2 + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^3 + X^2 + X + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^3 + X + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^2 + X + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
<tr>
<td>$X^8 + X^7 + X^6 + X^2 + 1$</td>
<td>3</td>
<td>$C_{X^8+1} + E^{[2,2]} + E^{[3,7]}$</td>
</tr>
</tbody>
</table>
Table 3.8: For each $s \leq 2048$ for which no irreducible trinomial of degree $s$ exists, this table presents a matrix of the form $C_{s+1} + E^{[i_1,j_1]} + E^{[i_2,j_2]}$ with irreducible characteristic pentanomial. Such a matrix is represented as a 4-tuple $(i_1,j_1,i_2,j_2)$.

In all cases, the characteristic polynomial is equal to $\lambda^s + \lambda^{s+i_1-j_1+i_2-j_2-2} + \lambda^{s+i_1-j_1-1} + \lambda^{i_2-j_2-1} + 1$.

<table>
<thead>
<tr>
<th>$s$</th>
<th>$(i_1,j_1)$</th>
<th>$s$</th>
<th>$(i_1,j_1)$</th>
<th>$s$</th>
<th>$(i_1,j_1)$</th>
<th>$s$</th>
<th>$(i_1,j_1)$</th>
<th>$s$</th>
<th>$(i_1,j_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>(1,1,1,1)</td>
<td>32</td>
<td>(1,1,1,1)</td>
<td>48</td>
<td>(1,1,1,1)</td>
<td>64</td>
<td>(1,1,1,1)</td>
<td>80</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>17</td>
<td>(1,1,1,1)</td>
<td>33</td>
<td>(1,1,1,1)</td>
<td>49</td>
<td>(1,1,1,1)</td>
<td>65</td>
<td>(1,1,1,1)</td>
<td>81</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>18</td>
<td>(1,1,1,1)</td>
<td>34</td>
<td>(1,1,1,1)</td>
<td>50</td>
<td>(1,1,1,1)</td>
<td>66</td>
<td>(1,1,1,1)</td>
<td>82</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>19</td>
<td>(1,1,1,1)</td>
<td>35</td>
<td>(1,1,1,1)</td>
<td>51</td>
<td>(1,1,1,1)</td>
<td>67</td>
<td>(1,1,1,1)</td>
<td>83</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>20</td>
<td>(1,1,1,1)</td>
<td>36</td>
<td>(1,1,1,1)</td>
<td>52</td>
<td>(1,1,1,1)</td>
<td>68</td>
<td>(1,1,1,1)</td>
<td>84</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>21</td>
<td>(1,1,1,1)</td>
<td>37</td>
<td>(1,1,1,1)</td>
<td>53</td>
<td>(1,1,1,1)</td>
<td>69</td>
<td>(1,1,1,1)</td>
<td>85</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>22</td>
<td>(1,1,1,1)</td>
<td>38</td>
<td>(1,1,1,1)</td>
<td>54</td>
<td>(1,1,1,1)</td>
<td>70</td>
<td>(1,1,1,1)</td>
<td>86</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>23</td>
<td>(1,1,1,1)</td>
<td>39</td>
<td>(1,1,1,1)</td>
<td>55</td>
<td>(1,1,1,1)</td>
<td>71</td>
<td>(1,1,1,1)</td>
<td>87</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>24</td>
<td>(1,1,1,1)</td>
<td>40</td>
<td>(1,1,1,1)</td>
<td>56</td>
<td>(1,1,1,1)</td>
<td>72</td>
<td>(1,1,1,1)</td>
<td>88</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>25</td>
<td>(1,1,1,1)</td>
<td>41</td>
<td>(1,1,1,1)</td>
<td>57</td>
<td>(1,1,1,1)</td>
<td>73</td>
<td>(1,1,1,1)</td>
<td>89</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>26</td>
<td>(1,1,1,1)</td>
<td>42</td>
<td>(1,1,1,1)</td>
<td>58</td>
<td>(1,1,1,1)</td>
<td>74</td>
<td>(1,1,1,1)</td>
<td>90</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>27</td>
<td>(1,1,1,1)</td>
<td>43</td>
<td>(1,1,1,1)</td>
<td>59</td>
<td>(1,1,1,1)</td>
<td>75</td>
<td>(1,1,1,1)</td>
<td>91</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>28</td>
<td>(1,1,1,1)</td>
<td>44</td>
<td>(1,1,1,1)</td>
<td>60</td>
<td>(1,1,1,1)</td>
<td>76</td>
<td>(1,1,1,1)</td>
<td>92</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>29</td>
<td>(1,1,1,1)</td>
<td>45</td>
<td>(1,1,1,1)</td>
<td>61</td>
<td>(1,1,1,1)</td>
<td>77</td>
<td>(1,1,1,1)</td>
<td>93</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>30</td>
<td>(1,1,1,1)</td>
<td>46</td>
<td>(1,1,1,1)</td>
<td>62</td>
<td>(1,1,1,1)</td>
<td>78</td>
<td>(1,1,1,1)</td>
<td>94</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>31</td>
<td>(1,1,1,1)</td>
<td>47</td>
<td>(1,1,1,1)</td>
<td>63</td>
<td>(1,1,1,1)</td>
<td>79</td>
<td>(1,1,1,1)</td>
<td>95</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>32</td>
<td>(1,1,1,1)</td>
<td>48</td>
<td>(1,1,1,1)</td>
<td>64</td>
<td>(1,1,1,1)</td>
<td>80</td>
<td>(1,1,1,1)</td>
<td>96</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>33</td>
<td>(1,1,1,1)</td>
<td>49</td>
<td>(1,1,1,1)</td>
<td>65</td>
<td>(1,1,1,1)</td>
<td>81</td>
<td>(1,1,1,1)</td>
<td>97</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>34</td>
<td>(1,1,1,1)</td>
<td>50</td>
<td>(1,1,1,1)</td>
<td>66</td>
<td>(1,1,1,1)</td>
<td>82</td>
<td>(1,1,1,1)</td>
<td>98</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>35</td>
<td>(1,1,1,1)</td>
<td>51</td>
<td>(1,1,1,1)</td>
<td>67</td>
<td>(1,1,1,1)</td>
<td>83</td>
<td>(1,1,1,1)</td>
<td>99</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>36</td>
<td>(1,1,1,1)</td>
<td>52</td>
<td>(1,1,1,1)</td>
<td>68</td>
<td>(1,1,1,1)</td>
<td>84</td>
<td>(1,1,1,1)</td>
<td>100</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>37</td>
<td>(1,1,1,1)</td>
<td>53</td>
<td>(1,1,1,1)</td>
<td>69</td>
<td>(1,1,1,1)</td>
<td>85</td>
<td>(1,1,1,1)</td>
<td>101</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>38</td>
<td>(1,1,1,1)</td>
<td>54</td>
<td>(1,1,1,1)</td>
<td>70</td>
<td>(1,1,1,1)</td>
<td>86</td>
<td>(1,1,1,1)</td>
<td>102</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>39</td>
<td>(1,1,1,1)</td>
<td>55</td>
<td>(1,1,1,1)</td>
<td>71</td>
<td>(1,1,1,1)</td>
<td>87</td>
<td>(1,1,1,1)</td>
<td>103</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>40</td>
<td>(1,1,1,1)</td>
<td>56</td>
<td>(1,1,1,1)</td>
<td>72</td>
<td>(1,1,1,1)</td>
<td>88</td>
<td>(1,1,1,1)</td>
<td>104</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>41</td>
<td>(1,1,1,1)</td>
<td>57</td>
<td>(1,1,1,1)</td>
<td>73</td>
<td>(1,1,1,1)</td>
<td>89</td>
<td>(1,1,1,1)</td>
<td>105</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>42</td>
<td>(1,1,1,1)</td>
<td>58</td>
<td>(1,1,1,1)</td>
<td>74</td>
<td>(1,1,1,1)</td>
<td>90</td>
<td>(1,1,1,1)</td>
<td>106</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>43</td>
<td>(1,1,1,1)</td>
<td>59</td>
<td>(1,1,1,1)</td>
<td>75</td>
<td>(1,1,1,1)</td>
<td>91</td>
<td>(1,1,1,1)</td>
<td>107</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>44</td>
<td>(1,1,1,1)</td>
<td>60</td>
<td>(1,1,1,1)</td>
<td>76</td>
<td>(1,1,1,1)</td>
<td>92</td>
<td>(1,1,1,1)</td>
<td>108</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>45</td>
<td>(1,1,1,1)</td>
<td>61</td>
<td>(1,1,1,1)</td>
<td>77</td>
<td>(1,1,1,1)</td>
<td>93</td>
<td>(1,1,1,1)</td>
<td>109</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>46</td>
<td>(1,1,1,1)</td>
<td>62</td>
<td>(1,1,1,1)</td>
<td>78</td>
<td>(1,1,1,1)</td>
<td>94</td>
<td>(1,1,1,1)</td>
<td>110</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>47</td>
<td>(1,1,1,1)</td>
<td>63</td>
<td>(1,1,1,1)</td>
<td>79</td>
<td>(1,1,1,1)</td>
<td>95</td>
<td>(1,1,1,1)</td>
<td>111</td>
<td>(1,1,1,1)</td>
</tr>
</tbody>
</table>

78
Chapter 4

On the Best Word Permutations for Lightweight AES-like Ciphers

This chapter is based on joint work with Gianira Alfarano, Stefan Kölbl and Gregor Leander. The author partly contributed to all of the results. His main contribution was the formalization of the method for classifying word permutations with regard to our defined notion of equivalence (i.e., Section 4.2) and the implementation of the classification algorithm for the particular case of the MixColumn matrix of Midori.

4.1 Introduction

The AES has influenced a large variety of lightweight designs. In particular, many designs start with the initial structure of the AES and tailor it with respect to one or more parts in order to fulfill their considered lightweight requirements. One of the more recent examples was presented in 2015 with the block cipher Midori [BBI+15]. While its design of follows the general outline of the AES, almost all components are modified in order to reach the goal of minimizing energy.

In particular, the authors decided to change the MixColumns operation in a way that it applies multiplication with a binary matrix of branch number 4, compared to the non-binary MixColumns operation in the AES with branch number 5. This has the benefit of reducing the energy consumption. Moreover, the implementation of the finite field multiplication can be avoided. However, the downside is that, a priori, the minimum number of active S-boxes reduces. More precisely, while for the AES we have at least 25 active S-boxes in any linear or differential 4-round trail, moving to a branch number of 4 reduces this number to 16 (see Theorem 2.4). For states of quadratic dimension and under the usage of a Mix-
Columns operation with optimal branch number, it was shown in [BJL+15] that one cannot increase the minimum number of active S-boxes when substituting the ShiftRows operation by another, arbitrary, word permutation. Therefore, the interesting, and maybe unexpected, observation made by the designers of Midori is that actually substituting ShiftRows can significantly increase the number of active S-boxes if a MixColumns operation with non-optimal branch number is employed. Indeed, by using the Midori word permutation as given in Section 2.5.3, the authors of Midori managed to increase the minimum number of active S-boxes, e.g., from 20 to 30 for 6 rounds. Later in 2016, Todo and Aoki analyzed which binary matrices lead to an improved number of active S-boxes for the classical ShiftRows permutation [TA16].

The interesting and important question raised by the designers of Midori is what the optimal choice of the word permutation actually is. The difficulty in answering this question comes from the fact that the theoretical approach described in Section 2.4.2 is not capable of proving better bounds on the minimum number of active S-boxes than one obtains by just iterating four-round trails. In other words, with our current knowledge we are not able to theoretically analyze those improved bounds on the number of active S-boxes, but rather have to rely on computer search techniques like Matsui’s algorithm [Mat95] or techniques based on Mixed-Integer Linear Programming (MILP) [MWGP12]. Quite some progress has been made on those tools in recent years, especially in the area of MILP (e.g. [SHW+14b]). For a given word permutation, even for a higher number of rounds, it is still possible to compute bounds on the minimum number of active S-boxes within reasonable time using a standard PC. However, there is a huge choice of possible permutations to be considered – roughly \(2^{44.25} \approx 16!\) in the case of a \(4 \times 4\) state as used in Midori – which immediately renders the naive approach of simply testing them all very inefficient.

The designers of Midori overcome this problem by heuristically reducing the search space of all word permutations to be considered. However, it stayed unclear if, by this reduction, we actually exclude the best possible permutations. In the design document of Midori, some conditions are given under which a permutation should lead to an optimal number of active S-boxes [BBL+15 pp. 15-16]. Unfortunately, those conditions are given without a proof or an intuition. Moreover, as we will see later, those conditions do not guarantee an optimal number of active S-boxes for all number of rounds simultaneously.

Thus, the final goal is clearly to gain theoretical insight on what exactly characterizes the optimal word permutations. However, as already mentioned, this seems out of reach with our current knowledge. We then focus on the task of computationally finding the best permutations among all permutations, i.e., without any restriction on the search space.
Results of this Chapter

The starting point is the simple, but useful, observation given in Section 4.2, i.e., for any AES-like cipher, there are several word permutations which basically lead to the same cipher. More precisely, if two word permutations $p$ and $p'$ differ only by conjugation with a permutation that commutes with the MixColumns operation, the entire two ciphers differ only by a permutation on the plaintext and ciphertext and a corresponding permutation of the round keys. This immediately implies that in particular, the ciphers have the same cryptographic resistance against any attack that does not involve details about the key-scheduling algorithm. Especially, the bounds on the number of active S-boxes are necessarily the same.

For our task of finding the best permutation, this means that we have to check only one of those word permutations, $p$ or $p'$. More formally, we show that being equal up to conjugation with a permutation that commutes with a given MixColumns operation actually defines an equivalence relation (see Definition 4.1) on the set of all possible permutations. We then have to check only one representative of each possible equivalence class.

This naturally leads to the task of classifying permutations with respect to this notion of equivalence. Again, when approaching this task in a naive manner, it quickly turns very inefficient. In order to keep it still manageable, we first study a weakened notion of equivalence, which allows us to significantly simplify the classification algorithm. We furthermore give an easy to verify condition on when this a priori weakened equivalence notion coincides with the equivalence notion mentioned above. The running time of the resulting classification algorithm as well as the number of existing equivalence classes strongly depends on the structure of the MixColumns operation.

The MixColumns operation used in Midori finally serves as a case study for our general approach (see Section 4.3). Focusing on Midori is especially interesting for the following two reasons. Firstly, the MixColumns operation fulfills the sufficient condition for which the weaker notion of equivalence coincides with the stronger notion of equivalence we are actually interested in. This allows the simplified classification mentioned above. Secondly, as we will explain in detail, the number of different equivalence classes is especially small for Midori. Indeed, our algorithm reveals that there are only about $2^{21.7}$ equivalence classes. Thus, compared to checking $2^{44.5}$ possible permutations, we gain a speed-up factor of more than $2^{22}$. All in all, this allows us to compare the actual best word permutation with respect to the number of active S-boxes – without any restriction on the search space – with the one actually used in Midori.

As it turns out, the permutation used in Midori gives optimal bounds for 1 up to 8 and 10 up to 12 rounds. For all other number of rounds up to 40, that word permutation is not optimal. As an alternative, we are able to present a word permutation that is optimal for most number of rounds up to 40, with few exceptions. It is worth noticing that the number of optimal permutations varies when considering different number of rounds and that there is actually no single permutation that achieves the optimal bound for all number of rounds simultaneously.
Our analysis indicates that the conditions on optimal word permutations given by the Midori designers (see [BBI+15, pp. 15-16]) do not precisely characterize the properties a designer would like to have on optimal word permutations.

It is worth remarking that the ciphers mCrypton and Mantis apply the same MixColumns operation than Midori. This suggests that our findings are not limited to Midori, but may instead be useful for future cipher designs.

4.2 Classifying Word Permutations

When designing a new block cipher, besides choosing a cryptographically strong S-box, a crucial goal of the designer is to choose an appropriate linear layer in order to maximize diffusion properties and prevent against differential and linear attacks. In the notion of AES-like ciphers as depicted in Figure 2.6 and formally defined in Definition 2.9, the linear layer is fully specified by a matrix \( M \in \text{GL}_{n_r}(\mathbb{F}_2) \) corresponding to the MixColumns operation and by a permutation \( p \in S_{n_r n_c} \). A natural designer’s approach is to first select the matrix \( M \) and then choose the word permutation that maximizes the minimum number of active S-boxes. Indeed, one of the major novelties of the Midori design was that the choice of a specific type of word permutation, in combination with the appropriate MixColumns matrix, can lead to much higher number of guaranteed active S-boxes, compared to just applying a simple ShiftRows-like operation as it was done in the AES. This analysis can be done by using automatic search tools. For Midori in particular, Matsui’s algorithm was used (see Algorithm 2.2 in Section 2.4.3).

In order to reduce the search space of all permutations in \( S_{n_r n_c} \), it is crucial to identify under which conditions two permutations lead to the same cryptographic properties. In particular, we base our work on the following simple observation: If we consider a permutation \( p \in S_{n_r n_c} \), then any permutation that is obtained from \( p \) by conjugation with some \( \vartheta \in S_{n_r n_c} \) for which \( \text{Mix}_M \circ \text{Permute}_{\vartheta} = \text{Permute}_{\vartheta} \circ \text{Mix}_M \) lead to the same cryptographic properties. In other words, the SP cipher instantiated with the AES-like round \( R_{\text{Sb}, \vartheta \circ \text{Permute}_{\vartheta}^{-1}, M} \) is just a permuted version of the SP cipher instantiated with \( R_{\text{Sb}, p, M} \) (under a permutation of the round keys), whenever the word permutation under \( \vartheta \) commutes with the operation \( \text{Mix}_M \). This fact is illustrated in Figure 4.1. Overall, this motivates the notion of \( M \)-equivalence as defined in the following. For given state dimensions \( n_r n_c \), we will denote the set of all word permutations \( \vartheta \) for which \( \text{Permute}_{\vartheta} \circ \text{Mix}_M \) commute with \( \text{Mix}_M \) by \( T(M) \).

**Definition 4.1** (\( M \)-equivalence). Let \( M \in \text{GL}_{n_r}(\mathbb{F}_2) \). We say that two permutations \( p, p' \in S_{n_r n_c} \) are \( M \)-equivalent, if there exists a permutation \( \vartheta \in T(M) \) such that \( p' = \vartheta \circ p \circ \vartheta^{-1} \). We write \( p \sim p' \) for two \( M \)-equivalent permutations \( p \) and \( p' \).

Note that \( T(M) \) is a subgroup of \( S_{n_r n_c} \) which implies that the relation \( \sim \) is symmetric, reflexive and transitive. Thus, \( \sim \) is indeed an equivalence relation on \( S_{n_r n_c} \).
If $p$ and $p'$ are $M$-equivalent permutations, by definition there exists a permutation $\vartheta$ such that

$$ R_{\text{Sb},p',M} = \text{Permute}_\vartheta \circ R_{\text{Sb},p,M} \circ \text{Permute}_{\vartheta^{-1}}. $$

It is important to note that this can be extended for an arbitrary number of rounds. In particular, for any $t \in \mathbb{N}$, we have

$$ R^t_{\text{Sb},p',M} = \text{Permute}_\vartheta \circ R^t_{\text{Sb},p,M} \circ \text{Permute}_{\vartheta^{-1}}. $$

Thus, if any cryptanalysis is done independently of the actual specification of the round keys (i.e., under the assumption of independent round keys, see Assumption 2.2), the corresponding ciphers share the same cryptographic properties. This holds in particular for the case of differential and linear cryptanalysis; $M$-equivalent permutations lead to the same bound on the minimum number of active S-boxes.

For given $n_r, n_c \in \mathbb{N}$ and $M \in \text{GL}_{n_r}(\mathbb{F}_2)$, we aim for classifying all permutations in $\mathcal{S}_{n_r,n_c}$ up to $M$-equivalence. As described above, such a classification would allow us to check only a single representative of each equivalence class for its cryptographic properties and it thus may significantly reduce the complexity of finding the best word permutation. However, there is a difficulty in achieving this classification. Namely, for an arbitrary $M$, it is not obvious how to efficiently determine $T(M)$ and to separate all permutations into their equivalence classes. In order to reach our goal, we therefore first consider a weaker notion of $M$-equivalence and describe an algorithm that enumerates all permutations up to this weak equivalence. Later, we will see that, for certain choices of $M$, this weak
equivalence coincides with the general notion of $M$-equivalence. Fortunately, this approach allows us to finally classify all word permutations up to $M$-equivalence in its stronger notion for the case of Midori.

**Definition 4.2** (weak $M$-equivalence). Let $\mathcal{P}_\approx$ denote the set of all word permutations that permute whole columns of the state and let $\mathcal{P}_\equiv$ be the set of word permutations that operate independently on the columns of the state. Formally,

$$\mathcal{P}_\approx = \{ p \in S_{n_r,n_c}: \exists \sigma \in S_{n_c}: \forall i \in \mathbb{N}_{n_r}, j \in \mathbb{N}_{n_r}: n_r(j-1)+i \mapsto n_r(\sigma(j)-1)+i \}$$

$$\mathcal{P}_\equiv = \{ p \in S_{n_r,n_c}: \exists \sigma_0, \ldots, \sigma_{n_c-1} \in S_{n_c}: \forall i \in \mathbb{N}_{n_r}, j \in \mathbb{N}_{n_r}: n_r(j)+i \mapsto n_r(j)+\sigma_j(i) \}$$

Then, we say that a word permutation $p$ is weakly $M$-equivalent to a word permutation $p'$, written $p \sim_w p'$, if there exists a word permutation $\vartheta \in \mathcal{T}(M)$ of the form $\vartheta = \pi \circ \phi$, with $\pi \in \mathcal{P}_\approx$ and $\phi \in \mathcal{P}_\equiv$, such that $p' = \vartheta \circ p \circ \vartheta^{-1}$.

Again, since $\{ \vartheta \in \mathcal{T}(M) : \vartheta = \pi \circ \phi \}$ is a subgroup of $S_{n_r,n_c}$, the relation $\sim_w$ is indeed an equivalence relation. Further, $p \sim_w p'$ trivially implies $p \sim p'$. For an equivalence class $[p]_{\sim_w}$, we consider the smallest permutation (in lexicographic ordering $<$) as its canonical representative. For a given $M$, we now describe a way to enumerate a single representative of each equivalence class.

### 4.2.1 Structure Matrix of a Word Permutation

Let $p \in S_{n_r,n_c}$ be a word permutation on an $n_r \times n_c$ state. We define the structure matrix of $p$ as the $n_c \times n_c$ matrix $A_p$ s.t. $A_{p,i,j}$ contains the number of words of column $i$ that are permuted to column $j$. Formally, for $i,j \in \mathbb{N}_{n_c}$,

$$A_{p,i,j} = \{ \{ k | k = n_r(i-1)+r \text{ for } r \in \mathbb{N}_{n_r}, \text{ and } n_r(i-1)+r \mapsto n_r(j-1)+r' \} , r', r' \in \mathbb{N}_{n_r} \}$$

**Example 4.1.**

$$\begin{bmatrix}
1 & 5 & 9 & 13 \\
2 & 6 & 10 & 14 \\
3 & 7 & 11 & 15 \\
4 & 8 & 12 & 16
\end{bmatrix} \xlongequal{p} \begin{bmatrix}
5 & 1 & 14 & 2 \\
6 & 7 & 15 & 3 \\
12 & 10 & 9 & 4 \\
16 & 13 & 8 & 11
\end{bmatrix}, \quad A_p = \begin{bmatrix}
0 & 1 & 0 & 3 \\
2 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 \\
1 & 1 & 2 & 0
\end{bmatrix}$$

Note that an $n_c \times n_c$ matrix is a valid structure matrix for some permutation if and only if the sum of each column as well as the sum of each row adds up to $n_r$. Let now $\sigma \in S_{n_c}$. For an $n_c \times n_c$ matrix $A$, we define $A^\sigma$ as the $n_c \times n_c$ matrix that is obtained from $A$ by both permuting the rows and the columns according to $\sigma$. Formally,

$$\forall i,j \in \mathbb{N}_{n_c} : A^\sigma_{i,j} : A_{\sigma(i),\sigma(j)}$$

We now define an equivalence relation $\sim$ on $n_c \times n_c$ structure matrices as

$$A \sim B \iff \exists \sigma \in S_{n_c} : B = A^\sigma.$$

The following proposition explains how the weak $M$-equivalence of permutations and equivalence of their corresponding structure matrices are related.

84
Proposition 4.1. Let \( n_r, n_c \in \mathbb{N} \) and let \( M \in \text{GL}_{n_c}(\mathbb{F}_2) \).

(i) If \( p \sim_w p' \) for two \( p, p' \in S_{n_r, n_c} \), then \( A_p \sim A_{p'} \).

(ii) Let \( A \sim B \) for two valid \( n_c \times n_c \) structure matrices of permutations in \( S_{n_r, n_c} \). Then, for any permutation \( p \in S_{n_r, n_c} \) that has \( A \) as structure matrix, there exist a permutation \( p' \in S_{n_r, n_c} \) that has \( B \) as structure matrix such that \( p \sim_w p' \).

Proof. (i) Let \( p \sim_w p' \). Then, by definition there exists permutations \( \pi \in \mathcal{P}_n \) and \( \phi \in \mathcal{P}_{r} \) such that \( p' = \pi \circ \phi \circ p \circ \phi^{-1} \circ \pi^{-1} \). Clearly, \( A_p = A_{\phi \circ p \circ \phi^{-1}} \). Let now be \( \sigma \in S_{n_c} \) such that, for all \( i \in \mathbb{N}_{\leq n_r} \) and \( j \in \mathbb{N}_{\leq n_c} \), \( n_r(j - 1) + i \mapsto n_r(\sigma(j) - 1) + i \). Then, for all \( i, j \in \mathbb{N}_{\leq n_c} \), it is

\[
(A_{p'})_{i,j} = (A_{p' \circ \pi})_{\sigma^{-1}(i),\sigma^{-1}(j)} = (A_{\pi^{-1} \circ p' \circ \pi})_{\sigma^{-1}(i),\sigma^{-1}(j)} = (A_{\phi \circ p' \circ \phi^{-1}})_{\sigma^{-1}(i),\sigma^{-1}(j)} = (A_p)_{\sigma^{-1}(i),\sigma^{-1}(j)}.
\]

This shows that \( A_{p'} = A_p^{\sigma^{-1}} = A_p^{-1} \).

(ii) Let \( p \in S_{n_r, n_c} \) such that \( A_p = A \). By definition of the equivalence between \( A \) and \( B \), there exist a \( \sigma \in S_{n_c} \) such that \( A_p^\sigma = B \). With the same observation as above, it follows that there exists a \( \pi \in \mathcal{P}_n \) such that \( A_p^\pi = A_{\pi \circ r \circ \pi^{-1}} \).

This means that \( \pi \circ p \circ \pi^{-1} \) has structure matrix \( B \). Now, since \( \text{Permute}_{\pi} \) commutes with \( \text{Mix}_M \), we have \( \pi \circ p \circ \pi^{-1} \sim_w p \).

\(\square\)

This result (ii) implies that, in order to characterize permutations up to weak \( M \)-equivalence, it is enough to separate all valid structure matrices into their equivalence classes, pick a representative of each class and search for all permutations (up to weak \( M \)-equivalence) that fulfill the respective structure matrix.

4.2.2 Search Algorithm

Algorithm 4.1 for enumerating all permutations up to weak \( M \)-equivalence for a given structure matrix \( A \) is given below. It works as follows. We start with an \( n_r \times n_c \) word permutation \( p_{\text{start}} \) which is undefined at any position (this is represented by a \(-1\) value). Then, we apply \texttt{EnumerateRecursive} to \( p_{\text{start}} \). After each call, the permutation is extended by another column until it is completely defined. Note that it can only be extended by a column which meets the requirements given by the structure matrix \( A \) (see line 12 of Algorithm 4.1). Further, the extension has to make sure that we can still get a permutation, i.e., no value except \(-1\) occurs twice. Only if the new extended permutation leads to a smallest representative of the equivalence class of conjugation with all \( \pi \in \mathcal{P}_1 \), the algorithm will proceed with this permutation. This is checked in line 14. Checking if \( p \) is the smallest representative can be done with at most \(|\mathcal{P}_1|\) iterations.

After the algorithm terminates, it outputs a list of word permutations that contains at least one representative of each equivalence class. However, it can
contain more than one representative. As a last step, for each \( p \) in this list, it is therefore required to check if \( p \) is the smallest permutation w.r.t. conjugation by all \( \pi \circ \phi \) with \( \phi \in \mathcal{P}_\dagger \) and \( \pi \in \mathcal{P}_\equiv \), that leave the structure matrix of \( p \) invariant. In other words, it is to check if \( p \) is smaller than any permutation \( \pi \circ \phi \circ \phi^{-1} \circ \pi^{-1} \), where \( \phi \in \mathcal{P}_\dagger \) and \( \pi \in \mathcal{P}_\equiv \) such that \( A_p = A_p^\sigma \) for the \( \sigma \in S_{n_r} \) corresponding to the permutation \( \pi \).

### 4.2.3 The Difference between the Equivalence Notions

In this section, we outline the relation between weak \( M \)-equivalence and the stronger notion of \( M \)-equivalence. The following proposition describes a sufficient condition on the matrix \( M \) such that the notions of weak \( M \)-equivalence and \( M \)-equivalence are the same. Note that, from now on, we focus on matrices \( M \in \text{GL}_{n_r}(\mathbb{F}_2) \) that consists of binary coefficients only, i.e., coefficients from \( \{0, 1\} \subseteq \mathbb{F}_2^2 \).

**Proposition 4.2.** Let \( M \in \text{GL}_{n_r}(\mathbb{F}_2) \) be an \( n_r \times n_r \) matrix with binary coefficients and let \( G \) be the directed graph with \( n_r \) vertices that has \( M \) as its adjacency matrix. Then, if \( G \) is a strongly connected directed graph, the notion of \( M \)-equivalence coincides with the notion of weak \( M \)-equivalence.

**Proof.** Let \( \vartheta \) be a permutation in \( T(M) \). We can write any word position \( k \in \{1, \ldots, n_r n_c\} \) of an \( n_r \times n_c \) state uniquely as \( k = n_r \cdot \text{Block}(k) + \text{Index}(k) \) with \( 1 \leq \text{Index}(k) \leq n_r \). We now have to show that, for all \( k, k' \in \{1, \ldots, n_r n_c\} \), \( \text{Block}(k) = \text{Block}(k') \) implies \( \text{Block}(\vartheta(k)) = \text{Block}(\vartheta(k')) \). This would imply that \( \vartheta \) can be written as \( \pi \circ \phi \) with \( \pi \in \mathcal{P}_\equiv \) and \( \phi \in \mathcal{P}_\dagger \).

We can represent the operation \( \text{Mix}_M \) as the \( n_r n_c \times n_r n_c \) binary block-diagonal matrix which consists of \( n_c \) blocks of the \( n_r \times n_r \) matrix \( M \), i.e.,

\[
\text{Mix}_M = \left( \begin{array}{cccc} M & & & \\ & M & & \\ & & \ddots & \\ & & & M \end{array} \right) =: (b_{i,j})_{i,j \in \mathbb{N}_{\leq n_r n_c}}.
\]

Since the permutation matrix of \( \vartheta \) has to commute with \( \text{Mix}_M \) considered as an \( n_r n_c \times n_r n_c \) binary matrix, we necessarily have the property that \( b_{i,j} = b_{\vartheta(i), \vartheta(j)} \) for all \( i, j \in \{1, \ldots, n_r n_c\} \). Let now \( k, k' \in \{1, \ldots, n_r n_c\} \). By the block-diagonal structure of the matrix \( \text{Mix}_M \), it is

\[
b_{k,k'} = 1 \iff \text{Block}(k) = \text{Block}(k') \text{ and } \text{Index}(k') \in T_{\text{Index}(k)},
\]

where \( T_i := \{ j \in \mathbb{N}_{\leq n_r} \mid M_{i,j} = 1 \} \). Since \( b_{k,k'} = b_{\vartheta(k), \vartheta(k')} \), we have that \( \text{Block}(\vartheta(k)) = \text{Block}(\vartheta(k')) \), for all \( k' \) with \( \text{Index}(k') \in T_{\text{Index}(k)} \) and \( \text{Block}(k) = \text{Block}(k') \).

