Algebraic attacks on certain stream ciphers

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von
Diplom-Mathematiker Frederik Armknecht
aus Worms

Mannheim, 2006

Dekan: Professor Dr. Matthias Krause, Universität Mannheim
Referent: Professor Dr. Matthias Krause, Universität Mannheim
Korreferent: Professor Dr. Willi Meier, Fachhochschule Nordwestschweiz

Tag der mündlichen Prüfung: 30. November 2006

Abstract

To encrypt data streams of arbitrary lengths, keystream generators are used in modern cryptography which transform a secret initial value, called the key, into a long sequence of seemingly random bits. Many designs are based on linear feedback shift registers (LFSRs), which can be constructed in such a way that the output stream has optimal statistical and periodical properties and which can be efficiently implemented in hardware. Particularly prominent is a certain class of LFSR-based keystream generators, called (ι, m)-combiners or simply combiners. The maybe most famous example is the E_0 keystream generator deployed in the Bluetooth standard for encryption.

To evaluate the combiner's security, cryptographers adopted an adversary model where the design and some parts of the input and output are known. An attack is a method to derive the key using the given knowledge. In the last decades, several kinds of attacks against LFSR-based keystream generators have been developed. In 2002 a new kind of attacks came up, named "algebraic attacks". The basic idea is to model the knowledge by a system of equation whose solution is the secret key. For several existing combiners, algebraic attacks represent the fastest theoretical attacks publicly known so far.

This thesis discusses algebraic attacks against combiners. After providing the required mathematical fundament and a background on combiners, we describe algebraic attacks and explore the two main steps (generating the system of equations and computing the solution) in detail. The efficiency of algebraic attacks is closely connected to the degree of the equations. Thus, we examine the existence of low-degree equations in several situations and discuss multiple design principles to thwart their existence. Furthermore, we investigate "fast algebraic attacks", an extension of algebraic attacks.

Zusammenfassung

Um Datenströme beliebiger Länge zu verschlüsseln werden in der modernen Kryptographie Schlüsselstromgeneratorern eingesetzt, welche einen geheimen Initialwert, den so genannten Schlüssel, in einen langen Strom scheinbar zufälliger Bits zu transformieren. Viele Designs basieren auf linearen Rückkopplungsschieberegistern (in Englisch: linear feedback shift registers (LFSRs)), welche so konstruiert werden können, dass die Ausgabeströme optimale statistische und periodische Eigenschaften aufweisen, und welche effizient in Hardware implementiert werden können. Besonders bedeutend ist eine spezielle Klasse der LFSR-basierten Schlüsselstromgeneratoren, welche (ι, m)-Combiner oder einfach nur Combiner genannt werden. Das vielleicht bekannteste Beispiel ist der E_0 Schlüsselstromgenerator, eingesetzt im Bluetooth Standard zur Verschlüsselung.

Um die Sicherheit eines Combiners zu evaluieren wenden Kryptographen ein Feindmodell an, in welchem sowohl das Design als auch Teile der Ein- und Ausgabe bekannt sind. Ein Angriff stellt eine Methode dar, das vorhandene Wissen zu nutzen, um den geheimen Schlüssel abzuleiten. In den letzten Jahrzehnten wurden unterschiedliche Arten von Attacken gegen LFSR-basierten Schlüsselstromgeneratoren entwickelt. 2002 kam ein neues Verfahren auf, genannt "algebraische Attacke". Die zugrundeliegende Idee ist es, das Wissen durch eine Gleichungssystem darzustellen, dessen Lösung dem geheimen Schlüssel entspricht. Algebraische Attacken stellen für manche existierende Combiner die bisher schnellsten öffentlich bekannten Angriffe dar.

Diese Arbeit behandelt algebraische Attacken gegen Combiner. Nach der Bereitstellung des notwendigen mathematischen Fundaments und des Hintergrundes zu Combinern erklären wir algebraische Attacken und untersuchen die beiden Hauptschritte (Erstellen des Gleichungssystems und Berechnung der Lösungen). Die Effizienz der algebraischen Attacken ist eng verknüpft mit dem Grad der Gleichungen. Aus diesem Grund erforschen wir die Existenz solcher Gleichungen mit niedrigem Grad und diskutieren mehrere Designprinzipien zur Vermeidung ihrer Existenz. Ausserdem untersuchen wir die so genannten "schnellen algebraischen Attaken", eine Erweiterung der algebraischen Attacken.

Acknowledgements

It belongs to the most pleasant and most difficult "duties" of an author to thank all the people who contributed directly or indirectly to his work. Pleasant as it gives the opportunity to credit all the helpful persons, which can never happen often enough. Difficult as the list is far too big to be covered in a few lines and many are likely to be overlooked. Nonetheless, I will try to do my best in expressing my gratitude at least to those who had the most impact on my work.

First and foremost, I wish to thank my two PhD supervisors, Prof. Dr. Matthias Krause and Prof. Dr. Willi Meier[1], for their support and the both pleasant and fruitful collaborations. I've always considered it to be an enormous luck to have the possibility of working and discussing with two such gifted minds. Matthias deserves additional credit for handling all the administrative problems of becoming a PhD - including but not limited to making sure that there would always be sufficient money to keep me going.[2]

Additional thanks go to Dr. Stefan Lucks. I enjoyed our inspiring brainstorming sessions and profited more than once from his broad and deep knowledge on cryptography. A special thank goes to my former colleague and good friend Dr. Erik Zenner, whose support accompanied me from the first day when he guided me through the University's campus until the end when he proofread most of my thesis. Without his good advices, the start would have been much harder.

I heartily thank all the nice colleagues at the University of Mannheim which made my work environment such a pleasant place, especially Dirk Stegemann for his technical support and for proofreading parts of my thesis.

I would also like to stress how pleasant the contact and work was and is with the cryptographic community. It is amazing to see and to talk to so many talented and intelligent people. And the fact that even the most prominent crypto experts treat us newcomers with respect cannot be esteemed enough.

No words can express my gratitude to my wife Ricarda for all the wonderful years we spent together and for the years to come. She and our son Leonard enrich my life more than anything else. They help me to keep my focus on what really matters in life. Therefore, this work is dedicated to her.

[1] Actually, the title of the thesis is a homage to one of his famous papers.

[2] This work was partially supported by grant Kr 1521/7-2 of the DFG (German Research Foundation).

To my wife Ricarda

Contents

1 Introduction

Modern cryptography

Historically, cryptography arose as a means to ensure the privacy of the information that communicating parties send to each other, even in the presence of an adversary with access to the communication channel. While first approaches were based on steganography, i.e., hiding messages in such a way that no one except for the intended recipient knows of the existence of the message, later on one tried to encrypt messages. Encryption is the process of obscuring information to make it unreadable without special knowledge.

Not surprisingly, cryptography was mainly the domain of the military, secret agencies and diplomats for a long time. However, the increased demand for confidential data exchange, last but not least pushed by the upcoming electronic communication, made cryptography also of interest for the regular citizen. But it was not before the early 1970s that two events brought it into the public domain: the creation of a public encryption standard (DES), and the invention of public-key cryptography.

Since then cryptography evolved a lot and broadened its scope. Although modern cryptography covers various subjects such as digital cash, authentication and electronic voting, encryption still remains the fundamental objective.

Keystream generators

In multiple occasions, there exists a need for encrypting data streams with high speed, e.g., for phone calls or video streams. Especially designed for this task are stream ciphers. Many stream ciphers used in practice and discussed in theory are based on keystream generators. A keystream generators produces an output stream of arbitrary length, named the keystream, after it has been initialized with a secret value, called the key. The keystream is then combined with the data stream to encrypt it. A legal receiver who knows the secret key and the used keystream generator can produce the same keystream by himself and decrypt the encrypted data stream.

For several reasons, linear feedback shift registers (LFSRs) have turned out to be very advantageous building blocks for keystream generators. They can be efficiently implemented in hardware and generate streams with good statistical properties. Examples for LFSR-based keystream generators used in practice are E_0 from the Bluetooth standard for wireless communication, $A5/1$ used in the GSM-encryption, and $RC4$ used in SSH, HTTPS, and WLAN.

If the keystream generator is to be used in a cryptographic setting, it should be impossible (or at least practically infeasible) for someone not

knowing the secret key to decrypt the encrypted data stream. Of course, an attacker could simply try to guess the secret key. We call this strategy a brute force attack. In order to avert this approach, modern key sizes are chosen of length of 128 bits or more. However, one cannot exclude the possibility that a potential adversary has some knowledge about both the keystream generator and the keystream. Thus, he might exploit possible structural weaknesses to break the scheme. To understand and evaluate the security of keystream generators, cryptographers developed several kinds of methods and attacks in the last decades. The underlying goal of cryptanalysis is not destructive, but constructive: Only by improving the understanding of possible weaknesses, it is possible to construct new design principles.

The newest attacks, dating back to 2002, are "algebraic attacks". The basic idea is to model the attacker's knowledge by a system of equation whose solution reveals the secret key. For several existing LFSR-based keystream generators, algebraic attacks represent the fastest theoretical attacks publicly known so far. Another remarkable fact is that under certain conditions the effort grows only polynomial with the key size. This differs from most attacks which have an exponential effort.

On this thesis

Since the upcoming of the first papers on algebraic attacks, cf. for example [Cou02, CouM03], the theory of algebraic attacks has evolved a lot, including the extension to a broader class of keystream generators [ArmK03], the introduction of fast algebraic attacks [Cou03, Arm04a], or the connection to annihilators [MeiPC04, Arm05a]. Hence, some statements or results in the earlier papers are no longer up-to-date. This fact and the steadily growing number of papers on this subject make it difficult for newcomers to orient on this field.

Therefore, one of the main goals of this thesis, apart from presenting own results of course, is to provide a comprehensible and comprehensive description of the main results concerning algebraic attacks. We will try to cover most main results on algebraic attacks and describe them in their context. Of course, the content and the presentation are influenced by our viewpoint and our results. Thus, we recommend the papers cited in this work for further reading.

To be self-contained, we pay attention to prove almost all statements[1] and to require as little knowledge as possible. Actually, we assume only some basic linear algebra. Everything else will be introduced and explained. The-

[1] If not otherwise stated, we conducted the proofs by ourself.

ory is only introduced if and when it is needed.

Furthermore, we will give examples for the majority of the definitions and statements. This hopefully makes it easier for the reader to understand the sometimes very technical content. Of course due to the rather theoretic nature of most parts, a certain amount of notations and definitions is unavoidable. Therefore, we included an index at the end of the thesis, which allows the reader to quickly look up the most important terms. These terms are also highlighted in bold in the main text to be easily recognizable. Also included at the beginning of the index is a list of the most important notations.[2]

Due to the rapidly growing number of publications on this field, it is out of hope to cover all results or to be up to date. Already in the time between April 2006, when we stopped writing down new content, and August 2006, when we finalized the text, several papers appeared which are not mentioned in this text. However, our hope is that the thesis suits as a good starting point for understanding algebraic attacks and some of the developments.

The thesis is organised as follows:

- Chapter 2 provides the general framework and the mathematical background of the thesis and introduces important notions and concepts. After introducing the ideas of keystream generators in Section 2.1, we provide the mathematical theory used throughout the thesis in Section 2.2. We presuppose for this purpose only basic knowledge in linear algebra. As the system of equations in algebraic attacks consists of functions over finite fields, we explore this topic in Section 2.3. In Section 2.4, we will learn more about LFSRs, and in Section 2.5 about the type of LFSR-based keystream generators that we will consider in the thesis. The chapter concludes with Section 2.6 where we give a short overview of already existing attacks.

- Chapter 3 describes algebraic attacks on LFSR-based keystream generators. We explicate the basic principles in Section 3.1,namely that algebraic attacks are based on creating and solving a system of equations. Consequently, we describe in Section 3.2 how to generate the system of equations, including our extensions to combiners with memory [ArmK03] and examine in Section 3.3 various methods to find the solutions. At the end, we give a rough effort analysis in Section 3.4.

- The efficiency of algebraic attacks is closely related to the existence of low-degree equations and how to find them. Chapter 4 is dedicated

[2]However, we could not always understand the ordering chosen by LaTeX.

completely to this topic. This includes a complete framework on the existence of such equations in Section 4.1 for combiners with memory over arbitrary finite fields [Arm05a], algorithms to find them in Section 4.2, including our special algorithms for permutation invariant combiners from [Arm04b], and design principles to avoid them in Section 4.3 (with our proposal for combiners with memory from [ArmKS05]). At the end, we describe in Section 4.4 our algebraic attacks based on generating low-degree equations by exploiting linear keyschedules [ArmLP04] or using fault attacks [ArmM05].

- Chapter 5 is on fast algebraic attacks. Their basic idea is to use the special structure of the system of equations to reduce the degree before starting to solve it, which is described in Section 5.1. In Section 5.2, we explain our results from [Arm04a] on the precomputation step. After briefly addressing another variant of fast algebraic attacks in Section 5.3, we explain in Section 5.4 extensions of the precomputation step to reduce the data effort to a minimum (both from [ArmA05]). The chapter concludes with Section 5.5 on our results on the immunity of simple combiners against fast algebraic attacks from [ArmCGKMR06].
- Chapter 6 provides final conclusions.

Publications:

The contents of this thesis are based on a number of publications by the author. We give here a complete list of our publications:

1. F. Armknecht: *A Linearization attack on the Bluetooth Keystream Generator*, Cryptology ePrint Archive, Report 2002/191, 2002.
2. F. Armknecht and M. Krause: *Algebraic Attacks on Combiners with Memory*, Crypto 2003.
3. F. Armknecht: *Improving Fast Algebraic Attacks*, Fast Software Encryption 2004.
4. F. Armknecht, S. Lucks: *Linearity of the AES Key Schedule*, 4th AES Coenference, 2004.
5. F. Armknecht, J. Lano, and B. Preneel: *Extending the Resynchronization Attack*, Selected Areas in Cryptography 2004.
6. F. Armknecht: *On the Existence of low-degree Equations for Algebraic Attacks*, Cryptology ePrint Archive, Report 2004/185, 2004.

7. F. Armknecht: *Algebraic Attacks and Annihilators*, WEWORC 2005.

8. F. Armknecht and W. Meier: *Fault Attacks on Combiners with Memory*, Selected Areas in Cryptography 2005.

9. F. Armknecht and G. Ars: *Introducing a New Variant of Fast Algebraic Attacks and Minimizing Their Successive Data Complexity*, Mycrypt 2005.

10. F. Armknecht, M. Krause, and D. Stegemann: *Design Principles for Combiners with Memory*, Indocrypt 2005.

11. F. Armknecht, J. Brandeis and E. Ilinykh: *Experimental Results on Algebraic Attacks on Stream Ciphers*, Sicherheit 2006.

12. F. Armknecht, C. Carlet, P. Gaborit, S. Künzli, W. Meier, and O. Ruatta: *Efficient Computation of Algebraic Immunity for Algebraic and Fast Algebraic Attacks*, Eurocrypt 2006.

13. F. Armknecht and M. Krause: *Constructing single- and multi-output Boolean functions with maximal algebraic immunity*, ICALP 2006.

2 Preliminaries

2.1 Keystream generators

Shannon described the principle of **encryption** in [Sha49] as a modification of his well-known communication model, proposed in [Sha48]. Involved are two entities, usually referred to as **sender** and **receiver**. The sender's goal is to confidentially transmit some data, named the **plaintext**, to the receiver. For this purpose he has access to two different communication channels:

Secret channel: The **secret channel** is completely confidential. None of the data transmitted over this channel can be eavesdropped by anybody except sender and receiver. However, the usage of this channel is limited to certain time points, e.g. when sender and receiver are in the same place.

Public channel: The **public channel** can be used anytime to transmit data of arbitrary size. Yet, it is insecure in so far as anybody can listen to the transmitted messages.

As the usage of the secret channel is limited to certain periods, it cannot always be used to transmit the plaintext. On the other hand, as an adversary has access to the public channel, the plaintext cannot be send on this way either.

One possibility to conceal the messages from eavesdroppers is to use a cryptosystem to encrypt it. This means that the data is modified before submission in such a way that the transmission, called the **ciphertext**, reveals no obvious information about the underlying message. Only a legitimate receiver should be able to un-do the modification of the data to recover the original meaning.

The formal notion of a cryptosystem is the following:

Definition 2.1. *[Sti02, Def. 1.1] A **cryptosystem** or **cipher** is a five-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, E, D)$, such that the following conditions are satisfied:*

- *$\mathcal{P}$ is a finite set of possible plaintexts.*
- *$\mathcal{C}$ is a finite set of possible ciphertexts.*
- *$\mathcal{K}$, the keyspace, is a finite set of possible keys.*
- *$E : \mathcal{K} \times \mathcal{P} \to \mathcal{C}$ is the encryption algorithm*
- *$D : \mathcal{K} \times \mathcal{C} \to \mathcal{P}$ is the decryption algorithm*
- *For each $K \in \mathcal{K}$, it holds that $D(K, E(K, P)) = P$ for all $P \in \mathcal{P}$.*

According to the established notation, we write $E_K(.)$ for $E(K,.)$ and similarly $D_K(.)$ for $D(K,.)$. Sender and receiver now use the following protocol to exchange confidential messages:

1. At first they agree on a secret key $K \in \mathcal{K}$. Therefor, the secret channel is used to exchange all necessary data. This has to be done only once.
2. Given a plaintext $P \in \mathcal{P}$, the sender uses the encryption function E_K to encrypt the plaintext P to the ciphertext $C := E_K(P)$.
3. Thereafter, the ciphertext C is sent via the public channel to the receiver.
4. A receiver, who knows the employed secret key K, decrypts the ciphertext by $P = D_K(C)$.

The whole situation is displayed in Figure 2.1.1.

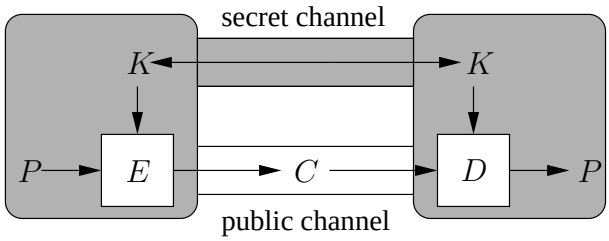

Figure 2.1.1: Shannon's model of encryption

Observe that any potential eavesdropper listening to the public channel gets to know only C but not P. If an adversary is still able to derive some information on P and/or K, the cryptosystem should certainly be regarded as insecure. This leads us directly to the question on how to evaluate the security of a cryptosystem.

First of all, it is necessary to introduce a third party, called the **attacker**. Following **Kerckhoffs' principle** [Ker83], it is assumed that an attacker has access to all public information. That includes both any data transmitted via the public channel and the definitions of the cryptosystem, that is the five-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, E, D)$. An attacker's goal is to derive some secret information, for instance the plaintext P and/or the secret key K. The algorithm deployed by him will be called **attack**.

Different definitions of attackers and attack models exist (e.g., see [Rue89, Rue92], and [Zen04, Chapter 2]). However, we will concentrate only on the following instance. An attacker is able to operate on a uniform computational model, like a Turing machine, whose computational behaviour is

similar to that of a programmable microprocessor. Or, less formal, the attacker has access to a personal computer to perform the computations of his attack. One single operation will be called a **basic operation**. To evaluate the efficiency of an attack, we will consider the following three values:

1. The number of data needed, e.g. the amount of plaintext and/or ciphertext.
2. The run time to perform the attacker, that is the number of basic operations.
3. The amount of memory required to store information for an attack.

Of course, different attackers may have access to machines with different capabilities. Thus, it doesn't make sense to refer to exact times and memory requirements when evaluating an attack. On the other hand, if one compares two different machines, any basic operation performed on one machine can be likewise executed by a constant number of basic operations on the other one. Therefore, to keep the evaluations of attacks independent from the deployed machine, we will describe the number of basic operations and memory consumptions in the usual $\mathcal{O}$-notation:

Definition 2.2. *Let $f : \mathbb{N} \to \mathbb{R}$ be a function from the set of integers to the set of real numbers. By $\mathcal{O}(f)$, we denote the set*

$$\mathcal{O}(f) := \{g : \mathbb{N} \to \mathbb{R} \mid \exists c > 0, \exists n_0 \geq 0 : \forall n > n_0 : g(n) < c \cdot f(n)\}$$

That is, any attack needs an amount of data in $\mathcal{O}(d)$, a number of (basic) operation in $\mathcal{O}(o)$ and a memory of size in $\mathcal{O}(m)$ where d, o, and m are some mappings from $\mathbb{N}$ to $\mathbb{R}$ and the input is normally the key size.

For certain applications in practice, e.g. encryption in mobile phones, there is a need for cryptosystems that are able to encrypt data of arbitrary length as fast as possible. One possibility, which is widely used, are stream ciphers based on keystream generators. Examples are the keystream generator E_0 from the Bluetooth standard for wireless communication [Blu99], $A5/1$ used in the GSM-encryption [BriGW98, ZenWL00], and $RC4$ used in SSH, HTTPS, and WLAN [FluMS01]. Keystream generators are finite state machines with an additional output function to produce sequences of outputs of arbitrary length. We first give a formal definition:

Definition 2.3. *A **keystream generator** consists of the following components:*

- *an internal state,*

- *a finite set $\mathcal{K}$ of possible internal states,*
- *an update function $\Upsilon : \mathcal{K} \to \mathcal{K}$,*
- *a finite set $\mathcal{Z}$ called the **keystream alphabet**, and*
- *an **output function** $f : \mathcal{K} \to \mathcal{Z}^*$.*

*At the beginning, the internal state is initialized to a key (or **initial state**) $S_0 \in \mathcal{K}$. Then, the following operations are performed repeatedly:*

1. *An output $z_t \in \mathcal{Z}$ is computed by $z_t = f(S_t)$.*
2. *The internal state is updated according to $S_{t+1} := \Upsilon(S_t)$.*

*We say that a **clock** is the time period within the two operations described above are executed. That is, in clock t the output z_t is produced and S_t changed to S_{t+1}. The stream $z_0, z_1, \ldots$ of outputs is called the **keystream**.*

Remark 2.4. *Usually one would prefer to choose a bijective update function Υ. Otherwise, different keys might eventually lead to the same keystream, which might weaken the cryptographic security if the other components are not appropriate designed.*

The idea of keystream generators is derived from the **one-time pad**, introduced in [Ver26]. Let $\mathcal{P} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$ for some integer $n \geq 1$. The encryption algorithm is defined as $E_K(P) := K \oplus P$ where $\oplus$ denotes the componentwise XOR. The corresponding decryption function is $D_K(C) = K \oplus C$. It has been proved in [Sha49] that this cryptosystem is perfectly secure if K is used only once (therefore the name one-time pad) and is chosen uniformly distributed from $\{0,1\}^n$. Thereby, perfectly secure means that if an attacker knows only C, then it is impossible for him to gain any information about P and/or K, even with unlimited time, memory and computational resources.

Of course, the one-time pad is infeasible for practical applications, as it would require to manage keys of the same size as the data that has to be encrypted. Thus, instead of using random keys of the same size as the message, one uses smaller keys to initialize a keystream generator to generate *pseudo-random* bitstreams of the length of the plaintext, giving a pseudo one-time pad. A formal definition of the cryptosystem is the following:[1]

[1] Although other kind of stream ciphers exist we will consider only these which are based on keystream generators as defined in [Sti02]. Notice that for practical applications further definitions are required, e.g. how to initialize the stream cipher and if, how, and when the value K should be updated.

Definition 2.5. *[Sti02, Def. 1.6] A (synchronous)* ***stream cipher*** *based on a keystream generator is a tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{Z}, E, D)$ together with a keystream generator KSG, such that the following conditions are satisfied:*

- *$\mathcal{P}$ is a finite set called the plaintext alphabet*
- *$\mathcal{C}$ is a finite set called the ciphertext alphabet*
- *$\mathcal{K}$, the keyspace, is a finite set of possible keys*
- *$\mathcal{Z}$ is a finite set called the keystream alphabet*
- *The keystream generator takes a key $K \in \mathcal{K}$ as input to generate a string $z_0, z_1, z_2, \ldots$, called the keystream, of arbitrary length with $z_t \in \mathcal{Z}$ for all $t \geq 0$.*
- *$E : \mathcal{Z} \times \mathcal{P} \to \mathcal{C}$ is the encryption algorithm*
- *$D : \mathcal{Z} \times \mathcal{C} \to \mathcal{P}$ is the decryption algorithm*
- *For each $z \in \mathcal{Z}$, it holds that $D(z, E(z, P)) = P$ for all $P \in \mathcal{P}$.*

Sender and receiver encrypt their messages as follows:

1. Sender and receiver agree on a secret key $K \in \mathcal{K}$. Therefor, the secret channel is used to exchange all necessary data. This has to be done only once.
2. Let $P = (p_0, p_1, p_2, \ldots)$ with $p_t \in \mathcal{P}$ denote the plaintext. The sender uses K and the keystream generator to generate a keystream $z_0, z_1, z_2, \ldots$ and encrypts P to $C := (c_0, c_1, c_2, \ldots)$ with $C_t := E_{z_t}(p_t)$. C is send to the receiver over the public channel.
3. The receiver, who has knowledge of the secret key K, can generate the same keystream $z_0, z_1, z_2, \ldots$ on his own.
4. Knowing the keystream, the receiver decrypts $p_t = D_{z_t}(c_t)$.

Figure 2.1.2 sketches this setting.

In cryptography, the stream cipher is claimed to be secure if and only if the underlying keystream generator is secure. For the security analysis of a keystream generator, we consider the following attack model.[2] An attacker knows all public information as for example the exact specifications Υ and f of the keystream generator. Additionally, he is able to observe the values

[2] Notice that other kind of attacks can be found in open literature as distinguishing attacks and predicting attacks.

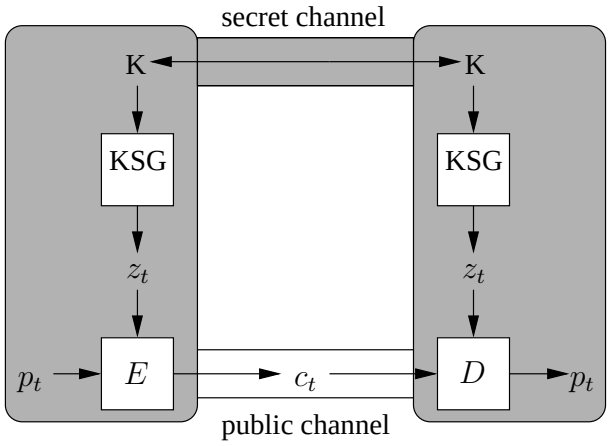

Figure 2.1.2: Encryption with a keystream generator

of some of the keystream elements z_t. The attacker's goal is to find out the secret key K. If he is successful, he can compute the keystream for himself and easily decrypt the whole ciphertext.

The simplest attack is the so called **brute force attack**, where the adversary exhaustively tries all $K' \in \mathcal{K}$ until a candidate is found that matches to the observed information. That is random values $K' \in \mathcal{K}$ are used to initialize the keystream generator and to produce some keystream. If this keystream is equal to the one observed, then the guess was seemingly correct. Observe that this attack requires a number of operations that is in $\mathcal{O}(|\mathcal{K}|)$. If an attacker wants to attack more than one entity of keystream generator, he can alternatively precompute for each possible key the corresponding keystream and store it in memory. Then, he could quickly look up in this table for each observed keystream the appropriate key. Of course, the time effort for the precomputation would be still in $\mathcal{O}(|\mathcal{K}|)$, likewise the memory requirements.

To avoid this kind of attack, nowadays key sizes are chosen such that the time and/or memory requirements for a brute force attack are by several magnitudes higher than it would be feasible in practice, even if an attacker would have access to many super computers. Consequently, an attack is claimed to be **successful** if it requires less time than it would require to try all possible keys and less space than the storage of all keys would need. However, we want to stress that a successful attack not necessarily means that the attack is also practical.

For security and synchronisation reasons, one would like to avoid that too much data is encrypted under the same key. Thus, in practice one has normally a secret master key $\hat{K}$, from which regularly different keys

are derived. This means that sender and receiver agree on a key schedule, which settles when and how the next key is computed from $\hat{K}$. But as in cryptography, each component should be as strong as possible, we will consider the keystream generator on its own and refer to the initial state as the secret key. In Section 4.4, we will come back to this topic and see that specific key schedules can weaken the employed keystream generator.

2.2 Mathematical preliminaries

In this section, we provide some mathematical preliminaries which are needed for the rest of the thesis. Readers who are familiar with algebra may skip this section. Indeed, we recommend to read remarks 2.26 and 2.44.

We start with the definitions of the basic structures group, ring and field.

2.2.1 Groups

Definition 2.6. *Let G be a set, together with an operation $\circ : G \times G \to G$. $(G, \circ)$ is called a **group** if it satisfies the group axioms below:*[3]

Associativity: *For all a, b, and c in G, $(a \circ b) \circ c = a \circ (b \circ c)$.*

Neutral element: *There is an element e in G such that for all a in G, $e \circ a = a \circ e = a$.*

Inverse element: *For all a in G, there is an element b in G such that $a \circ b = b \circ a = e$, where e is the identity element from the previous axiom.*

If $a \circ b = b \circ a$ for all $a, b \in G$, then the group is called abelian *.*

Usually, one encounters two different notations for the group operation: $+$ or $\cdot$. In the first case, the neutral element is denoted by 0 and the inverse of x by $-x$. In the case of the dot-notation, the neutral element is labeled with 1 and the inverse of x with x^{-1}. Obviously, these notations are inspired by the commonly used notations in $\mathbb{Q}$, the set of rational numbers. Consequently, one writes x^r for the r-times product of x, i.e. $x^r = x \cdot \ldots \cdot x$, and $r \cdot x$ for $x + \ldots + x$. In both cases, $r = 0$ gives the neutral element, i.e. $x^0 = 1$ and $0 \cdot x = 0$.

Example 2.7. *Let $\mathbb{Z}$ denote the set of integers, i.e. $\mathbb{Z} = \{0, 1, -1, 2, -2, \ldots\}$. $(\mathbb{Z}, +)$ is a group with the neutral element zero and the inverse $-x$. On the other side, $(\mathbb{Z}, \cdot)$ is not a group as no elements except +1 and -1 have an inverse. For example, $2 \cdot x \neq 1$ for all $x \in \mathbb{Z}$, showing that 2 has no inverse in $(\mathbb{Z}, \cdot)$.*

2.2.2 Rings

Definition 2.8. *Let R be a set, together with two operations $+, \cdot : R \times R \to R$, called addition and multiplication. $(R, +, \cdot)$ is a **ring** if*

[3] According to the established notation, we write $a \circ b$ instead of $\circ(a, b)$.

1. $(R, +)$ *is an abelian group with identity element 0.*
2. *There exists a neutral element* 1 *regarding the multiplication.*
3. $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ *for all* $a, b, c \in G$*.*
4. *Multiplication distributes over addition:*
   a) $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
   b) $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$

Definition 2.9. *Let* n *be an integer. By* $\mathbb{Z}_n$*, we define the set of integers* $\{0, \ldots, n-1\}$ *together with the operations* $x \circ y := (x \circ y) \mod n$ *where* $\circ \in \{+, \cdot\}$

Example 2.10. $(\mathbb{Z}, +, \cdot)$ *and* $(\mathbb{Z}_n, +, \cdot)$ *are rings.*

Example 2.11. *In* $\mathbb{Z}_2$*, only two elements do exist: 0 and 1. The operations* $+$ *and* $\cdot$ *are*

$+$	0	1
0	0	1
1	1	0

and

$\cdot$	0	1
0	0	0
1	0	1

Definition 2.12. *Let* $(R, +, \cdot)$ *be a ring. A subset* $I \subseteq R$ *is an* ***ideal*** *in* R *if the following three conditions are fulfilled:*

1. $0 \in I$
2. *For all* $x, y \in I$*, it is* $x + y \in I$
3. *For all* $x \in I$ *and* $c \in R$*, it is* $c \cdot x \in I$

Example 2.13. *Consider the ring* $(\mathbb{Z}, +, \cdot)$ *and* $n \in \mathbb{Z}$*. The set* $I_n := \{x \in \mathbb{Z} \mid n \text{ divides } x\}$ *is an ideal in* $\mathbb{Z}$*.*

Definition 2.14. *Let* R *be a ring and* $G \subseteq R$ *a finite subset. Then, the set*

$$\langle G \rangle := \{\sum_{g \in G} c_g \cdot g \mid c_g \in R\} \tag{2.2.1}$$

is an ideal in R*. It is called the* ***ideal generated by*** G*. If* $G = \{g\}$ *contains only one element, the ideal* $\langle g \rangle := \langle G \rangle$ *is called a* ***principal ideal****.*

Example 2.15. *Consider* $\mathbb{Z}[x, y, z]$*, the ring of polynomials over* $\mathbb{Z}$ *in some unknowns* x*,* y*, and* z*. Then,*

$$\left\langle x \cdot y, y^2 + z \right\rangle = \{\alpha \cdot x \cdot y \ + \ \beta \cdot (y^2 + z) | \alpha, \beta \in \mathbb{Z}\}$$

is the ideal in $\mathbb{Z}[x, y, z]$ *generated by* $G = \{x \cdot y, y^2 + z\}$*.*

The ideal $I_n \subset \mathbb{Z}$ *is generated by the element* n*. In other words,* $I_n := \langle n \rangle$*.*

Definition 2.16. *Let R be a ring and $I \subseteq R$ an ideal. A subset $B \subseteq I$ is called a **basis** of I if $I = \langle B \rangle$.*

Theorem 2.17. *Let R be a ring and $I \subseteq R$ an ideal. We define by R/I a set of equivalence classes in R where two elements $a, b \in R$ are equivalent iff $a - b \in I$. Then, R/I is a ring again.*

Proof. For $a \in R$, we denote by $[a] := \{b \in R | a - b \in I\}$ its equivalence class. We show that $[a] + [b] = [a + b]$ and $[a] \cdot [b] = [a \cdot b]$. Then, R/I inherits the ring properties directly from R.

Let $a, b \in R$ and $x \in [a]$ and $y \in [b]$. Then, there exist $\alpha, \beta \in I$ such that $x = a + \alpha$ and $y = b + \beta$. Some simple computations reveal:

$$\begin{aligned} x + y &= a + \alpha + b + \beta = (a + b) + \underbrace{\alpha + \beta}_{\in I} \in [a + b] \\ x \cdot y &= a \cdot b + \underbrace{a \cdot \beta + b \cdot \alpha + \alpha \cdot \beta}_{\in I} \in [a \cdot b]. \end{aligned}$$

□

Definition 2.18. *The ring R/I is called the **quotient ring**.*

Example 2.19.

1. *Let $R = \mathbb{Z}$ and $I = I_n = \langle n \rangle$. Then, $R/I = \mathbb{Z}/I_n = \mathbb{Z}_n$.*
2. *Let $R = \mathbb{R}[x]$ the ring of polynomials in $\mathbb{R}$ and $I := \langle x^2 - x \rangle$. Then, in R/I the equivalence classes $[x^2]$ and $[x]$ are equal. Therefore, R/I consists of the equivalence classes $[a]$, $[x]$ and $[x + a]$ with $a \in R$. In other words, R/I can be seen as a polynomial ring where only degrees 0 and 1 occur.*

2.2.3 Fields

Definition 2.20. *Let $(R, +, \cdot)$ be a ring. This structure is called a **field** if $(R \setminus \{0\}, \cdot)$ is a group. In other words, a ring is a field if each element in R except the additive neutral element 0 has a multiplicative inverse.*

*Let $\mathbb{F}$ be a field and 1 denote its identity element regarding multiplication. If $c \geq 1$ exists such that $0 = c \cdot 1 = \underbrace{1 + \ldots + 1}_{c \text{ times}}$, then the smallest constant $c \geq 0$ with this property is defined to be the **characteristic** of $\mathbb{F}$. If no such c exists, then the characteristic of $\mathbb{F}$ is defined to be zero.*

Example 2.21. *The set of rational numbers $\mathbb{Q}$, the set of real numbers $\mathbb{R}$, and the set of complex numbers $\mathbb{C}$ are perhaps the widest known fields. All of them have characteristic 0. The smallest field is the ring $\mathbb{Z}_2$, which contains only two elements (see also Example 2.11). It has characteristic 2.*

Proposition 2.22. *The ring $\mathbb{Z}_n$ is a field if and only if n is a prime number.*

Proof. As we already know that $\mathbb{Z}_n$ is a ring, it suffices to show that each non-zero element has a multiplicative inverse if and only if n is a prime number.

Let $x \in \mathbb{Z}_n$ be an element which has a multiplicative inverse. Hence, an element $y \in \mathbb{Z}_n$ exists with $x \cdot y = 1 \mod n$. Thus, there is a $k \in \mathbb{Z}$ such that $x \cdot y = k \cdot n + 1$ or, equivalently, $x \cdot y - k \cdot n = 1$. Let $g := \gcd(x, n)$ be the greatest common divisor of x and n. As g divides x and n, it divides also $1 = x \cdot y - k \cdot n$. This shows that g necessarily is equal to 1.

On the other hand, $\gcd(x, n) = 1$ is also a sufficient condition. If the greatest common divisor is equal to 1, then integers $\alpha, \beta \in \mathbb{Z}$ exists such that $\alpha \cdot x + \beta \cdot n = 1$. Hence, $\alpha \cdot x = 1 \mod n$.

Altogether, $x \neq 0$ has a multiplicative inverse if and only if it is co-prime to n. Therefore, $\mathbb{Z}_n$ is a field if and only if all $1 \leq x \leq n-1$ are co-prime to n. This is only true if and only if n is a prime. □

2.2.4 Multivariate polynomials

An important ring is the ring of (multivariate) polynomials over a field $\mathbb{F}$. Therefor, we fix some notations:

Definition 2.23. *For $E := (e_1, \ldots, e_n) \in \mathbb{N}^n$ and variables $X := (x_1, \ldots, x_n)$, we define the **monomial***

$$m_E(X) := m_{(e_1,\ldots,e_n)}(x_1, \ldots, x_n) := X^E := x_1^{e_1} \cdot \ldots \cdot x_n^{e_n} \tag{2.2.2}$$

*with the special case $m_0(X) = X^0 := 1$. A **term** is an expression $c \cdot X^E$ with $c \in \mathbb{F}$. The ring $\mathbb{F}[x_1, \ldots, x_n]$ is defined as the set of multivariate polynomials*

$$f(x_1, \ldots, x_n) = \sum_{E \in \mathbb{N}^n} f_E \cdot X^E, \quad f_E \in \mathbb{F}, \tag{2.2.3}$$

where only finitely many coefficients f_E are not equal to zero. The addition and multiplication are defined by

$$\begin{aligned} f(x_1, \ldots, x_n) + g(x_1, \ldots, x_n) &:= \sum_{E \in \mathbb{N}^n} (f_E + g_E) \cdot X^E \\ f(x_1, \ldots, x_n) \cdot g(x_1, \ldots, x_n) &:= \sum_{A \in \mathbb{N}^n} \sum_{B \in \mathbb{N}^n} f_A \cdot g_B \cdot x_1^{a_1+b_1} \cdot \ldots \cdot x_n^{a_n+b_n} \end{aligned}$$

with $A = (a_1, \ldots, a_n)$ and $B = (b_1, \ldots, b_n)$.

Definition 2.24. *Let $f(x_1, \ldots, x_n) = \sum_{E \in \mathbb{N}^n} f_E X^E \not\equiv 0$ be given. The **degree** of f is defined by*

$$\deg(f) := \max_E \{|E| \mid f_E \neq 0\} \tag{2.2.4}$$

where $|E| = |(e_1, \ldots, e_n)| = \sum_{i=1}^n e_i$.

Example 2.25. *For $f(x_1, x_2, x_3) := x_1 \cdot x_2 + x_2^2 \cdot x_3$, the degree of f is equal to three.*

Remark 2.26. *To distinguish between indeterminates and elements from a field, we will use bold letters in the second case. Furtheron, capital letters stand for tuples (or vectors) whereas small letters stand for single elements or indeterminates. If $f : \mathbb{F}^{n_1} \times \ldots \times \mathbb{F}^{n_\iota} \to \mathbb{F}$, we sometimes express f by $f(X_1, \ldots, X_\iota)$ where $X_i := (x_{i,1}, \ldots, x_{i,n_i})$.*

Example 2.27. *For $f : \mathbb{F}^n \to \mathbb{F}$, $f(x) \in \mathbb{F}[x]$ stands for a polynomial f with the variable x whereas $f(\boldsymbol{x})$ denotes the evaluation of f on the input $\boldsymbol{x} \in \mathbb{F}$.*

For $f : \mathbb{F}^{n_1} \times \ldots \times \mathbb{F}^{n_\iota} \to \mathbb{F}$, $f(X_1, \ldots, X_\iota)$ refers to a multivariate polynomial in the unknowns $X_1 = (x_{1,1}, \ldots, x_{1,n_1})$, ..., $X_\iota = (x_{\iota,1}, \ldots, x_{\iota,n_\iota})$ whereas $f(\boldsymbol{X_1}, \ldots, \boldsymbol{X_\iota})$ denotes the image of $(\boldsymbol{X_1}, \ldots, \boldsymbol{X_\iota}) \in \mathbb{F}^{n_1} \times \ldots \times \mathbb{F}^{n_\iota}$ under the function f.

Definition 2.28. *A polynomial $p(x) = \sum_{i=0}^d c_i \cdot x^i \in \mathbb{F}[x]$ of degree d is called **monic** if $c_d = 1$.*

Theorem 2.29. *Let $\mathbb{F}$ be a field and $\{0\} \neq I \subseteq \mathbb{F}[x]$ be a an ideal. Then, I is a principal ideal, generated by one unique monic polynomial $m(x) \in \mathbb{F}[x]$ which has the lowest degree of all polynomials in I.*

Proof. Let $d := \min\{\deg(f(x)) | f(x) \in I \setminus \{0\}\}$ and $m(X) \in I$ be a monic polynomial with $\deg(m(x)) = d$. Next, let $m'(x) \in I$ be an arbitrary monic polynomial, which has the same degree d. If $m(x) \neq m'(x)$, then $m(x) - m'(x)$ is a non-zero polynomial in I of degree $< d$ what contradicts the definitions of d and $m(x)$. Hence, $m(x)$ is unique.

Now, let $f(x) \in I$ be arbitrary. Then, $p(x), r(x) \in \mathbb{F}[x]$ exist such that

$$f(x) = p(x) \cdot m(x) + r(x)$$

with $r(x) \equiv 0$ or $\deg(r(x)) < \deg(m(x))$. But as we have seen that no non-zero polynomials exist with a degree strictly smaller than $\deg(m(x))$, it must be $r(x) \equiv 0$. This shows that $f(x)$ is divisible by $m(x)$ and therefore $I = \langle m(x) \rangle$. □

Definition 2.30. *A polynomial $f(x) \in \mathbb{F}[x]$ is called **irreducible**, if for any two polynomials $g(x), h(x) \in \mathbb{F}[x]$ with $g(x) \cdot h(x) = f(x)$, it is that either $g(x) \equiv c$ or $h(x) \equiv c$ for $c \in \mathbb{F}$.*

The property of being irreducible depends on the base field $\mathbb{F}$, as the following example shows:

Example 2.31. *The polynomial $x^2 - 2 \in \mathbb{Q}[x]$ is irreducible, but $x^2 - 2 = (x - \sqrt{2}) \cdot (x + \sqrt{2}) \in \mathbb{R}[x]$ is not.*

Likewise, $x^2 + 1 \in \mathbb{R}[x]$ is irreducible, but $x^2 + 1 = (x + i) \cdot (x - i) \in \mathbb{C}[x]$ is not.

2.2.5 Extension fields and finite fields

This subsection is on extension fields and finite fields. We start with the definition of extension fields:

Definition 2.32. *Let $\mathbb{F} \subseteq \mathbb{F}'$ be two fields. $\mathbb{F}'$ is called an **extension field** of $\mathbb{F}$. Furthermore, for $\mathbb{F}$ a field and $\alpha_1, \ldots, \alpha_n$ be elements not necessarily in $\mathbb{F}$, we define by $\mathbb{F}[\alpha_1, \ldots, \alpha_n]$ the set of all (formal) finite sums*

$$\sum_{E=(e_1,\ldots,e_n)\in\mathbb{N}^n} c_E \cdot \alpha_1^{e_1} \cdot \ldots \cdot \alpha_n^{e_n}$$

with $c_E \in \mathbb{F}$.[4]

Example 2.33. *$\mathbb{R}$ is an extension field of $\mathbb{Q}$ and $\mathbb{C}$ is an extension field of $\mathbb{Q}$ and $\mathbb{R}$.*

Definition 2.34. *Let $\mathbb{F} \subseteq \mathbb{F}'$ be two fields. $\alpha \in \mathbb{F}'$ is called **algebraic** over $\mathbb{F}$ if a non-zero polynomial $f(x) \in \mathbb{F}[x]$ exists such that $f(\alpha) = 0$.*

Example 2.35. *As $f(x) = x^2 - 2 \in \mathbb{Q}[x]$ gives zero on $\sqrt{2} \in \mathbb{R}$, $\sqrt{2}$ is algebraic over $\mathbb{Q}$.*

Theorem 2.36. *Let $\mathbb{F} \subseteq \mathbb{F}'$ be two fields and $\alpha \in \mathbb{F}'$ be algebraic over $\mathbb{F}$. There exists one unique monic polynomial $m(x) \in \mathbb{F}[x]$ with the lowest degree such that $m(\alpha) = 0$. $m(x)$ is called the **minimal polynomial** of α.*

Proof. Consider the set $I := \{f(x) \in \mathbb{F}[x] \mid f(\alpha) = 0\}$. One sees easily that I is an ideal in $\mathbb{F}[x]$. Thus, by Theorem 2.29, $I = \langle m(x) \rangle$ where $m(x)$ fulfills the conditions above. □

In Proposition 2.39, we encountered fields where the number of elements is finite, namely the fields $\mathbb{Z}_p$ with p being a prime. We will describe now how extension fields of $\mathbb{Z}_p$ can be defined. But first, we motivate the theory by a similar construction.

[4] This corresponds to the definition of $\mathbb{F}[x_1, \ldots, x_n]$ in Definition 2.23.

Example 2.37. *As usual, we refer by $\mathbb{R}$ to the field of real numbers. As it is widely known, the equation $x^2 + 1 = 0$ has no solution in $\mathbb{R}$. However, this has mathematicians not kept from* defining *a root of this polynomial, namely the so-called imaginary number i. That is, i is an identifier for a solution of $x^2 - 1 = 0$. Likewise, one can say that $x^2 - 1$ is the minimal polynomial of i. The interesting part is that one can compute with i like with any other real number x, as its behaviour is completely defined by its minimal polynomial. For example, take the two expressions $x + i \cdot y$ and $u + i \cdot v$ with $x, y, u, v \in \mathbb{R}$. Then, $(x + i \cdot y) + (u + i \cdot v) = (x + u) + i \cdot (y + v)$ and*

$$(x + i \cdot y) \cdot (u + i \cdot v) = x \cdot u + i \cdot (x \cdot v + y \cdot u) + i^2 \cdot y \cdot v.$$

But the last can be simplified. As we know that i is by definition a root of $x^2 - 1$, it holds that $i^2 = -1$. Thus, one can replace i^2 by -1 and gets

$$(x + i \cdot y) \cdot (u + i \cdot v) = x \cdot u + i \cdot (x \cdot v + y \cdot u) - \cdot y \cdot v.$$

Probably, the content of the example was already known to most of the readers. It is more or less common knowledge that appending i, the root of $x^2 - 1$, to the field $\mathbb{R}$ results in an extension field of $\mathbb{R}$, namely the field of complex numbers $\mathbb{C}$.

However, what we want to emphasize is the basic idea behind: Starting from a field $\mathbb{F}$ and polynomial $p(x)$ where one root is not in $\mathbb{F}$, one appends this "missing" root α to $\mathbb{F}$. It turns out that $\mathbb{F}[\alpha]$ is an extension field of $\mathbb{F}$. This is important as the same approach allows to construct extension fields to other fields, in particular to $\mathbb{Z}_p$.

Theorem 2.38. *Let p be a prime and $f(x) = f_0 + \ldots + f_{d-1} \cdot x^{d-1} + x^d \in \mathbb{Z}_p[x]$ be an irreducible polynomial of degree d. Then, if α is an identifier for a root of $f(x)$, then $\mathbb{Z}_p[\alpha, \alpha^2, \ldots, \alpha^{d-1}]$ is a field with p^d elements.*

Proof. First, we take a look at the operations in $\mathbb{Z}_p[\alpha, \alpha^2, \ldots, \alpha^{d-1}]$. Any element can be expressed as $c_0 + c_1 \cdot \alpha + \ldots + c_{d-1}\alpha^{d-1}$ with $c_i \in \mathbb{Z}_p$. This shows that exactly p^d different elements exist. The addition with a second element $c'_0 + \ldots + c_{d-1}\alpha^{d-1}$ is defined componentwise, that is

$$\left(c_0 + c_1 \cdot \alpha + \ldots + c_{d-1}\alpha^{d-1}\right) + \left(c'_0 + \ldots + c'_{d-1}\alpha^{d-1}\right) = (c_0 + c'_0) + \ldots + (c_{d-1} + c'_{d-1}) \cdot \alpha^{d-1}.$$

For multiplication, we consider only the product with α as the rest follows by induction. Hereby, we make use of the fact that $0 = p(\alpha) = p_0 + \ldots + p_{d-1} \cdot \alpha^{d-1} + \alpha^d$. Thus, it holds that $\alpha^d = -(p_0 + \ldots + p_{d-1} \cdot \alpha^{d-1})$ and therefore:

$$\begin{aligned} \alpha(c_0 + c_1\alpha + \ldots + c_{d-1}\alpha^{d-1}) &= c_0\alpha + c_1\alpha^2 + \ldots + c_{d-1}\alpha^d \\ &= c_0\alpha + c_1\alpha^2 + \ldots - c_{d-1} \cdot (p_0 + \ldots + p_{d-1}\alpha^{d-1}). \end{aligned}$$

Thus, addition and multiplication are well-defined in $\mathbb{Z}_p[\alpha, \alpha^2, \ldots, \alpha^{d-1}]$. The ring properties follow easily from the properties of $\mathbb{Z}_p$.

What remains is to show is that $\mathbb{Z}_p[\alpha, \alpha^2, \ldots, \alpha^{d-1}]$ is actually a field. As any $\boldsymbol{x} \in \mathbb{Z}_p$ has its inverse $\boldsymbol{x}^{-1}$, it suffices to show that α has an inverse. At first, it holds that the constant term p_0 of $p(x)$ is not equal to zero. Otherwise, $p(x)$ could be divided by x what is a contradiction to its irreducibility. Because of $p_1 \cdot \alpha + \ldots + p_{d-1} \cdot \alpha^{d-1} + \alpha^d = -p_0$, the element $(-p_0)^{-1} \cdot (p_1 + \ldots + p_{d-1} \cdot \alpha^{d-2} + \alpha^{d-1})$ is an inverse of α. □

The theorem explained how to construct fields with p^d elements. The following proposition shows that such a field is unique up to isomorphism.

Proposition 2.39. *[LidN86, Chapter 2] Let $\mathbb{F}$ be a **finite field**, that is a field with a finite number of elements. Then, $|\mathbb{F}| = p^d =: q$ for $p, d \geq 1$ and p being a prime number. Any two finite fields of the same size q are isomorphic. This allows to speak of* the *field $\mathbb{F}_q$.*

*Furthermore, in finite fields the so-called **field equations** are valid: $\boldsymbol{x}^q = \boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{F}_q$.*

Apart from the field equations and the associated polynomial $x^q - x$, there exist a group of other polynomials which will be important later. Let $p(x) \in \mathbb{F}_q[x]$ be an irreducible polynomial and $\alpha \in \mathbb{F}_{q^d}$ be one of its roots. By the field equation, it holds that $\alpha^{q^d-1} - 1 = 0$. That is α is a root of the polynomial $x^{q^d-1} - 1$. As this holds for any root of $p(x)$, this implies that $p(x)$ divides $x^{q^d-1} - 1$.

Definition 2.40. *Let $\mathbb{F} = \mathbb{F}_q$ be a finite field. $f(x) \in \mathbb{F}_q[x]$ is called **primitive** if $f(x)$ divides the polynomial $x^{q^d-1} - 1$ but none of the polynomials $x^e - 1$ with $e < q^d - 1$.*

For the sake of computability and representability, finite fields are the first choice in the context of computers. Hence, it is no surprise that (as far as we know) all cryptosystems are defined over finite fields.

Of specific importance is the finite field $\mathbb{F}_2$ as the elements can be represented by one bit. Besides, the operations in $\mathbb{F}_2$ have their counterpart as basic bit operations. The addition in $\mathbb{F}_2$ corresponds to the XOR-operation and the multiplication to the AND-operation. Thus, calculations in $\mathbb{F}_2$ can be efficiently performed. Following widely used conventions, we will express the addition in $\mathbb{F}_2$ sometimes by the symbol for XOR, namely $\oplus$.

Remark 2.41. *Describing finite fields in detail, as we have done it here, might seem exaggerated. However, there are two reasons for our approach. Firstly, the interplay between extension fields and roots of polynomials will become important later in the context of linear feedback shift registers when*

we will analyze their minimal polynomials (Section 5.2.2). Thus, this section might serve as a warm-up.

Secondly, finite fields are widely used in cryptography, so it is important to get used to them. Unfortunately, we had the impression that many people have their difficulties with finite fields in the beginning. On the other hand, the concept of $\mathbb{C}$ (computing with it and its relation to $\mathbb{R}$) is normally no problem for people interested in cryptography. As both principles are related, pointing out this similarity might help readers who are new to finite fields to get familiar with this subject.

2.3 Functions over finite fields

In this section, we provide some basic facts on functions over finite fields and introduce some notations. We start with some trivial remarks on functions.

Let $\mathbb{Z}_q$ be the ring of integers modulo q. Each $e \in \mathbb{Z}_q$ can be identified with exactly one integer e' in the range $0, \ldots, q-1$. This allows to define the absolute value of e by $|e| := e'$. This notion can be extended to $\mathbb{Z}_q^n$ by $|(e_1, \ldots, e_n)| := \sum_{i=1}^n |e_i|$ for $E = (e_1, \ldots, e_n) \in \mathbb{Z}_q^n$. We refer to this value also as the **weight** $wt(E)$ of $E = (e_1, \ldots, e_n)$. By $\mathbb{F}$, we denote a finite field. If we want to specify the number q of elements in $\mathbb{F}$, we write $\mathbb{F}_q$ instead. Furthermore, we use $\mathbb{F}^n$ and $\mathbb{F}_q^n$ for $n \geq 1$ to specify the vector spaces of dimension n over the finite fields $\mathbb{F}$ and $\mathbb{F}_q$, respectively.

In this section, we consider functions $\mathbb{F}^n \to \mathbb{F}^m$ for $n, m \geq 1$. As each $f : \mathbb{F}^n \to \mathbb{F}^m$ can be splitted into its **component functions** $f = (f_1, \ldots, f_m)$ with $f_i : \mathbb{F}^n \to \mathbb{F}$, we will mostly be interested in the case $m = 1$.

In cryptography, it is often desired for a function that each output has the same amount of preimages:

Definition 2.42. *A function $\mathbb{F}_q^n \to \mathbb{F}_q$ is called **balanced**, if each image $z \in \mathbb{F}_q$ has the same number of preimages. This means that $|f^{-1}(z)| = q^{n-1}$ for each value $z \in \mathbb{F}_q$.*

In the case of finite fields, it holds the special property that *every* function $\mathbb{F}^n \to \mathbb{F}$ can be expressed by a multivariate polynomial in n unknowns:

Theorem 2.43. *Each function $f : \mathbb{F}_q^n \to \mathbb{F}_q$, $n \geq 1$, has a unique expression*

$$f(X) = f(x_1, \ldots, x_n) = \sum_{E \in \{0,\ldots,q-1\}^n} f_E \cdot X^E \tag{2.3.5}$$

*in $\mathbb{F}[x_1, \ldots, x_n]$. (2.3.5) is called the **algebraic normal form** (ANF) of f.*

Proof. Fix an arbitrary function $f : \mathbb{F}_q^n \to \mathbb{F}_q$. We show that f can be expressed as a multivariate polynomial $f^* \in \mathbb{F}_q[x_1, \ldots, x_n]$. To do so, we introduce the functions

$$\delta_{(y_1,\ldots,y_n)}(x_1, \ldots, x_n) = \delta_Y(X) := \prod_{i=1}^n (1 - (x_i - y_i)^{q-1}).$$

Each variable x_i occurs only in powers of $q-1$ or less.

Now, let $x \in \mathbb{F}$ and look at x^{q-1}. As we know from the field equations, it holds that $0 = x^q - x = x \cdot (x^{q-1} - 1)$. Thus, if $x \neq 0$, then the second factor needs to be zero, which implies that $x^{q-1} = 1$ if $x \neq 0$. Therefore,

it holds that $\delta_{\boldsymbol{Y}}(\boldsymbol{X}) = 1$ if and only if $\boldsymbol{X} = \boldsymbol{Y}$. Consider now the following multivariate polynomial

$$f^*(x_1, \ldots, x_n) := \sum_{\boldsymbol{Y} \in \mathbb{F}_q^n} f(\boldsymbol{Y}) \cdot \delta_{\boldsymbol{Y}}(X). \tag{2.3.6}$$

For each $\boldsymbol{X} \in \mathbb{F}^n$, all terms of the sum in $f^*(\boldsymbol{X})$ are equal to zero except the product $f(\boldsymbol{X}) \cdot \delta_{\boldsymbol{X}}(\boldsymbol{X}) = f(\boldsymbol{X})$. Hence, it holds that $f^*(\boldsymbol{X}) = f(\boldsymbol{X})$. Thus, f^* is a representation of f.

The uniqueness follows from a simple counting argument. Observe that exactly $|\mathbb{Z}_q^n| = q^n$ different monomials X^E with $E \in \mathbb{Z}_q^n$ exist, implying that the total number of different expressions (2.3.5) is q^{q^n}.

On the other hand, any function $f : \mathbb{F}_q^n \to \mathbb{F}$ is uniquely defined by its outputs, i.e. $f(\boldsymbol{X}) \in \mathbb{F}_q$, $\boldsymbol{X} \in \mathbb{F}_q^n$. Thus, the number of functions $\mathbb{F}_q^n \to \mathbb{F}_q$ equals likewise q^{q^n}. Therefore, the algebraic normal form has to be unique. □

The uniqueness of the algebraic normal form (2.3.5) allows to define the notion of the degree of a function $f : \mathbb{F}^n \to \mathbb{F}$. We will later refer frequently to the number of monomials in $\mathbb{F}_q[x_1, \ldots, x_n]$ of degree $\leq d$, for which we introduce the identifier $\mu_q(n, d)$. For $q = 2$, it holds that $\mu_2(n, d) = \sum_{i=0}^{d} \binom{n}{i}$. Later, we will be interested if functions with certain properties exists such that the degree is less than or equal to a given value d.

Remark 2.44. *For the rest of thesis, we will always implicitly identify each function $\mathbb{F}_q^n \to \mathbb{F}_q$ with its unique expression as a multivariate polynomial with exponents in $\{0, \ldots, q-1\}$. Thus, whenever we refer to the degree of f we speak of the degree of this specific representation as a multivariate polynomial. More formally speaking, we assume that all computations are done in the ring $\mathbb{F}_q[x_1, \ldots, x_n] / \langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle$ (see also Example 2.19).*

Furthermore, we will implicitly use the fact that $p \cdot f(X) = 0$ for all $f(X) \in \mathbb{F}_{p^d}[X]$. In particular, it holds that $f(X) + f(X) = 0$ for all $f(X) \in \mathbb{F}_2[X]$, wherefore we will sometimes write $+f(X)$ instead of $-f(X)$ and vice versa.

Example 2.45. *We consider $\mathbb{F}_3[x_1, x_2]$, the ring of multivariate polynomials in two variables over $\mathbb{F}_3$. Let $f(x_1, x_2) := x_1^2 \cdot x_2$ and $g(x_1, x_2) := x_1 \cdot x_2^2$. As stated above, the product $f \cdot g$ would be identified with $x_1 \cdot x_2$ as $x_1^2 \cdot x_2 \cdot x_1 \cdot x_2^2 = x_1^3 \cdot x_2^3$ and the facts that $x_1^3 = x_1$ and $x_2^3 = x_2$ due to the field equations. Thus, the degree of $f \cdot g$ is equal to two. This shows that (unlike to the case of non-finite fields), the degree of the product of two functions can be less than the degrees of its factors. This property will play an important role later in Chapter 4 when we treat the problem of the existence of certain functions with a low degree.*

In some cases, we will encounter symmetric functions:

Definition 2.46. *$f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ is called **symmetric**, if for any permutation π on the set $\{1, \ldots, n\}$ it holds that*

$$f(x_1, \ldots, x_n) = \pi(f) := f(x_{\pi(1)}, \ldots, x_{\pi(n)}). \tag{2.3.7}$$

Example 2.47. *Let $n := 3$ and $f(x_1, \ldots, x_3) := x_1 + x_2$. f is not symmetric as for $\pi := [1 \to 1, 2 \to 3, 3 \to 2]$ the function $\pi(f) = x_1 + x_3$ is different from f.*

The probably most trivial example for a symmetric function is $f(x_1, \ldots, x_n) = x_1 + \ldots + x_n$.

2.4 Linear feedback shift registers

On the search for valuable keystream generators, linear feedback shift registers (LFSRs) turned out to be a good choice. Despite of their hardware friendly implementation, at least if $\mathbb{F} = \mathbb{F}_2$, they can be used to produce sequences with good statistical properties. In this chapter, we recapitulate some basic facts about LFSRs and lay the foundation for further analysis in following chapters. For further reading, we recommend the book by Lidl and Niederreiter [LidN86].

Definition 2.48. *Let $\mathbb{F}$ be a finite field. A **linear feedback shift registers** of **length** n is a finite state machine with an internal state of size n and a linear feedback function $\Lambda := \sum_{i=1}^{n} \lambda_{i-1} \cdot x_i \in \mathbb{F}[x_1, \ldots, x_n]$. An LFSR is regularly clocked. At each clock, a specific entry of the internal state is output, then the internal state is updated according to Λ. Let $S_0 := (s_0, \ldots, s_{n-1}) \in \mathbb{F}^n$ denote the content of the internal state at the beginning and $L(x_1, \ldots, x_n) := (x_2, \ldots, x_n,\ \Lambda(x_1, \ldots, x_n))$. Then, at each clock, two operations are executed:*

1. *Output one entry of the internal state.*
2. *Update the internal state S_t to S_{t+1} with L:*

$$S_{t+1} := L(S_t) = (s_{t+1}, \ldots, s_{t+n-1}, \Lambda(S_t)) = (s_{t+1}, \ldots, s_{t+n-1}, \sum_{i=0}^{n-1} \lambda_i \cdot s_{t+i}). \quad (2.4.8)$$

*The output at clock t is usually defined to be s_t, that is the left-most entry of S_t. We call the sequence $(s_t)_{t \geq 0}$ the **LFSR sequence**. Sometimes, we will identify L with its matrix expression[5],*

$$L := \begin{pmatrix} 0 & \ldots & \ldots & 0 & \lambda_0 \\ 1 & \ddots & & \vdots & \lambda_1 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \ldots & 0 & 1 & \lambda_{n-1} \end{pmatrix}, \quad (2.4.9)$$

*and set $S_{t+1} := S_t \cdot L$. Particularly, it holds that $S_t := S_0 \cdot L^t$. The matrix L is also called the **feedback matrix**. Observe that $\lambda_0 \neq 0$ would imply that the update of the internal state is reversible.*

Figure (2.4.3) displays a schematic figure of an LFSR. LFSRs can be used

[5] For simplicity, we refer to both the linear mapping and the corresponding matrix with L.

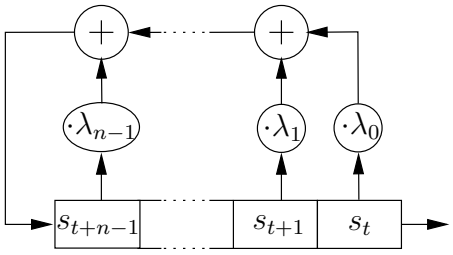

Figure 2.4.3: Schematic picture of an LFSR

to produce streams $(s_t)_{t\geq 0}$ of arbitrary length. The advantage of LFSRs is that they can be implemented efficiently in hardware (at least for the case $\mathbb{F} = \mathbb{F}_2$). That makes them particularly interesting for restricted devices as mobile phones. Another advantage is that LFSRs and their sequences are mathematically well understood.

Remark 2.49. *We will from now on assume that L is regular, that is $\lambda_0 \neq 0$.*

Definition 2.50. *A sequence $\mathcal{S} = (s_t)_{t\geq 0}$ is called **periodic**, if $\rho \geq 1$ exists such that $s_t = s_{t+\rho}$ for all $t \geq 0$. ρ is called the **period** of $\mathcal{S}$. If no other value $\rho' < \rho$ is a period of $\mathcal{S}$, we call ρ the **least period**.*

Lemma 2.51. *Let $\mathcal{S}$ be an LFSR-sequence over a finit field $\mathbb{F}_q$. Then $\mathcal{S}$ is periodic with a period $1 \leq \rho \leq q^n$.*

Proof. Let $S_t \in \mathbb{F}_q^n$ denote the internal states of the LFSR. As the number of elements in $\mathbb{F}_q^n$ is finite, namely q^n, there have to be two clocks $0 \leq t_0 < t_1 \leq q^n$ such that $S_{t_0} = S_{t_1}$. As L was assumed to be regular, it holds that

$$S_{t_0-1} = L^{-1}(S_{t_0}) = L^{-1}(S_{t_1}) = S_{t_1-1}.$$

This shows that $S_0 = S_{t_1-t_0}$ and therefore $\mathcal{S}$ is periodic with period $t_1 - t_0 \leq q^n$. □

Example 2.52. *Let $\mathbb{F} := \mathbb{F}_2$, $n := 2$ and $L := \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$. Then, initializing the initial state S_0 to $(1,1)$ gives*

t	0	1	2	3	4	5	6	7	8	...
S_t	1	1	0	1	1	0	1	1	0	...
	1	0	1	1	0	1	1	0	1	

yielding the bit stream $1,1,0,1,1,0,1,1,0,\ldots$ with a period of 3. Here, each entry in the second row displays one value of S_t. For example, it holds that $S_0 = (1,1)$, $S_1 = (1,0)$, $S_2 = (0,1)$ and so on.

If $\mathbb{F} = \mathbb{F}_3$, then the same values of n and L result in

t	0	1	2	3	4	5	6	7	8	9	10	...
S_t	1	1	2	0	2	2	1	0	1	1	2	...
	1	2	0	2	2	1	0	1	1	2	0	

,

and the stream $1, 1, 2, 0, 2, 2, 1, 0, 1, 1, 2, \ldots$. *In this case, the sequence has a period of 8.*

Definition 2.53. *Let* $\mathcal{S} := (s_t)_a^b$ *with* $a \leq b$ *denote a sequence* $s_a, \ldots, s_b$ *over a ring* R *where* $b = \infty$ *is possible.* $\mathcal{S}$ *is called a* ***linear recurring sequence*** *if* $r \geq 1$ *and* $\lambda_0, \ldots, \lambda_{r-1} \in R$*, not all zero, exist such that*

$$s_{t+r} = \sum_{i=0}^{r-1} \lambda_i \cdot s_{t+i} \tag{2.4.10}$$

holds for $a \leq t \leq b - r$.

By equation (2.4.8), it holds $s_{t+r} - \sum_{i=1}^{r} \lambda_i \cdot s_{t+i-1} = 0$ for the outputs of an LFSR for any clock $t \geq 0$. This shows that an LFSR-sequence is a linear recurring sequence as well.

By Lemma 2.51 any LFSR-sequence is periodic with period $\rho \leq q^n$. For cryptographic reasons, one is interested in sequences with periods a big as possible. However because of $L(0, \ldots, 0) = (0, \ldots, 0)$, the all-zero initial state $S_0 := (0, \ldots, 0)$ produces the all-zero sequence, that is $S_t = (0, \ldots, 0)$ for all $t \geq 0$. In particular, if $S_t = (0, \ldots, 0)$ for one t, then $S_{t'} = (0, \ldots, 0)$ for all $t' \geq t$. Therefore the best one can hope for is a sequence of period $q^n - 1$ where every value in $\mathbb{F}_q^n \setminus \{(0, \ldots, 0)\}$ occurs once before the same state reappears. We will call such a sequence a **maximum sequence**.

Consider two different maximum sequences. As the successor $L(S)$ of each state S is uniquely defined, the order of the states must be the same. This shows that each maximum sequence $(s_t)_{t \geq 0}$ can be transformed in any other maximum sequence $(s'_t)_{t \geq 0}$ by simply shifting the indices, i.e. $\delta \in \mathbb{Z}$ exists such that $s_t = s'_{t+\delta}$ for all t. Particularly, the property of a sequence being maximum is independent of the chosen initial value, as long as it is not equal to the zero vector. Consequently, it depends only on the linear feedback function L whether the sequence $(s_t)_{t \geq 0}$ is maximum or not. In the following, we derive a criterion on L which is sufficient and necessary to make maximum sequences possible.

Definition 2.54. *Let* $\mathcal{S} := (s)_1^\infty$ *be a linear recurring sequence over an arbitrary field* $\mathbb{F}$*. A polynomial* $p(x) = x^r + \sum_{i=0}^{r-1} \lambda_i \cdot x^i \in \mathbb{F}[x]$ *is called a* ***characteristic polynomial*** *of* $\mathcal{S}$ *if (2.4.10) holds for all* t.

For a polynomial $p(x) = \sum_i c_i x^i$*, we define its* ***companion matrix*** L_p *by*

$$L_p := \begin{pmatrix} 0 & \dots & \dots & 0 & \lambda_0 \\ 1 & \ddots & & \vdots & \lambda_1 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & 1 & \lambda_{n-1} \end{pmatrix}$$

Observe that the companion matrix of a characteristic polynomial is just the feedback matrix.