What we have now shown is that, for an arbitrary \( l \in \mathbb{N}_{\leq n_r} \) and for all \( k, k' \) with \( \text{Index}(k), \text{Index}(k') \in T_l \),

\[
\text{Block}(k) = \text{Block}(k') \implies \text{Block}(\vartheta(k)) = \text{Block}(\vartheta(k')).
\]
Algorithm 4.1 Enumerate all permutations up to weak $M$-equivalence for a given structure matrix $A$

1: procedure $\text{ENUMERATEPERMUTATIONS}(A)$
2: $R \leftarrow \{\}$
3: $p_{\text{start}} = \begin{bmatrix} -1 & \ldots & -1 \\ \vdots & \ddots & \vdots \\ -1 & \ldots & -1 \end{bmatrix}$
4: $\text{ENUMERATERECURSIVE}(A, 0, p_{\text{start}})$
5: return $R$
6: end procedure

7: procedure $\text{ENUMERATERECURSIVE}(A, j, p)$
8: if $j \geq n_c$ then
9: return
10: end if
11: for $q = [q_1, \ldots, q_{n_r}]^\top$ corresponding to $A_{.,j}$ do
12: $p_{\text{new}} = \text{EXTEND}(p, q)$
13: if $p_{\text{new}}$ is permutation and $\not\exists p' \in [p]_{\sim_w} : p' \prec p$ then
14: if $p_{\text{new}}$ is completely defined then
15: $R \leftarrow R \cup \{p_{\text{new}}\}$
16: else
17: return $\text{ENUMERATERECURSIVE}(A, j + 1, p_{\text{new}})$
18: end if
19: end if
20: end for
21: end procedure

22: procedure $\text{EXTEND}(p, q)$
23: for $j \in \mathbb{N}_{\leq n_c}$ do
24: if $p_{.,j} = [-1, \ldots, -1]^\top$ then
25: $p_{.,j} \leftarrow q$
26: return $p$
27: end if
28: end for
29: end procedure
Let now be \( k, k' \) given with \( \text{Block}(k) = \text{Block}(k') \) and \( \text{Index}(k), \text{Index}(k') \) not necessarily in the same \( T_i \). From the property that \( \mathcal{G} \) is a strongly connected directed graph, we obtain that there exists \( k^{(0)}, \ldots, k^{(t)} \) s.t. \( k = k^{(0)}, k' = k^{(t)} \), \( \text{Block}(k^{(i)}) = \text{Block}(k^{(j)}) \) for all \( i, j \leq t \), and

\[
\forall i \in \{0, \ldots, t - 1\} \exists l': \text{Index}(k^{(i)}), \text{Index}(k^{(i+1)}) \in T_{l'} .
\]

But then, \( \text{Block}(\vartheta(k^{(i)})) \) must be the same for all \( k^{(i)} \), and in particular,

\[
\text{Block}(\vartheta(k)) = \text{Block}(\vartheta(k')) .
\]

\[\square\]

**Example 4.2.** Let

\[
M = \begin{pmatrix}
0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0
\end{pmatrix}
\]

be the MixColumns matrix applied in Midori. Then, the directed graph \( \mathcal{G} \) with \( n_r \) vertices having adjacency matrix \( M \) can be given as

\[
\begin{array}{cccc}
1 & 2 & 4 & 3 \\
\end{array}
\]

which is a strongly connected directed graph. \[\square\]

**Corollary 4.1.** For the Midori MixColumns matrix \( M \), the notion of weak \( M \)-equivalence coincides with the notion of \( M \)-equivalence.

### 4.3 Case Study – The Best Word Permutations for Midori

Midori operates on an \( n_r \times n_c \) state with \( n_r = n_c = 4 \), using a word size of \( s = 4 \) for the 64-bit block-size version and \( s = 8 \) for the 128-bit version, respectively. For such state dimensions, there are 501 possible structure matrices upto equivalence. The Midori MixColumns operation \( \text{Mix}_M \) has the useful property that \( \text{Permute}_\varphi \) commutes with \( \text{Mix}_M \) for all \( 2^4 \) possible permutations \( \varphi \in \mathcal{P}_M \).

Applying Algorithm 4.1 can be done efficiently and thus, all word permutations can be enumerated upto \( M \)-equivalence. One finally obtains \( 3,413,774 \approx 2^{21.7} \).
distinct equivalence classes. Out of those, 14,022 permutations correspond to the all-1 structure matrix, i.e.,

\[ A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}. \]

Note that permutations having this particular structure matrix distribute the words within a column to all different columns. By intuition, those permutation should lead to the highest number of active S-boxes.

For each of the 3,413,774 distinct word permutations \( p \), we now want to evaluate the cryptographic properties of the corresponding cipher that is obtained by substituting the word permutation of \( \text{Midori} \) by \( p \). In particular, we would like to compute an exact lower bound on the minimum number of active S-boxes.

### 4.3.1 Computing the Minimum Number of Active S-boxes

In order to find the exact lower bounds on the minimum number of active S-boxes for all of the \( 2^{21.7} \) candidates, we applied Matsui’s algorithm [Mat95] (see Algorithm 2.2). In particular, the MixColumns matrix of \( \text{Midori} \) has the interesting property that only a very limited number of branching transitions are possible. For instance, 2 active words in a column will never lead to 3 active words after applying MixColumns. Figure 4.2 shows all 51 possible MixColumns transitions for a single column.

This leads to a very efficient running time of Matsui’s algorithm which allows to compute the bounds for the permutations for up to 40 rounds within a few days on a CPU cluster.\(^{1}\) Our most interesting observation is that the \( \text{Midori} \) word permutation is in fact not optimal for all number of rounds. For instance, there are four permutations up to equivalence that guarantee 44 active S-boxes for 9 rounds, while the permutation used in \( \text{Midori} \) only guarantees 41. Furthermore, up to 40 rounds, the \( \text{Midori} \) word permutation is never optimal from 13 rounds onwards. Instead, there are two alternative permutations that are optimal for most of the number of rounds up to 40. Interestingly, there does not exists a word permutations that is optimal for all of the number of rounds simultaneously. Some optimal permutations are listed in Table 4.1.

In the case of \( \text{Midori} \), the designers only looked at a subset of word permutations, denoted \( S_{\text{opt}} \), which they called “optimal”, by first filtering all row-based permutations according to Condition 1 and then applying a column permutation for which Condition 2 or Condition 3 holds. We recall those conditions as stated in [BBI+15, pp. 15-16].

\(^{1}\)If a permutation reached only 40 or less active S-boxes over 10 rounds, the computation of the bounds for more rounds was aborted. The overall running time for all of the \( 2^{21.7} \) permutations was roughly 1600 CPU days.
Figure 4.2: This figure shows all of the 51 possible (non-trivial) transition patterns of the MixColumn matrix in Midori.

Table 4.1: Some classes of permutations that, under the $\text{Mix}_M$ operation of Midori, lead to optimal bounds on the number of active S-boxes. All of the permutations have the all-1 structure matrix. An optimal bound for a particular number of rounds is emphasized in red. Here, optimal refers to the best bound over all permutations that have more than 40 active S-boxes over 10 rounds. The first line represents the equivalence class of the Midori permutation.
• **Condition 1**: After applying a cell-permutation once and twice, each input cell in a column is mapped into a cell in the different column.

• **Condition 2**: After applying a cell-permutation twice and twice inversely, each input cell in a column is mapped into a cell in the same row.

• **Condition 3**: After applying a cell-permutation once and three times inversely, each input cell in a column is mapped into a cell in the same row.

For our optimal word permutations for 9 rounds (and all permutations in this equivalence class), we checked whether any of them contains a member in \( \mathcal{S}_{opt} \). This is not the case, so it shows that these conditions are neither strictly necessary nor sufficient to maximize the number of active S-boxes.

### 4.4 Conclusion and Open Problems

In this chapter, we have seen that it is feasible to classify all word permutations for some lightweight AES-like ciphers like Midori and to find the optimal word permutations with respect to the minimum number of active S-boxes. We showed how the full search space can be reduced by classifying all the word permutations up to a reasonable notion of equivalence.

We provided an efficient algorithm for finding all those equivalence classes and then determined the exact bound on the minimum number of active S-boxes using Matsui’s approach. This demonstrates that, for certain designs including Midori, mCrypton and Mantis, it is feasible to cover all choices for the word permutation. We further provided several permutations which outperform the original word permutation used in Midori. For future work, it would be interesting to derive a further understanding on how optimal word permutations could be generated, e.g., in a similar way than the Midori designers generated the permutations from Conditions 1–3 above. We emphasize that this seems to be a very difficult problem.

Overall, we think the methods presented in this chapter will be particular useful for future designs as they allow to explore the whole design space for word permutations. For future work, it would be interesting to analyze other AES-like ciphers, for instance Skinny. However, due to the specific Mix\(M\) operation, the classification algorithm would lead to more classes that have to be handled with Matsui’s algorithm. On the other hand, Matsui’s algorithm for computing the bounds would be more efficient for the Mix\(M\) operation used in Skinny due to the even more limited possible MixColumns transitions.

Another topic for further research would be to study in which scenarios MILP is preferable over Matsui’s approach in terms of running time for computing bounds on the number of active S-boxes.
Chapter 5

The Tweakable Block Ciphers
Skinny and Mantis

The two designs Skinny and Mantis, described in Section 5.3 and Section 5.4, were previously published in [BJK+16a] which is joint work with Jérémie Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich and Siang Meng Sim. The author contributed to the design of the Skinny family of block ciphers, with a focus on the design of the linear layer and the choice of the tweakey permutation for maximizing the minimum number of active S-boxes in the related-tweakey (TK1) setting. Further, he contributed to the design of the low-latency cipher Mantis and derived the dedicated MILP model for computing the minimum number of active S-boxes in the related-tweak setting.

5.1 Introduction

In 2013, the National Security Agency (NSA) presented the Simon family of lightweight block ciphers [BSS+13]. The ciphers of those family consist of a very lightweight round function and offer extremely good performance on a variety of platforms. However, due to its innovative design, standard security arguments (e.g., those according to the wide-trail strategy) do not apply and the security analysis with regard to differential and linear attacks is not straightforward. Unfortunately, the designers did not provide any security analysis in the design document and all the cryptanalysis was basically conducted by external researchers. For instance, in [KLT15], the authors derive bounds on the probability of differential trails (resp. square correlation of linear trails) based on SAT/SMT solvers.

1The original article published by Springer-Verlag is available at DOI: 10.1007/978-3-662-53008-5_5 (© IACR 2016). A full version is available at eprint.iacr.org/2016/660 We omit many parts here, especially those that are not related to the author’s contribution.
In the more complex related-key adversary model, a model in which the adversary is allowed to query encryptions under different related keys, no such bounds are known.

Although no significant weaknesses of Simon have been published to date, it would be beneficial to have an alternative cipher at hand that competes with the performance of Simon, while additionally providing strong security arguments, even in the related-key model.

Outline of this Chapter

We first give a brief introduction into tweakable block ciphers and their formal security notions. Basically, a tweakable block cipher receives an additional (public) input, called the tweak, that serves as a parameter for achieving variability. We also explain the related-key adversary model for (classical) block ciphers. We then refrain from the formal security notions and explain adversaries that adhere to the more specific related-key differential attack. Those type of attacks allow the adversary to insert differences not only within the plaintext, but also within the key. They are by far the most important when it comes to argue on the resistance against related-key attacks. Similar to differential attacks in the single-key model, there are standard arguments used by block cipher designers for providing evidence on the resistance against related-key differential attacks. We then explain the TWEAKEY framework which was introduced as a unification of tweakable block cipher designs and the design of classical block ciphers resistant against related-key attacks.

In Section 5.3, we show Skinny, a family of lightweight tweakable block ciphers. It comes in different versions to support block length of 64-bit and 128-bit and different key/tweak sizes. The design according to the TWEAKEY framework allows the usage of Skinny as a flexible tweakable block cipher. Skinny is a design from academia whose goal is to compete with the NSA design Simon in terms of hardware/software performance, while in addition providing easy and strong security arguments on the resistance against differential and linear attacks. In comparison to Simon (which not allows a tweak input), we also consider the related-key/related-tweak model in the security analysis. To reach our goal, Skinny uses well-understood design principles. All components are optimized for performance, with lots of parts not strictly necessary for security being removed. After giving the specification of the Skinny family and the motivation behind the design choices, we explain the mixed-integer linear programming (MILP) approach for evaluating the security against differential and linear attacks in more detail. We then conclude the section by mentioning the best external cryptanalysis of Skinny so far.

In Section 5.4, we explain the lightweight tweakable block cipher Mantis, a design optimized for low-latency applications such as memory encryption [HT14]. Especially for the use case of memory encryption, the additional tweak input is beneficial as the memory address can be defined as the tweak. Such a low-latency block cipher should allow for a fast execution within a single clock cycle and
the overhead for additionally implementing decryption should be quite low. The
design is based on the low-latency (non-tweakable) block cipher Prince and the
block cipher Midori. After giving the specification and the motivation behind the
design choices, we explain how the security was evaluated using MILP and mention
the best external cryptanalysis so far.

5.2 Tweakable Block Ciphers

The concept of a block cipher that includes a public parameter for achieving vari-
ability goes back to the design of the Hasty Pudding Cipher [Sch98]. This was later
formalized in the notion of a tweakable block cipher [LRW02, LRW11]. Formally,
a tweakable block cipher can be defined as a family of block ciphers parametrized
by a public parameter, called the tweak.

Definition 5.1 (Tweakable Block Cipher). Let $n, \kappa, \tau \in \mathbb{N}$. An $(n, \kappa, \tau)$-tweakable
block cipher is a function

$$
\tilde{E}: \mathbb{F}_2^n \times \mathbb{F}_2^\kappa \times \mathbb{F}_2^\tau \rightarrow \mathbb{F}_2^n
$$

with the property that, for each $h \in \mathbb{F}_2^\tau$, the projection $h\tilde{E} := \tilde{E}(\cdot, \cdot, h)$ is a block
cipher. Thereby, $h$ is called the tweak. For a key $k \in \mathbb{F}_2^\kappa$, we denote by $\tilde{E}_k$ the
projection $\tilde{E}(\cdot, k, \cdot)$ and refer to it as a keyed instance of the tweakable block cipher
$\tilde{E}$. By $h\tilde{E}_k := \tilde{E}(\cdot, k, h)$, we denote the keyed instance of $\tilde{E}$ with key $k$ and tweak $h$.

There seems to be no substantial difference between the definition of an $(n, \kappa, \tau)$-
tweakable block cipher and the definition of an $(n, \kappa + \tau)$-block cipher, as a
key/tweak pair $(k, h)$ could be simply considered as one element $k||h \in \mathbb{F}_2^{\kappa+\tau}$,
serving as the key. The main point of separating the notion of a tweakable block
cipher from that of a classical block cipher is that the key is kept secret and
the tweak is assumed to be public information that serves as a parameter for
achieving variability of the actual instance. The motivation of allowing this ad-
ditional variability, as outlined in [LRW11], is that variability is needed at the
mode-of-operation level. For instance, when we recall the example of the CTR
mode (Equation 2.1), a counter is used for varying the encryption functions in
each block. The authors suggested that the source of variability should be directly
incorporated in the block cipher itself (as an example of a mode of operation,
each block could be encrypted with the same tweakable block cipher and different
counters are incorporated as the tweaks). Such a tweakable block cipher should
then be designed in a way that allows to change the tweak more efficiently than
changing the key.

The formal security notion of a tweakable block cipher has to model the tweak
as public information. More precisely, for a key $k$ chosen uniformly at random,
the keyed instance $\tilde{E}_k$ should be indistinguishable from a family of permutations
chosen independently and uniformly at random. This is formalized in the next
definition. Let us define $\text{Perm}_{n,\tau}$ as the set of all $(n, \tau)$-block ciphers, i.e., the set
of all families of $2^\tau$ permutations on $\mathbb{F}_2^n$ that are parametrized by $h \in \mathbb{F}_2^\tau$. 

95
Definition 5.2 (Pseudorandomness of a Tweakable Block Cipher (Definition 3 in \cite{LRW11})). Let $\tilde{E}$ be an $(n, \kappa, \tau)$-tweakable block cipher. Let $A_{q,t}$ be a $(q,t)$-adversary with oracle access to an element of $\text{Perm}_{n,\tau}$. The TPRP advantage of $A_{q,t}$ against $\tilde{E}$ is defined as

$$\text{Adv}_{\tilde{E}}^\text{TPRP-CPA}(A_{q,t}) := \text{Prob}_{k \in \mathbb{F}_2^n}^{\tilde{E}_k}(A_{q,t}^{\tilde{E}_k} \Rightarrow 1) - \text{Prob}_{\Pi \leftarrow \text{Perm}_{n,\tau}}(A_{q,t}^{\Pi} \Rightarrow 1).$$

Here, $\Pi \leftarrow \text{Perm}_{n,\tau}$ denotes that $\Pi$ is chosen as a family of $2^\tau$ independent, uniformly random permutations on $\mathbb{F}_2^n$. Thus, the probabilities are defined over the uniform choices of $k$ and $\Pi$ and over the random choices that the probabilistic adversary $A_{q,t}$ makes. Similar to the notion of a pseudorandom permutation for a block cipher, one considers $\tilde{E}$ to be secure (also called chosen-tweak secure) if, for reasonable restrictions on the computational resources $q,t$,

$$\max_{A_{q,t}} \text{Adv}_{\tilde{E}}^\text{TPRP-CPA}(A_{q,t}) \leq \epsilon$$

for a sufficiently small $\epsilon$. One can moreover consider CCA adversaries that have, like in the notion of a strong pseudorandom permutation, also oracle access to the inverse permutations (see \cite{LRW11, Definition 4}).

Examples of existing tweakable block ciphers in the literature can be divided into two classes. The first class constructs a tweakable block cipher from underlying primitives (e.g., classical block ciphers), see for instance the LRW construction \cite{LRW02} or the XE and XEX construction \cite{Rog04}. The security of the tweakable block cipher is then proven by reducing to the security of the underlying primitive. The second class contains tweakable block ciphers designed from scratch, i.e., block cipher designs that directly support the tweak input. Examples include the Hasty Pudding Cipher \cite{Sch98}, the ciphers Deoxys-BC, Joltik-BC and Kiasu-BC proposed along with the TWEAKEY framework \cite{JNP14}, or the low-latency tweakable block cipher QARMA \cite{Ava17}.

5.2.1 Related-Key Attacks

Related-key attacks \cite{Bih94a, Bih94b} refer to a special kind of adversary model on block ciphers. In a nutshell, this model allows the adversary to query different instances of the cipher which are related in some way. The goal of the adversary is to distinguish the original keyed instance from a random permutation. Bellare and Kohno formalized the related-key adversary model in \cite{BK03a} and introduced the notion of pseudorandomness with respect to related-key attacks. Formally, the adversary interacts with a special kind of oracle that exactly models the queries to related instances according to a previously defined relation.

Definition 5.3 (Related-Key Oracle \cite{BK03a}). Let $E$ be an $(n, \kappa)$-block cipher. For a keyed instance $E_k$, a related-key oracle $E_{RK,k}$ is defined as an oracle which takes as arguments a function $\phi : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and an element $x \in \mathbb{F}_2^n$, and returns the value of $E_{\phi(k)}(x)$ whenever queried at inputs $\phi$ and $x$. 
In this context, $\phi$ is also called a related-key deriving function. The notion of pseudorandomness in the related-key model will restrict the adversary to oracle queries of the form $(\phi, x)$, where the related-key deriving function $\phi$ must belong to a previously defined set of allowed functions.

**Definition 5.4** (Pseudorandomness with Respect to Related-Key Attacks (Definition 1 in [BK03a]). Let $E$ be an $(n, \kappa)$-block cipher and let $\Phi = \{\phi_i: \mathbb{F}^n_2 \rightarrow \mathbb{F}^n_2\}_i$ be a set of related-key deriving functions. Let $A_{q,t}$ be a $(q, t)$-adversary with oracle access to a related-key oracle and being restricted to queries of the form $(\phi, x)$, for $\phi \in \Phi$ and $x \in \mathbb{F}^n_2$. The $\Phi$-restricted related-key advantage of $A_{q,t}$ against $E$ is defined as

$$
\text{Adv}_{\Phi, E}^{\text{PRP-RKA}}(A_{q,t}) := \text{Prob}_{k \leftarrow \mathbb{F}^n_2} (A_{q,t}^{E_{\text{RK}, k}} \Rightarrow 1) - \text{Prob}_{\Pi \leftarrow \text{Perm}_n, k \leftarrow \mathbb{F}^n_2} (A_{q,t}^{\Pi_{\text{RK}, k}} \Rightarrow 1).
$$

The probabilities are defined over the uniform choices of $k$ and $\Pi$ and over the random choices that the probabilistic adversary $A_{q,t}$ makes. Similar to all other security notions we have mentioned, one considers the block cipher $E$ secure against $\Phi$-restricted related-key attacks if, for reasonable restrictions on the computational resources $q, t$, the maximum $\Phi$-restricted related-key advantage over all $q, t$ adversaries is sufficiently small. Moreover, the above definition can easily be extended to adversaries that are given related-key oracle access to the inverse permutations as well (see [BK03b, Definition 8.1]).

Bellare and Kohno formally showed several impossibility results for achieving related-key security. In particular, the set $\Phi$ of allowed related-key deriving functions must be defined in a way that excludes trivial attacks.

**Example 5.1** ((Proposition 4.1 in [BK03b])). Let $E$ be an $(n, \kappa)$-block cipher and let $c \in \mathbb{F}^n_2$. Let $\Phi$ be a set of related-key deriving functions that contains the constant function $\phi_c: k \mapsto c$. We consider the adversary $A$ with oracle access to a related-key oracle $O_{\text{RK}, k}$ given in Algorithm 5.1 below. It only makes a single oracle query (for the function $\phi_c$ contained in $\Phi$) and has basically the running time of one block cipher call. For the advantage, it is

$$
\text{Adv}_{\Phi, E}^{\text{PRP-RKA}}(A) := \text{Prob}_{x \leftarrow \mathbb{F}^n_2} (E_{\phi_c}(x) = E_c(x)) - \text{Prob}_{\pi \leftarrow \text{Perm}_n} (\pi(x) = E_c(x))
$$

$$
= 1 - \frac{1}{2^n}.
$$

This example illustrates that it is impossible to achieve resistance against all types of related-key attacks. The authors further showed impossibility results for more natural choices of $\Phi$. As a more important example of practically-relevant related-key deriving functions, they studied the set

$$
\Phi^\oplus := \{\phi: \mathbb{F}^n_2 \rightarrow \mathbb{F}^n_2 \mid \exists \iota \in \mathbb{F}^n_2 \text{ such that } \phi: k \mapsto k + \iota\} = \{\text{Add}_\iota\}_{\iota \in \mathbb{F}^n_2}
$$
Algorithm 5.1 Adversary \( \mathcal{A} \)

1: Choose \( x \leftarrow \mathbb{F}_2^n \)
2: Compute \( y \leftarrow \mathcal{O}_{\text{RK},k}(\phi_c, x) \)
3: if \( y = E_c(x) \) then
4:    return 1
5: end if
6: return 0

in further detail. Especially at the protocol- or mode-of-operation level, related-key access with regard to related-key deriving functions in \( \Phi_{\kappa}^{\oplus} \) might be a reasonable assumption on the power of the adversary and most block cipher designs (especially those not designed for lightweight purposes) aim for security against \( \Phi_{\kappa}^{\oplus} \)-restricted related-key attacks. Moreover, Bellare and Kohno formally proved that a block cipher which is secure against \( \Phi_{\kappa}^{\oplus} \)-restricted related-key adversaries gives rise to a chosen-tweak secure tweakable block cipher. In particular, if one has a block cipher resistant against \( \Phi_{\kappa}^{\oplus} \)-restricted related-key attacks, one can construct a tweakable block cipher by just XOR-ing the tweak to the key.

Theorem 5.1 (Theorem 7.1 in [BK03b]). Let \( E \) be an \((n, \kappa)\)-block cipher and let \( \tilde{E} \) be the \((n, \kappa, \kappa)\)-tweakable block cipher defined by \( h\tilde{E}_k = E_{k+h} \). Let \( \mathcal{A} \) be an adversary with oracle access to an element of \( \text{Perm}_{n, \kappa} \), one can construct a \( \Phi_{\kappa}^{\oplus} \)-restricted PRP-RKA-adversary \( \mathcal{B} \) against \( E \), with roughly the same number of oracle queries and running time, such that

\[
\text{Adv}^{\text{TPRP-CPA}}_{\tilde{E}}(\mathcal{A}) \leq \text{Adv}^{\text{PRP-RKA}}_{\Phi_{\kappa}^{\oplus}, E}(\mathcal{B}) .
\]

Related-Key Differential Attacks

When we consider related-key attacks for the set of related-key deriving functions \( \Phi_{\kappa}^{\oplus} \) and especially aim for designing a cipher resistant against \( \Phi_{\kappa}^{\oplus} \)-restricted related-key attack, a natural type of adversary to consider is a related-key differential attack [KSW96]. In contrast to the differential attack in the single-key model, the adversary is not only allowed to insert differences in the plaintext, but also in the key.

Definition 5.5 (Related-Key Differential). Let \( E \) be an \((n, \kappa)\)-block cipher. A differential \( ((\alpha, \iota), \beta) \) over \( E \) with \( \alpha, \beta \in \mathbb{F}_2^n \) and \( \iota \in \mathbb{F}_2^\kappa \) is said to be a related-key differential over \( E \).

One can give the related-key differential probability as

\[
\text{Prob}((\alpha, \iota) \xrightarrow{E} \beta) = \frac{|\{(x, k) \in \mathbb{F}_2^n \times \mathbb{F}_2^\kappa \mid E_k(x) + E_{k+\iota}(x + \alpha) = \beta\}|}{2^{n+\kappa}}.
\]

If the adversary wants to distinguish a keyed instance \( E_k \) of the cipher from a random permutation and is in possession of a related-key differential \( (\alpha, \iota) \xrightarrow{E} \beta \)
that holds with probability $> 2^{-n}$, it would query the related-key oracle on inputs
$(\text{Add}_0: k \mapsto k + 0, x)$ and $(\text{Add}_{\iota}: k \mapsto k + \iota, x + \alpha)$ for many $x \in \mathbb{F}_2^n$ and check
whether the XOR difference of the two oracle responses equals $\beta$ as often as one
would expect by the differential probability.

If $E$ is defined as a product cipher, similar as in the case of single-key differential
attacks, a security argument on the resistance against related-key differential
attacks is usually based on a single differential trail. In the related-key model, a
differential trail not only fixes the particular output differences after every round,
but also the differences of all the round keys. In particular, if $R_0, \ldots, R_t$ denote the
rounds of the product cipher (where $R_i: \mathbb{F}_2^n \times \mathbb{F}_2^{\kappa_i} \rightarrow \mathbb{F}_2^n$), as a security argument
one computes an upper bound on the value of

$$\Pr(t \xrightarrow{\text{KeySchedule}} (t_0, \ldots, t_t)) \prod_{i=0}^t \Pr((\alpha_i, t_i) \xrightarrow{R_i} \alpha_{i+1})$$

over all $t \in \mathbb{F}_2^n, \alpha_i \in \mathbb{F}_2^{\kappa_i}$, and $(t, \alpha_0) \neq (0, 0)$. In the case of a
t-round key-alternating cipher with unkeyed round functions $R_0, \ldots, R_t$, Equation
5.1 reduces to

$$\Pr(t \xrightarrow{\text{KeySchedule}} (t_0, \ldots, t_t)) \prod_{i=0}^t \Pr(\alpha_i \xrightarrow{R_i} \alpha_{i+1} + t_i).$$

Note that, whenever we fix $t = 0$ in Equation 5.2, this is exactly the security
argument on the resistance against single-key differential attacks. By additionally
considering all possible differences $t$ in the key, we include much more powerful
adversaries.

As a standard approach, the designer of a key-alternating cipher aiming for
resistance against related-key differential attacks would evaluate the number of
rounds $t$ for which Equation 5.2 can be upper bounded by $2^{-n}$, considering all
differences in the round inputs/outputs and all differences in the round keys (with
the restriction $(\alpha_0, t) \neq (0, 0)$ in order to avoid a trivial differential trail), and
specify the actual number of rounds of the cipher as $t + t_m$ for a reasonable security
margin $t_m$. As for single-key differential attacks, several automatic search tools
exist for finding the best differential trails and for deriving those upper bounds.
For instance, for word-oriented ciphers, Biryukov and Nikolić developed a tool for
finding related-key differential trails based on Matsui’s approach [BN10]. A search
algorithm based on MILP is presented in [SHW+14b].

We would like to mention that, for instance, there exist related-key attacks
(based on differential cryptanalysis) on AES-192 and AES-256 [BKN09] [BK09].

The Wide-Trail Approach

When the rounds of the cipher are defined as $R_i = L \circ S$ for a bijective linear
layer $L$ and for $S$ consisting of a parallel application of a bijective $s$-bit S-box $S_b$,
similar as already described in Section 2.4 for the single-key model, the probability
given in Equation 5.2 can be upper bounded by computing the minimum number of active S-boxes. In particular,

\[
\prod_{i=0}^{t} \text{Prob}(\alpha_i \rightarrow R_i \alpha_{i+1} + \iota_i) = \prod_{i=0}^{t} \text{Prob}(\alpha_i \rightarrow S_{i} L^{-1}(\alpha_{i+1}) + L^{-1}(\iota_i)) \leq P_{\text{st}}^{w_s(C)},
\]

where \(C = (\alpha_0, \ldots, \alpha_t)\) and \(w_s(C) = \sum_{i=0}^{t-1} w_s(\alpha_i)\).

The difference to the single-key model is that it has to be taken care of the addition of the differences in the key input, i.e., valid trails are not defined as those for which, for all \(i \in \{0, \ldots, t\}\), \(\text{Prob}(\alpha_i \rightarrow R_i \alpha_{i+1}) \neq 0\), but those for which \(\text{Prob}(\alpha_i \rightarrow R_i \alpha_{i+1} + \iota_i) \neq 0\), where \((\iota_0, \ldots, \iota_t)\) defines a possible output difference over the key schedule. If the key schedule is an affine function, the differences \(\iota_0, \ldots, \iota_t\) are completely determined by the initial key-input difference \(\iota\). In other words, for each \(i \in \mathbb{F}_2^t\), there is only one tuple \((\iota_0, \ldots, \iota_t)\) for which the differential probability \(\text{Prob}(\iota \rightarrow \text{KeySchedule} \leftarrow (\iota_0, \ldots, \iota_t)) \neq 0\). In fact, that differential probability is equal to one.

For Skinny and Mantis, the key schedule is indeed affine (linear up to addition of round constants), which renders the computation of bound on the minimum number of active S-boxes very efficient. After explaining those designs, we describe the MILP model for evaluating the bounds in Sections 5.3.3 and 5.4.5 respectively.

### 5.2.2 The TWEAKEY Framework

In [JNP14], the TWEAKEY framework was proposed in order to unify the design of block ciphers that resist related-key attacks and the design of tweakable block ciphers. A drawback of constructing a tweakable block cipher from a \(\Phi^{\oplus}_{\kappa}\)-restricted related-key secure block cipher by simply adding the tweak to the key, as given in Theorem 5.1, is that one loses the related-key security of the tweakable block cipher, i.e., changing the tweak \(h \rightarrow h + \delta\) and the key \(k \rightarrow k + \delta\) results in the same instance of the cipher. Instead, the idea of TWEAKEY is to consider the key and the tweak as basically the same type of input. In particular, the idea is to design an \((n, \kappa + \tau)\)-block cipher \(E\) and use it as an \((n, \kappa, \tau)\)-tweakable block cipher \(E\) by setting \(hE_k = E_k || h\), i.e., parts of the \(\kappa + \tau\)-bit key are assumed to be a public tweak. This allows to obtain tweakable block cipher designs with a flexible tweak length. It was suggested that such a design should be of a key-alternating construction with a very efficient key schedule. Further, as security of \(E\) against \(\Phi^{\oplus}_{\kappa + \tau}\)-restricted related-key attacks is necessary in order to employ it as such a flexible tweakable block cipher, the design should allow for arguments on the resistance against related-key differential attacks.

More precisely, the TWEAKEY framework specifies the key-scheduling algorithm, also called the `tweakey schedule`, as follows. From an initial input \(tk \in \mathbb{F}_2^{\kappa + \tau}\), called the `tweakey`, the round keys to be added in the key-alternating construction are generated by the iteration of an update function \(\text{upd}: \mathbb{F}_2^{\kappa + \tau} \rightarrow \mathbb{F}_2^{\kappa + \tau}\), an
extraction function $\text{extr}: \mathbb{F}_2^{\kappa+\tau} \rightarrow \mathbb{F}_2^n$, and the addition of a round constant. In particular, the round key $k_r$ is obtained from $tk$ by\(^2\)

$$k_r = \text{extr}(\text{upd}'(tk)) + c_r .$$

For AES-like ciphers, the authors presented the superposition tweakey (STK) construction as an efficient instance of the TWEAKEY framework that allows for an easier security analysis using automatic tools. In that construction, the length of the tweakey input $tk$ is a multiple of $n = s \cdot n_r \cdot n_c$. The STK tweakey schedule separates the tweakey input $tk$ into distinct AES-like $n_r \times n_c$ states with $s$-bit words. Then, $\text{upd}$ operates as a permutation of the words of each state, followed by a transformation operating separately on each $s$-bit word of the states.\(^3\) The extraction function $\text{extr}$ is simply defined as the $\mathbb{F}_2$-addition of each $s \cdot n_r \cdot n_c$-bit state. In the next section, we explain Skinny which was designed as a family of lightweight block ciphers following the STK construction.

### 5.3 The Skinny Family of Lightweight (Tweakable) Block Ciphers

In the following, we interchangeable use the expressions key, tweak and tweakey, as they basically refer to the same type of input.