The notion of characteristic polynomials is a useful tool to examine the properties of LFSRs and linear recurring sequences. However, a sequence $\mathcal{S}$ has more than one characteristic polynomial:

Example 2.55. *Let* $\mathbb{F} = \mathbb{F}_2$ *and* $\mathcal{S} := (110110110\dots) = (\overline{110})$*. As* $\mathcal{S}$ *has a period of three, a canonical characteristic polynomial is* $x^3 + 1$*.On the other hand, it holds for all* $t \geq 0$ *that* $s_t + s_{t+1} + s_{t+2} = 0$*. Hence, another characteristic polynomial is* $p'(x) = x^2 + x + 1$*.*

Now, consider $\mathcal{S}' := (111\dots) = (\overline{1})$*. Although* $\mathcal{S}$ *and* $\mathcal{S}'$ *are different, both share the same characteristic polynomial* $x^3 + 1$*.*

As the previous example shows, a characteristic polynomial of a linear recurring sequence is not sufficient to characterize the sequence. However, we will see next that amongst all characteristic polynomial exists one unique polynomial $m(x)$ which helps to examine and to distinguish linear recurring sequences.

Theorem 2.56. *Let* $\mathcal{S} := (s_t)$ *be a linear recurring sequence over an arbitrary field* $\mathbb{F}$*. The set* $Charpols(\mathcal{S})$ *of characteristic polynomials of* $\mathcal{S}$ *is a principal ideal in the ring* $\mathbb{F}[x]$*. More formally, it holds that* $Charpols(\mathcal{S}) = \langle m(x) \rangle$ *for a polynomial* $m(x) \in \mathbb{F}[x]$*.*

Proof. We first show that $Charpols(\mathcal{S})$ fulfills all three conditions to be an ideal:

1. $0 \in Charpols(\mathcal{S})$
2. $\forall p(x), p'(x) \in Charpols(s) : \; p(x) + p'(x) \in Charpols(s)$
3. $\forall p(x) \in Charpols(s), q(x) \in \mathbb{F}[x] : \; q(x) \cdot p(x) \in Charpols(\mathcal{S})$

The first point is obvious as the all-zero polynomial easily fulfills equation (2.4.10). Now let $p(x) = \sum_{i=0}^{n} \lambda_i \cdot x^i$ and $p'(x) = \sum_{i=0}^{n'} \lambda'_i \cdot x^i$ be two elements of $Charpols(\mathcal{S})$. This implies that $\sum_{i=0}^{n} \lambda_i \cdot s_{t+i} = 0$ and $\sum_{i=0}^{n'} \lambda'_i \cdot s_{t+i} = 0$ for all t.

In particular the sum of both expressions is zero again which shows that $p(x) + p'(x)$ is a characteristic polynomial again.

Finally, we consider $p(x) = \sum_{i=0}^{r} \lambda_i \cdot x^i \in Charpols(\mathcal{S})$ and $q(x) = \sum_{j=0}^{n} d_j \cdot x^j \in \mathbb{F}[x]$. The product of $p(x)$ and $q(x)$ can be written as $p(x) \cdot q(x) = \sum_{j=0}^{n} d_j \cdot (\sum_{i=0}^{r} \lambda_i \cdot x^{i+j})$. Furthermore, one can easily see that

$$\sum_{j=0}^{n} d_j \cdot \underbrace{(\sum_{i=0}^{r} \lambda_i \cdot s_{t+i+j})}_{=0} = 0$$

for all t. The fact that the terms inside brackets equal to zero is due to that $p(x)$ is a characteristic polynomial. This shows that $Charpols(\mathcal{S})$ is an ideal and hence, by Theorem 2.29, a principal ideal. □

Observe that each multiple of a characteristic polynomial is a characteristic polynomial. On the other hand, amongst all characteristic polynomials exists one which is the smallest in terms of the degree, called the minimal polynomial of the sequence.

Definition 2.57. *Let $\mathcal{S} := (s_t)_0^\infty$ be a linear recurring sequence over an arbitrary field $\mathbb{F}$ and $Charpols(\mathcal{S}) = \langle m(x) \rangle$. The polynomial $m(x)$ is called the **minimal polynomial** and its degree the **linear complexity** $\ell c(\mathcal{S})$ of $\mathcal{S}$. We will sometimes use the notation $\min(\mathcal{S})$ for the minimal polynomial of a sequence.*

Theorem 2.58. *An LFSR can produce a maximum sequence if and only if the minimal polynomial is primitive.*

Proof. Recall that a polynomial $p(x) \in \mathbb{F}[x]$ of degree n is defined to be primitive if and only if it divides the polynomial $x^{q^n-1} - 1$ but no other polynomial $x^d - 1$ with $d < q^n - 1$. Now let $\mathcal{S} \not\equiv (0)$ be a non-trivial sequence generated by an LFSR with least period ρ. This is equivalent to that $s_{t+\rho} = s_t$ for all t and hence $x^\rho - 1$ is a characteristic polynomial of $\mathcal{S}$. By Theorem 2.56, $x^\rho - 1$ must be multiple of the minimal polynomial $m(x)$. On the other hand, as ρ is assumed to be the least period, $m(x)$ does not divide a polynomial $x^{\rho'} - 1$ with $\rho' < \rho$.

This shows that $\mathcal{S}$ is maximum if and only if $m(x)$ divides the polynomial $x^{q^n-1} - 1$ but no other polynomial $x^d - 1$ with $d < q^n - 1$. This is exactly the definition of a polynomial being primitive. □

Now, let $\mathcal{S} = (s_t)$ be a periodic sequence, that is $s_{t+\rho} = s_t$ for all $t \geq 0$. Then, $\mathcal{S}$ can be generated by an LFSR of length ρ with the linear function

$$L(x_1, \ldots, x_\rho) := (x_2, \ldots, x_\rho, x_1)$$

and the initial state $(s_0, \ldots, s_{\rho-1})$. Of course, this doesn't exclude shorter LFSRs which might generate the same sequence (see also Example 2.52). The question of the shortest LFSR that can generate $\mathcal{S}$ is answered by the Berlekamp-Massey algorithm:

Theorem 2.59. *Let $\mathcal{S}$ be a linear recurring sequence with linear complexity ℓ. The **Berlekamp-Massey algorithm** [Mas69] can reconstruct in time $O(\ell^2)$ the minimal polynomial of the shortest LFSR that can generate S if at least 2ℓ successive LFSR-sequence elements are given.*

In Section 5.2, we will learn more on LFSRs and their minimal polynomials. In particular, we will show that LFSR-sequences can be expressed and analyzed by the roots of their minimal polynomials. This is connected to the theory of extension fields from finite fields discussed in Section 2.2.5.

2.5 (ι, m)-combiners

LFSRs provide a (hardware-)efficient way to produce sequences of arbitrary length with good statistical properties (see [Gol82]). Though, from a cryptographic point of view, LFSRs represent rather bad keystream generators. The reason is that an attack can be performed by solving a system of linear equations.

More formally, let $K = (k_0, \ldots, k_{n-1})$ be the unknown initial state and L be the *known* feedback matrix. This setting would produce a keystream $k_0, k_1, k_2, \ldots$ where $(k_t, \ldots, k_{t+n-1}) = K \cdot L^t$. Thus, if an adversary knows the values $\boldsymbol{k_{t_1}}, \ldots, \boldsymbol{k_{t_m}}$ of the keystream elements at the m clocks $t_1, \ldots, t_m$, he can use his knowledge to set up the following system of linear equations:

$$\begin{array}{rcl} \boldsymbol{k_{t_1}} & = & K \cdot L^{t_1} \cdot P \\ & \ldots & \\ \boldsymbol{k_{t_m}} & = & K \cdot L^{t_m} \cdot P \end{array} \tag{2.5.11}$$

where $P = (1, 0, \ldots, 0)^t$ denotes the projection on the first entry. If the rank of (2.5.11) is r, then he knows that the value of K belongs to the kernel of dimension $n - r$. As solving a system of linear equations is efficiently possible in practice (if n is not too big what is the case for cryptographic reasonable values of n), this attack is a realistic threat for this kind of keystream generators.

Thus, to strengthen LFSR-based keystream generators, one has to incorporate some kind of non-linearity. So far, designers have mainly pursued the following three strategies:

1. Make the clocking irregular
2. Apply a non-linear function to the outputs from several LFSRs (or several outputs from one LFSR) and output the result
3. Add a second finite state machine with a non-linear update function and combine the contents of both internal states to compute the output

An irregular clocking means that the number of clockings between two outputs is not constant. Famous representants of the irregularly clocked LFSR-based keystream generators are the shrinking generator [CopKM93], the selfshrinking generator [MeiS94] and A5/1 which is used in GSM mobile phone systems in most European countries (see also [BriGW98]). Although irregularly clocked LFSR-based keystream generators have gained some attentions in cryptography, we will not discuss them any further. The reason is that until now, no algebraic attacks are known against them.

The two other approaches can be described by a general kind of keystream generator, which we call (ι, m)-combiners. A formal definition is the following:

Definition 2.60. *Let $\mathbb{F}$ be a finite field. A (ι, m)-**combiner** consists of the following components:*

- *s LFSRs with lengths $n_1, \ldots, n_s$ and feedback matrices $L_1, \ldots, L_s$.*
- *An internal state $S \in \mathbb{F}^m \times \mathbb{F}^n$ where $n = n_1 + \ldots + n_s$.*
- *A matrix L over $\mathbb{F}$ of size $n \times n$, defined by*

$$L := \begin{pmatrix} L_1 & & 0 \\ & \ddots & \\ 0 & & L_s \end{pmatrix}.$$

- *A (projection) matrix P over $\mathbb{F}$ of size $n \times \iota$.*
- *A non-linear **next memory state function**. $\Psi : \mathbb{F}^m \times \mathbb{F}^\iota \to \mathbb{F}^m$.*
- *An **output function** $f : \mathbb{F}^m \times \mathbb{F}^\iota \to \mathbb{F}$.*

*If $m \geq 1$, then we speak of a **combiner with memory**, else of a **simple combiner**.*

The generation of the keystream works as follows. First, the internal state is initialized to a value $S_0 := (Q_0, K) \in \mathbb{F}^m \times \mathbb{F}^n$ where $K \in \mathbb{F}^n$ is the LFSRs' initial states and the content of $Q_0 \in \mathbb{F}^m$ is called the **memory**. As discussed in Section 2.4, the content of the LFSRs is updated by the matrix L. More precisely, $K = (K_0^{(1)}, \ldots, K_0^{(s)})$ where $K_0^{(i)}$ is the i-th LFSR's initial state.

In many cases, only some of the LFSRs' entries are used for the computation of the actual keystream element and the next state of the memory register. This is expressed by the projection matrix P. This means that only the values in $K \cdot L^t \cdot P$ are involved in the computations at clock t. Consequently, we define an abbreviation K_t for $K \cdot L^t \cdot P$ and call it the **input** of the (ι, m)-combiner. The memory is updated via $Q_{t+1} := \Psi(Q_t, K_t)$, whereas the keystream element z_t is computed by $z_t = f(Q_t, K_t)$. Thus, the update function Υ is defined by

$$\Upsilon(S_t) = \Upsilon(Q_t, K \cdot L^t) = (\Psi(Q_t, K \cdot L^t \cdot P), K \cdot L^{t+1}). \tag{2.5.12}$$

In the following we assume that Υ is bijective (see Remark 2.4) what implies that L is regular and $\Psi(Q, \boldsymbol{X}) : \mathbb{F}^m \to \mathbb{F}^m$ is bijective for all $\boldsymbol{X} \in \mathbb{F}^\iota$. A schematic picture of a (ι, m)-combiner is given in Figure 2.5.4.

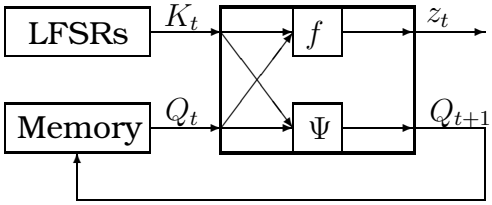

Figure 2.5.4: A (ι, m)-combiner

Notice that a sequence $z_t, \ldots, z_{t+r}$ of outputs depends only on Q_t and $K_t, \ldots, K_{t+r}$. We express this fact by defining the **extended output function** $f_\Psi(Q_t, K_t, \ldots, K_{t+r}) = (z_t, \ldots, z_{t+r})$. For the sake of simplicity, we use the same notation f_Ψ for different values of r.

Observe that the key size of a given (ι, m)-combiner can be easily altered by changing L and P but keeping the same update and output functions Ψ and f. Hence, it is natural to treat the production of the **internal stream** $K_0, K_1, \ldots$ and the computation of the keystream $z_0, z_1, \ldots$ separately. In particular, one could use other mechanisms to produce the internal stream as for example cellular automata (e.g., see [Wol86]).

In the following, we give some examples for (ι, m)-combiners, all defined over $\mathbb{F} = \mathbb{F}_2$. We will later come back to them to illustrate various concepts.

2.5.1 A toy (2,0)-combiner

We start with a very simple keystream generator. It uses two LFSRs of lengths two and three, respectively. Their initial states are denoted by $A_0 = (a_0, a_1)$ and $B_0 = (b_0, b_1, b_2)$. The minimal polynomial of LFSR $\mathcal{A}$ is defined as $m_a(x) := x^2 + x + 1$. That is, the sequence (a_t) produced by LFSR $\mathcal{A}$ fulfills the recursion $a_{t+2} = a_{t+1} + a_t$. The second minimal polynomial is set to $m_b(x) := x^3 + x^2 + 1$, which implies that $b_{t+3} = b_{t+2} + b_t$. In this case, it is $K := (a_0, a_1, b_0, b_1, b_2)$. The feedback matrix L and the projection matrix are defined as

$$L := \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \quad \text{and} \quad P := \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

At each clock t, the keystream element z_t is generated by the function $z_t = f(K_t) = f(a_t, b_t) = a_t \cdot b_t + a_t + b_t$. Hereby, a_t, b_t and z_t are identifiers for the output from LFSR $\mathcal{A}$, LFSR $\mathcal{B}$ and the keystream bit at clock t, respectively.

Let for example be $A_0 := (1, 1)$ and $B_0 := (1, 0, 1)$. Table 2.5.1 displays the first eleven LFSR bits and keystream bits. As the minimal polynomials m_a and m_b are primitive, the sequences (a_t) and (b_t) have the maximum period of $2^2 - 1 = 3$ and $2^3 - 1 = 7$, respectively. Furthermore, one sees that the keystream consists of far more values '1' than '0'. This is because of the fact that the output function is not balanced. Thus, the toy cipher is cryptographically weak. Notice that the toy cipher is a simple combiner, i.e. $m = 0$.

t	1	2	3	4	5	6	7	8	9	10	11
a_t	1	1	0	1	1	0	1	1	0	1	1
b_t	1	0	1	0	0	1	1	1	0	1	0
z_t	1	1	1	1	1	1	1	1	0	1	1

Table 2.5.1: The first eleven clocks of the toy cipher with initial state (1,1,1,0,1)

2.5.2 The Geffe generator

The Geffe generator is an $(3, 0)$-combiner and was introduced in [Gef73]. At each clock, an input of three bits is used to generate one keystream bit. The output function is defined by

$$z = f(a, b, c) = \begin{cases} a & , c = 1 \\ b & , \text{else} \end{cases} \tag{2.5.13}$$

So far, we concerned only simple combiners, that is $m = 0$. Next, we will give two examples for combiners with memory, i.e. $m > 0$.

2.5.3 The summation generator

The summation generator is a $(\iota, \lceil \log_2 \iota \rceil)$-generator and has been introduced by Rueppel in [Rue85]. It is based on integer addition. This means that both the input bits and the memory state are treated as integers and added together. The result forms the output and the next memory state.

More formally, at each clock t there are ι input bits $k_{t,1}, \ldots, k_{t,\iota}$ and the memory state $Q_t \in \mathbb{F}_2^{\lceil \log_2 k \rceil}$. The integer sum of these values is computed, i.e. $A_t := k_{t,1} + \ldots + k_{t,\iota} + Q_t$ where Q_t is taken as a value in $\{0, \ldots, 2^{\lceil \log_2 k \rceil}\}$.

Q_t	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
$\|K_t\|$	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4
A_t	0	1	2	3	4	1	2	3	4	5	2	3	4	5	6	3	4	5	6	7
z_t	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Q_{t+1}	0	0	1	1	2	0	1	1	2	2	1	2	2	2	3	1	2	2	3	3

Table 2.5.2: The various input/output possibilities for the summation generator with $k = 4$ input bits and $m = 2$ memory bits

Then, the output and the next memory state Q_{t+1} are computed by

$$z_t := A_t \bmod 2 = (k_{t,1} + \ldots + k_{t,\iota} + Q_t) \mod 2 \tag{2.5.14}$$

$$Q_{t+1} := A_t \operatorname{div} 2 = \left\lfloor \frac{(k_{t,1} + \ldots + k_{t,\iota} + Q_t)}{2} \right\rfloor \tag{2.5.15}$$

Observe that the values of the output z_t and Q_{t+1} remain the same if one permutes the inputs $K_t = (k_{t,1}, \ldots, k_{t,\iota})$. That is, these computations depend only on the value $|K_t|$ of the inputs. We refer to this kind of (ι, m)-combiners as **permutation invariant combiners**.

Table 2.5.2 displays the various input/output possibilities for the case $k = 4$ and $m = 2$.

2.5.4 The E_0 keystream generator

The keystream generator E_0 is part of the Bluetooth specifications [Blu99] for wireless communications. It consists of 4 regularly clocked LFSRs of lengths 25, 31, 33 and 39, respectively, yielding a key size of 128 bits. Additionally involved are $m = 4$ memory bits. With each clock, an output bit z_t is produced depending on the inputs $K_t = (k_{t,1}, k_{t,2}, k_{t,3}, k_{t,4})$ of the four LFSRs and the four memory bits $Q_t = (q_t, p_t, q_{t-1}, p_{t-1})$. This means that E_0 is a $(4, 4)$-combiner. Then the next memory bits $Q_{t+1} = (q_{t+1}, p_{t+1}, q_t, p_t)$ are calculated and so on. We see that the memory bits q_t and p_t are used in both clocks t and $t+1$. The output bit z_t and the memory bits are computed by the following equations

$$z_t = k_{t,1} + k_{t,2} + k_{t,3} + k_{t,4} + p_t \tag{2.5.16}$$

$$Q_{t+1} = (q_{t+1}, p_{t+1}, q_t, p_t) \tag{2.5.17}$$

$$= (S^1_{t+1} + q_t + p_{t-1}, S^0_{t+1} + p_t + q_{t-1} + p_{t-1}, q_t, p_t), \tag{2.5.18}$$

where

$$S_{t+1} = (S^1_{t+1}, S^0_{t+1}) = \left\lfloor \frac{k_{t,1} + k_{t,2} + k_{t,3} + k_{t,4} + 2 \cdot q_t + p_t}{2} \right\rfloor . \tag{2.5.19}$$

The values for Q_0 and the contents of the LFSRs must be set before the start, the other values will then be calculated.

2.6 Attacks on (ι, m)-combiners

By and by, cryptographers developed numerous attacks on (ι, m)-combiners. In this section, we provide a short overview of existing attacks. For a more detailed survey, we suggest [Zen04] which has served as a guideline for this section.

Statistical tests

Recall that the idea of keystream generators was to imitate the one-time pad where a random bit string was XORed to the message. Though the keystream is not really random, at least it shouldn't show any statistical particularities. Thus, several statistical tests, e.g. see [BekP82, Gol82, Knu81, Mau90], were developed to check if the keystream can be distinguished from a truly random sequence. This includes that the keystream shouldn't have a low period or a low linear complexity.

Time-memory-data tradeoff

As the update function Υ was assumed to be bijective, it suffices to find out the value of one of the internal states S_t, as $S_0 = \Upsilon^{-t}(S_t)$ can be easily reconstructed then. The basic idea behind time-memory-data tradeoff attacks is to perform a brute force attack over a smaller keyspace and to compute for each of the candidates $S_0^{(i)}$ corresponding keystream elements $z_0^{(i)}, \ldots, z_N^{(i)}$. If these occur in the observed keystream, the attacker is able to find out the actual internal state S_t and consequently the initial state S_0. For further reading, we refer to [BirS00].

Guessing attacks

As already discussed in Section 2.1, the size of the key space $\mathcal{K}$ makes a brute force attack usually infeasible. In guessing attacks, the adversary guesses first some parts of the secret key what, together with the observed keystream, imposes some conditions on the remainder. In certain cases, it is possible to discard a huge fraction of the remaining candidates without actually checking them.

This can be achieved in several ways. One example is the linear consistency test (LCT) described in [ZenYR89] for simple combiners which makes use of linear update function L. The guessing yields a system of linear equations in the remain of the secret key. Either this system of equations

possesses no solutions, which proves the guess to be incorrect, or it leaves a (hopefully small) set of remaining candidates which can be tested.

A similar approach is pursued in backtracking attacks. But as opposed to LCT, here parts of secret key are guessed bit by bit, yielding linear equations at a time. Thus, instead of building the whole system of linear equations at once, it is generated successively. If at one step the new equations yield a contradiction, the last guess is modified until the contradiction is resolved. If this is not possible, the attacker tracks back to the last but one guess, changes this value and the according equations and proceeds. For an elaborately treatment of these attacks, we recommend [Zen04].

BDD-based attacks

As the specifications of the (ι, m)-combiner are known to the attacker, each observed keystream element z_t provides the information that $S_t \in f^{-1}(z_t)$ which is a strict subset of $\mathbb{F}^{\iota+m}$ if f is not constant. Thus, if one could keep track of these information in an efficient way, this would make it eventually possible to identify the secret key.

In [Kra02], it was showed that binary decision diagrams (BDDs) provide such a method. A BDD is a data structure which allows to compactly store certain Boolean functions. Additionally, BDDs can be efficiently combined which allows to manage their information. More on BDDs can be found in [Weg87].

The attack from [Kra02] works as follows. At each clock, the information gained are expressed by BDDs of small size. It was showed that for some keystream generators, e.g. E_0, the theoretically estimated time and memory effort to combine these BDDs and thus recovering the secret key was less than for all previous attacks. The experimental results in [Sch02, Ste04] confirmed the general applicability and indicated that the asymptotic behaviour is as predicted. At the same time, it turned out that memory is actually the bottle neck. In [KraS06] BDD-based attacks were combined with guessing attacks described above, yielding a lower memory effort by only slightly increased running time.

Correlation attacks

One of the most important and successful classes of attacks are correlation attacks. Although most concepts should work for (ι, m)-combiners over arbitrary fields, the attacks are usually described for the case $\mathbb{F} = \mathbb{F}_2$, so we regard only this scenario. Correlation attacks on simple combiners are

based on finding and exploiting linear functions $l : \mathbb{F}_2^k \to \mathbb{F}_2$ such that

$$|Pr[l(K_t) = z_t]| \neq \frac{1}{2} \tag{2.6.20}$$

for all t. That is, a linear combination of the inputs is correlated to the keystream bits. In [Sie84, XiaM88], it was proved that the higher the resistance against such linear combinations, the lower the linear complexity of the keystream. Thus, correlation attacks are in principal always possible.

The first correlation attack was described in [Sie85]. Later on, in [MeiS88, MeiS89], faster algorithms taken from coding theory [Gal63] were proposed. The huge number of following publications, e.g. see [MihG90, CheS91, Pen96, ZenH88, ZenYR89, Ziv91a], demonstrates the community's strong interest in this subject. All of these proposals have in common that they require that the number of monomials in the LFSR's minimal polynomial is low. Alternative algorithms which works without this limitation have been given in [JohJ99, JohJ99b, CanT00, CheJS00, Fil00, JohJ02]. Each of these attacks have their advantages and drawbacks. For most cases, the best way is to combine these approaches, e.g. see [MihFI00, MihFI01, ChoJM02]

Although combiners with memory, i.e. (ι, m)-combiners with $m \geq 1$ were primary proposed in [Rue85] to avoid correlation attacks, it eventually turned out in [MeiS92, Gol93, Gol96] that linear combinations similar to (2.6.20) are possible. The difference however is that one has to consider linear combinations between the inputs K_t and keystream bits z_t of *several* consecutive clocks instead of only one as for simple combiners.[6]

Some final remarks

We want to stress that this is only a very brief overview of existing attacks and is far from being complete. Nonetheless, it shows how numerous the approaches are to cryptanalyze (ι, m)-combiners. Given that the first attacks are dating back to the eighties, the appearance of completely new kinds of attacks as BDD-attacks [Kra02] and algebraic attacks [CouM03, ArmK03] almost twenty years later show that cryptanalyzing (ι, m)-combiner is still an important cryptographic task.

Of course, some attacks might be useless for one cipher but very successful for other. Some presume the knowledge of many keystream bits but need only little memory space (e.g., correlation attacks) whereas for

[6] We will encounter the same issue again later in Section 3.2 in the context of finding equations for algebraic attacks in the case of simple combiners and combiners with memory.

others it is just the other way around (e.g., BDD-attacks). Hence, only in the rarest cases it is possible to say that one attack is actually "better" than another. While evaluating an attack, one should always take into account the estimated number of operations, memory and data. Often, it is only possible to give a rough estimate, making the comparison additionally difficult.

Regarding the number of operations, to the best of our knowledge all attacks have in common that either no precise estimates are possible or that this value is exponential in the keysize n. In this sense, algebraic attacks stand out as, in certain cases, it is possible to predict the number of operations being polynomial in n.

3 Algebraic attacks

3.1 Principles

In this section, we give a general description of algebraic attacks on (ι, m)-combiners. As explained in Section 2.1, the security model is that an adversary knows the specification of the combiner and additionally some of the keystream elements z_t. His goal is to use this knowledge to recover the initial settings $\mathcal{S}_0 = (Q_0, K)$ of the memory bits and the LFSRs. Usually, the size of the memory register is small compared to the LFSRs registers. Hence, once K has been found out, Q_0 can easily be reconstructed either by exploiting the structure of the keystream generators or by exhaustively trying all values. Therefore, we concentrate on attacks where the primal goal is to find out the value of K and refer to K as the secret key.

Observe that each part $z_t, \ldots, z_{t+r-1}$ of known keystream reveals some information on the corresponding inputs $K_t, \ldots, K_{t+r-1}$ and hence on K. Normally, the values of K_t are not yet uniquely determined but at least one can derive some conditions on those. This means that each piece of new information, that is each known value $\boldsymbol{z_t}$, allows to reduce the set of possible values for K. If the attacker gathers enough information, then finally this set is reduced to one element which is simply the value of K. Thus, each part of the keystream provides some *partial information* on K. If one has enough partial information at his disposal, he can (at least theoretically) combine them to get ultimately the value of K.

The difficult part is how to manage these partial information such that one can finally determine the value of K by practical means (or at least within a time effort which is below brute force). Thus, one could say that finding an efficient attack is equivalent to finding an efficient way to organize the information. For example, in the BDD-attack presented by Krause in [Kra02], each partial information is stored as special kind of graph, namely a binary decision diagram.

One possibility could be to simply store the set of possible keys and to cross out for each piece of new information the values which can impossibly produce these parts of keystream. But this approach is infeasible as it would require to store a huge set of values at the beginning. Hence, one needs other ways to collect and to manage the restrictions on the values of K.

However, the maybe most intuitive approach would be to express the partial information by equations. This would lead to a system of equations with its solution being exactly the value of K. This is more or less the idea behind the algebraic attacks.

Before we explain them in their generality, we demonstrate them on the toy cipher from Section 2.5.1. Table 3.1.1 displays the first states of the LFSRs and the expressions of the produced keystream bits z_t.

t	A_t	B_t	$z_t = a_t \cdot b_t + a_t + b_t$
0	(a_0, a_1)	(b_0, b_1, b_2)	$z_0 = a_0 \cdot b_0 + a_0 + b_0$
1	$(a_1, a_0 + a_1)$	$(b_1, b_2, b_0 + b_2)$	$z_1 = a_1 \cdot b_1 + a_1 + b_1$
2	$(a_0 + a_1, a_0)$	$(b_2, b_0 + b_2, b_0 + b_1 + b_2)$	$z_2 = (a_0 + a_1) \cdot b_2 + (a_0 + a_1) + b_2$
3	(a_0, a_1)	$(b_0 + b_2, b_0 + b_1 + b_2, b_0 + b_1)$	$z_3 = a_0 \cdot (b_0 + b_2) + a_0 + b_0 + b_2$

Table 3.1.1: The first clocks of the toy cipher

One sees that the information from the last column lead directly to a system of equations (3.1.1) in the known keystream elements $\boldsymbol{z_t}$ and the unknown key elements a_0, a_1, b_0, b_1, b_2.

$$\left.\begin{array}{rcl} \boldsymbol{z_0} & = & a_0 \cdot b_0 + a_0 + b_0 \\ \boldsymbol{z_1} & = & a_1 \cdot b_1 + a_1 + b_1 \\ \boldsymbol{z_2} & = & (a_0 + a_1) \cdot b_2 + (a_0 + a_1) + b_2 \\ \boldsymbol{z_3} & = & a_0 \cdot (b_0 + b_2) + a_0 + b_0 + b_2 \\ & \vdots & \end{array}\right\} \tag{3.1.1}$$

Hence, if enough values $\boldsymbol{z_t}$ are known to an attacker, he can set up a system of equations as described in (3.1.1) and solve it to get the secret key. In principle, this is the idea of an algebraic attack.

Algorithm 3.1 sketches the principles of an algebraic attack on an (ι, m)-combiner.

Algorithm 3.1.
Algebraic attack on an (ι, m)-combiner
Input: *An (ι, m)-combiner, initialized to a secret value $\mathcal{S}_0 = (Q_0, K)$, and the knowledge of some the keystream elements z_t generated by this setting*
Output: *The secret key K*

1: *Set up a system of equations in the unknowns K and the observed values $\boldsymbol{z_t}$*
2: *Recover K by solving the resulting system of equations*
3: ***return*** K

Of course, this leaves several open questions, some of them being:

1. How does one set up a system of equations for an arbitrary (ι, m)-combiner?

2. Are some systems of equations ”better” than others? If yes, how can one find the ”best ones”?
3. Which method is best suited to compute the solution?

In the following sections, we will consider these questions step by step.

3.2 Generating a system of equations

In this section, we discuss how to generate a system of equations that can be used for an algebraic attack. Here, we rather concentrate on the type of equations and postpone the question if such equations exist and how they can be found to Section 4.1.

3.2.1 Simple combiners

In this section, we consider the problem how to get a system of equations for a given simple combiner. As we have seen in the previous section, an obvious approach is to use the equation $z = f(X)$ implicated by the output function f to construct the following system of equations:

$$\left.\begin{array}{rcl} f(K_1) & = & z_1 \\ f(K_2) & = & z_2 \\ f(K_3) & = & z_3 \\ & \dots & \end{array}\right\} \tag{3.2.2}$$

An adversary can insert the known values $\boldsymbol{z_t}$ into (3.2.2) to get a system of equations in the unknowns K_t and the known values $\boldsymbol{z_t}$. Furtheron, he can exploit in the case of LFSR-based keystream generators, that K_t is equivalent to $K \cdot L^t \cdot P$ where $P \in \mathbb{F}^{n\times\iota}$ and $L \in \mathbb{F}^{n\times n}$ are publicly known matrices. Hence, he gets for each known $\boldsymbol{z_t}$ one equation

$$f(K \cdot L^t \cdot P) = \boldsymbol{z_t}. \tag{3.2.3}$$

However, this doesn't need to be the best one can get as the following example shows:

Example 3.1. *Let us look again at the toy cipher from Section 2.5.1. The keystream generation is defined by*

$$f(a_t, b_t) = a_t + b_t + a_t b_t = z_t \tag{3.2.4}$$

where a_t and b_t are the output bits from LFSRs $\mathcal{A}$ and $\mathcal{B}$, respectively. As already discussed in Section 3.1, equation 3.2.4 implies a system of quadratic equations $a_t + b_t + a_t b_t + z_t = 0$ for all $t \geq 0$. Indeed, one can do better. For example, one could multiply each equation with a_t, giving

$$0 = (f(a_t, b_t) + z_t) \cdot a_t = a_t(1 + z_t). \tag{3.2.5}$$

Recall that the degrees $\geq q$ are always reduced to $q - 1$ as explained in Remark 2.44. Hence, equation (3.2.5) implies a system of linear *equations in*

the variables of a_t only. Strictly speaking, the situation is as this. If $\boldsymbol{z_t} = 1$, then (3.2.5) reduces to the trivial equation $0 = 0$ which provides no information. But, whenever $\boldsymbol{z_t} = 1$, equation (3.2.5) tells us that $a_t = 0$. Of course, this linear system on $\mathcal{A}$ will always be underdefined but it helps already to reduce the number of possibilities.

A similar approach is possible regarding $\mathcal{B}$.

As we have seen, the "original" equation (3.2.4) is not necessarily the best choice to set up a system of equations. Therefore, we generalize the description a little. If an adversary knows a valid equation

$$F(X, f(X)) = 0 \tag{3.2.6}$$

where $f(X) = z$ is the output equation for the considered cipher, each known value $\boldsymbol{z_t}$ implies one equation

$$F(K \cdot L^t \cdot P, \boldsymbol{z_t}) = F(K_t, \boldsymbol{z_t}) = 0. \tag{3.2.7}$$

An example is the equation $0 = F(a_t, b_t, z_t) = a_t(1 + z_t)$ from example 3.1.

Finally, one can say that the first step consist mainly in finding an appropriate F such that (3.2.6) holds. This question has been first considered in [CouM03] and further examined in [MeiPC04]. Once such an F is found, the generation of a system of equations is quite straightforward.

Observe that the degree of the functions in equation (3.2.7) are bounded by $\max_{\boldsymbol{z}}\{\deg(F(X, \boldsymbol{z}))\}$ which is independent of the key size. This will play a very important role later for the linearization method (Section 3.3.2) to solve the system of equations.

3.2.2 Combiners with memory

Next, we will put our focus on combiners with memory. Again, we start with the output equation

$$f(Q_t, K_t) = z_t. \tag{3.2.8}$$

Assume now that one uses (3.2.8) to set up the following system of equations[1]:

$$\left.\begin{array}{rcl} f(Q_0, K_0) & = & \boldsymbol{z_0} \\ f(Q_1, K_1) & = & \boldsymbol{z_1} \\ f(Q_2, K_2) & = & \boldsymbol{z_2} \\ & \cdots & \end{array}\right\} \tag{3.2.9}$$

In this case, we encounter the problem that each equation contains the unknown Q_t. As the number of unknowns is hence always bigger than the

[1] For simplicity, we assume for the moment that the adversary knows all keystream elements $\boldsymbol{z_t}$.

number of equations, this specific system of equations remains unsolvable, no matter how many clocks one considers. Indeed, the unknowns Q_t are not independent from each other: each Q_t is uniquely defined by Q_0 and $K_0, \ldots, K_{t-1}$. Hence, one could derive functions $\Psi_t : \mathbb{F}^{m+(t-1)\iota} \to \mathbb{F}$ such that $Q_t = \Psi_t(Q_0, K_0, \ldots, K_{t-1})$ and insert them into (3.2.9):

$$\left.\begin{array}{lcl} f(Q_0, K_0) & = & \boldsymbol{z_0} \\ f(\Psi_1(Q_0, K_0), K_1) & = & \boldsymbol{z_1} \\ f(\Psi_2(Q_0, K_0, K_1), K_2) & = & \boldsymbol{z_2} \\ & \ldots & \end{array}\right\} \tag{3.2.10}$$

This leaves only the unknown Q_0 which can be simply guessed. But this approach poses two other difficulties. Firstly, it might be complicated or at least expensive to compute the functions Ψ_t. Secondly, it might not longer be true that the degrees of the equations in (3.2.10) are bounded by a constant d, independent of the key size. But this is a necessary condition to apply the linearization method (Section 3.3.2), which is the only method so far that allows an accurate estimation of the effort. In some cases, this might be not a problem and other, likewise efficient algorithms to solve the system of equations may exist. But in general, we cannot expect that this approach leads to an effective attack.

However, if it would be possible to to get rid of the unknowns Q_t, the situation would be similar as for simple combiners and, above all, allowing the usage of the linearization method. Actually, this is possible if one considers *several clocks* in one equation instead of only one as it has been done before. This approach has been first presented by us in [Arm02].

Definition 3.2. *Let a (ι, m)-combiner and $r \geq 1$ be fixed. A function $F : \mathbb{F}^{r\cdot\iota+r} \to \mathbb{F}$ is called an* r-***function*** *for the (ι, m)-combiner if*

$$F(X_1, \ldots, X_r, y_1, \ldots, y_r) = 0 \tag{3.2.11}$$

is true where $X_1, \ldots, X_r$ denotes r successive inputs to the output function f and $Y := (y_1, \ldots, y_r)$ the corresponding outputs. That is, each time the attacker knows the values of r successive keystream elements $\boldsymbol{z_t, \ldots, z_{t+r-1}}$, the following equation in K is true:

$$F(K_t, \ldots, K_{t+r-1}, \boldsymbol{z_t, \ldots, z_{t+r-1}}) = 0. \tag{3.2.12}$$

Example 3.3. *In Example 3.1, an 1-function is given by (3.2.5):*

$$F(X, y) = F(x_1, x_2, y) := x_1(1 + y).$$

Another one can be derived from (3.2.4):

$$F(X, y) = F(x_1, x_2, y) := f(x_1, x_2) + y = x_1 + x_2 + x_1x_2 + y.$$

In general, the 1-functions $F(X, y)$ in the case of simple combiners are exactly those functions which comply with (3.2.6).

Once an r-function is known, it can be used to set up a system of equations in the known outputs $\boldsymbol{z_t}$ and the unknown initial state K. Observe that the degree of the equations derived from (3.2.12) are again bounded, here by

$$\max\{\deg(F(X_1, \ldots, X_r, \boldsymbol{y_1}, \ldots, \boldsymbol{y_r}))|(\boldsymbol{y_1}, \ldots, \boldsymbol{y_r}) \in \mathbb{F}^r\}.$$

As we have seen, the existence of 1-functions is obvious for the case of simple combiners. In [ArmK03], we proved the existence of $(m+1)$-functions for general (ι, m)-combiners with memory. We will present later a slightly generalized version of this Theorem.

3.2.3 Z-functions

Summing up, algebraic attacks on both simple combiners and combiners can use systems of equations build from r-functions. Indeed, for several reasons we approve a slightly different approach to generate the system of equations. To motivate this, assume that s different r-functions $F_1, \ldots, F_s$ are known. Then each sequence of r known keystream elements $\boldsymbol{z_t}, \ldots, \boldsymbol{z_{t+r-1}}$ implies the following s equations

$$\left.\begin{array}{rcl} F_1(K_t, \ldots, K_{t+r-1}, \boldsymbol{z_t}, \ldots, \boldsymbol{z_{t+r-1}}) & = & 0 \\ & \vdots & \\ F_s(K_t, \ldots, K_{t+r-1}, \boldsymbol{z_t}, \ldots, \boldsymbol{z_{t+r-1}}) & = & 0. \end{array}\right\} \tag{3.2.13}$$

We will see later in Section 3.3 that a high number of equations can support the solving step enormously. Thus, it would make sense to use each of these equations for generating a system of equations. However, it might be that some of the equations are redundant due to several reasons. In this case, incorporating all s equations would increase the effort to solve the system of equations as one would have to deal with all of them although they provide no additional information.

For example, let F be an r-function and $g : \mathbb{F}^r \to \mathbb{F}$ be an arbitrary function. Let $Y = (y_1, \ldots, y_r)$. Then,

$$F'(X_1, \ldots, X_r, Y) := g(Y) \cdot F(X_1, \ldots, X_r, Y) \tag{3.2.14}$$

is obviously an r-function, too. But F' provides no more information than F already does. The reason is that substituting $y_1, \ldots, y_r$ by concrete values $\boldsymbol{Z} := \boldsymbol{z_t}, \ldots, \boldsymbol{z_{t+r-1}} \in \mathbb{F}^r$ yields

$$F'(K_t, \ldots, K_{t+r-1}, \boldsymbol{Z}) := \underbrace{g(\boldsymbol{Z})}_{\in \mathbb{F}} \cdot F(K_t, \ldots, K_{t+r-1}, \boldsymbol{Z}).$$

Thus, F' collapses to a constant multiple of F and $F'(K_t, \ldots, K_{t+r-1}, \boldsymbol{Z}) = 0$ provides the same information on $K_t, \ldots, K_{t+r-1}$ as $F(K_t, \ldots, K_{t+r-1}, \boldsymbol{Z}) = 0$. In general, it may be difficult or at least time-consuming to decide if (3.2.13) contains redundant equations for certain values $\boldsymbol{z_1}, \ldots, \boldsymbol{z_r}$.

Another problem could be as follows. Assume that two different r-functions F and F' are given. Furthermore, assume that for two different values $\boldsymbol{y_1}, \ldots, \boldsymbol{y_r}$ and $\boldsymbol{y'_1}, \ldots, \boldsymbol{y'_r}$ in $\mathbb{F}^r$ it holds that

$$\begin{array}{rcl} \deg(F(X_1, \ldots, X_r, \boldsymbol{y_1}, \ldots, \boldsymbol{y_r})) & < & \deg(F'(X_1, \ldots, X_r, \boldsymbol{y_1}, \ldots, \boldsymbol{y_r}) \quad \text{but} \\ \deg(F(X_1, \ldots, X_r, \boldsymbol{y'_1}, \ldots, \boldsymbol{y'_r})) & > & \deg(F'(X_1, \ldots, X_r, \boldsymbol{y'_1}, \ldots, \boldsymbol{y'_r}). \end{array}$$

Thereby, we mean the degree in the remaining unknowns $X_1, \ldots, X_r$. If an adversary uses the linearization method (see Section 3.3) to solve the system of equations, he prefers equations with a degree as low as possible. Hence, in this case it would make sense to use F each time the considered known part of keystream sequences equals $\boldsymbol{y_1}, \ldots, \boldsymbol{y_r}$ but not F'. On the other hand, if the keystream sequence is equal to $\boldsymbol{y'_1}, \ldots, \boldsymbol{y'_r}$, it would be better to use F' to set up an equation and not to use F.

Summing up, one has to be careful when using r-functions as substituting $y_1, \ldots, y_r$ by concrete values might cause some unwanted side effects. Therefore, we prefer a slightly different approach. Instead of using (or looking for) equations in unknowns $y_1, \ldots, y_r$ and to insert the observed keystream elements into it, we propose to work with functions which are valid on the inputs of f if the corresponding output is equal to some fixed values. The next definition makes this idea more precise.

Definition 3.4. *Let $K_t \in \mathbb{F}^\iota$ be the inputs of the output function f at clock t and (z_t) denote the produced keystream. A function $F_{\boldsymbol{Z}} : \mathbb{F}^{\iota \cdot r} \to \mathbb{F}$ for $\boldsymbol{Z} \in \mathbb{F}^r$ is called a **$\boldsymbol{Z}$-function** if whenever a part $z_t, \ldots, z_{t+r-1}$ of the keystream is equal to $\boldsymbol{Z}$ then $F_{\boldsymbol{Z}}$ is zero on the corresponding (possibly unknown) inputs $K_t, \ldots, K_{t+r-1}$. In other words, it must hold for any $t \geq 0$ that*

$$(z_t, \ldots, z_{t+r-1}) = \boldsymbol{Z} \quad \Rightarrow \quad F_{\boldsymbol{Z}}(K_t, \ldots, K_{t+r-1}) = 0. \tag{3.2.15}$$

Example 3.5. *Obviously, $F_{\boldsymbol{Z}} :\equiv 0$ is trivially a $\boldsymbol{Z}$-function. In the case of simple combiners, on easily gets $(\boldsymbol{z})$-functions $F_{(\boldsymbol{z})}$ for each $\boldsymbol{z} \in \mathbb{F}$ by $F_{(\boldsymbol{z})}(X) := g(X) \cdot (f(X) - \boldsymbol{z})$.*

More on the existence and properties of $\boldsymbol{Z}$-function will be given in Section 4.1. For the moment, we postpone the question how to find such equations and simply assume that, for a fixed $r \geq 1$, $\boldsymbol{Z}$-functions $F_{\boldsymbol{Z}}$ are known to the attacker for every $\boldsymbol{Z} \in \mathbb{F}^m$. Thus, if he knows the values $\boldsymbol{z_{t_1}}, \ldots, \boldsymbol{z_{t_1+r-1}}, \boldsymbol{z_{t_2}}, \ldots, \boldsymbol{z_{t_2+r-1}}, \ldots, \boldsymbol{z_{t_N}}, \ldots, \boldsymbol{z_{t_N+r-1}}$ of the corresponding

keystream elements, he can use this knowledge to generate the following system of equations:

$$\left.\begin{array}{rcl} F_{(z_{t_1},\ldots,z_{t_1+r-1})}(K_{t_1},\ldots,K_{t_1+r-1}) & = & 0 \\ F_{(z_{t_2},\ldots,z_{t_2+r-1})}(K_{t_2},\ldots,K_{t_2+r-1}) & = & 0 \\ & \ldots & \\ F_{(z_{t_N},\ldots,z_{t_N+r-1})}(K_{t_1},\ldots,K_{t_N+r-1}) & = & 0 \end{array}\right\} \tag{3.2.16}$$

In some cases, we will abbreviate $(z_t, \ldots, z_{t+r-1})$ by Z_t^r. For example, (3.2.16) could be rewritten to

$$\left.\begin{array}{rcl} F_{Z_{t_1}^r}(K_{t_1},\ldots,K_{t_1+r-1}) & = & 0 \\ F_{Z_{t_2}^r}(K_{t_2},\ldots,K_{t_2+r-1}) & = & 0 \\ & \ldots & \\ F_{Z_{t_N}^r}(K_{t_1},\ldots,K_{t_N+r-1}) & = & 0 \end{array}\right\} \tag{3.2.17}$$

By definition, K_t is equivalent to $K \cdot L^t \cdot P$ where $P \in \mathbb{F}^{\iota \times n}$ and $L \in \mathbb{F}^{n \times n}$ are publicly known matrices. Hence, the adversary can rewrite (3.2.16) to

$$\left.\begin{array}{rcl} F_{(z_{t_1},\ldots,z_{t_1+r-1})}(K \cdot L^{t_1} \cdot P,\ldots,K \cdot L^{t_1+r-1} \cdot P) & = & 0 \\ F_{(z_{t_2},\ldots,z_{t_2+r-1})}(K \cdot L^{t_2} \cdot P,\ldots,K \cdot L^{t_2+r-1} \cdot P) & = & 0 \\ & \ldots & \\ F_{(z_{t_N},\ldots,z_{t_N+r-1})}(K \cdot L^{t_N} \cdot P,\ldots,K \cdot L^{t_N+r-1} \cdot P) & = & 0 \end{array}\right\} \tag{3.2.18}$$

This finally leads to a system of equations in the known keystream words z_t and the secret key K. Due to the linearity of P and L, each equation in (3.2.18) has a degree equal to or lower than $\max_{Z}\{\deg(F_Z)\}$.

Actually, one can generalize this a little. Assume that for each value $Z \in \mathbb{F}^r$, there are $i(Z)$ linearly independent Z-functions $F_Z^{(1)}, \ldots, F_Z^{(i(Z))}$ with the same degree. Then, whenever $(z_t, \ldots, z_{t+r-1})$ equals Z, one can include

all $i(\boldsymbol{Z})$ equations into the system of equations. This is displayed in (3.2.19).

$$\left.\begin{array}{rcl} F^{(1)}_{\boldsymbol{Z}^r_{t_1}}(K_{t_1},\ldots,K_{t_1+r-1}) & = & 0 \\ & \cdots & \\ F^{(i(\boldsymbol{Z}^r_{t_1}))}_{\boldsymbol{Z}^r_{t_1}}(K_{t_1},\ldots,K_{t_1+r-1}) & = & 0 \end{array}\right\}$$

$$\left.\begin{array}{rcl} F^{(1)}_{\boldsymbol{Z}^r_{t_2}}(K_{t_2},\ldots,K_{t_2+r-1}) & = & 0 \\ & \cdots & \\ F^{(i(\boldsymbol{Z}^r_{t_2}))}_{\boldsymbol{Z}^r_{t_2}}(K_{t_2},\ldots,K_{t_2+r-1}) & = & 0 \end{array}\right\} \qquad (3.2.19)$$

$$\vdots$$

$$\left.\begin{array}{rcl} F^{(1)}_{\boldsymbol{Z}^r_{t_N}}(K_{t_N},\ldots,K_{t_N+r-1}) & = & 0 \\ & \cdots & \\ F^{(i(\boldsymbol{Z}^r_{t_N}))}_{\boldsymbol{Z}^r_{t_N}}(K_{t_N},\ldots,K_{t_N+r-1}) & = & 0 \end{array}\right\}$$

Observe that an adversary needs to know the values of at least r successive keystream elements to set up one valid equation. If one derives the $\boldsymbol{Z}$-equations directly, the value of r is in general rather moderate as the effort increases with r (see Section 4.2). An example is $r = 4$ in the case of E_0 (see [ArmK03]). However, in fast algebraic attacks (see Chapter 5), one uses linear combinations of a huge number of $\boldsymbol{Z}$-equations to get one new equation with a lowered degree. For these new equations, the value of r can be very high. For example, it was estimated for E_0 that $r = 8.822.188$ for E_0 (see [Cou03, Arm04a]). Thus, the value r, which is the number of successive keystream elements needed to set up one $\boldsymbol{Z}$-function based equation, is a parameter which should be considered when evaluating the complexity of an algebraic attack. Hence, we define a specific notion for it:

Definition 3.6. *Let $F_{\boldsymbol{Z}}$ be $\boldsymbol{Z}$-function for $\boldsymbol{Z} \in \mathbb{F}^r$. Then, r is named the* ***run length*** *of $F_{\boldsymbol{Z}}$.*

Remark 3.7. *The name* run length *is inspired by the fact that a run of r* known *successive keystream elements is required to use $F_{\boldsymbol{Z}}$ for building one equation in the key K.*

Example 3.8. *To illustrate the concept of $\boldsymbol{Z}$-functions we consider again the toy keystream generator from Section 2.5 with the output function $f(a,b) = a \cdot b + a + b$. One sees easily that $f(a,b) = 0$ if and only if $(a,b) = (0,0)$. Thus, whenever a keystream element z_t is equal to zero, we know immediately that $(a_t, b_t) = 0$, yielding the two (0)-functions $F^{(1)}_{(0)}(a,b) := a$ and and $F^{(2)}_0(a,b) = b$. For the output $z_t = 1$, we define the trivial (1)-function $F_{(1)}(a,b) := f(a,b) - 1 = a \cdot b + a + b - 1$.*

Let $K = (11101)$ be the secret key. The first eleven inputs and keystream elements are

t	0	1	2	3	4	5	6	7	8	9	10
a_t	**1**	**1**	0	1	1	0	1	1	0	1	1
b_t	**1**	**0**	**1**	0	0	1	1	1	0	1	0
z_t	1	1	1	1	1	1	1	1	0	1	1

Assume now that an adversary gets knowledge of the first eleven outputs, i.e. $(z_0, \ldots, z_{10}) = (1,1,1,1,1,1,1,1,0,1,1)$. Using the Z-functions defined above would lead to the following system of equations:

$$\begin{aligned}
0 &= F_1(a_0, b_0) = a_0b_0 + a_0 + b_0 - 1 \\
&\cdots \\
0 &= F_1(a_7, b_7) = a_7b_7 + a_7 + b_7 - 1 \\
0 &= F_0^{(1)}(a_8, b_8) = a_8 \\
0 &= F_0^{(2)}(a_8, b_8) = b_8 \\
0 &= F_1(a_9, b_9) = a_9b_9 + a_9 + b_9 - 1 \\
0 &= F_1(a_{10}, b_{10}) = a_{10}b_{10} + a_{10} + b_{10} - 1
\end{aligned} \tag{3.2.20}$$

Observe that this system of equations is independent *of the method how the internal bit streams (a_t) and (b_t) are generated. Generally, it is assumed that an adversary also knows the definition of the keystream generator. In this case, he would know that $a_t = a_{t-1} + a_{t-2}$, $t \geq 2$, and $b_t = b_{t-1} + b_{t-3}$, $t \geq 3$. Incorporating these information into (3.2.20) gives a system of equations in $K = (a_0, a_1, b_0, b_1, b_2)$ where each equation is either linear or quadratic. This is displayed in Table 3.2.2. The equations in the last column form the following system of equations:*

$$\begin{aligned}
1 &= a_0 + b_0 + a_0b_0 \\
1 &= a_1 + b_1 + a_1b_1 \\
1 &= a_0 + a_1 + b_2 + a_0b_2 + a_1b_2 \\
1 &= a_0 + b_0 + b_2 + a_0b_0 + a_0b_2 \\
1 &= a_1 + b_0 + b_1 + b_2 + a_1b_0 + a_1b_1 + a_1b_2 \\
1 &= a_0 + a_1 + b_0 + b_1 + a_0b_0 + a_0b_1 + a_1b_0 + a_1b_1 \\
1 &= a_0 + b_1 + b_2 + a_0b_1 + a_0b_2 \\
1 &= a_1 + b_0 + a_1b_0 \\
0 &= a_0 + a_1 \\
0 &= b_1 \\
1 &= a_0b_2 + a_0 + b_2 \\
1 &= a_1 + b_0 + b_2 + a_1b_0 + a_1b_2
\end{aligned} \tag{3.2.21}$$

Remark 3.9. *Although we prefer to use Z-functions to generate system of equations, in some cases r-functions are the better tool. One example are*

t	a_t	b_t	z_t	Equations
0	a_0	b_0	1	$0 = F_1(a_0, b_0) = a_0b_0 + a_0 + b_0 + 1$
1	a_1	b_1	1	$0 = F_1(a_1, b_1) = a_1b_1 + a_1 + b_1 + 1$
2	$a_0 + a_1$	b_2	1	$0 = F_1(a_0 + a_1, b_1)$ $= (a_0 + a_1)b_2 + a_0 + a_1 + b_2 + 1$
3	a_0	$b_0 + b_2$	1	$0 = F_1(a_0, b_0 + b_2)$ $= a_0(b_0 + b_2) + a_0 + b_0 + b_2 + 1$
4	a_1	$b_0 + b_1 + b_2$	1	$0 = F_1(a_1, b_0 + b_1 + b_2)$ $= a_1(b_0 + b_1 + b_2) + a_1 + b_0 + b_1 + b_2 + 1$
5	$a_0 + a_1$	$b_0 + b_1$	1	$0 = F_1(a_0 + a_1, b_0 + b_1)$ $= (a_0 + a_1)(b_0 + b_1) + a_0 + a_1 + b_0 + b_1 + 1$
6	a_0	$b_1 + b_2$	1	$0 = F_1(a_0, b_1 + b_2)$ $= a_0(b_1 + b_2) + a_0 + b_1 + b_2 + 1$
7	a_1	b_0	1	$0 = F_1(a_1, b_0) = a_1b_0 + a_1 + b_0 + 1$
8	$a_0 + a_1$	b_1	0	$0 = F_0^{(1)}(a_0 + a_1, b_1) = a_0 + a_1$ $0 = F_0^{(2)}(a_0 + a_1, b_1) = b_1$
9	a_0	b_2	1	$0 = F_1(a_0, b_2) = a_0b_2 + a_0 + b_2 + 1$
10	a_1	$b_0 + b_2$	1	$0 = F_1(a_1, b_0 + b_2)$ $= a_1(b_0 + b_2) + a_1 + b_0 + b_2 + 1$

Table 3.2.2: A system of equations using z-functions and the knowledge of the minimal polynomials for the toy cipher.

fast algebraic attacks (see Chapter 5) where it is necessary that the system of equations is build of r-functions. But this poses no problem, as one can easily construct an r-function from Z-functions. Let

$$\delta_{\boldsymbol{Z}}(\boldsymbol{y_1}, \ldots, \boldsymbol{y_r}) := \begin{cases} 1 & , (\boldsymbol{y_1}, \ldots, \boldsymbol{y_r}) = \boldsymbol{Z} \\ 0 & , \text{else} \end{cases} .$$

Then, one can construct an r-function F from the Z-functions by

$$F(X_1, \ldots, x_r, y_1, \ldots, y_r) := \sum_{\boldsymbol{Z} \in \mathbb{F}^r} \delta_{\boldsymbol{Z}}(y_1, \ldots, y_r) \cdot F_{\boldsymbol{Z}}(X_1, \ldots, X_r).$$

In the next section, we will treat the question of solving this type of systems of equations. It will turn out that for the methods considered, the degree of the Z-functions plays an important role in the efficiency of the solving step. In other words, the lower the degrees, the faster it is possible to compute the solution and thus to recover the secret key. Consequently, an attacker prefers low degree Z-functions. In Chapter 4, we will face this

problem again. If the degrees of the $\boldsymbol{Z}$-functions are not all equal, he might omit those with the higher degrees and concentrate only on those with the lowest degree. For instance, in Example 3.8, one could use only the linear (0)-functions to get a system of linear equations. Finally, we sum up algebraic attacks with $\boldsymbol{Z}$-functions in Algorithm 3.2.

Algorithm 3.2.
An algebraic attack on a (ι, m)-combiner using a system of equations composed of $\boldsymbol{Z}$-functions

Input: *A (ι, m)-combiner, initialized to a secret value $\mathcal{S}_0 = (Q_0, K)$ and the knowledge of some the keystream elements generated by this setting*

Output: *The secret key K*

1: *Fix $r \geq 1$ and find $i(\boldsymbol{Z})$ $\boldsymbol{Z}$-functions for each $\boldsymbol{Z} \in \mathbb{F}^r$ (see Section 4.1 for more details)*
2: *Initialize an empty system of equations*
3: *For each run of known keystream elements $(\boldsymbol{z_t}, \ldots, \boldsymbol{z_{z+r-1}}) = \boldsymbol{Z}_t^r$, add the following equations to the system of equations:*

$$\begin{aligned} F^{(1)}_{\boldsymbol{Z}_t^r}(K_t, \ldots, K_{t+r-1}) &= 0 \\ &\ldots \\ F^{(i(\boldsymbol{Z}_t^r))}_{\boldsymbol{Z}_t^r}(K_t, \ldots, K_{t+r-1}) &= 0 \end{aligned}$$

4: *Recover K by solving the resulting system of equations*
5: ***return*** K

3.3 Computing the solution

In the previous sections, we discussed different possibilities how to generate a system of equations whose solution reveals the LFSRs' initial state. Of course, this is only half the battle. Not less important is the question how to solve these systems efficiently, which is the topic of this section. We assume from now on that a system of equations has been constructed, depending on the known parts of the keystream, and the goal is to find a common root. We hereby implicitly assume that the system of equations has only one solution which is K.

In the following, we will describe various methods to solve a system of equations over a finite field $\mathbb{F}$. Let the system of equations be given by:

$$\begin{cases} f_1(x_1, \ldots, x_n) &= 0 \\ f_2(x_1, \ldots, x_n) &= 0 \\ & \ldots \\ f_N(x_1, \ldots, x_n) &= 0 \end{cases} \tag{3.3.22}$$

Actually, the idea to express a cipher by a system of equations is not new. Already Shannon mentioned in his seminal paper [Sha49] that breaking a good cipher should require *"as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type."* The reason for this recommendation is that solving systems of nonlinear equations is difficult in general. For example, it has been proven that finding a solution of a system of quadratic (!) equations is an NP-hard problem [HasPS93]. This means that probably[2] no polynomial time algorithms exist for solving general systems of non-linear equations over finite fields. Nonetheless, Gröbner bases algorithms have established themselves as a indispensable tool to solve systems of polynomial equations and other problems which can be reduced to this kind of system. As the computation of Gröbner bases is useful also for other problems related to algebraic attacks, we give now a short introduction to this topic. For a more detailed treatment of this subject, we suggest the book by Cox, Little and O'Shea [CoxLS96].

3.3.1 Gröbner bases

The theory of Gröbner bases has been initiated by Bruno Buchberger in his PhD thesis [Buc65], named after his supervisor Wolfgang Gröbner, as a tool to solve the ideal membership problem (see below). Simultaneously, he described the first algorithm to compute Gröbner bases and proved its

[2] Otherwise, it would hold that $P = NP$ what is doubted by most complexity theoreticians.

correctness and termination. The most efficient methods over a finite field $\mathbb{F}$ publicly known so far are F_4 and F_5, both due to Faugère [Fau99, Fau02].

Gröbner bases are special bases of ideals I in polynomial rings. For obvious reasons, we will concentrate only on $\mathbb{F}_q[x_1, \ldots, x_n]$, that is polynomials over finite fields. For certain problems, Gröbner bases are helpful as the answers can be directly read out the basis or can be solved directly with the basis. One such problem is the **ideal membership problem**: Given an ideal $I \subseteq \mathbb{F}_q[x_1, \ldots, x_n]$ and $f \in \mathbb{F}_q[x_1, \ldots, x_n]$, is $f \in I$? Another problem is that of solving a system of equations. If

$$I := \langle f_1(x_1, \ldots, x_n), \ldots, f_N(x_1, \ldots, x_n), x_1^q - x_1, \ldots, x_n^q - x_n \rangle ,$$

that is I is the ideal generated by the equations for which we want to find a solution plus the field equations, then certain Gröbner bases allow to read out the solution of (3.3.22) immediately.

An important fact is that Gröbner bases per se are not unique. Prior to the computation of a Gröbner basis, a term ordering has to be fixed.

Definition 3.10. *Let $\mathbb{F}[X] := \mathbb{F}[x_1, \ldots, x_n]$ be the ring of multivariate polynomials in n unknowns over some field $\mathbb{F}$. A **term ordering** is a relation $\prec$ on the set of monomials X^E, $E \in \mathbb{N}^n$, such that*

1. *$\prec$ is a total ordering. This means that for any monomials X^A and X^B it holds that either $X^A = X^B$, $X^A \prec X^B$ or $X^B \prec X^A$.*
2. *If $X^A \prec X^B$ then $X^A \cdot X^C \prec X^B \cdot X^C$ for each monomial X^C.*
3. *$\prec$ is a well-ordering. This means that every nonempty set of monomials has a smallest element under $\prec$.*

*We call a term ordering **graded** if $|A| < |B|$ implies that $X^A \prec X^B$ where $|A|$ denotes the weight of A (see definition on page 26).*

In most cases the term ordering is of high significance for the speed of the algorithm. Unfortunately, there is no guideline on how to choose the term ordering. In most cases it is unknown a priori which term ordering would result in the fastest computation.

Once a term ordering is specified, the terms of a function f can be ordered with respect to this ordering. In particular, there exists one term which is the greatest in respect to the chosen ordering:

Definition 3.11. *Let $f = \sum_E c_E \cdot X^E \not\equiv 0$ and $\prec$ be a term ordering on the monomials X^E. The **head term** $\mathrm{HT}(f)$ of f is defined to be the largest term of f with respect to the fixed ordering $\prec$. This means that $\mathrm{HT}(f) := c_{E'} \cdot X^{E'}$ with $X^{E'} = \max_\prec\{X^E | c_E \neq 0\}$.*

For a set $\mathcal{F}$ of polynomials, we define $HT(\mathcal{F}) := \{HT(f) \mid f \in \mathcal{F}\}$.

Remark 3.12. *The notion of the head term is well defined as a term ordering is a total ordering by definition. This implies the existence of one unique largest element.*

One possibility to define a term ordering is presented in the following proposition:

Proposition 3.13. *Let $\mathbb{F}[x_1, \ldots, x_n]$ be the ring of multivariate polynomials in n unknowns over some field $\mathbb{F}$. Furthermore, let R be the ring such that each $f \in \mathbb{F}[x_1, \ldots, x_n]$ can be expressed by*

$$f = f(x_1, \ldots, x_n) = \sum_{E \in \mathbb{N}^n} c_E X^E.$$

Then, any ordering $\prec_R$ on R^n induces a term ordering $\prec_{K[X]}$ on $k[X]$ by

$$X^E \prec_{k[X]} X^{E'} \quad \Longleftrightarrow \quad E \prec_R E'.$$

For example, one can use the graded reverse lexicographic ordering grevlex (see following definition). We will call the induced term ordering also grevlex.

Definition 3.14. *The **graded reverse lexicographic order (grevlex)** on $\mathbb{N}_q^n$ is defined as follows. For two elements $A := (a_1, \ldots, a_n), B := (b_1, \ldots, b_n) \in \mathbb{N}_q^n$, it is that $A \prec B$ if either $|A| < |B|$ or if $|A| = |B|$ and $a_i < b_i$ for the largest $i \in \{1, \ldots, n\}$ with $x_i \neq y_i$.*[3]

Example 3.15. *For the grevlex-ordering of the elements in $\mathbb{N}^3$, it holds for example*

$$(000) \prec (100) \prec (010) \prec (001) \prec (110) \prec (101) \prec (011) \prec (111).$$

This implies the following ordering of monomials in $\mathbb{F}_2[x_1, x_2, x_3]$:

$$1 \prec x_1 \prec x_2 \prec x_3 \prec x_1x_2 \prec x_1x_3 \prec x_2x_3 \prec x_1x_2x_3.$$

For example, this yields that $\mathrm{HT}(1 + x_1 + x_3 + x_1x_2x_3) = x_1x_2x_3$ and $\mathrm{HT}(x_2 + x_3 + x_1x_2 + x_2x_3 + x_1x_3) = x_2x_3$.

Now, we are ready to give the definition of a Gröbner basis:

Definition 3.16. *Let I be an ideal in the ring $\mathbb{F}[x_1, \ldots, x_n]$ and a term ordering $\prec$ be fixed. A set $G = \{g_1, \ldots, g_s\} \subseteq I$ is called a **Gröbner basis** of I with respect to "$\prec$" if for every $f \in I$ there exist $g_i \in G$ such that $\mathrm{HT}(g_i)$ divides $\mathrm{HT}(f)$.*

[3] Recall the definition of the weight $|A|$ of A from page 26.

Gröbner bases G are special bases of an ideal I (see Definition 2.16). Apart from $I = \langle G \rangle$, G has additional properties which make them useful for solving certain problems.

As mentioned above one of these problems is the ideal membership problem, being the question whether a function f is an element of an ideal I. Given a Gröbner basis of I it can be solved as follows. If $f = 0$, then $f \in I$ is certainly true. If none of the terms $\mathrm{HT}(g_i)$ divides $\mathrm{HT}(f)$, then $f \notin I$ by the definition of a Gröbner basis. Thus, the remaining case is that $\mathrm{HT}(g_i)$ divides $\mathrm{HT}(f)$ for some i but $f \neq 0$. Let T be the term such that $T \cdot \mathrm{HT}(g_i) = \mathrm{HT}(f)$ and set $f' := f - T \cdot g_i$. Now, it obviously holds that $\mathrm{HT}(f) \prec \mathrm{HT}(f')$. As for each term only finitely many smaller terms exists, a finite repetition of this step eventually leads to the case that either $f = 0$, which shows that $f \in I$, or to the case that the head term of f is not divisible by any head term $\mathrm{HT}(g_i)$, thus $f \notin I$.

Gröbner bases are often bigger than necessary. By eliminating unneeded generators, it is possible to derive a minimal Gröbner basis. Unfortunately, a given ideal may have many minimal Gröbner bases. Fortunately, it is possible to specify one minimal basis which is unique, the so-called **reduced Gröbner basis**:

Definition 3.17. *A reduced Gröbner basis G for a polynomial ideal I is a Gröbner basis for I such that:*

1. $\mathrm{HT}(g) = X^\alpha$ *for all* $g \in G$, *that is the head term has the coefficient 1*

2. *For all* $g \in G$, *no monomial of* g *lies in* $\langle \mathrm{HT}(G \setminus \{g\}) \rangle$

Remark 3.18. *Let a graded term ordering be fixed and $G = \{g_1, \ldots, g_s\}$ be the reduced Gröbner basis of an ideal I which contains the field equations. Despite its uniqueness (see [CoxLS96, Proposition 6 in 2.7] for a proof), it has another useful property. By definition, for any $f \in I$ it holds that at least one $\mathrm{HT}(g_i)$ divides $\mathrm{HT}(f)$. In particular, if $\deg(f) = d$, then at least one g_i has a degree $\leq d$. Thus, reduced Gröbner bases can be used to check whether I contains functions with a degree lower than a given bound. This property will become useful on several occasions concerning algebraic attacks.*

Before we proceed on Gröbner bases, we want to point out that there exist some parallels between reduced Gröbner bases and the reduced row echelon form in linear algebra:

Example 3.19. *Let*

$$\begin{aligned} f_1' &:= 3x_1 + 5x_2 + 5x_4 + 3x_5, \\ f_2' &:= x_1 + 4x_2 + 6x_3 + 5x_4 + 1x_5, \\ f_3' &:= 5x_1 + 6x_2 + 5x_3 + x_4 + 2x_5, \\ f_4' &:= 6x_1 + 3x_2 + 2x_3 + x_4 + 4x_5 \\ f_5' &:= x_1 + 4x_2 + 2x_3 + 2x_4 + 6x_5. \end{aligned}$$

be some linear functions in $\mathbb{F}_7[x_1, \ldots, x_5]$ *and* $I := \langle f_1', \ldots, f_5' \rangle$*. Choosing the grevlex-ordering, it holds that* $x_1 \prec \ldots \prec x_5$*.*

Next, we compute the row echelon form for the matrix given by the linear functions f_i'*. Thereby, each column represents one monomial* x_i*. More concretely, the first column represents* x_1*, the second* x_2*, and so on.*

$$\begin{pmatrix} 3 & 5 & 0 & 5 & 3 \\ 1 & 4 & 6 & 5 & 1 \\ 5 & 6 & 5 & 1 & 2 \\ 6 & 3 & 2 & 1 & 4 \\ 1 & 4 & 2 & 2 & 6 \end{pmatrix} \sim \begin{pmatrix} 3 & 5 & 0 & 5 & 3 \\ 0 & 0 & 6 & 1 & 0 \\ 0 & 0 & 5 & 2 & 4 \\ 0 & 0 & 2 & 5 & 5 \\ 0 & 0 & 2 & 5 & 5 \end{pmatrix} \sim \begin{pmatrix} 3 & 5 & 0 & 5 & 3 \\ 0 & 0 & 6 & 1 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix} \sim \begin{pmatrix} 1 & 4 & 0 & 4 & 0 \\ 0 & 0 & 1 & 6 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.3.23}$$

The last matrix is in the reduced row echelon form and describes new linear functions

$$\begin{aligned} f_1 &:= x_1 + 4x_2 + 4x_4, \\ f_2 &:= x_3 + 6x_4, \\ f_3 &:= x_5. \end{aligned}$$

Obviously, $G := \{f_1, f_2, f_3\}$ *is a basis of* I *with* $\mathrm{HT}(f_1) = x_1$*,* $\mathrm{HT}(f_2) = x_3$*, and* $\mathrm{HT}(f_3) = x_5$*. The coefficients of the head terms are all equal to 1. Furthermore, none of the monomials of the* f_i *is a multiples of a head terms from another function. Thus, the basis given by the reduced echelon form is the reduced Gröbner basis of* I*. In other words, the reduced Gröbner basis can be computed with usual Gaussian elimination if only linear functions are considered.*

The previous example shows that some similarities exist between the theory and methods of linear algebra and of Gröbner bases. Of course, this is no 1-to-1 correspondence, as the theory of Gröbner bases has to deal with products of polynomials, whereas in linear algebra only linear combinations matter. However, this comparison may help to shed some light on the ideas behind Gröbner bases.