The Skinny family of lightweight tweakable block ciphers comes in six different versions, supporting block length of $n = 64$ and $n = 128$ and tweakey length of $\kappa \in \{n, 2n, 3n\}$, respectively. We denote the $(n, \kappa)$-version of the cipher by Skinny-$n$-$\kappa$. As almost all block ciphers, Skinny is designed as a product cipher. In particular, all the versions of Skinny mentioned above can be given as

$$\text{Skinny-}n$-$\kappa: \mathbb{F}_2^n \times \mathbb{F}_2^\kappa \rightarrow \mathbb{F}_2^n$$

$$(x, k) \mapsto R_{t_k_t} \circ \cdots \circ R_{1k_1}(x) .$$

Thereby, the round keys $k_1, \ldots, k_t \in \mathbb{F}_2^n$ are derived from the initial key $k$ by a key-scheduling algorithm $\text{KeySchedule}: \mathbb{F}_2^\kappa \rightarrow (\mathbb{F}_2^n)^t$. The actual definition of the rounds $R_i$, the key-scheduling algorithm, and the number of rounds $t$ depends on the particular Skinny version. We outline their specifications in the following.\(^4\)

### 5.3.1 Specification

In all versions, the internal state is represented by a $4 \times 4$ array of words in $\mathbb{F}_2^s$, with $s = 4$ for $n = 64$, and $s = 8$ for $n = 128$. Therefore, we denote an internal

\(^2\)In the TWEAKEY paper, the addition of the round constant was not part of the tweakey schedule. Instead it was part of the cipher’s round functions.

\(^3\)In the TWEAKEY paper, this transformation is a finite field multiplication with pre-defined elements $\alpha_i \in \mathbb{F}_2^s$.

\(^4\)In contrast to the original design document, we give a slightly different representation that includes the round constants within the key schedule.
state \( x \in \mathbb{F}_2^{s-16} \) as
\[
  x = \begin{bmatrix}
    x_1 & x_5 & x_9 & x_{13} \\
    x_2 & x_6 & x_{10} & x_{14} \\
    x_3 & x_7 & x_{11} & x_{15} \\
    x_4 & x_8 & x_{12} & x_{16} 
  \end{bmatrix}, \quad x_i \in \mathbb{F}_2^s.
\]

Note that the initial state is loaded in row-wise manner, i.e., a plaintext \( m = m_1 || \ldots || m_{16} \) with \( m_i \in \mathbb{F}_2^s \) is mapped to the cipher’s initial state as
\[
  m \mapsto \begin{bmatrix}
    m_1 & m_2 & m_3 & m_4 \\
    m_5 & m_6 & m_7 & m_8 \\
    m_9 & m_{10} & m_{11} & m_{12} \\
    m_{13} & m_{14} & m_{15} & m_{16} 
  \end{bmatrix}.
\]

The design of the key schedule of the Skinny family follows the TWEAKEY framework, more precisely the STK construction. As already mentioned, each Skinny-\( n \) family naturally comes in three tweakey length versions, i.e., \( n, 2n, \) or \( 3n \). Therefore, the tweakey state will be represented as a collection of \( z \) distinct \( 4 \times 4 \) arrays of words in \( \mathbb{F}_2^s \). We denote these arrays \( TK^{(1)} \) when \( z = 1 \), \( TK^{(2)} \) when \( z = 2 \), and \( TK^{(3)} \) when \( z = 3 \). Moreover, by \( SK \), we denote the single-key adversary model and by \( TK^{(1)}, TK^{(2)} \) or \( TK^{(3)} \), we denote the adversary model in which the attacker can introduce differences in the particular tweakey states.

The cipher gets an initial key input \( k = \xi_1 || \xi_2 || \ldots || \xi_{16z} \) with \( \xi_i \in \mathbb{F}_2^s \). The initialization of the cipher’s initial tweakey state is then done in a row-wise manner by simply setting, for all \( j \in \{1, \ldots, z\} \),
\[
  TK^{(j)} = \begin{bmatrix}
    \xi_{16(j-1)+1} & \xi_{16(j-1)+2} & \xi_{16(j-1)+3} & \xi_{16(j-1)+4} \\
    \xi_{16(j-1)+5} & \xi_{16(j-1)+6} & \xi_{16(j-1)+7} & \xi_{16(j-1)+8} \\
    \xi_{16(j-1)+9} & \xi_{16(j-1)+10} & \xi_{16(j-1)+11} & \xi_{16(j-1)+12} \\
    \xi_{16(j-1)+13} & \xi_{16(j-1)+14} & \xi_{16(j-1)+15} & \xi_{16(j-1)+16} 
  \end{bmatrix}.
\]

Note that, whenever parts of the key material is dedicated for a public tweak, the user must ensure that the length of the actual secret key is always at least as big as the block length \( n \).

The Round Function

Table 5.1 specifies the number of rounds \( t \) that are applied in the particular Skinny version. All of the rounds \( R_i \) are defined by four operations, i.e., \( S_{Sb} \), Permute\( p \), Mix\( M \), and a round key addition, as
\[
  R_i : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n \\
  (x, k_i) \mapsto \text{Mix}_M \circ \text{Permute}_p \circ \text{Add}_{k_i} \circ S_{Sb}(x) .
\]

Those operations are defined as follows, depending on the block length \( n \).
Table 5.1: Number of rounds $t$ for Skinny-$n$-$\kappa$.

<table>
<thead>
<tr>
<th>Block length $n$</th>
<th>Tweakey length $\kappa$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n$</td>
</tr>
<tr>
<td>64</td>
<td>$t = 32$</td>
</tr>
<tr>
<td>128</td>
<td>$t = 40$</td>
</tr>
</tbody>
</table>

$S_{\text{Sb}_4}$ (SubCells). For the versions with $n = 64$, the 4-bit S-box $S_{\text{b}_4}: \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$ as given in Table 5.2 is applied to every word of the cipher’s internal state. Its construction is similar to the S-box employed in the lightweight block cipher Piccolo [SIH+11].

Table 5.2: The 4-bit S-box $S_{\text{b}_4}$ employed in the Skinny-64 versions.

<table>
<thead>
<tr>
<th>$x$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{\text{b}_4}(x)$</td>
<td>C</td>
<td>6</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>A</td>
<td>2</td>
<td>B</td>
<td>3</td>
<td>8</td>
<td>5</td>
<td>D</td>
<td>4</td>
<td>E</td>
<td>7</td>
<td>F</td>
</tr>
</tbody>
</table>

Note that $S_{\text{b}_4}$ can also be described by using four bit-wise NOR operations (i.e., $x_1 \lor x_2$ for $x_1, x_2 \in \mathbb{F}_2$) and four bit-wise XOR operations (i.e., addition in $\mathbb{F}_2$). If $(x_3, x_2, x_1, x_0) \in \mathbb{F}_2^4$ represents the S-box input, one simply applies the transformation

$$(x_3, x_2, x_1, x_0) \mapsto ((x_3, x_2, x_1, x_0 + (x_3 \lor x_2)) \ll 1),$$

iterated four times, except that the last iteration omits the cyclic rotation by one bit.

For the versions with $n = 128$, the 8-bit S-box $S_{\text{b}_8}: \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^8$ as given in Table 5.3 is applied to every word of the cipher’s internal state. It is constructed in a similar way as $S_{\text{b}_4}$ described above. In particular, if $(x_7, x_6, \ldots, x_0) \in \mathbb{F}_2^8$ represents the S-box input, $S_{\text{b}_8}$ applies the transformation

$$(x_7, x_6, \ldots, x_0) \mapsto (x_2, x_1, x_7, x_6, x_4 + (x_7 \lor x_6), x_0 + (x_3 \lor x_2), x_3, x_5),$$

which is iterated three times. Then, as a final step, the transformation

$$(x_7, x_6, \ldots, x_0) \mapsto (x_7, x_6, x_5, x_4 + (x_7 \lor x_6), x_3, x_1, x_2, x_0 + (x_3 \lor x_2))$$

is applied.

Add$_{k_i}$ applies the addition of the round key $k_i$ to the internal state $x$. How those round keys are derived from the initial tweakey by the tweakey schedule is described below.
Table 5.3: The S-box $S_b$ used in the Skinny-128 versions. For each (hexadecimal) value of $x$ and $y$, the table shows $S_b(x||y)$ as a hexadecimal value. For instance, $S_b(A4) = 19$.

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>65</td>
<td>4C</td>
<td>6A</td>
<td>42</td>
<td>4B</td>
<td>63</td>
<td>43</td>
<td>6B</td>
<td>55</td>
<td>75</td>
<td>5A</td>
<td>7A</td>
<td>53</td>
<td>73</td>
<td>5B</td>
<td>7B</td>
</tr>
<tr>
<td>1</td>
<td>35</td>
<td>8C</td>
<td>3A</td>
<td>81</td>
<td>89</td>
<td>33</td>
<td>80</td>
<td>3B</td>
<td>96</td>
<td>25</td>
<td>98</td>
<td>2A</td>
<td>90</td>
<td>23</td>
<td>99</td>
<td>2B</td>
</tr>
<tr>
<td>2</td>
<td>E5</td>
<td>CC</td>
<td>E8</td>
<td>C1</td>
<td>C9</td>
<td>E0</td>
<td>C0</td>
<td>E9</td>
<td>D5</td>
<td>F5</td>
<td>D8</td>
<td>F8</td>
<td>D0</td>
<td>F0</td>
<td>D9</td>
<td>F9</td>
</tr>
<tr>
<td>3</td>
<td>A5</td>
<td>1C</td>
<td>A8</td>
<td>12</td>
<td>1B</td>
<td>A0</td>
<td>13</td>
<td>A9</td>
<td>05</td>
<td>B5</td>
<td>0A</td>
<td>B8</td>
<td>03</td>
<td>B0</td>
<td>0B</td>
<td>B9</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>88</td>
<td>3C</td>
<td>85</td>
<td>8D</td>
<td>34</td>
<td>84</td>
<td>3D</td>
<td>91</td>
<td>22</td>
<td>9C</td>
<td>2C</td>
<td>94</td>
<td>24</td>
<td>9D</td>
<td>2D</td>
</tr>
<tr>
<td>5</td>
<td>62</td>
<td>4A</td>
<td>6C</td>
<td>45</td>
<td>4D</td>
<td>64</td>
<td>44</td>
<td>6D</td>
<td>52</td>
<td>72</td>
<td>5C</td>
<td>7C</td>
<td>54</td>
<td>74</td>
<td>5D</td>
<td>7D</td>
</tr>
<tr>
<td>6</td>
<td>A1</td>
<td>1A</td>
<td>AC</td>
<td>15</td>
<td>1D</td>
<td>A4</td>
<td>14</td>
<td>AD</td>
<td>02</td>
<td>B1</td>
<td>0C</td>
<td>BC</td>
<td>04</td>
<td>B4</td>
<td>0D</td>
<td>BD</td>
</tr>
<tr>
<td>7</td>
<td>E1</td>
<td>C8</td>
<td>EC</td>
<td>C5</td>
<td>CD</td>
<td>E4</td>
<td>C4</td>
<td>ED</td>
<td>D1</td>
<td>F1</td>
<td>DC</td>
<td>F4</td>
<td>DC</td>
<td>F4</td>
<td>DD</td>
<td>FD</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
<td>8E</td>
<td>38</td>
<td>82</td>
<td>8B</td>
<td>30</td>
<td>83</td>
<td>39</td>
<td>96</td>
<td>26</td>
<td>9A</td>
<td>28</td>
<td>93</td>
<td>20</td>
<td>9B</td>
<td>29</td>
</tr>
<tr>
<td>9</td>
<td>66</td>
<td>4E</td>
<td>68</td>
<td>41</td>
<td>49</td>
<td>60</td>
<td>40</td>
<td>69</td>
<td>56</td>
<td>76</td>
<td>58</td>
<td>78</td>
<td>50</td>
<td>70</td>
<td>59</td>
<td>79</td>
</tr>
<tr>
<td>A</td>
<td>A6</td>
<td>1E</td>
<td>AA</td>
<td>11</td>
<td>19</td>
<td>A3</td>
<td>10</td>
<td>AB</td>
<td>06</td>
<td>B6</td>
<td>08</td>
<td>BA</td>
<td>00</td>
<td>B3</td>
<td>09</td>
<td>BB</td>
</tr>
<tr>
<td>B</td>
<td>E6</td>
<td>CE</td>
<td>EA</td>
<td>C2</td>
<td>CB</td>
<td>E3</td>
<td>C3</td>
<td>EB</td>
<td>D6</td>
<td>F6</td>
<td>DA</td>
<td>FA</td>
<td>D3</td>
<td>F3</td>
<td>DB</td>
<td>FB</td>
</tr>
<tr>
<td>C</td>
<td>31</td>
<td>8A</td>
<td>3E</td>
<td>86</td>
<td>8F</td>
<td>37</td>
<td>87</td>
<td>3F</td>
<td>92</td>
<td>21</td>
<td>9E</td>
<td>2E</td>
<td>97</td>
<td>27</td>
<td>9F</td>
<td>2F</td>
</tr>
<tr>
<td>D</td>
<td>61</td>
<td>48</td>
<td>6E</td>
<td>46</td>
<td>4F</td>
<td>67</td>
<td>47</td>
<td>6F</td>
<td>51</td>
<td>71</td>
<td>5E</td>
<td>7E</td>
<td>57</td>
<td>77</td>
<td>5F</td>
<td>7F</td>
</tr>
<tr>
<td>E</td>
<td>A2</td>
<td>18</td>
<td>AE</td>
<td>16</td>
<td>1F</td>
<td>A7</td>
<td>17</td>
<td>AF</td>
<td>01</td>
<td>B2</td>
<td>0E</td>
<td>BE</td>
<td>07</td>
<td>B7</td>
<td>0F</td>
<td>BF</td>
</tr>
<tr>
<td>F</td>
<td>E2</td>
<td>CA</td>
<td>EE</td>
<td>C6</td>
<td>CF</td>
<td>E7</td>
<td>C7</td>
<td>EF</td>
<td>D2</td>
<td>F2</td>
<td>DE</td>
<td>FE</td>
<td>D7</td>
<td>F7</td>
<td>DF</td>
<td>FF</td>
</tr>
</tbody>
</table>
Permutes (ShiftRows) operates as a permutation of the words of the state. Similar to the AES, this operation cyclically rotates the words within the rows of the state by 0, 1, 2, and 3 positions, respectively. However, the rotations are to the right, instead of to the left. Formally,

\[
\begin{bmatrix}
x_1 & x_5 & x_9 & x_{13} \\
x_2 & x_6 & x_{10} & x_{14} \\
x_3 & x_7 & x_{11} & x_{15} \\
x_4 & x_8 & x_{12} & x_{16} \\
\end{bmatrix}
\mapsto
\begin{bmatrix}
x_7 & x_{16} & x_3 & x_{15} \\
x_1 & x_5 & x_9 & x_{13} \\
x_{14} & x_2 & x_6 & x_{10} \\
x_{11} & x_{15} & x_3 & x_7 \\
\end{bmatrix},
\]

which corresponds to the permutation of indices

\[
p = (1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16).
\]

MixM (MixColumns). Each column of the state is multiplied by the matrix

\[
M = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \in \text{GL}_4(\mathbb{F}_2).
\]

The final value of the internal state array (after the last MixM operation) then provides the ciphertext with the state words being mapped to the ciphertext row by row. Test vectors for Skinny can be found in \cite{BJK+16b}. Note that all components of the cipher have very simple inverses, thus decryption can be described and implemented in a similar way than encryption.

The Tweakey Schedule

The key-scheduling algorithm for all versions with \( z \in \{1, 2, 3\} \) is given as Algorithm 5.2. It shows how the tweakey states \( TK^{(i)} \) are updated and the particular round keys are extracted. For the update of the tweakey states, the key schedule applies the functions Permute\(_p\), LFSRupdate\(_s^{(2)}\), and LFSRupdate\(_s^{(3)}\), defined as follows:

Permute\(_p\) operates on a \( 4 \times 4 \) state \( TK^{(i)} \) as a permutation of the s-bit words as

\[
\begin{bmatrix}
TK_1^{(i)} & TK_5^{(i)} & TK_9^{(i)} & TK_{13}^{(i)} \\
TK_2^{(i)} & TK_6^{(i)} & TK_{10}^{(i)} & TK_{14}^{(i)} \\
TK_3^{(i)} & TK_7^{(i)} & TK_{11}^{(i)} & TK_{15}^{(i)} \\
TK_4^{(i)} & TK_8^{(i)} & TK_{12}^{(i)} & TK_{16}^{(i)} \\
\end{bmatrix}
\mapsto
\begin{bmatrix}
TK_7^{(i)} & TK_1^{(i)} & TK_{13}^{(i)} & TK_{8}^{(i)} \\
TK_{11}^{(i)} & TK_{12}^{(i)} & TK_{14}^{(i)} & TK_{15}^{(i)} \\
TK_1^{(i)} & TK_5^{(i)} & TK_9^{(i)} & TK_{13}^{(i)} \\
TK_2^{(i)} & TK_6^{(i)} & TK_{10}^{(i)} & TK_{14}^{(i)} \\
\end{bmatrix},
\]
This corresponds to the permutation of indices

\[ p_T = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 7 & 11 & 1 & 2 & 16 & 12 & 5 & 6 & 3 & 4 & 9 & 10 & 8 & 15 & 13 & 14 \end{pmatrix}. \]

LFSRupdate\(^{(2)}\) transmits each word in the first two rows of the tweakey state \( TK^{(2)} \) according to an LFSR operation. In particular, let us denote the \( j \)-th word of that tweakey state by \( TK_j^{(2)} = (x_{j,3}, x_{j,2}, x_{j,1}, x_{j,0}) \) if \( s = 4 \), and by \( TK_j^{(2)} = (x_{j,7}, \ldots, x_{j,1}, x_{j,0}) \) if \( s = 8 \), respectively. Then,

LFSRupdate\(^{(2)}\): \( TK^{(2)} \mapsto \widetilde{TK}^{(2)} \)

where, for each \( j \in \{1, 2, 5, 6, 9, 10, 13, 14\} \), \( \widetilde{TK}_j^{(2)} = (x_{j,2}, x_{j,1}, x_{j,0}, x_{j,3} + x_{j,2}) \),

and, for each other index \( j \), \( \widetilde{TK}_j^{(2)} = TK_j^{(2)} \).

If \( s = 8 \), we have

LFSRupdate\(^{(2)}\): \( TK^{(2)} \mapsto \widetilde{TK}^{(2)} \)

where, for each \( j \in \{1, 2, 5, 6, 9, 10, 13, 14\} \),

\( \widetilde{TK}_j^{(2)} = (x_{j,6}, x_{j,5}, x_{j,4}, x_{j,3}, x_{j,2}, x_{j,1}, x_{j,7} + x_{j,5}) \),

and, for each other index \( j \), \( \widetilde{TK}_j^{(2)} = TK_j^{(2)} \).

LFSRupdate\(^{(2)}\) transforms each word in the first two rows of the tweakey state \( TK^{(3)} \) according to an LFSR operation (another one as in LFSRupdate\(^{(2)}\)). In particular, let us denote the \( j \)-th word of that tweakey state by \( TK_j^{(3)} = (x_{j,3}, x_{j,2}, x_{j,1}, x_{j,0}) \) if \( s = 4 \), and by \( TK_j^{(3)} = (x_{j,7}, \ldots, x_{j,1}, x_{j,0}) \) if \( s = 8 \), respectively. Then,

LFSRupdate\(^{(3)}\): \( TK^{(3)} \mapsto \widetilde{TK}^{(3)} \)

where, for each \( j \in \{1, 2, 5, 6, 9, 10, 13, 14\} \), \( \widetilde{TK}_j^{(3)} = (x_{j,0} + x_{j,3}, x_{j,3}, x_{j,2}, x_{j,1}) \),

and, for each other index \( j \), \( \widetilde{TK}_j^{(3)} = TK_j^{(3)} \).

If \( s = 8 \), we have

LFSRupdate\(^{(3)}\): \( TK^{(3)} \mapsto \widetilde{TK}^{(3)} \)

where, for each \( j \in \{1, 2, 5, 6, 9, 10, 13, 14\} \),

\( \widetilde{TK}_j^{(3)} = (x_{j,0} + x_{j,6}, x_{j,7}, x_{j,6}, x_{j,5}, x_{j,4}, x_{j,3}, x_{j,2}, x_{j,1}) \),

and, for each other index \( j \), \( \widetilde{TK}_j^{(3)} = TK_j^{(3)} \).
The Round Constants $a_r$ and $b_r$, that are added to the round keys, are given in Table 5.4. For a lightweight implementation, those round constants can be generated by by a 6-bit affine LFSR as follows. Denote the LFSR state by $(x_5, x_4, x_3, x_2, x_1, x_0)$. In the first place, it is initialized by $(0, 0, 0, 0, 0, 0)$ and updated in every round according to

$$(x_5, x_4, x_3, x_2, x_1, x_0) \mapsto (x_4, x_3, x_2, x_1, x_0, x_5 + x_4 + 1).$$

Then, in every round $r$, the round constants are taken as $a_r = (x_3, x_2, x_1, x_0)$ and $b_r = (0, 0, x_5, x_4)$ if $s = 4$. If $s = 8$, the constants are padded by zeros in the most significant bits, i.e., $a_r = (0, 0, 0, 0, x_3, x_2, x_1, x_0)$ and $b_r = (0, 0, 0, 0, 0, x_5, x_4)$.

View as an AES-like Cipher

Although the round keys are added in between the rounds, it is important to note that the definition of the round $R_i$ can be re-written as

$$(x, k_i) \mapsto \text{Add}_{\text{Mix}_M \circ \text{Permute}_p(k_i)} \circ \text{Mix}_M \circ \text{Permute}_p \circ S_{\text{Sbb}}(x)$$

and thus, all Skinny versions can be defined in the notion of a key-alternating AES-like cipher, as explained in Section 2.4.2. Then, their unkeyed round functions follow the structure of an AES-like round $R_{\text{Sbb}, p, M}$ (Definition 2.9).
**Algorithm 5.2** The Key Schedule of Skinny

procedure KeySchedule(TK\(^{(1)}\))
for \(r = 1, \ldots, t\) do
\[
k_r \leftarrow \begin{bmatrix}
TK_1^{(1)} & TK_5^{(1)} & TK_9^{(1)} & TK_{13}^{(1)} \\
TK_2^{(1)} & TK_6^{(1)} & TK_{10}^{(1)} & TK_{14}^{(1)}
\end{bmatrix}
+ \begin{bmatrix}
a_r & 0 & 0 & 0 \\
b_r & 0 & 0 & 0 \\
2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]
\(TK^{(1)} \leftarrow \text{Permute}_{p_r}(TK^{(1)})\)
end for
end procedure

procedure KeySchedule(TK\(^{(1)}\), TK\(^{(2)}\))
for \(r = 1, \ldots, t\) do
\[
k_r \leftarrow \begin{bmatrix}
TK_1^{(1)} & TK_5^{(1)} & TK_9^{(1)} & TK_{13}^{(1)} \\
TK_2^{(1)} & TK_6^{(1)} & TK_{10}^{(1)} & TK_{14}^{(1)}
\end{bmatrix}
+ \begin{bmatrix}
TK_1^{(2)} & TK_5^{(2)} & TK_9^{(2)} & TK_{13}^{(2)} \\
TK_2^{(2)} & TK_6^{(2)} & TK_{10}^{(2)} & TK_{14}^{(2)}
\end{bmatrix}
\]
\(TK^{(1)} \leftarrow \text{Permute}_{p_r}(TK^{(1)})\)
\(TK^{(2)} \leftarrow \text{Permute}_{p_r}(TK^{(2)})\)
\(TK^{(2)} \leftarrow \text{LFSRUpdate}_s(TK^{(2)})\)
end for
end procedure
procedure KeySchedule($TK^{(1)}, TK^{(2)}, TK^{(3)}$)
    for $r = 1, \ldots, t$
        
        $k_r \leftarrow \begin{bmatrix}
        TK^{(1)}_1 & TK^{(1)}_5 & TK^{(1)}_9 & TK^{(1)}_{13} \\
        TK^{(1)}_2 & TK^{(1)}_6 & TK^{(1)}_{10} & TK^{(1)}_{14} \\
        0 & 0 & 0 & 0 \\
        0 & 0 & 0 & 0 \\
        \end{bmatrix} + \begin{bmatrix}
        TK^{(1)}_1 & TK^{(1)}_5 & TK^{(1)}_9 & TK^{(1)}_{13} \\
        TK^{(1)}_2 & TK^{(1)}_6 & TK^{(1)}_{10} & TK^{(1)}_{14} \\
        0 & 0 & 0 & 0 \\
        0 & 0 & 0 & 0 \\
        \end{bmatrix} + \begin{bmatrix}
        TK^{(2)}_1 & TK^{(2)}_5 & TK^{(2)}_9 & TK^{(2)}_{13} \\
        TK^{(2)}_2 & TK^{(2)}_6 & TK^{(2)}_{10} & TK^{(2)}_{14} \\
        a_r & 0 & 0 & 0 \\
        b_r & 0 & 0 & 0 \\
        2 & 0 & 0 & 0 \\
        0 & 0 & 0 & 0 \\
        \end{bmatrix} + \begin{bmatrix}
        TK^{(3)}_1 & TK^{(3)}_5 & TK^{(3)}_9 & TK^{(3)}_{13} \\
        TK^{(3)}_2 & TK^{(3)}_6 & TK^{(3)}_{10} & TK^{(3)}_{14} \\
        2 & 0 & 0 & 0 \\
        0 & 0 & 0 & 0 \\
        \end{bmatrix}$

        $TK^{(1)} \leftarrow $ Permute$_{pT}(TK^{(1)})$
        $TK^{(2)} \leftarrow $ Permute$_{pT}(TK^{(2)})$
        $TK^{(3)} \leftarrow $ Permute$_{pT}(TK^{(3)})$
        $TK^{(2)} \leftarrow $ LFSRupdate$_s^{(2)}(TK^{(2)})$
        $TK^{(3)} \leftarrow $ LFSRupdate$_s^{(3)}(TK^{(3)})$
    end for
end procedure

5.3.2 Design Rationale

In order to understand the motivation behind the particular construction of Skinny, we briefly recall some of the design considerations; both with regard to security and efficiency. For more details, we refer to [BJK + 16b, Section 3].

Several design choices are inspired by already existing constructions, e.g., the TWEAKEY framework and the building blocks of AES-like ciphers. However, all of the components employed in Skinny are optimized for lightweight purposes. Although the design performs well in most lightweight scenarios, a special focus was on minimizing the area required for round-based hardware implementations. The main competitor with regard to this metric is Simon. Indeed, the round-based ASIC implementations of Skinny-64-128, resp. Skinny-128-256, obtained in [BJK + 16b] slightly outperform the implementations of Simon-64-128, resp. Simon-128-256, given in [BSS + 15] with regard to area requirements. In addition, Skinny-128-128 performs very well compared to its Simon equivalent (see Figure 5.5 for details).\(^5\)

While all the components employed in Skinny are optimized for reducing area, an important criteria was that the round function itself still preserves a significant amount of cryptographic strength. Otherwise, the total number of rounds to

\(^5\)In [BJK + 16b], results for several other implementations, i.e., unrolled implementations, serial implementations, as well as FPGA and software implementations, are given. Also, a threshold implementation for protection against side-channel attacks was realized.
Table 5.5: Round-based ASIC implementations of the Skinny-64 and the Skinny-128 versions and comparison to Simon. (Cell library: UMC L180 0.18 µm for Skinny and IBM 8RF 130 nm for Simon)

<table>
<thead>
<tr>
<th></th>
<th>Area (GE)</th>
<th>Delay (ns)</th>
<th>Clock Cycles</th>
<th>Throughput @100KHz</th>
<th>Throughput @maximum</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skinny-64-64</td>
<td>1223</td>
<td>1.77</td>
<td>32</td>
<td>200.00</td>
<td>1130.00</td>
<td>[BJK+16b]</td>
</tr>
<tr>
<td>Skinny-64-128</td>
<td>1696</td>
<td>1.87</td>
<td>36</td>
<td>177.78</td>
<td>951.11</td>
<td>[BJK+16b]</td>
</tr>
<tr>
<td>Skinny-64-192</td>
<td>2183</td>
<td>2.02</td>
<td>40</td>
<td>160.00</td>
<td>792.00</td>
<td>[BJK+16b]</td>
</tr>
<tr>
<td>Skinny-128-128</td>
<td>2391</td>
<td>2.89</td>
<td>40</td>
<td>320.00</td>
<td>1107.20</td>
<td>[BJK+16b]</td>
</tr>
<tr>
<td>Skinny-128-256</td>
<td>3312</td>
<td>2.89</td>
<td>48</td>
<td>266.67</td>
<td>922.67</td>
<td>[BJK+16b]</td>
</tr>
<tr>
<td>Skinny-128-384</td>
<td>4268</td>
<td>2.89</td>
<td>56</td>
<td>228.57</td>
<td>790.86</td>
<td>[BJK+16b]</td>
</tr>
<tr>
<td>Simon-64-128</td>
<td>1751</td>
<td>1.60</td>
<td>46</td>
<td>145.45</td>
<td>870.00</td>
<td>[BSS+15]</td>
</tr>
<tr>
<td>Simon-128-128</td>
<td>2342</td>
<td>1.60</td>
<td>70</td>
<td>188.24</td>
<td>1145.00</td>
<td>[BSS+15]</td>
</tr>
<tr>
<td>Simon-128-256</td>
<td>3419</td>
<td>1.60</td>
<td>74</td>
<td>177.78</td>
<td>1081.00</td>
<td>[BSS+15]</td>
</tr>
</tbody>
</table>

be applied has to become extremely large in order to guarantee security of the cipher using standard arguments. In other words, the task was to find the exact spot for which we get optimal performance along with strong security arguments (in our design measured by the minimum number of active S-boxes). This goal was aimed to achieve by an iterative design process. In a nutshell, the intuition behind our final design choice is that removing any operation from the cipher would lead to much weaker design that significantly worsens the trade off between implementation efficiency and throughput (i.e., the number of rounds that have to be applied).

The General Structure

We chose to design Skinny as an SP cipher since there are well-known solutions for evaluating the security, i.e., by computing the minimum number of active S-boxes. Moreover, such a structure allows for designing the $n = 64$ and $n = 128$ version of the cipher in almost the same way by only changing the word length $s$ and using a different S-box. Further, fixing a particular version Skinny-$n$-$\kappa$, we wanted to keep the definition of all the rounds exactly the same. For instance, this means that no whitening key is applied, or that we avoid the usage of a different linear layer in the last round (as opposed to the AES, where the last MixColumns operation is omitted). This simplifies the overall description of the cipher and reduces its implementation overhead. However, this design decision implies that parts of the first and the last round do not contribute to the security of the cipher as the
adversary can simply invert all operations until the first (resp. last) key addition.

We further chose the STK construction of the TWEAKEY framework for introducing the tweakey input. This allows to design Skinny as a flexible tweakable block cipher with the benefit of obtaining security arguments against related-key differential attacks, using automatic tools like MILP. We explain the MILP model for computing the minimum number of active S-boxes in the TK1 adversary model in detail in Section 5.3.3. For a detailed description of the MILP approach for computing bounds in TK2 and TK3, we refer to [BJK+16].

SubCells

For the choice of the S-boxes $S_b^4$ and $S_b^8$ to be applied in the non-linear layer, we required that $p_{S_b^4}, p_{S_b^8} \leq 2^{-2}$ and $c_{S_b^4}, c_{S_b^8} \leq 2^{-1}$. An automatic search was conducted in order to find an S-box construction with a limited implementation cost and meeting the above requirements on the cryptographic properties. The final choice for the S-box $S_b^4$ has an algebraic degree of three, the final choice of $S_b^8$ has an algebraic degree of six. The two choices both meet the required bounds on the maximum differential probability and maximum absolute correlation by equality.

The Round Constants

The purpose of round constants is to differentiate the particular rounds and avoid symmetries. For instance, a bad choice of round constants may cause weaknesses with regard to invariant attacks. In Chapter 6, we study these attacks in more detail and actually show that Skinny-64-64 can be proven secure against a large class of invariant attacks. For avoiding to directly store the particular round constants as a look-up table, a lightweight LFSR is used for their generation.

The Tweakey Update Function and the Round Key Addition

In contrast to the STK construction as originally proposed in [JNP14], the words of the tweakey states are updated by affine LFSRs instead by applying finite field multiplications, mainly for reducing hardware area. But as the most important difference, the round keys are only extracted from the first two rows of the tweakey states, i.e., the third and fourth row of each round key array (except of the round constant 2 in the word in the first column of the third row) consist of only zeros. This saves the implementation of the XOR operations for adding the last two rows of the round key arrays. As only half of the words of each tweakey state are extracted in every round, only those halves are updates by the LFSR operations. Special care has been taken by choosing the actual LFSRs in order to guarantee a high number of rounds before cancellations of differences in TK2 and TK3 can happen. In particular, and due to the fact that only half of the state is affected by the LFSRs in every round, for a fixed word index, a single cancellation can only
happen every 30 rounds for TK2 and two cancellations can only happen every 30 rounds for TK3.

The tweakey permutation $p_T$ has been chosen to maximize the bounds on the number of active S-boxes in the related-key model (in the SK model, it has no impact). Additionally, we have enforced the special property of $p_T$ that all words in the lower half of the state are permuted to the upper half and vice versa. Since only the first and the second row of the tweakey states are added to the cipher’s internal state, this ensures that all words of the tweakey states will be added (almost) equally often to the cipher’s internal state. On top of that, we only considered those variants of $p_T$ that leave the words in the upper half of the tweakey state at the same relative position in the lower half and that consist of a single cycle.