The following theorem shows the connection between reduced Gröbner bases and the solution of a system of equations. For a proof we refer to [Ars05, Proposition 1.14].

Theorem 3.20. *Consider a system of equations as displayed in (3.3.22) over a finite field $\mathbb{F}_q$. Define the ideal*

$$I := \langle f_1, \ldots, f_n, x_1^q - x_1, \ldots, x_n^q - x_n \rangle . \tag{3.3.24}$$

Then, for any term ordering, the reduced Gröbner basis of I is equal to

1. $\{x_1 - \boldsymbol{x_1}, \ldots, x_n - \boldsymbol{x_n}\}$ *if* $(\boldsymbol{x_1}, \ldots, \boldsymbol{x_n})$ *is the unique solution of (3.3.22).*
2. $\{1\}$ *if (3.3.22) has no solution.*

In particular, one can easily read the solution from the reduced Gröbner basis.

Example 3.21. *To illustrate this approach, we consider again the toy cipher from Section 2.5. In Example 3.8, the following system of equations is valid in the case that the first eleven keystream bits are equal to* 11111111011*:*

$$\begin{aligned}
f_1 &= a_0 + b_0 + a_0b_0 + 1 \\
f_2 &= a_1 + b_1 + a_1b_1 + 1 \\
f_3 &= a_0 + a_1 + b_2 + a_0b_2 + a_1b_2 + 1 \\
f_4 &= a_0 + b_0 + b_2 + a_0b_0 + a_0b_2 + 1 \\
f_5 &= a_1 + b_0 + b_1 + b_2 + a_1b_0 + a_1b_1 + a_1b_2 + 1 \\
f_6 &= a_0 + a_1 + b_0 + b_1 + a_0b_0 + a_0b_1 + a_1b_0 + a_1b_1 + 1 \\
f_7 &= a_0 + b_1 + b_2 + a_0b_1 + a_0b_2 + 1 \\
f_8 &= a_1 + b_0 + a_1b_0 + 1 \\
f_9 &= a_0 + a_1 \\
f_{10} &= b_1 \\
f_{11} &= a_0b_2 + a_0 + b_2 + 1 \\
f_{12} &= a_1 + b_0 + b_2 + a_1b_0 + a_1b_2 + 1
\end{aligned} \tag{3.3.25}$$

Let $I := \langle f_1, \ldots, f_{12}, a_0^2 - a_0, \ldots, b_2^2 - b_2 \rangle \subsetneq \mathbb{F}_2[a_0, a_1, b_0, b_1, b_2]$ and consider the grevlex-ordering with $a_0 \prec a_1 \prec b_0 \prec b_1 \prec b_2$. Then, the reduced Gröbner basis of I is $G := \{a_0 + 1, a_1 + 1, b_0 + 1, b_1, b_2 + 1\}$ which allows to easily read out the secret key $K = (\boldsymbol{a_0}, \boldsymbol{a_1}, \boldsymbol{b_0}, \boldsymbol{b_1}, \boldsymbol{b_2}) = (1, 1, 1, 0, 1)$.

Several examples exist where the application of Gröbner bases algorithms has been quite successful for cryptanalysis. As an example, we refer to the attack on HFE [FauJ03]. Of course, due to the hardness of the underlying problem, Gröbner bases algorithms can have in the worst case a run time exponential in the number n of unknowns. Even worse, no methods are known so far to estimate the complexity a priori.

However, if the number of equations N is bigger than the number of unknowns, the complexity can drop significantly. For example, in [BarFS03] it was showed that the run time of the Gröbner basis algorithm F_5 over $\mathbb{F}_2$ is

sub-exponential $\mathcal{O}(e^{\frac{n \log\log n}{\log n}})$ if N, the number of equations, is in $\mathcal{O}(n \cdot \log^2(n))$. In [ArsF03], the case of simple combiners over $\mathbb{F}_2$ and the usage of F_5 were explicitly examined. The authors showed that if $N \geq \mu_2(d) = \sum_{i=0}^{d} \binom{n}{i}$ linearly independent equations are given with $d \leq \left\lfloor \frac{\iota+1}{2} \right\rfloor$, then the reduced Gröbner basis can be computed within $\mathcal{O}(\mu_q(d)^\omega)$ where ω is the exponent for solving systems of linear equations. Observe that $\mu_2(d) \in \mathcal{O}(n^d)$ and thus the overall complexity is polynomial in n for fixed d.[4]

3.3.2 Linearization

As mentioned before, the problem of finding a solution to a random system of quadratic equations is NP-complete. On the other hand, the case of linear systems of equations is rather easy as they can be solved efficiently by Gaussian elimination. Here, efficiently means that time and memory effort are polynomial in the number of unknowns. The reason is that if the number of linearly independent equations equals the number of *unknowns*, one can eliminate them step by step in the equations by computing the row echelon form to get the value of each monomial.

For the case of non-linear equations, one also has to deal with products of unknowns. The key idea of **linearization** is to re-write the system of non-linear equations in n unknowns as a new system of linear equations with a significantly increased number of unknowns, which can be easily solved by Gaussian elimination. More precisely, if the number of linearly independent equations is equal to the number of *monomials*, then one can apply the same algorithm to eliminate the monomials step by step as in the case of linear systems of equations.

Example 3.22. *Consider the following system of non-linear equations over* $\mathbb{F}_2$*:*

$$\begin{aligned} x + y &= 1 \\ y + xy &= 0 \\ xy &= 0 \end{aligned}$$

As xy *occurs only in the third equation, this equation can be used to eliminate* xy *in the second one. This leaves only* y*, so that it can be eliminated in the first line. What remains is the following system of equations:*

$$\begin{aligned} x &= 1 \\ y &= 0 \\ xy &= 0 \end{aligned}$$

[4] Although nowhere stated, we assume that F_5, which is based on processing iteratively generated matrices, just becomes the linearization method (see next section) in the particular case mentioned above, as they have both the same time and space efforts.

From this, we can easily read the solution $x = 1$ and $y = 0$. Thus, by successively eliminating the monomials in these equations, the solution could be easily computed.

Assume that the multivariate polynomials in 3.3.22 are linearly independent and let $\mathcal{E} \subseteq \{0, \ldots, q-1\}^n$ be the set of occurring exponents, that is $f_i = \sum_{E \in \mathcal{E}} c_E^{(i)} X^E$ for $1 \leq i \leq N$. In other words, each occurring monomial has an exponent in $\mathcal{E}$. Hence, there are exactly $|\mathcal{E}|$ different monomials in this system of equations. By introducing a new identifier for each monomial of degree > 1, one gets a new system of linear equations, but with the number of unknowns increased from n to $|\mathcal{E}|$. Thus, we *linearized* the system of non-linear equation. If $N = |\mathcal{E}| - 1$, then one can triangulate the system of equations and solve it analogously to Gaussian elimination. The time and memory effort are in $\mathcal{O}(|\mathcal{E}|^3)$ and $\mathcal{O}(|\mathcal{E}|^2)$, respectively. The complexity can be further reduced by using improved algorithms as for example the one from Strassen [Str69].

Remark 3.23. *Some might be wondering about the condition $N = |\mathcal{E}| - 1$ as many papers state $N = |\mathcal{E}|$ to be the requirement. Actually, both descriptions are correct, depending on how one treats the constant monomial 1. Consider a system of N linearly independent equations, given by*

$$f_i(X) = \sum_{E \in \mathcal{E}} (f_i)_E \cdot X^E = c_i \tag{3.3.26}$$

with $i = 1, \ldots, N$, $(f_i)_E, c_i \in \mathbb{F}_q$, and $\mathcal{E} \subseteq \{0, \ldots, q-1\}^n$. We assume that it has only one solution which can be found by linearization.

Now, we distinguish between two different cases. In the first case, 1 formally does not belong to the set of involved monomials $\{X^E \mid E \in \mathcal{E}\}$. This means that $(0, \ldots, 0) \notin \mathcal{E}$. Hence, if $N = |\mathcal{E}|$, one can transform (3.3.26) by linear transformations to $X^E = c_E, \quad c_E \in \mathbb{F}, E \in \mathcal{E}$.

However, if one treats the constants as multiples of the monomial 1[5], then 1 would belong to the set of involved monomials. This implies that $(0, \ldots, 0) \in \mathcal{E}$ and that (3.3.26)is homogeneous, i.e. $c_i = 0$. The matrix derived from the functions $f_i(X)$ has the size $N \times |\mathcal{E}|$ and has a kernel of dimension one. In particular, it holds $N = |\mathcal{E}| - 1$.

In a nutshell, if 1 is not counted as one of the monomials, then the condition is $N = |\mathcal{E}|$, else $N = |\mathcal{E}| - 1$. Both perspectives points are equal as the underlying systems of equations are the same.

Before we analyze the effort of this approach for the case of combiners with memory, we illustrate the linearization method on the following example:

[5] This is our viewpoint.

Example 3.24. *We consider again the system of equation (3.3.25). As each equation is quadratic or linear, not more than* $\binom{5}{2} + \binom{5}{1} + 1 = 16$ *different monomials can be present (including the constant monomial 1). Indeed, this is only a rough upper bound. Due to the structure of the* (z)*-functions, the list of actually involved monomials is smaller and consists of only the following eleven monomials:*

$$a_0, a_1,\ b_0, b_1, b_2,\ a_0b_0, a_1b_0,\ a_0b_1, a_1b_1,\ a_0b_2, a_1b_2$$

Next, we rewrite (3.3.25) as a matrix-vector-product:

$$\underbrace{\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}}_{=:M} \cdot \underbrace{\begin{pmatrix} 1 \\ a_0 \\ a_1 \\ b_0 \\ b_1 \\ b_2 \\ a_0 \cdot b_0 \\ a_1 \cdot b_0 \\ a_0 \cdot b_1 \\ a_1 \cdot b_1 \\ a_0 \cdot b_2 \\ a_1 \cdot b_2 \end{pmatrix}}_{=:V} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.3.27}$$

Thus, the vector

$$V_K := (1, \boldsymbol{a_0}, \boldsymbol{a_1},\ \boldsymbol{b_0}, \boldsymbol{b_1}, \boldsymbol{b_2},\ \boldsymbol{a_0 \cdot b_0}, \boldsymbol{a_1 \cdot b_0},\ \boldsymbol{a_0 \cdot b_1}, \boldsymbol{a_1 \cdot b_1},\ \boldsymbol{a_0 \cdot b_2}, \boldsymbol{a_1 \cdot b_2}),$$

derived from the secret key K *belongs to the kernel of* M*. Observe that we reduced the problem of solving the system of non-linear equations (3.3.25) to computing the kernel space of the matrix* M*. In this case, the kernel space has dimension 1 and is generated by the vector* $(1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1)$*. From this, one can easily read out the solution* $\boldsymbol{a_0} = 1$*,* $\boldsymbol{a_1} = 1$*,* $\boldsymbol{b_0} = 1$*,* $\boldsymbol{b_1} = 0$ *and* $\boldsymbol{b_2} = 1$*. which is consequently generated by* V_K*.*

The whole linearization method is described in Algorithm 3.3.

Algorithm 3.3.
Solving a system of equations with the linearization method
Input: *A system of equations* $f_1(X) = 0, \ldots, f_N(X) = 0$ *with* $f_i(X) \in \mathbb{F}_q[X]$.
Output: *All roots*

1: *Let* $\mathcal{E} \subset \{0, \ldots, q-1\}^n$ *be the set of all occurring exponents, that is* $f_i(X) = \sum_{E \in \mathcal{E}} c_E^{(i)} X^E$ *for all* f_i.
2: *Choose an arbitrary ordering in* $\mathcal{E}$.
3: *Create an empty matrix* M *of size* $N \times |\mathcal{E}|$. *The rows are indexed by the functions* f_i *and the columns by the exponents* $E \in \mathcal{E}$.
4: *For all* f_i *and* $E \in \mathcal{E}$, *the entry in row* f_i *and column* E *is set to* $c_E^{(i)}$
5: *Compute with Gaussian elimination a basis of the kernel of* M, *that is a maximum set of linearly independent vectors* $V \in \mathbb{F}_q^{|\mathcal{E}|}$ *such that* $M \cdot V = \overrightarrow{0}$.
6: *Derive from the kernel vectors all roots.*
7: ***return*** K

Observe that, as we assume that the system of equations has one unique solution, step 6 is quite easy. In general, one cannot expect to (efficiently) use the linearization method because of the following two reasons. Firstly, the degree of the functions f_i may be as high as $n \cdot (q-1)$ such that the effort to compute the kernel is exponential in n. Secondly, the number of equations might not be big enough, such that the linearized system of equations has more than one solution, although the original system of equations has only one.

However, the systems of equations occurring in algebraic attacks are far from being random. On the contrary, they possess the following remarkable properties:

1. The degree of the equations are all bounded by a value d which is independent of n.

2. The more keystream elements z_t are known, the more equations can be generated.

This makes (ι, m)-combiners perfect candidates for an algebraic attack with linearization.

However, algebraic attacks with linearization have their drawbacks too. First of all, they require that an attacker can set up at least $\mu_q(n, d) - 1$ different equations. Apart from being a rather unrealistic assumption, it implies

the additional problem of storing and accessing these equations. Hence, it is not surprising that our simulations indicated that memory is the bottle neck of this approach (see Section 3.3.4). Finally, there is still the open problem of how to determine the number of keystream elements needed to get enough linearly independent equations. In practice, one wouldn't know which keystream elements need to be known to generate sufficiently many linearly independent equations. Instead, one would use the known keystream elements piece by piece to generate equations. At this one cannot exclude the possibility that some of the equations are *not* linearly independent from the others. Therefore, it is an open question to derive a general estimation on how many and which keystream elements need to be known. Anyway in [ArmK03] we made some simulations regarding reduced versions of E_0. We generated a keystream and generated equations until the number of linearly independent equations was equal to $\mu_q(n,d)-1$. Let N be this value. It turned out for the test cases considered that N was close to the number of monomials. Therefore we assume that not much more than $\mu_q(n,d)$ known successive keystream elements are needed in most cases, but a concise answer to this question is not in sight.

Despite of these problems, from a complexity theoretical point of view the linearization approach possesses several interesting properties. First of all, if we ignore the problem how to get enough equations in practice, the estimated time and space efforts are polynomial in n but exponential in d. This is quite surprising as attacks usually have an effort exponential in n. Moreover, Gaussian elimination is a well studied method which has undergone several improvements (e.g., see Strassen's method [Str69]) where the time and memory effort are known more or less precisely.

3.3.3 Other methods

If not enough linearly independent equations are available, the kernel space has a dimension greater than 1 and therefore contains other vectors which are not derived from V_K. In this case, one might exhaustively test all vectors if the dimension of the kernel space is not too big. If this is not feasible, improved versions of linearization may succeed. We give a brief overview over existing methods.

In [KipS99], Kipnis and Shamir presented the relinearization algorithm for solving a system of quadratic equations over the finite field $\mathbb{F}_2$. The XL algorithm (XL stands for eXtended Linearization), introduced at Eurocrypt'00 [CouKPS00] by Shamir, Patarin, Courtois and Klimov, can be seen as a simplified and improved version of relinearization. Given a system of nonlinear equations, additional equations (and possibly new terms also)

may be gained by multiplying all equations with all possible monomials of some degree $\leq D$. In the lucky case, one gets enough new equations. The XL algorithm has proved to be useful in the cryptanalysis of the Toyocrypt cipher [Cou02]. It is an open question under which conditions the XL algorithm is successful. The authors proved that XL is as least as powerful as relinearization. On the other hand, in [ArsFIKS04] it was shown that XL is inferior compared to Gröbner bases algorithms as F_4 and F_5.

In [CouP02], the XSL algorithm, an extension of XL which uses the sparsity of systems of equations, was introduced by Courtois and Pieprzyk. XSL stands for eXtended Sparse Linearization. In the XSL algorithmthe equations are carefully multiplied by selected monomials. The idea is to use products of monomials that already appear in the system of equations. In their paper, they discuss how the XSL algorithm may be useful to mount attacks on AES and Serpent. Indeed, the correctness and the complexity of these attacks were subject of ongoing discussions. Recently, in [CidL05], it was showed that the XSL algorithm is not capable of solving the system of equations derived from AES.

We want to point out that these algorithms are not necessarily less suited to solve special systems of equations. In certain cases, a clever adaption of these ideas may lead to better results. However, for a general treatment of algebraic attacks, Gröbner bases methods and linearization are the two most important algorithms in algebraic attacks.

3.3.4 Experimental results

As explained in the previous section, Gröbner bases methods and linearization are the state-of-the-art algorithms to solve systems of equations appearing in algebraic attacks. We address now the question whether these algorithms imply practical attacks. Here, we suppose that an adversary is able to observe as many keystream elements as he wants, and the only difficulty lies in generating and solving a system of equations.

For this purpose, we cite our experimental results on algebraic attacks on reduced versions of E_0 from [ArmBI06]. Here, the original LFSRs of the E_0 were replaced by shorter LFSRs to reduce the keysize. To generate the systems of equations, we used the $\boldsymbol{Z}$-functions of degree 4 over $r = 4$ clocks, derived in [ArmK03] for the original E_0. We tested three different kinds of attacks: brute force, algebraic attacks using the linearization method and using the Gröbner bases algorithm F_4. In the first case, all possible values of K were tried until the correct one was found. In the second case, we produced keystream bits and created the according equations until the systems of equations could be solved by linearization. In the third case,

$\|K\|$	Brute force		Linearization			Gröbner bases		
	Time	Data	Time	Memory	Data	Time	Memory	Data
16	11s	40	$<$ 1m	$\leq$6.04	$\leq 2^{11.3}$	2m 2 s	65.49	350
17	32s	42	$<$ 1m	$\leq$9.85	$\leq 2^{11.65}$	17m 42 s	180.08	350
18	46s	44	$<$ 1m	$\leq$15.63	$\leq 2^{11.98}$	39m 23 s	303.37	350
19	1m 28s	46	$<$ 1m	$\leq$24.19	$\leq 2^{12.3}$	1h 12m	460.27	350
20	3m 44s	48	$<$ 1m	$\leq$36.61	$\leq 2^{12.6}$	2h 31m	696.63	350
22	18m 13s	52	52s	$\leq$79.13	$\leq 2^{13.15}$	14 h 14m	2150.33	400
23	20m 33s	54	1m 9s	$\leq$113.37	$\leq 2^{13.41}$	$>$ 24h	$>$3060.92	500
24	1h 12m	56	2m 2s	$\leq$159.96	$\leq 2^{13.66}$	-	-	
25	2 h	58	3m	$\leq$222.55	$\leq 2^{13.9}$	-	-	
26	4h 22m	60	4m 38s	$\leq$305.64	$\leq 2^{14.13}$	-	-	
27	-	-	6m 36s	$\leq$414.74	$\leq 2^{14.35}$	-	-	
28	-	-	9m 21s	$\leq$556.57	$\leq 2^{14.56}$	-	-	
29	-	-	16m 6s	$\leq$739.21	$\leq 2^{14.76}$	-	-	
30	-	-	20m 51s	$\leq$972.36	$\leq 2^{14.96}$	-	-	

Table 3.3.3: Experimental results on different attacks on E_0 with reduced key sizes

we limited the number of known keystream bits and tried to solve it with an implementation of the Gröbner bases algorithm F_4. The results are displayed in table 3.3.4. The time effort is given in hours, minutes and seconds, the memory consumption in megabytes and data is the number of keystream bits assumed to be known. The memory consumption for the brute force attack was constantly 3.1 megabytes.

The results are not directly comparable. The simulations have been conducted for different occasions and were originally not intended for a direct comparison. The linearization method and F_4 have been written in Java, the brute force attack simulated in the computer algebra system `Magma`. Brute force and Gröbner bases have both been simulated on the same computer, a Pentium 4 with 3.5 GHz and 3 Gigabytes RAM. The linearization attack has been simulated on a Pentium 4-M with 1.8 GHz and 512 Megabytes RAM. Since the amount of data and memory has not been recorded during the simulations on linearization, we provide some upper bounds instead. To compute the Gröbner bases, we used the F_4 algorithm and results from [BecW93].

Nevertheless, the simulation results give some clues about the attacks' behaviour with respect to the key size. As expected, the linearization method is the most successful, outperforming the two other attacks by far. It was possible to derive the secret value K for $|K| \leq 30$ within a rea-

sonable amount of time of several minutes. Indeed, instances with key sizes of 34 and higher were intractable due to memory shortness. Further on, the knowledge of several kilobytes of keystream has to be required, a rather unrealistic assumption.

In this respect, Gröbner bases are much more realistic. It was possible to find the value of K even if only the moderate amount of 350 keystream bits is known. Indeed, we found out that the attack is only feasible if the amount of data is significantly larger than $|K|$. Even worse, our naive brute force attack implementation proved to be faster and required less data and memory. However, we expect the opposite situation for larger key sizes.

Another problem are the memory requirements. Since the memory is mainly used to store sparse matrices, this bound may be pushed further by developing appropriate data structures and algorithms. Nevertheless, the experiments indicate that memory is again the bottle neck of this attack.

Concluding, one can say that algebraic attacks work in principle. In particular, the comparatively low time effort of the linearization method indicates that this approach actually beats other approaches. However, one has to admit that these attacks are far from being practical. Despite the memory requirements, the necessity of knowing huge parts of the keystream makes algebraic attacks based on linearization a rather theoretical construct than a realistic threat. Nonetheless, one must not forget that the whole area is relatively young, making further improvements imaginable. For example, so far a systematic examination of guessing some keybits to simplify the system of equations, as it has been done for example in [KraS06], has not been made. E. Zenner and D. Bernstein suggest that parallel computing might help to overcome some of the memory problems [Zen06]. Besides, the whole complexity is closely related to the degree of the equations. If one would find, for example, quadratic $\boldsymbol{Z}$-functions for E_0, the situation would change dramatically as overdefined system of quadratic equations can be solved within few days in certain cases [Ars04]. Thus, the existence of low-degree $\boldsymbol{Z}$-functions, methods to find them or design principles to avoid them is an important subject. Consequently, a lot of work has been done on these questions, which will be presented in the next chapter.

3.4 Summary and effort estimation

Coming to the end of this chapter, we give a short summary of what we have discussed so far. As sketched in Algorithm 3.1, algebraic attacks are based on constructing and solving systems of equations in the known keystream elements z_t and the unknown key K. Different ways exist for setting up the system of equations and computing the solution. From our point of view, it is preferable in the general case to build the system of equations from $\boldsymbol{Z}$-functions and to solve it with the linearization method (see Section 3.3.2).

The advantage of the linearization method is that it is (more or less) easy to analyze. Given enough linearly independent equations, the final step consists in using Gaussian elimination to compute the solution, from which K can be easily recovered. However, it is still an open problem to determine *in advance* the number of known keystream elements required for a successful attack. Simulations on reduced versions of E_0 in [ArmK03] indicated that most of the equations are linearly independent if all considered keystream elements are successive. But this doesn't need to be true in other cases, making further research necessary.

We end this chapter with an estimation on the effort of algebraic attacks. We thereby concentrate on the general case where $\boldsymbol{Z}$-functions are used for the equations and linearization is deployed for computing the solution. Let $\mu_q(n,d)$ and $\mu_q(\iota \cdot r, d)$ denote the number of monomials of degree $\leq d$ in n and $\iota \cdot r$ variables, respectively. The first step consists in finding $\boldsymbol{Z}$-functions. Observe that this has to be done only once, as the $\boldsymbol{Z}$-functions can be reused for further attacks on the same (ι, m)-combiner with different keys. For this step, it is not possible to specify the exact amount, as (again) various different approaches exist. In some cases, one can find them by careful analysis of the (ι, m)-combiner (e.g., see [ArmK03, LeeKHHM04, ChoP04]). A general method to find $\boldsymbol{Z}$-functions of degree $\leq d$ will be given in the next chapter, which is based on computing the kernel of a matrix of size $|X_{\boldsymbol{Z}}| \times \mu_q(\iota \cdot r, d)$ where $X_{\boldsymbol{Z}}$ is the set of all possible values for $K_t, \ldots, K_{t+r-1}$ if $(z_t, \ldots, z_{t+r-1})$ is equal to $\boldsymbol{Z}$. A formal definition and additional explanations will be given in the next chapter. Here we will only use that $|X_{\boldsymbol{Z}}|$ is upper bounded by $q^{m+(\iota-1)\cdot r}$.

The space required to store the matrix is in $\mathcal{O}(q^{m+(\iota-1)\cdot r} \times \mu_q(\iota \cdot r, d))$ and the number of basic operations in $\mathbb{F}$ to compute the kernel is in $\mathcal{O}(q^{m+(\iota-1)\cdot r} \times \mu_q(\iota \cdot r, d)^2)$.[6] We ignore the effort to set up the matrices as it is comparatively low. As this step has to be repeated for any $\boldsymbol{Z} \in \mathbb{F}_q^r$, the total effort is the

[6] For simplicity, we consider only naive Gaussian elimination and ignore more advanced methods.

value given above multiplied by q^r. As already mentioned, other approaches might exist in some cases to get $\boldsymbol{Z}$-functions, but as far as we know this is the only one which works in general.

The next step is to set up a system of equations, built from the $\boldsymbol{Z}$-functions derived in the first step. As we use linearization in the final step to get the solution, we assume that the equations are stored in a matrix of appropriate size. More precisely, the matrix has $\mu_q(n,d)$ columns, where each column corresponds to exactly one of the $\mu_q(n,d)$ possible monomials. In the concrete case the number of monomials will probably be lower, resulting in both a lower time and memory consumption. But for the general estimation, we have to consider the upper bound.

The rows represent the different equations. That is, for each equation $F_{\boldsymbol{Z}}(K_t) = 0$, $F_{\boldsymbol{Z}}(K_t)$ can be expressed by a linear sum of the $\mu_q(n,d)$ monomials, which can be bijectively mapped to a vector of size $\mu_q(n,d)$. We assume that this bijection can be performed with about $\mathcal{O}(\mu_q(n,d))$ operations. So, the space and time effort to generate the matrix is in $\mathcal{O}(N \cdot \mu_q(n,d))$ with N being the number of rows respectively equations. We hereby neglect the time to read the r keystream elements and to access the $\boldsymbol{Z}$-functions as this amount is comparatively low. As $N \geq \mu_q(n,d) - 1$ is a necessary condition to finally have enough linearly independent equations, we assume for simplicity that $N = c \cdot \mu_q(n,d)$ with $c \geq 1$. Then, we have an overall time and space effort of approximately $\mu_q(n,d)^2$.

The final step is to compute a non-trivial nullvector of the generated matrix, which is doable with Gaussian elimination. The time effort is in $\mu_q(n,d)^3$.

Finally, we take a look at the number of keystream elements an attacker needs to know to apply a successful attack. Let N denote the number of runs $z_t, \ldots, z_{t+r-1}$ the attacker has knowledge of. Observe that the knowledge of one run is required to set up one equation. This means that the overall number of known keystram elements is between $N + r - 1$, if all elements are successive, and $r \cdot N$, if they are all distinct.

Next, we will derive a lower bound for N. As $\mu_q(n,d)$ grows with d, an attacker would prefer only equations with the minimum degree. Thus, we assume that he considers for each $\boldsymbol{Z} \in \mathbb{F}^r$ only those $\boldsymbol{Z}$-functions which have the lowest degree. However, it still could be that the degree for one choice of $\boldsymbol{Z}$ is higher than for the other (e.g., see Example 3.8). Thus, we define $\mathcal{Z} \subseteq \mathbb{F}^r$ to be the set of these values $\boldsymbol{Z}$ such that the degree of the $\boldsymbol{Z}$-functions are minimal. More formally, it holds that $\deg(F_{\boldsymbol{Z}}) = \deg(F_{\boldsymbol{Z}'})$ for all $\boldsymbol{Z}, \boldsymbol{Z}' \in \mathcal{Z}$ but $\deg(F_{\boldsymbol{Z}}) < \deg(F_{\boldsymbol{Z}'})$ if $\boldsymbol{Z} \in \mathcal{Z}$ and $\boldsymbol{Z}' \in \mathbb{F}^r \setminus \mathcal{Z}$. Now, if we assume for any $\boldsymbol{Z} \in \mathcal{Z}$ that exactly a fraction of N/q^r known runs is equal to $\boldsymbol{Z}$ and as each of this parts gives $i(\boldsymbol{Z})$ many $\boldsymbol{Z}$-functions, the total number of equations based on the $\boldsymbol{Z}$-functions is $i(\boldsymbol{Z}) \cdot N/q^r$. As this holds for each

Step	Time	Space	Keystream
Finding $\boldsymbol{Z}$-functions	$\mathcal{O}(q^{m+\iota\cdot r}\cdot\mu_q(\iota\cdot r,d)^2)$	$\mathcal{O}(q^{m+\iota\cdot r}\cdot\mu_q(\iota\cdot r,d))$	-
Setting up linearized system of eqs.	$\mathcal{O}(\mu_q(n,d)^2)$	$\mathcal{O}(\mu_q(n,d)^2)$	$\geq \frac{q^r\cdot\mu_q(n,d)}{\sum_{\boldsymbol{Z}\in\mathcal{Z}} i(\boldsymbol{Z})}$
Solving	$\mathcal{O}(\mu_q(n,d)^3)$	$\mathcal{O}(\mu_q(n,d)^2)$	-

Table 3.4.4: Effort estimations for an algebraic attack on a (ι, m)-combiner with keysize n, based on $\boldsymbol{Z}$-functions and linearization

$\boldsymbol{Z}\in\mathcal{Z}$, we have altogether $N/q^r\cdot\sum_{\boldsymbol{Z}\in\mathcal{Z}} i(\boldsymbol{Z})$ equations. As we require that this value is at least $\mu_q(n,d)-1$ (see above), we get the following lower bound for N:

$$N \geq \frac{q^r\cdot\mu_q(n,d)}{\sum_{\boldsymbol{Z}\in\mathcal{Z}} i(\boldsymbol{Z})}.$$

The whole algebraic attack is summarized in Algorithm 3.4 and the efforts in Table 3.4.4.

From a theoretical point of view algebraic attacks are very interesting. Observe that the main time effort is $\mathcal{O}(\mu_q(n,d)^3)=\mathcal{O}(n^{d\cdot 3})$ in the case of $\mathbb{F}=\mathbb{F}_2$. Recall that the $\boldsymbol{Z}$-functions depend only on the update function f and the memory update function Ψ, but not on the choice of the LFSRs. In particular, the degree d remains constant, even if one chooses to increase the keysize n. That means that the number of operations grows only polynomial in the keysize n. As far as we know, this is the only class of attacks with this property. For all the other attacks mentioned in Section 2.6, the time effort is exponential in n or unpredictable. Thus, algebraic attacks have a far better asymptotic runtime than any other attack published so far.

Consequently, for some (ι, m)-combiners, the theoretically estimated time effort for an algebraic attack lies below the estimates for other attacks [CouM03, ArmK03, Cou03, LeeKHHM04, ChoP04].

One example is the Bluetooth keystream generator E_0, for which we proposed an algebraic attack in [ArmK03]. By analyzing the functions f and Ψ, we derived one $\boldsymbol{Z}$-function of degree four for each $\boldsymbol{Z}\in\mathbb{F}_2^4$. Table 3.4.5 displays the efforts for this algebraic attack, compared to the best previously published attack. Later on, fast algebraic attacks [Cou03] improved this attack.

Algorithm 3.4.
Algebraic attack on a (ι, m)-combiner with $\boldsymbol{Z}$-functions and linearization

Input: *A (ι, m)-combiner, initialized to a secret value $\mathcal{S}_0 = (Q_0, K)$ and the knowledge of some the keystream elements generated by this setting*

Output: *The secret key K*

1: *(Only once) Fix $r \geq 1$ and find $i(\boldsymbol{Z})$ $\boldsymbol{Z}$-functions for each $\boldsymbol{Z} \in \mathbb{F}^r$.*
2: *(Only once) Let $\mathcal{Z} \subseteq \mathbb{F}^r$ be the set of these values $\boldsymbol{Z}$ such that $\deg(F_{\boldsymbol{Z}}^{(i)})$ is minimal.*
3: *Initialize an empty system of equations.*
4: *For each sequence of known keystream elements $(\boldsymbol{z_t, \ldots, z_{z+r-1}}) = \boldsymbol{Z}$, add the following equations to the system of equations if $\boldsymbol{Z} \in \mathcal{Z}$. Otherwise, do nothing.*

$$\begin{aligned} F_{\boldsymbol{Z}}^{(1)}(K \cdot L^t \cdot P, \ldots, K \cdot L^{t+r-1} \cdot P) &= 0 \\ &\ldots \\ F_{\boldsymbol{Z}}^{(i(\boldsymbol{Z}))}(K \cdot L^t \cdot P, \ldots, K \cdot L^{t+r-1} \cdot P) &= 0 \end{aligned}$$

5: *If the number of linearly independent equations is one less than the number of monomials in the system of equations, linearize the system of equations and solve it with Gaussian elimination*
6: *Recover K from the solution*
7: ***return*** K

To the best of our knowledge, the algebraic attacks on E_0 are still the fastest attacks on *single* E_0. As the Bluetooth keystream generator consists of applying E_0 once to generate 128 bit internal bits, permuting them by a known permutation and feeding them into E_0 again to finally generate the keystream, any attack on E_0 can be extended to an attack on the whole stream cipher. The attack is applied twice: first to reconstruct the internal bits and second to get the original LFSRs' initial states. But, as in the Bluetooth encryption system the secret key is changed after 2745 clocks, an attacker never gets the required number of 2^{23} keystream bits.[7] Thus, algebraic attacks are the fastest attacks on E_0 but not on the Bluetooth stream cipher. For the Bluetooth cipher exist efficient correlation attacks [LuV04, LuMV05], exploiting the linear keyschedule.

[7] An alternative could be to use Gröbner bases (Sec. 3.3.1), but here the time effort cannot be estimated.

Attack	Time	Precomp.	Memory	Keystream
Backtracking [FluL01]	$\text{const} \cdot 2^{73}$	-	≈ 10638	$\text{const} \cdot 2^{43}$
BDD-based [Kra02]	$\text{const} \cdot 2^{77}$	-	$\text{const} \cdot 2^{77}$	≈ 128
Algebraic attack [ArmK03]	$\text{const} \cdot 2^{67.58}$	-	$\text{const} \cdot 2^{46.14}$	$\text{const} \cdot 2^{23.07}$
Fast alg. attacks [Cou03]	$\text{const} \cdot 2^{54.51}$	$\text{const} \cdot 2^{43}$	$\text{const} \cdot 2^{36.84}$	$\text{const} \cdot 2^{23.44}$

Table 3.4.5: Attack efforts for some attacks on E_0.

Of course, the estimates regarding algebraic attacks say little about their practicability. For example, they require the knowledge of many keystream elements which is rather unrealistic. Another problem is the huge memory requirements. So, it is no surprise that our simulations showed that memory is actually the bottle neck of the whole attack (see Section 3.3.4). Thus, apart from the excellent theoretical estimations for the efforts, practical algebraic attacks are out of reach for the moment.

However, one mustn't forget that the basic idea of algebraic attacks on (ι, m)-combiners is relatively young, dating back to 2003 (e.g., see [CouM03, ArmK03]). The rapidly growing number of publications on this subject proves the community's interest on these attacks. So, further improvements might be found in future, making these attacks theoretically *and* practically interesting.

4 On the equations in algebraic attacks

4.1 Criteria for low degree equations

As mentioned before, the existence of low degree Z-functions is crucial for an (efficient) algebraic attack. Following our discussions from Section 3.3.2, we face in this section the question of their existence. Therefor, we restate our general framework from [Arm04b, Arm05a] which is an extension of prior results in [ArmK03, MeiPC04].

4.1.1 Previous works and algebraic immunity

In the case of simple combiners over $\mathbb{F} = \mathbb{F}_2$, Courtois and Meier [CouM03] were the first to consider the question when functions $F_{(\boldsymbol{z})} : \mathbb{F}^{\iota} \to \mathbb{F}$ of degree $\leq d$ exist such that $F_{(\boldsymbol{z})}(K_t) = 0$ whenever $z_t = \boldsymbol{z}$. Furtheron, they examined the existence of equations (3.2.6) with a low degree. They presented several different scenarios in which appropriate functions F exist. Later on, this has been simplified to one criterion in [MeiPC04]. It turned out that for each function F it either holds $F \cdot f \equiv 0$ or $F \cdot (f \oplus 1) \equiv 0$ (see also example 3.5). They named functions g such that $F \cdot g \equiv 0$ as *annihilators* of F (see also Definition 4.2).

Furthermore, they introduced the notion **algebraic immunity**

$$AI(f) := \min\{\deg(g) \mid f \cdot g \equiv 0 \text{ or } (f \oplus 1) \cdot g \equiv 0\}. \tag{4.1.1}$$

As each balanced Boolean function in n unknowns has an algebraic immunity of $\leq \lceil n/2 \rceil$ (see [CouM03] for a proof), they called a function f with $AI(f) = \lceil n/2 \rceil$ an **algebraically immune function**. In other words, for algebraically immune functions all possible 1-functions have a degree of $\lceil n/2 \rceil$ or higher, which is the best one can hope for.

The introduction of the term "algebraic immunity" was an important step for understanding algebraic attacks and consequently evoked numerous subsequent works (e.g., see [Car04, DalGM04, Lob05, QuFL05, BraP05, DalGM05, DalMS05, Car05, ArmCGKMR06]). However, for several reasons we think that both the name "algebraic immunity" and the definition of algebraic immunity should be slightly adapted. First of all, a designer might be misled to believe that an algebraically immune output function provides the best resistance against algebraic attacks. But this is not necessarily the case as fast algebraic attacks (see [ArmCGKMR06] or Chapter 5, especially Section 5.5.4) or divide-and-conquer attacks (see [Gol04] or Section 5.3) might be possible. This is illustrated by the following example.

Example 4.1. *Consider the output function $f : \mathbb{F}_2^3 \to \mathbb{F}_2$, $(a, b, c) \to z := a \oplus c \cdot (a \oplus b)$ from the Geffe-Generator (see Section 2.5.2). Here, a, b and c denote*

the outputs from three LFSRs $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ and z the corresponding keystream bit. One can check[1] *that $AI(f) = 2 = \lceil 3/2 \rceil$. Hence, f is algebraically immune. Every valid equation $F(a, b, c, \boldsymbol{z}) = 0$ has a degree of 2 or more.*

However, f is not optimal. Take for example $F(a, b, c, z) := c \cdot (f(a, b, c) \oplus z) = c \cdot (b \oplus z)$. The degree of F is 2, but it is independent of a. In addition, we can exploit the structure of $F(a, b, c) = c \cdot b + c \cdot z$ to apply a fast algebraic attack to mount a system of linear *equations in c only.*[2] *By solving it, one gets the initial values of LFSR $\mathcal{C}$. Once these are known, each clock t where z_t and c_t are known reveals immediately that $a_t = z_t$ if $c_t = 0$ or $b_t = z_t$ if $c_t = 1$. Thus, figuring out the initial states from $\mathcal{A}$ and $\mathcal{B}$ again only requires to solve a system of linear equations.*

Summing up, the whole cipher can be broken by solving three systems of linear equations, although $AI(f) = 2$ shows that only quadratic 1-functions exist. This shows that the definition of algebraic immunity is not enough to characterize the vulnerability of a simple combiner against algebraic attacks.

Thus, an algebraically immune function might still be vulnerable against other kinds of algebraic attacks. In [DalMS05, Remark 1], the authors come to the same conclusion and propose to rename it to **annihilator immunity**.

A second drawback of the definition is that it fits only for simple combiners over the field $\mathbb{F}_2$ but not for general (ι, m)-combiners over arbitrary fields.

In [Arm05a], the ideas from [MeiPC04] (and in fact from [ArmK03] too) have been extended to a general theoretic framework which allows to set up a criterion for the existence of low degree equations for general combiners over arbitrary finite fields. In principle, it shows that there is a 1:1-correspondence between valid equations and annihilators of a specific set. Thus, the security of a (ι, m)-combiner against standard algebraic attacks is closely connected to the question if these sets have low degree annihilators. This leads to a slightly different definition of "algebraic immunity" which fits to a broader class of keystream generators (see Definition 4.20). However, to avoid confusion with the well-established definition "algebraic immunity" but to point out the connection with it, we will introduce the new term "lowest annihilator degree" (or short lad) to reflect this circumstance and will *not* use "algebraic immunity". We want to stress that we are not aiming to replace the term or definition of "algebraic immunity". Our reasons to use an alternative definition and name in this thesis are

1. to address simple combiners and combiners with memory over arbitrary finite fields by the same theory and

[1] Appropriate algorithms will be presented later in Section 4.2.
[2] How this can be achieved will be discussed in Chapter 5.

2. to avoid a mistake between the widely known and accepted definition of "algebraic immunity" and our definition.

4.1.2 Z-functions and annihilators

First, recall the definition of $\boldsymbol{Z}$-functions: a function $F_{\boldsymbol{Z}}$ is a $\boldsymbol{Z}$-function if for each part of the keystream that is equal to $\boldsymbol{Z}$ it is zero on the corresponding inputs. More formally, a $\boldsymbol{Z}$-function fulfills

$$\forall Q, X_1, \ldots, X_r : f_\Psi(Q, X_1, \ldots, X_r) = \boldsymbol{Z} \Rightarrow F_{\boldsymbol{Z}}(X_1, \ldots, X_r) = 0 \tag{4.1.2}$$

where f_Ψ is the extended output function (see definition on page 37). That is $F_{\boldsymbol{Z}}$ gives zero on the projection of the preimage of $\boldsymbol{Z} = (\boldsymbol{z_1}, \ldots, \boldsymbol{z_r})$ under f_Ψ. We introduce the following identifier for this set:

$$X_{(\boldsymbol{z_1},\ldots,\boldsymbol{z_r})} = X_{\boldsymbol{Z}} := \{(X_1, \ldots, X_r) \mid \exists Q : f_\Psi(Q, X_1, \ldots, X_r) = \boldsymbol{Z}\}. \tag{4.1.3}$$

For the case that $r = 1$, we omit the indices and write $X_{(\boldsymbol{z})}$ instead of $X_{(\boldsymbol{z_1})}$. One sees that $F_{\boldsymbol{Z}}$ is a $\boldsymbol{Z}$-function if and only if it is zero on the set $X_{\boldsymbol{Z}}$. This leads to the notion of annihilators:

Definition 4.2. *Let S be a subset of $\mathbb{F}^n$. An **annihilator** of S is a function $g : \mathbb{F}^n \to \mathbb{F}$ such that $g(\boldsymbol{X}) = 0$ for all $\boldsymbol{X} \in S$. The set of annihilators of S is denoted by*

$$\mathrm{ann}(S) := \{g : \ g \,\text{annihilator of}\, S\}. \tag{4.1.4}$$

Let $f : \mathbb{F}^n \to \mathbb{F}$ be a function. $g : \mathbb{F}^n \to \mathbb{F}$ is an annihilator of f if $f \cdot g \equiv 0$, i.e. $f(\boldsymbol{X}) \cdot g(\boldsymbol{X}) = 0$ for all $\boldsymbol{X} \in \mathbb{F}^n$. Furtheron, we define the set

$$\mathrm{ann}(f) := \{g : \ g \,\text{annihilator of}\, f\}. \tag{4.1.5}$$

With the observations made above, the following theorem is quite straightforward:

Theorem 4.3. *For a given (ι, m)-combiner and $\boldsymbol{Z} \in \mathbb{F}^r$, a function $F : \mathbb{F}^{r \cdot \iota} \to \mathbb{F}$ is a $\boldsymbol{Z}$-function if and only if F is an annihilator of the set $X_{\boldsymbol{Z}}$.*

Hence, to derive $\boldsymbol{Z}$-functions with a degree less or equal than a given bound d, one can proceed as follows:

1. Compute the set $X_{\boldsymbol{Z}}$.
2. Look for annihilators of $X_{\boldsymbol{Z}}$ with a degree less or equal than d.

In some cases, more direct methods may exist. However, this approach is (at least in theory) generally applicable. In the rest of this section, we will examine more the sets $X_{\boldsymbol{Z}}$ and the theory of annihilators.

As far as we know, it was proposed in [ArmK03] for the first time to consider the sets $X_{\boldsymbol{Z}}$ and to examine their properties.[3] It was also pointed out that $\boldsymbol{Z}$-functions can be found by solving a system of linear equations. In the next section, we will take a closer look on how to actually compute $\boldsymbol{Z}$-functions and present better methods.

4.1.3 The sets $X_{\boldsymbol{Z}}$

By definition, the set $X_{\boldsymbol{Z}}$ consists of all inputs which can lead to the output $\boldsymbol{Z}$. Observe, however, that $X_{\boldsymbol{Z}}$ is defined only by the functions f and Ψ but doesn't take into account the matrices L and P. This means, that for $(\boldsymbol{X_t}, \ldots, \boldsymbol{X_{t+r-1}}) \in X_{\boldsymbol{Z}}$, it is not guaranteed that $\boldsymbol{K} \in \mathbb{F}^n$ exists such that $\boldsymbol{X_i} = \boldsymbol{K} \cdot L^i \cdot P$.

Nonetheless, it makes sense to define $X_{\boldsymbol{Z}}$ in this way. As long as r is smaller than the length of the shortest LFSR, the conditions $X_i = K \cdot L^i \cdot P$ yield an underdefined system of linear equations, so that the existence of such a $\boldsymbol{K}$ is warranted. As the effort for computing annihilators for $X_{\boldsymbol{Z}}$ grows exponentially with r, the value of r will be in the most practical cases rather low, at least smaller than the length of the shortest LFSR. Thus, incorporating L and P into the definition would not imply any additional constraints in these cases. This shows in particular that the same $\boldsymbol{Z}$-functions remain valid if one increases the key size n and changes the matrices L and P accordingly.

In the case of simple combiners, i.e. $m = 0$, one can give a simpler description of the set $X_{(\boldsymbol{z})}$:

Proposition 4.4. *In the case of simple combiners, i.e. without any additional memory, it is* $X_{(\boldsymbol{z})} = \{(X_1, \ldots, X_r) \mid f(X_1, \ldots, X_r) = \boldsymbol{z}\} = f^{-1}(\boldsymbol{z})$.

In the case of combiners with memory, the connection between the set $X_{\boldsymbol{Z}}$, the output function f and the memory update function Ψ is not that apparent. As far as we know, no better methods exist than to generate for all inputs $(X_1, \ldots, X_r) \in \mathbb{F}^{r \cdot \iota}$ and $Q \in \mathbb{F}^m$ the outputs $\boldsymbol{Z}$ and then to sort them into the set $X_{\boldsymbol{Z}}$ (see Algorithm 4.1).

[3] In [ArmK03], the sets $X_{\boldsymbol{Z}}$ were called $NCrit_C(\boldsymbol{Z})$.

Algorithm 4.1.
Computation of the sets $X_{\boldsymbol{Z}}$ for a given (ι, m)-combiner
Input: *A (ι, m)-combiner with extended output function $f_\Psi : \mathbb{F}^{r\cdot\iota} \to \mathbb{F}^r$*
Output: *The sets $X_{\boldsymbol{Z}}$ for all $\boldsymbol{Z} \in \mathbb{F}^r$*

Set $X_{\boldsymbol{Z}} := \emptyset$ for all $\boldsymbol{Z} \in \mathbb{F}^r$
for *all $(\boldsymbol{X_1}, \ldots, \boldsymbol{X_r}) \in \mathbb{F}^{r\cdot\iota}$* ***do***
  for *all $\boldsymbol{Q} \in \mathbb{F}^m$* ***do***
    Compute $\boldsymbol{Z} := f_\Psi(\boldsymbol{Q}, \boldsymbol{X_1}, \ldots, \boldsymbol{X_r})$
    $X_{\boldsymbol{Z}} \leftarrow X_{\boldsymbol{Z}} \cup \{(\boldsymbol{X_1}, \ldots, \boldsymbol{X_r})\}$
  end for
end for
return *S*

Example 4.5. *To illustrate Algorithm 4.1, we consider the summation generator with $\iota = 2$ and $m = 1$. Table 4.1.1 displays all possible input-output combinations over $r = 2$ clocks. From this, one can derive for a given output $\boldsymbol{Z} \in \mathbb{F}_2^2$ the set of all possible inputs.*

Take for example the output $\boldsymbol{Z} = (0,0)$. We check for every $(X_1, X_2) \in \mathbb{F}_2^2 \times \mathbb{F}_2^2$ if a memory state Q exists such that $F_\Psi(Q, (X_1, X_2)) = (0,0)$. The results are given below. "-" means that none of the memory states fits to the according inputs and outputs. Due to space restriction, we write (0000) instead of $((0,0),(0,0))$ etc.

Input	Q	*Input*	Q	*Input*	Q	*Input*	Q
0000	*0*	0100	-	1000	-	1100	-
0001	-	0101	*1*	1001	*1*	1101	*0*
0010	-	0110	*1*	1010	*1*	1110	*0*
0011	*0*	0111	-	1011	-	1111	-

We see that the set of preimages of $Z = (0,0)$ contains only the half of all 16 possible inputs. In a similar way, one can determine all sets $X_{\boldsymbol{Z}}$:

$$\begin{aligned} X_{(0,0)} &= \{(0000),(0011),(0101),(0110),(1001),(1010),(1101),(1110)\} \\ X_{(0,1)} &= \{(0001),(0010),(0100),(0111),(1000),(1011),(1100),(1111)\} \\ X_{(1,0)} &= \{(0000),(0011),(0100),(0111),(1000),(1011),(1101),(1110)\} \\ X_{(1,1)} &= \{(0001),(0010),(0101),(0110),(1001),(1010),(1100),(1111)\} \end{aligned} \tag{4.1.6}$$

Note that each output $\boldsymbol{Z}$ can be produced by only half of the 16 possible inputs X_1, X_2.

Q	0	0	0	0	0	0	0	0
X_1	(0,0)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)
X_2	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)
$\boldsymbol{Z}$	00	01	01	00	10	11	11	10

Q	0	0	0	0	0	0	0	0
X_1	(1,0)	(1,0)	(1,0)	(1,0)	(1,1)	(1,1)	(1,1)	(1,1)
X_2	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)
$\boldsymbol{Z}$	10	11	11	10	01	00	00	01

Q	1	1	1	1	1	1	1	1
X_1	(0,0)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)
X_2	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)
$\boldsymbol{Z}$	10	11	11	10	01	00	00	01

Q	1	1	1	1	1	1	1	1
X_1	(1,0)	(1,0)	(1,0)	(1,0)	(1,1)	(1,1)	(1,1)	(1,1)
X_2	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)
$\boldsymbol{Z}$	01	00	00	01	11	10	10	11

Table 4.1.1: All possible input-output combinations over 2 clocks for the summation generator with $\iota = 2$ inputs and $m = 1$ memory bits

As discussed in Section 2.5, the summation generator belongs to the class of permutation invariant combiners. This means that (beside of Q_t of course) only the value $|X_t|$ plays a role in the computation of the output z_t and the next memory state Q_{t+1}. Thus, one can rewrite the sets $X_{\boldsymbol{Z}}$ in (4.1.6) to

$$\begin{array}{rcl} X_{(0,0)} & = & \{[00],[02],[21],[11]\} \\ X_{(0,1)} & = & \{[01],[20],[22],[10],[12]\} \\ X_{(1,0)} & = & \{[10],[12],[00],[02],[21]\} \\ X_{(1,1)} & = & \{[11],[01],[20],[22]\} \end{array} \tag{4.1.7}$$

where $[\boldsymbol{x_1 x_2}] = \{(X_1, X_2) \in \mathbb{F}_2^4 \mid |X_1| = \boldsymbol{x_1}, |X_2| = \boldsymbol{x_2}\}$.

Observe that no assumptions are made on the method how the inputs are generated. Thus, the theory in this section remains valid if other finite state machines than LFSRs are exploited to create the internal input stream.[4]

[4]Indeed, for an algebraic attack, a linear finite state machine is crucial as otherwise the degree of the equations in K can go up.

Only the output function f and the next memory state function Ψ play a role in the existence of $\boldsymbol{Z}$-functions. This has the advantage that a designer can adapt the keysize (i.e., the size of the LFSRs) without risking that the degree of the $\boldsymbol{Z}$-functions change. In Section 4.3, we will discuss several design principles how to design (ι, m)-combiners such that it is guaranteed that no $\boldsymbol{Z}$-functions with a degree below a given bound exist. This then provides some security against algebraic attacks.

Summing up, the existence of (low degree) $\boldsymbol{Z}$-functions depends only on the set $X_{\boldsymbol{Z}}$ and hence only on f and Ψ. The first observation is that non-trivial $\boldsymbol{Z}$-functions do only exist if $X_{\boldsymbol{Z}}$ is a real subset of $\mathbb{F}^{\iota \cdot r}$ as $\mathrm{ann}(\mathbb{F}^{\iota \cdot r})$ contains only the all-zero function. Thus, we turn our attention first to the question whether $X_{\boldsymbol{Z}}$ is a real subset of $\mathbb{F}^r$ or not. By Proposition 4.4 we know that $X_{\boldsymbol{Z}} = f^{-1}(\boldsymbol{z})$. Hence, as long as f is not a constant function, it is sure that $X_{\boldsymbol{Z}} \subset \mathbb{F}^r$. But in the case of combiners with memory, the situation is not that clear. Especially, if r is small, one often encounters the case that $X_{\boldsymbol{Z}} = \mathbb{F}^r$. However, if $r > m$, the following theorem, which is an extension of a theorem from [ArmK03], shows that at least for one value $\boldsymbol{Z}$ it holds that $X_{\boldsymbol{Z}} \subset \mathbb{F}^r$. This guarantees the existence of $\boldsymbol{Z}$-functions for arbitrary (ι, m)-combiners if r is not too small.

Theorem 4.6. *Consider an arbitrary (ι, m)-combiner over a finite field $\mathbb{F}_q$. There exists at least one output $\boldsymbol{Z} \in \mathbb{F}_q^{m+1}$ such that $|X_{\boldsymbol{Z}}| \leq \frac{1}{q} \cdot q^{\iota \cdot (m+1)}$. This means that this specific output cannot be generated by all possible inputs in $\mathbb{F}^{\iota \cdot (m+1)}$.*

Proof. Note that $(Q, X_1, \ldots, X_{m+1})$ uniquely determines the output $Z \in \mathbb{F}_q^{m+1}$ and therefore $\mathbb{F}^{m+\iota \cdot (m+1)} = \dot{\bigcup}_Z f_\Psi^{-1}(Z)$. Further, by definition it is $|f_\Psi^{-1}(Z)| \geq |X_Z|$.

Assume that the proposition is not true, i.e., $|X_Z| > q^{\iota \cdot (m+1)-1}$ for each $Z \in \mathbb{F}^{m+1}$. This leads to the contradiction

$$\begin{aligned} q^{m+\iota(m+1)} &= |\mathbb{F}^{m+\iota(m+1)}| = |\dot{\bigcup}_{Z \in \mathbb{F}^{m+1}} f_\Psi^{-1}(Z)| = \sum_{Z \in \mathbb{F}^{m+1}} |f_\Psi^{-1}(Z)| \\ &\geq \sum_{Z \in \mathbb{F}^{m+1}} |X_Z| > \sum_{Z \in \mathbb{F}^{m+1}} q^{\iota(m+1)-1} = q^{m+1} \cdot q^{\iota(m+1)-1} = q^{m+\iota(m+1)}. \end{aligned}$$

Hence, $|X_{\boldsymbol{Z}}| \leq \frac{1}{q} \cdot q^{\iota \cdot (m+1)}$ for at least one $Z \in \mathbb{F}^{m+1}$ and thus $X_{\boldsymbol{Z}} \subsetneq \mathbb{F}_q^{\iota \cdot (m+1)}$. □

Thus, we know so far that $\boldsymbol{Z}$-functions always exist if the combiner is reasonably designed (e.g., no constant output function f). Next, we examine the question how to get the sets $X_{\boldsymbol{Z}}$. If the values of r, ι and m are not too big, then one can compute the sets $X_{\boldsymbol{Z}}$ by exhaustively computing the

values $f_\Psi(\boldsymbol{Q}, \boldsymbol{X_1}, \ldots, \boldsymbol{X_r}) = (\boldsymbol{z_1}, \ldots, \boldsymbol{z_r})$ for all $\boldsymbol{Q} \in \mathbb{F}_q^m$ and $\boldsymbol{X_1}, \ldots, \boldsymbol{X_r} \in \mathbb{F}_q^\iota$ and by then including $(\boldsymbol{X_1}, \ldots, \boldsymbol{X_r})$ to the set $X_{(\boldsymbol{z_1},\ldots,\boldsymbol{z_r})}$ (see also Algorithm 4.1).

A more refined method is to first compute the sets $X_{\boldsymbol{Z'}}$ for $\boldsymbol{Z'} \in \mathbb{F}^{r'}$ with $r' < r$ and then to use them to build the sets $X_{\boldsymbol{Z}}$. In the case of simple combiners, this is comparatively easy:

Proposition 4.7. *Let a fixed simple combiner with output function f be given. If the inputs $X_1, \ldots, X_r$ from r clocks are independent[5], then it holds for all $\boldsymbol{Z} = (\boldsymbol{z_1}, \ldots, \boldsymbol{z_r})$ that*

$$X_{\boldsymbol{Z}} = X_{(\boldsymbol{z_1})} \times \ldots \times X_{(\boldsymbol{z_r})}. \tag{4.1.8}$$

Proof. W.l.o.g., we can take $r = 2$, as the rest follows by induction. It holds for all $(\boldsymbol{z_1}, \boldsymbol{z_2}) \in \mathbb{F}^2$:

$$\begin{aligned} X_{(\boldsymbol{z_1},\boldsymbol{z_2})} &\overset{\text{def}}{=} \{(\boldsymbol{X_1}, \boldsymbol{X_2}) \in \mathbb{F}^{2k} \mid f(\boldsymbol{X_1}) = \boldsymbol{z_1},\ f(\boldsymbol{X_2}) = \boldsymbol{z_2}\} \\ &\overset{X_1,X_2 \text{ indep.}}{=} \{\boldsymbol{X_1} \in \mathbb{F}^\iota \mid f(\boldsymbol{X_1}) = \boldsymbol{z_1}\} \times \{\boldsymbol{X_2} \in \mathbb{F}^\iota \mid f(\boldsymbol{X_2}) = \boldsymbol{z_2}\} \\ &\overset{\text{def}}{=} X_{(\boldsymbol{z_1})} \times X_{(\boldsymbol{z_2})}. \end{aligned}$$

□

Not surprisingly, the situation is different for combiners with memory as the following example shows:

Example 4.8. *Consider Example 4.5 with the summation generator over $\mathbb{F}_2$ with $\iota = 2$ inputs. From Table 4.1.1, one can derive all preimages of the outputs 0 and 1. For example, it holds that*

$$\begin{aligned} &f(0,(0,0)) = 0, \quad f(1,(0,1)) = 0, \quad f(1,(1,0)) = 0, \quad f(0,(1,1)) = 0, \\ &f(1,(0,0)) = 1, \quad f(0,(0,1)) = 1, \quad f(0,(1,0)) = 1, \quad f(1,(1,1)) = 1. \end{aligned}$$

This means that for any output $z \in \mathbb{F}_2$ and any input $X \in \mathbb{F}_2^2$ exists at least one memory state Q such that $f(Q, X) = z$. Therefore, it holds that $X_{(\boldsymbol{0})} = X_{(\boldsymbol{1})} = \mathbb{F}_2^2$. On the other hand, it is displayed in (4.1.6) that $X_{\boldsymbol{Z}} \subset \mathbb{F}_2^4$ for all $\boldsymbol{Z} \in \mathbb{F}_2^2$.

Hence, equation (4.1.8) does not hold for combiners with memory. One can only say that $X_{\boldsymbol{Z}||\boldsymbol{Z'}} \subset X_{\boldsymbol{Z}} \times X_{\boldsymbol{Z'}}$. Nonetheless, it is possible to put $X_{\boldsymbol{Z}}$

[5]This is for example the case if r is smaller than the length of the shortest LFSR involved and if at each clock only the current outputs from the LFSRs are used to compute the keystream.

together from smaller sets. For this purpose, we define the following two sets:

$$\begin{aligned} X_{\boldsymbol{Q},\boldsymbol{Z}} &:= \{(X_1,\ldots,X_r) \mid f_\Psi(\boldsymbol{Q},X_1,\ldots,X_r) = \boldsymbol{Z}\} \\ X_{\boldsymbol{Z},\boldsymbol{Q}} &:= \{(X_1,\ldots,X_r) \mid \exists Q' : f_\Psi(Q',X_1,\ldots,X_r) = \boldsymbol{Z}, \Psi^r(Q',X_1,\ldots,X_r) = Q\} \end{aligned}$$

$X_{\boldsymbol{Q},\boldsymbol{Z}}$ contains all inputs which produce the output $\boldsymbol{Z}$ if the state of the memory is equal to Q *at the beginning*. In contrary, $X_{\boldsymbol{Z},\boldsymbol{Q}}$ contains all inputs which produce the output $\boldsymbol{Z}$ if the state of the memory is equal to Q *at the end*. Then, one easily sees that

$$X_{\boldsymbol{Z}||\boldsymbol{Z}'} = \bigcup_Q X_{\boldsymbol{Z},Q} \times X_{Q,\boldsymbol{Z}'}. \tag{4.1.9}$$

This corresponds to the expression for simple combiners if one sets $X_{Q,\boldsymbol{Z}} := X_{\boldsymbol{Z},Q'} := X_{\boldsymbol{Z}}$.

4.1.4 Annihilators

As it has been pointed out at the beginning of this section, $\boldsymbol{Z}$-functions are exactly the annihilators of the sets $X_{\boldsymbol{Z}}$. In the following, we will present several statements on annihilators.

Definition 4.9. *For a function $f : \mathbb{F}^n \to \mathbb{F}$, its **support** and its **kernel** are defined by*

$$\mathrm{supp}(f) := \{\boldsymbol{X} \in \mathbb{F}^n : f(\boldsymbol{X}) \neq 0\} \quad \textit{and} \quad \ker(f) := \{\boldsymbol{X} \in \mathbb{F}^n : f(\boldsymbol{X}) = 0\}.$$

It holds that $\mathrm{supp}(f) \cup \ker(f) = \mathbb{F}^n$.

Example 4.10. *Let $f(x_1,x_2) \in \mathbb{F}_2[x_1,x_2]$ defined by $f(x_1,x_2) := x_1 + x_2$. Then, it is $f(0,0) = f(1,1) = 0$ and $f(0,1) = f(1,0) = 1$. Thus, $\ker(f) = \{(0,0),(1,1)\}$ and $\mathrm{supp}(f) = \{(0,1),(1,0)\}$.*

Lemma 4.11. *Let $f,g : \mathbb{F}^n \to \mathbb{F}$ be arbitrary. Then g is a multiple of f if and only if $supp(g) \subseteq supp(f)$.*

Proof. Let g be a multiple of f, i.e., a function h exists such that $g(\boldsymbol{X}) = f(\boldsymbol{X}) \cdot h(\boldsymbol{X})$ for all $\boldsymbol{X} \in \mathbb{F}^n$. Choose $\boldsymbol{X} \in supp(g)$. Then, $g(\boldsymbol{X}) \neq 0$ implies that $f(\boldsymbol{X}) \neq 0$ and thus $\boldsymbol{X} \in supp(f)$. This shows that $supp(g) \subseteq supp(f)$.

Now assume that $supp(g) \subseteq supp(f)$. We define a function h as follows:

$$h(\boldsymbol{X}) := \begin{cases} 0, & \boldsymbol{X} \notin supp(g) \\ g(\boldsymbol{X}) \cdot (f(\boldsymbol{X}))^{-1}, & \boldsymbol{X} \in supp(g) \end{cases}$$

Observe that h is well defined as $supp(g) \subseteq supp(f)$ implies that $f(\boldsymbol{X}) \neq 0$ for $\boldsymbol{X} \in supp(g)$. Now, one can easily check that $g = f \cdot h$, which concludes the proof. □

Definition 4.12. *Let $S \subseteq \mathbb{F}^n$. We define the* characteristic function $\delta_S : \mathbb{F}^n \to \mathbb{F}$ *of S by*

$$\delta_S(\boldsymbol{X}) := \begin{cases} 1 & , \boldsymbol{X} \in S \\ 0 & , \text{else} \end{cases} \tag{4.1.10}$$

The following proposition describes the connection between both annihilators of a function and annihilators of a set:

Proposition 4.13. *Let $f : \mathbb{F}^n \to \mathbb{F}$ and $S \subseteq \mathbb{F}^n$. It holds that*

$$\mathrm{ann}(f) = \mathrm{ann}(\mathrm{supp}(f)) \quad \text{and} \quad \mathrm{ann}(S) = \mathrm{ann}(\delta_S). \tag{4.1.11}$$

Proof. Let $f : \mathbb{F}^n \to \mathbb{F}$ be an arbitrary function. Then it holds for any function $g : \mathbb{F}^n \to \mathbb{F}$:

$$\begin{aligned} g \in \mathrm{ann}(f) &\Leftrightarrow f \cdot g \equiv 0 \\ &\Leftrightarrow \forall \boldsymbol{X} \in \mathbb{F}^n : f(\boldsymbol{X}) \cdot g(\boldsymbol{X}) = 0 \\ &\Leftrightarrow \forall \boldsymbol{X} \in \mathbb{F}^n : [f(\boldsymbol{X}) \neq 0 \Rightarrow g(\boldsymbol{X}) = 0] \\ &\Leftrightarrow \forall \boldsymbol{X} \in \mathbb{F}^n : [\boldsymbol{X} \in \mathrm{supp}(f) \Rightarrow g(\boldsymbol{X}) = 0] \\ &\Leftrightarrow g \in \mathrm{ann}(\mathrm{supp}(f)). \end{aligned}$$

The second claim follows from the fact that $\mathrm{supp}(\delta_S) = S$. □

Theorem 4.14. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$. Every annihilator of f is a multiple of $\delta_{\ker(f)}$.*

Proof. Let g be an annihilator of f. Then, $\mathrm{supp}(g) \subseteq \ker(f)$. Otherwise, the existence of an $\boldsymbol{X} \in \mathrm{supp}(g) \cap \mathrm{supp}(f)$ would imply $g(\boldsymbol{X}) \cdot f(\boldsymbol{X}) \neq 0$, which is a contradiction to the assumption that g is an annihilator of f. Lemma 4.11 and the fact that $\mathrm{supp}(\delta_{\ker(f)}) = \ker(f)$ concludes the proof. □

Corollary 4.15. *Let $f \in \mathbb{F}_q[x_1, \ldots, x_n]$. $\mathrm{ann}(f)$ is a principal ideal in $\mathbb{F}_q[x_1, \ldots, x_n]$ and is generated by $1 - f^{q-1}$.*

Proof. In any finite field $\mathbb{F}_q$ the field equation $\boldsymbol{x}^q = \boldsymbol{x}$ is true. This show in particular that $\boldsymbol{x}^{q-1} = 1$ for $\boldsymbol{x} \neq 0$. It follows that

$$f^{q-1}(\boldsymbol{X}) = \begin{cases} 1 & , \boldsymbol{X} \in \mathrm{supp}(f) \\ 0 & , \text{else} \end{cases} .$$

Hence $f^{q-1} = \delta_{Supp(f)}$ and $1 - f^{q-1} = \delta_{ker(f)}$. Finally, by Theorem 4.14, any annihilator is a multiple of $\delta_{ker(f)} = 1 - f^{q-1}$. □

Corollary 4.16. *For a (ι, m)-combiner and $\boldsymbol{Z} \in \mathbb{F}^r$, the set of $\boldsymbol{Z}$-functions is the ideal $\langle 1 - \delta_{X_{\boldsymbol{Z}}} \rangle$.*

Proof. By Theorem 4.3, the set of $\boldsymbol{Z}$-functions is equal to $\mathrm{ann}(X_{\boldsymbol{Z}})$. By Proposition 4.13, it holds that $\mathrm{ann}(X_{\boldsymbol{Z}}) = \mathrm{ann}(\delta_{X_{\boldsymbol{Z}}})$. Corollary 4.15 yields that the set of $\boldsymbol{Z}$-functions is the ideal $\left\langle 1 - \delta_{X_{\boldsymbol{Z}}}^{q-1} \right\rangle$.

Hence, we only have to show that $\delta_{X_{\boldsymbol{Z}}}^{q-1} = \delta_{X_{\boldsymbol{Z}}}$. As the images of $\delta_{X_{\boldsymbol{Z}}}$ are in $\{0, 1\}$ by definition and because of $0^{q-1} = 0$ and $1^{q-1} = 1$, it holds that $\delta_{X_{\boldsymbol{Z}}}^{q-1}(\boldsymbol{X}) = \delta_{X_{\boldsymbol{Z}}}(\boldsymbol{X})$ for all $\boldsymbol{X}$. Thus, $\delta_{X_{\boldsymbol{Z}}}^{q-1} = \delta_{X_{\boldsymbol{Z}}}$. □

4.1.5 On Z-functions and r-functions

In this section, we combine the results from the previous sections to characterize $\boldsymbol{Z}$-functions and r-functions for a fixed (ι, m)-combiner. First, we recall the difference between these two notions. Let as usual $z_t \in \mathbb{F}$ denote the keystream element produced by clock t and K_t and Q_t the corresponding inputs coming from the LFSRs' initial states and the memory, respectively. That is, it holds that $f(Q_t, K_t) = z_t$ for all clocks t.

Then, an r-function is a function $F : \mathbb{F}^{\iota \cdot r + r} \to \mathbb{F}$ such that

$$F(K_t, \ldots, K_{t+r-1}, z_t, \ldots, z_{t+r-1}) = 0$$

for all t. In some cases, as for example for fast algebraic attacks (see Chapter 5), it is preferable to set up the system of equations by r-functions.

On the other hand, we briefly sketched in Section 3.2.3 some situations where $\boldsymbol{Z}$-functions are the better choice. Informally, one could say that a $\boldsymbol{Z}$-function is an r-function where the input $z_t, \ldots, z_{t+r-1}$ is specified. It must hold for a $\boldsymbol{Z}$-function $F_{\boldsymbol{Z}}$ that $F_{\boldsymbol{Z}}(K_t, \ldots, K_{t+r-1}) = 0$ whenever $(\boldsymbol{z_t, \ldots, z_{t+r-1}}) = \boldsymbol{Z}$.