**The Linear Layer**

In order to compete with the performance of Simon with regard to hardware implementations, we extremely tailored the AES-like linear layer used in Skinny and chose a very sparse matrix $M$ for the $\text{Mix}_M$ operation. In particular, $M$ only consists of the coefficients $0, 1 \in \mathbb{F}_{2^s}$. It is sparser than the matrix employed in Midori and has a differential and linear branch number with respect to $s$-bit words of only two. This may look suspicious at a first glance and one certainly has to consider the existence of differential and linear trails with only a single active S-box per round. However, we designed $M$ in a way that whenever a branching transition with only two active S-boxes occurs, the next transition will likely lead to a much higher number of active S-boxes. For instance, when looking at the definition of $M$, the only way to get a branching transition with two active S-boxes is to have a non-zero input difference in either the second or the fourth component. But this necessarily leads to a non-zero input difference in the first or third component in the next round, which then diffuses to many output positions. A (multiple-round) differential trail with a single active S-box per round is therefore not possible. Actually, one can prove at least 96 active S-boxes over 20 rounds using the MILP tool (see Table 5.6 for the actual bounds). Similar observations can be made for linear trails, i.e., by considering $M^{-1}$.

We have considered all possibilities for $M$ that can be implemented with at most three XOR operations and kept those matrices that, in combination with $\text{Permute}_p$, guaranteed fast diffusion and led to strong bounds on the minimum number of active S-boxes in the SK model. Here, by full diffusion we refer to the number of rounds needed such that every bit of the internal state depends on every input bit. In all Skinny versions, six rounds are needed (both in the forward direction and for the inverse cipher) to guarantee full diffusion. Section 5.3.4 outlines how that diffusion was evaluated.

As a last criterion on $M$, we required that the round key input affects the whole internal state of the cipher as fast as possible. This is in particular crucial as only half of the state is added with non-zero key material in every round. Our
Table 5.6: Lower bounds on the minimum number of active S-boxes in Skinny. The numbers for SK Lin correspond to the active S-boxes in linear trails according to $M^{T^{-1}}$. In cases where solving the particular MILP instance did not finish in time, upper bounds are given between parentheses. $64$ active S-boxes are required in order to avoid differential and linear distinguishers based on a single trail (it is $p^{64}_{Sb} = 2^{-128} \leq 2^{-n}$ and $e^{64}_{Sb} = 2^{-128} \leq 2^{-n}$).

<table>
<thead>
<tr>
<th>Model</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>SK</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>26</td>
<td>36</td>
<td>41</td>
<td>46</td>
<td>51</td>
<td>55</td>
<td>58</td>
<td>61</td>
<td>66</td>
</tr>
<tr>
<td>TK1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>13</td>
<td>16</td>
<td>23</td>
<td>32</td>
<td>38</td>
<td>41</td>
<td>45</td>
<td>49</td>
</tr>
<tr>
<td>TK2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>16</td>
<td>21</td>
<td>25</td>
<td>31</td>
<td>35</td>
</tr>
<tr>
<td>TK3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>13</td>
<td>16</td>
<td>19</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>SK Lin</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>13</td>
<td>19</td>
<td>25</td>
<td>32</td>
<td>38</td>
<td>43</td>
<td>48</td>
<td>52</td>
<td>55</td>
<td>58</td>
<td>64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>SK</td>
<td>75</td>
<td>82</td>
<td>88</td>
<td>92</td>
<td>96</td>
<td>102</td>
<td>108</td>
<td>114</td>
<td>116</td>
<td>124</td>
<td>125</td>
<td>128</td>
<td>130</td>
<td>132</td>
<td>138</td>
</tr>
<tr>
<td>TK1</td>
<td>54</td>
<td>59</td>
<td>62</td>
<td>66</td>
<td>70</td>
<td>75</td>
<td>79</td>
<td>83</td>
<td>85</td>
<td>88</td>
<td>95</td>
<td>102</td>
<td>108</td>
<td>112</td>
<td>120</td>
</tr>
<tr>
<td>TK2</td>
<td>40</td>
<td>43</td>
<td>47</td>
<td>52</td>
<td>57</td>
<td>59</td>
<td>64</td>
<td>67</td>
<td>72</td>
<td>75</td>
<td>82</td>
<td>85</td>
<td>88</td>
<td>92</td>
<td>96</td>
</tr>
<tr>
<td>TK3</td>
<td>27</td>
<td>31</td>
<td>35</td>
<td>43</td>
<td>45</td>
<td>48</td>
<td>51</td>
<td>55</td>
<td>58</td>
<td>60</td>
<td>65</td>
<td>72</td>
<td>77</td>
<td>81</td>
<td>85</td>
</tr>
<tr>
<td>SK Lin</td>
<td>70</td>
<td>76</td>
<td>80</td>
<td>85</td>
<td>90</td>
<td>96</td>
<td>102</td>
<td>107</td>
<td>(110)</td>
<td>(110)</td>
<td>(118)</td>
<td>(112)</td>
<td>(122)</td>
<td>(128)</td>
<td>(136)</td>
</tr>
</tbody>
</table>

The final choice of $M$ is optimal with respect to full key diffusion in the sense that only a single round is required (for both the case of encryption and decryption) to ensure full round key diffusion over the internal state.

5.3.3 The MILP Model for Computing Active S-Boxes

We used the MILP approach based on the framework of Mouha et al. [MWGP12] for computing the minimum number of active S-boxes in SK (with regard to differential and linear trails) and TK1 (with regard to differential trails). In our MILP model, we need the following decision variables that can take values in $\{0,1\} \subseteq \mathbb{Z}$.

- $\bar{x}_{i,j,r}$, with $i,j \in \mathbb{N}_{<4}, r \in \mathbb{N}_{<t+1}$, for indicating the activity pattern at the S-box inputs. In particular, we are going to define the constraints such that $\bar{x}_{i,j,r} = 1$ if and only if the S-box at position $i + 4j + 1$ of the input state in round $r$ is active (here, the round index is starting from 0). The objective function that we want to minimize will then be defined as

$$\sum_{i,j\in\mathbb{N}_{<4}} \sum_{r\in\mathbb{N}_{<t}} \bar{x}_{i,j,r} .$$

- $\bar{y}_{i,j,r}$, with $i,j \in \mathbb{N}_{<4}, r \in \mathbb{N}_{<t}$, for indicating the activity pattern (with respect to $s$-bit words) of the internal state after the addition of the round keys.

113
• \( \xi_{i,j} \), with \( i, j \in \mathbb{N}_{\leq 4}^0 \), for indicating the activity pattern (with respect to \( s \)-bit words) of the initial tweakey state \( TK^{(1)} \).

• For modelling the branching transitions, we need two sets of auxiliary variables, i.e., \( d_{i,j,r}^{\oplus} \), with \( i \in \mathbb{N}_{\leq 2}^0, j \in \mathbb{N}_{<4}^0, r \in \mathbb{N}_{<t}^0 \), for modelling the round key addition layer (only needed for \( TK_1 \)) and \( d_{j,r}, d'_{j,r}, d''_{j,r} \), with \( j \in \mathbb{N}_{<4}^0, r \in \mathbb{N}_{<t}^0 \), for the MixM layer.

As the round key addition and the MixM layer only consist of word-wise XOR operations, the main building blocks of the model are the linear constraints for defining the branching transitions over those word-wise XOR operations. For shorter notations, we define the following sets.

Constraints for XOR. Let \( i_1, i_2, o, d \) be decision variables that can take values in \( \{0, 1\} \subseteq \mathbb{Z} \). We denote by \( C^{\oplus}[i_1, i_2, o, d] \) the set of linear constraints

\[
\{i_1 \leq d\} \cup \{i_2 \leq d\} \cup \{o \leq d\} \cup \{i_1 + i_2 + o \geq 2d\}.
\]

All of the constraints in \( C^{\oplus}[i_1, i_2, o, d] \) are fulfilled if and only if

\[
(i_1, i_2, o, d) \in \{(0,0,0,0), (0,1,1,1), (1,0,1,1), (1,1,0,1), (1,1,1,1)\}.
\]

The crucial observation is that, if \( x_1 \) is an \( s \)-bit word with activity \( i_1 \) and \( x_2 \) is an \( s \)-bit word with activity \( i_2 \), the activity \( o \) of the word \( x_1 + x_2 \) fulfills

(i) \( o = 0 \) if both \( i_1 = i_2 = 0 \),

(ii) \( o = 1 \) if \( (i_1, i_2) \in \{(0,1), (1,0)\} \), and

(iii) \( o \in \{0, 1\} \) if \( i_1 = i_2 = 1 \).

Those properties are modelled by the above linear constraints. The auxiliary variable \( d \) just indicates whether at least one input \( x_1 \) or \( x_2 \) is active.

Constraints for the Linear Mixing. Let \( i_1, \ldots, i_4, o_1, \ldots, o_4, d_1, d_2, d_3 \) be decision variables that can take values in \( \{0, 1\} \subseteq \mathbb{Z} \).

By \( C_M[i_1, i_2, i_3, o_1, o_2, o_3, o_4, d_1, d_2, d_3] \) we denote the set of linear constraints

\[
C^{\oplus}[i_1, i_3, o_4, d_1] \cup C^{\oplus}[o_1, i_4, o_1, d_2] \cup C^{\oplus}[i_2, i_3, o_3, d_3] \cup \{o_2 = i_1\}.
\]

Those constraints then model the (word-wise) differential branching transitions of the transformation

\[
\begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  x_4
\end{pmatrix}
\rightarrow
\begin{pmatrix}
  1 & 0 & 1 & 1 \\
  1 & 0 & 0 & 0 \\
  0 & 1 & 1 & 0 \\
  1 & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  x_4
\end{pmatrix}
=
\begin{pmatrix}
  x_1 + x_3 + x_4 \\
  x_1 \\
  x_2 + x_3 \\
  x_1 + x_3
\end{pmatrix}.
\]
where the \( x_i \) are \( s \)-bit words.\(^6\)

One now obtains a lower bound on the minimum number of active S-boxes over \( r \) rounds in \( \text{SK} \) by solving the following MILP instance:

Minimize

\[
\sum_{i,j \in \mathbb{N}_0^4} \sum_{r \in \mathbb{N}_0^t} \bar{x}_{i,j,r}
\]

Subject to:

1. Excluding the trivial solution with zero active S-boxes

\[
\{ \sum_{i,j \in \mathbb{N}_0^4} \bar{x}_{i,j,0} \geq 1 \}
\]

2. Application of the linear layer

\[
\bigcup_{r \in \mathbb{N}_0^t} \bigcup_{j \in \mathbb{N}_0^4} C_M[\bar{x}_{0,j,r}, \bar{x}_{1,j,0} \mod 4, r, \bar{x}_{2,j,-2} \mod 4, r, \bar{x}_{3,j,3} \mod 4, r, \bar{x}_{0,j,r+1}, \bar{x}_{1,j,r+1}, \bar{x}_{2,j,r+1}, \bar{x}_{3,j,r+1}, d_{j,r}, d'_{j,r}, d''_{j,r}]
\]

For \( \text{TK1} \), we have to optimize the following MILP model:

Minimize

\[
\sum_{i,j \in \mathbb{N}_0^4} \sum_{r \in \mathbb{N}_0^t} \bar{x}_{i,j,r}
\]

Subject to

1. Excluding the trivial solution

\[
\{ \sum_{i,j \in \mathbb{N}_0^4} \bar{x}_{i,j,0} + \xi_{i,j} \geq 1 \}
\]

2. Application of the tweakey addition to half of the state – Here, the tweakey state permutation \( p_T \) is denoted as a permutation on the indices \( (i,j) \in \mathbb{N}_0^4 \times \mathbb{N}_0^4 \)

\[
\bigcup_{r \in \mathbb{N}_0^t} \bigcup_{j \in \mathbb{N}_0^4} C_M[\bar{x}_{i,j,r}, \bar{\xi}_{i,j,r}, \bar{y}_{i,j,r}, d_{i,j,r}, d'_{i,j,r}, d''_{i,j,r}]
\]

3. Application of the linear layer

\[
\bigcup_{r \in \mathbb{N}_0^t} \bigcup_{j \in \mathbb{N}_0^4} C_M[\bar{y}_{0,j,r}, \bar{y}_{1,j,0} \mod 4, r, \bar{y}_{2,j,-2} \mod 4, r, \bar{y}_{3,j,3} \mod 4, r, \bar{x}_{0,j,r+1}, \bar{x}_{1,j,r+1}, \bar{x}_{2,j,r+1}, \bar{x}_{3,j,r+1}, d_{j,r}, d'_{j,r}, d''_{j,r}]
\]

For all of the number of rounds \( r \) that are given in Table 5.6, the according MILP instances were solved using Gurobi [GO16].

\(^6\)For computing bounds on the minimum number of active S-boxes in linear trails in the single-key model, one has to model the branching transitions of \( M^{-1} \). For that, one employs another auxiliary variable \( d \) and defines \( C_M^{-1}[i_1, i_2, i_3, i_4, o_1, o_2, o_3, o_4, d, d_1, d_2, d_3] \) as the set of constraints \( C_M[i_2, i_3, d, d_1] \cup C_M[i_1, d, o_2, d_2] \cup C_M[i_4, d, o_4, d_3] \cup \{ o_1 = i_4 \} \cup \{ o_3 = i_2 \} \).
On The Tightness of the MILP Bounds

The solution of the minimization problems defined above determines a lower bound on the number of active S-boxes in any (non-trivial) \( t \)-round trail in the SK, resp. TK1 case. If we consider the word-wise application of the S-box as a black box, then all of the computed bounds for SK are tight in the sense that one can construct a valid differential trail for a specific choice of S-boxes. In other words, the bound is tight if the S-box can be chosen independently for every word and every round. This is less clear in the related-key scenario and therefore, we only claim lower bounds. The actual minimum number of active S-boxes might be even higher.

5.3.4 Diffusion Test

When we analyze the diffusion properties in a cipher, we evaluate the minimum number of rounds \( r \) such that every bit of the internal state after the application of \( r \) rounds depends on every input bit. For an SP cipher like Skinny, the diffusion properties depend both on the linear layer and on the S-box. To formally define what full diffusion means and to evaluate the diffusion properties in Skinny with block length \( n = 16s \) in particular, we first define the diffusion matrix \( D_s \) as described in the following.

Since the linear layer in Skinny consists of a word permutation and a Mix\(_M\) operation with a binary \( 4 \times 4 \) matrix \( M \), one can represent the linear layer as a binary matrix \( L \in GL_{16}(\mathbb{F}_2^s) \). In particular,

\[
L = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

Furthermore, for an S-box \( S_b \), we define the dependency matrix \( \text{Dep}(S_b) \in \text{Mat}_s(\mathbb{Z}) \) by

\[
\text{Dep}(S_b)_{i,j} = \begin{cases} 
1 & \text{if } \exists x: S_b_i(x) \neq S_b_i(x + e_j) \\
0 & \text{else}
\end{cases}
\]
Thereby, $S_{bi}$ denotes the $i$-th coordinate function and $e_j$ the $j$-th unit vector. For the Skinny S-boxes $S_{b4}$ and $S_{b8}$, we have

$$\text{Dep}(S_{b4}) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}, \quad \text{Dep}(S_{b8}) = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$  

Now, we can define the Skinny diffusion matrix $D_s$ for $s \in \{4, 8\}$ as a block matrix $D_s \in \text{Mat}_{16}(\text{Mat}_s(\mathbb{Z}))$ by

$$D_{s,i,j} = \begin{cases} \text{Dep}(S_{bs}) & \text{if } L_{i,j} = 1 \\ 0_s & \text{if } L_{i,j} = 0 \end{cases}.$$  

One can now define full diffusion as follows.

**Definition 5.6.** The cipher achieves full diffusion after $r$ rounds, if $D_s$ contains no zero coefficient when $D_s$ is interpreted as a $16s \times 16s$ matrix over the integers.

For the Skinny-64 and Skinny-128 versions, we made sure that full diffusion is achieved after 6 rounds, both in forward direction and for the inverse. Note that the diffusion matrix of the inverse has to be computed separately.

### 5.3.5 Security Claim and Best Cryptanalysis so far

The security claim for Skinny is resistance against related-key/related-tweak attacks. The particular design according to the TWEAKEY framework, in particular the possibility of dedicating some key material to a public tweak input, should allow to use Skinny in scenarios where both related-key and related-tweak security is needed. We emphasize that, in cases where related-key security is not needed, one could also use all the tweakey input as a secret key and then XOR a public tweak to the key. From Theorem 5.1, one would directly obtain provable chosen-tweak security, but at the price of sacrificing related-key security.

In the design paper, various cryptanalysis on Skinny was already conducted by the authors. After the publication of the design, several third-party cryptanalysis followed, see [SMB16, ABC+17, LGST17, TAY17] for a selection. In Table 5.7 for all Skinny versions, we give the maximum number of rounds for which the round-reduced cipher can be broken by the best published attacks so far. Here, the term “broken” refers to a key-recovery attack (in the related-tweakey model) with a time complexity below $2^\kappa$ encryption operations and a data complexity below $2^n$. Those best attacks are based on impossible differential [Knu98, BBS05] and rectangle [BDK01] distinguishers. These results indicate that there is still a huge security margin left.

---

\(^7\)We exclude attacks that are based on accelerated brute force.
Table 5.7: Number of rounds of Skinny that are broken by the best key-recovery attacks so far, published in [LGS17]. All those key-recovery attacks make use of a distinguisher over a smaller number of rounds. The red percentage values show the ratio between number of rounds broken and the total number of rounds $t$.

<table>
<thead>
<tr>
<th>Block length $n$</th>
<th>Tweakey length $\kappa$</th>
<th>$n$</th>
<th>$2n$</th>
<th>$3n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td></td>
<td>19/32</td>
<td>23/36</td>
<td>27/40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>59.4%</td>
<td>63.9%</td>
<td>67.5%</td>
</tr>
<tr>
<td>128</td>
<td></td>
<td>19/40</td>
<td>23/48</td>
<td>27/56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>47.5%</td>
<td>47.9%</td>
<td>48.2%</td>
</tr>
</tbody>
</table>

5.4 The Mantis Family of Low-Latency Tweakable Block Ciphers

In this section, we give a tweakable block cipher design which is optimized for low-latency implementations.\footnote{We acknowledge the contribution of Roberto Avanzi to the design of Mantis. He first suggested us to combine Prince with the TWEAKEY framework, and also to modify the latter by permuting the tweak independently from the key, in order to save on the field multiplications of the tweak words. He then brainstormed with us on early versions of the design.}

The existing low-latency block cipher Prince [BCG+12] already provides a very good starting point for a low-latency design. Its round function basically follows the AES-like structure, employing a Mix$_M$ operation of branch number four. The main difference between the overall structure of Prince and AES (and actually all other key-alternating ciphers we have already considered) is that the design is symmetric around a linear layer in the middle. This allows to realize what was defined as $\alpha$-reflection, i.e., the decryption $E_k^{-1}$ under a key $k$ basically corresponds to encryption with a related key $k + \alpha$, where $\alpha$ is a fixed constant. A natural way of turning a Prince-like design into a tweakable block cipher is to define a tweak schedule and evaluate the number of rounds until the minimum number of active S-boxes (in the related-tweak model) is high enough.

However, the problem is that the latency of a cipher is directly related to the number of rounds. Thus, it is crucial to find a design that ensures security already with a low number of rounds. Here, components of the block cipher Midori turn out to be beneficial. As outlined already before, one of the key observations in Midori was that deviating from the ShiftRows operation used in the AES allows to significantly improve upon the number of active S-boxes (in the single key model) if a Mix$_M$ layer with a branch number of only four is used. Moreover, the designers of Midori designed a 4-bit S-box that was optimized with respect to circuit-depth. This directly leads to an improved version of Prince itself: Simply
replace the round function by the function of Midori while keeping the entire design symmetric around the middle in order to preserve the $\alpha$-reflection property. This simple change would result in a cipher with improved latency and improved security (measured by number of active S-boxes) compared to Prince. It is actually exactly this Prince-like Midori that we use as a starting point for designing the low-latency block cipher Mantis. The final step in the design of Mantis was to design a suitable tweak-scheduling algorithm that would guarantee a high number of active S-boxes in the setting where the attacker can control the difference in the tweak. Using again the MILP approach (see Section 5.4.5), we are able to demonstrate that only a slight increase in the number of rounds compared to Prince is already sufficient to get confidence in the resistance against differential attacks in the related-tweak model. It is important to note that we now make a distinction between related-key and related-tweak attacks. In the former, the adversary is allowed to insert differences in the key input, while in the latter case, the adversary is only allowed to insert differences in the tweak input. We have to make this distinction as Mantis is certainly not secure in the related-key setting because of the $\alpha$-reflection property.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{mantis.png}
\caption{Illustration of Mantis$_t$. Here, $P_\sigma$ is a short notation for Permute$_\sigma$, $M$ a short notation for Mix$_M$, and $\xi := \xi + \alpha$.}
\end{figure}

\subsection*{5.4.1 Specification}

Mantis is a family of tweakable block ciphers that comes in different versions, i.e.,

\[ \text{Mantis}_t: \mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^\tau \rightarrow \mathbb{F}_2^n \]

with a block length of $n = 64$, a key length of $\kappa = 128$, and a tweak length of $\tau = 64$. The only difference in the versions, parametrized by a natural number $t$, is the number of rounds. In particular, as Mantis$_t$ is defined as a reflection cipher, the parameter $t$ specifies the number of rounds of one half of the cipher. The overall design structure is illustrated in Figure 5.1. Compared to Skinny, we have to distinguish tweak and key input. The reason is that, because of the design structure as a reflection cipher, Mantis cannot resist related-key attacks, but only related-tweak attacks. By related-tweak (differential) attacks, we refer to the model in which the adversary is able to insert differences in the plaintext and in the tweak input, but not in the key input.

Similar to the block cipher Prince, the tweakable cipher Mantis$_t$ is based on the FX-construction \cite{KR96a} and thus applies whitening keys before and after
applying its core components. For that, the 128-bit initial key \(k\) is first split into 
\[ k = \zeta || \xi \], where \(\zeta, \xi \in \mathbb{F}_2^{64}\). Then, \(k\) is extended to the 192 bit key 
\[ (\zeta, \zeta', \xi) = (\zeta, (\zeta \gg 1) + (\zeta \gg 63), \xi) \],
and \(\zeta, \zeta'\) are used as whitening keys in an FX-construction. The subkey \(\xi\) is 
used as the round key for all of the \(2t\) rounds of \texttt{Mantis}_t. We decided to stick 
with the FX-construction for simplicity, even though other options as described in 
[BCKL17].

### Initialization

In all versions of \texttt{Mantis}, the cipher’s internal states, the states of the keys, and 
the tweak state are represented by a \(4 \times 4\) array of words in \(\mathbb{F}_2^4\), respectively. We 
denote an internal state (resp. key or tweak state) \(x \in \mathbb{F}_2^{4 \times 16}\) as

\[
\begin{bmatrix}
x_1 & x_5 & x_9 & x_{13} \\
x_2 & x_6 & x_{10} & x_{14} \\
x_3 & x_7 & x_{11} & x_{15} \\
x_4 & x_8 & x_{12} & x_{16}
\end{bmatrix}, \quad x_i \in \mathbb{F}_2^4.
\]

A plaintext \(m = m_1 || \ldots || m_{16} \in \mathbb{F}_2^{4 \times 16}\) is mapped to the cipher’s initial state in 
row-wise manner. Similarly, the initial tweak input \(h = h_1 || \ldots || h_{16} \in \mathbb{F}_2^{4 \times 16}\) and all of the key input \(\zeta = \zeta_1 || \ldots || \zeta_{16} \in \mathbb{F}_2^{4 \times 16}, \zeta' = \zeta'_1 || \ldots || \zeta'_{16} \in \mathbb{F}_2^{4 \times 16}\), and, 
\(\xi = \xi_1 || \ldots || \xi_{16} \in \mathbb{F}_2^{4 \times 16}\), are loaded row wise to the initial tweak state, resp. key 
states.

### The Round Functions

A keyed instance of the round \(R_i\) (i.e., \(R_{ik_i} = R_i(\cdot, k_i)\)) in \texttt{Mantis}_t operates on the 
cipher’s internal state as

\[ \text{Mix}_M \circ \text{Permute}_p \circ \text{Add}_{k_i} \circ \text{Add}_{c_i} \circ S_{\text{SubCells}}. \]

where the constant \(c_i\) depends on the round index \(i\). In the following, we describe 
all those components of the rounds.

\(S_{\text{SubCells}}\). The involutory S-box \(S_{\text{Mid64}}\) as given in Table 2.2 is applied 
to every word of the internal state. Using the Midori S-box is beneficial as it 
is especially optimized for small area and low circuit depth.

\(\text{Add}_{c_i}\) (AddConstant\(_i\)). In round \(R_i\), the \(i\)-th round constant \(c_i\), as defined below, 
is added to the internal state. For proving that the round constants are not 
chosen with intentional weaknesses, they are defined in a similar way as for \texttt{Prince}, i.e., we use the first fractional digits of the base-16 representation of 
the irrational number \(\pi\) to generate those constants (actually the very first
digits correspond to \( \alpha \) defined below). Note that, in contrast to Prince, the constants are added row-wise instead of column-wise.

\[
\alpha = \begin{bmatrix}
2 & 4 & 3 & F \\
6 & A & 8 & 8 \\
8 & 5 & A & 3 \\
0 & 8 & D & 3
\end{bmatrix},
\text{c}_1 = \begin{bmatrix}
1 & 3 & 1 & 9 \\
8 & A & 2 & E \\
0 & 3 & 7 & 0 \\
7 & 3 & 4 & 4
\end{bmatrix},
\text{c}_2 = \begin{bmatrix}
A & 4 & 0 & 9 \\
3 & 8 & 2 & 2 \\
2 & 9 & 9 & F \\
3 & 1 & D & 0
\end{bmatrix},
\]

\[
\text{c}_3 = \begin{bmatrix}
0 & 8 & 2 & E \\
F & A & 9 & 8 \\
E & C & 4 & E \\
6 & C & 8 & 9
\end{bmatrix},
\text{c}_4 = \begin{bmatrix}
4 & 5 & 2 & 8 \\
2 & 1 & E & 6 \\
3 & 8 & D & 0 \\
1 & 3 & 7 & 7
\end{bmatrix},
\text{c}_5 = \begin{bmatrix}
B & E & 5 & 4 \\
6 & 6 & C & F \\
3 & 4 & E & 9 \\
0 & C & 6 & C
\end{bmatrix},
\]

\[
\text{c}_6 = \begin{bmatrix}
C & 0 & A & C \\
2 & 9 & B & 7 \\
C & 9 & 7 & C \\
5 & 0 & D & D
\end{bmatrix},
\text{c}_7 = \begin{bmatrix}
3 & F & 8 & 4 \\
D & 5 & B & 5 \\
B & 5 & 4 & 7 \\
0 & 9 & 1 & 7
\end{bmatrix},
\text{c}_8 = \begin{bmatrix}
9 & 2 & 1 & 6 \\
D & 5 & D & 9 \\
8 & 9 & 7 & 9 \\
F & B & 1 & 8
\end{bmatrix},
\]

Permute_{p} (PermuteCells) is equal to the Permute_{p} transformation of Midori-64.
To recall, it operates as a permutation of the words of the state as

\[
\begin{bmatrix}
x_1 & x_5 & x_9 & x_{13} \\
x_2 & x_6 & x_{10} & x_{14} \\
x_3 & x_7 & x_{11} & x_{15} \\
x_4 & x_8 & x_{12} & x_{16}
\end{bmatrix}
\mapsto
\begin{bmatrix}
x_1 & x_{15} & x_{10} & x_8 \\
x_{11} & x_5 & x_4 & x_{14} \\
x_6 & x_{12} & x_{13} & x_3 \\
x_{16} & x_2 & x_7 & x_9
\end{bmatrix}.
\]

This corresponds to the permutation

\[
p = \begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
1 & 11 & 6 & 16 & 15 & 5 & 12 & 2 & 10 & 4 & 13 & 7 & 8 & 14 & 3 & 9
\end{pmatrix}.
\]

Note that this permutation ensures a higher number of active S-boxes compared to the choice made in Prince.

Mix_{M} (MixColumns). As in Midori, the involutory matrix

\[
M = \begin{pmatrix}
0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0
\end{pmatrix} \in \text{GL}_4(\mathbb{F}_2)\]

is applied to every of the four columns of the state.

**Encryption**

In the following, we define \( H_t : \mathbb{F}_2^{64} \times \mathbb{F}_2^{64} \times \mathbb{F}_2^{64} \rightarrow \mathbb{F}_2^{64} \) as the application of the \( t \) rounds \( R_1, \ldots R_t \) and one additional \( S_{\text{Mid64}} \) layer. More precisely, each instance is
defined as

\[ H_t(\cdot, \xi, h) = S_{\text{Sk64}} \circ R_t(\cdot, \text{Permute}_\sigma^t(h) + \xi) \circ \cdots \circ R_1(\cdot, \text{Permute}_\sigma^1(h) + \xi), \]

where \( \text{Permute}_\sigma \) permutes each word of the tweak state \( h \) as

\[
\begin{bmatrix}
  h_1 & h_5 & h_9 & h_{13} \\
  h_2 & h_6 & h_{10} & h_{14} \\
  h_3 & h_7 & h_{11} & h_{15} \\
  h_4 & h_8 & h_{12} & h_{16}
\end{bmatrix}
\mapsto
\begin{bmatrix}
  h_{10} & h_6 & h_{12} & h_{16} \\
  h_1 & h_5 & h_9 & h_{13} \\
  h_{14} & h_4 & h_8 & h_2 \\
  h_3 & h_7 & h_{11} & h_{15}
\end{bmatrix}.
\]

This corresponds to the permutation

\[ \sigma = \begin{pmatrix}
  1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
  10 & 1 & 14 & 3 & 6 & 5 & 4 & 7 & 12 & 9 & 8 & 11 & 16 & 13 & 2 & 15
\end{pmatrix}. \]

With this notation, we can define the tweakable block cipher \( \text{Mantis}_t : \mathbb{F}^{64}_2 \times \mathbb{F}^{128}_2 \times \mathbb{F}^{64}_2 \rightarrow \mathbb{F}^{64}_2 \) by giving each instance as

\[ \text{Mantis}_t(\cdot, k, h) = \text{Add}_{\zeta' + \xi + \alpha + h} \circ H_t^{-1}(\cdot, \xi + \alpha, h) \circ \text{Mix}_M \circ H_t(\cdot, \xi, h) \circ \text{Add}_{\zeta + \xi + h}. \]

**Decryption**

Because of the \( \alpha \)-reflection property, it is

\[ \text{Mantis}_t^{-1}(\cdot, k, h) = \text{Add}_{\zeta + \xi + \alpha + h} \circ H_t^{-1}(\cdot, \xi, h) \circ \text{Mix}_M \circ H_t(\cdot, \xi + \alpha, h) \circ \text{Add}_{\zeta' + \xi + \alpha + h}. \]

Test vectors for \( \text{Mantis}_t, t \in \{5, 6, 7, 8\} \), can be found in [BJK+16b].

### 5.4.2 Design Rationale

The goal was to design a cipher competitive to Prince in terms of latency with the advantage of being tweakable. In contrast to Skinny, we distinguish between tweak and key input. In particular, we allow an attacker to control the tweak but not the key. Thus, similar to Prince, we do not claim related-key security. In order to reach this goal, again, several components are borrowed from already existing ciphers. Note that, as we aim for an efficient unrolled implementation, one is not restricted to a classical round-iterated design. As latency was the main optimization goal, the security margin of Mantis is way smaller than that of Skinny. We chose to include the number of rounds as a defining parameter in each Mantis version in order to specify more aggressive and more conservative versions. Those can be targets for external cryptanalysis. While the small versions of Mantis for \( t \leq 4 \) can only be considered as “toy ciphers” and are just defined for the sake of completeness, Mantis\(_5\) refers to the most aggressive practical solution. We refer to Section 5.4.3 for the particular security claims and the best external cryptanalysis so far.
**α-Reflection Property**

Mantis is designed as a reflection cipher, i.e., encryption under a key $k$ equals decryption under a related key. This significantly reduces the implementation overhead for decryption. Therefore, the parameter $t$ denotes only half the number of rounds, as the second half of the cipher is basically the inverse of the first half. It is advantageous that Mix$_M$ is involutory since we need the middle part of the cipher to be an involution.

**The Choice of the Linear Layer**

To achieve low latency in a fully unrolled implementation, one is limited in the number rounds to be applied. Therefore, one has to achieve very fast diffusion and guarantee a high number of active S-boxes. To reach those requirements, we adopted the linear layer of Midori. It provides full diffusion after only three rounds and guarantees a high number of active S-boxes in the single-key setting. We refer to Table 5.8 for the actual bounds. The bounds (both in the single-key as in the related-tweak setting) were computed with the same MILP approach as in Skinny. Section 5.4.5 explains how the constraints for modelling the particular linear layer of Mantis were defined.

**The Choice of the Round Constants**

Compared to Midori, whose round constants are extremely sparse and structured, we decided to employ very dense (and basically random) round constants over the whole state, similar at it was done in Prince. This should in particular protect against invariant attacks. In Chapter 6 we look at the resistance of Mantis against invariant attacks in more detail.

**The Choice of the S-box**

For the S-box in Mantis we used the same S-box as in Midori-64. This S-box $S_{\text{Mid64}}$ has a low circuit depth and thus can be implemented to achieve a significantly lower latency than the Prince S-box. The maximum differential probability is $p_{S_{\text{Mid64}}} = 2^{-2}$ and the maximum absolute correlation is $c_{S_{\text{Mid64}}} = 2^{-1}$.

**The Choice of the Tweak Permutation $\sigma$**

Our aim was to choose a word permutation $\sigma \in S_{16}$ such that five rounds (plus one additional $S_{\text{Mid64}}$ layer) guarantee at least 16 active S-boxes in the related-tweak setting. This would guarantee at least 32 active S-boxes for Mantis$_2$, which is enough for the standard wide-trail argument on the resistance against (related-tweak) differential attacks (resp. linear attacks) based on a single trail. Since there are 16! possibilities for $h$, which is too much for an exhaustive search, we restricted ourself on a subclass of 8! tweak permutations. The restriction is that two complete rows (without changing the position of the words in those rows) are
permuted to different rows. In our case, the first and third row are permuted to
the second and fourth row, respectively. The bounds were derived using the MILP
approach. We tested several thousand choices for the permutation $\sigma$ and found
out that 16 active S-boxes were the best possible to reach over $H_5$. Out of these
optimal choices, we took the permutation that maximized the bound for $\text{Mantis}_5$
and, as a second step, for $\text{Mantis}_6$. We refer to Table 5.8 for the actual bounds.

Table 5.8: Lower bounds on the number of linear (and differential) active S-boxes
in the single-key model and on the number active S-boxes in related-tweak (RT)
differential attacks for $\text{Mantis}$.

<table>
<thead>
<tr>
<th></th>
<th>$\text{Mantis}_2$</th>
<th>$\text{Mantis}_3$</th>
<th>$\text{Mantis}_4$</th>
<th>$\text{Mantis}_5$</th>
<th>$\text{Mantis}_6$</th>
<th>$\text{Mantis}_7$</th>
<th>$\text{Mantis}_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>14</td>
<td>32</td>
<td>46</td>
<td>62</td>
<td>70</td>
<td>76</td>
<td>82</td>
</tr>
<tr>
<td>RT</td>
<td>6</td>
<td>12</td>
<td>20</td>
<td>34</td>
<td>44</td>
<td>50</td>
<td>56</td>
</tr>
</tbody>
</table>

5.4.3 Security Claim and Best Cryptanalysis so far

For $\text{Mantis}_7$, the original security claim is that any adversary who is in possession
of $2^n$ chosen plain/ciphertext pairs which were obtained under chosen tweaks, but
with a fixed unknown key, needs at least $2^{126-n}$ calls to the encryption function
in order to recover the secret key. Thus, the security claims are the same as for
$\text{Prince}$, except that also related-tweak security is claimed. Until now, no attack on
$\text{Mantis}_7$ that invalidates this claim has been published.

$\text{Mantis}$ was designed as a cipher having only a small security margin. In the
original design document, further cryptanalysis on the more aggressive versions of
$\text{Mantis}$ was explicitly encouraged. The designer’s claim on the aggressive version
$\text{Mantis}_5$ was security against practical attacks, similar to what has been considered
in the $\text{Prince}$ challenge. More precisely, it was claimed that no related-tweak attack
(better than the generic claim above) is possible against $\text{Mantis}_5$ with less than $2^{50}$
chosen (resp. $2^{40}$ known) plaintext/ciphertext pairs. It turned out that this claim
was too optimistic and it was invalidated by external cryptanalysis. In particular,
Dobraunig et al. presented a (related-tweak) key-recovery attack on $\text{Mantis}_5$ with a
(theoretical) data complexity of $2^{28}$ chosen plaintexts and a time complexity of
about $2^{38}$ block cipher operations. They implemented the attack and were able to
recover the key using $2^{30}$ chosen plaintext in about one core hour [DEKM16]. Very
recently, Eichlseder and Kales presented a (related-tweak) key-recovery attack on
$\text{Mantis}_6$ with a data complexity of $2^{53.94}$ chosen plaintexts and a time complexity
of $2^{53.94}$ computations [EK17]. The attacks use differential cryptanalysis methods
and exploit families of differential trails.
5.4.4 Unrolled Implementations

In Table 5.9 and Table 5.10, we list results of unrolled implementations for Mantis, constrained for the smallest area and the shortest latency, respectively.

Table 5.9: Unrolled implementations of several Mantis versions constrained for the smallest area (both encryption and decryption), Cell Library: UMC L180 0.18 µm.

<table>
<thead>
<tr>
<th>Area</th>
<th>Delay</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>Mantis5</td>
<td>8544</td>
<td>15.95 [BJK+16b]</td>
</tr>
<tr>
<td>Mantis6</td>
<td>9861</td>
<td>17.60 [BJK+16b]</td>
</tr>
<tr>
<td>Mantis7</td>
<td>11209</td>
<td>20.50 [BJK+16b]</td>
</tr>
<tr>
<td>Mantis8</td>
<td>12533</td>
<td>21.34 [BJK+16b]</td>
</tr>
<tr>
<td>Prince</td>
<td>8344</td>
<td>16.00 [MS16]</td>
</tr>
</tbody>
</table>

Table 5.10: Unrolled implementations of several Mantis versions constrained for the shortest delay (both encryption and decryption), Cell library: UMC L180 0.18 µm.

<table>
<thead>
<tr>
<th>Area</th>
<th>Delay</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>Mantis5</td>
<td>13424</td>
<td>9.00  [BJK+16b]</td>
</tr>
<tr>
<td>Mantis6</td>
<td>18375</td>
<td>10.00 [BJK+16b]</td>
</tr>
<tr>
<td>Mantis7</td>
<td>23926</td>
<td>11.00 [BJK+16b]</td>
</tr>
<tr>
<td>Mantis8</td>
<td>30252</td>
<td>12.00 [BJK+16b]</td>
</tr>
<tr>
<td>Prince</td>
<td>17693</td>
<td>9.00  [MS16]</td>
</tr>
</tbody>
</table>

5.4.5 The MILP Constraints for the MixColumns Operation

For computing the minimum number of active S-boxes in Mantis, we used the same MILP approach as for Skinny. The main difference is that the linear constraints for modelling the (word-wise) branching transitions of the transformation

\[
M : \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \mapsto \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x_2 + x_3 + x_4 \\ x_1 + x_3 + x_4 \\ x_1 + x_2 + x_4 \\ x_1 + x_2 + x_3 \end{pmatrix} =: \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}
\]

are more complex than for the linear mixing in Skinny. Therefore, we do not describe the complete model, but just give the constraints for M instead. In particular, let \(i_1, \ldots, i_4, o_1, \ldots, o_4, d\) be decision variables that can take values in \(\{0, 1\} \subseteq \mathbb{Z}\). We define

\[
C_M[i_1, i_2, i_3, i_4, o_1, o_2, o_3, o_4, d] := \bigcup_{j=1}^{5} C_j[i_1, i_2, i_3, i_4, o_1, o_2, o_3, o_4, d],
\]

where the five sets of linear constraints \(C_j[i_1, i_2, i_3, i_4, o_1, o_2, o_3, o_4, d]\), for \(j \in \{1, \ldots, 5\}\), are given as follows:

125
1. Constraints for describing the branch number:

\[ C_1[i_1, i_2, i_3, i_4, o_1, o_2, o_3, o_4, d] = \left\{ \sum_{j=1}^{4} i_j + o_j \geq 4d \right\} \cup \bigcup_{j=1}^{4} \{ i_j \leq d, \ o_j \leq d \} \]

2. An inactive input vector cannot turn active and vice versa:

\[ C_2[i_1, i_2, i_3, i_4, o_1, o_2, o_3, o_4, d] = \left\{ \sum_{j=1}^{4} i_j \geq d, \ \sum_{j=1}^{4} o_j \geq d \right\} \]

3. If \( y_l \) (resp. \( x_l \)) is active then at least one of the \( x_j \) (resp. \( y_j \)) with \( j \neq l \) is active:

\[ C_3[i_1, i_2, i_3, i_4, o_1, o_2, o_3, o_4, d] = \bigcup_{(j,l) \in \mathbb{N}_{\leq 4} \times \mathbb{N}_{\leq 4}} \{ i_j + i_l + o_l \geq o_j, \ o_j + o_l + i_l \geq i_j \} \]

4. If \( y_j \) (resp. \( x_j \)) is active then, for all \( l \neq j \), at least one of \( x_j, x_l, y_l \) (resp. \( y_j, y_l, x_l \)) is active:

\[ C_4[i_1, i_2, i_3, i_4, o_1, o_2, o_3, o_4, d] = \bigcup_{j,l \in \mathbb{N}_{\leq 4}} \{ i_j + i_l + o_l \geq o_j, \ o_j + o_l + i_l \geq i_j \} \]

5. Additional constraints:

\[ C_5[i_1, i_2, i_3, i_4, o_1, o_2, o_3, o_4, d] = \bigcup_{(d,j) \in \mathbb{N}_{\leq 4} \times \mathbb{N}_{\leq 4}} \{ o_j + \sum_{l \neq j}^{d} i_l \geq i_d, \ i_j + \sum_{l \neq d}^{l \neq j} o_l \geq o_d \} \]

The solutions for \( i_1, \ldots, i_4, o_1, \ldots, o_4 \) for which there exists a \( d \in \{0, 1\} \) such that all constraints in \( C_M[i_1, i_2, i_3, i_4, o_1, o_2, o_3, o_4, d] \) are fulfilled correspond to the possible branching transitions as depicted in Figure 4.2.

As for Skinny, the corresponding MILP instances were solved using Gurobi [GO16].
5.5 Conclusion and Future Work

We have shown the two tweakable block cipher designs **Skinny** and **Mantis**. While **Skinny** is designed to be a flexible lightweight cipher for various applications, **Mantis** is optimized for low-latency applications. Because of the extreme focus on latency, the security margin of **Mantis** is rather small. We explicitly encourage further cryptanalysis on the two designs.

Recently, Avanzi published the tweakable block cipher family **QARMA** for low-latency applications [Ava17]. It reuses components of **Mantis**, but adds several innovative design choices. For instance, its reflection property is built upon a non-involutory and keyed layer in the middle. Further, instead of using a MixColumns matrix with binary coefficients, it considers more general matrices over rings with zero-divisors for defining the linear layer. Those allow more possibilities while at the same time keeping the latency low. More analysis of **QARMA** and its general structure would be an interesting topic for future work.
Part II

Analysis of Lightweight Block Ciphers
Chapter 6

Invariant Attacks

Large parts of this chapter (with the exception of Section 6.3) are based on the publication [BCLR17], which is joint work with Anne Canteaut, Gregor Leander and Yann Rotella. All authors equally contributed. The main contribution of the author was in the first part of the paper (here Section 6.4), i.e., the algorithmic approach of proving the non-applicability of the invariant attack. In Section 6.5, he contributed to the proof of Theorem 6.1 and by proving Proposition 6.11.

6.1 Introduction

As explained in Section 2.3, the main idea of lightweight cryptography can be embraced as designing cryptographic primitives that put an extreme focus on performance. This in turn resulted in many new designs which achieve better performance by essentially removing any operations that are not strictly necessary (or believed to be necessary) for the security of the scheme. One particular interesting case of reducing the complexity is the design of the key schedule and the choice of round constants. Both of these are arguably the parts that we understand least and only very basic design criteria are available on how to choose a good key schedule or how to choose good round constants. Consequently, many of the lightweight block ciphers avoid any complexity in the key schedule at all. Instead, identical keys are used in the rounds and (often very simple and sparse) round constants are added on top (e.g., see LED [GPPR11], Skinny (Chapter 5), Prince [BCG+12], Mantis (Chapter 5), Midori [BBI+15], to mention a few).

However, several of those schemes were recently broken using a structural attack called invariant subspace attack [LAAZ11, LMR15], as well as the recently published generalization called nonlinear invariant attack [TLS16]. Indeed, those attacks have been successfully applied to quite a number of recent designs including PRINtCipheR [LAAZ11], Midori-64 [GJN+16, TLS16], iScream [LMR15].

1The original article published by Springer-Verlag is available at DOI: 10.1007/978-3-319-63715-0_22 (© IACR 2017). Here, parts of the text are rearranged, modified or omitted.
and SCREAM [TLS16], NORX v2.0 [CFG+17], Simpira v1 [Ron16] and Haraka v.0 [Jea16]. Both attacks, that we jointly call invariant attacks, notably exploit the fact that these lightweight primitives have a very simple key schedule where the same round key (up to the addition of a round constant) is applied in several rounds.

It is therefore of major importance to study invariant attacks in more detail and to determine whether a given primitive is vulnerable. More generally, it would be interesting to exhibit some design criteria for the building blocks of a cipher which guarantee the resistance against those attacks. As mentioned above, this would shed light on the fundamental open question on how to select proper round constants.

Results of this Chapter

After explaining the idea of invariant attacks, we show in Section 6.3 how they are related to the existence of linear approximations. In particular, we see that in many cases, the existence of an invariant for a keyed instance $E_k$ directly implies the existence of a linear approximation over $E_k$ with a significant high bias. This comes from the relation between invariant attacks and nonlinear approximations. Although this connection to linear cryptanalysis is a quite interesting observation, we are unfortunately not able to state anything more about those linear approximations besides their existence.

Then, we analyze the resistance of several lightweight substitution-permutation ciphers against invariant attacks. Our framework both covers the invariant subspace attack, as well as the recently published nonlinear invariant attack. By exactly formalizing the requirements of those attacks, we are able to reveal the precise mathematical properties that render those attacks applicable. Indeed, as we will detail below, the rational canonical form of the linear layer will play a major role in our analysis. Our results show that the linear layer and the round constants have a major impact on the resistance against invariant attacks, while this type of attacks was previously believed to be mainly related to the behaviour of the S-box, see e.g., [GJN+16]. In particular, if the number of invariant factors of the linear layer is small (for instance, if its minimal polynomial has a high degree), we can easily find round constants which guarantee the resistance to all types of invariant attacks, independently of the choice of the S-box layer. In order to ease the application of our results in practice, we implemented all our findings in Sage [StSDT16] and added the source code in Listing 6.1 in the end of this chapter.

In our framework, the resistance against invariant attacks is defined in the following sense: For each instantiation of the cipher with a fixed key, there is no function that is invariant for both the substitution layer and for the linear part of each round. This implies that any adversary who still wants to apply an invariant attack necessarily has to search for invariants over the whole round function, which appears to have a cost exponential in the block size in general. Indeed, all published invariant attacks we are aware of exploit weaknesses in the underlying
building blocks of the round. Therefore, our notion of resistance guarantees complete security against the major class of invariant attacks, including all variants published so far.

This analysis of the resistance against invariant attacks is split in two parts, a first part (Section 6.4) which can be seen as the attacker’s view on the problem and a second part (Section 6.5) which reflects more on the designer’s decision on how to avoid those attacks. More precisely, Section 6.4 details an algorithmic approach which enables an adversary to spot a possible weakness with respect to invariant attacks within a given cipher. For the lightweight block ciphers Skinny-64, Prince and Mantis7, the 7-round version of Mantis, this algorithm is used to prove the resistance against invariant attacks.

These results come from the following observation: Let $L$ denote the linear layer of the cipher in question and let $c_1, \ldots, c_t \in \mathbb{F}_2^n$ be the (XOR) differences between two round constants involved in rounds where the same round key is applied. Furthermore let $W_L(c_1, \ldots, c_t)$ denote the smallest $L$-invariant subspace of $\mathbb{F}_2^n$ that contains all $c_1, \ldots, c_t$. Then, one can guarantee resistance if $W_L(c_1, \ldots, c_t)$ covers the whole input space $\mathbb{F}_2^n$. As a direct result, we will see that in Skinny-64, there are enough differences between round constants to guarantee the full dimension of the corresponding $L$-invariant subspace. This directly implies the resistance of Skinny-64, and this result holds for any reasonable choice of the S-box layer. In contrast, for Prince and Mantis7, there are not enough suitable $c_i$ to generate a subspace $W_L(c_1, \ldots, c_t)$ with full dimension. However, for both primitives, we are able to keep the security argument by also considering the S-box layer, using the fact that the dimension of $W_L(c_1, \ldots, c_t)$ is not too low in both cases.

As a second part, in Section 6.5 we provide an in-depth analysis of the impact of the round constants and of the linear layer on the resistance against invariant attacks. The first question we study is the following:

*Given the linear layer $L$ of a cipher, what is the minimum number of round constants needed to guarantee resistance against the invariant attack, independently from the choice of the S-box?*

Figure 6.1 shows the maximal dimension that can be reached by $W_L(c_1, \ldots, c_t)$ when $t$ values of $c_i$ are considered. It shows in particular that the whole input space can be covered with only $t = 4$ values in the case of Skinny-64, while 8 and 16 values are needed for Prince and Mantis, respectively. This explains why, even though Prince and Mantis apply very dense round constants, the dimension does not increase rapidly for higher values of $t$. The observations in Fig. 6.1 are deduced from the invariant factors (or the rational canonical form) of the linear layer, as we show by the following theorem.

---

2 For Mantis7, the resistance against invariant attacks is only proven for the untweaked version, i.e., the tweak input is considered to be zero.

3 We have to provide that the S-box has no component of degree 1. If the S-box has such a linear or affine component, the cipher would be vulnerable against linear cryptanalysis.
Figure 6.1: For Skinny-64, Prince and Mantis, this figure shows the highest possible dimension of $W_L(c_1, \ldots, c_t)$ for $t$ values $c_1, \ldots, c_t$ (see Theorem 6.1).

Figure 6.2: For several lightweight ciphers, this figure shows the probability that $W_L(c_1, \ldots, c_t) = \mathbb{F}_2^n$ for uniformly random constants $c_i$ (see Theorem 6.3).

Theorem 6.1. Let $Q_1, \ldots, Q_r$ be the invariant factors of the linear layer $L$ and let $t \leq r$. Then

$$
\max_{c_1, \ldots, c_t \in \mathbb{F}_2^n} \dim W_L(c_1, \ldots, c_t) = \sum_{i=1}^{t} \deg Q_i.
$$

For the special case of a single constant $c$, the maximal dimension of $W_L(c)$ is equal to the degree of the greatest invariant factor of $L$, i.e., the minimal polynomial of $L$. We will also explain how the particular round constants must be chosen in order to guarantee the best possible resistance.

As designers often choose random round constants to instantiate the primitive, we were also interested in the following question:

*How many randomly chosen round constants are needed to guarantee the best possible resistance with a high probability?*

We derive an exact formula for the probability that the subspace $W_L(c_1, \ldots, c_t)$ has full dimension for $t$ uniformly random constants $c_i$. Fig. 6.2 gives an overview of this probability for several lightweight designs.

6.2 Preliminaries and Explanation of the Attack

Let $f \in B_n$ be an $n$-bit Boolean function. The *derivative of $f$ in direction $\alpha \in \mathbb{F}_2^n$* is the Boolean function defined by $\Delta_{\alpha} f := x \mapsto f(x + \alpha) + f(x)$. The following terminology will be extensively used in this chapter. It refers to the constant derivatives which play a major role in the context of invariant attacks.
Definition 6.1 (see [Lai95]). An element \( \alpha \in \mathbb{F}_2^n \) is said to be a linear structure of \( f \in \mathcal{B}_n \) if the corresponding derivative \( \Delta_\alpha f \) is constant. The set of all linear structures of a function \( f \) is a linear subspace of \( \mathbb{F}_2^n \) and is called the linear space of \( f \):

\[
\mathcal{L}(f) := \{ \alpha \in \mathbb{F}_2^n \mid \Delta_\alpha f = \epsilon, \ \epsilon \in \{0, 1\} \}.
\]

The nonlinear invariant attack was described in [TLS16] as a distinguishing attack on block ciphers. Indeed, the authors were able to construct an attack on the lightweight schemes Midori-64, Scream [GLS+15], and iScream [GLS+14]. For an \((n, \kappa)\)-block cipher \( E \), the idea is to find a subset \( S \subset \mathbb{F}_2^n \) for which the partition of the input set into \( S \cup (\mathbb{F}_2^n \setminus S) \) is preserved by the keyed instance \( E_k \) for as many keys \( k \) as possible, i.e.,

\[
E_k(S) = S \text{ or } E_k(S) = \mathbb{F}_2^n \setminus S.
\]

The keys for which this property is fulfilled are called weak keys. The special case when \( S \) is an affine space corresponds to the so-called invariant subspace attack [LAAZ11].

An equivalent formulation is obtained by considering the \( n \)-bit Boolean function \( g \in \mathcal{B}_n \) defined by \( g(x) = 1 \) if and only if \( x \in S \). Then, finding an invariant consists in finding a function \( g \in \mathcal{B}_n \) such that \( g + g \circ E_k \) is constant. This motivates the following definition.

Definition 6.2 (see [TLS16]). Let \( F : \mathbb{F}_2^n \to \mathbb{F}_2^n \) be a permutation. A Boolean function \( g \in \mathcal{B}_n \) for which \( g + g \circ F \in \{0, 1\} \) is called an invariant for \( F \). We denote the set of all invariants for \( F \) by

\[
\mathcal{U}(F) := \{ g \in \mathcal{B}_n \mid g + g \circ F \text{ is constant} \}.
\]

As observed in [TLS16], the set \( \mathcal{U}(F) \) is a linear subspace of \( \mathcal{B}_n \). If \( n \) is quite small, e.g., \( n = 4 \) in the example of a four-bit S-box, the space \( \mathcal{U}(F) \) can be efficiently computed. An important remark, which will be used later, is that if \( F \) has a cycle of odd length, then all \( g \in \mathcal{U}(F) \) satisfy \( g + g \circ F = 0 \).

Whenever we are trying to find an invariant for a keyed instance \( E_k \) of a block cipher, we obviously focus on non-trivial invariants, i.e., on \( g \notin \{0, 1\} \). When trying to find an invariant in practice, one usually tries to find a Boolean function \( g \in \mathcal{B}_n \) that is an invariant for all the building blocks of the cipher simultaneously. Indeed, all the invariant attacks we are aware of exploit the iterative structure of the block cipher. We now illustrate the attack on the block cipher Midori-64 as an example.

Example 6.1 (Invariant Attack on Midori-64 as presented in [TLS16]). The block length is \( n = 64 \) and the key length is \( \kappa = 128 \). If we denote the four-bit word in the \( j \)-th cell of the state by \((x_{j,3}, x_{j,2}, x_{j,1}, x_{j,0})\), the invariant \( g \in \mathcal{B}_n \) is given by

\[
g(x) = \sum_{j=1}^{16} (x_{j,3}x_{j,2} + x_{j,2} + x_{j,1} + x_{j,0})
\]
This function is an invariant for the S-box layer as the four-bit Boolean function

\[(x_{j,3}, x_{j,2}, x_{j,1}, x_{j,0}) \mapsto x_{j,3}x_{j,2} + x_{j,2} + x_{j,1} + x_{j,0}\]

is contained in \(U(S_{\text{Mid}64})\). Further, since this function is quadratic and since the MixColumns matrix of Midori is orthogonal and consists only of coefficient in \(\mathbb{F}_2\), Theorem 1 in [TLS16] applies and \(g\) is an invariant for the linear layer \(L\) and thus also for the (unkeyed) round function \(L \circ S\). To determine the set of weak keys, one has to identify the round keys \(k_i\) such that \(g\) is an invariant for \(\text{Add}_{k_i}\). For this, we rely on the following fact, which was already implicitly contained in [TLS16].

**Proposition 6.1.** Let \(g \in B_n\). Then, \(g\) is an invariant for the key addition \(\text{Add}_{k_i}\), if and only if \(k_i \in \text{LS}(g)\).

**Proof.** By the definition of the invariant property, there exists a constant \(\varepsilon \in \mathbb{F}_2^n\) such that

\[\forall x \in \mathbb{F}_2^n : g(x) + g(x + k_i) = \varepsilon.\]

This coincides with the definition of \(k_i \in \text{LS}(g)\). \(\square\)

This implies that the weak round keys in Midori-64 are those for which \(k_{i,j,2} = 0\) for all \(j\). Due to the simplicity of the key-scheduling algorithm, this happens for \(2^{64}\) out of all \(2^{128}\) possible initial keys \(k\).

### 6.3 A Link to Linear Cryptanalysis

In Section 5.2.3, we have already mentioned how the invariant attack is related to nonlinear approximations. In particular, an invariant \(g\) for a keyed instance \(E_k\) defines a nonlinear approximation with absolute correlation equal to one, i.e., \(|\text{cor}_{E_k}(g,g)| = 1\). Similar to the Theorem of Linear Trail Composition (Theorem 5.2), one can show the following for nonlinear approximations. For a Boolean function \(g \in B_n\) and a mask \(\gamma \in \mathbb{F}_2^n\), we define \(\text{cor}_g(\gamma) := \text{cor}_g(\gamma, 1) = 2 \cdot \text{Prob}_x (\langle \gamma, x \rangle = g(x)) - 1\).

**Proposition 6.2** (Nonlinear Trail Composition). Let \(F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n\) and let \(g, h \in B_n\). Then,

\[\text{cor}_F(g, h) = \sum_{\gamma, \gamma' \in \mathbb{F}_2^n} \text{cor}_g(\gamma) \text{cor}_F(\gamma, \gamma') \text{cor}_h(\gamma').\]

**Proof.** The proof is very similar to the proof of Theorem 5.2. We first show that, for any \(\gamma' \in \mathbb{F}_2^n\), \(\sum_{\gamma \in \mathbb{F}_2^n} \text{cor}_g(\gamma) \text{cor}_F(\gamma, \gamma') = \text{cor}_F(g, l_{\gamma'})\). For this, we use the fundamental fact given in Equation 5.4, i.e.,

\[\sum_{\gamma \in \mathbb{F}_2^n} (-1)^{\langle \gamma, x \rangle} = \begin{cases} 2^n & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases}.\]
In particular,
\[
\sum_{\gamma \in \mathbb{F}_2^n} \text{cor}_g(\gamma) \text{cor}_F(\gamma, \gamma') \\
= \sum_{\gamma \in \mathbb{F}_2^n} \frac{1}{2^n} \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^2} \sum_{x' \in \mathbb{F}_2^2} (-1)^{\langle \gamma, x \rangle + g(x)} \sum_{\gamma \in \mathbb{F}_2^n} (-1)^{\langle \gamma, x' \rangle + \langle \gamma', F(x') \rangle} \\
= \frac{1}{2^n} \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} \sum_{x' \in \mathbb{F}_2^n} (-1)^{g(x) + \langle \gamma', F(x') \rangle} \sum_{\gamma \in \mathbb{F}_2^n} (-1)^{\langle \gamma, x + x' \rangle} \\
= \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{g(x) + \langle \gamma', F(x) \rangle} = \text{cor}_F(g, l_{\gamma'}). 
\]

Using the fact proven above, we proceed to prove the result as follows:
\[
\sum_{\gamma \in \mathbb{F}_2^n} \sum_{\gamma' \in \mathbb{F}_2^n} \text{cor}_g(\gamma) \text{cor}_F(\gamma, \gamma') \text{cor}_h(\gamma') \\
= \sum_{\gamma' \in \mathbb{F}_2^n} \text{cor}_h(\gamma') \sum_{\gamma \in \mathbb{F}_2^n} \text{cor}_g(\gamma) \text{cor}_F(\gamma, \gamma') \\
= \sum_{\gamma' \in \mathbb{F}_2^n} \text{cor}_h(\gamma') \text{cor}_F(g, l_{\gamma'}) \\
= \frac{1}{2^n} \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} \sum_{x' \in \mathbb{F}_2^n} (-1)^{g(x') + h(x)} \sum_{\gamma' \in \mathbb{F}_2^n} (-1)^{\langle \gamma', x + F(x') \rangle} \\
= \frac{1}{2^n} \sum_{x' \in \mathbb{F}_2^n} (-1)^{g(x') + h(F(x'))} = \text{cor}_F(g, h). 
\]

This result shows how a nonlinear approximation is related to multiple linear approximations. When applying Proposition 6.2 to an invariant $g$ for $E_k$, one obtains
\[
1 = |\text{cor}_{E_k}(g, g)| = \sum_{\gamma, \gamma' \in \Gamma_g} \text{cor}_g(\gamma) \text{cor}_{E_k}(\gamma, \gamma') \text{cor}_g(\gamma''), \quad (6.1) 
\]
where $\Gamma_g := \{\gamma | \text{cor}_g(\gamma) \neq 0\}$. If we consider the invariant attacks on Midori-64, SCREAM and iSCREAM as presented in [TLS16], one observes that in those examples the absolute value of $\text{cor}_g(\gamma)$ is the same for all $\gamma$ in $\Gamma_g$. In particular, Equation 6.1 simplifies to
\[
|\sum_{\gamma, \gamma' \in \Gamma_g} (-1)^{f(\gamma) + f(\gamma')} \text{cor}_{E_k}(\gamma, \gamma')| = 2^{32}, \quad (6.2) 
\]
where $f$ is an $n$-bit Boolean function and $|\Gamma_g| = 2^{32}$.
Midori-64

We have \( \Gamma_g = \{ (\gamma_1, \ldots, \gamma_{16}) \mid \forall j: \gamma_{j,0} = \gamma_{j,1} = 1 \} \) and
\[
f(\gamma) = \sum_{j=1}^{16} \gamma_{j,3}\gamma_{j,2} + \gamma_{j,3}.
\]

SCREAM

It is \( n = 128 \) and the invariant is given by
\[
g(x) = \sum_{j=1}^{16} (x_{j,2}x_{j,1} + x_{j,5} + x_{j,2} + x_{j,0}).
\]

In Equation 6.2 we have
\[
\Gamma_g = \{ (\gamma_1, \ldots, \gamma_{16}) \mid \forall j: \gamma_{j,0} = \gamma_{j,5} = 1, \gamma_{j,3} = \gamma_{j,4} = \gamma_{j,6} = \gamma_{j,7} = 0 \}
\]
and
\[
f(\gamma) = \sum_{j=1}^{16} \gamma_{j,2}\gamma_{j,1} + \gamma_{j,2}.
\]

iSCREAM

It is \( n = 128 \) and the invariant is given by
\[
g(x) = \sum_{j=1}^{16} (x_{j,5}x_{j,4} + x_{j,6} + x_{j,0}).
\]

In Equation 6.2 we have
\[
\Gamma_g = \{ (\gamma_1, \ldots, \gamma_{16}) \mid \forall j: \gamma_{j,0} = \gamma_{j,6} = 1, \gamma_{j,1} = \gamma_{j,2} = \gamma_{j,3} = \gamma_{j,7} = 0 \}
\]
and
\[
f(\gamma) = \sum_{j=1}^{16} \gamma_{j,5}\gamma_{j,4}.
\]

From Equation 6.2 one can deduce that, for each weak key, there must exist linear approximations (over the whole cipher, and in fact for all possible number of rounds) with absolute correlation larger than or equal to \( 2^{-32} \). Since the block length of SCREAM and iSCREAM is \( n = 128 \), this implies the existence of linear approximations with absolute correlation larger than \( 2^{-\frac{32}{2}} \), i.e., linear approximations that can be exploited by a standard linear attack.

The particular reason for the nonlinear trail composition simplifying to Equation 6.2 in those three examples is that the invariants are quadratic. In general, one can prove the following theorem.\(^4\)

\(^4\)We gratefully acknowledge the contribution of Anne Canteaut, who communicated this relation.
Theorem 6.2 (Existence of Linear Approximations from Quadratic Invariants). Let $g \in B_n$ be a quadratic invariant for a permutation $F: \mathbb{F}_2^n \to \mathbb{F}_2^n$. Then, there exists a Boolean function $f \in B_n$ such that

$$\left| \sum_{\gamma, \gamma' \in \Gamma_g} (-1)^{f(\gamma) + f(\gamma')} \text{cor}_F(\gamma, \gamma') \right| = |\Gamma_g|$$

and $|\Gamma_g| = 2^{n-\text{dim} LS(g)}$. Moreover, if $g$ is balanced, there must exist $\gamma, \gamma' \neq 0$ for which $|\text{cor}_F(\gamma, \gamma')| \geq 2^{\text{dim} LS(g)-n}$.

Proof. From Proposition 6.2, we obtain

$$\left| \sum_{\gamma, \gamma' \in \Gamma_g} \text{cor}_g(\gamma) \text{cor}_g(\gamma') \text{cor}_F(\gamma, \gamma') \right| = 1 ,$$

where $\Gamma_g = \{ \gamma \in \mathbb{F}_2^n | \text{cor}_g(\gamma) \neq 0 \}$. From Theorem 4 in [Car07] it follows that, for any $\gamma \in \mathbb{F}_2^n$, it is $\text{cor}_g(\gamma) \in \{0, \pm 2^{\text{dim} LS(g)-n}\}$. This implies

$$\left| \sum_{\gamma, \gamma' \in \Gamma_g} (-1)^{f(\gamma) + f(\gamma')} \text{cor}_F(\gamma, \gamma') \right| = 2^{n-\text{dim} LS(g)}$$

for an appropriate function $f \in B_n$.

To see that $|\Gamma_g| = 2^{n-\text{dim} LS(g)}$, we use Parseval’s relation (see Corollary 3 in [Car07]) and obtain

$$1 = \sum_{\gamma \in \mathbb{F}_2^n} \text{cor}_g^2(\gamma) = \sum_{\gamma \in \Gamma_g} \text{cor}_g^2(\gamma) = |\Gamma_g|2^{\text{dim} LS(g)-n} .$$

The existence of $\gamma, \gamma' \in \Gamma_g$ for which $|\text{cor}_F(\gamma, \gamma')| \geq 2^{\text{dim} LS(g)-n}$ immediately follows. If $g$ is balanced, we have $0 \notin \Gamma_g$. \qed

6.4 Proving the Absence of Invariants in Lightweight SPNs

In the remainder of this chapter, we concentrate on block ciphers which follow the specific structure of SP networks as depicted in Figure 2.3.