The following theorem connects both notions:

Theorem 4.17. *Let a (ι, m)-combiner be fixed. A function $F(X_1, \ldots, X_r, z_1, \ldots, z_r) : \mathbb{F}^{\iota \cdot r + r} \to \mathbb{F}$ is an r-function if and only if it has the form*

$$F(X_1, \ldots, X_r, z_1, \ldots, z_r) = \sum_{\boldsymbol{Z} \in \mathbb{F}^r} \delta_{\boldsymbol{Z}}(z_1, \ldots, z_r) \cdot F_{\boldsymbol{Z}}(X_1, \ldots, X_r) \tag{4.1.12}$$

where $F_{\boldsymbol{Z}}$ denotes a $\boldsymbol{Z}$-function and $\delta_{\boldsymbol{Z}}(\boldsymbol{z_1, \ldots, z_r})$ is defined to be equal to 1 if $\boldsymbol{Z} = (\boldsymbol{z_1, \ldots, z_r})$, and equal to zero otherwise.

Proof. That the right hand side of (4.1.12) gives an r-function is easy to see. Hence, we only need to show that any r-function, i.e. the left-hand side, can be written in the form of the right-hand side.

First, any function $F(X_1, \ldots, X_r, z_1, \ldots, z_r)$ can be expressed by

$$F(X_1, \ldots, X_r, z_1, \ldots, z_r) = \sum_{\boldsymbol{Z} \in \mathbb{F}^r} \delta_{\boldsymbol{Z}}(z_1, \ldots, z_r) \cdot F'_{\boldsymbol{Z}}(X_1, \ldots, X_r)$$

for some arbitrary functions $F'_{\boldsymbol{Z}}$. By the definitions of r-functions and $\boldsymbol{Z}$-functions, F is an r-function if and only if $F(X_1, \ldots, X_r, \boldsymbol{z_1}, \ldots, \boldsymbol{z_r})$ is a $(\boldsymbol{z_1}, \ldots, \boldsymbol{z_r})$-function. Because of $F(X_1, \ldots, X_r, \boldsymbol{Z}) = F'_{\boldsymbol{Z}}(X_1, \ldots, X_r)$, each $F'_{\boldsymbol{Z}}$ must be a $\boldsymbol{Z}$-function. That concludes the proof. □

Now, Corollary 4.16 and Theorem 4.17 immediately imply

Corollary 4.18. *Let a (ι, m)-combiner and $r \geq 1$ be fixed. Then, any r-function has the form*

$$F(X_1, \ldots, X_r, z_1, \ldots, z_r) = \sum_{\boldsymbol{Z} \in \mathbb{F}^r} \delta_{\boldsymbol{Z}}(z_1, \ldots, z_r) \cdot g_{\boldsymbol{Z}}(X_1, \ldots, X_r) \cdot (1 - \delta_{X_{\boldsymbol{Z}}}(X_1, \ldots, X_r)).$$

Corollary 4.19. *For a simple combiner, i.e. $m = 0$, over $\mathbb{F} = \mathbb{F}_2$, any 1-function $F(X, z)$ can be written as*

$$\begin{aligned} F(X, z) &= (z \oplus 1) \cdot h_0(X) \cdot f(X) + z \cdot h_1(X) \cdot (f(X) \oplus 1) && (4.1.13) \\ &= z \cdot (f \cdot (h_0 + h_1) + h_1) + h_0 \cdot f && (4.1.14) \end{aligned}$$

with $h_0(X)$ and $h_1(X)$ being arbitrary Boolean functions.

Proof. Let f be the output function of the simple combiner. Because of $X_{(\boldsymbol{z})} = f^{-1}(\boldsymbol{z})$, it holds that

$$\delta_{X_{(\boldsymbol{0})}}(X) = f(X) \oplus 1 \quad \text{and} \quad \delta_{X_{(\boldsymbol{1})}}(X) = f(X).$$

The rest follows by Corollary 4.18. □

To the best of our knowledge, all algebraic attacks on (ι, m)-combiners published so far use systems of equations build from r-functions or $\boldsymbol{Z}$-functions. Of course, for a successful attack, the equations should be such that the solving step at the end is supported as much as possible. In the case of linearization, which is the only method so far that allows an effort estimation in advance (see also Section 3.3), this would mean that the degree is as low as possible. Therefor, we introduce the notion of the lowest annihilator degree.

Definition 4.20. *Let $S \subset \mathbb{F}^n$ for a finite field $\mathbb{F}$ and $n \geq 1$ an integer. We define its **lowest annihilator degree** (*lad*) by*

$$\mathrm{lad}(S) := \min\{\deg(g) | g \not\equiv 0, g \in \mathrm{ann}(S)\} \qquad (4.1.15)$$

If $\mathrm{ann}(S)$ *consists only of the all-zero-function, we define* $\mathrm{lad}(S) := \infty$*. We say that* S *is* d-***immune*** *if* $\mathrm{lad}(S) > d$*.*

Consequently, we define the lad *of a function* $f : \mathbb{F}^n \to \mathbb{F}$ *by* $\mathrm{lad}(f) := \mathrm{lad}(\mathrm{supp}(f))$ *and call the function* d*-immune if it has no annihilators of degree* $\leq d$*.*

For an attack on a (ι, m)-combiner using $\boldsymbol{Z}$-functions, the lowest annihilator degree gives a lower bound on the degree of the equations. More precisely, any $\boldsymbol{Z}$-function has a degree of at least $\mathrm{lad}(X_{\boldsymbol{Z}})$.

Definition 4.21. *Let a* (ι, m)*-combiner be fixed and* $r \geq 1$*. Then, we define*

$$\mathrm{lad}(r) := \min_{\boldsymbol{Z} \in \mathbb{F}^r} \{\mathrm{lad}(X_{\boldsymbol{Z}})\}. \tag{4.1.16}$$

Thus, $\mathrm{lad}(r)$ tells the lower bound of the degree of equations involving r clocks, which in a certain sense gives a feeling of the resistance against algebraic attacks with linearization. As any r-function can be equally interpreted as an $r + \epsilon$-function, it necessarily holds that $\mathrm{lad}(r + \epsilon) \leq \mathrm{lad}(r)$.

In fact, the values $\mathrm{lad}(r)$ might decrease with increasing r. Any cryptographically reasonable (ι, m)-combiner should be designed such that two different keys lead to different keystreams. Thus, if enough keystream elements are known, the secret key K is uniquely defined. So, if a fixed keystream $\boldsymbol{Z} \in \mathbb{F}^{r^*}$ can be produced by only one key K, then $X_{\boldsymbol{Z}} = \{K\}$, implying many linear annihilators. In this case, it holds that $\mathrm{lad}(r^*) = 1$, independent of the values of $\mathrm{lad}(r)$ for $r \leq r^*$.

As an example, we can consider the lowest annihilator degree in the context of E_0. In [ArmK03], we stated that $\mathrm{lad}(r) = 4$ for $r = 4 \ldots 6$. On the other hand, in [Cou03, Arm04a] it was showed that $\mathrm{lad}(8,882,188) \leq 3$. In fact, as the initial state of the whole E_0 keystream generator is described by 132 bits, the knowledge of more than 8 million keystream bits should specify the initial setting uniquely. That is, it presumably holds $\mathrm{lad}(8,882,188) = 1$, although no linear annihilators are known so far.[6]

Finally, we want to point out that the notion of the lowest annihilator degree is an extension of the notion of algebraic immunity. The reason is that for a simple combiner with output function f, it holds that

$$AI(f) = \mathrm{lad}(1).$$

If the conditions from Proposition 4.7 are met, i.e. $X_{\boldsymbol{Z}} = X_{\boldsymbol{z}_1} \times \ldots \times X_{\boldsymbol{z}_r}$, then it follows that $\mathrm{lad}(r) = \mathrm{lad}(1)$.

[6] The knowledge of such annihilators would imply a practical break of E_0 if that many keystream bits are known.

In some cases, it might be difficult to find the value of $\mathrm{lad}(S)$ for a given set S. In Section 4.2, we will discuss several algorithms to compute for a given set S annihilators with the minimum degree, i.e. $\mathrm{lad}(S)$. However, in the case of $\mathbb{F} = \mathbb{F}_2$, the size of S already implicates some bounds on $\mathrm{lad}(S)$, as we will show now.[7]

Theorem 4.22. *For any non-zero Boolean function f of degree $\leq d$, it holds that*

$$2^{n-d} \leq |\ker(f)| \leq (2^d - 1) \cdot 2^{n-d}.$$

Both bounds can be achieved.

Proof. We prove the theorem by induction over n for a fixed degree d. We will show only the upper bound. The lower bound follows from

$$|\ker(f)| = 2^n - |\ker(\underbrace{f \oplus 1}_{\deg=d})| \geq 2^n - \left(2^d - 1\right) \cdot 2^{n-d} = 2^{n-d}.$$

If $n = d$, f has the form $f = x_1 \cdot \ldots \cdot x_d$ and thus $|\ker(f)| = |\mathbb{F}_2^d \setminus \{(1, \ldots, 1)\}| = 2^d - 1 = (2^d - 1) \cdot 2^{n-d}$. Suppose now that the proposition is true for some $n \geq d$. I.e., $|\ker(f)| \leq (2^d - 1) \cdot 2^{n-d}$ for all non-zero $f \in \mathbb{F}_2[x_1, \ldots, x_n]$ of degree $\leq d$, and this bound is achieved for at least one f. Let $f \in \mathbb{F}_2[x_1, \ldots, x_{n+1}]$ be an arbitrary non-zero Boolean function of degree $\leq d$. Then, f can be written as follows

$$f(x_1, \ldots, x_{n+1}) = f'(x_1, \ldots, x_n) \oplus x_{n+1} \cdot f''(x_1, \ldots, x_n)$$

where $f', f'' \in \mathbb{F}_2[x_1, \ldots, x_n]$, $\deg(f') \leq d$, $\deg(f'') \leq d - 1$ and at least one of them is non-zero. We distinguish three cases:

$f' \neq 0$ and $f'' = 0$: Then $f \in \mathbb{F}_2[x_1, \ldots, x_n]$ and

$$|\ker(f)| \leq (2^d - 1)2^{n-d} \leq (2^d - 1)2^{n+1-d}$$

by assumption.

$f' = 0$ and $f'' \neq 0$: Then $\ker(f) = \{(x, 0)\} \dot\cup \{(x, 1) | x \in \ker(f'')\}$. Because of $\deg(f'') \leq d - 1$, it is

$$|\ker(f)| \leq 2^n + (2^{d-1} - 1) \cdot 2^{n-(d-1)} = (2^d - 1)2^{n-d}.$$

$f' \neq 0$ and $f'' \neq 0$: Then $\ker(f)$ can be expressed by

$$\ker(f) = \{(x, 0) | x \in \ker(f')\} \dot\cup \{(x, 1) | x \in \ker(f' \oplus f'')\}$$

[7] This Theorem has been proved independently in different papers, e.g. in [Arm04b].

If $f' \oplus f'' \equiv 0$ then $\deg f' = \deg f'' \leq d-1$ and

$$\begin{aligned} |\ker(f)| &= |\ker(f')| + |\ker(f' \oplus f'')| \\ &\leq (2^{d-1}-1) \cdot 2^{n-(d-1)} + 2^n = (2^d-1) \cdot 2^{n+1-d}. \end{aligned}$$

If $f' \oplus f'' \not\equiv 0$ then

$$\begin{aligned} |\ker(f)| &= |\ker(f')| + |\ker(f' \oplus f'')| \\ &\leq (2^d-1) \cdot 2^{n-d} + (2^d-1) \cdot 2^{n-d} = (2^d-1) \cdot 2^{n+1-d}. \end{aligned}$$

If we chose f' such that $|\ker(f')| = (2^d-1) \cdot 2^{n-d}$ and $f'' = 0$ then the bound $(2^d-1) \cdot 2^{n+1-d}$ is achieved by f. □

Theorem 4.22 can be used to exclude the existence of annihilators of degree $\leq d$:

Corollary 4.23. *Let $f \in \mathbb{F}_2[x_1, \ldots, x_n]$. If $|\operatorname{supp}(f)| > (2^d-1)2^{n-d}$ then $\operatorname{lad}(f) > d$.*

Proof. Let $g \in \operatorname{ann}(f)$. This implies that $\operatorname{supp}(f) \subseteq \ker(g)$ (see proof of 4.14) and therefore $|\operatorname{supp}(f)| \leq |\ker(g)|$. Hence, the assumption $\deg(g) \leq d$ would lead to the following contradiction:

$$(2^d-1)2^{n-d} < |\operatorname{supp}(f)| \leq |\ker(g)| \overset{Th.4.22}{\leq} (2^d-1)2^{n-d}.$$

□

Example 4.24. *In [ArmK03], Z-functions with $Z \in \mathbb{F}_2^4$ for the E_0 keystream generator were described. We've checked that $|X_Z| = |\operatorname{supp}(\delta_{X_Z})| = 53,248$ for all $Z \in \mathbb{F}_2^4$. Because of $|\operatorname{supp}(\delta_{X_Z})| = 53,248 > 49,152 = (2^2-1)2^{4 \cdot 4-2}$, it holds that $\operatorname{lad}(X_Z) \geq 3$.*

So, an attacker's task is not only to find Z-functions, but also to try that their degrees are as low as possible, preferably equal to $\operatorname{lad}(X_Z)$. From our point of view, this will require in most cases considering the sets X_Z to find Z-functions of degree $\operatorname{lad}(X_Z)$, although sometimes more direct methods may exist to construct Z-function. In the next sections, we will consider the questions how to find Z-functions with the minimum degree and how to design the (ι, m)-combiner to avoid them.

4.2 Finding low degree equations

After developing the theoretical background of $\boldsymbol{Z}$-functions, we turn our attention to the not less important question of how to actually find them. Although it is possible in some cases to derive $\boldsymbol{Z}$-functions directly from the specifications of the (ι, m)-combiner,[8] in the general case an algorithm is needed that efficiently computes a basis of annihilators with the minimum degree. In this section, we present several algorithms to do these computations.

4.2.1 Using Gröbner bases

Once again, Gröbner bases can be helpful (see also [ArsF03]). One possibility is to set up an ideal containing the expressions $f(Q_t, X_t) - z_t$, $\Psi(Q_t, X_t) - Q_{t+1}$, and the corresponding field equations. Then, compute the reduced Gröbner basis for a graded term order. With results from the intersection theory (see [CoxLS96]), one can derive a Gröbner basis which is independent of the memory entries Q_t. In other words, one uses the properties of Gröbner bases to find non-trivial equations in the inputs and keystream elements, which are independent of the memory. Of course, this is only possible if $\boldsymbol{Z}$-functions exist in this case.

Another approach is implied by Corollary 4.16, which says that any $\boldsymbol{Z}$-function is a multiple of $1 - \delta_{X_{\boldsymbol{Z}}}$. Thus, they all lie in the principal ideal generated by $1 - \delta_{X_{\boldsymbol{Z}}}$. Let I be the ideal generated by $1 - \delta_{X_{\boldsymbol{Z}}}$ and the field equations. Then, by Remark 3.18, computing a reduced Gröbner basis of I immediately reveals a basis of all annihilators with minimum degree.

4.2.2 A straightforward algorithm

As explained in the previous section, any $\boldsymbol{Z}$-function is an annihilator of the set $X_{\boldsymbol{Z}}$ and vice versa. Therefore, the task is to find annihilators for a given set of preferably low degree. One approach is to solve a system of linear equations. For this purpose, we generalize this problem as follows:

Given a set $S \subseteq \mathbb{F}^n$ and a set of linearly independent functions $\mathcal{F} \subset \mathbb{F}[x_1, \ldots, x_n]$, does a non-zero linear combination $f := \sum_{\hat{f} \in \mathcal{F}} c_{\hat{f}} \cdot \hat{f}$ of functions in $\mathcal{F}$ and $c_{\hat{f}} \in \mathbb{F}$ exist such that $f(\boldsymbol{X}) = 0$ for all $\boldsymbol{X} \in S$?

Observe that if $\mathcal{F}$ is the set of all monomials in $\mathbb{F}[x_1, \ldots, x_n]$ of degree $\leq d$, then this describes the case where one is interested in the existence of

[8] More on this at the end of this section.

annihilators of degree $\leq d$ for a given set S.

Definition 4.25. *Let $n \geq 1$, $S \subseteq \mathbb{F}^n$, and $\mathcal{F} \subset \mathbb{F}[x_1, \ldots, x_n]$ be a set of linearly independent functions. We define the matrix $M_{\mathcal{F}}(S)$ of size $|S| \times |\mathcal{F}|$ as follows. The columns are indexed by $\hat{f} \in \mathcal{F}$ and the rows by $\boldsymbol{X} \in S$. The entry in column $\hat{f}$ and row $\boldsymbol{X}$ is set to $\hat{f}(\boldsymbol{X})$. A schematic figure of $M_{\mathcal{F}}(S)$ is displayed in (4.2.17).*

$$\begin{array}{c} \\ \vdots \\ \boldsymbol{X} \in S \\ \vdots \end{array} \begin{array}{c} \ldots \hat{f} \in \mathcal{F} \ldots \\ \begin{pmatrix} \vdots \\ \ldots \quad \hat{f}(\boldsymbol{X}) \quad \ldots \\ \vdots \end{pmatrix} \end{array} \tag{4.2.17}$$

If $\mathcal{F}$ is exactly the set of all monomials of degree d or less, we abbreviate $M_{\mathcal{F}}(S)$ to $M_d(S)$.

Example 4.26. *Let $\mathbb{F} = \mathbb{F}_2$ and $S = \{(000), (011), (100), (110)\} \subset \mathbb{F}_2^3$ and $\mathcal{F} := \{1, x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3\}$ be the set of all monomials in x_1, x_2, and x_3 of degree ≤ 2. Then, $M_{\mathcal{F}}(S) = M_2(S)$ has the form*

$$\begin{array}{c} \\ (000) \\ (011) \\ (100) \\ (110) \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & x_1 & x_2 & x_3 & x_1x_2 & x_1x_3 & x_2x_3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \tag{4.2.18}$$

Theorem 4.27. *Let $n \geq 1$, $S \subseteq \mathbb{F}^n$, and $\mathcal{F} \subset \mathbb{F}[x_1, \ldots, x_n]$ be a set of linearly independent functions. A non-zero function $f := \sum_{\hat{f} \in \mathcal{F}} c_{\hat{f}} \cdot \hat{f}$ with $f(\boldsymbol{X}) = 0$ for all $\boldsymbol{X} \in S$ exists if and only if $M_{\mathcal{F}}(S)$ has not **full column rank**. By full column rank, we mean that the rank of the matrix equals the number of columns.*

Proof. We fix arbitrary orderings on S and $\mathcal{F}$. Let $C := (c_{\hat{f}})_{\hat{f} \in \mathcal{F}}$ be an arbitrary vector in $\mathbb{F}^{|\mathcal{F}|}$ and $f(X) := \sum_{\hat{f} \in \mathcal{F}} c_{\hat{f}} \hat{f}$. Set $V := M_{\mathcal{F}}(S) \cdot C$. V is a vector over $\mathbb{F}$ of size $|S|$, indicated by the elements in S. By the definition of $M_{\mathcal{F}}(S)$, the entry at position $\boldsymbol{X} \in S$ is equal to $\sum_{\hat{f} \in \mathcal{F}} c_{\hat{f}} \cdot \hat{f}(\boldsymbol{X}) = f(\boldsymbol{X})$. Thus, $f(\boldsymbol{X}) = 0$ for all $\boldsymbol{X} \in S$ if and only if $V = \vec{0}$. But this is only possible if the kernel of $M_{\mathcal{F}}(S)$ is non-trivial or rather the matrix has not full column rank. □

Example 4.28. *Consider again Example 4.26. The matrix $M_2(S)$, displayed in (4.2.18), has 7 columns but only a rank of 4. Therefore, the vector space of vectors C such that $M_2(S) \cdot C = \vec{0}$ has the dimension 3. A possible basis is*

$C_1 = (0,0,1,0,1,0,1)$, $C_2 = (0,0,0,1,0,0,1)$, and $C_3 = (0,0,0,0,0,1,0)$. This can be translated into three different annihilators for S of degree ≤ 2:

$$\begin{aligned}
C_1: &\quad 0\cdot 1 + 0\cdot x_1 + 1\cdot x_2 + 0\cdot x_3 + 1\cdot x_1x_2 + 0\cdot x_1x_3 + 1\cdot x_2x_3 &&= x_2 + x_1x_2 + x_2x_3\\
C_2: &\quad 0\cdot 1 + 0\cdot x_1 + 0\cdot x_2 + 1\cdot x_3 + 0\cdot x_1x_2 + 0\cdot x_1x_3 + 1\cdot x_2x_3 &&= x_3 + x_2x_3\\
C_3: &\quad 0\cdot 1 + 0\cdot x_1 + 0\cdot x_2 + 0\cdot x_3 + 0\cdot x_1x_2 + 1\cdot x_1x_3 + 0\cdot x_2x_3 &&= x_1x_3
\end{aligned}$$

Definition 4.29. *Let $n \geq 1$, $S \subseteq \mathbb{F}^n$ and $\mathcal{F} \subset \mathbb{F}[x_1, \ldots, x_n]$. If $M_{\mathcal{F}}(S)$ has full rank, we say that S is $\mathcal{F}$-**immune**. If $\mathcal{F}$ consists of all monomials with degree d or less, then we simply say that S is d-immune.*[9]

*Furthermore, we name the set of functions $f := \sum_{\hat{f}\in\mathcal{F}} c_{\hat{f}}\hat{f}$ with $f|_S \equiv 0$ the $\mathcal{F}-\text{kernel}$ resp. d-**kernel** of S. Obviously, S is $\mathcal{F}$-immune (resp. d-immune) if and only if its $\mathcal{F}$-kernel (resp. d-kernel) consists of only the all-zero function.*

The following is easy to show with basic linear algebra (see also Example 4.28):

Corollary 4.30. *Let $n \geq 1$, $S \subseteq \mathbb{F}^n$ and $\mathcal{F} \subset \mathbb{F}[x_1, \ldots, x_n]$. If $|S| < |\mathcal{F}|$, then S is not $\mathcal{F}$-immune.*

Proof. In the case of $|S| < |\mathcal{F}|$, the number of columns in $M_{\mathcal{F}}(S)$ is bigger than the number of rows. Thus, the columns cannot be linearly independent, proving the existence of a non-trivial kernel. □

Theorem 4.27 implies Algorithm 4.2 which computes the $\mathcal{F}$-kernel of a given set S. This approach has been described in [ArmK03, MeiPC04].

[9] This coincides with the definition of d-immune given in Definition 4.20.

Algorithm 4.2.
Basis of $\mathcal{F}$-kernel of S
Input: *$n \geq 1$, $S \subseteq \mathbb{F}^n$ and $\mathcal{F} \subset \mathbb{F}[x_1, \ldots, x_n]$*
Output: *A basis of the $\mathcal{F}$-kernel*

1: *Choose an arbitrary ordering of the elements in S and $\mathcal{F}$, respectively.*
2: *Create an all-zero matrix $M := M_{\mathcal{F}}(S)$ of size $|S| \times |\mathcal{F}|$ where the rows and columns are indicated by the elements in S and $\mathcal{F}$, respectively.*
3: ***for*** $\boldsymbol{X} \in S$ ***do***
4: ***for*** $\hat{f} \in \mathcal{F}$ ***do***
5: *Set $M_{\boldsymbol{X},\hat{f}} := \hat{f}(\boldsymbol{X})$*
6: ***end for***
7: ***end for***
8: *Compute a basis B for the kernel of $M_{\mathcal{F}}(S)$, that is vectors $C = (c_{\hat{f}})_{\hat{f}\in\mathcal{F}}$ such that $M_{\mathcal{F}}(S) \cdot C = \vec{0}$.*
9: ***return*** $\{\sum_{\hat{f}\in\mathcal{F}} c_{\hat{f}}\hat{f} \mid (c_{\hat{f}})_{\hat{f}\in\mathcal{F}} \in B\}$

The next example illustrates Algorithm 4.2.

Example 4.31. *Consider the following subset of $\mathbb{F}_2^4$:*

$$S := \{ \begin{array}{l} (1000), (0100), (0010), (0001), (1100), (1010), \\ (0101), (1110), (1101), (1011), (0111)\}. \end{array}$$

We are interested in if S has any annihilators of degree 2. The matrix $M_2(S)$ has the following form:

$$\begin{array}{c} \\ 1000 \\ 0100 \\ 0010 \\ 0001 \\ 1100 \\ 1010 \\ 0101 \\ 1110 \\ 1101 \\ 1011 \\ 0111 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} 1 & x_1 & x_2 & x_3 & x_4 & x_1x_2 & x_1x_3 & x_1x_4 & x_2x_3 & x_2x_4 & x_3x_4 \end{array} \\ \left(\begin{array}{ccccccccccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{array} \right) \end{array} \tag{4.2.19}$$

$M_2(S)$ has the size 11×11 but the rank is only 10. Thus, the kernel has dimension 1 and is generated by the vector $(1,1,1,1,0,0,0,1,1,1,1,)^T$. This implies this quadratic annihilator for S:

$$\mathbf{1}\cdot 1 + \mathbf{1}\cdot x_1 + \mathbf{1}\cdot x_2 + \mathbf{1}\cdot x_3 + \mathbf{0}\cdot x_4 + \mathbf{0}\cdot x_1x_2 + \mathbf{0}\cdot x_1x_3 + \mathbf{1}\cdot x_1x_4 + \mathbf{1}\cdot x_2x_3 + \mathbf{1}\cdot x_2x_4 + \mathbf{1}\cdot x_3x_4$$
$$= 1 + x_1 + x_2 + x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4$$

In [MeiPC04], the authors described an improvement of Algorithm 4.2 for the case $\mathbb{F} = \mathbb{F}_2$. The idea is to consider the values in S with increasing weight to exclude certain monomials in the ANF of any potential annihilator as soon as possible. They estimated a speed up factor of 8 compared to Algorithm 4.2.

Next, we take a look on an alternative to Algorithm 4.2. The runtime behaviour of this algorithm is the same as for Algorithm 4.2. But as opposed to Algorithm 4.2, where the $\mathcal{F}$-kernel is computed all at once, the approach here is slightly different. Starting with the whole set $\mathcal{F}$, this set is reduced, if necessary, for each element $\boldsymbol{X} \in X_{\boldsymbol{Z}}$ step by step. For certain systems, it might be more efficient to store a list of functions $\mathcal{F}_i$ instead of the matrix $M_{\mathcal{F}}(S)$. At least, this was our observation while doing experiments with the computer algebra system `MAPLE`. Another reason why we present this algorithm is that it can be adapted for permutation invariant (ι, m)-combiners, providing an enormous speed-up in some cases.

Algorithm 4.3.
Basis of $\mathcal{F}$-kernel of S
Input: *$n \geq 1$, $S \subseteq \mathbb{F}^n$ and $\mathcal{F} \subset \mathbb{F}[x_1, \ldots, x_n]$ a set of linearly independent functions*
Output: *A basis of the $\mathcal{F}$-kernel*

1: *Let $S = \{\boldsymbol{X_1}, \ldots, \boldsymbol{X_s}\}$*
2: *Set $\mathcal{F}_0 := \mathcal{F}$*
3: ***for*** *$i = 1, \ldots, s$* ***do***
4: *Set $\mathcal{F}_i := \{f \in \mathcal{F}_{i-1} | f(\boldsymbol{X}_i) = 0\}$ and $\mathcal{F}_i^* := \mathcal{F}_{i-1} \setminus \mathcal{F}_i^0$*
5: *Choose $f^* \in \mathcal{F}_i^*$*
6: ***for*** *$g \in \mathcal{F}_i^* \setminus \{f^*\}$* ***do***
7: *Insert $f^*(X) - \frac{f^*(\boldsymbol{X_i})}{g(\boldsymbol{X_i})} \cdot g(X)$ into the set $\mathcal{F}_i$*
8: ***end for***
9: ***end for***
10: ***return*** *$\mathcal{F}_s$*

A detailed proof of correctness can be found in [Arm04b]. We sketch only the main arguments. Observe that the function defined in line 7 gives zero

on $\boldsymbol{X_i}$. Hence, after the i-th loop, $\mathcal{F}_i$ contains a set of functions that all cancel $\{\boldsymbol{X_1}, \ldots, \boldsymbol{X_i}\}$. These functions are linearly independent as they are linear combinations of the functions in $\mathcal{F}_{i-1}$ which are linearly independent by assumption. Thus, at the end $\mathcal{F}_s$ is a set of linearly independent functions that are zero on $\{\boldsymbol{X_1}, \ldots, \boldsymbol{X_s}\} = S$. Therefore it is a basis of the $\mathcal{F}$-kernel of S as claimed.

Also notice that if $\mathcal{F}_i^*$ contains only one single element, line 7 is not invoked in the i-th loop. For example, let $\mathcal{F} = \mathcal{F}_0$ be the set of monomials of degree $\leq d$ over $\mathbb{F}_2$, sorted by the degree, and $\boldsymbol{X_1} = \overrightarrow{0}$. Then any monomial is zero on $\boldsymbol{X_1}$ except the constant 1 and therefore $\mathcal{F}_1 = \mathcal{F}_0 \setminus \{1\}$. This is related to the principle used in [MeiPC04] to improve the algorithm 4.2. Actually, the ideas from there can likewise applied here to make Algorithm 4.3 faster.

Finally, we want to point out that the methods described in this section are only practical for small values of k and r. Otherwise, it is normally infeasible to compute the sets $X_{\boldsymbol{Z}}$, not to mention computing the kernel of the resulting matrix.

However, it can sometimes be possible to get $\boldsymbol{Z}$-functions by direct manipulation of given equations. As far as we know, the first example for such an approach has been given in [CouM03] for the simple combiner Toyocrypt, a submission to the Japanese government Cryptrec call for cryptographic primitives. The number of unknowns is $n = 128$ and the output function is the following function of degree 63

$$\begin{aligned} f(x_0, \ldots, x_{127}) &= x_{127} + \textstyle\sum_{i=0}^{62} x_i x_{\alpha_i} + x_{10}x_{23}x_{32}x_{42} + \\ &\quad x_1x_2x_9x_{12}x_{18}x_{20}x_{23}x_{25}x_{26}x_{28}x_{33}x_{38}x_{41}x_{51}x_{53}x_{59} \\ &\quad + \textstyle\prod_{i=0}^{62} x_i \end{aligned} \tag{4.2.20}$$

where $\{\alpha_0, \ldots, \alpha_{62}\}$ being some permutation of the set $\{63, \ldots, 125\}$. The degree of f is far too high to make f practical for an algebraic attack. Using the methods described in this section, this would require computing the sets $X_{(0)}$ and $X_{(1)}$ and looking for low degree annihilators. But this is infeasible because of the large number of inputs. In [CouM03], the authors made the observation that both $f \cdot (x_{23} - 1)$ and $f \cdot (x_{42} - 1)$ have a degree of 3. Thus, the $\boldsymbol{z}$-functions

$$\begin{aligned} F(x_0, \ldots, x_{127}, z) &= (f(X) - z) \cdot (x_{23} - 1) \\ F'(x_0, \ldots, x_{127}, z) &= (f(X) - z) \cdot (x_{42} - 1) \end{aligned}$$

can be used to mount an algebraic attack. This shows that in some cases, a direct analysis can lead further than the general approach.

Another example where $\boldsymbol{Z}$-functions could be found by direct analysis is the E_0 keystream generator (see [ArmK03]). Recall its definition from

Section 2.5.4. Obviously, the value of q_{t+1} depends only on K_t, q_t, p_t and p_{t-1} and the value of p_{t+1} on K_t, q_t, q_{t-1}, p_t and p_{t-1}. The calculations of q_{t+1} and p_{t+1} are done via the following equations (see [Arm02, Appendix A] for details):

$$q_{t+1} = \sigma_4(t) + \sigma_3(t)p_t + \sigma_2(t)q_t + \sigma_1(t)p_tq_t + q_t + p_{t-1} \quad (4.2.21)$$

$$p_{t+1} = \sigma_2(t) + \sigma_1(t)p_t + q_t + q_{t-1} + p_{t-1} + p_t \quad (4.2.22)$$

with $\sigma_s(t) := \sigma_s(K_t)$ being the s-th elementary symmetric function[10] in the unknowns K_t. If we define the following additional variables

$$\begin{aligned} a(t) &= \sigma_4(t) + \sigma_3(t)p_t + p_{t-1} \\ b(t) &= \sigma_2(t) + \sigma_1(t)p_t + 1, \end{aligned}$$

equations (4.2.21) and (4.2.22) can be rewritten to

$$q_{t+1} = a(t) + b(t)q_t \quad (4.2.23)$$

$$p_{t+1} = b(t) + 1 + p_{t-1} + p_t + q_t + q_{t-1}. \quad (4.2.24)$$

By multiplying (4.2.23) with $b(t)$ we get another equation

$$0 = b(t)(a(t) + q_t + q_{t+1}). \quad (4.2.25)$$

Equation (4.2.24) is equivalent to

$$q_t + q_{t-1} = b(t) + 1 + p_{t-1} + p_t + p_{t+1}. \quad (4.2.26)$$

Now we insert (4.2.26) into (4.2.25) with index $t+1$ instead of t and get

$$0 = b(t)\left(a(t) + b(t+1) + 1 + p_t + p_{t+1} + p_{t+2}\right).$$

Using (2.5.16), we eliminate all memory bits in the equation and get the following equation which holds for every clock t:

$$\begin{aligned} 0 = {} & z_{t-1} + z_t + z_{t+1} + z_{t+2} + \\ & \sigma_1(t) \cdot (z_{t-1} + z_{t+1} + z_{t+2} + z_tz_{t-1} + z_tz_{t+1} + z_tz_{t+2}) + \\ & \sigma_2(t) \cdot (z_{t-1} + z_t + z_{t+1} + z_{t+2}) + \sigma_4(t) + \\ & \sigma_1(t-1) + \sigma_1(t-1)\sigma_1(t)(1 + z_t) + \sigma_1(t-1)\sigma_2(t) + \\ & \sigma_1(t+1)z_{t+1} + \sigma_1(t+1)\sigma_1(t)z_{t+1}(z_t + 1) + \sigma_1(t+1)\sigma_2(t)z_{t+1} + \\ & \sigma_2(t+1) + \sigma_2(t+1)\sigma_1(t)(1 + z_t) + \sigma_2(t+1)\sigma_2(t) + \\ & \sigma_1(t+2) + \sigma_1(t+2)\sigma_1(t)(1 + z_t) + \sigma_1(t+2)\sigma_2(t). \end{aligned}$$

We have seen that several ways exist to find (low degree) Z-equations. In some cases, Gröbner bases methods or direct analysis can be tried, in particular if the general method is infeasible. However, the general method has the advantage that one knows for sure that no other Z-functions of a lower degree exist.

[10] A definition will be given later in Definition 4.37.

4.2.3 An algorithm adapted to permutation invariant (ι, m)-combiners

So far, we examined only general (ι, m)-combiners. But if the (ι, m)-combiner is permutation invariant, we showed in [Arm04b] that this property can be exploited to accelerate the computation. Recall that permutation invariant means that the output of these functions depends only on the hamming weight of the inputs and is therefore invariant under permutations of the inputs. This is equivalent to that both $f(\boldsymbol{Q}, X)$ and $\Psi(\boldsymbol{Q}, X)$ are symmetric for any $\boldsymbol{Q} \in \mathbb{F}^m$. Symmetric Boolean functions play an important role in practice as they can be efficiently implemented in hardware (e.g., see [Weg87]). Examples for permutation invariant (ι, m)-combiners are E_0 and the summation generator.

The goal of this section is to derive several statements from the context of permutation invariant (ι, m)-combiners. This will lead finally to Algorithm 4.4 from [Arm04b]. It exploits the special structure to avoid some unnecessary operations.

Definition 4.32. *Let $\mathbf{\Pi}_n$ be the group of permutations on $\{1, \ldots, n\}$. For $X = (x_1, \ldots, x_n)$ and $\pi \in \mathbf{\Pi}_n$ we define*

$$\pi(X) := (x_{\pi(1)}, \ldots, x_{\pi(n)}) \quad \text{and} \quad [X] := \{\pi(X) \mid \pi \in \mathbf{\Pi}_n\}.$$

Further on, for $f \in \mathbb{F}[x_1, \ldots, x_n]$, we define

$$\pi(f) = \pi(f)(x_1, \ldots, x_n) := f(x_{\pi(1)}, \ldots, x_{\pi(n)}).$$

Let $\Pi \subseteq \mathbf{\Pi}_n$. We say that a set $S \subseteq \{0,1\}^n$ is Π-invariant if $\pi(X) \in S$ for all $\pi \in \Pi$ and all $X \in S$. Consequently, we say that a function f is Π-invariant if $\pi(f) = f$ for all $\pi \in \Pi$.

Proposition 4.33. *Let $\Pi \subseteq \mathbf{\Pi}_n$ and $\langle \Pi \rangle$ denote the subgroup of $\mathbf{\Pi}_n$ generated by the elements of Π. A set $S \subseteq \mathbb{F}^n$ is Π-invariant if and only if it is $\langle \Pi \rangle$-invariant.*

Proof. Because of $\Pi \subseteq \langle \Pi \rangle$, a $\langle \Pi \rangle$-invariant set is always Π-invariant also.

Let S denote a Π-invariant set. We have to show that for all $\pi, \tilde{\pi} \in \Pi$ it is $(\pi \circ \tilde{\pi})(S) \subseteq S$ and $\pi^{-1}(S) \subseteq S$. The first proposition is obvious because of $\pi(S) \subseteq S$ and $\tilde{\pi}(S) \subseteq S$ by assumption. The reason for the second proposition is that the set $\{\mathrm{id}(= \pi^0), \pi^1, \pi^2, \ldots\} \subseteq \mathbf{\Pi}_n$ is finite and therefore π^{-1} can be expressed by π^m for an appropriate m. □

Proposition 4.34. *Let $\Pi \subseteq \mathbf{\Pi}_n$, $f \in \mathbb{F}[x_1, \ldots, x_n]$ and $z' \in \mathbb{F}$ arbitrary. If $f^{-1}(z)$ is Π-invariant for all $z \in \mathbb{F} \setminus \{z'\}$, then $f^{-1}(z')$ is Π-invariant too, In particular, this shows that f is Π-invariant.*

Proof. Assume that $f^{-1}(\boldsymbol{z})$ is Π-invariant for all $\boldsymbol{z} \in \mathbb{F}$, $\boldsymbol{z} \neq \boldsymbol{z}'$, but not $f^{-1}(\boldsymbol{z}')$. Then, $\boldsymbol{X}' \in f^{-1}(\boldsymbol{z}')$ and $\pi \in \Pi$ exist with $\boldsymbol{X} := \pi(\boldsymbol{X}') \in f^{-1}(\boldsymbol{z})$ for $\boldsymbol{z} \neq \boldsymbol{z}'$. But this implies $\pi^{-1}(\boldsymbol{X}) \in f^{-1}(\boldsymbol{z}') \neq f^{-1}(\boldsymbol{z})$. Hence, $f^{-1}(\boldsymbol{z})$ is not $\langle\Pi\rangle$-invariant. By Proposition 4.33, this is equivalent to that $f^{-1}(\boldsymbol{z})$ is not Π-invariant what contradicts the assumption. □

Definition 4.35. *Let $n_1, \ldots, n_r \geq 1$ be arbitrary integers and $\Pi_i \subseteq \mathbf{\Pi}_{n_i}$. By $\Pi_1 \times \ldots \times \Pi_r$, we denote the set of all permutations π in $\mathbf{\Pi}_{n_1+\ldots+n_r}$ such that*

$$\pi(X_1, \ldots, X_r) = (\pi_1(X_1), \ldots, \pi_r(X_r))$$

with $\pi_i \in \Pi_i$ and $X_i = (x_{i,1}, \ldots, x_{i,n_i})$.

Proposition 4.36. *Consider a (ι, m)-combiner, $r \geq 1$ and $\Pi \subseteq \mathbf{\Pi}_\iota$ such that for all $\boldsymbol{X} \in \mathbb{F}^\iota$, $\boldsymbol{Q} \in \mathbb{F}^m$ and $\pi \in \Pi$, it holds that $f(\boldsymbol{Q}, \boldsymbol{X}) = f(\boldsymbol{Q}, \pi(\boldsymbol{X}))$ and $\Psi(\boldsymbol{Q}, \boldsymbol{X}) = \Psi(\boldsymbol{Q}, \pi(\boldsymbol{X}))$. Then $X_{\boldsymbol{Z}}$ is $\Pi^r = \Pi \times \ldots \times \Pi$ invariant for all $\boldsymbol{Z} \in \mathbb{F}^r$.*

Proof. Let $\pi_1, \ldots, \pi_r \in \Pi$. Because of the assumptions, it holds that

$$f_\Psi(Q, X_1, \ldots, X_r) = f_\Psi(Q, \pi(X_1), \ldots, \pi(X_r))$$

for all $Q \in \mathbb{F}^m$ and $X_1, \ldots, X_r \in \mathbb{F}^k$. The rest follows easily from the definition of $X_{\boldsymbol{Z}}$, the claim follows. □

The proposition was motivated by combiners with memory as the summation generator or the E_0 keystream generator. For $\Pi := \mathbf{\Pi}_\iota$ and r smaller than the shortest LFSR, the assumptions of Proposition 4.36 are fulfilled. Hence, we know that $X_{\boldsymbol{Z}}$ is $\mathbf{\Pi}_\iota^r$-invariant which shows that $\delta_{X_{\boldsymbol{Z}}}^{-1}(1)$ is $\mathbf{\Pi}_\iota^r$-invariant. As $\delta_{X_{\boldsymbol{Z}}}(\boldsymbol{X}) \in \{0, 1\}$ for all $\boldsymbol{X}$, Proposition 4.34 yields that $\delta_{X_{\boldsymbol{Z}}}$ is $\mathbf{\Pi}_\iota^r$-invariant.

In fact, this can be used to compute $\delta_{X_{\boldsymbol{Z}}}$ faster which by Corollary 4.16 might support the computation of low-degree $\boldsymbol{Z}$-functions in some cases. Therefor, we need to develop some more theory on symmetric functions.

Definition 4.37. *Let $n \geq 1$. For $s \in \{1, \ldots, n\}$ and $d \in \{1, \ldots, q-1\}$, we define the **elementary symmetric function** $\sigma_s^d(x_1, \ldots, x_n) \in \mathbb{F}_q[x_1, \ldots, x_n]$ by*

$$\sigma_s^d(x_1, \ldots, x_n) := \sum_{1 \leq i_1 < \ldots < i_s \leq n} x_{i_1}^d \cdot \ldots \cdot x_{i_\iota}^d. \tag{4.2.27}$$

For $\mathbb{F} = \mathbb{F}_2$, the exponent d is necessarily always equal to one, which is why we omit it and write $\sigma_s(X)$ instead.

Example 4.38. *For $n = 4$ and $q = 2$, the four elementary symmetric functions are*

$$\begin{aligned}
\sigma_1(x_1, x_2, x_3, x_4) &:= x_1 + x_2 + x_3 + x_4, \\
\sigma_2(x_1, x_2, x_3, x_4) &:= x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4, \\
\sigma_3(x_1, x_2, x_3, x_4) &:= x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4, \\
\sigma_4(x_1, x_2, x_3, x_4) &:= x_1x_2x_3x_4.
\end{aligned}$$

The importance of elementary symmetric functions is showed in the following theorem:

Theorem 4.39. *Let $f(X) \in \mathbb{F}_q[X]$ be a symmetric function in n indeterminates. Then, the algebraic normal form of f can be written as*

$$f(x) = \sum_{\substack{1 \le s \le n \\ 1 \le d \le q-1}} c_{s,d}\sigma_s^d(X) + c_0. \tag{4.2.28}$$

Proof. We prove the claim by a simple counting argument. By definition, it holds that $f(\boldsymbol{X}) = f(\pi(\boldsymbol{X}))$ for all $\boldsymbol{X} \in \mathbb{F}^n$ and $\pi \in \boldsymbol{\Pi}_n$. This implies that if $|\boldsymbol{X}| = |\boldsymbol{X}'|$, then $f(\boldsymbol{X}) = f(\boldsymbol{X}')$. Thus, any symmetric function is uniquely defined by the expressions $f(i) := f(\boldsymbol{X})$ for one $\boldsymbol{X} \in \mathbb{F}^n$ with $|\boldsymbol{X}| = i$. As $\boldsymbol{X} = (\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}) \in \mathbb{F}_q^n$, it holds that $0 \le |\boldsymbol{x_i}| \le q - 1$ for $1 \le i \le n$, implying $|\boldsymbol{X}| = \sum_{i=1}^n |\boldsymbol{x}_i| \in \{0, \ldots, n \cdot (q-1)\}$. Summing up, f has a unique expression by its truth table $T(f) = (f(0), \ldots, f(n \cdot (q-1))) \in \mathbb{F}_q^{1+n\cdot(q-1)}$. This shows that in total $q^{1+n\cdot(q-1)}$ different symmetric functions exist over $\mathbb{F}_q$.

Now take a look at the elementary symmetric functions. Obviously, they are symmetric. Therefore, any function of the form

$$\sum_{\substack{1 \le r \le n \\ 1 \le d \le q-1}} c_{r,d}\sigma_r^d(X) + c_0 \tag{4.2.29}$$

is symmetric too. As there exist $1+n\cdot(q-1)$ different choices for $(c_0, c_{1,1} \ldots, c_{n,q-1})$, there are altogether $q^{1+n\cdot(q-1)}$ different symmetric functions of the form (4.2.29). Hence, these functions must be exactly the $q^{1+n\cdot(q-1)}$ different symmetric functions. □

With this theorem, it is easy to derive the following corollary:

Corollary 4.40. *Any $\boldsymbol{\Pi}_\iota^r$-invariant function $f(X_1, \ldots, X_r) \in \mathbb{F}_q[X_1, \ldots, X_r]$ with $X_i = (x_{i,1}, \ldots, x_{i,\iota})$ has an algebraic normal form of the following form:*

$$f(X_1, \ldots, X_r) = \sum_{\sigma_{i_1}, \ldots, \sigma_{i_r}} c_{\sigma_{i_1}, \ldots, \sigma_{i_r}} \sigma_{i_1}(X_1) \cdots \sigma_{i_r}(X_r) \tag{4.2.30}$$

with $\sigma_{i_j}(X_j)$ being an elementary symmetric function in X_i or a constant.

In the case that $\delta_{X_{\boldsymbol{Z}}}$ is $\mathbf{\Pi}_\iota^r$-invariant, one can use expression (4.2.30) to efficiently compute $\delta_{X_{\boldsymbol{Z}}}$ from $X_{\boldsymbol{Z}}$, which might help as the set of $\boldsymbol{Z}$-functions is given by the ideal $I = \langle 1 - \delta_{X_{\boldsymbol{Z}}} \rangle$ (see Corollary 4.16).

An alternative improvement to compute the set of low degree $\boldsymbol{Z}$-functions in the case of $\mathbf{\Pi}_\iota^r$-invariant sets $X_{\boldsymbol{Z}}$ is given in Algorithm 4.4.

Algorithm 4.4.
Computing a basis of $\boldsymbol{Z}$-functions for permutation invariant (ι, m)-combiners
Input: *The set $X_{\boldsymbol{Z}}$ and $d \geq 1$*
Output: *A basis for all annihilators of $\boldsymbol{X} \in X_{\boldsymbol{Z}}$ with degree $\leq d$*

1: *Let $\mathcal{F}_0 \subseteq \mathbb{F}[x_1, \ldots, x_n]$ be the set of all monomials with degree $\leq d$*
2: *Divide $X_{\boldsymbol{Z}}$ into equivalence classes: $X_{\boldsymbol{Z}} = \{[\boldsymbol{X_1}], \ldots, [\boldsymbol{X_s}]\}$*
3: ***for*** *i from 1 to s* ***do***
4: *Test if $f(\boldsymbol{X_i}) = 0$ for all $f \in \mathcal{F}_{i-1}$*
5: ***if*** *yes* ***then***
6: *Set $\mathcal{F}_i := \mathcal{F}_{i-1}$*
7: ***else***
8: *Compute an $\mathcal{F}_{i-1}$-kernel of the set $[\boldsymbol{X_i}]$ with Algorithm 4.3, and assign the identifier $\mathcal{F}_i$ to it.*
9: ***end if***
10: ***end for***
11: ***return*** *The set $\mathcal{F}_s$*

Similar to Algorithm 4.3, the set $\mathcal{F}$ is reduced bit by bit for each element $\boldsymbol{X} \in X_{\boldsymbol{Z}}$. Step 2 is possible by Proposition 4.36 as $X_{\boldsymbol{Z}}$ is Π_ι^r-invariant. The improvements of Algorithm 4.4 can be found in lines 4-7. The general algorithm, Algorithm 4.3, requires the evaluations of $f(\boldsymbol{X})$ for all $f \in \mathcal{F}_{i-1}$ and $\boldsymbol{X} \in X_{\boldsymbol{Z}}$ which have not been considered so far. In Algorithm 4.4, it is possible to skip some superfluous computations. These rely on the Proposition 4.41 which says that

$$f(\boldsymbol{X_i}) = 0 \ \forall f \in \mathcal{F}_{i-1} \quad \Rightarrow \quad f(\boldsymbol{X}) = 0 \ \forall \boldsymbol{X} \in [\boldsymbol{X_i}], \forall f \in \mathcal{F}_{i-1}.$$

Thus, if $f(\boldsymbol{X_i}) = 0 \ \forall f \in \mathcal{F}_{i-1}$, then we don't need to consider $f(\boldsymbol{X})$ for the other values $\boldsymbol{X} \in [\boldsymbol{X_i}] \setminus \{\boldsymbol{X_i}\}$. Assume for example that $\mathcal{F}_i$ for $i < s$ is already a basis of the annihilators of degree $\leq d$. Then the algorithm notices that by only evaluating $f(\boldsymbol{X_j})$ for $f \in \mathcal{F}_i$ and $i < j \leq s$. This means that the bigger the sizes of the sets $[\boldsymbol{X_j}]$ for $i < j \leq s$, the more operations can be skipped. Consequently, we observed in our simulations that a good strategy is to

consider the sets $\boldsymbol{X_i}$ with increasing sizes. However, if no annihilators of degree $\leq d$ exist or if the choice of the representants $\boldsymbol{X_i}$ is inauspicious[11], the running time might be close to the unadapted Algorithm 4.3. Nonetheless, our observation was that in most cases, Algorithm 4.4 provided a significant speed-up.

What remains is to prove the correctness of this algorithm.

Proposition 4.41. *Let the identifiers be as in Algorithm 4.4 and $i \geq 0$. If $f(\boldsymbol{X_i}) = 0$ for all $f \in \mathcal{F}_{i-1}$ then $f(\boldsymbol{X}) = 0$ for all $f \in \mathcal{F}_{i-1}$ and all $\boldsymbol{X} \in [X_i]$.*

Proof. The proof relies on the fact that $\langle \mathcal{F}_{i-1} \rangle$ is Π-invariant what will be shown later. Assume that $f(\boldsymbol{X_i}) = 0$ for all $f \in \mathcal{F}_{i-1}$. Let $f \in \mathcal{F}_{i-1}$ and $\pi \in \Pi$ be arbitrary. We have to show that $f(\pi(\boldsymbol{X_i})) = 0$ also. As $\langle \mathcal{F}_{i-1} \rangle$ is Π-invariant $f_1, \ldots, f_r$ and $c_1, \ldots, c_r \in \mathbb{F}$ exist such that $\pi(f) = \sum_{j=1}^{r} c_j \cdot f_j(X)$. This implies

$$f(\pi(\boldsymbol{X_i})) = \pi(f)(\boldsymbol{X_i}) = \underbrace{c_1 f_1(\boldsymbol{X_i})}_{=0} + \ldots + \underbrace{c_r f_r(\boldsymbol{X_i})}_{=0} = 0.$$

What remains is to show that $\mathcal{F}_{i-1}$ is Π-invariant. We do this by induction. $\mathcal{F}_0$, the set of all monomials with degree $\leq d$, is certainly Π-invariant. Assume now that $\mathcal{F}_{j-1}$ is Π-invariant. After the j-th loop, $\mathcal{F}_j$ is either equal to $\mathcal{F}_{j-1}$ or is the result from Algorithm 4.3 with input ($\mathcal{F}_{j-1}$,$[X_j]$). In the first case, the proposition is trivial.

In the second case, $\mathcal{F}_j$ is a basis for the vector space of all $f \in \langle \mathcal{F}_{j-1} \rangle$ with $f(\boldsymbol{X}) = 0$ for all $\boldsymbol{X} \in [X_j]$. Let $f \in \langle \mathcal{F}_j \rangle$ and $\pi \in \Pi$ be arbitrary. Then $\pi(f)(\boldsymbol{X}) = f(\pi(\boldsymbol{X})) = 0$ for all $\boldsymbol{X} \in [X_j]$. Additionally it holds $\pi(f) \in \langle \mathcal{F}_{j-1} \rangle$ by assumption. Therefore, $\pi(f)$ is a function from $\langle \mathcal{F}_{j-1} \rangle$ which is zero on all $\boldsymbol{X} \in [X_j]$. Hence, $\pi(f) \in \langle \mathcal{F}_j \rangle$, showing that $\langle \mathcal{F}_j \rangle$ is Π-invariant. □

Remark 4.42. *With the theory of this section, one may be tempted to consider only Π-invariant annihilators for an Π-invariant set S. Then, however, low-degree annihilators may be overlooked.*

One example is the $\boldsymbol{\Pi_3}$-invariant set[12] $S := \{[000], [100], [111]\} \subset \mathbb{F}_2^3$ with $\mathrm{lad}(S) = 2$. A possible basis for all degree-2-annihilators is the set $\{x_1 \cdot (x_2 + x_3), x_3 \cdot (x_1 + x_2)\}$. On the other hand, $\mathrm{ann}(S)$ contains no non-zero $\boldsymbol{\Pi_3}$-invariant functions of degree ≤ 2. This can be shown as follows.

Assume that a $\boldsymbol{\Pi_3}$-invariant annihilator $f := c_0 + c_1 \cdot \sigma_1 + c_2 \cdot \sigma_2 \not\equiv 0$ of degree

[11] Observe that the choice of the representants $\boldsymbol{X_i}$ has a direct impact on the concrete elements in $\mathcal{F}_i$.

[12] Recall the definition from page 85 that $[\boldsymbol{x_1} \ldots \boldsymbol{x_r}] = \{(X_1, \ldots, X_r) | \ |X_i| = \boldsymbol{x_i}\}$.

≤ 2 *exists. Then, the condition* $f(\boldsymbol{X}) = 0$ *for all* $\boldsymbol{X} \in S$ *yields*

$$\begin{aligned} 0 &= f([000]) = c_0 \\ 0 &= f([100]) = c_0 + c_1 \\ 0 &= f([111]) = c_0 + c_1 + c_2 \end{aligned}$$

This implies $c_0 = c_1 = c_2 = 0$ *and therefore* $f \equiv 0$*. I.e., the* $\boldsymbol{\Pi}_3$*-invariant set* S *has quadratic annihilators which are all* ***not*** $\boldsymbol{\Pi}_3$*-invariant.*

Until know, it is an open question to find efficient methods to decide for a Π-invariant set S if any of the Π-invariant annihilators has a degree equal to $\mathrm{lad}(S)$. To know that this is the case would enormously accelerate the search for low-degree $\boldsymbol{Z}$-functions for permutation invariant (ι, m)-combiners like E_0.

4.2.4 A quadratic time algorithm

The algorithms discussed so far for computing the lowest annihilator degree d of a function f with n variables are all based on computing the kernel of the matrix $M_d(\mathrm{supp}(f))$, giving a running time in $\mathcal{O}(\mu_2(n,d)^3)$. In [ArmCGKMR06], a new algorithm was proposed which reduced the time effort to an amount in $\mathcal{O}(\mu_2(n,d)^2)$, being the first quadratic time algorithm to compute AI so far.

It allowed to compute $AI(f)$ for Boolean functions $f : \mathbb{F}_2^n \to \mathbb{F}_2$ for higher values of n than it has been achieved before. For example, the authors evaluated the values of AI for the inverse function and some Kasami type power functions (e.g., see [CanV02]) for $12 \leq n \leq 20$. However, the authors clearly point out that it is a difficult (but doable) task to implement the algorithm in such a way that the effort estimations are fulfilled[13].

We sketch here only the algorithm itself for the case $\mathbb{F} = \mathbb{F}_2$ (Algorithm 4.5) and refer for details to [ArmCGKMR06]. The algorithm is based on iteratively solving systems of linear equations $U \cdot G = B$ with U being an upper triangular matrix. One by one it takes the elements in $\mathrm{supp}(f)$ to compute an annihilator g derived from G for the elements already considered. If g is already an annihilator of $\mathrm{supp}(f)$, then the algorithm stops, otherwise it takes a new element from $\mathrm{supp}(f)$ and modifies g accordingly. The construction of G is done in such a way that g consists after the i-th round of a linear combinations of the first i-th monomials, ordered by the degree. Therefore, the first annihilator found for the whole set $\mathrm{supp}(f)$ has

[13] In particular the memory management is not easy.

necessarily the lowest possible degree. Worth mentioning is that the algorithm can be easily adapted to derive a basis of all annihilators with the minimum degree $\mathrm{lad}(f)$.

Algorithm 4.5.
Computation of an annihilator of minimum degree
Input: *f, $S := \mathrm{supp}(f) = \{\boldsymbol{X}_1, \ldots\}$, $\mathcal{E}_{\leq \lceil n/2 \rceil} = \{E_1, \ldots\}$ being the set of exponents of degree $\leq \lceil n/2 \rceil$ such that $|E_i| \leq |E_j|$ for $i \leq j$.*
Output: *An annihilator of f of minimum degree.*

1: *Initialize the following variables:* $U_1 \leftarrow (\boldsymbol{X}_1^{E_1})$, $v_1 \leftarrow f(\boldsymbol{X}_1) \oplus 1$, $G_1 \leftarrow (1) \in \mathbb{F}_2^1$, $P \leftarrow (x_1)$, $i \leftarrow 1$.
2: ***while*** *the polynomial associated to G is not an annihilator of f* ***do***
3: $i \leftarrow i + 1$.
4: $$\begin{pmatrix} U_i & P(\boldsymbol{X}_1^{E_i}, \ldots, \boldsymbol{X}_{i-1}^{E_i}) \\ \boldsymbol{X}_i^{E_1} \ldots \boldsymbol{X}_i^{E_{i-1}} & \boldsymbol{X}_i^{E_i} \end{pmatrix} \overset{\text{row ops.}}{\rightarrow} \underbrace{\begin{pmatrix} U_i & P(\boldsymbol{X}_1^{E_i}, \ldots, \boldsymbol{X}_i^{E_i}) \\ 0 \ldots 0 & \end{pmatrix}}_{=:U_{i+1}}$$
5: *Use the same row operations from line 4 to perform the update* $(P(v_1, \ldots, v_{i-1}), v_i) \mapsto P(v_1, \ldots, v_i)$.
6: *Solve* $U_i \cdot G_i = P(v_1, \ldots, v_i)$ *with* $G_i = (g_1, \ldots, g_i)$.
7: ***end while***
8: *Output* $g(X) := \bigoplus_{j=1}^{i} g_j \cdot X^{E_j}$.

4.3 Design principles

After considering the question how to find low degree equations for an algebraic attack on (ι, m)-combiners in the previous section, we come now to the opposite task, namely how to *avoid* them. In this section, we will provide several design principles for simple combiners and combiners with memory such that a lower bound for $\mathrm{lad}(r)$ is guaranteed for all values of r between 1 and the length of the shortest LFSR.

At first, we will provide several statements on lad, and after that, we explain some design methods for simple combiners and combiners with memory.

4.3.1 Properties of the lowest annihilator degree

In this subsection, we collect some basic facts on the lowest annihilator degree. We consider first the values of lad for related sets:

Theorem 4.43. *Let $S \subseteq \mathbb{F}^n$. It holds that*

1. *For $S' \subseteq S$, it holds that $\mathrm{lad}(S') \leq \mathrm{lad}(S)$.*
2. *For a regular matrix $M \in GL_n(\mathbb{F})$ and a vector $V \in \mathbb{F}^n$, we define*

$$S \cdot M + V := \{\boldsymbol{X} \cdot M + V | \boldsymbol{X} \in S\}.$$

It holds that $\mathrm{lad}(S \cdot M + V) = \mathrm{lad}(S)$.

Proof. The first claim is obvious, as any annihilator of S gives automatically zero on any subset of S.

For the second claim, recall that the inverse of an affine bijective mapping $S \mapsto S \cdot M + V$ is affine too. Therefore, it suffices to show that $\mathrm{lad}(S \cdot M + V) \geq \mathrm{lad}(S)$. Let $f(X) = f(x_1, \ldots, x_n)$ be an annihilator of $S \cdot M + V$ of degree $\mathrm{lad}(S \cdot M + V)$ and set $\hat{f}(X) := f(X \cdot M + V)$. Observe that $\deg(\hat{f}) = \deg(f)$, as the degree of a function is invariant under linear operations on the variables. Now, it holds for any $\boldsymbol{X} \in S$ that

$$\hat{f}(\boldsymbol{X}) \stackrel{\text{def}}{=} f(\underbrace{\boldsymbol{X} \cdot M + V}_{\in S \cdot M + V}) = 0.$$

Thus, $\hat{f}$ is an annihilator of S. As f was an arbitrary annihilator of $S \cdot M + V$, this shows that $\mathrm{lad}(S) \leq \deg(\hat{f}) = \mathrm{lad}(S \cdot M + V)$. Hence, the second claim is proved. □

Assume now that we are interested in constructing a d-immune set $S \in \mathbb{F}^n$ from scratch. The certainly most simple approach is given in Algorithm 4.6. It is based on Theorem 4.27 which says that a set S is d-immune if and only if $M_d(S)$ has full column rank. The algorithm works as follows. As long as $M_d(S)$ has not full column rank, a new element $\boldsymbol{X} \notin S$ is searched such that $\operatorname{rank}(M_d(S \cup \{\boldsymbol{X}\})) > \operatorname{rank}(M_d(S))$. Then, the set S is updated to $S \cup \{\boldsymbol{X}\}$ and so on.

Algorithm 4.6.
Constructing a d-immune set $S \subseteq \mathbb{F}^n$
Input: $n, d \geq 1$
Output: *A set $S \subseteq \mathbb{F}^n$ with* $\operatorname{lad}(S) \geq d+1$

1: *Set $S := \{\boldsymbol{X}\}$ for a random element $\boldsymbol{X} \in \mathbb{F}^n$*
2: ***while*** *$M_d(S)$ has not the full column rank* ***do***
3: *Search random $\boldsymbol{X} \in \mathbb{F}^n \setminus S$ such that* $\operatorname{rank}(M_d(S)) < \operatorname{rank}(M_d(S \cup \{\boldsymbol{X}\}))$
4: *Set $S := S \cup \{\boldsymbol{X}\}$*
5: ***end while***
6: ***return*** S_i

Although simple, the algorithm is quite efficient to construct some d-immune set. Observe that in practice, one would transform $M_d(S)$ in row echelon form, so that testing if the rank of $M_d(S \cup \{\boldsymbol{X}\})$ is higher than the rank of $M_d(S)$ is quite easy. Furthermore, due to [ZenYR89], the probability that i random vectors in $\mathbb{F}_q^D$ with $D := \mu_q(n,d)$ are linearly independent is equal to

$$\text{Prob} = \prod_{j=D-i+1}^{D} \left(1 - \frac{1}{q^j}\right). \tag{4.3.31}$$

Although the rows of $M_d(S)$ are not really random, one can expect (and simulations confirmed this) that at each loop, only few elements $\boldsymbol{X} \in \mathbb{F}^n \setminus S$ need to be tested until one is found that gives a vector that is linearly independent from the ones chosen before.

Of course, generating random sets S is quite inefficient in practice. Furthermore, it makes it hard to analyze further properties. Thus, one would prefer to construct d-immune sets more systematically. In the case of $\mathbb{F} = \mathbb{F}_2$, Algorithm 4.2 implies a canonical d-immune set.

Proposition 4.44. *Let $\mathbb{F} := \mathbb{F}_2$ and $n, d \geq 1$. The set $X_{\leq d} := \{\boldsymbol{X} \in \mathbb{F}^n || \boldsymbol{X}| \leq d\}$ is d-immune, i.e.* $\operatorname{lad}(X_{\leq d}) > d$.

Proof. Let $\mathcal{F} := \{X^E |\ E \in \{0,1\}^n, |E| \leq d\}$. We show that the matrix $M_{\mathcal{F}}(X_{\leq d})$ has full rank. To do so, we order both the elements in $\mathcal{F}$ and in S with respect to the grevlex-ordering $\succ$.

Then, the diagonal elements of $M_{\mathcal{F}}(X_{\leq d})$ are $m_{\boldsymbol{X}}(\boldsymbol{X})$ which is equal to 1. Above these entries are the values of $\boldsymbol{X}^E$ with $E \succ \boldsymbol{X}$ which are 0. Hence, $M_{\mathcal{F}}(X_{\leq d})$ is a lower triangular matrix with full rank. □

Example 4.45. *Let $\mathbb{F} := \mathbb{F}_2$, $n := 4$ and $d := 2$. Consider the set*

$$X_{\leq 2} := \{(0000), (1000), (0100), (0010), (0001), (1100), (1010), (1001), (0110), (0101), (0011)\}.$$

The matrix $M_2(X_{\leq 2})$ is

$$\begin{array}{c} \\ 0000 \\ 1000 \\ 0100 \\ 0010 \\ 0001 \\ 1100 \\ 1010 \\ 1001 \\ 0110 \\ 0101 \\ 0011 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} 1 & x_1 & x_2 & x_3 & x_4 & x_1x_2 & x_1x_3 & x_1x_4 & x_2x_3 & x_2x_4 & x_3x_4 \end{array} \\ \left(\begin{array}{ccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) \end{array}$$

As $M_2(X_{\leq 2})$ has a triangular form, it has full rank. Thus, in this case no annihilators of degree 2 exist.

4.3.2 Simple combiners

By Theorem 4.3, any $\boldsymbol{Z}$-function is an annihilator of $X_{\boldsymbol{Z}}$. From Proposition 4.4, we know that we can limit our considerations to the sets $X_{\boldsymbol{z}}$ for $\boldsymbol{z} \in \mathbb{F}$. Algorithm 4.7 uses these observations to construct an output function f with $\mathrm{lad}(1) > d$ for a given value d.

Algorithm 4.7.
Constructing a function $f : \mathbb{F}_q^n \to \mathbb{F}_q$ *with* $\mathrm{lad}(1) > d$
Input: $n \geq 1$, *a finite field* $\mathbb{F}_q$, *an integer* d
Output: *A function* $f : \mathbb{F}_q^n \to \mathbb{F}_q$ *such that* $\mathrm{lad}(1) > d$

1: *Find* q *distinct sets* $S_{\boldsymbol{z}}$, *all being* d*-immune*
2: *Define* $f : \mathbb{F}_q^n \to \mathbb{F}_q$ *as follows*

$$f(\boldsymbol{X}) := \begin{cases} \boldsymbol{z} & , \boldsymbol{X} \in S_{\boldsymbol{z}} \\ \text{arbitrary} & , \text{else} \end{cases} \tag{4.3.32}$$

3: ***return*** f

Because of $\mathrm{lad}(X_{\boldsymbol{z}}) \geq \mathrm{lad}(S_{\boldsymbol{z}}) > d$ for all $\boldsymbol{z} \in \mathbb{F}$, it holds that $\mathrm{lad}(1) > d$ as demanded. Observe that Algorithm 4.7 depicts rather a general approach than a concrete design proposal. For example, it completely ignores the problem how to generate q disjoint sets, all being d-immune. One possibility is to construct the sets $S_{\boldsymbol{Z}}$ in parallel, using the property that $\mathrm{lad}(S) = \mathrm{lad}(S+V)$ for any set S and any vector V (see Theorem 4.43). In this context we want to refer to the algorithm in [ArmK06] based on matroids, which solves a similar problem and might be of help for the case here.

Once again, the case of $\mathbb{F} = \mathbb{F}_2$ is easier as only the two sets $X_{(0)}$ and $X_{(1)}$ need to be considered. Theorem 4.43 and Proposition 4.44 provide the background for using $S_{(0)} := X_{\leq d}$ and looking for a vector $V \in \mathbb{F}_2^n$ such that $S_{(0)}$ and $S_{(1)} := S_{(0)} + V$ are disjoint. This is for example the case for $V := (1, \ldots, 1)^T$. Let $\boldsymbol{X} \in S_{(0)}$, then $|\boldsymbol{X}| \leq d$ by definition. As XORing $\boldsymbol{X}$ with V means to flip each entry, it holds that $|\boldsymbol{X}+V| \geq n-d > d$ if $2d < n$. The last condition is equivalent to $d \leq \lfloor n/2 \rfloor$. In other words, defining f as shown in (4.3.32) gives a simple combiner with $\mathrm{lad}(r) > d$ if $d \leq \lfloor n/2 \rfloor$. This means that f is defined as follows:

$$f(\boldsymbol{X}) := \begin{cases} 0 & , \boldsymbol{X} \in X_{\leq d} \\ 1 & , \boldsymbol{X} \in X_{\leq d} + (1, \ldots, 1)^T \end{cases} = \begin{cases} 0 & , |\boldsymbol{X}| \leq d \\ 1 & , |\boldsymbol{X}| \geq n-d \end{cases} . \tag{4.3.33}$$

For $d = \lfloor n/2 \rfloor$, (4.3.33) describes the so-called majority function:[14]

Proposition 4.46. *The* ***majority function*** maj *with* n *inputs is defined by*

$$\mathrm{maj}(\boldsymbol{X}) := \begin{cases} 0 & , |\boldsymbol{X}| \leq \lceil n/2 \rceil - 1 \\ 1 & , |\boldsymbol{X}| \geq \lfloor n/2 \rfloor + 1 \end{cases} . \tag{4.3.34}$$

[14] The name comes from the fact that $f(\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}) = 0$ if the majority of the entries is equal to zero, and equal to 1 otherwise.

For n odd, this defines a balanced function. In the case of n even, one can assign arbitrary outputs for the inputs $\boldsymbol{X}$ of weight $n/2$. The majority function is an algebraically immune function. This means that $\mathrm{lad}(\mathrm{maj}) = \mathrm{lad}(\mathrm{maj}+1) = \lfloor n/2 \rfloor + 1$.