In practice, the technique applied for finding invariants for the cipher usually consists in exploiting its iterative structure and in searching for functions which are invariant for all constituent building blocks. Indeed, computing invariants for the round function is in general infeasible for a proper block size, typically $n = 64$ or $n = 128$. Despite the fact that all published invariant attacks we are aware of exploit invariants for all the constituent building blocks, the algorithm described in [LMR15] searches for invariant subspaces over the whole round function. However, it can only be applied in the special case for finding an invariant subspace,
and not for detecting an arbitrary invariant set. Moreover, it only detects spaces of large dimension efficiently.

Therefore, we consider in the following only those invariants that are invariant under both the substitution layer $S$ and the linear parts $\text{Add}_{k_i} \circ L$ of all rounds. The linear spaces of these invariants have then a very specific structure as pointed out in the following proposition.

**Proposition 6.3.** Let $L$ be a linear permutation on $\mathbb{F}_2^n$. Let $g \in B_n$ be an invariant for both $\text{Add}_{k_i} \circ L$ and $\text{Add}_{k_j} \circ L$ for two round keys $k_i$ and $k_j$. Then, $\text{LS}(g)$ is a linear space invariant under $L$ which contains $(k_i + k_j)$.

**Proof.** By definition of $g$, there exist $\epsilon_i, \epsilon_j \in \mathbb{F}_2$ such that, for all $x \in \mathbb{F}_2^n$,

$$g(x) = g(L(x) + k_i) + \epsilon_i \text{ and } g(x) = g(L(x) + k_j) + \epsilon_j.$$  

This implies that, for all $x \in \mathbb{F}_2^n$,

$$g(L(x) + k_i) + g(L(x) + k_j) = \epsilon_i + \epsilon_j,$$

or equivalently, by replacing $(L(x) + k_j)$ by $y$:  

$$g(y + k_i + k_j) + g(y) = \epsilon_i + \epsilon_j, \quad \forall y \in \mathbb{F}_2^n$$

and thus $(k_i + k_j) \in \text{LS}(g)$. We then have to show that $\text{LS}(g)$ is invariant under $L$. Let $s \in \text{LS}(g)$. Then, there exists a constant $\epsilon \in \mathbb{F}_2$ such that $g(x) = g(x + s) + \epsilon$. Since $g$ is an invariant for $\text{Add}_{k_i} \circ L$, we deduce

$$g(L(x) + k_i) + \epsilon_i = g(x) = g(x + s) + \epsilon = g(L(x) + L(s) + k_i) + (\epsilon_i + \epsilon).$$

Finally, we set $y := L(x) + k_i$ and obtain

$$g(y) = g(y + L(s)) + \epsilon$$

(6.3)

which completes the proof.

Therefore, the attack requires the existence of an invariant for the substitution layer whose linear space is invariant under $L$ and contains all differences between the round keys.\footnote{Note that a similar observation was already made in [Ava17] in the context of the invariant subspace attack.} The difference between two round keys, which should be contained in $\text{LS}(g)$, is in general dependent on the initial key. However, if we consider only designs where some round keys are equal up to the addition of a round constant, we obtain that the differences between these round constants must belong to $\text{LS}(g)$. Then, $\text{LS}(g)$ is a linear space invariant under $L$ which contains the differences $(c_i + c_j)$ for any pair $(i, j)$ of rounds such that $k_i = k + c_i$ and $k_j = k + c_j$. The smallest such subspaces are spanned by the cycles of $L$ as shown by the following lemma.
Lemma 6.1. Let $L$ be a linear permutation of $\mathbb{F}_2^n$. For any $c \in \mathbb{F}_2^n$, the smallest $L$-invariant linear subspace of $\mathbb{F}_2^n$ which contains $c$, denoted by $W_L(c)$, is

$$\text{span}\{L^i(c) \mid i \geq 0\}.$$  

Proof. Obviously, $\text{span}\{L^i(c) \mid i \geq 0\}$ is included in $W_L(c)$, since $W_L(c)$ is a linear subspace of $\mathbb{F}_2^n$ and is invariant under $L$. Moreover, we observe that $\text{span}\{L^i(c) \mid i \geq 0\}$ is invariant under $L$. Indeed, for any $\lambda_1, \lambda_2 \in \mathbb{F}_2$ and any $(i, j)$,

$$L(\lambda_1 L^i(c) + \lambda_2 L^j(c)) = \lambda_1 L^{i+1}(c) + \lambda_2 L^{j+1}(c)$$

and then belongs to $\text{span}\{L^i(c) \mid i \geq 0\}$. Then, this subspace is the smallest linear subspace of $\mathbb{F}_2^n$ invariant under $L$ which contains $c$. \qed

Let now $D$ be a set of known differences between round keys, i.e., a subset of all $k_i + k_j = (c_i + c_j)$. We define the subset

$$W_L(D) := \sum_{c \in D} \text{span}\{L^i(c) \mid i \geq 0\} = \sum_{c \in D} W_L(c).$$

We then deduce from the previous observations that the invariant attack applies only if there is a non-trivial invariant $g$ for the S-box layer such that $W_L(D) \subseteq \text{LS}(g)$. A SageMath code that computes the linear space $W_L(D)$ for a predefined list $D$ is given in Listing 6.1 in the end of this chapter. It has been used for determining the dimension of $W_L(D)$ corresponding to the round constants in several lightweight ciphers.

Skinny-64

Considering the untweaked version Skinny-64-64, one observes that the round keys repeat every 16 rounds. We define

$$D := \{c_1 + c_{17}, c_2 + c_{18}, c_3 + c_{19}, c_4 + c_{20}, c_5 + c_{21}\},$$

where the round constants $c_i$ are those of Skinny and of the form

$$c_i = \begin{bmatrix} a_r & 0 & 0 & 0 \\ b_r & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

as described in Algorithm 5.2. We obtain $\text{dim} W_L(D) = 64$. 

141
Skinny-128

In Skinny-128, the round constants are all of the following form:

\[
\begin{bmatrix}
  a & 0 & 0 & 0 \\
  b & 0 & 0 & 0 \\
  02 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

with 8-bit values \(a \in \{00, \ldots, 0F\}\) and \(b \in \{00, \ldots, 03\}\). Then, as the linear layer is defined by a binary matrix, we can see that the dimension of \(W_L(D)\) is at most 64, because none of the four most significant bits will be activated with any round constant.

Prince

Prince uses ten round keys \(k_i, 1 \leq i \leq 10\), which are all of the form \(k_i = k + c_i\). The so-called \(\alpha\)-reflection property implies that, for any \(i\), \(k_i + k_{11-i} = \alpha\) where \(\alpha\) is a fixed constant. The particular values of the round constants \(c_i\) and \(\alpha\) can be found in the specification of Prince [BCG+12]. We can then consider the set of (independent) round constant differences

\[D = \{\alpha, c_1 + c_2, c_1 + c_3, c_1 + c_4, c_1 + c_5\}\.

We obtain that \(\dim W_L(D) = 56\).

Mantis

As Prince, also Mantis has the \(\alpha\)-reflection property. We therefore consider the following set of round constant differences:\(^6\)

\[D = \{\alpha, c_1 + c_2, c_1 + c_3, c_1 + c_4, c_1 + c_5, c_1 + c_6, c_1 + c_7\},
\]

where the \(c_i\) and \(\alpha\) are as given in Section 5.4. We obtain that \(\dim W_L(D) = 42\).

Midori-64

In Midori-64, the round constants are only added to the least significant bit of each word and the linear layer does not provide any mixing within the words. Then \(W_L(D) = \{(0, 0, 0, 0), (0, 0, 0, 1)\}^{16}\), and has dimension 16 only.

As the invariant attack applies only if there is a non-trivial invariant \(g\) for the S-box layer such that \(W_L(D) \subseteq LS(g)\), by intuition, the attack should be harder as the dimension of \(W_L(D)\) increases. In the following, we analyze the impact of the dimension of \(W_L(D)\) to the applicability of the attack in detail and present a method to prove the non-existence of invariants based on this dimension.

\(^6\)For Mantis, we assume the tweak to be zero. Otherwise the key schedule is not of the simple form as described above.
6.4.1 The Simple Case

We first consider a simple case, i.e., when the dimension of $W_L(D)$ is at least $n - 1$.

**Proposition 6.4.** Suppose that the dimension of $W_L(D)$ is at least $n - 1$. Then, any $g \in B_n$ such that $W_L(D) \subseteq \text{LS}(g)$ is linear, affine or constant. As a consequence, there is no non-trivial invariant $g$ of the S-box layer such that $W_L(D) \subseteq \text{LS}(g)$, unless the S-box layer has a component of degree 1.

**Proof.** From [Car07, Prop. 14], it follows that

$$\dim \text{LS}(g) \geq k \Leftrightarrow \deg g \leq \begin{cases} n - k & \text{if } k \neq n \\ 1 & \text{if } k = n \end{cases}.$$  

This implies that $g$ must be linear, affine or constant. Invariants of algebraic degree 1 imply the existence of a linear approximation with probability 1, or equivalently that the S-box has a component (i.e., a linear combination of its coordinates) of degree 1.

In the rest of the chapter, we will implicitly exclude the case when the S-box has a component of degree 1, as the cipher would be already broken by linear cryptanalysis.

**Skinny-64**

As shown before, for the untweaked version Skinny-64 one obtains $\dim W_L(D) = 64$. This indicates that the round constants do not allow non-trivial invariants that are invariant for both the substitution and the linear parts of Skinny-64, and this result holds for any choice of the S-box layer.

Unfortunately, the dimension of $W_L(D)$ is not high enough for the other ciphers we considered. For those primitives, we therefore cannot prove the resistance against invariant attacks based on the linear layer only.

6.4.2 When the Dimension is Smaller

Not every cipher applies round constants such that the dimension of $W_L(D)$ is larger than or equal to $n - 1$. Even for Prince and Mantis, which have very dense round constants, it is not the case and we cannot directly rely on this argument. However, if $n - \dim W_L(D)$ is small, we can still prove that the invariant attack does not apply but only by exploiting some information on the S-box layer. This can be done by checking whether there exists a non-trivial invariant $g$ for the S-box layer which admits some given elements as 0-linear structures, in the sense of the following definition.
Definition 6.3. A linear structure $\alpha$ of a Boolean function $f$ is called a $0$-linear structure if the corresponding derivative equals the all-zero function. The set of all $0$-linear structures of $f$ is a linear subspace of $\text{LS}(f)$ denoted by $\text{LS}_0(f)$. Elements $\beta$ s.t. $\Delta_\beta g = 1$ are called $1$-linear structures of $f$.

Note that $0$-linear structures are also called invariant linear structures. It is well known that the dimension of $\text{LS}_0(f)$ drops by at most $1$ compared to $\text{LS}(f)$ [DW97].

Checking that all invariants are constant based on $0$-linear structures.

In the following, we search for an invariant $g$ for the S-box layer $S$ that is also invariant for the linear part of each round. Suppose now, in a first step, that we know a subspace $Z$ of $\text{LS}(g)$ which is composed of $0$-linear structures only. In other words, we now search for an invariant $g$ for $S$ such that $\text{LS}_0(g) \supseteq Z$ for some fixed $Z$. If the dimension of this subspace $Z$ is close to $n$, we can try to prove that any such invariant is constant based on the following observation.

Proposition 6.5. Let $g$ be an invariant for a permutation $S: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ such that $\text{LS}_0(g) \supseteq Z$ for some given subspace $Z \subset \mathbb{F}_2^n$. Then

(i) $g$ is constant on each coset of $Z$;

(ii) $g$ is constant on $S(Z)$.

Proof. Since $Z \subseteq \text{LS}_0(g)$, for any $a \in \mathbb{F}_2^n$, we have that $g(a + z) = g(a)$ for all $z \in Z$, i.e., $g$ is constant on all $(a + Z)$. Now, we use that $g$ is an invariant for $S$, which means that there exists $\varepsilon \in \mathbb{F}_2$ such that $g(S(x)) = g(x) + \varepsilon$. Since $g$ is constant on $Z$, we deduce that $g$ is constant on $S(Z)$.

To show that $g$ must be trivial, the idea is to evaluate the S-box layer at some points in $Z$ and deduce that $g$ takes the same value on all corresponding cosets. The number of distinct cosets of $Z$ equals $2^{n - \dim Z}$, which is not too large when $\dim Z$ is close to $n$. Then, we hope that all cosets will be hit when evaluating $S$ at a few points in $Z$. In this situation, $g$ must be a constant function. In other words, we are able to conclude that there do not exist non-trivial invariants for both the substitution layer and the linear part.

In our experiments, we used the following very simple algorithm. If it terminates, all invariants must be constant. An efficient SageMath implementation of Algorithm 6.1 is given in Listing 6.1 at the end of this chapter.

Determining a suitable $Z$ from $W_L(D)$

Up to now, we assumed the knowledge of a subspace $Z$ of $W_L(D)$ for which $Z \subseteq \text{LS}_0(g)$ for all invariants $g$ we are considering. But, the fact that some elements are $0$-linear structures depends on the actual invariant $g$ and thus, each of the elements $d \in W_L(D)$ might or might not be a $0$-linear structure. However,
Algorithm 6.1 Checking that $\mathcal{U}(S) \cap \{ g \in B_n \mid Z \subseteq \text{LS}_0(g) \}$ is trivial

1: $R = \{\}$
2: repeat
3: $z \overset{?}{\in} Z$
4: Compute $S(z)$
5: Add to $R$ a representative of the coset defined by $S(z)$
6: until $|R| = 2^n - \dim Z$

some 0-linear structures can be determined by using one of the two following approaches.

**First approach.** The first observation comes from (6.3) in the proof of Prop. 6.3

**Lemma 6.2.** Let $L$ be a linear permutation on $\mathbb{F}_2^n$. Let $g \in B_n$ be an invariant for $\text{Add}_{k_i} \circ L$ for some $k_i$ and let $V$ be a subspace of $\text{LS}(g)$ which is invariant under $L$. Then, for any $v \in V$, $(v + L(v)) \in \text{LS}_0(g)$.

**Proof.** Let $v \in V$. Similar as in the proof of Prop. 6.3, we use that $g$ is an invariant for $\text{Add}_{k_i} \circ L$ and see that there exists an $\epsilon \in \mathbb{F}_2$ such that, for all $x \in \mathbb{F}_2^n$,

$$g(x) = g(x + v) + \epsilon = g(x + L(v)) + \epsilon.$$

We finally set $y := x + v$ and obtain

$$g(y) = g(y + v + L(v)),$$

implying that $v + L(v)$ is a 0-linear structure for $g$. \qed

Following the previous lemma, one option is to just run Algorithm 6.1 on $Z = \text{WL}(D')$ with $D' = \{d + L(d) \mid d \in D\}$. The disadvantage is that the dimension of $Z$ might be too low and therefore the algorithm might be too inefficient. In this case, one can also consider a different approach and run the algorithm several times, by considering all possible choices for the 0-linear structures among all elements in $D$. Suppose that, in the initial set of constants $D = \{d_1, d_2, \ldots, d_m, \ldots, d_t\}$, the elements $d_1, \ldots, d_m$ are all 1-linear structures and the elements $d_{m+1}, \ldots, d_t$ are all 0-linear structures for some invariant $g$ with $\text{LS}(g) \supseteq \text{WL}(D)$. One can now consider

$$D' = \{d_1 + L(d_1), d_2 + L(d_2), \ldots, d_m + L(d_m), d_{m+1}, \ldots, d_t, d_1 + d_2, \ldots, d_1 + d_m\}$$

which increases the dimension of $\text{WL}(D')$ by adding the sums of the 1-linear structures. We then have $\text{WL}(D') \subseteq \text{LS}_0(g)$ and we can apply Algorithm 6.1 on $Z = \text{WL}(D')$. Since we cannot say in advance which of the constants are 1-linear structures, there are $2^t$ possible choices of such a subspace $\text{WL}(D')$ and we run Algorithm 6.1 on all of them. This approach still might be very inefficient due to the smaller dimension of $\text{WL}(D')$ and since Algorithm 6.1 has to be run $2^t$ times.
Second approach. If the S-box layer $S$ of the cipher has an odd-length cycle (i.e., if every S-box has an odd-length cycle), we can come up with the following.

Proposition 6.6. Let $g \in \mathcal{U}(S)$ where $S: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is a permutation with an odd cycle. Then, any linear structure of $g$ which belongs to the image set of $(S + \text{id}_n)$, i.e., $\{S(x) + x \mid x \in \mathbb{F}_2^n\}$, is a 0-linear structure of $g$.

Proof. If the S-box layer has an odd cycle, then any $g \in \mathcal{U}(S)$ necessarily fulfills $g(x) = g(S(x))$ for all $x \in \mathbb{F}_2^n$. Now let $g \in \mathcal{U}(S)$ and $c \in \text{LS}(g) \cap \text{Im}(S + \text{id}_n)$. Then there exists an $x_0 \in \mathbb{F}_2^n$ such that $S(x_0) = x_0 + c$. We then deduce that $g(x_0) = g(S(x_0)) = g(x_0 + c)$, implying that $c$ is a 0-linear structure of $g$. 

Therefore, if we find enough of these $c \in \mathcal{W}_L(D) \cap \text{Im}(S + \text{id}_n)$, we can just apply Algorithm 6.1 on the resulting set. This approach will be used on Mantis, as explained next.

6.4.3 Results for some Lightweight Ciphers

We now apply the techniques explained above to some existing lightweight designs.

Prince

For Prince, we apply the first approach to $D' = \{d + L(d) \mid d \in D\}$ where

$$D = \{\alpha, c_1 + c_2, c_1 + c_3, c_1 + c_4, c_1 + c_5\}.$$  

Then, $\dim \mathcal{W}_L(D') = 51$. We run Algorithm 6.1 on $\mathcal{W}_L(D')$ and the algorithm terminates within a few minutes on a standard PC. We now have proven that there are no non-trivial invariants that are invariant for both the substitution layer and the linear parts of all rounds in Prince.

Mantis

Since $\dim \mathcal{W}_L(D) = 42$ for Mantis, applying our algorithm 27 times on a subspace of codimension 23 is a quite expensive task. We therefore exploit Proposition 6.6. Let $S_b$ denote the 4-bit S-box of Mantis as given in Table 2.2. Indeed, the S-box layer of Mantis is the parallel application of $S_b$.

The S-box layer has an odd cycle because $S_b$ has a fixed point. Moreover, the image set of $(S_b + \text{id}_4)$ is composed of 7 values $\{0, 9, \text{A, B, C, E, F}\}$. The $c \in \mathcal{W}_L(D)$ for which each 4-bit word is equal to a value in $\text{Im}(S_b + \text{id}_4)$ is a 0-linear structure. For a random value $c \in \mathbb{F}_2^{34}$, we expect that every 4-bit word belongs to $\text{Im}(S_b + \text{id}_4)$ with a probability of $(\frac{7}{16})^{16} \approx 2^{-19.082}$. In fact, one can find enough such $c \in \mathcal{W}_L(D)$ in a reasonable time that generate the whole invariant space $\mathcal{W}_L(D)$, implying that $\mathcal{W}_L(D) \subseteq \text{LS}_0(g)$ for all invariants $g \in \mathcal{U}(S)$ with
$WL(D) \subseteq LS(g)$. We then run Algorithm 6.1 on $Z = WL(D)$. The algorithm terminates and we therefore deduce the non-existence of any non-trivial invariant which is invariant for $S$ and the linear parts of all rounds in Mantis7 (where the tweak is assumed to be zero).

Midori-64

For Midori-64, $WL(D) = \{(0,0,0,0),(0,0,0,1)\}$ and has dimension 16 only. Then, there are $2^{48}$ different cosets of $WL(D)$, implying that our algorithm is not efficient. Indeed, the cipher consists of a significant space of weak keys as shown in [GJN+16, TLS16] (see also Example 6.1).

6.5 Design Criteria on the Linear Layer and on the Round Constants

In this section, we study the properties of $WL(D)$ in more detail and explain the different behaviors which have been previously observed. Most notably, we would like to determine whether the differences in the dimensions of $WL(D)$ we noticed are due to a bad choice of the round constants or if they are inherent to the choice of the linear layer. At this aim, we analyze the possible values for the dimension of $WL(D)$ from a more theoretical viewpoint. We first consider the $L$-invariant subspace $WL(c)$ generated by a single element $c$.

6.5.1 The Possible Dimensions of the $L$-Invariant Subspaces

We show that, for a single element $c$, the dimension of $WL(c)$ is upper-bounded by the degree of the minimal polynomial of the linear layer. Recall that, for a linear invertible mapping $L : F_n^2 \rightarrow F_n^2$, the minimal polynomial of $L$ is defined as the monic polynomial $m_L \in F_2[X]$ of smallest degree such that $m_L(L) = 0$. Moreover, we have to consider the minimal polynomial with respect to single elements in $F_n^2$ as defined as follows.

Definition 6.4 (e.g., page 176 of [Gan59]). The minimal annihilating polynomial of an element $c \in F_n^2$ (w.r.t $L$) (aka the order polynomial of $c$ or simply the minimal polynomial of $c$) is the monic polynomial $ord_L(c) = \sum_{i=0}^d \pi_i X^i \in F_2[X]$ of smallest degree such that $ord_L(c)(L)(c) = 0$.

Proposition 6.7. Let $L$ be a linear permutation of $F_2^n$. For any non-zero $c \in F_2^n$, the dimension of $WL(c)$ is the degree of the minimal polynomial of $c$. 

147
Proof. We know from Lemma 6.1 that $W_L(c)$ is spanned by all $L^i(c), i \geq 0$. Let $d$ be the smallest integer such that \{c, L(c), \ldots, L^{d-1}(c)\} are linearly independent. By definition, $d$ corresponds the degree of the minimal polynomial of $c$ since the fact that $L^d(c)$ belongs to span\{L^i(c) | 0 \leq i < d\} is equivalent to the existence of $\pi_0, \ldots, \pi_{d-1} \in \mathbb{F}_2$ such that $L^d(c) = \sum_{i=0}^{d-1} \pi_i L^i(c)$, i.e., $P(L)(c) = 0$ with $P = X^d + \sum_{i=0}^{d-1} \pi_i X^i \in \mathbb{F}_2[X]$. It follows that $d \leq \dim W_L(c)$.

We now need to prove that $d = \dim W_L(c)$, i.e., that all $L^{d+t}(c)$ for $t \geq 0$ belong to the linear subspace spanned by \{c, L(c), \ldots, L^{d-1}(c)\}. This can be proven by induction on $t$. The property holds for $t = 0$ by definition of $d$. Suppose now that $L^{d+t}(c) \in \text{span}\{c, L(c), \ldots, L^{d-1}(c)\}$. Then,

$$
L^{d+t+1}(c) = L \left( L^{d+t}(c) \right) = L \left( \sum_{i=0}^{d-1} \lambda_i L^i(c) \right) = \sum_{i=0}^{d-1} \lambda_i L^{i+1}(c) \in \text{span}\{c, \ldots, L^{d-1}(c)\}.
$$

\[ \square \]

Obviously, the minimal polynomial of $c$ is a divisor of the minimal polynomial of $L$. The previous proposition then provides an upper bound on the dimension of any subspace $W_L(c)$, for $c \in \mathbb{F}_2^n \setminus \{0\}$.

**Corollary 6.1.** Let $L$ be a linear permutation of $\mathbb{F}_2^n$. For any $c \in \mathbb{F}_2^n$, the dimension of $W_L(c)$ is at most the degree of the minimal polynomial of $L$.

We can even get a more precise result and show that the possible values for the dimension of $W_L(c)$ correspond to the degrees of the divisors of $m_L$. Moreover, there are some elements $c$ which lead to any of these values. In particular, the degree of $m_L$ can always be achieved. This result can be proven in a constructive way by using the representation of the associated matrix as a block diagonal matrix whose diagonal consists of companion matrices.

Let us first focus on the special case when the minimal polynomial of $L$ has degree $n$. It is well known that, in this case, there is a basis such that the matrix of $L$ is the companion matrix of $m_L$ (e.g., [Her75, Lemma 6.7.1]). Using this property, we can prove the following proposition.

**Proposition 6.8.** Let $L$ be a linear permutation of $\mathbb{F}_2^n$ corresponding to the multiplication by some companion matrix $C_Q$ with $Q \in \mathbb{F}_2[X]$ of degree $n$. For any non-constant divisor $P$ of $Q$ in $\mathbb{F}_2[X]$, there exists $c \in \mathbb{F}_2^n$ such that ord$_L(c) = P$.

**Proof.** When the matrix of the linear permutation we consider is a companion matrix $C_Q$, then the elements \{c^T, c^T C_Q, c^T C_Q^2, \ldots\}, can be seen as the successive internal states of the $n$-bit LFSR with characteristic polynomial $Q$ and initial state $c$ (see [LN94, Lemma 6.12 and p. 195]). It follows that ord$_L(c)$ corresponds
to the minimal polynomial of the sequence produced by the LFSR with characteristic polynomial $Q$ and initial state $c$ (see [LN94 Theorem 6.51]). On the other hand, it is well-known that there is a one-to-one correspondence between the sequences $(s_t)_{t \geq 0}$ produced by the LFSR with characteristic polynomial $Q$ and the set of polynomials $C \in \mathbb{F}_2[X]$ with $\deg C < \deg Q$ [LN94 Theorem 6.40]. This comes from the fact that the generating function of any LFSR sequence can be written as

$$\sum_{t \geq 0} s_t X^t = \frac{C(X)}{Q^*(X)} ,$$

where $Q^*$ denotes the reciprocal of polynomial $Q$, i.e., $Q^*(X) = X^{\deg Q} Q(1/X)$, and $C$ is defined by the LFSR initial state.

Let now $P$ be any non-constant divisor of $Q$, i.e., $Q(X) = P(X)R(X)$ with $P \neq 1$. Then, the reciprocal polynomials satisfy $Q^*(X) = P^*(X)R^*(X)$. It follows that, for $C(X) = R^*(X)$,

$$\frac{C(X)}{Q^*(X)} = \frac{1}{P^*(X)} .$$

Therefore, the sequence generated from the initial state defined by $C = R^*$ has minimal polynomial $P$. This is equivalent to the fact that the order polynomial of this initial state equals $P$.

When the degree of the minimal polynomial of the linear layer is smaller than the block size, the previous result can be generalized by representing $L$ by a block diagonal matrix whose diagonal is composed of companion matrices. It leads to the following general result on the possible dimensions of $W_L(c)$.

**Proposition 6.9.** Let $L$ be a linear permutation of $\mathbb{F}_2^n$ and $m_L$ be its minimal polynomial. Then, for any divisor $P$ of $m_L$, there exists $c \in \mathbb{F}_2^n$ such that

$$\dim W_L(c) = \deg P .$$

Most notably,

$$\max_{c \in \mathbb{F}_2^n} \dim W_L(c) = \deg m_L .$$

**Proof.** If $P$ equals the constant polynomial of degree zero, i.e., $P = 1$, we choose $c = 0$. Therefore, we assume in the following that $P$ is of positive degree.

Let us factor the minimal polynomial of $L$ in

$$m_L(X) = M_1(X)^{e_1} M_2(X)^{e_2} \cdots M_k(X)^{e_k}$$

where $M_1, \ldots, M_k$ are distinct irreducible polynomials over $\mathbb{F}_2$. From Theorem 6.7.1 and its corollary in [Her75], $\mathbb{F}_2^n$ can be decomposed into a direct sum of $L$-invariant subspaces

$$\mathbb{F}_2^n = \bigoplus_{i=1}^k \bigoplus_{j=1}^{r_i} V_{i,j}$$

such that the matrix of the linear transformation induced by $L$ on $V_{i,j}$ is the companion matrix of $M_i^{\ell_{i,j}}$ where the $\ell_{i,j}$ are integers such that $\ell_{i,1} = e_i$ (the
polynomials $M_{i}^{e_{i}}$ are called the elementary divisors of $L$). Let now $P$ be a non-constant divisor of $m_{L}$. Thus, we assume w.l.o.g. that

$$P(X) = M_{1}(X)^{a_{1}}M_{2}(X)^{a_{2}}\ldots M_{n}(X)^{a_{n}} \text{ with } 1 \leq a_{i} \leq e_{i}.$$  

Since each $M_{i}^{a_{i}}$ is a non-constant divisor of $M_{i}^{e_{i}}$, we know from Proposition 6.8 that there exists $u_{i} \in V_{i,1}$ such that $\text{ord}_{L_{i}}(u_{i}) = M_{i}^{a_{i}}$, where $L_{i}$ denotes the linear transformation induced by $L$ on $V_{i,1}$. Let us now consider the element $c \in \bigoplus_{i=1}^{\kappa} V_{i,1}$ defined by

$$c = \sum_{i=1}^{\kappa} u_{i}.$$  

Let $\pi_{0}, \ldots, \pi_{d-1} \in \mathbb{F}_{2}$ such that $R(X) := X^{d} + \sum_{t=0}^{d-1} \pi_{t}X^{t}$ equals the order polynomial of $c$. In particular,

$$L_{d}(c) = \sum_{t=0}^{d-1} \pi_{t}L_{t}(c).$$  

Using that $L_{t}(c) = \sum_{i=1}^{\kappa} L_{t}(u_{i})$ and the direct sum property, we deduce that, for any $1 \leq i \leq \kappa,$

$$L_{d}(u_{i}) = \sum_{t=0}^{d-1} \pi_{t}L_{t}(u_{i}).$$  

Then, $R$ is a multiple of the order polynomials of all $u_{i}$. It follows that $R$ must be a multiple of $\text{lcm}(M_{1}^{a_{1}}, \ldots, M_{\kappa}^{a_{\kappa}}) = P$. Since $P(L(c)) = 0$, we deduce that the order polynomial of $c$ is equal to $P$.  

LED

The minimal polynomial of the linear layer in LED is

$$m_{L} = (X^{8} + X^{7} + X^{5} + X^{3} + 1)^{4}(X^{8} + X^{7} + X^{6} + X^{5} + X^{2} + X + 1)^{4} \in \mathbb{F}_{2}[X].$$  

Since its degree equals the block size, we deduce from the previous proposition that there exists an element $c \in \mathbb{F}_{64}^{2}$ such that $W_{L}(c)$ covers the whole space.

Skinny

The linear layer in Skinny with a 16s-bit state, $s \in \{4, 8\}$, is an $\mathbb{F}_{2s}$-linear permutation of $(\mathbb{F}_{2s})^{16}$ defined by a $16 \times 16$ matrix $M$ with coefficients in $\mathbb{F}_{2}$. Moreover, the multiplicative order of this matrix in $GL_{16}(\mathbb{F}_{2})$ equals 16, implying that the minimal polynomial of $L$ is a divisor of $X^{16} + 1$. It can actually be checked that $(M + \text{id}_{16})^{e} \neq 0$ for all $e < 16$, implying that

$$m_{L} = X^{16} + 1 = (X + 1)^{16} \in \mathbb{F}_{2}[X].$$  

It follows that there exist some elements $c \in (\mathbb{F}_{2s})^{16}$ such that $\dim W_{L}(c) = d$ for any value of $d$ between 1 and 16. Elements $c$ which generate a subspace $W_{L}(c)$ of given dimension can be easily exhibited using the construction detailed in the proof of Proposition 6.8. Indeed, up to a change of basis, the matrix of $L$ in $GL_{16}(\mathbb{F}_{2})$
corresponds to the companion matrix of \((X^{16} + 1)\), i.e., to a mere rotation of 16-bit vectors. In other words, we can find a matrix \(U \in \text{GL}_{16}(\mathbb{F}_2)\) such that \(M = U \times C_{X^{16}+1} \times U^{-1}\). Let us now consider elements \(c \in (\mathbb{F}_2^s)^{16}\) for which only the least significant bits of the cells can take non-zero values. Let \(b\) be the 16-bit vector corresponding to these least significant bits, then \(\dim W_L(c) = d\) where \(d\) is the length of the shortest LFSR generating \(b' = U^{-1}b\).

**Prince**

The minimal polynomial of the linear layer in **Prince** is

\[
m_L = X^{20} + X^{18} + X^{16} + X^{14} + X^{12} + X^8 + X^6 + X^4 + X^2 + 1
\]

\[
= (X^4 + X^3 + X^2 + X + 1)^2(X^2 + X + 1)^4(X + 1)^4 \in \mathbb{F}_2[X].
\]

The maximal dimension of \(W_L(c)\) is then 20 and the factorization of \(m_L\) shows that there exist elements which generate subspaces of much lower dimension.

**Mantis and Midori-64**

**Mantis** and **Midori-64** share the same linear layer, which has minimal polynomial

\[
m_L = (X + 1)^6 \in \mathbb{F}_2[X].
\]

We deduce that \(\dim W_L(c) \leq 6\).

### 6.5.2 Considering More Round Constants

We can now consider more than one round constant and determine the maximum dimension of \(W_L(c_1, \ldots, c_t)\) spanned by \(t\) elements. This value is related to the so-called *invariant factor form* (aka. the *rational canonical form*) of the linear layer, as defined in the following proposition.