To use this kind of functions (and similar variants) has been proposed in several papers, for example in [DalMS05, BraP05, ArmKS05].

Altogether, we have seen that creating an output function f such that $\mathrm{lad}(1)$ is maximum is doable, at least for the case $\mathbb{F} = \mathbb{F}_2$. But, to be resistant against other attacks, f has to meet other criteria, as high correlation immunity [Sie84], high nonlinearity (e.g, see [MeiS89]) and others. More intelligent algorithms to obtain Boolean functions with maximum lad have been described in [DalGM05, QuFL05], but to the best of our knowledge, it is still an open task to design output functions f with a good trade-off between these demands, including high lad.

4.3.3 Combiners with memory

Combiners with memory were originally introduced in [Rue85] to avoid correlation attacks. Although this was not achieved (see according statements in Section 2.6), they still might be a good way to achieve better trade-offs between the vulnerability against existing attacks than in the case of simple combiners. However, almost all papers on strengthening combiners against algebraic attacks focus only on simple combiners and neglect combiners with memory. The only exception we are aware of is our paper [ArmKS05] where a design principle is developed for combiners with memory which guarantees a lower bound for $\mathrm{lad}(r)$ with r between 1 and the length of the shortest LFSR. In this section we present the main result but refer for more details to the paper.

The following theorem, a slight extension from a theorem in [ArmKS05], shows that under certain conditions, each "local" lower bound for $\mathrm{lad}(X_{\boldsymbol{Q},(\boldsymbol{z}_r)})$[15] is also a "global" lower bound for $\mathrm{lad}(X_{(\boldsymbol{z}_1,\ldots,\boldsymbol{z}_r)})$.

Theorem 4.47. *Let $\alpha : \mathbb{F}^\iota \to \mathbb{F}$ be a function such that $\mathrm{lad}(\alpha^{-1}(\boldsymbol{z})) = d$ for all $\boldsymbol{z} \in \mathbb{F}$, and $\beta : \mathbb{F}^m \to \mathbb{F}$ be arbitrary. Furthermore, we denote by $r^* \geq 0$ the maximum number such that $X_{\boldsymbol{Q},\boldsymbol{Z}} \neq \emptyset$ for all $\boldsymbol{Q} \in \mathbb{F}^m$ and $\boldsymbol{Z} \in \mathbb{F}^{r^*}$.*[16]

If the output function f can be expressed as

$$f(Q, X) := \alpha(X) + \beta(Q) = z, \tag{4.3.35}$$

[15] Recall the definition of $X_{Q,Z}$ on page 88.

[16] In a properly designed (ι, m)-combiner, r^* would be the length of the shortest LFSR involved.

then it holds for all $r^* \geq r \geq 1$, $\boldsymbol{Z} = (\boldsymbol{z_1}, \ldots, \boldsymbol{z_r}) \in \mathbb{F}^r$ *and* $\boldsymbol{Q} \in \mathbb{F}^m$ *that*

$$\mathrm{lad}(X_{\boldsymbol{Z}}) \geq \mathrm{lad}(X_{\boldsymbol{Q},\boldsymbol{Z}}) = d.$$

Proof. Because of $X_{\boldsymbol{Q},\boldsymbol{Z}} \subseteq X_{\boldsymbol{Z}}$, the first inequality follows by Theorem 4.43.

It remains to show that $\mathrm{lad}(X_{\boldsymbol{Q},\boldsymbol{Z}}) = d$ for all $r^* \geq r \geq 1$, $\boldsymbol{Z} = (\boldsymbol{z_1}, \ldots, \boldsymbol{z_r}) \in \mathbb{F}^r$ and $\boldsymbol{Q} \in \mathbb{F}^m$. For $r = 1$ it holds for all choices $\boldsymbol{z} \in \mathbb{F}$ and $\boldsymbol{Q} \in \mathbb{F}^m$ that $X_{\boldsymbol{Q},(\boldsymbol{z})} = \alpha^{-1}(\beta(\boldsymbol{Q}) - \boldsymbol{z})$ and therefore $\mathrm{lad}(X_{\boldsymbol{Q},(\boldsymbol{z})}) = d$ by assumption on α.

Let $r \geq 1$, $\boldsymbol{Z} = (\boldsymbol{z_1}, \ldots, \boldsymbol{z_r}) \in \mathbb{F}^r$, $\boldsymbol{Q_1} \in \mathbb{F}^m$ and $g(X_1) \in \mathbb{F}[X_1]$ be an annihilator of $X_{\boldsymbol{Q_1},(\boldsymbol{z_1})}$. Then, g can be interpreted as an element in $\mathbb{F}[X_1, \ldots, X_r]$ which annihilates $X_{\boldsymbol{Q_1},\boldsymbol{Z}}$, too. This shows that $\mathrm{lad}(X_{\boldsymbol{Q_1},\boldsymbol{Z}}) \leq \mathrm{lad}(X_{\boldsymbol{Q_1},(\boldsymbol{z_1})}) = d$.

We prove now by induction over r that $\mathrm{lad}(X_{\boldsymbol{Q_1},\boldsymbol{Z}}) \geq d$ for all choices of $\boldsymbol{Q_1}$ and $\boldsymbol{Z}$. The case $r = 1$ has be considered already. Now let $1 < r \leq r^*$ be fixed and the claim be true for all $r' < r$. Fix $\boldsymbol{Z} = (\boldsymbol{z_1}, \ldots, \boldsymbol{z_r})$ and $\boldsymbol{Q_1}$ and choose $g(X_1, \ldots, X_r) \in \mathrm{ann}(X_{\boldsymbol{Q_1},\boldsymbol{Z}})$ having the minimal degree $\mathrm{lad}(X_{\boldsymbol{Q},\boldsymbol{Z}})$. Choose an arbitrary value $(\boldsymbol{X_1}, \ldots, \boldsymbol{X_r}) \in X_{\boldsymbol{Q_1},\boldsymbol{Z}}$ which is not empty by assumption. Set $\boldsymbol{Q_2} := \Psi(\boldsymbol{Q_1}, \boldsymbol{X_1})$. Then $g^*(X_2 \ldots, X_r) := g(\boldsymbol{X_1}, X_2, \ldots, X_r)$ annihilates $X_{\boldsymbol{Q_2},(\boldsymbol{z_2},\ldots,\boldsymbol{z_r})}$. This shows that

$$\mathrm{lad}(X_{\boldsymbol{Q_1},\boldsymbol{Z}}) = \deg(g) \geq \deg(g^*) \geq \mathrm{mindeg}(X_{\boldsymbol{Q_2},(\boldsymbol{z_2},\ldots,\boldsymbol{z_r})}) \geq d,$$

where the last inequality is true by induction assumption. □

The theorem implies the following strategy. Choose an output function α such that $\mathrm{lad}(\alpha^{-1}(\boldsymbol{z}))$ is the maximum possible value, for example with the algorithms mentioned in Section 4.3.2. This will guarantee the same lower bound for the degree of *any* $\boldsymbol{Z}$-functions, as long as the inputs $X_1, \ldots, X_r$ are independent elements in $\mathbb{F}^\iota$.

We emphasize that for combiners with memory Theorem 4.47 yields the only lower bound for algebraic attacks proposed so far. Furtheron, this approach can be combined with other methods presented in [ArmKS05] to achieve maximum resistance against certain correlation attacks. One of our results was that already small changes of E_0 increase the resistance against certain attacks considerably.

In the case of $\mathbb{F} = \mathbb{F}_2$, we know that the highest value of d is equal to $\lceil \iota/2 \rceil$ which might not be enough to ensure a reasonable security margin. But, in our simulations on reduced version of E_0 in [ArmKS05], it was observed that $\mathrm{lad}(X_{\boldsymbol{Z}})$ was actually bigger than d. Besides, one can increase the value of d by increasing ι. This can be achieved without augmenting the number of LFSRs. Instead, α could for example use the outputs of each LFSRs from two successive clocks, giving a function $\alpha : \mathbb{F}^{2\iota} \to \mathbb{F}$ and by doing so doubling the value of d. The drawback of course is that now it takes two clocks for generating one output bit, halving the speed of the (ι, m)-combiner.

4.4 Algebraic attacks with related keys

In the preceding sections, we derived criteria for the existence of low degree equations for an algebraic attack and discussed how they can be avoided by designers. However, even if the (ι, m)-combiner is designed optimally in this sense, other ways may exist to express the keystream generator by a system of low degree equations. One possibility are fast algebraic attacks, the whole next chapter being dedicated to this subject. Another possibility is given in the case that the keystream generator is run several times with different unknown initial states, the keys, of the LFSRs but where some relations between the keys are known. At the first sight, it may seem to be a rather unrealistic scenario. But actually, there exist at least two occasions where this might happen:

Linear key schedule

Keystream generators have to be initialized with a secret key on which the sender and the receiver agree beforehand. For security reasons and to avoid synchronization loss, a **key schedule** is deployed in general. This means that both communication partners agree secretly on one key $K \in \mathbb{F}_q^n$ and on a public algorithm. The algorithm derives after a fixed number of clocks a new initial state from K and from some public parameters, e.g a counter or the actual system time. For the sake of efficiency, key schedules are often chosen to be linear (e.g. in Bluetooth). In [DaeGV93], it was shown that linear key schedules may weaken simple combiners.

Fault analysis

If the device on which the keystream generator is executed is not tamper-resistant, an attacker might have the ability to control it in a limited way. More precisely, the attack model is that the adversary can re-set the device to the same unknown initial setting as often as he wants and re-start it to generate the keystream. While the keystream generator is running, he can induce (possibly unknown) errors. Thereby, he knows the affected register but can neither control the point in time of the disturbance nor the induced error itself.

Fault analysis was first introduced in 1996 in [BonDL97] to attack number theoretic public key cryptosystems such as RSA, and later in [BihS97] to attack product block ciphers such as DES. These attacks are practical, and various techniques have been described that induce faults during cryptographic computations (cf. [SkoA02]). More recently, fault analysis of stream ciphers has been introduced by Hoch and Shamir in [HocS04].

As for other cryptosystems, fault analysis on stream ciphers is a powerful cryptanalytic tool which can work even if direct attacks are inefficient. In [HocS04], general techniques are applied to attack standard constructions of stream ciphers based on LFSR's, and some specialized techniques are introduced that work, e.g., against RC4. In [BihGN05], different and improved fault analysis methods on RC4 are derived.

Algebraic attacks with related keys

Both attack scenarios described in the previous sections can be summarized by the following attack model:

Definition 4.48. *The attack model for the **related key scenario** is as follows. Once the (ι, m)-combiner is initialized, it generates a keystream of length T. This time period is called a **frame**. At the beginning of the i-th frame, the (ι, m)-combiner is initialized to $S_0^{(i)} := (Q_0^{(i)}, K^{(i)}) \in \mathbb{F}^m \times \mathbb{F}^n$ and produces the keystream $z_0^{(i)}, \ldots, z_{T-1}^{(i)}$. Thereby, the keys $K^{(i)}$, called the **frame keys**, are related in the following sense. There exists a value $K \in \mathbb{F}^n$, called the **master key**, a matrix $M \in GL_n(\mathbb{F})$ and vectors $\Delta^{(i)} \in \mathbb{F}^n$ such that*

$$K^{(i)} = K \cdot M + \Delta^{(i)}. \tag{4.4.36}$$

Particularly, for $i, j \in \{1, \ldots, N\}$, where N specifies the number of frames considered, one can specify a vector $\Delta^{(i,j)}$ such that $K^{(j)} = K^{(i)} + \Delta^{(i,j)}$. Regarding the initial states $Q_0^{(i)}$ of the memory, it is not necessary to know any relations between $Q^{(i)}$ and $Q^{(j)}$.

The assumptions are that an adversary knows the specifications of the (ι, m)-combiner, the matrix M, the vectors $\Delta^{(i)}$ and some of the keystream elements $z_t^{(i)}$ with $0 \leq i \leq N$. The attacker's goal is to find out the value of the master key K.

*We thereby distinguish between a **passive attacker**, who can only observe, and an **active attacker**, who is able to choose the values of M and $\Delta^{(i)}$.*

Remark 4.49. *In a fault attack, it holds that M is the identity matrix, as only a fault $\Delta^{(i)}$ is added to K. But, as opposed to the related key attack scenario described in Definition 4.48, the fault attack model assumes that an attacker does not know the values of the induced faults.*

In the case that the fault is induced only into the memory register, all the keys $K^{(i)}$ are equal, i.e. $\Delta^{(i)} = 0$, so that the assumption from the related key scenario is valid.

In the case that faults are induced into the LFSRs, the values of $\Delta^{(i)}$ are unknown at first. However, in [ArmM05], we described several methods to

reconstruct the faults $\Delta^{(i)}$, if the fault is induced only into one LFSR and if the attacker is able to observe as many keystream bits as he needs (and T is not too small). Simulations confirmed the validity of these methods. As the effort is closely related to the length of the affected LFSR, we assumed that the fault was induced into the shortest length.

Although there are still many open questions as for the practicability of these methods or for possible improvements, we know at least that it is possible in principle to mount fault attacks such that the faults $\Delta^{(i)}$ are known. Therefore, we leave the question of how to determine (efficiently) the values of the induced faults as a separate topic and concentrate only on algebraic attacks in the related key scenario.

In the following, we will describe algebraic attacks in the related key scenario. This covers our results from [ArmLP04, ArmM05]. Before we start, we simplify the notations. W.l.o.g., we can assume that M is equal to the identity matrix.[17] Furthermore, we can suppose that $\Delta^{(0)} = \overrightarrow{0}$. This yields that

$$\begin{aligned} K^{(0)} &= K \\ K^{(i)} &= K + \Delta^{(i)},\ 1 \leq i \leq N. \end{aligned}$$

We refer to frame number 0 as the **original frame** and omit the exponent (0) for it. That is, $z_t = z_t^{(0)}$.

As usual, we denote by $K_t^{(i)} = K^{(i)} \cdot L^t \cdot P$ the inputs to f at clock t in frame i and by $K_t = K \cdot L^t \cdot P$ the inputs in the original frame. Due to the linearity in the LFSRs and in the key schedule, one can derive a simple relation between K_t and $K_t^{(i)}$:

$$\begin{aligned} K_t^{(i)} &= K^{(i)} \cdot L^t \cdot P = (K + \Delta^{(i)}) \cdot L^t \cdot P = K \cdot L^t \cdot P + \underbrace{\Delta^{(i)} \cdot L^t \cdot P}_{=:\Delta_t^{(i)}} \\ &= K_t + \Delta_t^{(i)}. \end{aligned}$$

Our attack is similar to the "conventional" attack as described in Chapter 3. Thus, whenever an adversary knows that $(z_t^{(i)}, \ldots, z_{t+r-1}^{(i)})$ is equal to some value $\boldsymbol{Z} \in \mathbb{F}^r$, he knows that $(K_t + \Delta_t^{(i)}, \ldots, K_{t+r-1} + \Delta_{t+r-1}^{(i)}) \in X_{\boldsymbol{Z}}$. Let

$$\begin{aligned} \Delta_{t\ldots t+r-1}^{(i)} &:= (\Delta_t^{(i)}, \ldots, \Delta_{t+r-1}^{(i)}), \\ X_{\boldsymbol{Z}} - \Delta_{t\ldots t+r-1}^{(i)} &:= \{X - \Delta_{t\ldots t+r-1}^{(i)} \in \mathbb{F}^{\iota \cdot r} \mid X \in X_{\boldsymbol{Z}}\}. \end{aligned}$$

[17] Set $K' := K \cdot M$ and treat K' as the master key. Once K' is found, the "original" master key can be easily reconstructed as M is regular. Actually, the matrix M plays only a role in the case of active attackers who can influence M.

Thus, $(z_t^{(i)}, \ldots, z_{t+r-1}^{(i)}) = \boldsymbol{Z}$ implies that

$$(K_t, \ldots, K_{t+r-1}) \in X_{\boldsymbol{Z}} - \Delta^{(i)}_{t\ldots t+r-1},$$

allowing an adversary to set up an equation

$$F(K_t, \ldots, K_{t+r-1}) = 0,$$

with F being an annihilator of $X_{\boldsymbol{Z}} - \Delta^{(i)}_{t\ldots t+r-1}$ of degree $\mathrm{lad}(X_{\boldsymbol{Z}} - \Delta^{(i)}_{t\ldots t+r-1})$.

As long as one considers only one frame, this is exactly the algebraic attack as described in Chapter 3. But if the attacker additionally knows the values of $(z_t^{(j)}, \ldots, z_{t+r-1}^{(j)}) = \boldsymbol{Z}'$, $i \neq j$, this yields

$$(K_t, \ldots, K_{t+r-1}) \in \left(X_{\boldsymbol{Z}} - \Delta^{(i)}_{t\ldots t+r-1}\right) \cap \left(X_{\boldsymbol{Z}'} - \Delta^{(j)}_{t\ldots t+r-1}\right),$$

which implies

$$\tilde{F}(K_t, \ldots, K_{t+r-1}) = 0$$

with $\tilde{F}$ being an annihilator of $\left(X_{\boldsymbol{Z}} - \Delta^{(i)}_{t\ldots t+r-1}\right) \cap \left(X_{\boldsymbol{Z}'} - \Delta^{(j)}_{t\ldots t+r-1}\right)$ having the lowest annihilator degree.

Recall that the lower the degree of the equations, the better for the attack. Because of $X_{\boldsymbol{Z}} - \Delta^{(i)}_{t\ldots t+r-1} \supseteq \left(X_{\boldsymbol{Z}} - \Delta^{(i)}_{t\ldots t+r-1}\right) \cap \left(X_{\boldsymbol{Z}'} - \Delta^{(j)}_{t\ldots t+r-1}\right)$ and Theorem 4.43, it holds that $\deg(\tilde{F}) \leq \deg(F)$. Thus, knowing the outputs from different frames for related keys might allow an attacker to set up equations in r clocks with a degree below $\mathrm{lad}(r)$.

Example 4.50. *Consider the summation generator from Section 2.5.3 over $\mathbb{F}_2$ with $k = 2$ inputs and $m = 1$ memory bits. From Example 4.5 on page 84, we know that the sets $X_{\boldsymbol{Z}}$ for all $\boldsymbol{Z} \in \mathbb{F}_2^2$ are*

$$\begin{aligned}
X_{(0,0)} &= \{(0000), (0011), (0101), (0110), (1001), (1010), (1101), (1110)\} \\
X_{(0,1)} &= \{(0001), (0010), (0100), (0111), (1000), (1011), (1100), (1111)\} \\
X_{(1,0)} &= \{(0000), (0011), (0100), (0111), (1000), (1011), (1101), (1110)\} \\
X_{(1,1)} &= \{(0001), (0010), (0101), (0110), (1001), (1010), (1100), (1111)\}
\end{aligned}$$

It is easy to calculate with the Algorithms from Section 4.2 that $\mathrm{lad}(X_{\boldsymbol{Z}}) = 2$ for every $\boldsymbol{Z} \in \mathbb{F}_2^2$. Thus, any equation over two clocks has to be at least quadratic.

Now, assume that $(z_t^{(i)}, z_{t+1}^{(i)}) = (0, 0)$, $\Delta^{(i)}_{t\ldots t+1} = (1000)$, $(z_t^{(j)}, z_{t+1}^{(j)}) = (1, 1)$ and $\Delta^{(j)}_{t\ldots t+1} = (1101)$. It holds that[18]

$$\begin{aligned}
X_{(0,0)} - (1000) &= \{(1000), (1011), (1101), (1110), (0001), (0010), (0101), (0110)\}, \\
X_{(1,1)} - (1101) &= \{(1100), (1111), (0100), (1011), (1000), (0111), (0001), (0010)\}.
\end{aligned}$$

[18] Recall that in $\mathbb{F}_2$, the operations $'-'$ and $\oplus$ are actually the same.

This implies that

$$\begin{aligned}(K_t, K_{t+1}) \;\in\; & \big(X_{(0,0)} - (1000)\big) \cap \big(X_{(1,1)} - (1101)\big) \\ & = \{(1000), (1011), (0001), (0010)\} =: S.\end{aligned}$$

Observe that $\mathrm{lad}(S) = 1$*, as* $F(x_1, x_2, x_3, x_4) := x_2$ *is a linear annihilator of* S*. Thus, one can set up the following linear equation in* K_t *and* K_{t+1}*:*

$$0 = F(K_t, K_{t+1}) = F(k_{t,1}, k_{t,2}, k_{t+1,1}, k_{t+1,2}) = k_{t,2}.$$

Summing up, the information from two different frames made it possible to derive a linear 2-function, even though it holds that $\mathrm{lad}(2) = 2$*.*

So, we have seen that knowing the outputs $(z_t^{(i)}, \ldots, z_{t+r-1}^{(i)})$ from different frames i may help to set up equations with a degree lower than then the value $\mathrm{lad}(r)$ which would normally be a lower bound. The following theorem assures that having the information from enough different frames allows to decrease the degree at least by one:

Theorem 4.51. *Consider a fixed* (ι, m)*-combiner and* $r \geq 1$*. Assume that the values* $\mathrm{lad}(X_{\boldsymbol{Z}})$ *are all equal to the same value* d *for all* $\boldsymbol{Z} \in \mathbb{F}^r$*. Let* $F(X_1, \ldots, X_r, y_1, \ldots, y_r) : \mathbb{F}^{\iota \cdot r + r} \to \mathbb{F}$ *be an* r*-function of degree* d*.*[19] *Let*

$$F(X_1, \ldots, X_r, y_1, \ldots, y_r) = \sum_{(E,E') \in \mathcal{E}} c_{(E,E')} \cdot m_{E'}(y_1, \ldots, y_r) \cdot m_E(X_1, \ldots, X_r)$$

be the algebraic normal form of F*. We set* $\mathcal{E}_d := \{(E, E') \in \mathcal{E} \mid |E| = d\}$*, that is the set of exponents in* X_i *which yields the monomials of degree* d*. Furthermore, we denote by* $\epsilon := |\mathcal{E}_d|$ *its size.*

If the values of $(z_t^{(i)}, \ldots, z_{t+r-1}^{(i)})$ *for at least* $\epsilon + 1$ *different frames are known, one can set up an equation in* $K_t, \ldots, K_{t+r-1}$ *of degree* $\leq d - 1$*.*

Proof. W.l.o.g., we assume that the frames, where the keystream parts are known, are numbered from 0 to ϵ. The idea is to find coefficients $\gamma_0, \ldots, \gamma_\epsilon$ such that

$$\sum_{i=0}^{\epsilon} \gamma_i \cdot F(K_t + \Delta_t^{(i)}, \ldots, K_{t+r-1} + \Delta_{t+r-1}^{(i)}, z_t^{(i)}, \ldots, z_{t+r-1}^{(i)})$$

is a function of degree $\leq d - 1$ in the unknowns $K_t, \ldots, K_{t+r-1}$. As each of the summands is equal to zero, this yields the claimed equation.

[19] Recall Theorem 4.17 for constructing r-functions from Z-functions.

First, observe that for any $\Delta_1, \ldots, \Delta_r$ and any exponent E with $|E| = d$, it holds that

$$\underbrace{m_E(X_1 - \Delta_1, \ldots, X_r - \Delta_r)}_{\deg=d} = \underbrace{m_E(X_1, \ldots, X_r)}_{\deg=d} + \underbrace{R(X_1, \ldots, X_r)}_{\deg\leq d-1}$$

with an appropriate function R. This can be seen by simply outfactoring the expression $m_E(X_1 - \Delta_1, \ldots, X_r - \Delta_r)$.

We introduce the abbreviations $K_{t\ldots t+r-1} := (K_t, \ldots, K_{t+r-1})$ and $z^{(i)}_{t\ldots t+r-1} := (z^{(i)}_t, \ldots, z^{(i)}_{t+r-1})$. Let $\mathcal{E}_d := \{(E_1, E'_1), \ldots, (E_\epsilon, E'_\epsilon)\}$. Then, the knowledge of the values of $z^{(i)}_{t\ldots t+r-1}$ and the r-function F yield the following equations

$$\begin{aligned}
0 &= F(K^{(0)}_{t\ldots t+r-1} - \Delta^{(0)}_{t\ldots t+r-1}, \boldsymbol{z^{(0)}_{t\ldots t+r-1}}) \\
&= \underbrace{R^{(0)}(K^{(0)}_{t\ldots t+r-1}, \boldsymbol{z^{(0)}_{t\ldots t+r-1}})}_{\deg<d} + \underbrace{\sum_{j=1}^{\epsilon} c_{(E_j,E'_j)} \cdot m_{E'_j}(\boldsymbol{z^{(0)}_{t\ldots t+r-1}}) \cdot m_{E_j}(K^{(0)}_{t\ldots t+r-1})}_{\deg=d} \\
&\vdots \\
0 &= F(K^{(\epsilon)}_{t\ldots t+r-1} - \Delta^{(\epsilon)}_{t\ldots t+r-1}, \boldsymbol{z^{(\epsilon)}_{t\ldots t+r-1}}) \\
&= \underbrace{R^{(\epsilon)}(K^{(\epsilon)}_{t\ldots t+r-1}, \boldsymbol{z^{(\epsilon)}_{t\ldots t+r-1}})}_{\deg<d} + \underbrace{\sum_{j=1}^{\epsilon} c_{(E_j,E'_j)} \cdot m_{E'_j}(\boldsymbol{z^{(\epsilon)}_{t\ldots t+r-1}}) \cdot m_{E_j}(K^{(\epsilon)}_{t\ldots t+r-1})}_{\deg=d}
\end{aligned}$$

where $R^{(i)}(K^{(i)}_{t\ldots t+r-1}, \boldsymbol{z^{(i)}_{t\ldots t+r-1}})$ are functions of degree $\leq d-1$. Let

$$V^{(i)} := \left(c_{(E_1,E'_1)} \cdot m_{E'_1}(\boldsymbol{z^{(i)}_{t\ldots t+r-1}}),\ \ldots,\ c_{(E_\epsilon,E'_\epsilon)} \cdot m_{E'_\epsilon}(\boldsymbol{z^{(i)}_{t\ldots t+r-1}})\right) \in \mathbb{F}^\epsilon.$$

Thus, we have $\epsilon + 1$ vectors $V^{(i)}$, each of dimension ϵ. Hence, they must be linearly dependent, proving the existence of coefficients $\gamma_0, \ldots, \gamma_\epsilon$ such that $\sum_{i=0}^{\epsilon} \gamma_i \cdot V^{(i)} = \overrightarrow{0}$. This implies that

$$\begin{aligned}
0 &= \sum_{i=0}^{\epsilon} \gamma_i \cdot F(K^{(\epsilon)}_{t\ldots t+r-1} - \Delta^{(\epsilon)}_{t\ldots t+r-1}, \boldsymbol{z^{(\epsilon)}_{t\ldots t+r-1}}) \\
&= \sum_{i=0}^{\epsilon} \gamma_i \cdot R^{(i)}(K^{(i)}_{t\ldots t+r-1}, \boldsymbol{z^{(i)}_{t\ldots t+r-1}}) \\
&\quad + \sum_{j=1}^{\epsilon} m_{E_j}(K^{(0)}_{t\ldots t+r-1}) \cdot \underbrace{\sum_{i=0}^{\epsilon} \gamma_i \cdot c_{(E_j,E'_j)} \cdot m_{E'_j}(\boldsymbol{z^{(i)}_{t\ldots t+r-1}})}_{=0} \\
&= \sum_{i=0}^{\epsilon} \gamma_i \cdot R^{(i)}(K^{(i)}_{t\ldots t+r-1}, \boldsymbol{z^{(i)}_{t\ldots t+r-1}}),
\end{aligned}$$

which is a linear combination of functions of degree $\leq d-1$. Hence, the total degree is $\leq d-1$, giving an equation in $K_{t\ldots t+r-1}$ of degree $\leq d-1$. □

Of course, this approach doesn't necessarily lead to an equation with the minimum degree. To be sure to consider only equations with the lowest possible degree, one should reduce the sets of possible values for $K_{t\ldots t+r-1}$ for each observed part $z^{(i)}_{t\ldots t+r-1}$ of the keystream, derive annihilators for these sets with the minimum degree[20] and use them to set up equations in $K_{t\ldots t+r-1}$. In principle, this corresponds to the algebraic attacks for the related key scenario we proposed in [ArmLP04, ArmM05]. They are summarized in Algorithm 4.8.

Algorithm 4.8.
An algebraic attack for the related key scenario
Input: *An integer* $r \geq 1$*, the values of the known keystream elements* $z^{(i)}_t$ *with* $0 \leq t \leq T-1$ *and* $0 \leq i \leq N$
Output: *Equations in* $K_{t\ldots t+r-1}$ *with the minimum possible degree*

1: *Initialize the sets* $\mathcal{K}_t := \mathbb{F}^{\iota \cdot r}$
2: ***for*** $i = 0, \ldots, N$ ***do***
3: ***for*** $t = 0, \ldots, T-r$ ***do***
4: ***if*** *the values of* $z^{(i)}_{t\ldots t+r-1}$ *are known* ***then***
5: *Set* $\mathcal{K}_t := \mathcal{K}_t \cap \left(X_{\boldsymbol{z}^{(i)}_{t\ldots t+r-1}} - \Delta^{(i)}_{t\ldots t+r-1} \right)$
6: ***end if***
7: ***end for***
8: ***end for***
9: ***for*** $t = 0, \ldots, T-r$ ***do***
10: ***if*** $\mathcal{K}_t \subsetneq \mathbb{F}^{\iota \cdot r}$ ***then***
11: *Derive all annihilators* F *of* $\mathcal{K}_t$ *of degree* $\mathrm{lad}(\mathcal{K}_t)$ *and set up* $F(K_{t\ldots t+r-1}) = 0$
12: ***end if***
13: ***end for***
14: ***return*** *the equations from line 11*

Apart from Theorem 4.51, we are not aware of other statements on how much the degree can drop if the outputs from different frames are considered. As this depends on the sets $X_{\boldsymbol{Z}}$, the actual values of $z^{(i)}_{t\ldots t+r-1}$ and $\Delta^{(i)}_{t\ldots t+r-1}$, we assume that giving a general statement is rather difficult.

[20] This can be done with the algorithms from Section 4.2.

Anyway, our simulations in [ArmLP04, ArmM05] indicate that the attack described by Algorithm 4.8 can be extremely successful in some cases. For example, in [ArmM05], computer simulations showed that a passive attacker can break the E_0 keystream generator by solving a system of linear equations if about 2500 keystream bits are available from 13 frames.

Observe that it is not always necessary to compute the sets $X_{\boldsymbol{Z}}$ to apply the attacks. In [ArmM05], we developed a related key attack with a passive attacker against the stream cipher SNOW 2.0 [EkdJ02], whose output function has a 64-bit memory, making the computation of the sets $X_{\boldsymbol{Z}}$ infeasible. Nonetheless, we could describe an algebraic attack in the related key scenario that describes the secret key by a system of linear equations. Our estimations are that two frames are enough if the values of $\Delta_t^{(i)}$ were all equal to zero[21] and about 2^{12} output words are known.

Not surprisingly, an active attacker is by far more powerful than a passive one. He is able to attack (ι, m)-combiners where it is impractical to compute the sets $X_{\boldsymbol{Z}}$ as long as an r-function is given, e.g. see the attack on Toyocrypt described in [ArmLP04].

Concluding we can say that algebraic attacks in the related key scenario can be extremely dangerous and should be considered by the design of a new (ι, m)-combiner and its field of application.

[21] That is, we considered a fault attack were the fault was only induced to the memory register, leaving the LFSR-register unchanged.

5 Fast algebraic attacks

5.1 Principles

In this chapter, we explain "fast algebraic attacks", introduced in [Cou03]. After discussing the basic principles, we will describe several improvements. Although some of the concepts are valid for arbitrary finite fields, several ideas have been developed specifically for the case of $\mathbb{F}_2$. Thus, we will consider only the finite field $\mathbb{F}_2$ in this chapter.

So far, the fact that we are dealing with keystream generators based on LFSRs was only exploited to generate a system of equations with a bounded degree:

$$\begin{aligned} F_{\boldsymbol{z_{t_1}},\ldots,\boldsymbol{z_{t_1+r-1}}}(K_{t_1},\ldots,K_{t_1+r-1}) &= 0 \\ &\vdots \\ F_{\boldsymbol{z_{t_N}},\ldots,\boldsymbol{z_{t_N+r-1}}}(K_{t_1},\ldots,K_{t_N+r-1}) &= 0 \end{aligned} \tag{5.1.1}$$

with appropriate $\boldsymbol{Z}$-functions $F_{\boldsymbol{Z}}$. But apart from that, the particular structure of the system of equations (5.1.1) has played no role. In [Cou03], it was shown that in certain cases, the properties of the equations allow for reducing the overall degree of the system of equations in an efficient precomputation step. As the time effort of linearization is exponential in the degree of the equations, this results an enormous speed-up of the whole attack.

However, the applicability of fast algebraic attacks is connected to two conditions. First, the system of equations is generated by one r-function $F(X_1,\ldots,X_r,y_1,\ldots,y_r)$. That is, the system of equations (5.1.1) has the form:

$$\begin{aligned} F(K\cdot L^{t_1}\cdot P,\ldots,K\cdot L^{t_1+r-1}\cdot P,\boldsymbol{z_{t_1}},\ldots,\boldsymbol{z_{t_1+r-1}}) &= 0 \\ &\vdots \\ F(K\cdot L^{t_N}\cdot P,\ldots,K\cdot L^{t_N+r-1}\cdot P,\boldsymbol{z_{t_N}},\ldots,\boldsymbol{z_{t_N+r-1}}) &= 0 \end{aligned} \tag{5.1.2}$$

L denotes as usual the linear update function and K the LFSRs' initial states as defined in Section 2.5. This is not really a constraint as an r-function can be easily constructed from $\boldsymbol{Z}$-functions (see Remark 3.9 and Theorem 4.17). Given $F = F(X_1,\ldots,X_r,y_1,\ldots,y_r)$, the second condition is that it can be rewritten to

$$\underbrace{F(X_1,\ldots,X_r,y_1,\ldots,y_r)}_{\deg=d} = \underbrace{G(X_1,\ldots,X_r)}_{\deg=d} + \underbrace{H(X_1,\ldots,X_r,y_1,\ldots,y_r)}_{\deg=d'<d} \tag{5.1.3}$$

where $\deg(G)=d$ and $\deg(H)=d'<d$. That is the terms in the variables X_i, the key variables, of the highest degree are independent of the variables y_i, the keystream variables. Examples for which these conditions hold are the three ciphers E_0, Toyocrypt and LILI-128 [SimDGM00].

To keep the following notations short, we introduce the following identifiers:

$$\begin{aligned} F_t(X, y_1, \ldots, y_r) &:= F(X \cdot L^t \cdot P, \ldots, X \cdot L^{t+r-1} \cdot P, y_1, \ldots, y_r) \\ G_t(X) &:= G(X \cdot L^t \cdot P, \ldots, X \cdot L^{t+r-1} \cdot P) \\ H_t(X, y_1, \ldots, y_r) &:= H(X \cdot L^t \cdot P, \ldots, X \cdot L^{t+r-1} \cdot P, y_1, \ldots, y_r) \end{aligned}$$

for $X = (x_1, \ldots, x_n)$. Hence, (5.1.2) can be rewritten to

$$\begin{array}{ccccccc} F_{t_1}(K, \boldsymbol{z_{t_1}, \ldots, z_{t_1+r-1}}) & = & G_{t_1}(K) & + & H_{t_1}(K, \boldsymbol{z_{t_1}, \ldots, z_{t_1+r-1}}) & = 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \underbrace{F_{t_N}(K, \boldsymbol{z_{t_N}, \ldots, z_{t_N+r-1}})}_{\deg=d} & = & \underbrace{G_{t_N}(K)}_{\deg=d} & + & \underbrace{H_{t_N}(K, \boldsymbol{z_{t_N}, \ldots, z_{t_N+r-1}})}_{\deg=d'<d} & = 0 \end{array}$$

The idea is now to eliminate the parts $G_t(K)$ in the system of equations to get a new function of degree $d' < d$.

Proposition 5.1. *The sequence $(G_t(X))_{t\geq 0}$ is a linear recurring sequence. That is, there exist an integer $T \geq 1$ and coefficients $\gamma_0, \ldots, \gamma_T \in \mathbb{F}_2$ such that*

$$\sum_{i=0}^{T} \gamma_i \cdot G_{t+i}(X) \equiv 0 \quad \forall t \geq 0. \tag{5.1.4}$$

We denote for the linear recurring sequence $(G_t(X))_{t\geq 0}$ the corresponding minimal polynomial by $\min(G_t(X))$.

Proof. First, we prove the existence of such coefficients for the case $t = 0$. $G_i(X)$ are Boolean functions of degree $\leq \deg(G) = d$ in n unknowns. Thus, the number of different monomials that can occur is upper bounded by $\mu_2(n, d) = \binom{n}{0} + \ldots + \binom{n}{d}$. Therefore, the functions $G_0(X)$, $G_1(X)$,..., $G_{\mu_2(n,d)}(X)$ are linearly dependent, which implies the existence of coefficients of $\gamma_0, \ldots, \gamma_T \in \mathbb{F}_2$ with

$$\sum_{i=0}^{T} \gamma_i \cdot G_i(X) \equiv 0 \tag{5.1.5}$$

and $T \leq \mu_2(n, d)$. Let $t \geq 0$ and define $X' := X \cdot L^t$. As the identifiers X in (5.1.5) are unspecified, it just as well holds that

$$0 \equiv \sum_{i=0}^{T} \gamma_i \cdot G_i(X') = \sum_{i=0}^{T} \gamma_i \cdot G_i(X \cdot L^t) = \sum_{i=0}^{T} \gamma_i \cdot G_{t+i}(X).$$

Since t was arbitrary, the proof is complete. □

Remark 5.2. *To keep the notations short, we will sometimes abbreviate* $G_t(X)$ *by* G_t *and* $\min(G_t(X))$ *by* $\min(G)$.

Now, if an attacker knows the values of the keystream bits $z_t, \ldots, z_{t+T+r-1}$ and a characteristic polynomial $\sum_{i=0}^{T} \gamma_i x^i$ of the linear recurring sequence $(G_t(X))_{t\geq 0}$, he can compute a linear combination of the corresponding equations of degree d to get the following equation of degree d':

$$\begin{aligned} 0 &= \sum_{i=0}^{T} \gamma_i \cdot F_{t+i}(K, \boldsymbol{z_{t+i}, \ldots, z_{t+i+r-1}}) \\ &= \underbrace{\sum_{i=0}^{T} \gamma_i \cdot G_{t+i}(K)}_{\equiv 0} + \sum_{i=0}^{T} \gamma_i \cdot H_{t+i}(K, \boldsymbol{z_{t+i}, \ldots, z_{t+i+r-1}}) \\ &=: F^*(K \cdot L^t, \boldsymbol{z_t, \ldots, z_{t+T+r-2}}). \end{aligned} \tag{5.1.6}$$

As we made no assumptions on t, the equation is valid for any $t \geq 0$. Hence, F^* given by (5.1.6) describes a $(T + r - 1)$-function. Its run length is higher than that of F, but the degree d' is lower. Thus, an algebraic attack using linearization profits significantly from this approach.

For example, it was estimated in [Cou03] that the time effort to attack E_0 with fast algebraic attacks is about 2^{54} operations which is several times lower than the 2^{72} operations estimated in the original algebraic attack in [ArmK03]. However, the drawback is that the run length has raised from $r = 4$ to $T + r - 2 \approx 8,882,188$. This imposes two problems. First, the effort to substitute the known keystream values into F^* is much higher than into F. If done naively, this can even be more complex than the attack itself. However, this problem has been solved in [HawR04] (explained later in this section). The other problem is that an algebraic attack with a system of equations built on F^* is less practical as more successive keystream bits need to be known to set up one equation. Therefore, we assume that an attacker is only interested in determining the minimal polynomial $\min(G_t(X))$ as it reduces the value of T to the minimum. Actually, we will see in Section 5.4 that it is possible with our methods from [ArmA05] to go even below this bound without any additional effort.

Example 5.3. *We consider once again the toy* $(2, 0)$*-combiner from Section 2.5.1 and use the canonical* 1*-function* $f(X) - y$ *to generate a system of equations. More precisely, we define* $F_t(X, y) := f(X \cdot L^t \cdot P) - y$ *with* L *and* P *defined as in Section 2.5.1. Thus, the system of quadratic equations has the*

form

$$
\begin{aligned}
0 &= F_0(K, z_0) = a_0b_0 + a_0 + b_0 - z_0 \\
0 &= F_1(K, z_1) = a_1b_1 + a_1 + b_1 - z_1 \\
0 &= F_2(K, z_2) = a_0b_2 + a_0 + a_1b_2 + a_1 + b_2 - z_2 \\
0 &= F_3(K, z_3) = a_0b_0 + a_0b_2 + a_0 + b_0 + b_2 - z_3 \\
0 &= F_4(K, z_4) = a_1b_0 + a_1b_1 + a_1b_2 + a_1 + b_0 + b_1 + b_2 - z_4 \\
0 &= F_5(K, z_5) = a_0b_0 + a_0b_1 + a_0 + a_1b_0 + a_1b_1 + a_1 + b_0 + b_1 - z_5 \\
0 &= F_6(K, z_6) = a_0b_1 + a_0b_2 + a_0 + b_1 + b_2 - z_6 \\
0 &= F_7(K, z_7) = a_1b_0 + a_1 + b_0 - z_7 \\
0 &= F_8(K, z_8) = a_0b_1 + a_0 + a_1b_1 + a_1 + b_1 - z_8 \\
0 &= F_9(K, z_9) = a_0b_2 + a_0 + b_2 - z_9 \\
0 &= F_{10}(K, z_{10}) = a_1b_0 + a_1b_2 + a_1 + b_0 + b_2 - z_{10} \\
0 &= F_{11}(K, z_{11}) = a_0b_0 + a_0b_1 + a_0b_2 + a_0 + a_1b_0 + a_1b_1 + a_1b_2 + a_1 + b_0 + b_1 + b_2 - z_{11} \\
&\vdots
\end{aligned}
$$

with $K = (a_0, a_1, b_0, b_1, b_2)$*. We divide* f *into two parts*

$$f(x_1, x_2, y) = \underbrace{x_1 \cdot x_2}_{=:g(x_1,x_2)} + \underbrace{x_1 + x_2 - y}_{=:h(x_1,x_2,y)}.$$

Now, F *can obviously be split as required for an algebraic attack:*

$$F_t(X, y) = \underbrace{g(X \cdot L^t \cdot P)}_{=:G_t(X)} + \underbrace{h(X \cdot L^t \cdot P, y)}_{=:H_t(X,y)}.$$

We are interested in finding a linear combination in the following expressions that is equal to zero:

$$
\begin{array}{rcccccccc}
G_0(X) & = & x_1x_3 & & & & & & \\
G_1(X) & = & & & & & x_2x_4 & & \\
G_2(X) & = & & & x_1x_5+ & & & & x_2x_5 \\
G_3(X) & = & x_1x_3+ & & x_1x_5 & & & & \\
G_4(X) & = & & & & x_2x_3+ & x_2x_4+ & & x_2x_5 \\
G_5(X) & = & x_1x_3+ & x_1x_4+ & & x_2x_3+ & x_2x_4 & & \\
G_6(X) & = & & x_1x_4+ & x_1x_5 & & & & \\
G_7(X) & = & & & & x_2x_3 & & & \\
G_8(X) & = & & x_1x_4+ & & & x_2x_4 & & \\
G_9(X) & = & & & x_1x_5 & & & & \\
G_{10}(X) & = & & & & x_2x_3+ & & & x_2x_5 \\
G_{11}(X) & = & x_1x_3+ & x_1x_4+ & x_1x_5+ & x_2x_3+ & x_2x_4+ & & x_2x_5
\end{array}
$$

One can easily check that the sum of the following expressions is equal to zero:

$$\begin{array}{lcl}
G_0(X) & = & x_1x_3 \\
G_2(X) & = & x_1x_5+ \quad x_2x_5 \\
G_4(X) & = & x_2x_3+ \quad x_2x_4+ \quad x_2x_5 \\
G_5(X) & = & x_1x_3+ \quad x_1x_4+ \quad x_2x_3+ \quad x_2x_4 \\
G_6(X) & = & x_1x_4+ \quad x_1x_5
\end{array}$$

By Proposition 5.1, this remains true for the equations for one clock later:

$$\begin{array}{lcl}
G_1(X) & = & x_2x_4 \\
G_3(X) & = & x_1x_3+ \quad x_1x_5 \\
G_5(X) & = & x_1x_3+ \quad x_1x_4+ \quad x_2x_3+ \quad x_2x_4 \\
G_6(X) & = & x_1x_4+ \quad x_1x_5 \\
G_7(X) & = & x_2x_3
\end{array}$$

Also, one clock later, the sum is zero:

$$\begin{array}{lcl}
G_2(X) & = & x_1x_5+ \quad x_2x_5 \\
G_4(X) & = & x_2x_3+ \quad x_2x_4+ \quad x_2x_5 \\
G_6(X) & = & x_1x_4+ \quad x_1x_5 \\
G_7(X) & = & x_2x_3 \\
G_8(X) & = & x_1x_4+ \quad x_2x_4
\end{array}$$

Altogether, it holds that

$$G_t(X) + G_{t+2}(X) + G_{t+4}(X) + G_{t+5}(X) + G_{t+6}(X) \equiv 0 \tag{5.1.7}$$

for all $t \geq 0$. Hence, the polynomial $1 + x^2 + x^4 + x^5 + x^6$ is a characteristic polynomial of the sequence $(G_t(X))_{t\geq 0}$. We use this to generate a new system of equations where each equation is linear instead of quadratic:

$$\begin{array}{rcl}
0 & = & F_0(K, z_0) + F_2(K, z_2) + F_4(K, z_4) + F_5(K, z_5) + F_6(K, z_6) \\
 & = & a_1 + b_0 + b_1 + b_2 - z_0 - z_2 - z_4 - z_5 - z_6 \\
0 & = & F_1(K, z_1) + F_3(K, z_3) + F_5(K, z_5) + F_6(K, z_6) + F_7(K, z_7) \\
 & = & a_0 + a_1 + b_0 + b1 - z_1 - z_3 - z_5 - z_6 - z_7 \\
0 & = & F_2(K, z_2) + F_4(K, z_4) + F_6(K, z_6) + F_7(K, z_7) + F_8(K, z_8) \\
 & = & a_0 + b_1 + b_2 - z_2 - z_4 - z_6 - z_7 - z_8 \\
0 & = & F_3(K, z_3) + F_5(K, z_5) + F_7(K, z_7) + F_8(K, z_8) + F_9(K, z_9) \\
 & = & a_1 + b_0 - z_3 - z_5 - z_7 - z_8 - z_9 \\
0 & = & F_4(K, z_4) + F_6(K, z_6) + F_8(K, z_8) + F_9(K, z_9) + F_{10}(K, z_{10}) \\
 & = & a_0 + a_1 + b_1 - z_4 - z_6 - z_8 - z_9 - z_{10} \\
0 & = & F_5(K, z_5) + F_7(K, z_7) + F_9(K, z_9) + F_{10}(K, z_{10}) + F_{11}(K, z_{11}) \\
 & = & a_0 + b_2 - z_5 - z_7 - z_9 - z_{10} - z_{11}
\end{array}$$

Thus, by using the relation in (5.1.7), one can transform the system of quadratic equations into a system of linear equations. Observe that it is not required that the keystream bits are known. One can apply this simplification step once and for all and use it afterwards for attacks on concrete instances.

Algorithm 5.1.
Fast algebraic attack on a (ι, m)-combiner with linearization

Input: *A (ι, m)-combiner, initialized to a secret value $\mathcal{S}_0 = (Q_0, K)$ and the knowledge of some keystream elements generated by this setting*

Output: *The secret key K*

1: *(Only once) Fix $r \geq 1$ and find an r-function $F(X, y_1, \ldots, y_r)$, where $F(K \cdot L^t \cdot P, z_t, \ldots, z_{t+r-1}) = 0$ for all t, such that F can be split as*

$$F(X, y_t, \ldots, y_{t+r-1}) = G(X) + H(X, y_t, \ldots, y_{t+r-1})$$

with $d = \deg(F) = \deg(G) > \deg(H) = d'$.
2: *(Only once) Compute $\min(G_t(X)) = \sum_{i=0}^{T} \gamma_i x^i$.*
3: *Initialize an empty system of equations.*
4: *For each sequence of known keystream elements $(\boldsymbol{z_t, \ldots, z_{t+T+r-1}}) = \boldsymbol{Z}$, add the following equations of degree d' to the system of equations:*

$$F_t^* := \sum_{i=0}^{T} \gamma_i \cdot H_{t+i}(K, \boldsymbol{z_{t+i}, \ldots, z_{t+i+r-1}}) = 0.$$

5: *If number of linearly independent equations is one less than the number of monomials in the system of equations, linearize the system of equations and solve it with Gaussian elimination.*
6: *Recover K from the solution.*
7: ***return*** K

This illustrates the basic principle of fast algebraic attacks. The main difference to algebraic attacks is that before one starts to compute the solution, the degree of the equations is reduced by building a linear combination of many equations. As discussed in Section 3.4, the time and space effort of several steps in an algebraic attack are exponential in the degree d. Thus, if the degree reduction is achievable with sufficiently little effort, fast algebraic attacks are superior to algebraic attacks. Before we turn

our attention to these questions, we sum up the ideas presented so far in Algorithm 5.1.

Remark 5.4. *In the original description of fast algebraic attacks in [Cou03], it is proposed to set up the system of equations with reduced degree once in advance and to substitute the variables z_t with the actual observed values z_t for each attack. However, as memory is mostly the bottle neck here, one should avoid storing equations which are not needed for the attack. Thus, precomputing the equations $F^*(K \cdot L^t, z_t, \ldots, z_{t+T+r-1}) = 0$ makes only sense if the attackers has always access to the same keystream elements, which is rather improbable.*

So far, we have seen that if one knows a linear combination that cancels the degree-d parts of F, this can be used to generate new equations of a lower degree d'. What remains is the question how to find $\min(G_t(X))$.

Of course, it is vital for the whole approach that the minimal polynomial $\min(G_t(X))$ can be found efficiently. For this task, the Berlekamp-Massey algorithm mentioned in Theorem 2.59 from Section 2.4 is certainly the first choice. However, this would require to sum up many functions $G_t(X)$ with $\mu_2(n, d)$ different monomials in the worst case. To avoid this, a modified approach was proposed in [Cou03]. Instead of considering a sequence of polynomials $(G_0(X), G_1(X), \ldots)$, it is transformed into a sequence of bits. The corresponding algorithm is stated in Algorithm 5.2.

Algorithm 5.2.
Computation of $\min(G_t(X))$ *for a fast algebraic attack*
Input: *Boolean function* $G : \mathbb{F}^n \to \mathbb{F}$*, a regular matrix* L *of size* $n \times n$
Output: *The minimal polynomial* $\min(G_t(X))$*, that is coefficients* $\gamma_0, \ldots, \gamma_T$ *such that* $\sum_{i=0}^{T} G_{t+i}(X) \equiv 0$ *for all* $t \geq 0$

1: *Initialize the LFSRs with a random* $\boldsymbol{X} \in \mathbb{F}^n$*. Hereby, one has to pay attention that none of the LFSRs is set to zero.*
2: *Compute the* $\mathbb{F}_2$*-sequence* $G_0(\boldsymbol{X}), G_1(\boldsymbol{X}), \ldots$.
3: *Apply the Berlekamp-Massey algorithm to find* $\min(G_t(\boldsymbol{X}))$*, that is coefficients such that*

$$\sum_{i=0}^{T} \gamma_i G_{t+i}(\boldsymbol{X}) = 0 \quad \forall t \tag{5.1.8}$$

and T *being minimum.*
4: ***return*** $(\gamma_0, \ldots, \gamma_T)$

Example 5.5. *We use the toy* $(2,0)$*-combiner from Section 2.5.1 to illustrate Algorithm 5.2. Let the two LFSRs be initialized with* $(1,0)$ *and* $(1,0,0)$*, respectively. That is,* $\boldsymbol{X} := (1,0,1,0,0)$*. Then, the sequence* $G_0(\boldsymbol{X}), G_1(\boldsymbol{X}), \ldots, G_{19}(\boldsymbol{X})$ *is equal to*

$$(1,0,0,1,0,1,0,0,0,0,0,1,1,0,1,0,0,1,1,0).$$

This is a linear recurring sequence. With the help of the Berlekamp-Massey algorithm, one can compute its characteristic polynomial $p(x) = 1+x^2+x^4+x^5+x^6$*. Thus, the output of the algorithm would be* $(\gamma_0, \ldots, \gamma_6) = (1,0,1,0,1,1,1)$*, which corresponds to the coefficients used in Example 5.3.*

Coming to the end of this section, we give an estimation of the effort of fast algebraic attacks. We set $D_n := \mu_2(n,d)$, $D_{\iota \cdot r} := \mu_2(\iota \cdot r, d)$ and $D'_n := \mu_2(n,d')$. Observe that most steps are similar to our description of algebraic attacks in Algorithm 3.4. Instead of looking for $\boldsymbol{Z}$-functions, one is now interested in r-functions which can be split as displayed in (5.1.3). As far as we know, no algorithm especially adapted to this problem has been proposed yet. The best thing one can do is to look for r-functions in general and, once they are found, check if they meet condition (5.1.3). For simplicity, we assume that this is successful and estimate the same effort as in conventional algebraic attacks.

Observe that the splitting of the r-function, the computation of coefficients γ_i and the reduction of the degree applies to *one* r-function. In other words, if an attacker has several r-functions $F^{(i)}$ which can all be split as given in (5.1.3) at his disposal, and if he wants to use all of them for an algebraic attack, then he needs to compute $\min(G_t^{(i)}(X))$ for each of them separately. Likewise, the computation of the functions $(F^{(i)})^*$ has to be done for each r-function $F^{(i)}$. Altogether, he can reduce the number of known keystream bits by a constant factor, but increases the efforts for the steps by the same factor. For simplicity, we assume in our effort estimations that only one r-function is involved.

Once a suitable r-function is found, the next step is to find $\min(G_t(X))$. Using Algorithm 5.2 requires to generate $2T$ keystream bits and $\mathcal{O}(T^2)$ operations.[1]

The next step is to generate a system of equations with the functions F^*. As $\deg(F^*) = d'$, linearization in the final step requires about D'_n equations to solve the system of equations. Recall that

$$F^*(K \cdot L^t, z_t, \ldots, z_{t+T+r-1}) = \sum_{i=0}^{T} \gamma_i \cdot H_t(K, z_{t+i}, \ldots, z_{t+i+r-1}) \ .$$

[1] In the next sections, we will encounter better methods that apply to special cases.

What is the effort to set up this system of equations? First of all, we take a look at generating *one* equation $F^*(\ldots) = 0$. Two different approaches are imaginable. One could either first insert the keystream bits into the T different functions H, which would yield $H_t(K, \boldsymbol{z_t, \ldots, z_{t+r-1}})$,$\ldots$, $H_{t+T}(K, \boldsymbol{z_{t+T}, \ldots, z_{t+T+r-1}})$, and then compute the linear sum. Or, one could do it the other way around: first compute the function

$$\sum_{i=0}^{T} \gamma_i H_{t+i}(K, z_{t+i}, \ldots, z_{t+i+r-1})$$

and then substitute the values $\boldsymbol{z_t, \ldots, z_{t+T+r-1}}$. However, the second way requires to initially deal with additional $T+r$ variables $z_t, \ldots, z_{t+T+r-1}$. Thus, from our perspective the first method is preferable.

Now, the effort to compute $F^* = \sum_{i=0}^{T} \gamma_i H_{t+i}(K, \boldsymbol{z_{t+i}, \ldots, z_{t+i+r-1}})$ requires the addition of at most T functions $H_{t+i}(\ldots)$, where each has at most D'_n monomials. Thus, the effort to compute one equation with F^* is in $\mathcal{O}(T \cdot D'_n)$. As we require about D'_n linearly independent equations for applying linearization afterwards, the overall effort is in $\mathcal{O}(D'^2_n \cdot T)$. As explained before, T is upper bounded by D_n, giving a total effort in $\mathcal{O}(D'^2_n \cdot D_n)$. Because of $D_n = \sum_{i=0}^{d} \binom{n}{i} \in \mathcal{O}(n^d)$, the effort can be equivalently described by $\mathcal{O}(n^{2d'+d})$.

The effort of the substitution step has been completely ignored in the first papers on fast algebraic attacks. Thus, the authors overlooked that in some cases the effort of the substitution step dominates the whole attack. This has been observed in [HawR04], where also a solution was presented. However, this solution relies on the condition that an attacker knows large parts of successive keystream bits, preferably $T + r$ bits. We abbreviate $F^*(K \cdot L^t, z_t, \ldots, z_{t+T+r-1})$ by F^*_t. The solution relies on the following two observations:

1. Each F^*_t is computed by building the sum with the same coefficients γ_i.
2. In the computations of successive functions F^*_t the same expressions $H_{t+i}(K, \boldsymbol{z_{t+i}, \ldots, z_{t+i+r-1}})$ are involved.

The solution they developed was to first transform the sequence $(H_t(K))$ into a new sequence with the fast Fourier transform [CooT65] and then apply suitable operations (depending on γ_i) on this new sequence to finally get the sequence (F^*_t). In the best case, when an attacker knows the values of $T + r$ successive keystream bits, the time effort to compute the D'_n equations $F^*_t = 0$ is reduced from $\mathcal{O}(D'^2_n \cdot D_n)$ to $\mathcal{O}(D'_n \cdot D_n \cdot \log_2(D_n))$. But we want to point out that the advantage is directly proportional to the length of the known keystream subsequence. In the worst case, when an attacker

Step	Time	Space	Keystream
Finding r-functions	$\mathcal{O}(2^{m+\iota \cdot r} \cdot D_{\iota \cdot r}^2)$	$\mathcal{O}(2^{m+(\iota-1) \cdot r} \cdot D_{\iota \cdot r})$	-
Computing $\min(G_t(X))$	$\mathcal{O}(D_n^2)$	$\mathcal{O}(D_n)$	-
Setting up linearized system of eqs with F^*	$\mathcal{O}(D'_n \cdot D_n \cdot \log_2(D_n))$	$\mathcal{O}(D'^2_n)$	$\geq D'_n + r$ succ.
Solving the system	$\mathcal{O}(D'^{\omega}_n)$	$\mathcal{O}(D'^2_n)$	-

Table 5.1.1: Effort estimations for a fast algebraic attack on a (ι, m)-combiner with keysize n, based on r-functions and linearization

knows only T distinct parts $(z_t, \ldots, z_{t+r-1})$ of the keystream bits, one has to fall back on simple substitution. But in all other cases, the methods from [HawR04] represent a tremendous improvement compared to simple substitution.

Once the system of equations of degree d' with $D'_n - 1$ linearly independent equations is build, the solution is computed within $\mathcal{O}(D'^{\omega}_n)$ operations. As $D'_n \in \mathcal{O}(n^{d'})$, the effort of the final step is reduced from $\mathcal{O}(n^{d \cdot \omega})$ to $\mathcal{O}(n^{d' \cdot \omega})$. The efforts for the whole attack are summed up in Table 5.1.1.

Fast algebraic attacks have outmatched some algebraic attacks, which had previously been the fastest attacks [Cou03]. Table 5.1.2 gives an overview of the estimations from [Cou03, HawR04].

In the following sections, we describe several improvements of fast algebraic attacks. First, we discuss the correctness of Algorithm 5.2 and how it can be improved by using the knowledge of G and L [Arm04a]. Then, we present a recent modification of Algorithm 5.2 which minimizes the value of T [ArmA05].

Cipher	Toyocrypt		
Attack	[MihH02]	Alg. att. [CouM03]	Fast alg. att. [Cou03]
Time	$\text{const} \cdot 2^{70}$	$\text{const} \cdot 2^{55}$	$\text{const} \cdot 2^{26}$
Memory	$\text{const} \cdot 2^{48}$	$\text{const} \cdot 2^{37}$	$\text{const} \cdot 2^{14}$
Keystream	$\text{const} \cdot 2^{48}$	$\text{const} \cdot 2^{18}$	$\text{const} \cdot 2^{18}$
Cipher	LILI-128		
Attack	[Saa02]	Alg. att. [CouM03]	Fast alg. att. [Cou03]
Time	$\text{const} \cdot 2^{62}$	$\text{const} \cdot 2^{63}$	$\text{const} \cdot 2^{37}$
Memory	$\text{const} \cdot 2^{51}$	$\text{const} \cdot 2^{43}$	$\text{const} \cdot 2^{24}$
Keystream	$\text{const} \cdot 2^{46}$	$\text{const} \cdot 2^{57}$	$\text{const} \cdot 2^{60}$
Cipher	E_0		
Attack	[FluL01]	Alg. att. [ArmK03]	Fast alg. att. [Cou03]
Time	$\text{const} \cdot 2^{73}$	$\text{const} \cdot 2^{68}$	$\text{const} \cdot 2^{55}$
Memory	≈ 10638	$\text{const} \cdot 2^{48}$	$\text{const} \cdot 2^{37}$
Keystream	$\text{const} \cdot 2^{43}$	$\text{const} \cdot 2^{23}$	$\text{const} \cdot 2^{24}$

Table 5.1.2: Comparison of (fast) algebraic attacks and the previously fastest attacks

5.2 On the precomputation step

As explained in Section 5.1, fast algebraic attacks are based on building new equations with a lower degree. This is achieved by computing linear combinations of the equations from the system of equations. For this, one needs a linear combination that cancels all these parts which have the degree d. This linear combination is derived by Algorithm 5.2 in a precomputation step.

However, Algorithm 5.2 is only correct if

$$\sum_{i=0}^{T} \gamma_i \cdot G(\boldsymbol{X} \cdot L^i) = 0 \in \mathbb{F} \quad \Rightarrow \quad \sum_{i=0}^{T} \gamma_i \cdot G(X \cdot L^i) \equiv 0 \in \mathbb{F}_2[X]$$

for the chosen value $\boldsymbol{X}$. In other words, $\min(G_t(\boldsymbol{X}))$ is equal to (or at least a multiple of) $\min(G_t(X))$. This has not been proved in [Cou03]. The only correctness argument that was indicated there was based on the assumption that the sequences produced by the LFSRs have pairwise co-prime periods. But this is not true in general. A counter-example is the keystream generator E_0 used in the Bluetooth standard for wireless communication. Two of the employed LFSRs are of lengths 33 and 39 (see also Section 2.5.4) with primitive minimal polynomials. Thus, by Theorem 2.58, both produce maximum sequences with periods $2^{33} - 1$ and $2^{39} - 1$, respectively, which share the common factor 7. This raises the question if Algorithms 5.2 gives the correct result under weaker conditions.

Anyhow, one can show that Algorithm 5.2 would not work correctly in general without any preconditions. An example for which it fails is the following:

Example 5.6. *Consider the case of two LFSRs $\mathcal{A}$ and $\mathcal{B}$ with the primitive minimal polynomials $m_a(x) = 1 + x + x^2$ and $m_b(x) = 1 + x + x^4$. Notice that the polynomials are co-prime but the corresponding periods $2^2 - 1 = 3$ and $2^4 - 1 = 15$ are not. As usual, we denote with (a_t) the sequence produced by LFSR $\mathcal{A}$ and similarly (b_t) for $\mathcal{B}$. Thus, the initial states are (a_0, a_1) and (b_0, b_1, b_2, b_3), respectively. Let*

$$G_t(a_0, a_1, b_0, b_1, b_2, b_3) := a_t + b_t + b_t b_{t+1} + b_t b_{t+1} b_{t+2}.$$

We define $\boldsymbol{X} := (1, 1, 1, 1, 1, 1)$ and $\boldsymbol{X'} := (1, 1, 1, 1, 1, 0)$. It holds that

$$\begin{aligned} \min(G_t(\boldsymbol{X})) &= 1 + x^5 + x^{10} \quad \text{and} \\ \min(G_t(\boldsymbol{X'})) &= 1 + x + x^3 + x^4 + x^5 + x^7 + x^8 \end{aligned}$$

Thus, the output of Algorithm 5.2 is not unique but depends on the chosen values in $\mathbb{F}_2^6$. As on the other hand $\min(G_t(X))$ is unique, at least one of the two has to be wrong.[2]

This raises the question which preconditions are sufficient for the correctness of Algorithm 5.2 and whether they are fulfilled for the case of E_0.

Furthermore, in Algorithm 5.2, the knowledge of G and the matrix L is only used to generate a linear recurring sequence on which the Berlekamp-Massey algorithm is applied, but not for the computation of $\min(G_t(X))$.

The aims of this section are twofold and represent our results from [Arm04a]. After collecting some facts on LFSRs, we derive a new (weaker) sufficient criterion for the correctness of the precomputation step. As this weaker precondition is met in the case of E_0, this proves the applicability of Algorithm 5.2 for a fast algebraic attacks on E_0.

Afterwards, we will discuss how the information on G and L can be used to adapt the Berlekamp-Massey algorithm to improve the efficiency of the precomputation step.

5.2.1 A new sufficient criterion for the correctness

In this section, we derive a sufficient condition for the correctness of Algorithm 5.2. This new condition is weaker than the recommendation given in [Cou03] and is satisfied in the case of E_0. Recall that the task of Algorithm 5.2 is to find $\min(G_t(X)) = \sum_{i=0}^{T} \gamma_i x^i$, i.e., coefficients $\gamma_0, \ldots, \gamma_T \in \mathbb{F}_2$ such that

$$\sum_{i=0}^{T} \gamma_i \cdot G_{t+i}(X) \equiv 0 \quad \forall t \geq 0 \tag{5.2.9}$$

with T being minimum. But, what the algorithm provides is $\min(G_t(\boldsymbol{X})) = \sum_{i=0}^{T'} \gamma'_i x^i$ for a chosen value $\boldsymbol{X} \in \mathbb{F}_2^n$:

$$\sum_{i=0}^{T'} \gamma'_i \cdot G_{t+i}(\boldsymbol{X}) = 0 \quad \forall t \geq 0 \ . \tag{5.2.10}$$

As Equation 5.2.9 remains true if one substitutes X by concrete values, $\min(G_t(X))$ is obviously a characteristic polynomial for $(G_t(\boldsymbol{X}))$. Thus, the minimal polynomial $\min(G_t(X))$ of the sequence $(G_t(X))$ is a multiple of the minimal polynomial $\min((G_t(\boldsymbol{X})))$ of the sequence $(G_t(\boldsymbol{X}))$ but not necessarily equal to it. In this context, recall Example 5.6 where , depending on the

[2] Of course, the other result could be a characteristic polynomial of the sequence $(G_t(X))$ which could likewise be used for a fast algebraic attack. But it is not $\min(G_t(X))$.

chosen start value $\boldsymbol{X}$, two different minimal polynomials

$$\begin{aligned} 1 + x^5 + x^{10} &= (1 + x + x^2) \cdot (1 + x + x^4) \cdot (1 + x^3 + x^4) \quad \text{and} \\ 1 + x + x^3 + x^4 + x^5 + x^7 + x^8 &= (1 + x + x^4) \cdot (1 + x^3 + x^4) \end{aligned}$$

have been the results, with the first being obviously a multiple of the second.

Indeed, one possible approach could be to compute the minimal polynomials $\min(G_t(\boldsymbol{X}))$ for different choices of $\boldsymbol{X}$ and then to take the least common multiple. However, without any further thoughts one would still not be sure whether one has found $\min(G_t(X))$ or only a divisor of it. But, if we knew that the minimal polynomials $\min(G_t(\boldsymbol{X}))$ are the same for *any* choice of $\boldsymbol{X}$, then it would follow that $\min(G_t(X)) = \min(G_t(\boldsymbol{X}))$, as the following proposition shows.

Proposition 5.7. *Let $G(X)$ be a Boolean function over $\mathbb{F}_2$ and L be a regular $n \times n$ matrix. If the minimal polynomials $\min(G_t(\boldsymbol{X}))$ are all the same for all $\boldsymbol{X} \in \mathbb{F}_2^n$, then $\min(G_t(X)) = \min(G_t(\boldsymbol{X}))$ for all $\boldsymbol{X}$.*

Proof. Let $\min(G_t(\boldsymbol{X})) = \sum_{i=0}^{T'} \gamma_i' x^i$ be the common minimal polynomial for all $\boldsymbol{X}$. This means that $\sum_{i=0}^{T'} \gamma_i' G_{t+i}(\boldsymbol{X}) = 0$ for all $\boldsymbol{X} \in \mathbb{F}_2^n$ and in particular $\sum_{i=0}^{T'} \gamma_i' G_{t+i}(X) \equiv 0$. Hence, $\min(G_t(\boldsymbol{X})) = \sum_{i=0}^{T'} \gamma_i' x^i$ is a multiple of $\min(G_t(X)) = \sum_{i=0}^{T} \gamma_i x^i$. On the other hand, we have already discussed that $\min(G_t(X))$ is also a multiple of $\min(G_t(\boldsymbol{X}))$, which shows the equality. □

Motivated by this observation, we will next address the question how one can be sure that the minimal polynomials $\min(G_t(\boldsymbol{X}))$ are the same for all $\boldsymbol{X}$.

For this, we need some more facts on linear recurring sequences and LFSRs. In the following, $\mathcal{S} = (s_i)_{i \geq 0}$ will denote a linear recurring sequence and $\min(\mathcal{S})$ its minimal polynomial. The next theorems are on the interplay between the roots of $\min(\mathcal{S})$ and the elements in $\mathcal{S}$.

Theorem 5.8. *[LidN86, Theorem 2.14] Let $f(x) \in \mathbb{F}_2[x]$ be an irreducible polynomial of degree d, then f has a root α in $\mathbb{F}_{2^d}$. Furthermore, all the roots of f are given by the distinct elements $\alpha, \alpha^2, \alpha^{2^2}, \ldots, \alpha^{2^{d-1}}$.*

Theorem 5.9. *Let $\mathcal{S} = (s_t)$ be a sequence over $\mathbb{F}_2$ with characteristic polynomial $f(x) = \prod_{i=1}^{d}(x - \alpha_i)$ where the roots lie in $\mathbb{F}_{2^d}^*$. If the roots $\alpha_1, \ldots, \alpha_d$ are all distinct, i.e., each root has multiplicity one, then the elements s_i can be expressed in the following way:*

$$s_i = \sum_{l=1}^{d} c_{\alpha_l} \cdot \alpha_l^i \tag{5.2.11}$$

where $c_{a_1}, \ldots, c_{a_d} \in \mathbb{F}_{2^d}$ are uniquely determined by the initial values of the sequence $\mathcal{S}$.