**Proposition 6.10** (Invariant factors (see Chapter 12, Theorem 16 of [DF04])).

Let \(L\) be a linear permutation of \(\mathbb{F}_2^n\). A basis of \(\mathbb{F}_2^n\) can be found in which the matrix of \(L\) is of the form

\[
\begin{pmatrix}
C_{Q_r} \\
C_{Q_{r-1}} \\
\vdots \\
C_{Q_1}
\end{pmatrix}
\]

for polynomials \(Q_i\) such that \(Q_r \mid Q_{r-1} \mid \cdots \mid Q_1\). The polynomial \(Q_1\) equals the minimal polynomial of \(L\). In this decomposition, the \(Q_i\) are called the invariant factors of \(L\).
The invariant factors of the linear layer then define the maximal value of 
\( W_L(c_1, \ldots, c_r) \), as stated in Theorem 6.1 which we restate below.

**Theorem 6.1.** Let \( Q_1, \ldots, Q_r \) be the invariant factors of the linear layer \( L \) and let \( t \leq r \). Then

\[
\max_{c_1, \ldots, c_t \in \mathbb{F}_q^n} \dim W_L(c_1, \ldots, c_t) = \sum_{i=1}^t \deg Q_i .
\]

Most notably, the minimal number of elements that must be considered in \( D \) in order to generate a space \( W_L(D) \) of full dimension is equal to the number of invariant factors of the linear layer.

**Proof of Theorem 6.1**

We represent \( L \) in invariant factor form as in Proposition 6.10. We denote by \( V_1, \ldots, V_r \) the invariant subspaces such that \( \mathbb{F}_q^n = \bigoplus_{i=1}^r V_i \) and the linear transformation induced by \( L \) on \( V_i \), denoted \( L|_{V_i} \), is represented by the companion matrix \( C_{Q_i} \). We define \( e_{V_i} \) as the first unit vector in \( V_i \), i.e., \( V_i = \text{span}\{L^k(e_{V_i}) \mid 0 \leq k < \deg Q_i\} \) and \( \text{ord}_{L|_{V_i}}(e_{V_i}) = Q_i \). Using Proposition 6.8 one can prove the following lemma.

**Lemma 6.3.** Let \( t \leq r \). Then

\[
\max_{c_1, \ldots, c_t \in \mathbb{F}_q^n} \dim W_L(c_1, \ldots, c_t) \geq \sum_{i=1}^t \deg Q_i .
\]

**Proof.** We choose \( c_1 = e_{V_1} \) and obtain \( W_L(c_1) = W_{L|_{V_1}}(c_1) = V_1 \). Then \( \dim V_1 = \deg Q_1 \). We now continue with \( L|_{V_2 \oplus \cdots \oplus V_{m}} \) which has minimal polynomial \( Q_2 \) and choose \( c_2 \) accordingly. Iterating this until \( c_t \), we construct \( W_L(c_1, \ldots, c_t) \) as the direct sum \( \bigoplus_{i=1}^t W_L(c_i) \) which has dimension \( \sum_{i=1}^t \deg Q_i \).

In order to prove equality, we need the following two lemmas.

**Lemma 6.4.** Let \( c \in \mathbb{F}_q^n = \bigoplus_{j=1}^r V_j \) be represented as \( c = \sum_{j \in J} u_j \) with \( J \subseteq \{1, \ldots, r\} \) and \( u_j \in V_j \setminus \{0\} \). Then \( W_L(c) \subseteq W_L(\bar{c}) \) with \( \bar{c} := \sum_{j \in J} e_{V_j} \).

**Proof.** Let \( v \in W_L(c) \). Then

\[
v = \sum_{i \in \mathbb{N}} \alpha_i L^i(c) = \sum_{i \in \mathbb{N}} \alpha_i L^i(\sum_{j \in J} u_j) = \sum_{i \in \mathbb{N}} \sum_{j \in J} \alpha_i L^i(u_j) = \sum_{i \in \mathbb{N}} \sum_{j \in J} \alpha_i L^i \left( \sum_{k \in \mathbb{N}} \beta_k L^k \left( e_{V_j} \right) \right) = \sum_{i \in \mathbb{N}} \sum_{j \in J} \sum_{k \in \mathbb{N}} \alpha_i \beta_k L^{i+k}(e_{V_j}) = \sum_{i \in \mathbb{N}} \sum_{k \in \mathbb{N}} \alpha_i \beta_k L^{i+k}(\bar{c}) \in W_L(\bar{c}) .
\]
This implies that for any \( c_1, \ldots, c_t \in \mathbb{F}_2^n \), it is \( W_L(c_1, \ldots, c_t) \subseteq W_L(\tilde{c}_1, \ldots, \tilde{c}_t) \). Thus, we can assume w.l.o.g. that all \( c_i \) are of the form \( \tilde{c}_i = \sum_{j=1}^r \gamma_{ij} e_{V_j} \) with \( \gamma_{ij} \in \mathbb{F}_2 \). Then, to any \( t \)-tuple \( (c_1, \ldots, c_t) \in (\mathbb{F}_2^n)^t \) where each \( c_i \) is of the form described above, we associate a \( t \times t \) matrix \( M_{(c_1, \ldots, c_t)} := [\gamma_{ij}]_{i,j} \) over \( \mathbb{F}_2 \).

**Lemma 6.5.** Let \( (c_1, \ldots, c_t) \in (\mathbb{F}_2^n)^t \) be such that \( c_i = \sum_{j=1}^r \gamma_{ij} e_{V_j} \) and let \( M_{(c'_1, \ldots, c'_t)} \) be any matrix obtained from \( M_{(c_1, \ldots, c_t)} \) by elementary row operations. Then, for \( (c'_1, \ldots, c'_t) \) corresponding to \( M_{(c'_1, \ldots, c'_t)} \), we have

\[
W_L(c'_1, \ldots, c'_t) = W_L(c_1, \ldots, c_t).
\]

**Proof.** For a \( t \times t \) matrix over \( \mathbb{F}_2 \), an elementary row operation is either

(i) a swap of two different rows or

(ii) an addition of one row to another.

Transforming a matrix \( M_{(c_1, \ldots, c_r, \ldots, c_t)} \) by operation (i) results in the matrix \( M_{(c_1, \ldots, e_{V_r}, \ldots, c_t)} \) and obviously \( \sum_{i=1}^t W_L(c_i) \) is commutative.

We therefore only have to show that for two constants \( c_r, c_s \) the equality

\[
W_L(c_r) + W_L(c_s) = W_L(c_r + c_s) + W_L(c_s)
\]

holds. Let \( v \in W_L(c_r) + W_L(c_s) \). Then,

\[
u = \sum_{i \in \mathbb{N}} (\alpha_i L_i^r(c_r) + \beta_i L_i^s(c_s)) = \sum_{i \in \mathbb{N}} (\alpha_i L_i(c_r) + \alpha_i L_i^r(c_s) + \beta_i L_i^s(c_s)) \\
= \sum_{i \in \mathbb{N}} (\alpha_i L_i(c_r + c_s) + (\alpha_i + \beta_i) L_i(c_s)) \in W_L(c_r + c_s) + W_L(c_s).
\]

The other inclusion \( \supseteq \) follows accordingly.

Now, we can prove the main theorem.

**Proof of Theorem 6.4.** The only thing left to show is \( \leq \). Given \( c_1, \ldots, c_t \in \mathbb{F}_2^n \) with \( t \leq r \). By Lemma 6.4, \( W_L(c_1, \ldots, c_t) \subseteq W_L(\tilde{c}_1, \ldots, \tilde{c}_t) \) for appropriate \( \tilde{c}_i = \sum_{j=1}^r \gamma_{ij} e_{V_j} \) with \( \gamma_{ij} \in \mathbb{F}_2 \).

Consider the matrix \( M_{(\tilde{c}_1, \ldots, \tilde{c}_t)} \). Using elementary row operations, one can bring \( M_{(\tilde{c}_1, \ldots, \tilde{c}_t)} \) in reduced row-echelon form \( M_{(\tilde{c}_1, \ldots, \tilde{c}_t)} \). Now, by Lemma 6.5, the \( \tilde{c}_i \) are such that \( W_L(\tilde{c}_1, \ldots, \tilde{c}_t) = W_L(c_1, \ldots, c_t) \) and, most importantly, \( W_L(\tilde{c}_1, \ldots, \tilde{c}_t) = \sum_{i=1}^t W_{L_i|V_{i_1} \oplus \cdots \oplus V_{i_t}}(\tilde{c}_i) \). This is because \( \tilde{c}_i = \sum_{j=1}^r \gamma_{ij} e_{V_j} \) has \( \gamma_{ij} = 0 \) for all \( j < i \).

Since the minimal polynomial of \( L_{|V_{i_1} \oplus \cdots \oplus V_{i_t}} \) equals \( Q_i \), one finally obtains:

\[
\dim W_L(c_1, \ldots, c_t) \leq \dim W_L(\tilde{c}_1, \ldots, \tilde{c}_t) = \dim \sum_{i=1}^t W_{L_i|V_{i_1} \oplus \cdots \oplus V_{i_t}}(\tilde{c}_i) \leq \sum_{i=1}^t \deg Q_i
\]

\( \square \)
Prince

The linear layer of Prince has 8 invariant factors:

\[
Q_1 = Q_2 = m_L = X^{20} + X^{18} + X^{16} + X^{14} + X^{12} + X^8 + X^6 + X^4 + X^2 + 1 \in \mathbb{F}_2[X]
\]

\[
Q_3 = Q_4 = X^8 + X^6 + X^2 + 1 = (X + 1)^4(X^2 + X + 1)^2 \in \mathbb{F}_2[X]
\]

\[
Q_5 = Q_6 = Q_7 = Q_8 = (X + 1)^2 \in \mathbb{F}_2[X]
\]

Then, from any set \( D \) with 5 elements, the maximal dimension we can get for \( W_L(D) \) is 20 + 20 + 8 + 8 + 2 = 58, while we get 56 for the particular \( D \) derived from the effective round constants \( D = \{\alpha, c_1 + c_2, c_1 + c_3, c_1 + c_4, c_1 + c_5\} \). We can then see that the round constants are not optimal, but that we can never achieve the full dimension with the number of rounds used in Prince.

Mantis and Midori-64

The linear layer of Mantis (resp. Midori-64) has 16 invariant factors:

\[
Q_1 = \ldots, Q_8 = (X + 1)^6 \text{ and } Q_9 = \ldots, Q_{16} = (X + 1)^2.
\]

From the set \( D \) of size 7 (resp. 8) obtained from the actual round constants of Mantis7 (resp. Mantis8), we generate a space \( W_L(D) \) of dimension 42 (resp. 48) which is then optimal. We also see that one needs at least 16 round constant differences \( c_1, \ldots, c_{16} \) to cover the whole input space. It is worth noticing that the round constants in Midori are only non-zero on the least significant bit in each cell, implying that \( W_L(D) \) has dimension at most 16. This is the main weakness of Midori-64 with respect to invariant attacks and this explains why the use of the same linear in Mantis does not lead to a similar attack.

The maximal dimension we can reach from a given number of round constants for the linear layers of Prince and of Mantis is then depicted in Fig. 6.1 in the beginning of this chapter.

6.5.3 Choosing Random Round Constants

Often, the round constants of a cipher are chosen randomly. In this section, we want to compute the probability that a set of uniformly random chosen elements \( D \) generates a space \( W_L(D) \) of maximal dimension. Again, we first consider the case of a single constant, i.e., \( D = \{c\} \).

**Proposition 6.11.** Let \( L \) be a linear permutation of \( \mathbb{F}_2^n \). Assume that

\[
m_L(X) = M_1(X)^{e_1} M_2(X)^{e_2} \ldots M_k(X)^{e_k}
\]

154
where \(M_1, \ldots, M_k\) are distinct irreducible polynomials in \(F_2[X]\). Then, the probability for a uniformly chosen \(c \in F_2^n\) to obtain \(\dim W_L(c) = \deg m_L\) is

\[
\text{Prob}_{c \in F_2^n} (\dim W_L(c) = \deg m_L) = \prod_{i=1}^k \left(1 - \frac{1}{2\mu_i \deg M_i} \right),
\]

where \(\mu_i\) is the number of invariant factors of \(L\) which are multiples of \(M_i^{e_i}\).

**Proof.** We use the decomposition based on the elementary divisors, as in the proof of Proposition 6.9. From [Her75, Page 308], \(F_2^n\) can be decomposed into a direct sum

\[
F_2^n = \bigoplus_{i=1}^k \bigoplus_{j=1}^{r_i} V_{i,j}
\]

such that the matrix of the linear transformation induced by \(L\) on \(V_{i,j}\) is the companion matrix of \(M_{i,j}^{e_i}\) where, for each \(i\), the \(\ell_{i,j}, 1 \leq j \leq r_i\), form a decreasing sequence of integers such that \(\ell_{i,1} = e_i\). Then, the minimal polynomial of any element \(u\) in \(V_{i,j}\) is a divisor of \(M_{i,j}^{e_i}\). It follows that, if \(c = \sum_{i=1}^k \sum_{j=1}^{r_i} u_{i,j} \in \bigoplus_{i=1}^k \bigoplus_{j=1}^{r_i} V_{i,j}\), then \(\text{ord}_L(c) = m_L\) if and only if, for any \(i\), there exists an index \(j\) such that \(\text{ord}_L(u_{i,j}) = M_i^{e_i}\). Obviously, this situation can only occur if \(\ell_{i,j} = e_i\). This last condition is equivalent to the fact that \(j \leq \mu_i\), where \(\mu_i = \max\{j \mid \ell_{i,j} = e_i\}\). Using that the invariant factors of \(L\) are related to the decomposition of \(m_L\) by

\[
Q_v = \prod_{i=1}^k M_i^{\ell_{i,v}}
\]

where \(\ell_{i,v} = 0\) if \(v > r_i\), we deduce that \(\mu_i\) is the number of invariant factors \(Q_v\) which are multiples of \(M_i^{e_i}\). Let us now define the event

\[
E_{i,j}: \; \text{ord}_L(u_{i,j}) = M_{i,j}^{e_i}.
\]

Then, we have

\[
\text{Prob}_{c \in F_2^n} (\dim W_L(c) = \deg m_L) = \prod_{i=1}^k \text{Prob} \left( \bigcup_{j=1}^{\mu_i} E_{i,j} \right).
\]

It is important to note that for a fixed \(i\), the probability of the event \(E_{i,j}\) is the same for all \(j\). This probability corresponds to the proportion of polynomials of degree less than \(\deg M_{i,j}^{e_i}\) which are coprime to \(M_{i,j}^{e_i}\). Indeed, as noticed in the proof of Proposition 6.8, there is a correspondence between the elements in \(V_{i,j}\) and the initial states of the LFSR with characteristic polynomial \(M_i^{e_i}  j\). Recall that the number of polynomials coprime to a given polynomial \(P\) is

\[
\phi(P) := |\{f \in F_2[X] \mid \deg f < \deg P, \gcd(f, P) = 1\}|.
\]

155
If $P$ is irreducible, then for any power of $P$ we have $\phi(P^k) = 2^{(k-1)\deg P} (2^{\deg P} - 1)$. We then deduce that

$$\text{Prob}(E_{i,j}) = \frac{\phi(M_i^{\ell_{i,j}})}{2^{\ell_{i,j}\deg M_i}} = \frac{2^{(\ell_{i,j}-1)\deg M_i}(2^{\deg M_i} - 1)}{2^{\ell_{i,j}\deg M_i}} = 1 - \frac{1}{2^{\deg M_i}}.$$  

To compute $\text{Prob} \left( \bigcup_{j=1}^{\mu_i} E_{i,j} \right)$, we use the inclusion-exclusion principle and obtain

$$\text{Prob} \left( \bigcup_{j=1}^{\mu_i} E_{i,j} \right) = \sum_{j=1}^{\mu_i} (-1)^{j-1} \binom{\mu_i}{j} \left(1 - \frac{1}{2^{\deg M_i}}\right)^j = \left(1 - \frac{1}{2^{\mu_i \deg M_i}}\right).$$  

\[\square\]

**LED**

The minimal polynomial of the linear layer in LED is

$$m_L = (X^8 + X^7 + X^5 + X^3 + 1)^4(X^8 + X^7 + X^6 + X^5 + X^2 + X + 1)^4 \in \mathbb{F}_2[X].$$

A single constant $c$ is sufficient to generate the whole space. Since $m_L$ has two irreducible factors, each of degree 8, we get from the previous proposition that the probability that $W_L(c) = \mathbb{F}_2^n$ for a uniformly chosen constant $c$ is

$$\text{Prob} \left( W_L(c) = \mathbb{F}_2^n \right) = (1 - 2^{-8})^2 \approx 0.9922.$$  

**Probability to generate the whole space with several random constants**

One can also give a formula for the probability to get the maximal dimension with $t$ randomly chosen round elements, when $t$ varies. This probability highly depends on the degrees of the irreducible factors of the minimal polynomial of $L$.

**Theorem 6.3.** Let $L$ be a linear permutation of $\mathbb{F}_2^n$. Assume that

$$m_L(X) = M_1(X)^{e_1} M_2(X)^{e_2} \cdots M_k(X)^{e_k}$$

where $M_1, \ldots, M_k$ are distinct irreducible polynomials over $\mathbb{F}_2$. Then, the probability that $W_L(c_1, \cdots, c_t)$ equals $\mathbb{F}_2^n$ is

$$\text{Prob}_{c_1, \ldots, c_t \in \mathbb{F}_2} \left( W_L(c_1, \cdots, c_t) = \mathbb{F}_2^n \right) = \prod_{j=1}^{k} \prod_{i_j=0}^{r_j-1} \left(1 - \frac{1}{2^{(t-i_j) \deg M_j}}\right),$$

where $r_j$ is the number of invariant factors of $L$ which are multiples of $M_j$.

It is worth noticing that, when $t < r$ with $r$ the number of invariant factors, the product equals zero which corresponds to the fact that we need at least $r$ constants to generate the whole space. The proof of this Theorem can be found in the original publication [BCLR17].
Recall that the minimal polynomial of the linear layer in Prince is
\[ m_L = X^{20} + X^{18} + X^{16} + X^{14} + X^{12} + X^8 + X^6 + X^4 + X^2 + 1 \]
\[ = (X^4 + X^3 + X^2 + X + 1)^2 (X^2 + X + 1)^4 (X + 1)^4 \in \mathbb{F}_2[X]. \]
It then has three irreducible factors
\[ M_1 = X^4 + X^3 + X^2 + X + 1, \quad M_2 = X^2 + X + 1 \quad \text{and} \quad M_3 = (X + 1). \]
Moreover, we know that the eight invariant factors of \( L \) are
\[ Q_1 = Q_2 = m_L, \]
\[ Q_3 = Q_4 = (X + 1)^4 (X^2 + X + 1)^2, \]
\[ Q_5 = Q_6 = Q_7 = Q_8 = (X + 1)^2. \]
We then deduce that \( \mu_1 = 2, \mu_2 = 2 \) and \( \mu_3 = 4 \). Proposition 6.11 then implies that \( \dim W_L(c) \leq 20 \) and
\[ \text{Prob} (\dim W_L(c) = 20) = (1 - 2^{-8})(1 - 2^{-4})^2 \approx 0.8755 \]
for a uniformly chosen \( c \). Since \( L \) has 8 invariant factors, at least \( t = 8 \) elements \( c_1, \ldots, c_8 \) are needed to reach \( W_L(c_1, \ldots, c_t) = \mathbb{F}_2^{64} \). The number of invariant factors in which each of the \( M_i \) appears is given by \( r_1 = 2, r_2 = 4 \) and \( r_3 = 8 \). From Theorem 6.3, we get that the probability that \( W_L(c_1, \ldots, c_8) = \mathbb{F}_2^{64} \) is
\[ \prod_{i=0}^{1} \left( 1 - 2^{-8-(8-i)-4} \right) \times \prod_{i=0}^{3} \left( 1 - 2^{-8-(8-i)-2} \right) \prod_{i=0}^{7} \left( 1 - 2^{-8-(8-i)} \right) \approx 0.2895. \]

**Mantis and Midori-64**

The minimal polynomial of the linear layer of Mantis and Midori-64 has a single irreducible factor, which is \( (X + 1) \). This linear layer has 16 invariant factors. Since the first 8 invariant factors equal the minimal polynomial, which has degree 6, we derive from Proposition 6.11 that the probability that a uniformly chosen element generates a subspace of dimension 6 is
\[ \text{Prob} (\dim W_L(c) = 6) = (1 - 2^{-8}) \approx 0.9961. \]
We need at least 16 elements \( c_1, \ldots, c_{16} \) to cover the whole space and this occurs with probability
\[ \prod_{j=1}^{16} \left( 1 - \frac{1}{2^7} \right) \approx 0.28879. \]
It is worth noticing that when we increase the number of random round constants from 16 to 20, this probability increases to 0.93879.
Figure 6.2 in the beginning of this chapter shows how the probability that the whole space is covered increases with the number of randomly chosen elements, for the linear layers of LED, Skinny-64, Prince and Mantis. The fact that the curve corresponding to Skinny-64, Prince and Mantis have a similar shape comes from the fact that all three linear layers have a minimal polynomial divisible by \((X + 1)\), and this factor appears in all invariant factors. Then, the term corresponding to the irreducible factor of degree 1, namely
\[
\prod_{j=t-r+1}^{t} \left(1 - \frac{1}{2^j}\right)
\]
is the dominant term in the formula in Theorem 6.3. Most notably, for \(t = r\), the probability is close to \((1 - 2^{-1})(1 - 2^{-2})(1 - 2^{-3})(1 - 2^{-4}) \simeq 0.3\).

6.6 Conclusion and Open Problems

For lightweight substitution-permutation ciphers with a simple key scheduling, we provided a detailed analysis on the impact of the design of the linear layer and the particular choice of the round constants to the applicability of both the invariant subspace attack and the more recently published nonlinear invariant attack. With an algorithmic approach, a designer is now able to easily check the soundness of the chosen round constants, in combination with the choice of the linear layer, with regard to the resistance against invariant attacks and can thus easily avoid possible weaknesses by design. We stress that in many cases, this analysis can be done independently of the choice of the substitution layer. We directly applied our methods to several existing lightweight ciphers and showed in particular why Skinny-64-64, Prince, and Mantis\textsuperscript{7} are secure against invariant attacks; unless the adversary exploits weaknesses which are not based on weaknesses of the underlying building blocks, i.e., substitution layer and linear layer. In fact, we are not aware of any such strong attacks in the literature.

As future work, one can think about further generalizations of invariant attacks. As it was already mentioned in [TLS16], it would be interesting to know if one can make use of statistical invariant attacks, i.e., invariant attacks that only work with a certain probability instead for all possible plaintexts. In other words, it is possible to utilize nonlinear approximations that hold with an absolute correlation less than 1? Further, it would be nice to study invariant attacks under more complex key schedules.

We have also seen a relation between a quadratic invariant \(g\) for an instance \(E_k\) and the existence of high-biased linear approximations over \(E_k\). For future work, one could try to understand more about this relation. As a starting point, one could try to analyze the distribution of the correlations \(\text{cor}_{E_k}(\gamma, \gamma')\) over all \(\gamma, \gamma' \in \Gamma_g\). There might also be a link to the observations made in [LAAZ11, Section 4.3] and [AABL12, Section 5].
Listing 6.1: Sage code for proving the non-applicability of the invariant attack.

```python
from sage.geometry.hyperplane_arrangement.affine_subspace import AffineSubspace

# converts an integer to a binary vector.
def to_binary_vector(a, length):
    ls = Integer(a).bits()[::-1]
    return vector(GF(2), length, [0]*(length-len(ls))+ls)

# Evaluates the S-box layer with S-box Sb on vector v
def sbox_layer_eval(Sb, bit_Sb, v):
    w = copy(v)
    for i in range(len(w)/bit_Sb):
        w[(i*bit_Sb):((i+1)*bit_Sb)] =
        list(to_binary_vector(Sb[ZZ(list(w[(i*bit_Sb):((i+1)*bit_Sb)][::-1]), base = 2)], bit_Sb))
    return w

# returns complement C of V such that C.intersection(V) is trivial
def decomposition_complement(V):
    L1 = list(V.basis())
    L2 = list(V.ambient_vector_space().basis())
    R = []
    for v in L2:
        if (v not in span(L1)):
            L1.append(v)
            R.append(v)
    return span(R)

# input: list of differences D, linear layer L as a matrix
# output: the subspace W_L(D)
def W_space(D, L):
    R = []
    for c in D:
        for j in range(L.multiplicative_order()):
            R.append((L**j)*c)
    return span(R)

# input: S-box S, subspace Z of W_L(D)
# if true, the constants prevent invariant attacks
def check_with_sbox(S, Z):
    bit_S = int(log(len(S),2))

    # define 0 + Z as an affine space and choose a complement Q of Z
    # Q is isomorphic to (GF(2)^n)/Z and each q in Q is a
element of a different coset q + Z
    A = AffineSubspace(0, Z)
    Q = AffineSubspace(0, decomposition_complement(Z))

    # ls will indicate all cosets "hit" by the S-box layer
    ls = set()
    k = 2**Q.dimension()
    print(repr(k) + ' cosets to check')
    percent_done = 0
```

159
# repeat this until each coset is hit

```python
while (len(ls) < k):
    a = A.linear_part().random_element() + A.point()
    b = sbox_layer_eval(S, bit_S, a)
    # q gives the unique representative of the coset in Q
    q = Q.intersection(AffineSubspace(b, Z)).point()
    # add integer representation of q in the set of cosets hit.
    ls.add(ZZ(list(q), base=2))
    if (len(ls)/k >= (percent_done+1)/100):
        percent_done = percent_done + 1
        print(repr(percent_done) + ' % done')
```

return true
Chapter 7

Differential Trails in Simon-like Ciphers

This chapter is a revised version of the author’s publication [Bei16].

7.1 Introduction

Once a new cipher is proposed, the designers are expected to provide security arguments on the resistance differential and linear attacks. In particular, any new design itself should allow for an, if possible simple, security argument. In SP ciphers, the separation into linear and non-linear components offers the advantage of analyzing the structure more easily. As we have already introduced in Section 2.4.2 and Section 2.4.3, two principles are common. Firstly, one can try to obtain theoretical bounds on the minimum number of active S-boxes according to the wide-trail strategy (see Theorem 2.4), or secondly, one employs computer-aided methods. The advantage of having provable bounds on the minimum number of active S-boxes is one reason why so many AES-like designs occurred over the last years. It also emphasizes that designers prefer well-understood principles. While for AES-like ciphers counting the number of active S-boxes can be somehow done independently of the choice of the S-box, some other strategies use specific properties of the non-linear components. For instance, the designers of Present showed that any valid five-round differential trail has at least 10 active S-boxes, using properties of the actual S-boxes [BKL+07].

The other strategy is evaluating the security using computer-aided search methods. For instance, one can model the propagation of differential and linear trails as a mixed-integer linear programming instance [MWGP12, SHW+14b, BJL+15].

1The original article published by Springer-Verlag is available at DOI: 10.1007/978-3-319-44618-9_23 (© Springer International Publishing Switzerland 2016).
Examples of a design which uses experimental arguments are the hash function Keccak [BDPA11], the block cipher Serpent [BAK98], and also the AES-like designs Skinny and Mantis (Chapter 5). However, the bounds obtained with this approach, although very useful, are not verifiable without a machine and do not contribute significantly to a better understanding of the design itself.

Basically, in both strategies, (if the non-linear component is not too weak) the design of the linear layer is the crucial step when it comes to providing security against differential and linear attacks. While a single round can often be analyzed quite easily, the analysis of the linear layer usually has to be done using a more complex argument over multiple rounds. Unfortunately, not many constructions are known that allow to prove security using arguments that are verifiable by hand. One therefore may seek for alternative design principles. Especially for lots of Feistel designs, the constructions might be less clear and less understood. However, there are some fundamental results on bounding the differential and linear behavior [NK95]. There are also Feistel designs which consist of SP-type round functions [SP04, SS04] combining the advantages of the Feistel construction and the simple arguments of the wide-trail strategy.

In contrast to a scientific design process, the NSA presented the Simon family of lightweight block ciphers [BSS+13]. Besides its specification, no arguments on the security are provided. Especially since Simon is an innovative Feistel cipher, its design is harder to analyze. Besides its non-bijective round function and the combining of the branches after every round, the difficulties are caused by the bit-wise structure. Since the design choices were left unclear in the first place, one seeks for a deeper understanding of the cipher.

Related Work

The appearance of the Simon family of block ciphers inspired the cryptographic community taking further investigations on the possible design rationale. Therefore, several cryptanalytic results followed. For instance, see [ABG+13, AL13, WLV+14, WWJZ14, AAA+15, ALLW15, Ash15, BRV15, CW16, KSI16, TM16] for a selection. They are mostly based on experimental search. No significant weaknesses have been found so far and Simon still offers a reasonable security margin based on existing cryptanalysis.

In [KLT15], Kölbl, Leander and Tiessen pointed out some interesting properties of Simon-like round functions. Those observations were then used for a further analysis of the differential and linear behavior over multiple rounds. Although the analysis of the round function was done in a mathematical rigorous manner, the multi-round behavior was derived using a computer-aided approach. As one result, the rotation constants of Simon turned out to be in some sense not optimally chosen.

Inspired by the design of Simon, Yang et al. proposed the Simeck family of lightweight block ciphers in [YZS+15]. It can be seen as a Simon-like cipher using different rotation constants in its round function and a key schedule inspired by
At the NIST lightweight workshop in 2015, the designers of Simon presented a follow-up paper covering some considerations with regard to implementation BSS+15.

Very recently, in [LLW17b], Liu, Li and Wang derived an upper bound on the absolute correlation of non-trivial linear trails in Simon and Simeck, adapting our methods to the case of linear cryptanalysis.

Results of this Chapter

After describing a generalization of the Simon design by decoupling the round function into a linear and a non-linear component, we show that the structure of a Simon-like design allows for a proof on the resistance against differential attacks under standard simplifying assumptions. The question whether the proof works depends on the interaction between those two components. If the non-linear part $\rho$ is of the form $\rho(x) = (x \ll a) \land (x \ll b)$, it can be in general formulated as a property of the linear part. A sufficient condition is that the linear part has a branch number (with respect to bits) of at least 11. Since this is not the case for Simon and Simeck, we consider those designs separately. In particular, for all versions of Simon and Simeck, we are able to show that

$$\prod_{i=1}^{t} \text{Prob}(\alpha_{i-1} \xrightarrow{F_{F}} \alpha_i) \leq 2^{-2t+2},$$

where $t$ denotes the number of rounds of the particular cipher, $F_{F}$ the (unkeyed) round of the key-alternating Feistel structure, and $(\alpha_0, \ldots, \alpha_t)$ any non-zero $t$-round differential trail. We show this in detail for the example of Simon. Most importantly, for all versions of Simon and Simeck, the number of rounds $t$ is such that $2^{-2t+2} \leq 2^{-n}$, where $n$ denotes the block length.

In clear distinction to prior work such as [KLT15], our argument is a formal proof covering multiple rounds and can thus be verified without experimental tools. In our approach, we use a well-known property of the Simon-like Feistel function, namely that the set of possible output differences $U_{\alpha}$ defines an affine subspace depending on the input difference $\alpha$ and that the differential probability highly depends on the Hamming weight of $\alpha$. The main idea is that we extend the analysis of the Feistel function to the cases where $\alpha$ has a Hamming weight equal to two and consider the propagation of Hamming weights over the Feistel structure.\footnote{Note that recently, in [LLW17a], Liu, Li and Wang presented an improved result on the differential probability of differences $\alpha$ with Hamming weight less than $\frac{n}{4}$. Their result would allow for some simplifications in the proof of our main result.}

Figure 7.1 illustrates the bounds proven with our method and, as a comparison, the bounds obtained from experimental search described in [KLT15, Section 5.2] for two instances of Simon. It is to mention that, although our bounds are worse
than the experimental results, they are still much better than the bounds one
obtains by trivially multiplying the worst-case probabilities for every round.

Figure 7.1: Comparison of the experimental bounds for Simon-32 and Simon-48
as described in [KLT15, Section 5.2] and our provable bounds.

7.2 Preliminaries

For a slightly more simplified notation in our arguments, throughout this chapter
we will denote the bits within vectors \( x \in \mathbb{F}_2^d \) with indices starting from zero, i.e.,
\( x = (x_0, x_1, \ldots, x_{d-1}) \). We may sometimes also use a superscript notation for
denoting the position of a bit. For example, the element \((0, \ldots, 0, y, 0, 0, \ldots, 0)\)
denotes a vector \((x_0, x_1, \ldots, x_{d-1})\) with \(x_i = y\) and \(x_j = 0\) for all \(j \neq i\). Moreover,
when \(n\) denotes the block length of a cipher, we define \(n' = \frac{n}{2}\).

Argument on the Resistance Against Differential Attacks

When designing a new block cipher, in the ideal case, one would like to avoid the
existence of non-trivial differentials with high probability for all possible keyed
instances. However, since in general, computing the maximum differential proba-
bility of multi-round differentials is a non-trivial task, one adheres to simplifying
assumptions. In most cases, and as outlined in detail in Section 2.3.1, one con-
centrates on upper bounding the product of the differential probabilities over the
rounds contained in any non-zero differential trail. In particular, if one is going to
design a key-alternating cipher as depicted in Figure 2.2, a typical approach is to
estimate the number of rounds \(t\) such that \( \prod_{i=1}^{t} \text{Prob}(\alpha_i \rightarrow R_{i} \alpha_i) < 2^{-n} \) for any
non-zero \( t \)-round differential trail \((\alpha_0, \alpha_1, \ldots, \alpha_t)\) and finally specify the number of rounds of the primitive as \( t + t_m \), where \( t_m \) defines a reasonable security margin. This provides a sound argument on the resistance against differential attacks under the hypothesis of stochastic equivalence (i.e., Assumption 2.1), the assumption of independent round keys (i.e., Assumption 2.2), and the assumption that the value of the sum over the probabilities given in Equation 2.3 is dominated by a single differential trail.