Proof. (See also [LidN86], Theorem 6.21) As the roots are all distinct and non-zero, the matrix

$$M := \begin{pmatrix} 1 & \ldots & 1 \\ \alpha_1 & \ldots & \alpha_d \\ \vdots & & \vdots \\ \alpha_1^{d-1} & \ldots & \alpha_d^{d-1} \end{pmatrix}$$

is a Vandermonde-matrix over $\mathbb{F}_{2^d}$ and therefore of full rank. Thus, for any $V_S := (s_0, \ldots, s_{d-1}) \in \mathbb{F}_{2^d}^d$, there exists a unique vector $V_C := (c_{\alpha_1}, \ldots, c_{\alpha_d}) \in \mathbb{F}_{2^d}^d$ such that $M \cdot V_C = V_S$. This shows that the first d elements $s_0, \ldots, s_{d-1}$ of $\mathcal{S}$ can be generated as showed in (5.2.11).

Now, let (5.2.11) be true for some elements $s_0, \ldots, s_{t+d-1}$ with $t \geq 0$ and let $f(x) = \gamma_0 + \gamma_1 \cdot x + \ldots + \gamma_{d-1} \cdot x^{d-1} + x^d$. First of all, as α_i is a root of f, it holds that

$$\begin{aligned} 0 = f(\alpha_i) &= \gamma_0 + \gamma_1 \cdot \alpha_i + \ldots + \gamma_{d-1} \cdot \alpha_i^{d-1} + \alpha_i^d \\ &\Leftrightarrow \\ \alpha^d &= \gamma_0 + \gamma_1 \cdot \alpha_i + \ldots + \gamma_{d-1} \cdot \alpha_i^{d-1} \end{aligned}$$

for any $1 \leq i \leq d$. Second, as $f(x)$ is a characteristic polynomial of $\mathcal{S}$, this leads to

$$\begin{aligned} s_{t+d} &= \sum_{i=0}^{d-1} \gamma_i \cdot s_{t+i} = \sum_{i=0}^{d-1} \gamma_i \left(\sum_{l=1}^{d} c_{\alpha_l} \cdot \alpha_l^{t+i} \right) \\ &= \sum_{l=1}^{d} c_{\alpha_l} \cdot \alpha_l^t \cdot \underbrace{\left(\sum_{i=0}^{d-1} \gamma_i \alpha_l^i \right)}_{=\alpha_l^d} = \sum_{l=1}^{d} c_{\alpha_l} \cdot \alpha_l^{t+d} \quad . \end{aligned}$$

□

As the sequence $(G_t(\boldsymbol{X}))$ is in fact a combination of products and sums of several linear recurring sequences, and as each of them can be expressed by the roots of the corresponding polynomials, we will next consider the question how to embed these operations into extension fields of $\mathbb{F}_2$.

Theorem 5.10. *[LidN86, Theorem 2.6] Let $\mathbb{F}_q$ be a subfield of the finite field $\mathbb{F}_{2^n}$. Then, it holds that $q = 2^m$ where m is a positive divisor of n.*

Corollary 5.11. *Let $n, m \geq 0$ be two integers. Then, $\mathbb{F}_{2^{\mathrm{lcm}(n,m)}}$ is the smallest extension field of $\mathbb{F}_{2^n}$ and $\mathbb{F}_{2^m}$.*

Proof. Similar to the proof of Theorem 2.38, one can either construct $\mathbb{F}_{2^{\mathrm{lcm}(n,m)}}$ by $\mathbb{F}_{2^n}[\alpha,\ldots,\alpha^{d-1}]$ with α a root of an irreducible polynomial $p(x)$ of degree $d = \mathrm{lcm}(n,m)/n$ or by $\mathbb{F}_{2^m}[\beta,\ldots,\beta^e]$ with β being a root of an irreducible polynomial $q(x)$ of degree $e = \mathrm{lcm}(n,m)/m$. Thereby, it is known that such polynomials $p(x)$ and $q(x)$ always exist (e.g., see [LidN86]). Thus, both fields $\mathbb{F}_{2^n}$ and $\mathbb{F}_{2^m}$ have at least one common extension field.

Assume that $\mathbb{E}$ is another common extension field such that $\mathbb{E} \subseteq \mathbb{F}_{2^{\mathrm{lcm}(n,m)}}$. As $\mathbb{E}$ is finite and has the same characteristic, namely 2, it must hold that $\mathbb{E} = \mathbb{F}_{2^r}$. Furthermore, it follows by Theorem 5.10 that n and m both divide r. Thus, r is a multiple of $\mathrm{lcm}(n,m)$ which shows that $\mathbb{F}_{2^{\mathrm{lcm}(n,m)}}$ is the smallest common extension field. □

Definition 5.12. *Let $n, m \geq 0$ be integers, $\alpha \in \mathbb{F}_{2^n}$ and $\beta \in \mathbb{F}_{2^m}$. Then, by $\alpha+\beta$ and $\alpha \cdot \beta$, we denote the corresponding sum resp. product in the extension field $\mathbb{F}_{2^{\mathrm{lcm}(n,m)}}$.*

*For polynomials $p_1(x),\ldots,p_r(x) \in \mathbb{F}_2[x]$, we define their **splitting field** $\mathbb{F}$ to be the smallest extension field of $\mathbb{F}_2$ such that each root of every $p_i(x)$ is an element of $\mathbb{F}$.*

Coming back to our original question, we can restate the situation as follows: We are given a Boolean function $G(x_1,\ldots,x_n)$ and r different LFSRs of lengths $\ell_1,\ldots,\ell_r$. Computing a sequence $(G_t(\boldsymbol{X}))$ for a chosen value $\boldsymbol{X} \in \mathbb{F}_2^n$ is equivalent to initializing the r LFSRs with some initial states $\boldsymbol{S_0^{(j)}} := (\boldsymbol{s_0^{(j)}},\ldots,\boldsymbol{s_{\ell_j-1}^{(j)}}) \in \mathbb{F}_2^{\ell_j}$ with $1 \leq j \leq r$ and computing the sequence

$$(G(\underbrace{\boldsymbol{s_i^{(1)}},\ldots,\boldsymbol{s_{i+\ell_1-1}^{(1)}}}_{=\boldsymbol{S_i^{(1)}}},\ldots,\underbrace{\boldsymbol{s_i^{(r)}},\ldots,\boldsymbol{s_{i+\ell_r-1}^{(r)}}}_{=\boldsymbol{S_i^{(r)}}}))_{i\geq 0}. \tag{5.2.12}$$

By Theorem 5.9, each of the r sequences can be expressed by

$$\boldsymbol{s_i^{(j)}} = \sum_{l=1}^{\ell_j} \boldsymbol{c_{a_{j,l}}} \cdot \alpha_{j,l}^i \ . \tag{5.2.13}$$

where $m_j(x) = \sum_{l=1}^{\ell_j}(x-\alpha_{j,l})$ is the minimal polynomial of the j^{th} LFSR. Now, inserting (5.2.13) into (5.2.12) yields

$$G_i(\boldsymbol{X}) = G(\sum_{l=1}^{\ell_1} \boldsymbol{c_{a_{1,l}}} \cdot \alpha_{1,l}^i,\ldots,\sum_{l=1}^{\ell_r} \boldsymbol{c_{a_{r,l}}} \cdot \alpha_{r,l}^{i+\ell_r-1}) = \sum_{\pi\in\Pi\subseteq\mathbb{F}_{2^{\mathrm{lcm}(\ell_1,\ldots,\ell_r)}}} \boldsymbol{c_\pi} \cdot \pi^i \tag{5.2.14}$$

with π being the occurring products $\alpha_{j_1,i_1}\cdots\alpha_{j_k,i_k}$ in $\mathbb{F}_{2^{\mathrm{lcm}(\ell_1,\ldots,\ell_r)}}$.

Example 5.13. *Recall Example 5.3 where two LFSRs $\mathcal{A}$ and $\mathcal{B}$ with their sequences (a_t) respectively (b_t) are involved. In demand were to find $\min(G_t(K))$ with $K = (a_0, a_1, b_0, b_1, b_2)$ and $G_t(K) = (a_t \cdot b_t)$.*

Let $m_a(x) = (x - \alpha_1) \cdot (x - \alpha_2)$ and $m_b(x) = (x - \beta_1) \cdot (x - \beta_2) \cdot (x - \beta_3)$ be the two minimal polynomials, then $a_t = c_{\alpha_1}\alpha_1^t + c_{\alpha_2}\alpha_2^t$ and $b_t = c_{\beta_1}\beta_1^t + c_{\beta_2}\beta_2^t + c_{\beta_3}\beta_3^t$. It follows that the sequence is expressible by

$$G_t(K) = a_t \cdot b_t = \sum_{1 \le i \le 2} \sum_{1 \le j \le 3} c_{\alpha_i} \cdot c_{\beta_j} \alpha_i^t \cdot \beta_i^t \ .$$

Thus, in this case, Π would be

$$\Pi = \{\alpha_1\beta_1, \alpha_1\beta_2, \alpha_1\beta_3, \alpha_2\beta_1, \alpha_2\beta_2, \alpha_2\beta_3\} \subset \mathbb{F}_{2^6} = \mathbb{F}_{2^{\mathrm{lcm}(2,3)}}.$$

Before we proceed, we pause for a moment and recapitulate. Given a Boolean function G and a regular matrix L, we are interested in finding $\min(G_t(X))$. For this purpose, it was proposed in [Cou03] to fix a value $\boldsymbol{X} \in \mathbb{F}_2^n$ and to compute $\min(G_t(\boldsymbol{X}))$. However, as we have seen in Example 5.6, it may hold that $\min(G_t(\boldsymbol{X})) \neq \min((G_t(\boldsymbol{X}')))$ for $\boldsymbol{X} \neq \boldsymbol{X}'$. Thus, our goal is to derive sufficient conditions such that the minimal polynomial is the same for all choices of $\boldsymbol{X}$.

Let $\boldsymbol{X} \neq \boldsymbol{X}' \in \mathbb{F}_2^n$ and consider the sequences

$$(G_t(\boldsymbol{X})) = \left(\sum_{\pi \in \Pi} \boldsymbol{c_\pi} \cdot \pi^t \right) \quad \text{and} \quad (G_t(\boldsymbol{X}')) = \left(\sum_{\pi \in \Pi} \boldsymbol{c'_\pi} \cdot \pi^t \right) \ . \tag{5.2.15}$$

where some of the coefficients $\boldsymbol{c_\pi}$ and $\boldsymbol{c'_\pi}$ may be zero. Can we decide from these expressions if $\min((G_t(\boldsymbol{X}))) = \min((G_t(\boldsymbol{X}')))$ or not? The answer is yes and given by the following theorems:

Theorem 5.14. *Let $S \subseteq \mathbb{F}_{2^d}$. Then, there exists a unique non-trivial polynomial $m(x) \in \mathbb{F}_2[x]$ with the lowest degree such that $m(\alpha) = 0$ for all $\alpha \in S$. We denote this polynomial by $\min(S)$.*

Proof. Let $I \subseteq \mathbb{F}_2[x]$ be the set of all polynomials which are zero on S. It is easy to see that I is an ideal. Furthermore, by the field equation, it holds that $\alpha^{q^d-1} - 1 = 0$ for all $\alpha \in \mathbb{F}_{2^d}$. That is, α is a root of the polynomial $x^{q^d-1} - 1$, proving that $I \neq \langle 0 \rangle$. Finally, Theorem 2.29 yields that I is a principal ideal, generated by one unique polynomial, namely $\min(S)$. □

Theorem 5.15. *Let $\mathcal{S} = (s_t)$ be a sequence with $s_t = \sum_{\pi \in \Pi} c_\pi \pi^t$, $\Pi \subseteq \mathbb{F}_{2^d}$ and* non-zero *coefficients c_π. Then it holds that*

$$\min(\Pi) = \min(S). \tag{5.2.16}$$

In particular, each root of $min(\mathcal{S})$ has multiplicity one.[3]

Proof. We show that $f(x) \in \mathbb{F}_2[x]$ is a characteristic polynomial of $\mathcal{S}$ if and only if $f(\pi) = 0$ for all $\pi \in \Pi$. Thus, $m(x)$ is the characteristic polynomial with the lowest degree, which is the definition of $min(\mathcal{S})$. Let $f(x) = \sum_{i=0}^{T} \gamma_i x^i$ and $\Pi = \{\pi_1, \ldots, \pi_l\}$. Then for each t, we have

$$\sum_{i=0}^{T} \gamma_i s_{t+i} = \sum_{i=0}^{T} \gamma_i \left(\sum_{j=1}^{l} c_{\pi_j} \pi_j^{t+i} \right) = \sum_{j=1}^{l} \left(c_{\pi_j} \sum_{i=0}^{T} \gamma_i \pi_j^i \right) \pi_j^t = \sum_{j=1}^{l} \left(c_{\pi_j} f(\pi_j) \right) \pi_j^t \tag{5.2.17}$$

For $1 \leq j \leq l$ and $0 \leq t \leq l-1$, let $M := (\pi_j^t)$ be a Vandermonde-matrix of size $l \times l$. As the elements π_j are pairwise distinct, M is regular. Thus, (5.2.17) is equal to zero for each t if and only if $(c_1 f(\pi_1), \ldots, c_l f(\pi_n)) \in \mathbb{F}_2^l$ is an element of the kernel of M, i.e., $c_{\pi_j} f(\pi_j) = 0$. As the coefficients c_{π_j} were assumed to be non-zero, this is equivalent to $f(\pi_j) = 0$ for all i. □

The important point about the previous theorems is that, given a sequence $(\sum_\pi c_\pi \cdot \pi^t)$, the minimal polynomial is uniquely defined by the set $\{\pi : c_\pi \neq 0\}$. More precisely, it holds that

$$\min((\sum_{\pi} c_\pi \cdot \pi^t)) = \min(\{\pi : c_\pi \neq 0\}) \ . \tag{5.2.18}$$

Hereby, the actual values of c_π are completely unimportant. Coming back to the expressions in (5.2.15), this means that if we can guarantee that for each $\pi \in \Pi$ it holds that $c_\pi \neq 0$ if and only if $c'_\pi \neq 0$, then the sets $\{\pi : c_\pi \neq 0\}$ and $\{\pi : c'_\pi \neq 0\}$ are equal, implying the equality of the minimal polynomials. Thus, the whole proof strategy behind the sufficient condition for the correctness of Algorithm 5.2 can be summarized as follows:

$$\begin{aligned}
& \forall \boldsymbol{X} \neq \boldsymbol{X}' : \{\pi : c_\pi \neq 0\} = \{\pi : c'_\pi \neq 0\} \\
\Rightarrow \quad & \forall \boldsymbol{X} \neq \boldsymbol{X}' : \min(\{\pi : c_\pi \neq 0\}) = \min(\{\pi : c'_\pi \neq 0\}) \\
\overset{\text{Theo. 5.15}}{\Rightarrow} \quad & \forall \boldsymbol{X} \neq \boldsymbol{X}' : \min(G_t(\boldsymbol{X})) = \min(G_t(\boldsymbol{X}')) \\
\overset{\text{Prop. 5.7}}{\Rightarrow} \quad & \forall \boldsymbol{X} : \min(G_t(\boldsymbol{X})) = \min(G_t(X)) \\
\Rightarrow \quad & \text{Output of Algorithm 5.2 is correct.}
\end{aligned}$$

So, how can we know that the sets $\{\pi : c_\pi \neq 0\}$ are all equal? To answer this question, we first try to get a feeling why these sets can be different by considering again Example 5.6.

[3] Although we doubt that this Theorem is really new, we neither knew it before we developed it for [Arm04a] nor could we find it somewhere in open literature.

Example 5.16. *First, we recall Example 5.6. Involved are two LFSRs $\mathcal{A}$ and $\mathcal{B}$ with the primitive minimal polynomials $m_a(x) = 1 + x + x^2$ and $m_b(x) = 1 + x + x^4$ and initial states $A_0 \in \mathbb{F}_2^2 \setminus \{\overrightarrow{0}\}$ and $B_0 \in \mathbb{F}_2^4 \setminus \{\overrightarrow{0}\}$, respectively. G was defined by*

$$G_t(a_0, a_1, b_0, b_1, b_2, b_3) := a_t + b_t + b_t b_{t+1} + b_t b_{t+1} b_{t+2}.$$

Our goal is to explore further why $\min(G_t(A_0, B_0)) \neq \min(G_t(A'_0, B'_0))$ can happen for $(A_0, B_0) \neq (A'_0, B'_0)$. Let $\alpha_0, \alpha_1 \in \mathbb{F}_{2^2}$ be the roots of $m_a(x)$ and $\beta_0, \beta_1, \beta_2, \beta_3 \in \mathbb{F}_{2^4}$ those of $m_b(x)$. By Theorem 5.9, a_t resp. b_t can be expressed by

$$a_t = \sum_i c_{\alpha_i} \alpha_i^t \quad \text{and} \quad b_t = \sum_j c_{\beta_j} \beta_j^t \ .$$

As the minimal polynomials are assumed to be primitive, the states A_t resp. B_t reach all possible values in $\mathbb{F}_2^2 \setminus \{\overrightarrow{0}\}$ and $\mathbb{F}_2^4 \setminus \{\overrightarrow{0}\}$. In particular, there exist integers δ_a, δ_b such that $A_{\delta_a} = A'_0$ and $B_{\delta_b} = B'_0$. Consequently, the shifted sequences $(a'_t) = (a_{t+\delta_a})$ and $(b'_t) = (b_{t+\delta_b})$ can be expressed by

$$\begin{aligned} a_{t+\delta_a} &= \sum_i c_{\alpha_i} \alpha_i^{t+\delta_a} = \sum_i c_{\alpha_i} \alpha_i^{\delta_a} \alpha_i^t \\ b_{t+\delta_b} &= \sum_i c_{\beta_i} \beta_i^{t+\delta_b} = \sum_i c_{\beta_i} \beta_i^{\delta_b} \beta_i^t \ . \end{aligned}$$

This yields the following expressions for the sequences $(G_t(A_0, B_0))$ and $(G_t(A'_0, B'_0))$:

$$\begin{aligned} G_t(A_0, B_0) &= a_t + b_t + b_t b_{t+1} + b_t b_{t+1} b_{t+2} \\ &= \sum_i c_{\alpha_i} \alpha_i^t + \sum_i c_{\beta_i}^t \beta_i + \sum_{i,j} c_{\beta_i} c_{\beta_j} \beta_i^t \beta_j^{t+1} + \sum_{i,j,k} c_{\beta_i} c_{\beta_j} c_{\beta_k} \beta_i^t \beta_j^{t+1} \beta_k^{t+2} \\ &= \sum_{\pi \in \Pi} c_\pi \pi^t, \\ G_t(A'_0, B'_0) &= a_{t+\delta_a} + b_{t+\delta_b} + b_{t+\delta_b} b_{t+1+\delta_b} + b_{t+\delta_b} b_{t+1+\delta_b} b_{t+2+\delta_b} \\ &= \sum_i c_{\alpha_i} \alpha_i^{\delta_a} \alpha_i^t + \sum_i c_{\beta_i} \beta_i^{\delta_b} \beta_i^t + \sum_{i,j} c_{\beta_i} c_{\beta_j} \beta_i^{\delta_b} \beta_j^{\delta_b} \beta_i^t \beta_j^{t+1} \\ &\quad + \sum_{i,j,k} c_{\beta_i} c_{\beta_j} c_{\beta_k} \beta_i^{\delta_b} \beta_j^{\delta_b} \beta_k^{\delta_b} \beta_i^t \beta_j^{t+1} \beta_k^{t+2} \\ &= \sum_{\pi \in \Pi} c'_\pi \pi^t \end{aligned}$$

where

$$\Pi = \underbrace{\{\alpha_0, \alpha_1\}}_{=:\Pi_A} \cup \underbrace{\{\beta_i | 1 \leq i \leq 4\} \cup \{\beta_i \beta_j | 1 \leq i < j \leq 4\} \cup \{\beta_i \beta_j \beta_k | 1 \leq i < j < k \leq 4\}}_{=:\Pi_B}.$$

As the minimal polynomials are primitive, 0 is not an element in Π*. Therefore* Π_B *consists of* $\binom{4}{1}+\binom{4}{2}+\binom{4}{3} = 14$ *elements in* $\mathbb{F}_{2^4}\setminus\{0\}$*. Because of* $|\mathbb{F}_{2^4}\setminus\{0\}| = 15$*,* Π_B *covers almost the whole set. As* Π_A *contains two distinct elements from* $\mathbb{F}_{2^2}\setminus\{0\} \subset \mathbb{F}_{2^4}\setminus\{0\}$*, there exists at least one element in* Π_A *which is an element in* Π_B *too.*

Let $\pi \in \Pi_A \cap \Pi_B$*. Then there exist two functions* C_A *and* C_B *such that*

$$\begin{aligned} c_\pi &= C_A(c_{\alpha_0}, c_{\alpha_1}) + C_B(c_{\beta_0}, c_{\beta_1}, c_{\beta_2}, c_{\beta_3}) \quad \text{and} \\ c'_\pi &= C_A(c_{\alpha_0}\alpha_0^{\delta_a}, c_{\alpha_1}\alpha_1^{\delta_a}) + C_B(c_{\beta_0}\beta_0^{\delta_b}, c_{\beta_1}\beta_1^{\delta_b}, c_{\beta_2}\beta_2^{\delta_b}, c_{\beta_3}\beta_3^{\delta_b}). \end{aligned}$$

Hence, depending on C_A*,* C_B*,* δ_a*, and* δ_b *it might be that one of the two coefficients* c_π *and* c'_π *is zero whereas the other one is not.*

We have seen that if one of the products π can be expressed in more than one way, it might happen that c_π is zero for some initial states but non-zero for others. This can be avoided if we premise some kind of "unique factorization" for the elements in Π. This is specified in the following definition:

Definition 5.17. *Let* $R^{(1)}, \ldots, R^{(r)} \subseteq \mathbb{F}_{2^d}$ *be pairwise disjunct,* $1 \le i_1, \ldots, i_n \le r$ *and* $R := R^{(i_1)} \times \ldots \times R^{(i_n)}$*. As usual, we define for* $V = (v_1, \ldots, v_n) \in R$ *and* $E = (e_1, \ldots, e_n) \in \{0,1\}^n$ *the expression* $V^E := v_1^{e_1} \cdot \ldots \cdot v_n^{e_n}$*.*

We introduce now an equivalence relation $\equiv_R$ *on the expressions* V^E*:*

$$V^E \equiv_R W^{E'} \quad \Longleftrightarrow \quad \forall l \in \{1, \ldots, r\} : \prod_{j:i_j=l} v_j^{e_j} = \prod_{j:i_j=l} w_j^{e'_j}. \tag{5.2.19}$$

One can see easily that $V^E \equiv_R W^{E'}$ implies that

$$V^E = \prod_{l=1}^{r} \left(\prod_{j:i_j=l} v_j^{e_j} \right) \overset{V^E \equiv_R W^{E'}}{=} \prod_{l=1}^{r} \left(\prod_{j:i_j=l} w_j^{e_j} \right) = W^E,$$

but the notion is stronger than pure equality. It is additionally required that the parts of the products belonging to $R^{(i)}$ are equal as well.

Example 5.18. *Set* $R^{(1)} := \{\alpha, \alpha\beta\}$*,* $R^{(2)} := \{\beta\}$ *with* $\beta \neq 1$ *and* $R := R^{(1)} \times R^{(2)}$*. Let* $V := (v_1, v_2) := (\alpha, \beta)$ *and* $W := (w_1, w_2) := (\alpha\beta, \beta)$ *from* R*,* $E := (1,1)$ *and* $E' := (1,0)$*. Then, it holds that*

$$V^E = \alpha^1 \cdot \beta^1 = \alpha\beta = (\alpha\beta)^1\beta^0 = W^E.$$

Thus, V^E *and* $W^{E'}$ *are equal. But, because of* $v_1^{e_1} = \alpha^1 \neq (\alpha\beta)^1 = w_1^{e'_1}$*, we have* $V^E \not\equiv_R W^{E'}$*.*

Example 5.19. *Consider again Examples 5.6 and 5.16. Let* $R^{(1)} := \{\alpha_0, \alpha_1\} \subseteq \mathbb{F}_{2^2} \subseteq \mathbb{F}_{2^4}$ *and* $R^{(1)} := \{\beta_0, \beta_1, \beta_2, \beta_3\} \subseteq \mathbb{F}_{2^4}$*. We define* $R := R^{(1)} \times R^{(2)} \times R^{(2)} \times R^{(2)}$*. As we have seen in Example 5.16, there exist* $\pi \in \Pi_A \cap \Pi_B$ *with* $\pi = \alpha_i = \beta_r^{e'_2} \cdot \beta_s^{e'_3} \cdot \beta_t^{e'_4}$ *with* $(e'_2, e'_3, e'_4) \in \{0,1\}^3$*. Translating into the formalism from Definition 5.17, this shows the existence of* $V = (\alpha_i, v_2, v_3, v_4), W = (w_1, \beta_r, \beta_s, \beta_t) \in R$*, and* $E = (1,0,0,0), E' = (0, e'_2, e'_3, e'_4) \in \{0,1\}^4$ *such that* $V^E = \pi = W^{E'}$*. For* $V^E \equiv_R W^{E'}$ *being true, it must hold*

$$\alpha_i^1 = w_1^0 \quad \text{and} \quad v_2^0 \cdot v_3^0 \cdot v_4^0 = \beta_r^{e'_2} \cdot \beta_s^{e'_3} \cdot \beta_t^{e'_4}. \tag{5.2.20}$$

But this is certainly not true. Recall that $m_a(x)$ *was assumed to be primitive, hence* $\alpha_i \neq 1$*. Because of* $w_1^0 = 1$*, the first part of (5.2.20) is not fulfilled. Therefore,* $V^E \not\equiv_R W^{E'}$*.*

In the preceding example, we have seen that in the case where it *might happen* that c_π is non-zero only for some of all possible initial states, expressions V^E and $W^{E'}$ could be constructed such that $\pi = V^E = W^{E'}$ but $V^E \not\equiv_R W^{E'}$. The following theorem shows that this was no coincidence. More precisely, $V^E = W^{E'} \iff V^E \equiv_R W^{E'}$ is a sufficient condition to ensure $c_\pi \neq 0 \iff c'_\pi \neq 0$.

Theorem 5.20. *Let* $\mathcal{S}^{(1)} = (s_t^{(1)}), \ldots, \mathcal{S}^{(r)} = (s_t^{(r)})$ *be sequences with pairwise co-prime minimal polynomials which have only non-zero roots. Let* $R^{(i)}$ *denote the set of roots of* $min(\mathcal{S}^{(i)})$*,* $G : \mathbb{F}_2^n \to \mathbb{F}_2$ *a Boolean function,* $i_1, \ldots, i_n \in \{1, \ldots, r\}$*,* $R := R^{(i_1)} \times \ldots \times R^{(i_n)}$ *and* $d_{i_1}, \ldots, d_{i_n} \in \mathbb{N}$*.*

Furtheron, for arbitrary $\delta := (\delta_1, \ldots, \delta_r) \in \mathbb{N}^r$ *the sequences* $\mathcal{Z} := (z_t)$ *and* $\mathcal{Z}^{(\delta)} := (z_t^{(\delta)})$ *are defined by*

$$z_t := G(s_{t+d_{i_1}}^{(i_1)}, \ldots, s_{t+d_{i_n}}^{(i_n)}),\ z_t^{(\delta)} := G(s_{t+d_{i_1}+\delta_{i_1}}^{(i_1)}, \ldots, s_{t+d_{i_n}+\delta_{i_n}}^{(i_n)})$$

Let $G(X) = \sum_{E \in \mathcal{E} \subseteq \{0,1\}^n} X^E$ *be the algebraic normal form of* G*. If for all* $V, W \in R$ *and* $E, E' \in \mathcal{E}$ *it holds that*

$$V^E = W^{E'} \Rightarrow V^E \equiv_R W^{E'},$$

then $min(\mathcal{Z}) = min(\mathcal{Z}^{(\delta)})$ *for every choice of* δ*.*

Proof. By Theorem 5.15, all roots in $R^{(i)}$ have multiplicity one. Therefore, by Theorem 5.9, each sequence $\mathcal{S}^{(i)}$ can be expressed by $s_t^{(i)} = \sum_{\alpha \in R^{(i)}} c_\alpha \alpha^t$ with unique coefficients c_α. For each i, it holds that

$$s_{t+d_i}^{(i)} = \sum_{\alpha \in R^{(i)}} c_\alpha \alpha^{t+d_i} = \sum_{\alpha \in R^{(i)}} \left(c_\alpha \alpha^{d_i}\right) \alpha^t$$

and therefore

$$\begin{aligned} z_t &= G(\sum_{\alpha\in R^{(i_1)}}(c_\alpha\alpha^{d_{i_1}})\alpha^t,\ldots,\sum_{\alpha\in R^{(i_n)}}(c_\alpha\alpha^{d_{i_n}})\alpha^t) \quad \text{and} \\ z_t^{(\delta)} &= G(\sum_{\alpha\in R^{(i_1)}}(c_\alpha\alpha^{d_{i_1}})\alpha^{t+\delta_{i_1}},\ldots,\sum_{\alpha\in R^{(i_n)}}(c_\alpha\alpha^{d_{i_n}})\alpha^{t+\delta_{i_n}}) \end{aligned}$$

We set $\Pi := \{(\alpha_1,\ldots,\alpha_n)^E | (\alpha_1,\ldots,\alpha_n) \in R, E \in \mathcal{E}\}$. The sequences $\mathcal{Z}$ and $\mathcal{Z}^{(\delta)}$ can be expressed by

$$z_t = \sum_{\pi\in\Pi} c_\pi \pi^t, \quad z_t^{(\delta)} = \sum_{\pi\in\Pi} c_\pi^{(\delta)} \pi^t$$

with unique coefficients c_π and $c_\pi^{(\delta)}$. We show that c_π is non-zero if and only if $c_\pi^{(\delta)}$ is non-zero. Then the equality of $min(\mathcal{Z})$ and $min(\mathcal{Z}^{(\delta)})$ follows by Theorem 5.15.

We express the coefficients c_π and $c_\pi^{(\delta)}$ in dependence of the coefficients c_α and show next that both differ only by a non-zero factor which depends on π and δ.

For $V^E = (\alpha_1,\ldots,\alpha_n)^E = \pi \in \Pi$, we define by π^δ the product

$$(\prod_{\alpha_i\in R^{(1)}} \alpha_i^{e_i})^{\delta_1} \cdot \ldots \cdot (\prod_{\alpha_i\in R^{(r)}} \alpha_i^{e_i})^{\delta_r}.$$

π^δ is well defined. Assume that $V^E = (\alpha_1,\ldots,\alpha_n)^E = \pi = (\beta_1,\ldots,\beta_n)^{E'} = W^{E'}$. By assumption, it holds that $V^E \equiv_R W^{E'}$, which implies the following equations:

$$\begin{aligned} V^E \equiv_R W^{E'} &\Rightarrow \forall l \in \{1,\ldots,r\} : \prod_{\alpha_i\in R^{(l)}} \alpha_i^{e_i} = \prod_{\beta_i\in R^{(l)}} \beta_i^{e_i'} \\ &\Rightarrow \forall l \in \{1,\ldots,r\} : (\prod_{\alpha_i\in R^{(l)}} \alpha_i^{e_i})^{\delta_l} = (\prod_{\beta_i\in R^{(l)}} \beta_i^{e_i'})^{\delta_l} \\ &\Rightarrow (\prod_{\alpha_i\in R^{(1)}} \alpha_i^{e_i})^{\delta_1} \cdot \ldots \cdot (\prod_{\alpha_i\in R^{(r)}} \alpha_i^{e_i})^{\delta_r} = (\prod_{\beta_i\in R^{(1)}} \beta_i^{e_i'})^{\delta_1} \cdot \ldots \cdot (\prod_{\beta_i\in R^{(r)}} \beta_i^{e_i'})^{\delta_r}. \end{aligned}$$

This shows that the definition of π^δ is independent of the "factorization" of π.

As the final step we show that $c_\pi^{(\delta)} = c_\pi \cdot \pi^\delta$. As the roots of the minimal polynomials are all non-zero by assumption, it holds that $\pi^\delta \neq 0$ and in particular $\pi^\delta \neq 0$. Therefore, $c_\pi \neq 0$ if and only if $c_\pi^{(\delta)} \neq 0$. This concludes the proof.

To prove the claim given above, we define analogously to V^E for $V = (\alpha_1, \ldots, \alpha_n) \in R$ the expression $c_V^E := c_{\alpha_1}^{e_1} \cdots c_{\alpha_n}^{e_n}$. It holds that

$$z_t = \sum_{\pi \in \Pi} \underbrace{\left(\sum_{E \in \mathcal{E}} \sum_{\substack{V=(\alpha_1,\ldots,\alpha_n) \in R \\ V^E = \pi}} c_V^E \right)}_{=c_\pi} \cdot \underbrace{\left(\prod_{\alpha_i \in R^{(1)}} \alpha_i^{e_i} \right)^t \cdot \ldots \cdot \left(\prod_{\alpha_i \in R^{(r)}} \alpha_i^{e_i} \right)^t}_{=(V^E)^t = \pi^t}$$

Together with the observation on π^δ made above, it holds that

$$\begin{aligned} z_t^{(\delta)} &= \sum_{\pi \in \Pi} \underbrace{\left(\sum_{E \in \mathcal{E}} \sum_{\substack{V=(\alpha_1,\ldots,\alpha_n) \in R \\ V^E = \pi}} c_V^E \right)}_{=c_\pi} \cdot \left(\prod_{\alpha_i \in R^{(1)}} \alpha_i^{e_i} \right)^{t+\delta_1} \cdot \ldots \cdot \left(\prod_{\alpha_i \in R^{(r)}} \alpha_i^{e_i} \right)^{t+\delta_r} \\ &= \sum_{\pi \in \Pi} c_\pi \cdot \underbrace{\left(\prod_{\alpha_i \in R^{(1)}} \alpha_i^{e_i} \right)^{\delta_1} \cdot \ldots \cdot \left(\prod_{\alpha_i \in R^{(r)}} \alpha_i^{e_i} \right)^{\delta_r}}_{=\pi^\delta} \cdot \underbrace{\left(\prod_{\alpha_i \in R^{(1)}} \alpha_i^{e_i} \right)^t \cdot \ldots \cdot \left(\prod_{\alpha_i \in R^{(r)}} \alpha_i^{e_i} \right)^t}_{=\pi^t} \\ &= \sum_{\pi \in \Pi} c_\pi \cdot \pi^\delta \cdot \pi^t = \sum_{\pi \in \Pi} c_\pi^{(\delta)} \pi^t. \end{aligned}$$

This shows that $c_\pi^{(\delta)} = c_\pi \cdot \pi^\delta$ which concludes the proof. □

Thus, Algorithm 5.2 computes the right output, if the conditions of Theorem 5.37 are met by the corresponding Boolean function G and the involved LFSRs. In the case of E_0, we could verify that this is true, which shows that Algorithm 5.2 is applicable although the periods of the LFSRs are not co-prime.

Finally, we show that Theorem 5.20 applies to a large class of LFSR-based ciphers automatically, proving the correctness of Algorithm 5.2 in these cases. Let $m_1(x), \ldots, m_r(x) \in \mathbb{F}_2[x]$ be the primitive minimal polynomials of the used LFSRs such that the roots $R^{(i)}$ are all pairwise distinct and non-zero. For the following classes of ciphers, the assumptions of Theorem 5.20 are always satisfied:

1. The cipher is a filter generator. This is a $(\iota, 0)$-combiner using only one LFSR, i.e. $r = 1$.

2. The degrees of the minimal polynomials are pairwise co-prime.

Let $R^{(1)} \dot\cup \ldots \dot\cup R^{(r)} = \{\alpha_1, \ldots, \alpha_n\}$ be the set of all roots of the minimal polynomials and $\Pi := \{V^E = \prod_{i=1}^n \alpha_i^{e_i} | E \in \{0,1\}^n\}$. That is, Π consists of all possible products of roots. We show now that in both cases, $V^E = W^{E'}$

implies $V^E \equiv_R W^{E'}$ as well. As Π includes all possible products, this proves that the conditions of Theorem 5.20 are satisfied for any Boolean function G.

Let from now on $V^E = (\alpha_1, \ldots, \alpha_n)^{(e_1,\ldots,e_n)}$ and $W^{E'} = (\beta_1, \ldots, \beta_n)^{(e'_1,\ldots,e'_n)}$ be elements from Π with $V^E = W^{E'}$. In the first case, only one LFSR is involved, that is $r = 1$. Thus, the "factorization" of π is always unique:

$$\alpha_1^{e_1} \cdot \ldots \cdot \alpha_n^{e_n} = \beta_1^{e'_1} \cdot \ldots \cdot \beta_n^{e'_n} \overset{r=1}{\Leftrightarrow} \forall l \in \{1, \ldots, r\} : \prod_{\alpha_i \in R^{(l)}} \alpha_i^{e_i} = \prod_{\beta_j \in R^{(l)}} \beta_j^{e'_j} \ .$$

This concludes the first case.

For the second case, we remember the fact that $\mathbb{F}_{2^n} \subseteq \mathbb{F}_{2^m}$ iff n divides m (Theorem 5.10). In particular, $\mathbb{F}_{2^n} \cap \mathbb{F}_{2^m} = \mathbb{F}_{2^c}$ with $c := gcd(n, m)$. We denote by d_i the degree of the minimal polynomial $m_i(x)$. As $\mathbb{F}_{2^{d_i}}$ is a field, it holds for $\alpha \in R^{(i)} \subseteq \mathbb{F}_{2^{d_i}}$ that both α^{-1} and all multiple products are elements in $\mathbb{F}_{2^{d_i}}$. Let l be arbitrary with $1 \leq l \leq r$ and set $\hat{d}_l := d_1 \cdot \ldots \cdot d_{l-1} \cdot d_{l+1} \cdot \ldots \cdot d_r$. Then $\alpha_1^{e_1} \cdot \ldots \cdot \alpha_n^{e_n} = \beta_1^{e'_1} \cdot \ldots \cdot \beta_n^{e'_n}$ implies

$$\underbrace{\prod_{\alpha_i \in R^{(l)}} \alpha_i^{e_i} \prod_{\beta_j \in R^{(l)}} (\beta_j^{e'_j})^{-1}}_{\in \mathbb{F}_{2^{d_l}}} = \underbrace{\prod_{\alpha_i \notin R^{(l)}} \alpha_i^{e_i} \prod_{\beta_j \notin R^{(l)}} (\beta_j^{e'_j})^{-1}}_{\in \mathbb{F}_{2^{\hat{d}_l}}} =: \rho_l \ .$$

Therefore, $\rho_l \in \mathbb{F}_{2^{d_l}} \cap \mathbb{F}_{2^{\hat{d}_l}}$. By assumption the values d_i are pairwise co-prime. Hence, it is $gcd(d_l, \hat{d}_l) = 1$ and $\rho_l \in \mathbb{F}_{2^1} = \mathbb{F}_2$. As the roots are all non-zero, ρ_l equals to 1 for each choice of l. Thus, it holds for all l that

$$\prod_{\alpha_i \in R^{(l)}} \alpha_i^{e_i} \prod_{\beta_j \in R^{(l)}} (\beta_j^{e'_j})^{-1} = 1 \iff \prod_{\alpha_i \in R^{(l)}} \alpha_i^{e_i} = \prod_{\beta_j \in R^{(l)}} \beta_j^{e'_j} \ .$$

This concludes the second case.

5.2.2 An improved precomputation algorithm

In this section, we show how the search for $\min(G_t(X))$ can be improved. One advancement has been proposed in [HawR04]. The authors showed for the case of one LFSR with primitive minimal polynomial, that with the help of the Fast Fourier Transform one can efficiently construct a polynomial $\min(d) := \sum_{i=0}^{T'} \gamma'_i x^i$ such that

$$\sum_{i=0}^{T'} \gamma'_i G_{t+i}(X) \equiv 0$$

for all $t \geq 0$ and *all* Boolean functions G with $\deg(G) \leq d$. The advantage is that the time effort was estimated to be in

$$O(\mu_2(n,d) \cdot (n \log_2(n) + \log_2(\mu_2(n,d))^3)). \tag{5.2.21}$$

As discussed on page 132 it holds that $T \leq \mu_2(n,d)$. If we use the estimation $T \approx \mu_2(n,d)$, then this method is certainly faster than the $O(T + T^2)$ operations from Algorithm 5.2.[4] However, one has to pay the price that $T' = \deg(\min(d))$ might be higher than $T = \deg(\min(G))$. For example, the authors estimated that for the parameters of E_0, i.e., $n = 128$ and $d = 4$, that $T' = \deg(\min(d)) = 11,017,633$ whereas in [Arm04a] it was estimated that $T = \deg(\min(G)) = 8,822,188$ (see also the arguments later in the section, especially on pages 154f). In some cases, one might want to live with this drawback and prefer the faster algorithm from [HawR04]. But in other cases, it is certainly preferable to get $\min(G)$.

Therefore, we restate our results from [Arm04a] which show how Algorithm 5.2 can be improved. Observe that the knowledge of G and L is only exploited to compute the sequence $(G_t(\boldsymbol{X}))_{i \geq 0}$ but not to support the Berlekamp-Massey algorithm. The idea is to compute $\min(G) = \sum_{i=0}^{T} \gamma_i \cdot x^i$ and/or the parameter T more or less directly from the known minimal polynomials and G. For this purpose, we cite some statements about minimal polynomials.

Definition 5.21. *Consider two co-prime polynomials $f(x) = \prod_{i=1}^{n}(x - \alpha_i)$ and $g(x) = \prod_{j=1}^{m}(x - \beta_j)$ with no multiple roots. Then we define*

$$f(x) \otimes g(x) := \prod_{i,j}(x - \alpha_i\beta_j), \quad f(x) \otimes f(x) := \prod_{1 \leq i < j \leq n}(x - \alpha_i\alpha_j) \cdot f(x) \ .$$

Theorem 5.22. *Let $\mathcal{S}^{(1)} = (s_t^{(1)}), \ldots, \mathcal{S}^{(r)} = (s_t^{(r)})$ be sequences with pairwise co-prime $\min(\mathcal{S}^{(i)})$. Then*

$$\begin{aligned} \min(\mathcal{S}^{(1)} + \ldots + \mathcal{S}^{(r)}) &= \min(\mathcal{S}^{(1)}) \cdot \ldots \cdot \min(\mathcal{S}^{(r)}) \\ \min(\mathcal{S}^{(i)} \cdot \mathcal{S}^{(j)}) &= \min(\mathcal{S}^{(i)}) \otimes \min(\mathcal{S}^{(j)}), \quad \forall i \neq j \end{aligned}$$

where $\mathcal{S}^{(1)} + \ldots + \mathcal{S}^{(r)} := (s_t^{(1)} + \ldots + s_t^{(r)})_{t \geq 0}$ and $\mathcal{S}^{(i)} \cdot \mathcal{S}^{(j)} := (s_t^{(i)} \cdot s_t^{(j)})_{t \geq 0}$.

Proof. A proof can be found in [LidN86, Th. 6.57 + 6.67]. But with the statements developed so far, it is possible to show it directly.

[4] The authors claimed that the method can be extended for the case of several LFSRs, but gave neither details nor an estimation for the effort. However, as they used the same formula (5.2.21) to guess the effort for E_0, LILI-128 and Toyocrypt (see [HawR04, Table 2]), we assume that (5.2.21) is valid for the general case.

Let $R^{(i)}$ denote as usual the roots of the minimal polynomial of the sequence $\mathcal{S}^{(i)}$. Then, by Theorem 5.9, the elements of the sequences can be expressed by $s_t^{(i)} = \sum_{\alpha \in R^{(i)}} c_\alpha \alpha^t$ with at least one non-zero coefficient c_α per sequence. It follows that

$$\begin{aligned} s_t^{(1)} + \ldots + s_t^{(r)} &= \sum_{\alpha \in R^{(1)} \dot\cup \ldots \dot\cup R^{(r)}} c_\alpha \alpha^t \quad \text{and} \\ s_t^{(i)} \cdot s_t^{(j)} &= \sum_{\substack{\alpha \in R^{(i)} \\ \beta \in R^{(j)}}} c_\alpha c_\beta (\alpha\beta)^t \ . \end{aligned}$$

The rest follows by Theorem 5.15. □

Example 5.23. *We refer again to the toy cipher, especially Examples 5.3 and 5.5. Here, two LFSRs are involved with minimal polynomials* $m_a(x) = x^2 + x + 1 = (x - \alpha_1) \cdot (x - \alpha_2)$ *and* $m_b(x) = x^3 + x^2 + 1 = (x - \beta_1) \cdot (x - \beta_2) \cdot (x - \beta_3)$*. We denote the corresponding sequences by* $(a_t) = (c_{\alpha_1}\alpha_1^t + c_{\alpha_2}\alpha_2^t)$ *and* $(b_t) = (c_{\beta_1}\beta_1^t + c_{\beta_2}\beta_2^t + c_{\beta_3}\beta_3^t)$*. Thus, the product of these two sequences can be expressed by*

$$a_t \cdot b_t = \sum_{i=1}^{2} \sum_{j=1}^{3} c_{\alpha_i} c_{\beta_j} (\alpha_i \cdot \beta_j)^t \ .$$

With Theorem 5.15, it follows that

$$\min((a_t \cdot b_t)) = m_a(x) \otimes m_b(x) = 1 + x^2 + x^4 + x^5 + x^6 \ ,$$

which is exactly the minimal polynomial derived in Examples 5.3 and 5.5. However, instead of applying Algorithm 5.2, we computed $\min((a_t \cdot b_t))$ *directly from* $m_a(x)$ *and* $m_b(x)$*.*

Theorem 5.24. *([Key76, Th. 1]) Let* $\mathcal{S} = (s_t)$ *be a sequence and* $l := \deg(\min(\mathcal{S}))$*. If* d *is an integer with* $1 \leq d < l$*, then the sequence* $(s_t \cdot s_{t+d})$ *has the minimal polynomial* $\min(\mathcal{S}) \otimes \min(\mathcal{S})$ *of degree* $\frac{l(l+1)}{2}$*.*

Before we proceed further, we take a closer look at the complexities of the operators "$\cdot$" and "$\otimes$".

Theorem 5.25. *([Sch77]) Let two polynomials* $f(x), g(x) \in \mathbb{F}_2[x]$ *of degrees* $\leq m$ *be given. Then, the product* $f(x) \cdot g(x)$ *can be computed with an effort of* $\mathcal{O}(m \log m \log\log m)$ *operations over* $\mathbb{F}_2$*.*

Theorem 5.26. *(Bostan, Flajolet, Salvy, Schost [BosFSS06], Theorem 1) Let* $f(x)$ *resp.* $g(x)$ *be two co-prime polynomials of degree* n *resp.* m *with no*

multiple roots. Then the polynomial $f(x) \otimes g(x)$ can be computed directly within

$$\mathcal{O}(\underbrace{nm \log^2(nm/2) \log\log(nm/2) + nm \log(nm) \log\log(nm)}_{=:\mathcal{T}(nm)})$$

operations in $\mathbb{F}_2$ without requiring the knowledge of the roots of $f(x)$ or $g(x)$.

This implies a divide-and-conquer approach for computing $\min(G_t(X))$. The trick is to split G into two or more functions $G^{(1)}, \ldots, G^{(l)}$ such that the corresponding minimal polynomials $\min(G_t^{(i)}(X))$ are pairwise co-prime. This is for example trivially fulfilled if the functions G^i depend on the outputs of different LFSRs. By Theorem 5.22 it holds that

$$\min(G_t(X)) = \min(G_t^{(1)}(X)) \cdot \ldots \cdot \min(G_t^{(l)}(X)).$$

In some cases, such a partition can be hard to find or may not even exist. If the minimal polynomials $\min(G_t^{(i)}(X))$ are not pairwise co-prime, the product $p(x) := \min(G_t^{(1)}(X)) \cdot \ldots \cdot \min(G_t^{(l)}(X))$ is a characteristic polynomial of the sequence $(G_t(X))$. , i.e., the coefficients of $p(x)$ also fulfill equation (5.1.4). Therefore, the coefficients of $p(x)$ can likewise be used for a fast algebraic attack, although it may consume more known keystream bits than really necessary.

We compare the effort of this approach to that of Algorithm 5.2. For simplicity we assume $l = 2$, i.e. $G_t(X) = G_t^{(1)}(X) \cdot G_t^{(2)}(X)$ such that $\min(G_t(X)) = \min(G_t^{(1)}(X)) \cdot \min(G_t^{(2)}(X))$. Let $T_1 := \deg(\min(G^{(1)})) \leq \deg(\min(G^{(2)})) =: T_2$. Then $\deg(\min(G)) = T_1 + T_2$.

Algorithm 5.2 contains mainly two steps. The first is to compute the sequence $G_0(\boldsymbol{X}), \ldots, G_{2(T_1+T_2)-1}(\boldsymbol{X})$ for a chosen value $\boldsymbol{X} \in \mathbb{F}_2^n$. In general, the exact value of T is unknown but an upper bound is the maximum number of different monomials occurring. However, it is not necessary to know (or estimate) T in advance, as one needs to compute $G_t(\boldsymbol{X})$ only if it is required by the Berlekamp-Massey algorithm. As the effort to evaluate $G_t(\boldsymbol{X})$ is at least linear in the size of $\boldsymbol{X}$, i.e. n, we can give $\mathcal{O}(2(T_1 + T_2) \cdot n)$ as a lower bound for the first step. The second step, the Berlekamp-Massey algorithm, needs a number of operations in $\mathcal{O}((T_1+T_2)^2)$. Altogether, Algorithm 5.2 needs about

$$\mathcal{O}(T_1^2 + T_2^2 + 2(T_1 + T_2)n + 2T_1T_2))$$

basic operations. However, keep in mind that this is only a lower bound.

Instead of using Algorithm 5.2, we can do the following: First compute $\min(G_t^{(1)}(X))$ and $\min(G_t^{(2)}(X))$. In general, this can be done with Algorithm 5.2 or in some cases by using the $\otimes$-product (we will see details later). If we

use Algorithm 5.2, the complexity of these operations are $\mathcal{O}(T_1^2+2T_1\cdot n)$ resp. $\mathcal{O}(T_2^2 + 2T_2 \cdot n)$. Notice that both operations can be performed in parallel. Having computed $\min(G_t^{(1)}(X))$ and $\min(G_t^{(2)}(X))$, the second and final step consists of computing the product $\min(G_t^{(1)}(X)) \cdot \min(G_t^{(2)}(X)) = \min(G_t(X))$. By Theorem 5.25, this has an effort of $\mathcal{O}(T_2 \log T_2 \log\log T_2)$ which implies an overall effort of

$$\mathcal{O}(T_1^2 + T_2^2 + 2(T_1 + T_2)n + T_2 \log T_2 \log\log T_2) \ .$$

In general, it is $\log T_2 \log\log T_2 \ll 2T_1$. Thus, our new approach is faster than Algorithm 5.2. The advantage increases if G can be divided into more than two parts.

In some cases, the precomputation step can be improved even a bit further. Assume that at least one of the $G^{(i)}$ mentioned above can be written as a product $G^{(i)} = G^{(i,1)} \cdot G^{(i,2)}$ such that $\min(G_t^{(i,1)}(X))$ and $\min(G_t^{(i,2)}(X))$ are coprime. This is for example almost always the case when $G^{(i)}$ is a monomial. Then, by Theorem 5.22 it holds that $\min(G^{(i)}) = \min(G_t^{(i,1)}(X)) \otimes \min(G_t^{(i,2)}(X))$, which implies a similar strategy. Let again be

$$T_1 := \deg(\min(G_t^{(i,1)}(X))) \leq \deg(\min(G_t^{(i,2)}(X))) =: T_2.$$

Using Algorithm 5.2 would need about

$$\mathcal{O}(T_1^2 T_2^2 + 2T_1 T_2 n)$$

operations. Instead, we can we compute $\min(G_t^{(i,1)}(X))$ and $\min(G_t^{(i,2)}(X))$ with Algorithm 5.2 in a first step. This takes $\mathcal{O}(T_1^2 + T_2^2 + 2(T_1 + T_2)n)$ operations. If $\min(G_t^{(i,1)}(X))$ and/or $\min(G_t^{(i,2)}(X))$ are already known, then this step can be omitted. This is for example the case if $G^{(i)}$ is the product of the output of two or several distinct LFSRs (see the E_0 example later in this section). In the second step, we use the algorithm described in [BosFSS06] to compute $\min(G_t^{(i,1)}(X)) \otimes \min(G_t^{(i,2)}(X))$. The effort is $\mathcal{O}(\mathcal{T}(T_1 T_2))$, which is in $\mathcal{O}(T_1 T_2 \log^2(T_1 T_2) \log\log(T_1 T_2))$. Altogether, this approach needs

$$\mathcal{O}(T_1 T_2 \log^2(T_1 T_2) \log\log(T_1 T_2) + T_1^2 + T_2^2 + 2(T_1 + T_2)n)$$

operations. This shows the improvement. If we perform the operations of the first step in parallel, the time needed to get the result can be decreased further. In the following, we summarize our approaches by proposing the following new algorithm:

Algorithm 5.3.
Computation of $\min(G_t(X))$
Input: *A Boolean function* $G : \mathbb{F}_2^n \to \mathbb{F}_2$ *for which the inputs are taken from* r *LFSRs with minimal polynomials* $m^{(i)}(x)$
Output: *The polynomial* $\min(G_t(X))$

1: *Split* G *into a sum* $G = G^{(1)} + \ldots + G^{(l)}$ *such that the corresponding minimal polynomials* $\min(G_t^{(i)}(X))$ *are pairwise co-prime.*
2: ***if*** $G^{(i)} = G^{(i,1)} \cdot \ldots \cdot G^{(i,s)}$ *with co-prime* $\min(G_t^{(i,j)}(X))$ *(e.g.,* $G^{(i)}$ *is a monomial)* ***then***
3: *Compute* $\min(G_t^{(i)}(X)) = \min(G_t^{(i,1)}(X)) \otimes \ldots \otimes \min(G_t^{(i,s)}(X))$ *with the* $\otimes$*-algorithm described in [BosFSS06]*
4: ***end if***
5: *Compute* $\min(G_t(X)) = \min(G_t^{(1)}(X)) \cdot \ldots \cdot \min(G_t^{(l)}(X))$ *using the product-algorithm in [Sch77].*
6: ***return*** *the polynomial* $\min(G_t(X))$

Application to the E_0 keystream generator

We demonstrate now the efficiency of Algorithm 5.3 on the E_0 keystream generator. Recall that it employs four different LFSRs with pairwise co-prime primitive minimal polynomials m_1, m_2, m_3, m_4 of degrees $d_1 = 25$, $d_2 = 31$, $d_3 = 33$, and $d_4 = 39$.

In [ArmK03], an r-function F of degree 4 with $r = 4$ was derived. Furthermore, in [Cou03] it was shown that F can be divided into

$$F(X_1, \ldots, X_4, y_1, \ldots, y_4) = G(X_1, \ldots, X_4) + H(X_1, \ldots, X_4, y_1, \ldots, y_4)$$

with $\deg(G) = 4 > 3 = \deg(H)$. Thus, condition (5.1.3) from page 124 is satisfied and fast algebraic attacks are possible in principle. G has the form

$$G_t(X) = \sum_{\substack{1 \le i < j \le 4 \\ 1 \le k < l \le 4}} s_t^{(i)} s_t^{(j)} s_{t+1}^{(k)} s_{t+1}^{(l)} + s_t^{(1)} s_t^{(2)} s_t^{(3)} s_t^{(4)}$$

where $\mathcal{S}^{(i)} = (s_t^{(i)})$ is the sequence produced by LFSR i. Let R_i be the set of roots of m_i and

$$\Pi := \{\alpha_i \alpha_j \alpha_k \alpha_l | \ \alpha_s \in R_s, 1 \le i < j \le 4, 1 \le k < l \le 4\} \cup \{\alpha_1 \alpha_2 \alpha_3 \alpha_4 \mid \alpha_i \in R_i\} \ .$$

We have checked with the computer algebra system `Maple` that the weaker assumptions from Theorem 5.20 are fulfilled here, making Algorithm 5.2 applicable.

Furtheron, it can be shown that G can be written as $G = G^{(1)} + \ldots + G^{(11)}$ such that the minimal polynomials $\min(G_t^{(i)}(X))$ are pairwise co-prime. Let from now on i, j, k, l denote integers from the set $\{1, 2, 3, 4\}$. We define the following three sets of indices

$$\begin{aligned} I_2 &:= \{(i,j) \mid i < j\} \\ I_3 &:= \{(i,j,k) \mid i < j < k\} \\ I_4 &:= \{(i,j;k,l) \mid i < j, k < l, \{i,j\} \cup \{k,l\} = \{1,2,3,4\}\} \end{aligned}$$

Then, $G_t(X)$ can be rewritten to

$$G_t(X) = \sum_{(i,j)\in I_2} \underbrace{s_t^{(i)} s_{t+1}^{(i)} s_t^{(j)} s_{t+1}^{(j)}}_{=:G_t^{(i,j)}(X)} \tag{5.2.22}$$

$$+ \sum_{(i,j,k)\in I_3} \underbrace{\left(f_{ij} \cdot s_t^{(k)} s_{t+1}^{(k)} + f_{ik} \cdot s_t^{(j)} s_{t+1}^{(j)} + f_{jk} \cdot s_t^{(i)} s_{t+1}^{(i)}\right)}_{=:G_t^{(i,j,k)}(X)} \tag{5.2.23}$$

$$+ \underbrace{\sum_{(i,j;k,l)\in I_4} s_t^{(i)} s_{t+1}^{(j)} s_t^{(k)} s_{t+1}^{(l)} + s_t^{(1)} s_t^{(2)} s_t^{(3)} s_t^{(4)}}_{\tilde{G}_t(X)} \tag{5.2.24}$$

where $f_{ij} = s_t^{(i)} s_{t+1}^{(j)} + s_t^{(j)} s_{t+1}^{(i)}$. We define the sets

$$\begin{aligned} R_i^2 &:= \{\alpha \cdot \alpha' \mid \alpha, \alpha' \in R_i\} \\ R^{(i,j)} &:= \{\alpha_i \alpha_j \mid (i,j) \in I_2, \alpha_s \in R_s \cup R_s^2\} \\ R^{(i,j,k)} &:= \{\alpha_i \alpha_j \alpha_k \mid (i,j,k) \in I_3, \alpha_s \in R_s \cup R_s^2\} \\ \tilde{R} &:= \{\alpha_i \alpha_j \alpha_k \alpha_l \mid (i,j;k,l) \in I_4, \alpha_s \in R_s\} \ . \end{aligned}$$

One can show, e.g. with `Maple`, that the sets defined above are all pairwise disjoint. Furthermore, it is easy to see that the roots of $\min(G_t^{(i,j)}(X))$ are a subset of $R^{(i,j)}$, and so on. Thus, the minimal polynomials are pairwise co-prime, and we can write by Theorem 5.22 $\min(G_t(X))$ as the product of 11 different minimal polynomials:

$$\begin{aligned} \min(G_t(X)) &= \textstyle\prod_{(i,j)\in I_2} \min(G_t^{(i,j)}(X)) \cdot \prod_{(i,j,k)\in I_3} \min(G_t^{(i,j,k)}(X)) \cdot \min(\tilde{G}_t(X)) \\ &= \textstyle\prod_{i=1}^{11} \min(G_t^{(i)}(X)) \end{aligned} \tag{5.2.25}$$

Actually, even more can be said. Using Theorem 5.22 and the fact that the polynomials m_i are pairwise co-prime, we have

$$\min(G_t^{(i,j)}(X)) = (m_i \otimes m_i) \otimes (m_j \otimes m_j)$$

of degree $d_i(d_i+1)/2 \cdot T_j(T_j+1)/2$. Before we can estimate $\min(G_t^{(i,j,k)}(X))$ and $\min(\tilde{G}_t(X))$, we need the following theorem:

Theorem 5.27. *([LidN86, Th. 6.55]) Given sequences $\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(r)}$, the minimal polynomial* $\min(\mathcal{S}^{(1)} + \ldots + \mathcal{S}^{(r)})$ *divides* $\operatorname{lcm}(\min(\mathcal{S}^{(1)}), \ldots, \min(\mathcal{S}^{(r)}))$.

First, we consider the minimal polynomial of

$$\begin{aligned} G_t^{(i,j,k)}(X) &= (s_t^{(i)} s_{t+1}^{(j)} + s_t^{(j)} s_{t+1}^{(i)}) \cdot s_t^{(k)} s_{t+1}^{(k)} + (s_t^{(i)} s_{t+1}^{(k)} + s_t^{(k)} s_{t+1}^{(i)}) \cdot s_t^{(j)} s_{t+1}^{(j)} + \\ &\quad (s_t^{(j)} s_{t+1}^{(k)} + s_t^{(k)} s_{t+1}^{(j)}) \cdot s_t^{(i)} s_{t+1}^{(i)} \end{aligned}$$

It holds that

$$\begin{aligned} \min((s_t^{(i)} s_{t+1}^{(j)})) = \min((s_t^{(j)} s_{t+1}^{(i)})) &\overset{\text{Th. 5.22}}{=} m_i \otimes m_j \quad \text{and} \\ \min(s_t^{(k)} s_{t+1}^{(k)}) &\overset{\text{Th. 5.24}}{=} m_k \otimes m_k. \end{aligned}$$

By Theorem 5.27, the minimal polynomial $\min(G_t^{(i,j,k)}(X))$ divides the least common multiple of the polynomials $(m_i \otimes m_i) \otimes (m_j \otimes m_k)$, $(m_j \otimes m_j) \otimes (m_i \otimes m_k)$, and $(m_k \otimes m_k) \otimes (m_i \otimes m_j)$. Notice that all three polynomials share the common factor $m_i \otimes m_j \otimes m_k$.[5] Thus, the degree of $\min(G_t^{(i,j,k)}(X))$ is upper bounded by

$$\begin{aligned} & d_i d_j d_k + \frac{d_i(d_i-1)}{2} d_j d_k + \frac{d_j(d_j-1)}{2} d_i d_k + \frac{d_k(d_k-1)}{2} d_i d_j \\ = \; & d_i d_j d_k \frac{d_i + d_j + d_k - 1}{2} \; . \end{aligned}$$

The minimal polynomial $\min(\tilde{G}_t(X))$ can be found exactly. We observe that

$$\min(s_t^{(i)} s_{t+1}^{(j)} s_t^{(k)} s_{t+1}^{(l)}) = \min(s_t^{(1)} s_t^{(2)} s_t^{(3)} s_t^{(4)}) = m_1 \otimes m_2 \otimes m_3 \otimes m_4 =: m \; .$$

Thus, $\min(\tilde{G}_t(X))$ divides m. As $m_1, \ldots, m_4$ are irreducible, this holds for m, too. Therefore, $\min(\tilde{G}_t(X))$ is equal to 1 or equal to m. But the first case would imply that the expression in (5.2.24) is the all-zero sequence which is obviously wrong. Ergo, $\min(\tilde{G}_t(X))$ is equal to m of degree $d_1 d_2 d_3 d_4$. Summing up, for the degree T of $\min(G_t(X))$ it holds that

$$T \leq \sum_{(i,j) \in I_2} \frac{d_i(d_i+1) d_j(d_j+1)}{4} + \sum_{(i,j,k) \in I_3} d_i d_j d_k \frac{d_i + d_j + d_k - 1}{2} + d_1 d_2 d_3 d_4 \; . \quad \text{(5.2.26)}$$

In the specification of E_0, the degrees d_1, d_2, d_3, d_4 are defined as 25, 31, 33, 39 respectively. Thus, the degree of $\min(G_t(X))$ is $\leq 8,822,188 \approx 2^{23.07}$. The computation of $\min(G_t(X))$ using Algorithm 5.2 would need about $2^{46.15}$ basic operations.

[5] The reason is that f divides $f \otimes f$ and that $(f \cdot g) \otimes h = (f \otimes h) \cdot (g \otimes h)$.

Minimal polynomials	Alg. 5.2	Alg. 5.3	Ratio
$1+x+x^3,\ 1+x+x^4,$ $1+x^2+x^5,\ 1+x^4+x^7$	10h 41m 43s	12m 3s	53.32
$1+x^2+x^3,\ 1+x^3+x^4,$ $1+x^3+x^5,\ 1+x^6+x^7$	11h 2m 49s	12m 7s	54.75
"	10h 50m 0s	11m 59s	54.30
"	10h 52m 59s	11m 55s	54.86
"	10h 53m 31s	11m 58s	54.65
$1+x+x^3,\ 1+x^2+x^4,$ $1+x^3+x^5,\ 1+x^2+x^6+x^8+x^{11}$	3d 6h 30m 16s	1h 43m 25s	45.55
$1+x^2+x^3,\ 1+x^2+x^5,$ $1+x+x^7,\ 1+x+x^2+x^6+x^{11}$	18d 18h 26m	13h 50m 7s	32.56

Table 5.2.3: Comparison between the time efforts of Algorithms 5.2 and 5.3 applied to reduced versions of E_0

Equation (5.2.25) implies the usage of Algorithm 5.3. In the first step we apply Algorithm 5.2 to compute the minimal polynomials $\min(G_t^{(i)}(X))$, $i = 1, \ldots, 11$. This takes an overall number of basic operations of $\approx 2^{43.37}$. Exploiting parallelism, we have only to wait the time needed to perform $\approx 2^{41.91}$ basic operations.[6]

The second step is the computation of the product of these minimal polynomials. Here, this takes altogether about $\approx 2^{28.25}$ basic operations. Performed in sequential, Algorithm 5.3 needs about $2^{43.37}$ basic operations, which is almost 8 times faster than Algorithm 5.2. If we exploit the parallelism mentioned above, the number of basic operations we have to wait for is about $2^{41.91}$ which is more than 16 times faster than in Algorithm 5.2.

An ad-hoc implementation in `Maple` (without using parallelism and the algorithm of [BosFSS06]), applied to reduced versions of E_0 with shorter LFSRs, confirmed the improved efficiency of our new algorithm. The results can be found in table 5.2.3. In the first column, the used minimal polynomials are given. The next two columns show the time consumptions of Algorithm 5.2 and 5.3, respectively, and the ratio in the last column. In all cases, our new Algorithm 5.3 is significantly faster than Algorithm 5.2, even without using parallelism. The speed-up factor is much higher than predicted theoretically and depends on the chosen minimal polynomials and the initial states.

In all cases, the degree of $\min(G_t(X))$ was equal to the upper bound estimated in (5.2.26). Hence, we expect that the upper bound is tight for the

[6] Of course, the total number of operations remains the same.

real E_0 keystream generator, too. Therefore, we make the assumption that $\deg(\min(G_t(X))) = 8,822,188$.

Summary

As we have seen, different methods exist to compute $\min(G_t(X))$. In the general case, one can use Algorithm 5.2 as proposed originally in [Cou03]. As discussed in Section 5.2.1, it works correctly in most cases, even if the lengths of the LFSRs are not co-prime. The time effort is mostly the effort of the Berlekamp-Massey algorithm, which is in $O(D^2)$. Actually, there exists an improved version of the Berlekamp-Massey algorithm which has a faster asymptotic runtime behaviour, namely in $O(D \cdot \log_2(D))$ (see [Bla83, BreGY80, Dor87]). But whether it is really faster for the cases that we are interested in has not been examined so far.

The method proposed in [HawR04] is extremely fast as the time effort was estimated to lie in $O(D \cdot (n \log_2(n) + \log_2(D)^3))$. However, the drawback is that it does not compute $\min(G_t(X))$ but $\min(d) = \sum_{i=0}^{T'} \gamma'_i x^i$ with $\sum_i \gamma'_i G'_{t+i}(K) \equiv 0$ for all t and for all G' of degree $\leq d$. Obviously, the coefficients γ'_i can be used for a fast algebraic attack as well, but this strategy might demand knowing more successive keystream bits than actually necessary for computing the coefficients γ_i.

In certain cases, if more than one LFSR is employed in the (ι, m)-combiner, it is possible to exploit the information on G and the LFSRs to derive $\min(G_t(X))$ more directly. This is the idea behind Algorithm 5.3. Simulations on E_0 with reduced key sizes confirmed the reduced time effort.

5.3 Divide-and-conquer fast algebraic attacks

As explained in Section 5.1, fast algebraic attacks (FAAs) are based on processing the system of equations in a precomputation step in order to decrease the degree of the equations. In this section, we show that a similar approach may be used to reduce the number of unknowns. Before we give a general description, we motivate the attack by an example.

Example 5.28. *The example uses five LFSRs of lengths $n_1, \ldots, n_5$ filtered by a memory function. Let $s_t^{(i)}$ denote the output of the i^{th} LFSR at clock t and $n := n_1 + \ldots + n_5$ the key size. The output z_t for $t \geq 0$ is computed as follows:*

$$\begin{aligned} z_t &:= s_t^{(1)} + s_{t-1}^{(1)} \cdot (\sigma_{t-1}^2 + \sigma_t^2) + \sigma_t^3 + \sigma_{t-1}^2 + Q_t \cdot \sigma_{t-1}^3 \\ Q_{t+1} &:= z_t \end{aligned}$$

where Q_0 is some initial memory of the function and σ_t^d is the d-th elementary symmetric polynomial in the outputs $s_t^{(2)}$, $s_t^{(3)}$, $s_t^{(4)}$, $s_t^{(5)}$. By inserting z_{t-1} into Q_t one gets an algebraic relation that is independent of the memory:

$$z_t := s_t^{(1)} + s_{t-1}^{(1)} \cdot (\sigma_{t-1}^2 + \sigma_t^2) + \sigma_t^3 + \sigma_{t-1}^2 + z_{t-1} \cdot \sigma_{t-1}^3 \tag{5.3.27}$$

Equation (5.3.27) is of degree 3 in the key bits, which yields an algebraic attack with an effort in $\mathcal{O}\left(\binom{n}{3}^{\omega}\right)$. We have tested with the methods described in Section 4.2 that no quadratic relations exist over two clocks. Because of the term $z_{t-1} \cdot \sigma_{t-1}^3$, it is not possible to split (5.3.27) as shown in (5.1.3). Therefore, a FAA as described in [Cou03] respectively Section 5.1 is not applicable in this case.

Furtheron, we checked that what we would call a "local divide-and-conquer" attack is possible (e.g., see [Gol04]): equations exist which are independent of $s_t^{(i)}$ and $s_{t-1}^{(i)}$ for one $i \in \{2, 3, 4, 5\}$, but there is no equation independent of variables $s_t^{(1)}$ and $s_{t-1}^{(1)}$. But as the lowest degree of these equations is 4, this approach would increase the complexity instead of reduce it. Altogether, it seems that the naive algebraic attack is the best algebraic attack in this case.

Actually, one can do better. In a "conventional" fast algebraic attack, one would split F into two parts to cancel out the components of high degree. Here, the motivation is slightly different. Instead of reducing the degree, the number of unknowns is reduced. Similar to (5.1.3), we rewrite (5.3.27) to

$$0 = \underbrace{x_t^{(1)} + x_{t-1}^{(1)} \cdot (\sigma_{t-1}^2 + \sigma_t^2)}_{=:G_t(X)} + \underbrace{z_t + \sigma_t^3 + \sigma_{t-1}^2 + z_{t-1} \cdot \sigma_{t-1}^3}_{=:H_t(X)} \ .$$

Attack	degree	#unknowns	time	memory
algebraic attack	3	128	$\approx 2^{55}$	$\approx 2^{37}$
fast algebraic attack	/	/	/	/
local divide-and-conquer	4	128-33=95	$\approx 2^{65}$	$\approx 2^{43}$
our attack	3	128-33=95	$\approx 2^{52}$	$\approx 2^{35}$

Table 5.3.4: Different algebraic attacks against the given example

The splitting is chosen such that H is independent *of the outputs of LFSR 1. Thus, cancelling the G part leads to a new equation which depends only on the outputs from LFSRs 2-5.*

Let $\min(G_t(X)) = \sum_{i=0}^{T} \gamma_i x^i = 0$. Then,

$$0 = \sum_{i=0}^{T} \gamma_i F_{t+i}(K, z_{t+i}, \ldots, z_{t+i+r-1}) = \sum_{i=0}^{T} \gamma_i H_{t+i}(K, z_{t+i}, \ldots, z_{t+i+r-1})$$

is a $(T + r - 1)$-equation of degree 3 which is independent of LFSR 1. This reduces the number of monomials from $\mathcal{O}\left(\binom{n}{3}\right)$ to $\mathcal{O}\left(\binom{n-n_1}{3}\right)$ and the effort to solve the corresponding linearized system of equations from $\mathcal{O}\left(\binom{n}{3}^{\omega}\right)$ to $\mathcal{O}\left(\binom{n-n_1}{3}^{\omega}\right)$. If the feedback polynomials are pairwise co-prime, then the run length is $T = n_1 + n_1 . \sum_{2 \le i < j \le 5} n_i n_j$. For $(n_1, \ldots, n_5) = (33, 27, 26, 25, 17)$ and $\omega = 3$, the time and space efforts for different algebraic attacks are displayed in Table 5.3.4.

We give now a more general description of this approach. Required is a r-function F which can be split as shown here:

$$0 = F_t(X, y_1, \ldots, y_r) = G_t(X) + H_t(\hat{X}_j, y_1, \ldots, y_r). \tag{5.3.28}$$

$\hat{X}_j$ denotes X where the part belonging to the j^{th} LFSR is left out. More precisely, it is $\hat{X}_j := (X_1, \ldots, X_{j-1}, X_{j+1}, \ldots, X_m)$. The next step is to compute $\min(G_t(X))$, e.g., with the methods discussed in Section 5.2. Observe that $\sum_{i=0}^{T} \gamma_i H_t(\hat{K}_j, z_{t+i}, \ldots, z_{t+i+r-1}) = 0$ is a valid equation and independent of K_j. Repeating this step for several t gives a system of equations which is independent of K_j:

$$\underbrace{\begin{array}{rcl} 0 &=& F_0(K, z_0, \ldots, z_{r-1}) \\ 0 &=& F_1(K, z_1, \ldots, z_r) \\ & \ldots & \end{array}}_{n \text{ unknowns}} \mapsto \underbrace{\begin{array}{rcl} 0 &=& \sum_i \gamma_i H_i(\hat{K}_j, \ldots) \\ 0 &=& \sum_i \gamma_i H_{1+i}(\hat{K}_j, \ldots) \\ & \ldots & \end{array}}_{n - n_j \text{ unknowns}} \tag{5.3.29}$$

This reduces the number of computation steps from (roughly) $\binom{n}{d}^{\omega}$ to $\binom{n-n_j}{d}^{\omega}$ and the amount of space from $\binom{n}{d}^{2}$ to $\binom{n-n_j}{d}^{2}$. Afterwards, the values of K_j can be easily reconstructed or even guessed.

5.4 Adapted precomputation steps with minimized run length

5.4.1 Motivation

So far, all variants of fast algebraic attacks discussed in the preceding sections are based on the same principle. The first step is to find an r-function $F(X, y_1, \ldots, y_r)$ which can be rewritten to

$$F(X, y_1, \ldots, y_r) = G(X) + H(X, y_1, \ldots, y_r) \ ,$$

where H has some desired attribute $\mathcal{A}$, but G has not. Hereby, $\mathcal{A}$ could mean that the degree is less than $d = \deg(F)$ (this corresponds to the original fast algebraic attack as described in Section 5.1) or that the output is independent of the inputs of some LFSRs (as used for the divide-and-conquer attack from Section 5.3). In other words, F is divided into H which has attribute $\mathcal{A}$, and G which has not.

Then, the second step consists of computing $\min(G_t(X)) = \sum_{i=0}^{T} \gamma_i x^i$. Thus, $\sum_{i=0}^{T} \gamma_i \cdot F_{t+i}(K, z_t, \ldots, z_{t+r-1})$ gives a $(T + r - 1)$-function which has attribute $\mathcal{A}$.

However, one has to pay the price that the run length may increase enormously, for example from 4 to $T \approx 8,822,188$ in the case of E_0. Thus, in terms of practicability, it is desirable to reduce T as much as possible. This means for example that one should use the coefficients from $\min(G_t(X))$ instead of those from a characteristic polynomial of the sequence $(G_t(X))$. But even then, further reductions may be possible as the following example shows:

Example 5.29. *Consider the following 2-function*

$$F_t(X) = \underbrace{s_t^{(2)} s_{t+1}^{(2)} s_t^{(1)}}_{=:G_t(X)} + \underbrace{s_t^{(1)} s_t^{(2)} z_t}_{=:H_t(X)} \tag{5.4.30}$$

where $s_t^{(i)}$ are produced by LFSRs with minimal polynomials $m_1(x) = x^2 + x + 1$ and $m_2(x) = 1 + x + x^3$. That is, the desired attribute $\mathcal{A}$ here is to be quadratic.

The first elements of the sequence $(G_t(X)) = (G_t(x_1, x_2, x_3, x_4, x_5))$ *are:*

$$\begin{aligned}
G_0(X) &= && x_1x_3x_4 \\
G_1(X) &= && x_2x_4x_5 \\
G_2(X) &= x_1x_5 + x_2x_5+ && x_1x_3x_5 + x_2x_3x_5 \\
G_3(X) &= x_1x_3 + x_1x_5+ && x_1x_3x_4 + x_1x_4x_5 \\
G_4(X) &= x_2x_3 + x_2x_4+ && x_2x_3x_5 + x_2x_4x_5 \\
G_5(X) &= x_1x_4 + x_2x_4+ && x_1x_3x_4 + x_1x_3x_5 + x_1x_4x_5+ \\
& && x_2x_3x_4 + x_2x_3x_5 + x_2x_4x_5 \\
G_6(X) &= && x_1x_3x_4 + x_1x_3x_5 \\
G_7(X) &= && x_2x_3x_4 \\
G_8(X) &= && x_1x_4x_5 + x_2x_4x_5 \\
G_9(X) &= x_1x_5+ && x_1x_3x_5 \\
G_{10}(X) &= x_2x_3 + x_2x_5+ && x_2x_3x_4 + x_2x_4x_5 \\
G_{11}(X) &= x_1x_3 + x_2x_3 + x_2x_4 + x_1x_4+ && x_1x_3x_5 + x_1x_4x_5 + x_2x_3x_5 + x_2x_4x_5 \\
G_{12}(X) &= x_1x_4+ && x_1x_3x_4 + x_1x_3x_5 + x_1x_4x_5
\end{aligned}$$

Now, with the methods described so far, one can compute

$$\min(G_t(X)) = 1 + x + x^3 + x^4 + x^6 + x^8 + x^9 + x^{11} + x^{12}.$$

Thus, it holds that

$$\begin{aligned}
G_t(X) + G_{t+1}(X) + G_{t+3}(X) + G_{t+4}(X) + G_{t+6}(X) + \\
G_{t+8}(X) + G_{t+9}(X) + G_{t+11}(X) + G_{t+12}(X) \quad \equiv \quad 0 \quad \forall t \ .
\end{aligned}$$

For example, the sum of the following functions is equal to zero:

$$\begin{aligned}
G_0(X) &= && x_1x_3x_4 \\
G_1(X) &= && x_2x_4x_5 \\
G_3(X) &= x_1x_3 + x_1x_5+ && x_1x_3x_4 + x_1x_4x_5 \\
G_4(X) &= x_2x_3 + x_2x_4+ && x_2x_3x_5 + x_2x_4x_5 \\
G_6(X) &= && x_1x_3x_4 + x_1x_3x_5 \\
G_8(X) &= && x_1x_4x_5 + x_2x_4x_5 \\
G_9(X) &= x_1x_5+ && x_1x_3x_5 \\
G_{11}(X) &= x_1x_3 + x_2x_3 + x_2x_4 + x_1x_4+ && x_1x_3x_5 + x_1x_4x_5 + x_2x_3x_5 + x_2x_4x_5 \\
G_{12}(X) &= x_1x_4+ && x_1x_3x_4 + x_1x_3x_5 + x_1x_4x_5
\end{aligned}$$

This implies that

$$\deg(H_t + H_{t+1} + H_{t+3} + H_{t+4} + H_{t+6} + H_{t+8} + H_{t+9} + H_{t+11} + H_{t+12})) = 2 \ , \quad (5.4.31)$$

where the run length is 13. But now, we take a look at the sum of the

following functions:

$$\begin{array}{lll} G_0(X) = & & x_1x_3x_4 \\ G_1(X) = & & x_2x_4x_5 \\ G_2(X) = & x_1x_5 + x_2x_5 + & x_1x_3x_5 + x_2x_3x_5 \\ G_4(X) = & x_2x_3 + x_2x_4 + & x_2x_3x_5 + x_2x_4x_5 \\ G_6(X) = & & x_1x_3x_4 + x_1x_3x_5 \end{array}$$

As all cubic expressions cancel out, the result

$$G_0(X) + G_1(X) + G_2(X) + G_4(X) + G_6(X) = x_1x_5 + x_2x_5 + x_2x_3 + x_2x_4$$

is a quadratic equation. In particular, this implies that

$$H_t(X) + H_{t+1}(X) + H_{t+2}(X) + H_{t+4}(X) + H_{t+6}(X) \tag{5.4.32}$$

is a quadratic equation as well, but with a run length of 7 instead of 13 as in (5.4.31).

Summing up, instead of looking for $\min(G_t(X))$ to cancel the G-part, it suffices to find coefficients $\gamma'_0, \ldots, \gamma'_{T'}$ such that $\sum \gamma'_i G_{t+i}$ possess attribute $\mathcal{A}$. In the context of normal fast algebraic attacks, this has been mentioned the first time in [HawR04]. The goal of this section is therefore to develop according methods. First, we make the notions of $\mathcal{A}$ more precise.

Definition 5.30. *For an integer $a = \sum_{i=0} a_i \cdot 2^i$, $a_i \in \{0,1\}$, we define its **binary weight** $wt_{bin}(a) := \sum_i a_i$. For a vector $(a^{(1)}, \ldots, a^{(m)})$ of integers, we extend this definition to $wt_{bin}(a^{(1)}, \ldots, a^{(n)}) = \sum wt_{bin}(a^{(i)})$.*

Example 5.31. *The following tabular displays the binary weight for the first eight non-negative integers.*

a	0	1	2	3	4	5	6	7
(a_2, a_1, a_0)	(000)	(001)	(010)	(011)	(100)	(101)	(110)	(111)
$wt_{bin}(a)$	0	1	1	2	1	2	2	3

Definition 5.32. *Let $n_1, \ldots, n_m$ be some non-negative integers and $X_i := (x_{i,0}, \ldots, x_{i,n_i-1})$. For $e_i = \sum_{j=0}^{n_i-1} e_{i,j} 2^j$, we define $(X_i)^{e_i} := \prod_{j=1}^{n_i} x_{i,j}^{e_{i,j}}$ and for $E = (e_1, \ldots, e_m)$ the expression $X^E := \prod_{j=1}^{m} X_j^{e_j}$.*

Furthermore, let $G(X_1, \ldots, X_m) = \sum_{E=(e_1,\ldots,e_m)} c_E \cdot X^E$ be a Boolean function in $n := n_1 + \ldots + n_m$ unknowns with $c_E \in \mathbb{F}_2$. We define the two different types of degree:

$$\begin{aligned} \deg(G) &:= \max\{wt_{bin}(E) \,|\, c_E \neq 0\} \\ \deg_{X_i}(G) &:= \max\{wt_{bin}(e_i) \,|\, c_{(e_1,\ldots,e_m)} \neq 0\} \end{aligned}$$

If $deg_{X_i}(G) = 0$, then G is independent of the variables in X_i.

Remark 5.33. *The relationship between the definitions of X^E from Definition 5.32 and Definition 2.23 is as follows. Let $X_i := (x_{i,1}, \ldots, x_{i,n_i})$, $i = 1, \ldots, n$ and $e_i := \prod_{j=0}^{n_i-1} e_{i,j} \cdot 2^j$. Then,*

$$(X_1, \ldots, X_n)^{(e_1,\ldots,e_n)} \stackrel{\text{Def. 5.32}}{=} \prod_{i=1}^{n} \prod_{j=1}^{n_i} x_{i,j}^{e_{i,j}} \stackrel{\text{Def. 2.23}}{=} (x_{1,1}, \ldots, x_{1,n_1}, x_{2,1}, \ldots, x_{n,n_n})^{(e_{1,1},\ldots,e_{n,n_n})}.$$

We express now the attributes $\mathcal{A}$ for the different kind of fast algebraic attacks with notions introduced in Definition 5.32 in the following table:

Fast algebraic attack	Explained in Section	Goal: Find γ_i such that
degree-decreasing	5.1	$\deg(\sum_i \gamma_i \cdot G_{t+i}(X)) \leq \deg(H)$
divide-and-conquer	5.3	$\deg_{X_i}(\sum_i \gamma_i \cdot G_{t+i}(X)) = 0$

So, if we are able to modify the search for the coefficients γ_i such that the adapted conditions are fulfilled, one can reduce the run length and make the whole attack more efficient. At first glance, one may expect that this requires more complicated and time consuming methods than solely looking for $\min(G_t(X))$. But, somewhat surprisingly, we found out in [ArmA05] that one can use a combination of the algorithms from Section 5.2, i.e., it is possible to derive equations for a fast algebraic attacks with minimum run length with a rather small increase of the effort. Readers who are only interested in these adapted algorithms may jump directly to the last subsection. The other sections are devoted solely to deriving these algorithms and proving their correctness.

Algorithm 5.4 gives a preview on our approach to find adapted linear sums of G_t. The correctness of this approach is ensured by the following fact that will be proved later.

$$\sum_i \gamma_i G_{t+i}(X) \text{ has property } \mathcal{A} \iff p(x) = \sum_i \gamma_i x^i \ : \ p(\alpha) = 0 \ \forall \alpha \in \Omega_{\mathcal{A}} \ .$$

Actually, this is an extension of Theorem 5.15 where we already showed the connection between $\sum_i \gamma_i G_{t+i}(X)$ and the minimal polynomial of a subset of an extension field $\mathbb{F}_{2^d}$. The degree T of $p(x)$ determines the number of successive equations. If it is possible to compute the minimal polynomial of $\Omega_{\mathcal{A}}$ efficiently, then this yields immediately the coefficients for the minimum run length.

Algorithm 5.4.
Computation of coefficients γ_i such that $\sum_i \gamma_i G_{t+i}(X)$ has a specific property $\mathcal{A}$ (e.g., the degree is $\leq e$ or is independent of certain inputs)
Input: *Boolean function $G : \mathbb{F}^n \to \mathbb{F}$, a property $\mathcal{A}$ that has to be met by the linear sum*
Output: *Coefficients $\gamma_0, \ldots, \gamma_{T-1}$ such that $\sum_{i=0}^{T-1} G_{t+i}(X)$ fulfills $\mathcal{A}$*

1: *Use Φ from the Section 5.4.2 to embed $G \mapsto \Phi(G) = \mathcal{G} \in \mathbb{F}[Y_1, \ldots, Y_m]$ with $\mathbb{F}$ an extension field of $\mathbb{F}_2$.*
2: *Derive a set $\Omega_{\mathcal{A}} \subsetneq \mathbb{F}$ depending on $\mathcal{G}$ and the desired attribute $\mathcal{A}$ as explained in Section 5.4.3.*
3: *Determine the minimal polynomial $\min(\Omega_{\mathcal{A}}) = \sum_i \gamma_i x^i$ with the methods described in Section 5.4.4.*
4: ***return*** $(\gamma_0, \ldots, \gamma_{T-1})$

5.4.2 A special representation of Boolean function

In this section, we present an embedding Φ of multivariate Boolean functions into the set of multivariate polynomial over $\mathbb{F} \supset \mathbb{F}_2$. From this representation, we will later derive the sets $\Omega_{\mathcal{A}}$ mentioned before. The first step to develop Φ is to introduce a special bijection $\mathbb{F}_2^n \to \mathbb{F}_{2^n}$ and extend it to $\mathbb{F}_2^{n_1} \times \ldots \times \mathbb{F}_2^{n_m} \to \mathbb{F}_{2^{n_1}} \times \ldots \times \mathbb{F}_{2^{n_m}}$. Although we describe the results in their generality, the reader should keep in mind that our reasoning is motivated by fast algebraic attacks. In particular, the polynomials p correspond to the feedback polynomials of the LFSRs and L to their feedback matrices.

Theorem 5.34. *Let $p(x) = \sum_{i=0}^{n} c_i \cdot x^i \in \mathbb{F}_2[x]$ be an irreducible polynomial of degree n and with root α. Furtheron, let $L \in GL_n(\mathbb{F}_2)$ be its companion matrix and $X := (x_0, \ldots, x_{n-1})$. There exists a $\mathbb{F}_2$-linear bijection $\varphi_L : \mathbb{F}_2^n \to \mathbb{F}_{2^n}$ with $\varphi_L(X \cdot L^i) = \alpha^i \varphi_L(X)$.*

Proof. As L is similar to L^T, there exists a unique matrix $S \in GL_n(\mathbb{F}_2)$ with $S \cdot L^T = L \cdot S$. Let $s_0, \ldots, s_{n-1}$ denote the n $\mathbb{F}_2$-linear functions with $(s_0(X), \ldots, s_{n-1}(X)) = X \cdot S$. We define $\varphi_L(X) := \sum_{i=0}^{n-1} s_i(X) \cdot \alpha^i$. As the functions s_i are linearly independent and $1, \alpha, \ldots, \alpha^{n-1}$ forms a basis of $\mathbb{F}_{2^n} = \mathbb{F}_2[1, \alpha, \ldots, \alpha^{n-1}]$ (see Theorem 2.38), φ_L is clearly a bijection. As φ_L is the sum of linear functions s_i, it is linear. What is left to show is that $\varphi_L(X \cdot L^i) = \alpha^i \cdot \varphi_L(X)$. To do so, we define the vector $\overrightarrow{\alpha} = (1, \alpha, \ldots, \alpha^{n-1})^T$. Then it holds that $\varphi_L(X) = \langle \overrightarrow{\alpha}, X \cdot S \rangle$ where $\langle ., . \rangle$ denotes the usual vector

product. As α is a root of $p(x)$, it is $\alpha^n = \sum_{i=0}^{n-1} c_i \alpha^i$. This implies

$$\begin{aligned}
\alpha \cdot \varphi_L(X) &= \sum_{i=0}^{n-1} s_i(X) \cdot \alpha^{i+1} = \sum_{i=0}^{n-2} s_i(X) \cdot \alpha^{i+1} + s_{n-1}(X) \cdot \alpha^n \\
&= \sum_{i=0}^{n-2} s_i(X) \cdot \alpha^{i+1} + s_{n-1}(X) \cdot \sum_{i=0}^{n-1} c_i \alpha^i \\
&= c_0 s_{n-1}(X) + (s_0(X) + c_1 s_{n-1}(X))\alpha + \ldots \\
&\quad + (s_{n-2}(X) + c_{n-1} s_{n-1}(X))\alpha^{n-1} \\
&= \left\langle \overrightarrow{\alpha} \ , \ \Big(c_0 s_{n-1}(X), s_0(X) + c_1 s_{n-1}(X), \ldots, s_{n-2}(X) + c_{n-1} s_{n-1}(X) \Big) \right\rangle \\
&= \left\langle \overrightarrow{\alpha} \ , \ (s_0(X), \ldots, s_{n-1}(X)) \cdot L^T \right\rangle = \langle \overrightarrow{\alpha}, (X \cdot S) \cdot L^T \rangle \\
&= \langle \overrightarrow{\alpha}, X \cdot (S \cdot L^T) \rangle = \langle \overrightarrow{\alpha}, X \cdot (L \cdot S) \rangle = \langle \overrightarrow{\alpha}, (X \cdot L) \cdot S \rangle = \varphi_L(X \cdot L) \ .
\end{aligned}$$

The rest follows by induction. □

Corollary 5.35. *Let $p_1(x), \ldots, p_m(x) \in \mathbb{F}_2[x]$ be irreducible polynomials of degrees $n_1, \ldots, n_m$ and with roots $\alpha_1, \ldots, \alpha_m$, respectively. Let $L_1, \ldots, L_m$ be their companion matrices. Then there exists a linear bijection $\varphi := \varphi_{L_1,\ldots,L_m} : \mathbb{F}_2^{n_1} \times \ldots \times \mathbb{F}_2^{n_m} \to \mathbb{F}_{2^{n_1}} \times \ldots \times \mathbb{F}_{2^{n_m}}$ such that for all $(X_1, \ldots, X_m) \in \mathbb{F}_2^{n_1} \times \ldots \times \mathbb{F}_2^{n_m}$, we have*

$$\begin{aligned}
\varphi(X_1 \cdot L_1^i, \ldots, X_m \cdot L_m^i) &:= (\varphi_{L_1}(X_1 \cdot L_1^i), \ldots, \varphi_{L_m}(X_m \cdot L_m^i)) \\
&= (\alpha_1^i \varphi_{L_1}(X_1), \ldots, \alpha_m^i \varphi_{L_m}(X_m)).
\end{aligned}$$

The function φ allows to give an alternative description of the values $\deg(G)$ and $\deg_{X_j}(G)$:

Theorem 5.36. *Let $p_j(x) \in \mathbb{F}_2[x]$, $1 \le j \le m$, be irreducible polynomials of degrees n_j and $\mathbb{F}_{2^{n'}}$ their splitting field. Set $X_i = (x_0^{(i)}, \ldots, x_{n_i-1}^{(i)})$. There exists a linear injection $\Phi = \Phi_{p_1,\ldots,p_m}$:*

$$\begin{aligned}
\Phi : \mathbb{F}_2[X_1, \ldots, X_m] &\hookrightarrow \mathbb{F}_{2^{n'}}[y_1, \ldots, y_m] / \langle y_i^{2^{n'}} - y_i, \forall i \rangle \\
G(X_1, \ldots, X_m) &\mapsto \mathcal{G}(y_1, \ldots, y_m) = \sum_{0 \le e_1, \ldots, e_m \le 2^{n'}-1} c_{e_1,\ldots,e_m} \cdot y_1^{e_1} \cdot \ldots \cdot y_m^{e_m}
\end{aligned}$$

such that

$$\begin{aligned}
\deg(G) &= \max\{wt_{bin}((e_1, \ldots, e_m)) \,|\, c_{e_1,\ldots,e_m} \neq 0\} \text{ and} \\
\deg_{X_j}(G) &= \max\{wt_{bin}(e_j) \,|\, c_{e_1,\ldots,e_m} \neq 0\} \ .
\end{aligned}$$

In other words, Φ maps a function G over $\mathbb{F}_2$ in $n := n_1 + \ldots + n_m$ variables to a function $\mathcal{G}$ in m variables over the splitting field $\mathbb{F}_{2^{n'}}$ such that the different notions of degrees of G can be retrieved from the expression of $\mathcal{G}$.

Proof. Let $G(X_1, \ldots, X_m) : \mathbb{F}_2^{n_1} \times \ldots \times \mathbb{F}_2^{n_m} \to \mathbb{F}_2$ with $X_i = (x_0^{(i)}, \ldots, x_{n_i-1}^{(i)})$. We use the $\mathbb{F}_2$-linear bijections $\varphi_{L_i} : \mathbb{F}_2^{n_i} \to \mathbb{F}_{2^{n_i}}$ from Theorem 5.34 to set $\tilde{x}_i := \varphi_{L_i}(X_i)$. As both φ_{L_i} and the projection are linear, there exist linear functions $\ell_j^{(i)} \in \mathbb{F}_{2^{n_i}}[x]$ with $x_j^{(i)} = \ell_j^{(i)}(\tilde{x}_i)$.

Next, we argue why $\ell_j^{(i)}(\tilde{x}_i)$ can be equivalently expressed by $\ell_0^{(i)}(\alpha_i^j \cdot \tilde{x}_i)$ with α_i being a root of $p_i(x)$. Recall that by the definition of the companion matrix L_i, it holds that $X_i \cdot L_i^j = (x_j^{(i)}, \ldots, x_{n_i-1}^{(i)}, *, \ldots, *)$ with " $*$ " being appropriate linear expressions in X_i. Then, the definition of $\ell_0^{(i)}$ yields

$$\begin{aligned} x_j^{(i)} &= \ell_0^{(i)}(\varphi_{L_i}(x_j^{(i)}, \ldots, x_{n_i-1}^{(i)}, *, \ldots, *)) \\ &= \ell_0^{(i)}(\varphi_{L_i}(X \cdot L^j)) = \ell_0^{(i)}(\alpha_i^j \varphi_{L_i}(X)) = \ell_0^{(i)}(\alpha_i^j \cdot \tilde{x}_i). \end{aligned}$$

Observe that $\alpha_1, \ldots, \alpha_m$ depend only on the feedback polynomials $p_1, \ldots, p_m$ and are thus independent of the choice of G. We use the preceding observations to define the following function:

$$\begin{aligned} \tilde{G} : \mathbb{F}_{2^{n_1}} \times \ldots \times \mathbb{F}_{2^{n_m}} &\to \mathbb{F}_2 \\ (\tilde{x}_1, \ldots, \tilde{x}_m) &\mapsto G(\ell_0^{(1)}(\tilde{x}_1), \ldots, \ell_0^{(1)}(\alpha_1^{n_1-1}\tilde{x}_1), \ldots, \\ &\qquad \ell_0^{(m)}(\tilde{x}_m), \ldots, \ell_0^{(m)}(\alpha_m^{n_m-1}\tilde{x}_m)) \ . \end{aligned} \tag{5.4.33}$$

As $\tilde{x} \mapsto (\ell_0(\tilde{x}), \ldots, \ell_0(\alpha n - 1 \cdot \tilde{x}))$ is a bijection $\mathbb{F}_{2^n} \to \mathbb{F}_2^n$, it holds that $\tilde{G}_1 = \tilde{G}_2$ implies $G_1 = G_2$. Because of $\mathbb{F}_{2^{n_i}} \subseteq \mathbb{F}_{2^{n'}}$ for $1 \leq i \leq m$, one can easily extend $\tilde{G}$'s domain to $(\mathbb{F}_{2^{n'}})^m$:

$$\mathcal{G}(y_1, \ldots, y_m) := G(\ell_0^{(1)}(y_1), \ldots, \ell_0^{(1)}(\alpha_1^{n_1-1}y_1), \ldots, \ell_0^{(m)}(y_m), \ldots, \ell_0^{(m)}(\alpha_m^{n_m-1}y_m)) \ .$$

Let $\Phi(G) := \mathcal{G}$. Observe that

$$\begin{aligned} & \mathcal{G}_1 = \Phi(G_1) \equiv \Phi(G_2) = \mathcal{G}_2 \\ \Rightarrow \ & \mathcal{G}_1|_{\mathbb{F}_{2^{n_1}} \times \ldots \times \mathbb{F}_{2^{n_m}}} \equiv \mathcal{G}_2|_{\mathbb{F}_{2^{n_1}} \times \ldots \times \mathbb{F}_{2^{n_m}}} \\ \Leftrightarrow \ & \tilde{G}_1 = \tilde{G}_2 \\ \Rightarrow \ & G_1 = G_2. \end{aligned}$$

Thus, Φ is an injection.

It is easy to see that Φ is linear, i.e., $\Phi(G_1 + G_2) = \Phi(G_1) + \Phi(G_2)$, and that $\Phi(G_1 \cdot G_2) = \Phi(G_1) \cdot \Phi(G_2)$. Therefore, it suffices to restrict on the case $m = 1$ and G a monomial. We set $m := 1$, $n := n_1$, $\alpha := \alpha_1$, $y := y_1$ and $G(X) = G(x_0, \ldots, x_{n-1}) := \prod_{i \in I} x_i$ for $I := \{i_1, \ldots, i_d\} \subseteq \{0, \ldots, n-1\}$. Thus, $\deg(G) = d$. As the function ℓ_0 is $\mathbb{F}_2$-linear, there exist elements $c_0, \ldots, c_{n-1} \in \mathbb{F}_{2^n}$ such that $\ell_0(y) = \sum_{j=0}^{n-1} c_j y^{2^j}$ (see [LidN86]). Therefore, the following equation is

valid

$$\begin{aligned}
\Phi(G(X)) &= \mathcal{G}(y_1,\ldots,y_m) \stackrel{\text{def}}{=} G(\ell_0(y),\ldots,\ell_0(\alpha^{n-1}y)) = \prod_{i\in I}\ell_0(\alpha^i y) \\
&= \prod_{i\in I}(\sum_{j=0}^{n-1} c_j \alpha^{2^j\cdot i} y^{2^j})) = (\sum_{j_1=0}^{n-1} c_{j_1}\alpha^{2^{j_1}\cdot i_1} y^{2^{j_1}})) \cdot \ldots \cdot (\sum_{j_d=0}^{n-1} c_{j_d}\alpha^{2^{j_d}\cdot i_d} y^{2^{j_d}})) \\
&= \sum_{j_1,\ldots,j_d=0}^{n-1} \alpha^{\sum_{k=1}^{d} i_k 2^{j_k}} \cdot \prod_{r=1}^{d} c_{j_r} \cdot y^{\sum_{s=1}^{d} 2^{j_s}} \\
&= \sum_{\substack{0\le e\le 2^n-1 \\ wt_{bin}(e)\le d}} c_e y^e.
\end{aligned}$$

This shows that

$$\max\{wt_{bin}(e)\,|c_e \neq 0\} \leq \deg(G)\ .$$

Now we consider $\Phi(G) = \mathcal{G} = \sum_e c_e y^e$. As Φ is an injection, there exists by the definition of Φ a unique function $\tilde{G}(\tilde{x}) = \sum_e c_e \tilde{x}^e$ with $\tilde{x} = \varphi_L(X)$ (see (5.4.33)). We use this relation for a bijection between the sets $\{\mathbb{F}_{2^n} \to \mathbb{F}_{2^n}\}$ and $\{\mathbb{F}_2^n \to \mathbb{F}_{2^n}\}$ and to reconstruct from $\mathcal{G} = \sum_e c_e y^e$ a description of G. From Theorem 5.34, we know that φ_L is linear and thus coefficients $\mu_0,\ldots,\mu_{n-1} \in \mathbb{F}_{2^n}$ exist with $\tilde{x} = \varphi_L(X) = \varphi_L(x_0,\ldots,x_{n-1}) = \sum_{i=0}^{n-1}\mu_i \cdot x_i$. As all fields have characteristic 2 and $x_i \in \mathbb{F}_2$, it holds that $\tilde{x}^{2^j} = \varphi_L^{2^j}(x_0,\ldots,x_{n-1}) = \sum_{i=0}^{n-1}\mu_i^{2^j}\cdot x_i$ for all j. This implies for all $e = 2^{e_1} + \ldots + 2^{e_d}$ that

$$\tilde{x}^e = \prod_{k=1}^{d}\tilde{x}^{2^{e_k}} = \prod_{k=1}^{d}\underbrace{(\sum_{i=0}^{n-1}\mu_i^{2^{e_k}}\cdot x_i)}_{\text{linear in } (x_0,\ldots,x_{n-1})} =: P_e(X) \in \mathbb{F}_{2^n}[x_0,\ldots,x_{n-1}]$$

with $\deg P_e \leq d = wt_{bin}(e)$. Hence, we can rewrite $\tilde{G} = \sum_e c_e\tilde{x}^e$ to $P(X) := \sum_e c_e P_e(X)$.

As $P(\boldsymbol{x_0},\ldots,\boldsymbol{x_{n-1}}) = G(\boldsymbol{x_0},\ldots,\boldsymbol{x_{n-1}}) \in \mathbb{F}_2$ for all $(\boldsymbol{x_0},\ldots,\boldsymbol{x_{n-1}}) \in \mathbb{F}_2^n$, $P = G$ must hold. It follows that

$$\deg G = \deg P \leq \max\{wt_{bin}(e)\,|\,c_e \neq 0\}.$$

Altogether, we have showed that for $\Phi(G) = \sum_e c_e y^e$ it holds that

$$\deg(G) = \max\{wt_{bin}(e)|c_e \neq 0\}\ .$$

The claim on $\deg_{X_i}(G)$ can be showed in a similar way. □

5.4.3 Connecting attributes $\mathcal{A}$ to certain sets $\Omega_{\mathcal{A}}$

Next, we will show that finding coefficients $\gamma_0, \ldots, \gamma_T$ such that $\sum \gamma_j G_{t+j}(K)$ has a certain property $\mathcal{A}$ is equivalent to finding $\min(\Omega_{\mathcal{A}}) = \sum \gamma_j x^i \in \mathbb{F}_2[x]$ for corresponding sets $\Omega_{\mathcal{A}}$.

Table 5.4.5 gives an overview of the polynomials considered in this section. These polynomials $\sum_i \gamma_i x^i$ are grouped in pairs, corresponding to the property of $\sum_i \gamma_i G_i(X)$. For example, the coefficients of the polynomials $\min(G)$ and $\min(d)$ cause the sum to be equal to zero, as required for fast algebraic attacks as introduced in [Cou03]. Here, the coefficients from $\min(G)$ accomplish this effect for a specific G, whereas the coefficients from $\min(d)$ work for *any* Boolean function of degree $\leq d$.

As we have seen in Example 5.29 and as mentioned in [HawR04], it is not always necessary for a fast algebraic attack to demand that $\sum_i \gamma_i G_i(X) \equiv 0$. In principle, it suffices that the degree is less than or equal to $d' = \deg(H)$. This corresponds to eliminating all monomials of degree between d' and d. For this purpose, we introduce the polynomials $\min_{(d',d]}(G)$ resp. $m_{(d',d],d}$. Again, the coefficients from the first polynomial are derived for a specific G, whereas the coefficients from the second work for any G of degree $\leq d$. Because of $\deg(\min_{(d',d]}(G)) \leq \deg(\min(G))$ and $\deg(m_{(e,d],d}) \leq \deg(\min(d))$, this can reduce the run length. As both approaches finally give a system of equations, each with a degree $\leq d'$, the effort of solving the system of equations with linearization is asymptotically the same.[7]

The last two polynomials, $\min_{X_i}(G)$ and $\min_{X_i}(d)$, belong to the cases of the divide-and-conquer attacks described in Section 5.3. The coefficients from the first one cause the cancellation of all monomials which have variables from X_i in the case that G is concretely specified. For an unknown G of degree $\leq d$, the coefficients from the second one guarantee this effect.

Now, we come to defining these polynomials more precisely:

Theorem 5.37. *Let* $\Phi(G) = \mathcal{G} = \sum_E c_E Y^E$ *with* $Y^E := y_1^{e_1} \cdot \ldots \cdot y^{e_m}$ *and* $E = (e_1, \ldots, e_m)$. *It holds that*

$$\Phi(G((X_1, \ldots, X_m) \cdot L^i) = \mathcal{G}(\alpha_1^i y_1, \ldots, \alpha_m^i y_m) = \sum_E c_E \alpha^{i \cdot E} Y^E \tag{5.4.34}$$

with $\alpha^{i \cdot E} = \alpha_1^{i \cdot e_1} \cdots \alpha_m^{i \cdot e_m}$. *We define*

$$\Omega_G := \{\alpha^E | c_E \neq 0\},\ \Omega_{(d',d],G} := \{\alpha^E | c_E \neq 0,\ d' < wt_{bin}(E) \leq d\}\ .$$

[7] In particular cases, more different monomials may occur in the second approach. This could result in a slightly higher complexity.

Polynomial $\sum_{j=0}^{T}\gamma_j x^j$	Boolean function G	Property $\mathcal{A}$ of $\sum_{j=0}^{T}\gamma_j G_{t+j}(K)$
$\min(G)$	fixed	equal to zero
$\min(d)$	any with $\deg(G)\leq d$	equal to zero
$\min_{(d',d]}(G)$	fixed with $\deg(G)\leq d$	degree $\leq d'$
$\min_{(d',d]}(d)$	any with $\deg(G)\leq d$	degree $\leq d'$
$\min_{X_i}(G)$	fixed	independent of X_i
$\min_{X_i}(d)$	any with $\deg(G)\leq d$	independent of X_i

Table 5.4.5: The different minimal polynomials discussed in this section.

Let $p(x)=\sum_{j=0}^{T}\gamma_j x^j\in\mathbb{F}_2[x]$ be a Boolean function. Then, we have the following equivalences:

$$\begin{array}{rcll}\sum\gamma_j G_{t+j}(X)\equiv 0 & \iff & p(\alpha)=0 & \forall\alpha\in\Omega_G\\ \deg(\sum\gamma_j G_{t+j}(X))\leq e & \iff & p(\alpha)=0 & \forall\alpha\in\Omega_{(d',d],G}\end{array}\tag{5.4.35}$$

In particular, for the unique minimal polynomials $\min(G):=\min(\Omega_G)$ resp. $\min_{(d',d]}(G):=\min(\Omega_{(d',d],G})$, the values $T'=\deg(\min(G))$ resp. $T=\deg(\min_{(d',d]}(G))$ are the minimum run length. Further on, this means $T\leq T'$.

Proof. Let $p(x)=\sum_{i=0}^{T}\gamma_i x^i$ be an arbitrary polynomial. As Φ is linear, we have for all $t\geq 0$:

$$\begin{aligned}&\Phi(\sum_{i=0}^{T}\gamma_i G_{t+i}(X_1,\ldots,X_m))=\sum_{i=0}^{T}\gamma_i\Phi_{L_i}(G_{t+i}(X_1,\ldots,X_m))\\ =\;&\sum_{i=0}^{T}\gamma_i\left(\sum_E c_E\alpha^{(t+i)\cdot E}Y^E\right)=\sum_E\underbrace{\alpha^{t\cdot E}}_{\neq 0}c_E\underbrace{(\sum_{i=0}^{T}\gamma_i(\alpha^E)^i}_{=p(\alpha^E)}Y^E=\sum_E\alpha^{i\cdot E}c_E p(\alpha^E)Y^E\end{aligned}$$

Let $\hat{G}:=\sum_{i=0}^{T}\gamma_i G_{t+i}$. We have seen that $\Phi(\hat{G})\equiv 0$ if and only if $p(x)$ is zero on the set Ω_G. As Φ is injective, this is equivalent to $\hat{G}\equiv 0$. Furthermore, it follows with Theorem 5.36 that

$$\deg(\sum_{i=0}^{T}\gamma_i G_{t+i}(X_1,\ldots,X_m))=\deg(\hat{G})=\max\{wt_{bin}(E)\,|\,c_E p(\alpha^E)\neq 0\}\ .$$

This shows (5.4.35). The claim that $T'\leq T$ follows from the fact that $\Omega_{(d',d],G}\subseteq\Omega_G$. □

As we have discussed before, instead of the coefficients from $\min(G)$, one can use those from $\min_{(d',d]}(G)$ for a fast algebraic attack in which the degree of the equations is decreased from d to d'. Observe that in both cases, the degree of the new equations is the same, namely d', but the run length may be reduced as $\deg(\min_{(d',d]}(G)) = T' \leq T = \deg(\min(G))$.

In some cases, it may be difficult to calculate the sets Ω_G resp. $\Omega_{(d',d],G}$, or G might be an unknown function of degree d. The following corollary specifies "general" polynomials $\sum \gamma'_j x^j$ and $\sum \gamma_j x^j$ such that (5.4.35) holds for *any* Boolean function G of degree $\leq d$.[8] The proof is similar to the one of Theorem 5.37.

Corollary 5.38. *Let $p_1(x), \ldots, p_m(x) \in \mathbb{F}_2[x]$ be irreducible polynomials of degrees $n_1, \ldots, n_m$, $L = diag(L_1, \ldots, L_m)$ the block matrix of the companion matrices L_i of p_i and $\alpha = (\alpha_1, \ldots, \alpha_m)$, where α_i is a root of p_i. Further on, let $\Omega_d := \{\alpha^E | 0 < wt_{bin}(E) \leq d\}$ resp. $\Omega_{(d',d],d} := \{\alpha^E | d' < wt_{bin}(E) \leq d\}$. Let $p(x) := \sum_{j=0}^{T} \gamma_j x^j$. Then, the following equivalences are true for* any *Boolean function G of degree $\leq d$:*

$$\begin{array}{rcll} \sum_j \gamma_j G_{t+j}(X) = 0 & \iff & p(\alpha) = 0 & \forall \alpha \in \Omega_d \\ \deg(\sum_j \gamma_j G_{t+j}(X)) \leq d' & \iff & p(\alpha) = 0 & \forall \alpha \in \Omega_{(d',d],d} \end{array}$$

Therefore, the minimal polynomials $\min(d)$ and $\min_{(d',d]}(G)$ of the sets Ω_d and $\Omega_{(d',d],G}$ specify the minimum run length and the coefficients γ_i.

So far, we proved that optimal coefficients γ_i for reducing the degree can be derived by computing the minimal polynomials of certain sets. The following theorem shows that the same is possible for the divide-and-conquer attack described in Section 5.3. Also here is the proof similar to the proof of Theorem 5.37.

Theorem 5.39. *Let $p_1(x), \ldots, p_m(x) \in \mathbb{F}_2[x]$ be irreducible polynomials of degrees $n_1, \ldots, n_m$, $L \in GL_n(\mathbb{F}_2)$ the block matrix of the companion matrices of all p_i and $\alpha = (\alpha_1, \ldots, \alpha_m)$, where α_i is a root of p_i. Let $G \in \mathbb{F}_2[X_1, \ldots, X_m]$ and $\Phi(G) =: \mathcal{G} = \sum_e c_E Y^E$. We set $\Omega_{X_i,G} := \{\alpha^E = \alpha_1^{e_1} \cdot \ldots \cdot \alpha_m^{e_m} | c_E \neq 0, e_i \neq 0\}$. Then for $p(x) := \sum_{j=0}^{T} \gamma_j x^j$, we have the equivalence:*

$$\deg_{X_i} \left(\sum_{j=0}^{T} \gamma_j G_{t+j}(X) \right) = 0 \iff p(\alpha) = 0 \quad \forall \alpha \in \Omega_{X_i,G} \ .$$

The left side means that $\sum_{j=0}^{T} \gamma_j G_{t+j}(X)$ is independent of X_i.

[8] Of course, it does no longer guarantee the minimality of T' and T.

Furtheron, let $\Omega_{X_i,d} := \{\alpha^e | wt_{bin}(e) \leq d, e_i \neq 0\}$. *Then for any Boolean function* $G \in \mathbb{F}_2[X_1, \ldots, X_m]$ *of degree* $\leq d$, *we have the equivalence:*

$$\deg_{X_i}\left(\sum_{j=0}^{T} c_j G_{t+j}(X)\right) = 0 \iff p(\alpha') = 0 \quad \forall \alpha' \in \Omega_{X_i,d}.$$

Again do the minimal polynomials $\min_{X_i}(G)$ *resp.* $\min_{(d',d]}(G)$ *of the sets* $\Omega_{X_i,G}$ *resp.* $\Omega_{X_i,d}$ *specify the minimum run length and the coefficients* γ_i.

5.4.4 Efficient precomputation steps with the minimum run length

In the previous section, we showed that finding the appropriate linear combination for the precomputation steps in different kind of fast algebraic attacks is equivalent to computing the minimal polynomial of certain sets. Crucial for the applicability of our approaches is that this can be done efficiently, which will be demonstrated in this section. First, we show how to express the polynomials from Table 5.4.5 by other polynomials. The following fact is obvious resp. well known:

Lemma 5.40. *Let* $\Omega, \Omega' \subseteq \mathbb{F}_{2^n}$. *Then* $\min(\Omega \cap \Omega') = \gcd(\min(\Omega), \min(\Omega'))$. *If* $\min(\Omega)$ *and* $\min(\Omega')$ *are co-prime, then* $\min(\Omega \cup \Omega') = \min(\Omega) \cdot \min(\Omega')$.

Theorem 5.41. *Let* $f(x) \in \mathbb{F}_2[x]$ *be an irreducible polynomial of degree* n *and* $\alpha, \alpha' \in \mathbb{F}_{2^n}$ *be two roots. Then* $\alpha = (\alpha')^{2^k}$ *for an appropriate* $k \in \{0, \ldots, n-1\}$.

Proof. We show that for a root α, α^2 is a root as well. Then, the rest follows by induction and a counting argument.

As $\mathbb{F}_{2^n}$ has the characteristic 2, the so-called "Freshman's rule" is applicable here: $g(\beta)^2 = g(\beta^2)$ for all $g(x) \in \mathbb{F}_2[x]$ and $\beta \in \mathbb{F}_{2^n}$. This means for f and its root α that $0 = f(\alpha) = f(\alpha)^2 = f(\alpha^2)$. Therefore, α^2 is a root of f. □

Theorem 5.42. *Let the involved polynomials* $p_1, \ldots, p_m$ *be pairwise co-prime with degrees* $n_1, \ldots, n_m$. *In addition, we assume that for* $\alpha^E = \prod_{i=1}^m \alpha_i^{e_i}$,$\alpha^{E'} = \prod_{i=1}^m \alpha_i^{e_i'} \in \Omega_G$ *it holds that*[9]

$$\alpha^E = \alpha^{E'} \iff e_i = e_i' \ \forall i \ . \tag{5.4.36}$$

Then, the minimal polynomials can be computed by

$$\begin{aligned} \min_{(d',d]}(G) &= \tfrac{\min(G)}{\gcd(\min(d'),\min(G))} \\ \min_{X_i}(G) &= \gcd(\min(G), \min_{X_i}(d)) \ . \end{aligned} \tag{5.4.37}$$

[9] We want to point out that this is exactly the weaker condition for the correctness of Algorithm 5.2 derived in Theorem 5.20.

Proof. First, we observe that $\Omega_{(d',d],G} \cup (\Omega_{d'} \cap \Omega_G) = \Omega_G$. Further on, we claim that $\min_{(d',d]}(G)$ and $\min(\Omega_{d'} \cap \Omega_G)$ are co-prime. Then it follows with Lemma 5.40 that

$$\begin{aligned} \min_{(d',d]}(G) \cdot \gcd(\min(d'), \min(G)) &= \min_{(d',d]}(G) \cdot \gcd(\min(\Omega_{d'}), \min(\Omega_G)) \\ &= \min(\Omega_{(d',d],G}) \cdot \min(\Omega_{d'} \cap \Omega_G) \\ &= \min(\Omega_{(d',d],G} \cup (\Omega_{d'} \cap \Omega_G)) \\ &= \min(\Omega_G) = \min(G) \ , \end{aligned}$$

which yields the first claim.

What is left is to show that $\min_{(d',d]}(G)$ and $\min((\Omega_{d'} \cap \Omega_G)$ are co-prime. Assume that this is not the case. Then there exists an irreducible polynomial $f \not\equiv 1$ which divides $\min_{(d',d]}(G)$ and $\min((\Omega_{d'} \cap \Omega_G)$. As $\min_{(d',d]}(G)$ and $\min((\Omega_{d'} \cap \Omega_G)$ are minimal, there exists $\alpha^E \in \Omega_{(d',d],G}$ and $\alpha^{E'} \in \Omega_{d'} \cap \Omega_G$ with $f(\alpha^E) = f(\alpha^{E'}) = 0$ and $wt_{bin}(E') \leq d' < wt_{bin}(E)$. This shows that $\alpha^E \neq \alpha^{E'}$. Due to Theorem 5.41 and the irreducibility of f, there exists an integer $k \geq 1$ such that $\left(\alpha^E\right)^{2^k} = \alpha^{E'}$. Let $E = (e_1, \ldots, e_m)$ and $E' = (e'_1, \ldots, e'_m)$. We have

$$\left(\alpha^E\right)^{2^k} = \alpha_1^{2^k \cdot e_1} \cdots \alpha_m^{2^k \cdot e_m} = \alpha_1^{e'_1} \cdots \alpha_m^{e'_m} \ .$$

By assumption, it must hold that $2^k \cdot e_i = e'_i$ for $i = 1, \ldots, m$, which yields the contradiction

$$d' < wt_{bin}(E) = \sum wt_{bin}(e_i) = \sum wt_{bin}(2^k e_i) = \sum wt_{bin}(e'_i) = wt_{bin}(E') \leq d' \ .$$

Thus, such a function f cannot exist, which shows that the two minimal polynomials are co-prime.

The second claim is true because of $\Omega_{X_i,G} = \Omega_G \cap \Omega_{X_i,d}$ and Lemma 5.40. □

Now we argue why the expressions on the right side of (5.4.37) can be computed efficiently. Let $D'_n := \sum_{i=0}^{d'} \binom{n}{i}$ and $D_n := \sum_{i=0}^{d} \binom{n}{i}$. $\min(d')$ can be constructed by the method explained in [HawR04, Section 6] with an effort in $\mathcal{O}(D'_n[n(\log_2 n)^2 + (\log_2 D'_n)^3])$. $\min(G)$ can be computed with the methods described so far. In any case, the effort is at most the effort for the Berlekamp-Massey algorithm, which is at most $\mathcal{O}(D_n^2)$. The effort for gcd-computation and division of two polynomials of degree $\leq D_n$ over $\mathbb{F}_2$ are in $\mathcal{O}(M(D_n)log(D_n))$ and $\mathcal{O}(M(D_n))$ respectively [AhoHU74]. Hereby, $M(D_n)$ denotes the effort of computing the product of two polynomials of degree D_n which is in $\mathcal{O}(D_n \log D_n \log\log D_n)$ [Sch77]. Because of $D'_n \leq D_n$, The overall effort of computing $\min_{(d',d]}(G)$ is in

$$\mathcal{O}(\underbrace{D'_n[n(\log_2 n)^2 + (\log_2 D'_n)^3]}_{\text{Comp. } \min(d')} + \underbrace{D_n^2}_{\text{Comp. } \min(G_t(X))} + \underbrace{(D_n \log D_n \log\log D_n)(1 + \log D_n))}_{\text{Comp. gcd and div.}} \ .$$

In the case of the divide-and-conquer attack, the computation of the minimal polynomial $\min_{X_i}(G)$ is even easier. First, the algorithm from [HawR04, Section 6] can be easily adapted to compute $\min_{X_i}(d)$ (just choose appropriate Ψ). Similarly to above, we can argue that the effort for computing $\min_{X_i}(G)$ is in

$$\mathcal{O}(\underbrace{D_n[n(\log_2 n)^2 + (\log_2 D_n)^3]}_{\text{Comp. } \min_{X_i}(d)} + \underbrace{D_n^2}_{\text{Comp. } \min(G_t(X))} + \underbrace{D_n \log^2 D_n \log\log D_n}_{\text{Comp. gcd}}) \ .$$

Summing up, it is possible to compute efficiently the minimal polynomials described in the previous section. As the coefficients of these monomials are exactly the coefficients of the linear combinations used in the precomputation steps, both attacks are feasible with a minimum run length. Compared to the methods where the run length is not minimized, the effort is only slightly more. The asymptotic effort is even the same.

To get a feeling on the amount of benefit when minimizing the run length, we consider the degree-decreasing fast algebraic attack from [Cou03]. Recall that E_0 uses four LFSRs A, B, C and D of lengths $n_a = 25$, $n_b = 31$, $n_a = 33$, and $n_a = 39$, We denote their outputs at clock t by a_t, b_t, c_t and d_t and use the 4-function $F = G + H$ derived in [ArmK03, Cou03] (or see pages 100ff). The corresponding G is $G = \sigma_t^4 + \sigma_t^2 \cdot \sigma_{t+1}^2$, where $\sigma_t^4 = a_t b_t c_t d_t$ and $\sigma_t^2 = a_t b_t + a_t c_t + a_t d_t + b_t c_t + b_t d_t + c_t d_t$ (see end of Section 5.2.2, pages 152ff). By Theorem 5.42, it is $\deg(\min_{(d',d]}(G)) = \deg(\min(G)) - \deg(gcd(\min(d'), \min(G)))$. It was estimated in [Arm04a] that $\deg(\min(G)) = 8,822,188$ (also end of Section 5.2.2). Furtheron, an upper bound for $\deg(gcd(\min(d'), \min(G)))$ is the number of monomials of degree ≤ 3 which can occur in the system of equations given by G. We calculated this number to be $\approx 326,080$. This reduces the number of successive bits required for one equation of degree 3 to $\approx 8,496,108$. The effort to calculate $\min_{(d',d]}(G))$ is $\approx 2^{47}$ which is negligible compared to the attack effort of $\approx 2^{54}$ [Cou03].

A general estimation for the advantage is the following. Let $\min(G) = \sum_{i=0}^{T} \gamma_i x^i$ and $\min_{(d',d]}(G) = \sum_{j=0}^{T'} \gamma'_j x^j$. That is, the γ_i are chosen such that all monomials in the sum $\sum_i \gamma_i G_i(X)$ are eliminated, whereas in the sum $\sum_j \gamma'_j G_j(X)$, only these monomials are canceled with a degree in $(d', d]$. A general upper bound for the number of monomials with degree $\leq d$ is $\binom{n}{0} + \ldots + \binom{n}{d}$, whereas at most $\binom{n}{d'+1} + \ldots + \binom{n}{d}$ different monomials with a degree between d' and d exist. Thus, a rough estimation for T and T' is given by

$$T \leq \binom{n}{0} + \ldots + \binom{n}{d}, \quad T' \leq \binom{n}{d'+1} + \ldots + \binom{n}{d} \ .$$

Therefore, in the extreme case it is possible to reduce the run length by a value of $\binom{n}{0} + \ldots + \binom{n}{d'}$.

	n	8	9	10	11	12
$\min(G)$	T	161,2	255	384,8	561	793
	$\#\{\gamma_j \neq 0\}$	84,3	132,6	195,2	278,4	413,6
$\min_{(d',d]}(G)$	T'	71	126	210	330	495
	$\#\{\gamma'_j \neq 0\}$	31	61,3	102,1	159,1	243
	n	13	14	15	16	
$\min(G)$	T	1092	1469,2	1940	2516	
	$\#\{\gamma_j \neq 0\}$	543	741,7	963,4	1253,5	
$\min_{(d',d]}(G)$	T'	715	1001	1365	1820	
	$\#\{\gamma'_j \neq 0\}$	351,7	502,3	677,6	917	

Table 5.4.6: Simulation results for fast algebraic attacks based on the coefficients from $\min(G)$ and $\min_{(d',d]}(G)$

Inspired by the example of E_0 where $d' = 3$ and $d = 4$, we have implemented some simulations to compare the average values of T with T' and the average number of non-zero coefficients γ'_j and γ_j. Observe that the last values reflect how many equations have to be summed up. The examples are based on one LFSR and a fixed random G which is the sum of monomials of degree 4. The results can be found in Table 5.4.6.

5.5 Immunity of simple combiners against fast algebraic attacks

In this final section of the chapter on fast algebraic attacks, we restate our results from [ArmCGKMR06] on the immunity of simple combiners against fast algebraic attacks. More precisely, we consider the following task: Given a simple combiner with output function f, does a 1-function $F(X,z)$ exist such that

$$F(X,z) = \underbrace{G(X)}_{\deg=d} + \underbrace{H(X,z)}_{\deg=d'}$$

with $d > d'$. By Corollary 4.19, any 1-function $F(X,z)$ for a simple combiner has the form $z\cdot(f\cdot(h_0+h_1)+h_1)+h_0\cdot f$ with h_0, h_1 arbitrary Boolean functions. We will consider only the special case[10] $h := h_0 = h_1$, which leads to

$$F(X,z) = h(X)\cdot(f(X)+z) = \underbrace{h(X)\cdot f(X)}_{=:G(X)} + \underbrace{h(X)\cdot z}_{=:H(X,z)}.$$

Observe that the expression $G(X)$ and $H(X,z)$ are chosen such that they fit the notations defined in Section 5.1. The question will be, given a specific function $f(X)$, do functions $G(X)$ and $h(x)$ exist of degree d and d', respectively. First, we provide some basic facts on different representations of Boolean functions.

5.5.1 Boolean Functions

We have already seen in Section 2.3 that any Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ can be represented by its algebraic normal form $\sum_{E\in\{0,1\}^n} f_E X^E$. From now on, we will identify each $E = (e_1,\ldots,e_n) \in \{0,1\}^n$ with the integer $E = \sum_{i=0}^{n-1} e_{n-i}2^i$. Then, we can derive from the algebraic normal form the **coefficients vector**

$$C(f) := (f_0,\ldots,f_{2^n-1}). \tag{5.5.38}$$

An alternative way of characterizing f is its **truth table** $T(f)$, which is defined by

$$T(f) := (f(0),\ldots,f(2^n-1)) \ . \tag{5.5.39}$$

[10] Fast algebraic attacks based on the general form as given in Corollary 4.19 has not been examined yet and remains open for further research. However, we expect due to first simulations and internal discussions [KueM06] that the general form allows better attacks only in rare situations.

Example 5.43. *Consider the Boolean function*

$$\begin{aligned} f(x_1, x_2) := x_1 + x_1x_2 &= f_{(0,0)} \cdot 1 + f_{(0,1)} \cdot x_2 + f_{(1,0)} \cdot x_1 + f_{(1,1)} \cdot x_1x_2 \\ &= f_0 \cdot 1 + f_1 \cdot x_2 + f_2 \cdot x_1 + f_3 \cdot x_1x_2 \ . \end{aligned}$$

The coefficients vector of f is $C(f) = (0, 0, 1, 1)$. In addition, one can easily compute that $f(0, 0) = f(0, 1) = f(1, 1) = 0$ and $f(1, 0) = 1$, so the truth table is given by $T(f) = (0, 0, 1, 0)$.

Observe that $T(f + g) = T(f) + T(g)$ and $T(f \cdot g) = T(f) \cdot T(g)$ (defined componentwise). For the coefficients vector, $C(f + g) = C(f) + C(g)$ also holds, but $C(f \cdot g) \neq C(f) \cdot C(g)$. Next, we show how to transform $C(f)$ into $T(f)$ and vice versa. We will use for binomial coefficients $\binom{a}{b}$ the extended definition $\binom{a}{b} = 0$ for $b > a$ (and $a, b > 0$) or $b < 0$ (and $a \geq 0$). Binomial coefficients modulo 2 can be implemented very efficiently, using the fact that $\binom{a}{b} \mod 2 = 1$ if and only if $(\sim a) \wedge b = 0$ (for $a, b > 0$). Recall Lucas' Theorem which says that $\binom{a}{b} \mod 2 = \prod \binom{a_i}{b_i}$ where a_i and b_i are the binary expression of a and b, respectively.

Definition 5.44. *Let $\boldsymbol{X} := (\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}) \in \mathbb{F}^n$. The **support** of $\boldsymbol{X}$ is defined to be the set $\operatorname{supp}(\boldsymbol{X}) := \{i | \boldsymbol{x_i} = 1\}$ (recall Definition 4.9). Observe that $|\boldsymbol{X}| = |\operatorname{supp}(\boldsymbol{X})|$. Furthermore, each $\boldsymbol{X}$ is characterized by its support and vice versa. For an integer $x = \sum_{i=0}^{n} x_i \cdot 2^i$ with $x_i \in \{0, 1\}$, we define its **support** by $\operatorname{supp}(x) := \{i | x_i = 1\}$.*

We have the following equivalence:

$$\binom{a}{b} \mod 2 = 1 \iff \forall i : b_i = 1 \Rightarrow a_1 = 1 \iff \operatorname{supp}(b) \subseteq \operatorname{supp}(a) \ . \quad (5.5.40)$$

Definition 5.45. *For $N \geq 1$, we define recursively the **Hadamard matrix** of size $N \times N$. For $N = 1$, we set $H_1 := (1)$. For $N = 2^{n+1}$, we define $H_{2^{n+1}} = \begin{pmatrix} H_{2^n} & 0 \\ H_{2^n} & H_{2^n} \end{pmatrix}$. For intermediate values $2^n < N < 2^{n+1}$, we define H_N to be the submatrix of $H_{2^{n+1}}$ which consists of the first N rows and columns.*

Example 5.46. *The matrices H_N for $1 \leq N \leq 4$ read*

$$H_1 = (1), \ H_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \ H_3 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \ H_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \ .$$

Lemma 5.47. *H_N is a lower triangular, self-inverse matrix. Furthermore, it holds that $H_N(i, j) = \binom{i}{j} \mod 2$ for all (i, j) with $0 \leq i, j \leq N-1$, where $H_N(i, j)$ is the entry of H_N at position (i, j).*

Proof. That H_N is a lower triangular matrix follows directly from the definition. We show that it is self-inverse by induction over N. For $N = 1$, it is obvious that $H_1 \cdot H_1 = I_1$, where I_n denotes the unitary matrix of size $n \times n$. Next, we assume that the claim is true for $N = 2^n \geq 1$. Then

$$\begin{aligned} H_{2^{n+1}} \cdot H_{2^{n+1}} &= \begin{pmatrix} H_{2^n} & 0 \\ H_{2^n} & H_{2^n} \end{pmatrix} \cdot \begin{pmatrix} H_{2^n} & 0 \\ H_{2^n} & H_{2^n} \end{pmatrix} \\ &= \begin{pmatrix} H_{2^n} \cdot H_{2^n} & 0 \\ H_{2^n} \cdot H_{2^n} + H_{2^n} \cdot H_{2^n} & H_{2^n} \cdot H_{2^n} \end{pmatrix} \\ &= \begin{pmatrix} I_{2^n} & 0 \\ 0 & I_{2^n} \end{pmatrix} = I_{2^{n+1}} \end{aligned}$$

Finally, let $2^n < N < 2^{n+1}$. Then, it is

$$\begin{aligned} \begin{pmatrix} I_N & 0 \\ 0 & I_{2^{n+1}-N} \end{pmatrix} &= I_{2^{n+1}} = H_{2^{n+1}} \cdot H_{2^{n+1}} \\ &= \begin{pmatrix} H_N & 0 \\ * & * \end{pmatrix} \cdot \begin{pmatrix} H_N & 0 \\ * & * \end{pmatrix} = \begin{pmatrix} H_N \cdot H_N & 0 \\ * & * \end{pmatrix}. \end{aligned}$$

This shows that $H_N \cdot H_N = I_N$.

For the claim $H_N(i,j) = \binom{i}{j} \mod 2$, we assume w.l.o.g. that $N = 2^n$ and conduct the proof again by induction. For $N = 1$ and $N = 2$ the statement can be easily checked. Now consider H_{2N}. By definition, H_{2N} has the form $H_{2N} = \begin{pmatrix} H_N & 0 \\ H_N & H_N \end{pmatrix}$. We define the matrix $\tilde{H} = \begin{pmatrix} \tilde{H}_0 & \tilde{H}_1 \\ \tilde{H}_2 & \tilde{H}_3 \end{pmatrix}$ of size $2N \times 2N$ by $\tilde{H}(i,j) := \binom{i}{j} \mod 2$ and show that $H_{2N} = \tilde{H}$ "corner by corner".

First of all, it is $\tilde{H}_0 = H_N$ by assumption. Further on, it is $\tilde{H}_1 = 0$ as $\binom{i}{j} = 0$ for $j > i$. Finally, we use the facts that $\binom{i+2^n}{j} \mod 2 = \binom{i}{j} \mod 2$ and $\binom{i+2^n}{j+2^n} \mod 2 = \binom{i}{j} \mod 2$, which is a consequence of Luca's Theorem. This shows that $\tilde{H}_2 = \tilde{H}_0 = H_N$ and $\tilde{H}_3 = \tilde{H}_0 = H_N$, respectively. □

Now, we are ready to connect the expressions $C(f)$ and $T(f)$.

Theorem 5.48. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be an arbitrary Boolean function. Then,*

$$C(f) = H_{2^n} \cdot T(f) \quad \textit{and} \quad T(f) = H_{2^n} \cdot C(f)\,. \tag{5.5.41}$$

Proof. Set $N := 2^n$. Due to Lemma 5.47, H_N is self-inverse and therefore both equations are equal. Hence, it suffices to show that $C(f) = H_N \cdot T(f)$. We prove the claim by induction on n. Let $n = 1$. Then $f(x) = f(0) + x \cdot (f(0) + f(1))$ and therefore $C(f) = (f(0), f(0) + f(1))$. This is obviously equal to $H_2 \cdot T(f)$.

Example 5.49. *Consider the function $f(x_1, x_2) = x_1 + x_1x_2$ from Example 5.43. We determined that $C(f) = (0, 0, 1, 1)$ and $T(f) = (0, 0, 1, 0)$. One can easily check that*

$$H_4 \cdot C(f) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = T(f)$$

and vice versa.

Now assume that the assumption is true for $n \geq 1$. Let $f \in \mathbb{F}[x_1, \ldots, x_{n+1}]$ be an arbitrary Boolean function. We set

$$f_0(x_2, \ldots, x_{n+1}) := f(0, x_2, \ldots, x_{n+1}) \text{ and } f_1(x_2, \ldots, x_{n+1}) := f(1, x_2, \ldots, x_{n+1}).$$

It is $f = f_0 + x_1 \cdot (f_0 + f_1)$. This implies that

$$\begin{aligned} C(f) &= \begin{pmatrix} C(f_0) \\ C(f_0) + C(f_1) \end{pmatrix} \overset{\text{assump.}}{=} \begin{pmatrix} H_N \cdot T(f_0) \\ H_N \cdot T(f_0) + H_N \cdot T(f_1) \end{pmatrix} \\ &= \begin{pmatrix} H_N & 0 \\ H_N & H_N \end{pmatrix} \cdot \begin{pmatrix} T(f_0) \\ T(f_1) \end{pmatrix} = H_{2^{n+1}} \cdot T(f) \ . \end{aligned}$$

□

5.5.2 How to set up Equations for Arbitrary Functions

Setting up Equations

As said at the beginning of the section, we consider three Boolean functions $f(X)$, $G(X)$ and $h(X)$ with $G(X) = f(X) \cdot h(X)$. Notwithstanding the previous notations, we will express the coefficients of their algebraic normal forms by Greek letters:

$$f(X) = \sum_{\alpha \in \mathbb{F}_2^n} f_\alpha X^\alpha, \quad h(X) = \sum_{\beta \in \mathbb{F}_2^n} h_\beta X^\beta, \quad G(X) = \sum_{\gamma \in \mathbb{F}_2^n} G_\gamma X^\gamma.$$

If not otherwise stated, it will be $\alpha = (\alpha_1, \ldots, \alpha_n)$, $\beta = (\beta_1, \ldots, \beta_n)$ and $\gamma = (\gamma_1, \ldots, \gamma_n)$. We introduce a semi-ordering on these coefficients:

$$\beta \subseteq \gamma \iff \text{supp}(\beta) \subseteq \text{supp}(\gamma).$$

We will also use the notation $\alpha \vee \beta := (\alpha_1 \vee \beta_1, \ldots, \alpha_n \vee \beta_n)$. With the next proposition, we are able to express a single coefficient G_γ in f and h only.

Proposition 5.50. *Let* $f(X) = \sum_\alpha f_\alpha X^\alpha$ *and* $h(X) = \sum_\beta h_\beta X^\beta$. *Set* $G(X) := \sum_\gamma G_\gamma X^\gamma := f(X) \cdot h(X)$. *Then, it is for each* γ

$$G_\gamma = \sum_{(\alpha,\beta):\alpha\vee\beta=\gamma} f_\alpha \cdot h_\beta \ . \tag{5.5.42}$$

Proof. We first show that the following statement holds for the product of two monomials. Because of the field equation $x^2 = x$, it holds for $\alpha, \beta \in \mathbb{F}^n$:

$$X^\alpha \cdot X^\beta = \prod_{i=1}^{n} x_i^{\alpha_i+\beta_i} \overset{\text{field eq.}}{=} \prod_{i=1}^{n} x_i^{\max\{\alpha_i,\beta_i\}} = X^{\alpha\vee\beta}.$$

With this preparation, we can conclude:

$$\begin{aligned} G(X) &= \sum_\gamma G_\gamma X^\gamma \overset{def}{=} f(X)\cdot h(X) = \Big(\sum f_\alpha \cdot X^\alpha\Big)\cdot\Big(\sum h_\beta \cdot X^\beta\Big) \\ &= \sum_{(\alpha,\beta)} f_\alpha \cdot h_\beta \cdot X^{\alpha\vee\beta} = \sum_\gamma \left(\sum_{(\alpha,\beta):\alpha\vee\beta=\gamma} f_\alpha \cdot h_\beta\right) X^\gamma. \end{aligned}$$

As the algebraic normal form of G is unique, the equation immediately implies the claim.

□

Based on this proposition and on the Hadamard transformation introduced in Section 5.5.1, we find another representation of G_γ in the next Theorem. As a notational convention, if V and $C(f)$ (resp. V and $T(f)$) are Boolean vectors of identical size, a product of type $V\cdot C(f)$ (resp. $V\cdot C(f)$) will denote a scalar product $\sum_{k=0}^{n}[V]_k \cdot [C(f)]_k$ where $[V]_k$ denotes the k^{th} entry of V.