For the key-alternating Feistel cipher Simon, whereas the unkeyed Feistel round is denoted by \( F^{fS} \) and the block length denoted by \( n \), we show that

\[
\prod_{i=1}^{t} \text{Prob}(\alpha_{i-1} \xrightarrow{F^{fS}} \alpha_{i}) < 2^{-n}
\]

for any non-zero \( t \)-round differential trail \((\alpha_0, \alpha_1, \ldots, \alpha_t)\), where \( t \) is strictly smaller than the specified number of rounds, still leaving a reasonable security margin. Note that our security argument still implicitly assumes independent round keys, although the key is only added to one half of the \( n \)-bit state.

A Remark on the Key-Alternating Feistel Construction

In Section 2.2.3, we have explained the notion of a key-alternating Feistel structure. Recall that, for a vectorial Boolean function \( f : \mathbb{F}_2^{n'} \rightarrow \mathbb{F}_2^{n'} \) and \( k \in \mathbb{F}_2^{n'} \), we have given the keyed instance of the round as

\[
F^f_k : \mathbb{F}_2^{n'} \times \mathbb{F}_2^{n'} \rightarrow \mathbb{F}_2^{n'} \times \mathbb{F}_2^{n'}
\]

\[
(x_l, x_r) \mapsto (f(x_l) + x_r + k, x_l).
\]

Thereby, \( f \) is called the Feistel function (or simply \( f \)-function) and \( k \) is the round key. For simplicity, we will consider an identical Feistel function \( f \) in every round. With regard to the key-alternating structure, we define \( F^I := F^0 \) as the unkeyed round of the cipher. The application of those unkeyed rounds is then interleaved by the round key addition, where the left half of each round key is always equal to zero.

We are going to denote a difference within the Feistel structure as a pair \((\gamma, \delta) \in \mathbb{F}_2^{n'} \times \mathbb{F}_2^{n'}\), describing the differences in the left and the right branch, respectively. For a \( t \)-round differential trail \( C = ((\gamma_0, \delta_0), (\gamma_1, \delta_1), \ldots, (\gamma_t, \delta_t)) \) over (unkeyed) Feistel rounds \( F^I \), we define

\[
P(C) := \prod_{i=1}^{t} \text{Prob}(\gamma_{i-1} \xrightarrow{f} \gamma_{i} + \delta_{i-1}).
\]

It is straightforward to deduce that

\[
\prod_{i=1}^{t} \text{Prob}((\gamma_{i-1}, \delta_{i-1}) \xrightarrow{F^I} (\gamma_i, \delta_i)) = \begin{cases} P(C) & \text{if } \forall i \in \mathbb{N}_{\leq t} : \gamma_{i-1} = \delta_i \\ 0 & \text{else} \end{cases}.
\]

165
Lemma 7.1 presents a general observation on the Feistel construction. It states that, having upper bounds on the value of $P(C)$ for all differential trails $C$ starting with $(0, \alpha)$ and ending with $(0, \beta)$, one can easily upper bound the value of $P(C)$ of any differential trail $C$.

**Lemma 7.1.** For $t \geq 1$, let for all non-zero differences $\alpha, \beta \in \mathbb{F}_2^n$, the value of $P(C)$ of any $t$-round trail $C$ (over round functions $F^f$) starting with $(0, \alpha)$ and ending with $(0, \beta)$, be upper bounded by $p(t)$.

Let further $p(0) := 1$ and $q := \max_{\alpha \neq 0, \beta} \text{Prob}(\alpha \xrightarrow{f} \beta)$. Then, for any $T > 0$ and any non-zero $T$-round trail $C$ over round functions $F^f$, it is

$$P(C) \leq \max \left\{ \max_{l \in \mathbb{N}_0} \{p(l)q^{T-l-1}\}, p(T) \right\}.$$ 

**Proof.** Let us fix a non-zero $T$-round trail $C = (\gamma_0, \delta_0) \xrightarrow{f} (\gamma_1, \delta_1) \xrightarrow{f} \ldots \xrightarrow{f} (\gamma_T, \delta_T)$. The proof is now split into two cases.

(i) Let us assume that there exist distinct $i, j$ such that $\gamma_i = \gamma_j = 0$. Then, w.l.o.g., one can choose two distinct indices $i', j'$ such that $\gamma_{i'} = \gamma_{j'} = 0$ and $\gamma_l \neq 0$ for all $l < i'$ and all $l > j'$. If $(i', j') = (0,T)$, we already have $P(C) \leq p(T)$ by definition. Thus, let $(i', j') \neq (0,T)$. Now, by definition

$$P((\gamma_{i'}, \delta_{i'}), \ldots, (\gamma_{j'}, \delta_{j'})) \leq p(j' - i').$$ 

Since $\gamma_{j'} = 0$ and, for all $l < i'$ and $l > j'$, $\gamma_l \neq 0$, we have

$$P(C) \leq p(j' - i') \prod_{l=0}^{i'-1} \text{Prob}(\gamma_l \xrightarrow{f} \gamma_{l+1} + \delta_l) \prod_{l=j'+1}^{T-1} \text{Prob}(\gamma_l \xrightarrow{f} \gamma_{l+1} + \delta_l)$$ 

$$\leq p(j' - i')q^{T-(j'+1)} = p(j' - i')q^{T-(j'-i')-1}.$$ 

and $j' - i' < T$.

(ii) If $\gamma_i = 0$ for at most one $i$, then

$$\prod_{l=0}^{T-1} \text{Prob}(\gamma_l \xrightarrow{f} \gamma_{l+1} + \delta_l) \leq \prod_{l \in \mathbb{N}_0} \text{Prob}(\gamma_l \xrightarrow{f} \gamma_{l+1} + \delta_l) \leq q^{T-1} = p(0)q^{T-1}.$$ 

As Lemma 7.1 states a general property for all Feistel structures, we give a simplified version in Section 7.3 as Corollary 7.1. It covers the special case of the round function in Simon and a Simon-like round function, which will be defined next.

166
Simon and Simon-like Ciphers

We generalize the design of the Simon block cipher to the Simon-like structure. Figure 7.2 illustrates this construction. For the Simon-like design, one requires a rotational invariant function of algebraic degree two as the non-linear component. A vectorial function \( f: \mathbb{F}_2^d \rightarrow \mathbb{F}_2^d \) is called rotational invariant if \( f(x \ll r) = (f(x) \ll r) \) for all \( x \in \mathbb{F}_2^d \) and all offsets \( r \). This leads to the following definition.

**Definition 7.1.** A Simon-like \( f \)-function \( f_S \) is composed of an \( \mathbb{F}_2 \)-linear function \( \theta: \mathbb{F}_2^n' \rightarrow \mathbb{F}_2^n' \) and a function \( \rho: \mathbb{F}_2^n' \rightarrow \mathbb{F}_2^n' \) of algebraic degree two of the form \( \rho(x) = \vartheta_1(x) \land \vartheta_2(x) \), with \( \mathbb{F}_2 \)-linear and rotational invariant \( \vartheta_i: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n \), as

\[
\begin{align*}
f_S: x & \mapsto \rho(x) + \theta(x) .
\end{align*}
\]

In this context, a Simon-like cipher employs such an \( f \)-function in a key-alternating Feistel construction.

Note that the rotational invariance is, in this general case, not required for the linear part \( \theta \).

![Figure 7.2: Illustration of the Simon round function and the generalized Simon-like round function.](image)

7.3 Analysis of Differential Trails

In this section, we analyze the propagation characteristics of differences over several rounds. We rely on the fact that a single Simon-like round is quite well understood. Let

\[
L_\alpha(x) := (\vartheta_1(x) \land \vartheta_2(\alpha)) + (\vartheta_1(\alpha) \land \vartheta_2(x)).
\]

We first recall the observation that for any input difference \( \alpha \in \mathbb{F}_2^n' \) into a Simon-like round function \( f_S \), a possible output difference lies in the affine subspace \( U_\alpha := \text{Im} L_\alpha + f_S(\alpha) \). The main reason is that \( f_S \) has algebraic degree two. This is formally stated in Proposition 7.1.
Proposition 7.1 (Kölbl, Leander, Tiessen [KLT15]). Let $f_S$ be a Simon-like $f$-function. For an input difference $\alpha \in \mathbb{F}_2^{n'}$ into $f_S$, the set of possible output differences defines an affine subspace $U_\alpha$ such that $\operatorname{Prob}(\alpha \xrightarrow{f_S} \beta) \neq 0$ if and only if $\beta \in U_\alpha$. Defining $d_\alpha := \dim \operatorname{Im} L_\alpha$, it holds that
\[
\beta \in U_\alpha \iff \beta + f_S(\alpha) \in \operatorname{Im} L_\alpha
\]
and $\operatorname{Prob}(\alpha \xrightarrow{f_S} \beta) = 2^{-d_\alpha}$ for all valid differentials (i.e., differentials with non-zero differential probability) over $f_S$.

Since the probability is the same for all output differences $\beta$ in this subspace, we simply write $p_\alpha$ for $\operatorname{Prob}(\alpha \xrightarrow{f_S} \beta)$ with $\beta \in U_\alpha$. For all output differences which are not elements in this subspace, the differential probability is equal to zero.

Because of the rotational invariance, it holds that $\operatorname{Im} L_{(\alpha \ll r)} = (\operatorname{Im} L_\alpha \ll r)$ with $p_{(\alpha \ll r)} = p_\alpha$. One can thus restrict the consideration to a single representative of this equivalence class if only one round function is analyzed.

### 7.3.1 Restriction to Rotations as Rotational Invariant Functions

From now on, we consider $\vartheta_1, \vartheta_2$ of the form $\vartheta_1(x) = (x \ll a)$ and $\vartheta_2(x) = (x \ll b)$, respectively. This describes the most simple structure of the generalized Simon-like cipher. For the $\theta$ step defined as $\theta(x) = (x \ll c)$, one obtains Simon and Simeck as a special case using $(8, 1, 2)$, resp. $(5, 0, 1)$, as a choice for the rotation constants $(a, b, c)$. The following lemma states that we can obtain an upper bound on the differential probability over $f_S$ depending on the Hamming weight of the input difference. While a weaker version of Lemma 7.2 can be deduced from [KLT15, Theorem 3, p. 9], we improve the bound from [KLT15] if the Hamming weight of the input difference equals 2. Although this improvement seems to be of little importance at a first glance, it is exactly this tighter bound which allows us to prove the main result. Thus, Lemma 7.2, and especially case (ii), is one of the core components in our argument.\(^{3}\)

**Lemma 7.2.** Let $\vartheta_1(x) = (x \ll a)$ and $\vartheta_2(x) = (x \ll b)$. Assume that $n' \geq 6$ is even and $\gcd(a - b, n') = 1$. Let $\alpha = (\alpha_0, \ldots, \alpha_{n'-1}) \in \mathbb{F}_2^{n'}$ be an input difference into $f_S$. Then, for the differential probability over $f_S$, it holds that
\[
\begin{align*}
(i) \quad \text{If } w_1(\alpha) = 1, & \quad \text{then } p_\alpha \leq 2^{-2}. \\
(ii) \quad \text{If } w_1(\alpha) = 2, & \quad \text{then } p_\alpha \leq 2^{-3}. \\
(iii) \quad \text{If } w_1(\alpha) \neq n', & \quad \text{then } p_\alpha \leq 2^{-w_1(\alpha)}. \\
(iv) \quad \text{If } w_1(\alpha) = n', & \quad \text{then } p_\alpha \leq 2^{-n'+1}.
\end{align*}
\]

\(^{3}\)In [LLW17a], Liu, Li and Wang extended case (ii) to all $\alpha$ with $w_1(\alpha) < \frac{n'}{2}$. In particular, they showed that $p_\alpha \leq 2^{-w_1(\alpha)-1}$ in those cases.
Proof. Without loss of generality one can assume that $b = 0$ and $a < \frac{n'}{2}$, $a \neq 0$ because of the rotational invariance and since $a - b$ and $n'$ are coprime. According to [KLT15, Theorem 3, p. 9], it is $p_\alpha = 2^{-d_{\alpha}}$ with

$$d_\alpha = \begin{cases} w_1 \left( ((\alpha \ll a) \lor \alpha) + (\alpha \land (\alpha \ll a)) \land (\alpha \ll 2a) \right) & \text{if } w_1(\alpha) \neq n' \\ n' - 1 & \text{if } w_1(\alpha) = n' \end{cases}$$

Note that $d_\alpha = \dim \text{Im } L_\alpha$, where

$$L_\alpha(x) = ((x \ll a) \land \alpha) + ((\alpha \ll a) \land x).$$

(i), (iii) and (iv) follow directly from the above formula. In order to show (ii), we construct three linearly independent elements in $\text{Im } L_\alpha$.

Let $w_1(\alpha) = 2$ with $\alpha_0 = 0$. W.l.o.g., let $\alpha_i \leq \frac{n'}{2}$, $\alpha_i \neq 0$, since every $\alpha$ with a Hamming weight of two is rotational equivalent to that assumed form. Now, consider the following three elements $x, y, z \in \mathbb{F}_2^n$:

- $x = (0, \ldots, 0, 1(a), 0, \ldots, 0) \Rightarrow L_\alpha(x) = (1(0), 0, \ldots, 0, \alpha_{2a}, 0, \ldots, 0)$
- $y = (0, \ldots, 0, 1(a-i), 0, \ldots, 0) \Rightarrow L_\alpha(y) = (0, \ldots, 1(i), 0, \ldots, 0, \alpha_{i+2a \mod n'}, 0, \ldots, 0)$
- $z = (1, 1, \ldots, 1) \Rightarrow L_\alpha(z) = (\alpha \ll a) + \alpha$.

Clearly, $L_\alpha(x)$ and $L_\alpha(y)$ are linearly independent. To show that $L_\alpha(z) \notin \text{span}\{L_\alpha(x), L_\alpha(y)\}$, we consider the two cases

1. $\alpha_{i+2a \mod n'} = 0$ : Then $L_\alpha(y)i+a = 0$. Since $L_\alpha(z)n'-a = 1$ and $n'-a \notin \{0, i, a\}$, the linear independence follows.

2. $\alpha_{i+2a \mod n'} = 1$ : Then $i + 2a \mod n' \in \{0, i\}$ because of the construction of $\alpha$. However, since $2a \neq 0 \mod n'$, it follows that $i + 2a = 0 \mod n'$. Hence, $2a = n' - i$. Now $2a \neq i$, because otherwise $n' = 4a$ which is contradictory to $\gcd(a, n') = 1$ (since $n' \geq 6$). Thus $L_\alpha(x)_a = 0$. In addition, $i \neq a$ because otherwise $3a = 0 \mod n'$ which is also contradictory to $\gcd(a, n') = 1$. Now, $L_\alpha(z)i+a \mod n' = 1$ and $i + a \notin \{0, i, i + a\}$.

In all cases, we thus have $p_\alpha \leq 2^{-2}$ if $\alpha \neq 0$ and $p_0 = 1$. The interesting property is the fact that $p_\alpha \leq 2^{-w_1(\alpha)-1}$ if $\alpha$ has a Hamming weight of 2. This is what we make use of in the following arguments. The basic idea is to guarantee enough transitions with a probability $\leq 2^{-3}$ before a zero input difference into $f_S$ occurs (then $p_0 = 1$). This allows us to catch up the factor $2^{-2}$ that we lose for the zero input difference. Otherwise, if we were not able to guarantee the tighter bound described in Lemma 7.2 (ii), the input difference into $f_S$ of every second round might be equal to zero in the worst case and our argument would only provide the trivial bound of $2^{-T}$ over $T$ rounds. See also Figure 7.1 for an illustration. For the formal proof, we give Corollary 7.1 at first. It is an implication of Lemma 7.1 for the Simon-like $f$ function.

169
Corollary 7.1. Let, for all non-zero differences $\alpha, \beta$ and all $t \geq 1$, the value of $P(C)$ for any $t$-round differential trail $C$ (over Simon-like Feistel rounds $F^{fs}$) starting with $(0, \alpha)$ and ending with $(0, \beta)$ be upper bounded by $2^{-2t}$. Let further $p_\gamma \leq 2^{-2}$ for all $\gamma \neq 0$. Then, for any non-zero $T$-round trail $C$ over Simon-like rounds $F^{fs}$, with $T > 0$, it is

$$P(C) \leq 2^{-2T+2}.$$  

Proof. With the notation in Lemma 7.1 it is $p(t) = 2^{-2t}$ and $q \leq 2^{-2}$. Thus,

$$P(C) \leq \max \left\{ \max_{l \in \mathbb{N}_0 < T} \{ p(l)q^{T-l-1}, p(T) \} \right\} \leq \max \left\{ \max_{l \in \mathbb{N}_0 < T} \{ 2^{-2(l/2+2l+2)}, 2^{-2T} \} \right\}$$

$$= \max \{ 2^{-2T+2}, 2^{-2T} \} = 2^{-2T+2}. \tag*{$\square$}$$

Thus, in order to prove an upper bound on the value of $P(C)$ of $2^{-2T+2}$ for any non-zero $T$-round differential trail $C$, we only have to concentrate on $t$-round trails of the form $(0, \alpha) \rightarrow \cdots \rightarrow (0, \beta)$ and prove an upper bound of $2^{-2t}$ for all of those. We further can restrict ourselves to the shortest trails of this form, e.g., $\gamma_i \neq 0$ for all intermediate $\gamma_i$. The reason is that one can easily concatenate these short trails to longer ones for which the property holds as well.

We have to do the analysis for a specific choice of the linear mapping $\theta$. As a more general case, Theorem 7.1 formulates a sufficient condition on $\theta$ for our argument to work. It requires that the branch number of $\theta$ (with respect to bits) is high enough.\(^4\)

Theorem 7.1. Let $B_1(\theta) \geq 11$. Then for any distinct $a, b$ and any $n'$ fulfilling the properties of Lemma 7.2, the value of $P(C)$ for any non-zero $T$-round differential trail $C$ over round functions $F^{fs}$ is upper bounded by $2^{-2T+2}$.

Proof. Fix a $t$-round trail of the form

$$(0, \alpha) \rightarrow (\gamma_1 = \alpha, 0) \rightarrow (\gamma_2, \delta_2) \rightarrow \cdots \rightarrow (\gamma_{t-1}, \delta_{t-1}) \rightarrow (0, \beta)$$

with $\gamma_i \neq 0$ for all $i \in \{1, \ldots, t-1\}$. Thus, we have $p_{\gamma_i} \leq 2^{-2}$ for all those $i$. Since $\gamma_1 = \alpha$ and $(0, \alpha)\rightarrow(\alpha, 0)$ holds with certainty ($p_0 = 1$), we have to show that either $p_{\gamma_i} \leq 2^{-4}$ for at least one $i$ or that $p_{\gamma_i}, p_{\gamma_j} \leq 2^{-3}$ for at least two distinct indices $i, j$. In other words, one has to make sure to gain a factor of $2^{-2}$ within the trail. In order to show this, we make use of Lemma 7.2. If $w_1(\alpha) \geq 4$, we are clearly done since $p_{\gamma_1} = p_\alpha \leq 2^{-w_1(\alpha)}$. We thus have to distinguish 3 cases.

(i) $w_1(\alpha) = 1$: Because of the branch number, it is $w_1(\theta(x) + \theta(x + \alpha)) \geq 10$. Since further $w_1(\rho(x) + \rho(x + \alpha)) \leq 2$, we have $w_1(\gamma_2) \geq 8$ and $p_{\gamma_2} \leq 2^{-4}$.

\(^4\)Using the improvements recently shown in [LLW17a], a branch number of 8 instead of 11 would suffice.
(ii) \( w_1(\alpha) = 2 \): It is \( w_1(\theta(x) + \theta(x + \alpha)) \geq 9 \) and \( w_1(\rho(x) + \rho(x + \alpha)) \leq 4 \). Thus, \( w_1(\gamma_2) \geq 5 \) and therefore \( p_{\gamma_2} \leq 2^{-4} \).

(iii) \( w_1(\alpha) = 3 \): We already have \( p_{\alpha} \leq 2^{-3} \). Since \( w_1(\theta(x) + \theta(x + \alpha)) \geq 8 \) and \( w_1(\rho(x) + \rho(x + \alpha)) \leq 6 \), it is \( w_1(\gamma_2) \geq 2 \) and therefore \( p_{\gamma_2} \leq 2^{-3} \).

See also Figure 7.3 for the propagation of the Hamming weights of differences \( \alpha \).

We recall that \( \theta \) does not have to be rotational invariant. Nevertheless, having a branch number of at least 11 is a quite restrictive property on a linear layer and in fact, for \( n' = 16 \), there does not exist such a linear mapping. The reason is that the minimum distance \( d \) of any \([32, 16, d]\) code over \( \mathbb{F}_2 \) is at most 8 \cite{Gra07}. However, for \( n' \in \{24, 32, 48, 64\} \) which corresponds to most of the block length in Simon, such a linear mapping \( \theta \) exists as one can also deduce from \cite{Gra07}. As the previous argument is more generic one, we investigate the linear part of Simon in more detail in the following.

7.3.2 Obtaining the Upper Bound for Simon and Simeck

In the following, we consider the linear part \( \theta(x) = (x \ll c) \), which has a branch number of only 2. Choosing \((8, 1, 2)\) for the rotation constants \((a, b, c)\), we obtain the round function of Simon. Theorem 7.2 states the same bound as above for all versions of Simon with regard to the supported block lengths. Note that the results are dependent on the specific choice of the rotation constants, but can be proven for other choices in a similar way. Of course, it does not hold for all possible \( a, b \) and \( c \). For example, if \( c = a \) or \( c = b \), one obtains the trivial bound of \( P(C) \leq 2^{-t} \) for any non-zero \( t \)-round trail \( C \) since

\[
C' = ((1,0,\ldots,0),0) \rightarrow (0,(1,0,\ldots,0)) \rightarrow ((1,0,\ldots,0),0)
\]
would be a valid two-round trail with \( P(C') = 2^{-2} \) that is iterative and can thus be concatenated to longer trails.

**Theorem 7.2** (Bounds for Simon). Let \( n' \in \{16, 24, 32, 48, 64\} \) and let \( \theta(x) = (x \ll 2) \). For the rotation constants \( a = 8, b = 1 \), the value of \( P(C) \) for any \( T \)-round differential trail \( C \) over round functions \( F^{F_{x}} \) is upper bounded by \( 2^{-2T+2} \).

**Proof.** Again, fix a \( t \)-round trail of the form

\[
(0, \alpha) \rightarrow (\gamma_1 = \alpha, 0) \rightarrow (\gamma_2, \delta_2) \rightarrow \cdots \rightarrow (\gamma_{t-1}, \delta_{t-1}) \rightarrow (0, \beta)
\]

with \( \gamma_i \neq 0 \) for all \( i \in \{1, \ldots, t-1\} \). We have to show that either \( p_{\gamma_i} \leq 2^{-4} \) for at least one \( i \) or that \( p_{\gamma_i}, p_{\gamma_j} \leq 2^{-3} \) for at least two distinct indices \( i, j \). In order to show this, Lemma 7.2 is used several times within this proof. Again, we have to distinguish three cases. Note that for simplicity with indices, we assume rotations to the right in the following. We use the * symbol to indicate a bit with unspecified value.

(i) \( w_1(\alpha) = 1 \): Considering the rotational equivalence, let, w.l.o.g.,

\[
\alpha = (1, 0, \ldots, 0)
\]

Recall that we get \( U_{\alpha} = \text{Im} L_{\alpha} + f_{S}(\alpha) \). Since we assume

\[
f_{S}: x \mapsto (x \gg 8) \land (x \gg 1) + (x \gg 2),
\]

we obtain

\[
\gamma_2 = (0, *, 1, 0, 0, 0, 0, *, 0, 0, 0, 0, 0, 0 \ldots) \in U_{\alpha} + 0.
\]

**Case 1** (*= 0): Then,

\[
\begin{align*}
\gamma_3 &= (1, 0, *, *, 1, 0, 0, 0, 0, *, 0, 0, 0, 0 \ldots) \in U_{\gamma_2} + \alpha, \\
\gamma_4 &= (0, *, *, *, *, 1, 0, 0, 0, 0, *, 0, *, 0, 0, 0 \ldots) \in U_{\gamma_3} + \gamma_2.
\end{align*}
\]

If now the weight of \( \gamma_4 \) is larger than 1, then \( p_{\gamma_3}, p_{\gamma_4} \leq 2^{-3} \). Thus, let \( w_1(\gamma_4) = 1 \). It follows that

\[
\gamma_5 = (1, 0, *, *, 1, 0, 0, *, 1, *, 0, 0, 0 \ldots) \in U_{\gamma_4} + \gamma_3
\]

and thus \( p_{\gamma_5} \leq 2^{-3} \).

**Case 2** (*= 1): Then \( p_{\gamma_2} \leq 2^{-3} \) already holds and

\[
\begin{align*}
\gamma_3 &= (0, *, 0, 0, 1, 0, 0, *, 0, 0, 0, 0, 0 \ldots) \in U_{\gamma_2} + \alpha.
\end{align*}
\]

\[\dagger\]This bit is only unknown if the length is 16 bit (\( n' = 16 \)). Therefore, w.l.o.g., we assume this bit to be unknown. In the following, we may also consider certain bits to be unknown if the actual value does not matter for the proof.
Again, w.l.o.g., let $w_1(\gamma_3) = 1$. It follows that
\[ \gamma_4 = (0, *, 1, 0, \ 0, *, 1, 0, \ 1, 0, 0, 0, \ *, 0, 0, 0 \ \ldots) \in U_{\gamma_3} + \gamma_2 \]
and thus $p_{\gamma_4} \leq 2^{-3}$.

(ii) $w_1(\alpha) = 2$: Considering the rotational equivalence, let, w.l.o.g.,
\[ \alpha = (1, 0, \ldots, 0, 1^{(i)}, 0, \ldots, 0) \]
with $i \leq \frac{n'}{2}$. It follows that already $p_\alpha \leq 2^{-3}$.

**Case 1** ($i = 1$): Then,
\[ \gamma_2 = (0, *, *, 1, \ 0, 0, 0, 0, \ *, 0, 0, 0 \ \ldots) \in U_{\alpha} + 0 . \]
Again, w.l.o.g., let $w_1(\gamma_2) = 1$. Then,
\[ \gamma_3 = (1, 1, 0, 0, \ *, 1, 0, 0, \ 0, 0, 0, 0 \ \ldots) \in U_{\gamma_2} + \alpha \]
and thus $p_{\gamma_3} \leq 2^{-3}$.

**Case 2** ($i = 4$): Then,
\[ \gamma_2 = (0, *, 1, 0, \ 0, *, 1, 0, \ *, 0, 0, 0 \ \ldots) \in U_{\alpha} + 0 \]
and $p_{\gamma_2} \leq 2^{-3}$.

**Case 3** ($i \neq 1, i \neq 4$): Then,
\[ \gamma_2 = (*, *, 1, *, \ *, *, *, *, \ *, *, *, *, \ *, *, \ *, *, \ \ldots) \in U_{\alpha} + 0 . \]
Again, w.l.o.g., let $w_1(\gamma_2) = 1$. Then,
\[ \gamma_3 = (1, *, *, *, \ 1, *, *, *, \ *, *, *, *, \ *, *, *, *, \ \ldots) \in U_{\gamma_2} + \alpha \]
and thus $p_{\gamma_3} \leq 2^{-3}$.

(iii) $w_1(\alpha) = 3$: W.l.o.g., let $\alpha = (1, 0, \ldots, 1^{(i)}, 0, \ldots, 1^{(j)}, 0, \ldots, 0)$ with $i \geq \frac{n'}{3}$ because of the rotational invariance. Again, $p_\alpha \leq 2^{-3}$. Since $n' \geq 16$, it is $i \geq 6$. We distinguish the following cases:

**Case 1** ($j \neq n' - 6, i \neq n' - 6$): Then,
\[ \gamma_2 = (*, *, 1, *, \ *, *, *, *, \ \ldots \ *, *, *, *, \ *, *, *, *, *, *) \in U_{\alpha} + 0 \]
and for $w_1(\gamma_2) = 1$ we obtain
\[ \gamma_3 = (1, 0, 0, *, \ 1, 0, *, *, \ \ldots \ *, *, *, \ *, *, *, *, *, *, *) \in U_{\gamma_2} + \alpha \]

\[^{1}\text{Of course, this bit is already equal to 1 if the length } n' \text{ is larger than 16.}\]
such that $p_{\gamma_3} \leq 2^{-3}$.

**Case 2** ($i = n' - 6$): Then,

$$\gamma_2 = (\ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, 1, \ast, \ast) \in U_\alpha + 0$$

if $j \neq n' - 5$ and

$$\gamma_2 = (\ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, 1, \ast, \ast) \in U_\alpha + 0$$

if $j = n' - 5$. In both cases, for $w_1(\gamma_2) = 1$ we obtain

$$\gamma_3 = (1^{(0)}, 0, 0, 0, \ast, \ast, 0, 0, \ldots, 0, 0, 1^{(i)}, \ast, \ast, \ast, \ast) \in U_{\gamma_2} + \alpha$$

such that $p_{\gamma_3} \leq 2^{-3}$.

**Case 3** ($j = n' - 6$): Now, we still have to consider the two possibilities $j - i \neq 6$ and $j - i = 6$. For the first case, one gets

$$\gamma_2 = (\ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, 1, \ast, \ast) \in U_\alpha + 0$$

and for $w_1(\gamma_2) = 1$,

$$\gamma_3 = (1^{(1)}, 0, 0, 0, \ast, \ast, 0, 0, \ldots, 0, 0, 1^{(i)}, \ast, \ast, \ast, \ast) \in U_{\gamma_2} + \alpha$$

If $j - i = 6$, then,

$$\gamma_2 = (\ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, 1^{(i+2)}, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast) \in U_\alpha + 0$$

and for $w_1(\gamma_2) = 1$,

$$\gamma_3 = (1^{(1)}, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, 1^{(i)}, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast, \ast) \in U_{\gamma_2} + \alpha$$

Using a similar argument, one obtains the bounds for Simeck as the following theorem states.

**Theorem 7.3** (Bounds for Simeck). Let $n' \in \{16, 24, 32\}$ and $\theta(x) = (x \ll 1)$. For the rotation constants $a = 5, b = 0$, the value of $P(C)$ for any $T$-round differential trail $C$ over round functions $F_S^f$ is upper bounded by $2^{-2T+2}$.

Interestingly, for every version of Simon and Simeck, it turns out that our approach is sufficient in order to bound $P(C)$ below $2^{-n}$ for a smaller number of rounds than specified in the actual cipher. Recall that $n'$ denotes the length of one Feistel branch, i.e., half the block length. For $n'$ up to 32, the security margin $t_m$ of the corresponding cipher(s) can be considered as reasonable. See Table 7.1 for a comparison.
Table 7.1: Number of rounds \( t \) needed for bounding the value of \( P(C) \), where \( C \) is any non-zero \( t \)-round differential trail, below \( 2^{-n} \) for all versions of Simon and Simeck. The \( \ast \) symbol indicates that there is an appropriate version of Simeck with the same number of rounds.

<table>
<thead>
<tr>
<th></th>
<th>rounds</th>
<th>rounds needed</th>
<th>margin ( t_m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simon-32-64*</td>
<td>32</td>
<td>17</td>
<td>15</td>
</tr>
<tr>
<td>Simon-48-72</td>
<td>36</td>
<td>25</td>
<td>11</td>
</tr>
<tr>
<td>Simon-48-96*</td>
<td>36</td>
<td>25</td>
<td>11</td>
</tr>
<tr>
<td>Simon-64-96</td>
<td>42</td>
<td>33</td>
<td>9</td>
</tr>
<tr>
<td>Simon-64-128*</td>
<td>44</td>
<td>33</td>
<td>11</td>
</tr>
<tr>
<td>Simon-96-96</td>
<td>52</td>
<td>49</td>
<td>3</td>
</tr>
<tr>
<td>Simon-96-144</td>
<td>54</td>
<td>49</td>
<td>5</td>
</tr>
<tr>
<td>Simon-128-128</td>
<td>68</td>
<td>65</td>
<td>3</td>
</tr>
<tr>
<td>Simon-128-192</td>
<td>69</td>
<td>65</td>
<td>4</td>
</tr>
<tr>
<td>Simon-128-256</td>
<td>72</td>
<td>65</td>
<td>7</td>
</tr>
</tbody>
</table>

7.4 Conclusion and Open Problems

We presented a more general description of Simon-like designs by separating the round function into a linear and a non-linear component and developed a non-experimental security argument on full-round versions of Simon that can be verified by hand. We hope that this work encourages further research on analyzing Simon-like designs. An open question is whether our approach can be generalized in order to obtain better bounds over multiple rounds. The recent improvements shown in [LLW17a] seem to be a good starting point. Furthermore, it would be favorable to avoid the consideration of every special case individually. This is related to the question of how to design the linear part \( \theta \) in this set-up.


Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications.


[LW16] Yongqiang Li and Mingsheng Wang. On the construction of lightweight circulant involutory MDS matrices. In Thomas Peyrin,


Siwei Sun, Lei Hu, Meiqin Wang, Peng Wang, Kexin Qiao, Xiaoshuang Ma, Danping Shi, Ling Song, and Kai Fu. Towards finding