Theorem 5.51. *Let* $f(X) = \sum_\alpha f_\alpha X^\alpha$ *and* $h(X) = \sum_\beta h_\beta X^\beta$. *Set* $G(X) := \sum_\gamma G_\gamma X^\gamma := f(X)\cdot h(X)$. *Let* $A_{\gamma,\beta}$ *denote an element in* $\mathbb{F}_2$, *and* $\bar{B}_{i,j}$ *a vector of size* 2^n *over* $\mathbb{F}_2$. *Then, it is for each* γ

$$G_\gamma = \sum_{\beta\subseteq\gamma} A_{\gamma,\beta}\cdot h_\beta \quad \text{with} \tag{5.5.43}$$

$$A_{i,j} := \bar{B}_{i,j}\cdot T(f) = \bar{B}_{i,i-j}\cdot C(f) \quad \text{and} \tag{5.5.44}$$

$$\left[\bar{B}_{i,j}\right]_k := \binom{i}{k}\cdot\binom{k}{j} \mod 2 \ . \tag{5.5.45}$$

Proof. Starting from (5.5.42), we can set up the following equation for G_γ:

$$G_\gamma = \sum_{(\alpha,\beta):\alpha\vee\beta=\gamma} f_\alpha \cdot h_\beta = \sum_{\beta\subseteq\gamma} \underbrace{\left(\sum_{\alpha\subseteq\gamma:\alpha\vee\beta=\gamma} f_\alpha \right)}_{=:A_{\gamma,\beta}} \cdot h_\beta.$$

Next, we prove that $[\bar{B}_{\gamma,\gamma-\beta}]_\alpha = 1$ if and only if $\alpha \subseteq \gamma$ and $\alpha \vee \beta = \gamma$. Then, this shows that $A_{\gamma,\beta} = \bar{B}_{\gamma,\gamma-\beta} \cdot C(f)$. First, one can rewrite the conditions $\alpha \subseteq \gamma$ and $\alpha \vee \beta = \gamma$ for $\beta \subseteq \gamma$ to

$$\mathrm{supp}(\gamma) \setminus \mathrm{supp}(\beta) \subseteq \mathrm{supp}(\alpha) \subseteq \mathrm{supp}(\gamma). \tag{5.5.46}$$

Observe that $\beta \subseteq \gamma$ implies that $\mathrm{supp}(\gamma) \setminus \mathrm{supp}(\beta) = \mathrm{supp}(\gamma - \beta)$ as no carries occur in the subtraction.

Due to (5.5.40), (5.5.46) is true if and only if $\binom{\gamma}{\alpha} \mod 2 = 1$ and $\binom{\alpha}{\gamma-\beta} \mod 2 = 1$. This is equivalent to $1 = \binom{\gamma}{\alpha}\binom{\alpha}{\gamma-\beta} \mod 2 = [\bar{B}_{\gamma,\gamma-\beta}]_\alpha$.

Finally, we show that $A_{\gamma,\beta} = \bar{B}_{\gamma,\beta} \cdot T(f)$. It holds that

$$C(G) = C(f \cdot h) = H_{2^n} \cdot T(f \cdot h) = H_{2^n} \cdot (T(f) * T(h)) = H_{2^n} \cdot (T(f) * (H_{2^n} \cdot C(h))),$$

where $*$ denotes the componentwise product of vectors. Notice that for a matrix M and vectors V and W, it holds that $V * (M \cdot W) = (V * M) \cdot W$, where $V * M$ means that the i-th row of M is multiplied with the i-th entry of V. As $H_{2^n}(i,j) = \binom{i}{j} \mod 2$, it is $C(G) = A(f) \cdot C(h)$ with $A(f) := H_{2^n} \cdot (T(f) * H_{2^n})$. It follows that

$$\begin{aligned} A_{\gamma,\beta} &= [A(f)]_{\gamma,\beta} = [H_{2^n} \cdot (T(f) * H_{2^n})]_{\gamma,\beta} = \sum_\alpha [H_{2^n}]_{\gamma,\alpha} \cdot [T(f) * H_{2^n}]_{\alpha,\beta} \\ &= \sum_\alpha \binom{\gamma}{\alpha}\binom{\alpha}{\beta} f(\alpha) \mod 2 = \bar{B}_{\gamma,\beta} \cdot T(f) \ . \end{aligned}$$

□

With this theorem, the coefficients $A_{i,j}$ can be computed either with the coefficients vector of f or with the truth table of f. Notice that the vector $\bar{B}_{i,j}$ is independent of f. We give an example how to set up an equation.

Example 5.52. *Let us set up an equation for $\gamma = (1,0,1) \in \mathbb{F}_2^3$. Consequently, (5.5.43) results to*[11]

$$G_{(101)} = A_{(101),(000)} \cdot h_{(000)} + A_{(101),(001)} \cdot h_{(001)} + A_{(101),(100)} \cdot h_{(100)} + A_{(101),(101)} \cdot h_{(101)} \ .$$

[11] For the sake of brevity, we abbreviate $(1,0,1)$ to (101), etc.

If we use the Equations (5.5.44) and (5.5.45), we find

$$\begin{aligned}
A_{(101),(000)} &= f_{(101)} = f(000) + f(100) + f(001) + f(101) \\
A_{(101),(100)} &= f_{(001)} + f_{(101)} = f(100) + f(101) \\
A_{(101),(001)} &= f_{(100)} + f_{(101)} = f(001) + f(101) \\
A_{(101),(101)} &= f_{(000)} + f_{(100)} + f_{(001)} + f_{(101)} = f(101) \ .
\end{aligned}$$

Determining the Existence of Solutions

We consider now the question whether a function $h(x)$ of degree $\leq d'$ exists such that $G(X) = f(x) \cdot h(X)$ is of degree $\leq d$ or not. Theorem 5.51 provides all information necessary to rewrite the product $G(X) = f(x) \cdot h(X)$ by a system of linear equations in the coefficients G_γ and h_β. Because of our assumptions that $\deg(G) \leq d$ and $\deg(h) \leq d'$, it holds that $G_\gamma = 0$ for $|\gamma| > d$ and similarly $h_\beta = 0$ for $|\beta| > d'$. Thus, equation (5.5.43) yields homogeneous equations in h_β for $|\gamma| > d'$.

Let $D' := \mu_2(n, d') = \sum_{i=0}^{d'} \binom{n}{i}$ and $D := \mu_2(n, d) = \sum_{i=0}^{d} \binom{n}{i}$ denote the number of monomials of degree $\leq d'$ and $\leq d$, respectively. Because of $\deg(h) \leq d'$, the number of non-zero coefficients in the algebraic normal form of h is bounded by D'. Therefore, considering about D' equations in h_β as described above should give an answer whether appropriate h and G exist and what they look like. This is formalized in Algorithm 5.5. Here, each column represents one coefficient h_β and each row one equation $0 = \sum_{\beta \subseteq \gamma} A_{\gamma,\beta} \cdot h_\beta$ for a γ with $d < |\gamma| \leq d' + \deg(f)$. Observe that $f \cdot h$ has no terms of degree greater than $\deg(f \cdot h)$. Therefore, considering γ with $|\gamma| > d' + \deg(f) \geq \deg(f \cdot h)$ would not yield any constraints on h_β.

Algorithm 5.5.
Finding a 1-function for a fast algebraic attack on a simple combiner

Input: *A Boolean function f of dimension n, degrees $\deg h = d'$ and $\deg G = d$.*

Output: *Determine if G and h exist such that $f \cdot h = G$.*

1: *Initialize a $D' \times D'$ matrix M, and let each entry be zero.*
2: *Choose an arbitrary ordering on $\{\beta : |\beta| \leq d'\}$ and index the columns of M accordingly.*
3: ***for*** *i from 1 to D'* ***do***
4: *Choose a random γ with weight $d < |\gamma| \leq d' + \deg(f)$.*
5: *Determine the set $\mathcal{B} = \{\beta : \beta \subseteq \gamma,\ |\beta| \leq d'\}$.*
6: ***for all*** *β in $\mathcal{B}$* ***do***
7: *Let the entry of M in row i and column β be $A_{\gamma,\beta}$.*
8: ***end for***
9: ***end for***
10: *Solve the linear system of equations, and output* `no` *G* `and` *h* `of corresponding degree` *if there is only a trivial solution.*

If the rows in Algorithm 5.5 are not linearly independent, this does not imply the existence of functions G and h of appropriate degree. Other choices of γ may have produced additional linear independent rows in M, which would show that M has no non-trivial nullvectors. However, one can argue theoretically that, given a set S of linear independent vectors which is not a basis, if one chooses a random vector V, it is more likely that V is linear independent to S (see also (4.3.31) on page 110 and the corresponding explanations). Similarly, one can expect that if one repeats the loop of Algorithm 5.5 at little more than D' times, one gets with high probability a linear system of equations of full rank if no such functions G and h exist. If non-trivial solutions exist, one can easily check afterwards if $\deg(f \cdot h) = d$ to exclude "wrong solutions".

Example 5.53. *Recall the majority function* maj *from Proposition 4.46. For $n = 5$ inputs, it has the algebraic normal form*

$$f(x_1, \ldots, x_5) = f(X) = \sigma_3(X) + \sigma_4(X) \tag{5.5.47}$$

where $\sigma_s(X)$ denotes the j^{th} elementary symmetric function (4.37). This shows that $f_\alpha = 1$ if and only if $|\alpha| = 3$ or $|\alpha| = 4$. For a fast algebraic attacks using

$f(X) - z$ *as the 1-function, one would split it as follows:*

$$0 = f(X) - z = \underbrace{\sigma_4(X)}_{=:G(X)} + \underbrace{\sigma_3(X) + z}_{=:H(X,z)}$$

with $d = 4$ *and* $d' = 3$*. The question is if better solutions exist.*

As $AI(\mathrm{maj}) = \lfloor n/2 \rfloor + 1 = 3$ *(or equivalently* $\mathrm{lad}(1) = 3$*), we set* $d = 3$*. Assume that we are intersted in finding out if* $d' = 1$ *is possible. Then,* $D' = \binom{5}{0} + \binom{5}{1} = 6$*. We choose the ordering (00000), (00001), (00010), (00100), (01000), (10000) for the* β *with* $|\beta| \leq d' = 1$ *and consider*

$$\gamma \in \mathcal{G} := \{(11110), (11101), (11011), (10111), (01111), (11111)\}.$$

To set up the matrix M*, we have to compute the values* $A_{\gamma,\beta}$ *for* $\gamma \in \mathcal{G}$ *and* $|\beta| \leq 1$*. Let us start with* $\gamma = (11110)$*. It holds that*

$$\begin{array}{lcllcl} A_{(11110),(00000)} & = & f_{(11110)} = 1 & , & A_{(11110),(00001)} & = & 0, \\ A_{(11110),(00010)} & = & f_{(11100)} + f_{(11110)} = 0 & , & A_{(11110),(00100)} & = & f_{(11010)} + f_{(11110)} = 0, \\ A_{(11110),(01000)} & = & f_{(10110)} + f_{(11110)} = 0 & , & A_{(11110),(10000)} & = & f_{(01110)} + f_{(11110)} = 0. \end{array}$$

This yields that the first row of M *is equal to* $(1\ 0\ 0\ 0\ 0\ 0)$*. In a similar way, one can show that the same holds for the rows corresponding to* $\gamma = (11101), (11011), (10111), (01111)$*. For the final row, corresponding to* $\gamma = (11111)$*, one has*

$$\begin{array}{lll} A_{(11111),(00000)} = 0, & A_{(11111),(00001)} = f_{(11110)} = 1, & A_{(11111),(00010)} = f_{(11101)} = 1, \\ A_{(11111),(00100)} = f_{(11011)} = 1, & A_{(11111),(01000)} = f_{(10111)} = 1, & A_{(11111),(10000)} = f_{(01111)} = 1. \end{array}$$

Thus, the matrix M *has the form*

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

A possible basis of the kernel space is

$$\{(0,1,1,0,0,0), (0,1,0,1,0,0), (0,1,0,0,1,0), (0,1,0,0,0,1)\}.$$

which corresponds to the following choices (or linear combination of them) for h*:* $x_1 + x_2,\ x_1 + x_3,\ x_1 + x_4,\ x_1 + x_5$*. The validity of the solutions can be easily checked via multiplication, e. g.*

$$0 = (\mathrm{maj}(X) - z) \cdot (x_1 + x_2) = \underbrace{(x_1 + x_2) \cdot (x_3x_4 + x_3x_5 + x_4x_5)}_{=G(X)} + \underbrace{(x_1 + x_2) \cdot z}_{=H(X,Z)}.$$

Because of $\deg(G) = d = 3$ *and* $\deg(H) = d' = 1$*, this solution is better for a fast algebraic attack than the naive approach displayed in (5.5.47).*

5.5.3 How to set up Equations for Symmetric Functions

In this section, we consider symmetric Boolean functions. We give a general analysis of the resulting system of equations and present a generic and a specific algorithm in order to determine the existence of G (and h) of low degree.

Setting up Equations

Recall the definition of symmetric functions and the underlying theory described in Section 4.2.3. Let $f(x)$ be symmetric, that is $f(X) = f(x_1, \ldots, x_n)$ is invariant under exchanging the variables x_i. Therefore, it is $f(\boldsymbol{X}) = f(\boldsymbol{X}')$ if $|\boldsymbol{X}| = |\boldsymbol{X}'|$ and we can identify $f(X)$ with its (abbreviated) truth table$T^\sigma(f) := (f(0), f(1), \ldots, f(n)) \in \mathbb{F}^{n+1}$, where $f(i) := f(\boldsymbol{X})$ for a $\boldsymbol{X}$ with $|\boldsymbol{X}| = i$.

By Theorem 4.39, we can likewise identify each symmetric function with its abbreviated coefficient vector $C^\sigma(f) = (f_0, \ldots, f_n)$. Again, we can express the relationship between $C^\sigma(f)$ and $T^\sigma(f)$ by the Hadamard matrix:

Theorem 5.54. *Let $f(x) = f(x_1, \ldots, f_n)$ be a symmetric function. Then*

$$C^\sigma(f) = H_{n+1} \cdot T^\sigma(f) \quad \textit{and} \quad T^\sigma(f) = H_{n+1} \cdot C^\sigma(f)\ . \tag{5.5.48}$$

Proof. Due to Lemma 5.47, H_{n+1} is self-inverse. Hence, we only show that $T^\sigma(f) = H_{n+1} \cdot C^\sigma(f)$.

For this purpose, we define a matrix S_{n+1} of size $(n+1) \times (n+1)$ such that $S(i,j) := \sigma_j(i)$ for $0 \leq i, j \leq n$, where $\sigma_j(i)$ is the j^{th} elementary symmetric polynomial evaluated on an input with weight i. This means that the rows are labeled by the different input weights i and the columns by the different elementary symmetric polynomials σ_j.

Next, we define the vector $T := S_{n+1} \cdot C^\sigma(f)$. The i-th entry of T is $\sum_{j=0}^{n} f_j \cdot \sigma_j(i) = f(i)$. This shows that $T = S_{n+1} \cdot C^\sigma(f) = T^\sigma(f)$. Further on, one can easily check that $S(i,j) = \sigma_j(i) = \binom{i}{j} \mod 2$. Together with Lemma 5.47, it follows that $S_{n+1} = H_{n+1}$, which concludes the proof. □

Example 5.55. *We consider again the majority function* maj *with $n = 5$ inputs. Using the abbreviated representation based on the input's weight, it holds that $f(0) = f(1) = f(2) = 0$ and $f(3) = f(4) = f(5) = 1$. The corresponding truth table is $T^\sigma(f) = (0, 0, 0, 1, 1, 1)$. Thus, we can compute the coefficients*

vector by

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} .$$

Thus, the majority function with three inputs has the algebraic normal form

$$\begin{aligned} f(X) &= 0 \cdot \sigma_0(X) + 0 \cdot \sigma_1(X) + 1 \cdot \sigma_2(X) + 1 \cdot \sigma_3(X) + 1 \cdot \sigma_4(X) + 0 \cdot \sigma_5(X) \\ &= \sigma_3(X) + \sigma_4(X) \end{aligned}$$

as claimed in Example 5.53.

With the help of Proposition 5.50, one can derive a much simpler description of the coefficients G_γ in the case of symmetric functions. Remember that $\beta \subseteq \gamma$ is an abbreviation for $\mathrm{supp}(\beta) \subseteq \mathrm{supp}(\gamma)$.

Theorem 5.56. *Let $f(X) = \sum_{i=0}^{n} f_i \sigma_i(X)$ a symmetric function and $h(X) = \sum_\beta h_\beta X^\beta$. Set $G(X) := \sum_\gamma G_\gamma X^\gamma := f(x) \cdot h(x)$. Let $A^\sigma_{|\gamma|,|\beta|}$ denote an element in $\mathbb{F}_2$ and $\bar{B}^\sigma_{i,j}$ a vector of size $n+1$ over $\mathbb{F}_2$. Then, it is for each γ*

$$G_\gamma = \sum_{\beta \subseteq \gamma} A^\sigma_{|\gamma|,|\beta|} \cdot h_\beta = \sum_{b=0}^{|\gamma|} A^\sigma_{|\gamma|,b} \sum_{\beta \subseteq \gamma : |\beta| = b} h_\beta \tag{5.5.49}$$

$$A^\sigma_{i,j} := \bar{B}^\sigma_{i,j} \cdot T^\sigma(f) = \bar{B}^\sigma_{i,i-j} \cdot C^\sigma(f) \tag{5.5.50}$$

$$\left[\bar{B}^\sigma_{i,j}\right]_k := \binom{i-j}{i-k} \mod 2 \, . \tag{5.5.51}$$

With this theorem, the coefficients $A^\sigma_{|\gamma|,|\beta|}$ can be computed very cheaply, either with the simplified coefficients vector of f, or with the value vector of f. Notice that the vector $\bar{B}^\sigma_{i,j}$ is again independent of f. Before we prove the theorem, we give an example how to set up an equation.

Example 5.57. *Let us set up an equation for $\gamma = (101)_2$. It is $|\gamma| = 2$, and (5.5.49) results in*

$$\begin{aligned} G_{(101)} = {} & A^\sigma_{2,0} \cdot (h_{(000)}) \\ & + A^\sigma_{2,1} \cdot (h_{(100)} + h_{(001)}) \\ & + A^\sigma_{2,2} \cdot (h_{(101)}) \, . \end{aligned}$$

If we use (5.5.50) and (5.5.51) in order to determine the coefficients $A^{\sigma}_{i,j}$*, we find*

$$\begin{aligned} A^{\sigma}_{2,0} &= f_2 = f(0) + f(2) \\ A^{\sigma}_{2,1} &= f_1 + f_2 = f(1) + f(2) \\ A^{\sigma}_{2,2} &= f_0 + f_2 = f(2)\ . \end{aligned}$$

The rest of this subsection is exclusively dedicated to the task of proving Theorem 5.56. First, we need some additional statements based on the definition of the support.

Proposition 5.58. *Let* $\alpha, \beta, \gamma \in \mathbb{F}^n$ *such that* $\alpha \vee \beta = \gamma$*. Then* $\mathrm{supp}(\alpha) \cup \mathrm{supp}(\beta) = \mathrm{supp}(\gamma)$*. In particular, it is* $\alpha \subseteq \gamma$ *and* $\beta \subseteq \gamma$*.*

Proof. By assumption, it is $\gamma_i = \alpha_i \vee \beta_i$ for $i = 1, \ldots, n$. Let $i \in \mathrm{supp}(\gamma)$. Then $1 = \gamma_i = \alpha_i \vee \beta_i$ shows that either $\alpha_i = 1$ or $\beta_i = 1$. This shows that $\mathrm{supp}(\gamma) \subseteq \mathrm{supp}(\alpha) \cup \mathrm{supp}(\beta)$. Next we show that $\mathrm{supp}(\alpha) \subseteq \mathrm{supp}(\gamma)$. Assume that this is not true. Then there exists $j \in \mathrm{supp}(\alpha) \setminus \mathrm{supp}(\gamma)$. This means that $\alpha_j = 1$ and $\gamma_j = 0$. But this would imply the contradiction $1 = \alpha_j \vee \beta_j = \gamma_j = 0$. Hence, we have $\mathrm{supp}(\alpha) \subseteq \mathrm{supp}(\gamma)$. The rest follows from the symmetry of $\alpha \vee \beta$. □

Proposition 5.59. *Let* $\gamma, \beta \in \mathbb{F}^n$ *with* $\beta \subseteq \gamma$*. For each* $i \in \{0, \ldots, |\beta|\}$ *exist* $\binom{|\beta|}{i}$ *different* $\alpha \in \mathbb{F}^n$ *such that*

1. $\alpha \vee \beta = \gamma$
2. $|\alpha| = |\gamma| - |\beta| + i$

Proof. First, we derive some conditions on α_i to achieve $\alpha_i \vee \beta_i = \gamma_i$. For this purpose, we distinguish between three cases:

$i \in \mathrm{supp}(\beta)$: This means that $\beta_i = \gamma_i = 1$. Hence, α_i can be either 0 or 1. There are no restrictions on α_i.

$i \notin \mathrm{supp}(\beta), i \in \mathrm{supp}(\gamma)$: This means that $\beta_i = 0$ but $\gamma_i = 1$. Hence, $\alpha_i := 1$.

$i \notin \mathrm{supp}(\beta), i \notin \mathrm{supp}(\gamma)$: We have $\beta_i = \gamma_i = 0$. This implies $\alpha_i := 0$.

We see that it is always possible to construct α such that $\alpha \vee \beta = \gamma$. The first and second case show that each α with $\alpha \vee \beta = \gamma$ can be characterized by

$$\mathrm{supp}(\alpha) = (\mathrm{supp}(\gamma) \setminus \mathrm{supp}(\beta)) \dot{\cup}\ S \tag{5.5.52}$$

with $S \subseteq \mathrm{supp}(\beta)$ (including $S = \emptyset$). Furthermore, it is $|\alpha| = |\gamma| - |\beta| + |S|$. Now let $i \in \{0, \ldots, |\beta|\}$. As we have seen, each α with $\alpha \vee \beta = \gamma$ and $|\alpha| = |\gamma| - |\beta| + i$ corresponds to exactly one set $S \subseteq \mathrm{supp}(\beta)$ with $|S| = i$. There exist $\binom{|\mathrm{supp}(\beta)|}{i} = \binom{|\beta|}{i}$ such sets S, which implies the claim. □

Now we are ready to simplify Theorem 5.51 for symmetric functions. Let $f(X) = \sum_\alpha f_\alpha X^\alpha = \sum_{i=0}^n f_i \sigma_i(X)$ be a symmetric function and $h(X) = \sum_\beta h_\beta X^\beta$. Set $G(X) := \sum_\gamma G_\gamma X^\gamma := f(X) \cdot h(X)$. Due to Proposition 5.50, each coefficient of G can be expressed in the following way.

$$\begin{aligned}
G_\gamma &= \sum_{(\alpha,\beta):\alpha\vee\beta=\gamma} f_\alpha \cdot h_\beta = \sum_{b=0}^{|\gamma|} \sum_{\beta\subseteq\gamma:|\beta|=b} h_\beta \cdot \left[\sum_{\alpha\subseteq\gamma:\alpha\vee\beta=\gamma} f_\alpha \right] \\
&= \sum_{b=0}^{|\gamma|} \sum_{\beta\subseteq\gamma:|\beta|=b} h_\beta \cdot \left[\sum_{i=0}^{b} \sum_{\substack{\alpha\subseteq\gamma:\\ \alpha\vee\beta=\gamma,\\ |\alpha|=|\gamma|-b+i}} f_\alpha \right] \\
&\overset{f \text{ symm.}}{=} \sum_{b=0}^{|\gamma|} \sum_{\beta\subseteq\gamma:|\beta|=b} h_\beta \cdot \left[\sum_{i=0}^{b} \sum_{\substack{\alpha\subseteq\gamma:\\ \alpha\vee\beta=\gamma,\\ |\alpha|=|\gamma|-b+i}} f_{|\gamma|-b+i} \right] \\
&\overset{\text{Prop. 5.59}}{=} \sum_{b=0}^{|\gamma|} \sum_{\beta\subseteq\gamma:|\beta|=b} h_\beta \cdot \left[\sum_{i=0}^{b} \binom{b}{i} f_{|\gamma|-b+i} \right] \\
&= \sum_{b=0}^{|\gamma|} \left[\sum_{\beta\subseteq\gamma:|\beta|=b} h_\beta \right] \cdot \underbrace{\left[\sum_{i=0}^{b} \binom{b}{i} f_{|\gamma|-b+i} \right]}_{=:A^\sigma_{|\gamma|,b}} = \sum_{\beta\subseteq\gamma} h_\beta A^\sigma_{|\gamma|,|\beta|} \ .
\end{aligned}$$

The last equation is true as the part in brackets depends only on $b = |\beta|$ and not on the value of β itself. Next, we intend to simplify the term $\sum_{i=0}^b \binom{b}{i} f_{|\gamma|-b+i}$ in the above equation. Define $j := |\gamma|-b+i$, hence $i = j+b-|\gamma|$ and

$$\begin{aligned}
A^\sigma_{|\gamma|,|\beta|} &= \sum_{i=0}^{b} \binom{b}{i} f_{i+|\gamma|-b} = \sum_{j+b-|\gamma|=0}^{j+b-|\gamma|=b} \binom{b}{j+b-|\gamma|} f_j \\
&= \sum_{j=|\gamma|-b}^{|\gamma|} \binom{b}{|\gamma|-j} f_j = \sum_{j=0}^{n} \binom{b}{|\gamma|-j} f_j = \bar{B}^\sigma_{|\gamma|,|\gamma|-|\beta|} \cdot C^\sigma(f) \ .
\end{aligned}$$

The last equality holds because of $b \le |\gamma|$ and our extended definition of binomial coefficients. Finally, we intend to find a similar expression for $T^\sigma(f)$ instead of $C^\sigma(f)$. Remember that $C^\sigma(f) = H_{n+1} \cdot T^\sigma(f)$ with the Hadamard-matrix H. The next proposition is on the product $\bar{B}^\sigma_{|\gamma|,|\gamma|-|\beta|} \cdot H_{n+1}$.

Proposition 5.60. *Let $\gamma \in \{0,1\}^n$, $b \in \{0,\ldots,|\gamma|\}$ and $i \in \{0,\ldots,n\}$. It holds that*

$$\left[\sum_{k=0}^{|\gamma|} \binom{b}{|\gamma|-k} \cdot \binom{k}{i} = \binom{|\gamma|-b}{i-b}\right] \mod 2 \ . \tag{5.5.53}$$

Proof. We prove the statement by induction on $|\gamma|$ and $b \leq |\gamma|$. Let $\gamma := 0$. Hence, $b = 0$ and (5.5.53) can be rewritten to

$$\sum_{k=0}^{|\gamma|} \binom{b}{|\gamma|-k} \cdot \binom{k}{i} = \underbrace{\binom{0}{0}}_{=1} \cdot \binom{0}{i} = \binom{|\gamma|}{i} = \binom{|\gamma|-b}{i-b} \ .$$

Next, let $|\gamma|$ and i be arbitrary and assume that the statement is true for all $|\gamma'| < |\gamma|$ and $b' \leq |\gamma'|$. We show now that (5.5.53) is true for all $0 \leq b \leq |\gamma|$ by induction on b. First, we consider the case $b = 0$. It holds that

$$\sum_{k=0}^{|\gamma|} \binom{b}{|\gamma|-k} \cdot \binom{k}{i} \ = \ \sum_{k=0}^{|\gamma|-1} \underbrace{\binom{0}{|\gamma|-k}}_{0} \cdot \binom{k}{i} + \underbrace{\binom{0}{0}}_{=1} \cdot \binom{|\gamma|}{i} = \binom{|\gamma|-b}{i-b} \ .$$

Now, assume that the claim is true for $|\gamma|$ and $0 \leq b < |\gamma|$. For the sake of readability, we will omit the string "mod 2" from now on and will use the notation $\equiv_2$ instead. A famous theorem on binomial coefficients says that $\binom{r+1}{s} = \binom{r}{s} + \binom{r}{s-1}$ or $\binom{r+1}{s} - \binom{r}{s} = \binom{r}{s-1}$. Because of $-x \equiv_2 x$, in our case we get $\binom{r+1}{s} \equiv_2 \binom{r}{s} - \binom{r}{s-1}$. The following equations are all modulo 2. However, this is only important for the first equation. The other equations are true also in the ring of integers.

$$\begin{aligned}
\sum_{k=0}^{|\gamma|} \binom{b+1}{|\gamma|-k} \cdot \binom{k}{i} \ &\equiv_2 \ \sum_{k=0}^{|\gamma|} \left[\binom{b}{|\gamma|-k} - \binom{b}{|\gamma|-k-1}\right] \cdot \binom{k}{i} \\
&= \ \sum_{k=0}^{|\gamma|} \binom{b}{|\gamma|-k} \cdot \binom{k}{i} - \sum_{k=0}^{|\gamma|} \binom{b}{|\gamma|-k-1} \binom{k}{i} \\
&= \ \binom{|\gamma|-b}{i-b} - \sum_{k=0}^{|\gamma|-1} \binom{b}{(|\gamma|-1)-k} \binom{k}{i} \\
&= \ \binom{|\gamma|-b}{i-b} - \binom{|\gamma|-1-b}{i-b} \\
&= \ \binom{|\gamma|-1-b}{i-b-1} = \binom{|\gamma|-(b+1)}{i-(b+1)} \ .
\end{aligned}$$

This concludes the proof. □

Together with $\binom{|\gamma|-b}{i-b} = \binom{|\gamma|-b}{(|\gamma|-b)-(i-b)} = \binom{|\gamma|-b}{|\gamma|-i}$, this shows that

$$A^{\sigma}_{|\gamma|,|\beta|} = \bar{B}^{\sigma}_{|\gamma|,|\gamma|-|\beta|} \cdot C^{\sigma}(f) = \bar{B}^{\sigma}_{|\gamma|,|\gamma|-|\beta|} \cdot H_{n+1} \cdot T^{\sigma}(f) = \bar{B}^{\sigma}_{|\gamma|,|\beta|} \cdot T^{\sigma}(f).$$

This finishes the proof of Theorem 5.56.

Determining the Existence of Solutions

If we consider again the situation described in Section 5.5.2, we may use Algorithm 5.5 analogously for symmetric functions, where $A\gamma, \beta$ in step 7 now becomes $A^{\sigma}_{|\gamma|,|\beta|}$ as defined for symmetric functions. In addition, as this coefficient only depends on the weight j of β, it may be computed outside the loop (preliminary to step 6 of Algorithm 5.5). The discussion of this slightly modified algorithm is similar to Section 5.5.2. However, computation of $A^{\sigma}_{|\gamma|,|\beta|}$ requires only $n+1$ evaluations of the function f and has to be computed only once for all elements in $\mathcal{B}$, hence it is much cheaper compared to the corresponding step in Algorithm 5.5 and can be neglected here. Consequently, the time effort to set up equations is only in $\mathcal{O}(D'^2)$.

Example 5.61. *Let* maj *again be the majority function with* $n = 5$ *inputs. Like in Example 5.53, we choose the ordering (00000), (00001), (00010), (00100), (01000), (10000) for the* β *with* $|\beta| \leq d' = 1$ *and consider*

$$\gamma \in \mathcal{G} := \{(11110), (11101), (11011), (10111), (01111), (11111)\}.$$

(5.5.49) yields the following equations[12]

$$\begin{aligned}
0 = G_{(11110)} &= A^{\sigma}_{4,0} \cdot h_{(00000)} + A^{\sigma}_{4,1} \cdot (h_{(00010)} + h_{(00100)} + h_{(01000)} + h_{(10000)}) \\
0 = G_{(11101)} &= A^{\sigma}_{4,0} \cdot h_{(00000)} + A^{\sigma}_{4,1} \cdot (h_{(00001)} + h_{(00100)} + h_{(01000)} + h_{(10000)}) \\
0 = G_{(11011)} &= A^{\sigma}_{4,0} \cdot h_{(00000)} + A^{\sigma}_{4,1} \cdot (h_{(00001)} + h_{(00010)} + h_{(01000)} + h_{(10000)}) \\
0 = G_{(10111)} &= A^{\sigma}_{4,0} \cdot h_{(00000)} + A^{\sigma}_{4,1} \cdot (h_{(00001)} + h_{(00010)} + h_{(00100)} + h_{(10000)}) \\
0 = G_{(01111)} &= A^{\sigma}_{4,0} \cdot h_{(00000)} + A^{\sigma}_{4,1} \cdot (h_{(00001)} + h_{(00010)} + h_{(00100)} + h_{(01000)}) \\
0 = G_{(11111)} &= A^{\sigma}_{5,0} \cdot h_{(00000)} + A^{\sigma}_{5,1} \cdot (h_{(00001)} + h_{(00010)} + h_{(00100)} + h_{(01000)} + h_{(10000)})
\end{aligned}$$

Because of $T^{\sigma}(f) = (0, 0, 0, 1, 1, 1)$ *and Equations (5.5.49) and (5.5.51), it holds*

[12] Recall that we assume $h_{\beta} = 0$ for $|\beta| > 1$.

that $A^{\sigma}_{i,j} = \binom{i-j}{i-3} + \binom{i-j}{i-4} + \binom{i-j}{i-5} \mod 2$. *This yields*

$$\begin{aligned} A^{\sigma}_{4,0} &= \binom{4}{1} + \binom{4}{0} + \binom{4}{(-1)} \mod 2 = 1, \\ A^{\sigma}_{4,1} &= \binom{3}{1} + \binom{3}{0} + \binom{3}{(-1)} \mod 2 = 0, \\ A^{\sigma}_{5,0} &= \binom{5}{2} + \binom{5}{1} + \binom{5}{0} \mod 2 = 0, \quad \text{and} \\ A^{\sigma}_{5,1} &= \binom{4}{2} + \binom{4}{1} + \binom{4}{0} \mod 2 = 1. \end{aligned}$$

This results into exactly the same matrix M *as in Example 5.53, yielding of course the same solutions. However, the generation of* M *is much simple now.*

Next, we will derive a method with a very low effort to determine the existence of $h(X)$ and $G(X)$ of low degree for a symmetric function $f(X)$, but at the expense of the method's only using sufficient conditions (i.e., some solutions may be lost). More precisely, we constrict ourselves to homogeneous functions $h(X)$ of degree d' (i.e., $h(X)$ contains monomials of degree d' only), and (5.5.49) becomes

$$G_\gamma = A^{\sigma}_{|\gamma|,d'} \sum_{\beta \subseteq \gamma : |\beta| = d'} h_\beta \ .$$

Remember that $G_\gamma = 0$ for $|\gamma| > d$, hence the homogeneous function $h(X)$ is determined by the system of equations $0 = A^{\sigma}_{|\gamma|,d'} \sum_{\beta \subseteq \gamma : |\beta| = d'} h_\beta$ for all γ with $|\gamma| = d+1, d+2, \ldots, n$. In this system of equations, the coefficient $A^{\sigma}_{|\gamma|,d'} \in \mathbb{F}_2$ is reused for a number of $\binom{n}{|\gamma|}$ equations. If $A^{\sigma}_{|\gamma|,d'} = 0$, then all of these equations are linearly dependent (i.e., of type $0 = 0$). On the other hand, if $A^{\sigma}_{|\gamma|,d'} = 1$, then there is a number of $\binom{n}{|\gamma|}$ additional equations that are possibly linearly independent. Consequently, if the sum of all possibly linearly independent equations for $|\gamma| = d+1, d+2, \ldots, n$ is smaller than the number of variables $\binom{n}{d'}$, then non-trivial homogeneous functions $h(X)$ exist. This sufficient criterion is formalized by

$$\sum_{i=d+1}^{n} A^{\sigma}_{i,d'} \cdot \binom{n}{i} < \binom{n}{d'} \ . \tag{5.5.54}$$

Notice that the remaining equations may still be linearly dependent; however, as the system is very overdefined, we expect the loss of solutions to be marginal. Consequently, given some d', the goal is to find the minimum

value of d such that (5.5.54) holds. This can be done incrementally, starting from $d = n$, and for all $d' = 0, 1, \ldots, \lceil n/2 \rceil$. We formalized the idea in Algorithm alg:alg3 which has an effort in $\mathcal{O}(n^3)$. In other words, the algorithm uses the counting argument explained above to derive values d and d' such that the existence of functions G and h of degree d and d', respectively, with $f \cdot h = G$ is guaranteed. Observe that Algorithm 5.6 neither computes such functions nor gives any statements on the existence of functions with degrees not covered by the counting argument. The algorithm turned out to be very powerful (but not necessarily optimal) in practice, see Section 5.5.5.

Algorithm 5.6.
Fast algebraic attacks on a symmetric Boolean function (specific version)

Input: *A symmetric Boolean function f of dimension n.*
Output: *Determine degrees of specific homogeneous functions G and h such that $f \cdot h = G$ is possible.*

1: ***for*** d' *from* 0 *to* $\lceil n/2 \rceil$ ***do***
2: *Let* $d = n$, *number of equations* $= 0$, *number of variables* $= \binom{n}{d'}$
3: ***while*** *Number of equations is smaller than number of variables and* $d + 1 > 0$ ***do***
4: *Compute* $b \leftarrow A^{\sigma}_{d,d'}$.
5: *Add* $b \cdot \binom{n}{d}$ *to the number of equations.*
6: $d \leftarrow d - 1$.
7: ***end while***
8: *Output* (d', d).
9: ***end for***

Given a symmetric Boolean function f defined for different dimensions n, we would also like to give some general statements concerning the degrees of functions G and h for any n. In the next section, we will apply the technique based on Algorithm 5.6.

5.5.4 Fast Algebraic Attacks on the Majority Function

In Proposition 4.46, we have seen that using the majority function with n inputs as an output function guarantees that any 1-function has a degree of $\geq \lfloor n/2 \rfloor + 1$, which is the best one can achieve. In the following, we will examine how good it is in the context of fast algebraic attacks.

Theorem 5.62. *Consider the majority function* maj *with any* $n \geq 2$ *input variables. Then there exist Boolean functions* G *and* h *such that* $f \cdot h = G$*, where* $\deg G = \lfloor n/2 \rfloor + 1$ *and* $d' = \deg h = d - 2^j$*, and where* $j \geq 0$ *is maximum so that* $\deg h > 0$*. If* n *is even, the vector space of solutions* h *has a dimension* $\geq \binom{n}{d'} - \binom{n}{d'-2}$*, and in the case of* n *odd a dimension* $\geq \binom{n}{d'} - \binom{n}{d'-1}$*.*

Begin of proof of Theorem 5.62

First, we show that the coefficients $A^{\sigma}_{|\gamma|,d'}$ from Theorem 5.56 have a simple form in the case of the majority function.

Proposition 5.63. *Let* maj *be the majority function in* n *inputs and* $d = \lfloor n/2 \rfloor + 1$*. Then*

$$A^{\sigma}_{i,j} = \begin{cases} 0 & , i < d \\ \binom{i-j-1}{d-j-1} \mod 2 & , \text{else} \end{cases} .$$

Proof. For the sake of readability, we omit the expression "mod 2" in the equations. Recall that $\mathrm{maj}(k) = 0$ for $k < d$. Hence, due to Theorem 5.56 it is

$$A^{\sigma}_{i,j} \overset{\text{Th. 5.56}}{=} \sum_{k=0}^{n} \binom{i-j}{k-j} \cdot \mathrm{maj}(k) = \sum_{k=d}^{n} \binom{i-j}{k-j} = \sum_{k=d}^{i} \binom{i-j}{k-j} .$$

The last equation is true because of $\binom{a}{b} = 0$ for $a < b$. This shows that $A^{\sigma}_{i,j} = 0$ for $i < d$. Now, let $i \geq d \geq 1$. Then, $A^{\sigma}_{i,j}$ can be simplified to

$$\begin{aligned} A^{\sigma}_{i,j} &= \sum_{k=d}^{i} \binom{i-j}{k-j} = \sum_{k=d}^{i} \binom{i-j-1}{k-j} + \sum_{k=d}^{i} \binom{i-j-1}{k-j-1} \\ &= \sum_{k=d}^{i-1} \binom{(i-1)-j}{k-j} + \sum_{k=d}^{i} \binom{(i-1)-j}{(k-1)-j} = A^{\sigma}_{i-1,j} + \sum_{k=d-1}^{i-1} \binom{(i-1)-j}{k-j} \\ &= 2 \cdot A^{\sigma}_{i-1,j} + \binom{i-j-1}{d-j-1} = \binom{i-j-1}{d-j-1} . \end{aligned}$$

□

Example 5.64. *With Proposition 5.63 at hand, one can easily verify the values of* $A^{\sigma}_{|\gamma|,b}$ *in Example 5.61:*

$$\begin{aligned} A^{\sigma}_{4,0} &= \tbinom{4-0-1}{3-0-1} \mod 2 = 1, & A^{\sigma}_{4,1} &= \tbinom{4-1-1}{3-1-1} \mod 2 = 0 \\ A^{\sigma}_{5,0} &= \tbinom{5-0-1}{3-0-1} \mod 2 = 0, & A^{\sigma}_{5,1} &= \tbinom{5-1-1}{3-1-1} \mod 2 = 1. \end{aligned}$$

Consider now $\mathrm{maj}(X) \cdot h(X) = G(X)$, where maj is the majority function, $\deg h = d'$ and $\deg G = d$. The proof of the claim consists of two main steps. First, we set up a system of equations in the coefficients of h only, then we

show that nontrivial solutions exist for appropriate values of d' and d. For this purpose, we apply the simplified form of $A^{\sigma}_{|\gamma|,|\beta|}$.

Additionally, we assume that h is homogeneous. This means that $h_\beta = 0$ for all β with $|\beta| \neq d'$. Together with Theorem 5.56 and $\deg G = d$, we get the following sufficient conditions on h:

$$0 = A^{\sigma}_{|\gamma|,d'} \cdot \sum_{\beta \subseteq \gamma : |\beta| = d'} g_\beta \qquad \forall \gamma : |\gamma| > d \ . \tag{5.5.55}$$

Observe that each γ with $|\gamma| > d$ gives one linear equation in the coefficients h_β of h. Our strategy is as follows. We set d' to a value which guarantees that the number of coefficients $A^{\sigma}_{|\gamma|,d'}$ which might be equal to 1 is less than the number of coefficients h_β. In other words,

$$\#\{\gamma : |\gamma| > d, A^{\sigma}_{|\gamma|,d'} = 1\} < \#\{h_\beta : |\beta| = d'\} = \binom{n}{d'} \ . \tag{5.5.56}$$

If Equation (5.5.56) is true, then (5.5.55) yields an underdefined system of linear equations which must have consequently a solution h. For this purpose, we set $d' := d - 2^j$ where j is chosen maximum such that $d' \geq 1$.

It is $d - d' - 1 = 2^j - 1 = \sum_{i=0}^{j-1} 2^i$. Recall that for $a = \sum a_i \cdot 2^i$, it holds that

$$\binom{a}{d-d'-1} \mod 2 = \prod_{i=0}^{j-1} \binom{a_i}{1} \cdot \prod_{i=j}^{n} \binom{a_i}{0} \mod 2 = \prod_{i=0}^{j-1} \binom{a_i}{1} \mod 2.$$

This shows that $\binom{a}{d-d'-1} \mod 2 = 1$ if and only if $a_i = 1$ for $i = 0, \ldots, j-1$, or equivalently,

$$a = \left(\sum_{i=0}^{j-1} 2^i\right) + \left(\sum_{i=j}^{n} a_i \cdot 2^i\right) = \left(2^j - 1\right) + \left(\sum_{i=j}^{n} a_i \cdot 2^i\right) = \left(2^j - 1\right) + 2^j \cdot \left(\sum_{i=j}^{n} (a_i \cdot 2^{i-j}\right).$$

Hence, $A^{\sigma}_{d+i,d'} = \binom{d-d'-1+i}{d-d'-1} \mod 2 = \binom{(2^j-1)+i}{2^j-1} \mod 2 = 1$ if and only if i is a multiple of $2^j = d - d'$. In other words, if $d = \lfloor n/2 \rfloor + 1$ and if $d' = \lfloor n/2 \rfloor + 1 - 2^j$, only equations (5.5.55) with $|\gamma| = d + k \cdot (d - d')$ and $k \geq 1$ impose conditions on the coefficients h_β, whereas the others are necessarily equal to zero. For $k = 1$, we have $A^{\sigma}_{2d-d',d'} = 1$.

We will prove now that $d + k \cdot (d - d') > n$ for $k = 2$. The consequence is that only the coefficients $A^{\sigma}_{|\gamma|,d'}$ for $|\gamma| = d + (d - d') = 2d - d'$ are equal to 1. By the definition of j, it holds that

$$\lfloor n/2 \rfloor + 1 - 2^{j+1} \leq 0 \Leftrightarrow 2^{j+1} \geq \lfloor n/2 \rfloor + 1 \Leftrightarrow 2(d - d') \geq d \ .$$

As $d = \lfloor n/2 \rfloor + 1$, this shows for $k \geq 2$ that $d + k \cdot (d - d') \geq d + 2 \cdot (d - d') \geq d + d = 2 \cdot (\lfloor n/2 \rfloor + 1) > n$.

Altogether, the number of non-trivial equations in (5.5.55) is $\binom{n}{2d-d'}$, which is equal to $\binom{n}{d'-2}$ for n even and $\binom{n}{d'-1}$ for n odd. In both cases, this value is less than $\binom{n}{d'}$, the number of coefficients h_β. Consequently, the system of equations (5.5.55) is underdefined and non-trivial solutions for h exist. Further on, we can see that the dimension of the solution space is at least $\binom{n}{d'} - \binom{n}{d'-2}$ for n even and at least $\binom{n}{d'} - \binom{n}{d'-1}$ for n odd.

End of proof of Theorem 5.62

Example 5.65. *Recall Example 5.53 where we have chosen the parameters* $d = \lfloor n/2 \rfloor + 1 = 3$ *and* $d' = d - 2^1 = 1$. *As stated by Theorem 5.62, solutions exist and the dimension of the solution space is* $4 \geq \binom{n}{d'} - \binom{n}{d'-1} = \binom{5}{1} - \binom{5}{0} = 4$.

Table 5.5.4 display more examples. Theorem 5.62 was verified for $5 \leq n \leq 16$, see Section 5.5.5. For n odd, Theorem 5.62 predicts the existence of a function $G(X)$ with degree $d = AI(\mathrm{maj})$, but for n even, it is $d = AI(\mathrm{maj}) + 1$, which may increase the complexity of fast algebraic attacks significantly. However, experiments indicate that Theorem 5.62 is optimal (for both d and d'), see Section 5.5.5. Notice that some trivial solutions exist due to the low degree of maj (e.g. for $n = 7$, $\deg \mathrm{maj} = 4$ and one could choose $\deg g = 1$). For values $n = 2, 3, 4, 6$ only, Theorem 5.62 is not meaningful. A very interesting subcase is $n = 2^{j+1}$ and $n = 2^j + 1$ for $j \geq 2$, for which Boolean functions h with $\deg h = 1$ exist.

n	5	6	7	8	9	10	11	12	13	14	15	16
$\deg h$	1	2	2	1	1	2	2	3	3	4	4	1
$\deg G$	3	4	4	5	5	6	6	7	7	8	8	9

Table 5.5.7: Degrees of the functions G and h (from $f \cdot h = G$) for dimension n, according to Theorem 5.62.

Theorem 5.62 can be applied to Boolean functions other than the majority function. Considering Theorem 4.43, binary affine transformations in the input variables and complementation are invariant transformations with respect to (fast) algebraic attacks. In [BraP05], groups of symmetric Boolean functions with maximum annihilator immunity are presented; a single group consists of a representative function, and other functions in the group are derived by such invariant transformations of the representative function. The representative function of group 1 is the majority function, consequently Theorem 5.62 is valid for all functions in their group 1.

One of the functions in group 1 was proposed in a recent paper, discussing design principles of stream ciphers [BraL05, BraLMPV05].

5.5.5 Experimental Results

Given a Boolean function f with n input variables, one may apply Algorithm 5.5 or Algorithm 5.6 in order to find degrees of g and h. In this section, we present the results for some examples of symmetric and non-symmetric Boolean functions f with maximum annihilator immunity.

Symmetric Functions with Maximum Annihilator Immunity

Let us first summarize the experimental results for the majority function with $n = 5, 6, \ldots, 16$. Indeed, Algorithm 5.6 finds the solutions d and d' according to Theorem 5.62. Furthermore, application of Algorithm 5.5 (modified for symmetric functions) brings out that Theorem 5.62 is optimal for these functions. Algorithm 5.5 may be also used to find explicit functions $h(X)$ which have been used to perform a formal expansion of $f(X) \cdot h(X)$ and verify the degrees. For any value of n, we found the following special homogeneous function $h(X)$, with degree d' according to Theorem 5.62,

$$h(X) = \prod_{i=1}^{d'} (x_{2i-1} + x_{2i}) \ . \tag{5.5.57}$$

This fits to Example 5.53 where we found the solution x_1+x_2 for the majority function in $n = 5$ inputs and parameters $d = 3$ and $d' = 1$.

In [BraP05], a large pool of symmetric Boolean functions defined for n even with maximum annihilator immunity is presented [13]. One of these functions is the majority function, whereas the other functions are non-linear transformations of the majority function. For example, one function consists of the value vector of the majority function, with the last entry set to zero. Let us summarize the experimental results for all of these functions with $n = 6, 8, 10, 12, 14, 16$. Application of Algorithm 5.6 brings out that g and h with degrees according to Theorem 5.62 exist for all functions f (remember that Theorem 5.62 was proven for the majority function only). For some functions f, Algorithm 5.6 finds better solutions than predicted by Theorem 5.62 (e.g. for $T = [0, 0, 0, 1, 1, 0, 1]$), which means that Theorem 5.62 is not optimal for all classes of functions. All solutions found by Algorithm 5.6 can be constructed according to (5.5.57) (a formal expansion

[13] Notice that for n odd, it is verified in [DalMS05] for up to $n = 11$ that the majority function is the only symmetric Boolean functions with maximum annihilator immunity.

of $f(X) \cdot h(X)$ is performed to verify the degrees). Furthermore, Algorithm 5.5 finds a few solutions which are better than predicted by Algorithm 5.6 (e.g. for $T = [0, 0, 0, 1, 1, 1, 0]$), which means that Algorithm 5.6 is not optimal for all functions.

Non-Symmetric Functions with Maximum Annihilator Immunity

In [DalGM05], a class of non-symmetric Boolean function with maximum annihilator immunity is presented. We summarize our experimental results for their examples with $n = 5, 6, 7, 8, 9, 10$. We applied Algorithm 5.5, the results are listed in Table 5.5.8 (where $\dim$ denotes the dimension of the solution space for $h(X)$); the degrees can be summarized by $d = AI(f) = \lceil n/2 \rceil$ and $d' = 1$. Explicit functions $h(X)$ with corresponding degree are also obtained by Algorithm 5.5. Again, we performed an explicit multiplication of h with the functions f in order to verify the results.

Table 5.5.8: Degrees of the functions g and h for DGM-functions f with n input variables.

n	$\deg f$	$\deg G$	$\deg h$	$h(X)$	$\dim$
5	4	3	1	$1 + x_4$	4
6	4	3	1	$1 + x_6$	4
7	5	4	1	$1 + x_4 + x_5$	1
8	5	4	1	$1 + x_5 + x_6$	1
9	8	5	1	$x_4 + x_5 + x_6 + x_7$	1
10	8	6	1	$x_5 + x_6 + x_7 + x_8$	1

6 Conclusion

In modern cryptography, the development and analysis of stream ciphers is one of the major topics. For example, the ECRYPT Stream Cipher Project[1] is fully dedicated to the search for new stream ciphers that might become suitable for widespread adoption. Stream ciphers are often based on keystream generators that are used to transform a small initial value, the key, into a long sequence of seemingly random outputs, the keystream. Many keystream generator designs are in turn based on linear feedback shift registers (LFSRs), which are efficiently implementable in hardware and produce output streams with excellent statistical properties.

To understand the security of cryptographic algorithms is the task of cryptanalysis, which is the activity of searching for security weaknesses. The underlying goal of cryptanalysis is not destructive, but constructive: Only by improving the understanding of potential problems, it is possible to propose new design criteria for cryptographic systems.

The thesis' focus lies on algebraic attacks, which are the newest cryptanalytic method for LFSR-based keystream generators. We treated both the attack, together with the according theory and several extensions, and resulting construction principles.

Chapter 1 gave a short introduction both into this topic and the thesis.

In Chapter 2, we provided the general framework and the mathematical background of the thesis and introduced important notions and concepts. This included sections on LFSRs, LFSR-based keystream generators, and a short survey of existing attacks.

Chapter 3 explained algebraic attacks on LFSR-based keystream generators. We described the two major steps, generating a system of equations and computing the solution, in detail and gave a rough effort analysis.

Chapter 4 was dedicated completely to equations for algebraic attacks. This included a complete framework on the existence of low-degree equations, algorithms to find them, and design principles to avoid them. At the end, we described our algebraic attacks based on generating low-degree equations by exploiting linear keyschedules or using fault attacks.

In Chapter 5, we explored fast algebraic attacks. After explaining the basic idea, we described our improvements on the precomputation step (reducing the time and data effort). We concluded with our results on the immunity of simple combiners against fast algebraic attacks.

Although algebraic attacks pose no practical threats at the moment, the enormous development on this subject indicates that the potential of these attacks is yet not fully understood. We expect that several improvements simply wait to be discovered and invite any researcher to join us on our quest.

[1] `http://www.ecrypt.eu.org/stream/index.html`

Bibliography

[AhoHU74] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley, 1974. AHO a 74:1 1.Ex. 172

[Arm02] F. Armknecht. A linearization attack on the Bluetooth keystream generator. Cryptology ePrint Archive, Report 2002/191, 2002. `http://eprint.iacr.org/`. 51, 101

[Arm04a] F. Armknecht. Improving fast algebraic attacks. In Roy and Meier [DBLP:conf/fse/2004], pages 65–82. 5, 7, 55, 92, 133, 136, 141, 148, 173

[Arm04b] F. Armknecht. On the existence of low-degree equations for algebraic attacks. Cryptology ePrint Archive, Report 2004/185, 2004. `http://eprint.iacr.org/`. 7, 80, 93, 99, 102

[Arm05a] F. Armknecht. Algebraic attacks and annihilators. In C. Wolf, S. Lucks, and P. Yau, editors, *WEWORC*, volume P-74 of *LNI*, pages 13–21, 2005. 5, 7, 80, 81

[ArmA05] F. Armknecht and G. Ars. Introducing a new variant of fast algebraic attacks and minimizing their successive data complexity. In E. Dawson and S. Vaudenay, editors, *Mycrypt*, volume 3715 of *Lecture Notes in Computer Science*, pages 16–32. Springer, 2005. 7, 126, 133, 163

[ArmBI06] F. Armknecht, J. Brandeis, and E. Ilinykh. Experimental results on algebraic attacks on stream ciphers. In *Proceedings of SICHERHEIT 2006*, Lecture Notes in Informatics, 2006. 70

[ArmCGKMR06] F. Armknecht, C. Carlet, P. Gaborit, S. Künzli, W. Meier, and O. Ruatta. Efficient computation of algebraic immunity for algebraic and fast algebraic attacks. will appear in the Proceedings of Eurocrypt 2006, 2006. 7, 80, 107, 175

[ArmK03] F. Armknecht and M. Krause. Algebraic attacks on combiners with memory. In Boneh [DBLP:conf/crypto/2003], pages 162–175. 5, 6, 43, 52, 55, 69, 70, 73, 75, 77, 80, 81, 83, 86, 92, 94, 97, 100, 126, 134, 152, 173

[ArmK06] F. Armknecht and M. Krause. Constructing single- and multi-output Boolean functions with maximal algebraic immunity. Accepted to ICALP'06, 2006. 112

[ArmKS05] F. Armknecht, M. Krause, and D. Stegemann. Design principles for combiners with memory. In Maitra et al. [DBLP:conf/indocrypt/2005], pages 104–117. 7, 113, 114

[ArmLP04] F. Armknecht, J. Lano, and B. Preneel. Extending the resynchronization attack. In *Selected Areas in Cryptography*, pages 19–38, 2004. 7, 117, 121, 122

[ArmM05] F. Armknecht and W. Meier. Fault attacks on combiners with memory. In Preneel and Tavares [DBLP:conf/sacrypt/2005], pages 36–50. 7, 116, 117, 121, 122

[Ars04] G. Ars. Private discussion. 72

[Ars05] G. Ars. *Applications des base de Gröbner à la cryptographie*. PhD thesis, Université de Rennes I, 2005. 63

[ArsF03] G. Ars and J.-C. Faugère. An algebraic cryptanalysis of nonlinear filter generators using Gröbner bases. INRIA Report 4739, 2003. `http://www.inria.fr/rrrt/rr-4739.html`. 65, 95

[ArsFIKS04] G. Ars, J.-C. Faugère, H. Imai, M. Kawazoe, and M. Sugita. Comparison between XL and Gröbner basis algorithms. In Lee [DBLP:conf/asiacrypt/2004], pages 338–353. 70

[BarFS03] M. Bardet, J.-C. Faugère, and B. Salvy. Complexity of Gröbner basis computation for semi-regular overdetermined sequences over GF(2) with solutions in GF(2). Research Report 5049, Inria, December 2003. 19 pages. 64

[BecW93] T. Becker and V. Weispfenning. *Gröbner bases - a computational approach to commutative algebra*. 71

[BekP82] H. Beker and F. Piper. *Cipher systems - The protection of communications*. 41

[BihGN05] E. Biham, L. Granboulan, and P. Nguyen. Impossible fault analysis of RC4 and differential fault analysis of RC4. In Gilbert and Handschuh [DBLP:conf/fse/2005], pages 359–367. 116

[BihS97] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In B. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997. 115

[BirS00] A. Biriyukov and A. Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In T. Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2000. 41

[Bla83] R. E. Blahut. *Theory and Practice of Error Control Codes*. 156

[Blu99] Bluetooth specification v1.1, 1999. `http://www.bluetooth.com/`. 12, 39

[BonDL97] D. Boneh, R. DeMillo, and R. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In *EUROCRYPT*, pages 37–51, 1997. 115

[BosFSS06] A. Bostan, P. Flajolet, B. Salvy, and É. Schost. Fast computation of special resultants. *Journal of Symbolic Computation*, 41(1):1–29, jan 2006. 149, 151, 152, 155

[BraL05] A. Braeken and J. Lano. On the (im)possibility of practical and secure nonlinear filters and combiners. In Preneel and Tavares [DBLP:conf/sacrypt/2005], pages 159–174. 195

[BraLMPV05] A.Braeken, J. Lano, N. Mentens, B. Preneel, and I. Verbauwhede. SFINKS: A synchronous stream cipher for restricted hardware environments. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/026, 2005. `http://www.ecrypt.eu.org/stream`. 195

[BraP05] A. Braeken and B. Preneel. On the algebraic immunity of symmetric Boolean functions. In Maitra et al. [DBLP:conf/indocrypt/2005], pages 35–48. 80, 113, 194, 195

[BreGY80] R. Brent, F. Gustavson, and D. Yun. Fast solution of Toeplitz system of equations and computation of Padè approximants. *J. Algorithms*, 1:259–295, 1980. 156

[BriGW98] M. Briceno, I. Goldberg, and D. Wagner. A pedagogical implementation of A5/1, 1998. http://jya.com/a51-pi.htm. 12, 35

[Buc65] B. Buchberger. Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. Dissertation an dem Math. Inst. der Universität von Innsbruck, 1965. 59

[CanT00] A. Canteaut and M. Trabbia. Improved fast correlation attacks using parity-check equations of weight 4 and 5. In Preneel [DBLP:conf/eurocrypt/2000], pages 573–588. 43

[CanV02] A. Canteaut and M. Videau. Degree of composition of highly nonlinear functions and applications to higher order differential cryptanalysis. In Knudsen [DBLP:conf/eurocrypt/2002], pages 518–533. 107

[Car04] C. Carlet. Improving the algebraic immunity of resilient and nonlinear functions and constructing bent functions. Cryptology ePrint Archive, Report 2004/276, 2004. `http://eprint.iacr.org/`. 80

[Car05] C. Carlet. A lower bound on the higher order nonlinearity of algebraic immune functions. Cryptology ePrint Archive, Report 2005/469, 2005. `http://eprint.iacr.org/`. 80

[CheJS00] V. Chepyzhov, T. Johansson, and B. Smeets. A simple algorithm for fast correlation attacks on stream ciphers. In Schneier [DBLP:conf/fse/2000], pages 181–195. 43

[CheS91] V. Chepyzhov and B. Smeets. On a fast correlation attack on certain stream ciphers. In *EUROCRYPT*, pages 176–185, 1991. 43

[ChoJM02] P. Chose, A. Joux, and M. Mitton. Fast correlation attacks: An algorithmic point of view. In Knudsen [DBLP:conf/eurocrypt/2002], pages 209–221. 43

[ChoP04] J. Cho and J. Pieprzyk. Algebraic attacks on SOBER-t32 and SOBER-t16 without stuttering. In Roy and Meier [DBLP:conf/fse/2004], pages 49–64. 73, 75

[CidL05] C. Cid and G. Leurent. An analysis of the XSL algorithm. In B. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 333–352. Springer, 2005. 70

[CooT65] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965. 132

[CopKM93] D. Coppersmith, H. Krawczyk, and Y. Mansour. The shrinking generator. In *CRYPTO '93: Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, pages 22–39, New York, NY, USA, 1994. Springer-Verlag New York, Inc. 35

[Cou02] N. Courtois. Higher order correlation attacks, XL algorithm and cryptanalysis of Toyocrypt. In P. Lee and C. Lim, editors, *ICISC*, volume 2587 of *Lecture Notes in Computer Science*, pages 182–199. Springer, 2002. 5, 70

[Cou03] N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In Boneh [DBLP:conf/crypto/2003], pages 176–194. 5, 55, 75, 77, 92, 124, 126, 130, 133, 134, 135, 136, 140, 152, 156, 157, 168, 173

[CouKPS00] N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Preneel [DBLP:conf/eurocrypt/2000], pages 392–407. 69

[CouM03] N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In E. Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer, 2003. 5, 43, 50, 75, 77, 80, 100, 134

[CouP02] N. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Y. Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer, 2002. 70

[CoxLS96] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer Verlag, 1996. 59, 62, 95

[DBLP:conf/asiacrypt/2004] P. Lee, editor. *Advances in Cryptology - ASIACRYPT 2004, 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004, Proceedings*, volume 3329 of *Lecture Notes in Computer Science*. Springer, 2004. 202, 210

[DBLP:conf/crypto/1999] M. Wiener, editor. *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999. 209

[DBLP:conf/crypto/2003] D. Boneh, editor. *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*. Springer, 2003. 202, 205, 208

[DBLP:conf/eurocrypt/1993] R. Rueppel, editor. *Advances in Cryptology - EUROCRYPT '93, 1993, Proceedings*, volume 658 of *Lecture Notes in Computer Science*. Springer, 1993. 207, 208

[DBLP:conf/eurocrypt/2000] B. Preneel, editor. *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*. Springer, 2000. 204, 205

[DBLP:conf/eurocrypt/2002] L. Knudsen, editor. *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*. Springer, 2002. 204, 209

[DBLP:conf/fse/2000] B. Schneier, editor. *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*. Springer, 2001. 204, 211

[DBLP:conf/fse/2004] B. Roy and W. Meier, editors. *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*. Springer, 2004. 201, 204, 210

[DBLP:conf/fse/2005] H. Gilbert and H. Handschuh, editors. *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*. Springer, 2005. 203, 207

[DBLP:conf/indocrypt/2005] S. Maitra, C. Madhavan, and R. Venkatesan, editors. *Progress in Cryptology - INDOCRYPT 2005, 6th International Conference on Cryptology in India, Bangalore, India, December 10-12, 2005, Proceedings*, volume 3797 of *Lecture Notes in Computer Science*. Springer, 2005. 202, 203

[DBLP:conf/sacrypt/2001] S. Vaudenay and A. Youssef, editors. *Selected Areas in Cryptography, 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16-17, 2001, Revised Papers*, volume 2259 of *Lecture Notes in Computer Science*. Springer, 2001. 208

[DBLP:conf/sacrypt/2005] B. Preneel and S. Tavares, editors. *Selected Areas in Cryptography, 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers*, volume 3897 of *Lecture Notes in Computer Science*. Springer, 2006. 202, 203

[DaeGV93] R. Govaerts J. Daemen and J. Vandewalle. Resynchronization weaknesses in synchronous stream siphers. In Rueppel [DBLP:conf/eurocrypt/1993], pages 159–167. 115

[DalGM04] D. Dalai, K. Gupta, and S. Maitra. Results on algebraic immunity for cryptographically significant Boolean functions. In A. Canteaut and K. Viswanathan, editors, *INDOCRYPT*, volume 3348 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2004. 80

[DalGM05] D. Dalai, K. Gupta, and S. Maitra. Cryptographically significant Boolean functions: Construction and analysis in terms of algebraic immunity. In Gilbert and Handschuh [DBLP:conf/fse/2005], pages 98–111. 80, 113, 196

[DalMS05] D. Dalai, S. Maitra, and S. Sarkar. Basic theory in construction of Boolean functions with maximum possible annihilator immunity. Cryptology ePrint Archive, Report 2005/229, 2005. `http://eprint.iacr.org/`. 80, 81, 113, 195

[Dor87] J. Dornstetter. On the equivalence between Berlekamp's and Euclid's algorithms. *IEEE Transactions of Information Theory*, IT-33(3):428–431, 1987. 156

[EkdJ02] P. Ekdahl and T. Johansson. A new version of the stream cipher SNOW. In K. Nyberg and H. Heys, editors, *Selected Areas in*

Cryptography, volume 2595 of *Lecture Notes in Computer Science*, pages 47–61. Springer, 2002. 122

[Fau02] J. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero F_5. In *International Symposium on Symbolic and Algebraic Computation Symposium - ISSAC 2002, Villeneuve d'Ascq, France*, 2002. 60

[Fau99] J. Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999. 60

[FauJ03] J. Faugère and A. Joux. Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using Gröbner bases. In Boneh [DBLP:conf/crypto/2003], pages 44–60. 64

[Fil00] E. Filiol. Decimation attack of stream ciphers. In B. Roy and E. Okamoto, editors, *INDOCRYPT*, volume 1977 of *Lecture Notes in Computer Science*, pages 31–42. Springer, 2000. 43

[FluL01] S. Fluhrer and S. Lucks. Analysis of the E_0 encryption system. In Vaudenay and Youssef [DBLP:conf/sacrypt/2001], pages 38–48. 77, 134

[FluMS01] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. In Vaudenay and Youssef [DBLP:conf/sacrypt/2001], pages 1–24. 12

[Gal63] R. Gallager. *Low-density paritiy check codes*. 43

[Gef73] P. Geffe. How to protect data with ciphers that are really hard to break. *Electronics*, 46(1):99–101, Jan 1973. 38

[Gol04] J. Golić. Vectorial Boolean functions and induced algebraic equations. Cryptology ePrint Archive, Report 2004/225, 2004. `http://eprint.iacr.org/`. 80, 157

[Gol82] S. Golomb. *Shift register sequences*. Aegean Park Press, 1982. 35, 41

[Gol93] J. Golić. Correlation via linear sequential circuit approximation of combiners with memory. In Rueppel [DBLP:conf/eurocrypt/1993], pages 113–123. 43

[Gol96] J. Golić. Correlation properties of of a general binary combiner with memory. *Journal of Cryptology*, 9(2):111–126, 1996. 43

[HasPS93] J. Håstad, S. Phillips, and S. Safra. A well-characterized approximation problem. *Inf. Process. Lett.*, 47(6):301–305, 1993. 59

[HawR04] P. Hawkes and G. Rose. Rewriting variables: The complexity of fast algebraic attacks on stream ciphers. In M. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 390–406. Springer, 2004. 126, 132, 133, 147, 148, 156, 162, 168, 172, 173

[HocS04] J. Hoch and A. Shamir. Fault analysis of stream ciphers. In M. Joye and J. Quisquater, editors, *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 240–253. Springer, 2004. 115, 116

[JohJ02] T. Johansson and F. Jönsson. Theoretical analysis of a correlation attack based on convolutional codes. *IEEE Transactions on Information Theory*, 48(8):2173–2181, 2002. 43

[JohJ99] T. Johansson and F. Jönsson. Improved fast correlation attacks on stream ciphers via convolutional codes. In *EUROCRYPT*, pages 347–362, 1999. 43

[JohJ99b] T. Johansson and F. Jönsson. Fast correlation attacks based on turbo code techniques. In Wiener [DBLP:conf/crypto/1999], pages 181–197. 43

[Ker83] A. Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, pages 161–191, 1883. 11

[Key76] E. Key. An analysis of the structure and complexity of nonlinear binary sequence generators. *IEEE Transactions in Information Theory*, 22(6):732–736, Nov 1976. 149

[KipS99] A. Kipnis and A. Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In Wiener [DBLP:conf/crypto/1999], pages 19–30. 69

[Knu81] D. Knuth. *The art of computer programming*, volume 2: Seminumerical algorithms. Addison-Wesley, second edition, 1981. 41

[Kra02] M. Krause. BDD-based cryptanalysis of keystream generators. In Knudsen [DBLP:conf/eurocrypt/2002], pages 222–237. 42, 43, 46, 77

[KraS06] M. Krause and D. Stegemann. Reducing the space complexity of BDD-based attacks on keystream generators. In V. Rijmen and M. Robshaw, editors, *FSE*, Lecture Notes in Computer Science. Springer, 2006. 42, 72

[KueM06] S. Künzli W. Meier. Private communication, 2006. 175

[LeeKHHM04] D. Lee, J. Kim, J. Hong, J. Han, and D. Moon. Algebraic attacks on summation generators. In Roy and Meier [DBLP:conf/fse/2004], pages 34–48. 73, 75

[LidN86] R. Lidl and H. Niederreiter. *Introduction to finite fields and their applications*. Cambridge University Press, New York, NY, USA, 1986. 24, 29, 137, 138, 139, 148, 154, 166

[Lob05] M. Lobanov. Tight bound between nonlinearity and algebraic immunity. Cryptology ePrint Archive, Report 2005/441, 2005. `http://eprint.iacr.org/`. 80

[LuMV05] Y. Lu, W. Meier, and S. Vaudenay. The conditional correlation attack: A practical attack on Bluetooth encryption. In V. Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 97–117. Springer, 2005. 76

[LuV04] Y. Lu 0002 and S. Vaudenay. Cryptanalysis of Bluetooth keystream generator two-level E_0. In Lee [DBLP:conf/asiacrypt/2004], pages 483–499. 76

[Mas69] J. Massey. Shift register synthesis and BCH decoding. *Journal of Complexity*, 15:122–127, 1969. 34

[Mau90] U. Maurer. A universal statistical test for random bit generators. In A. Menezes and S. Vanstone, editors, *CRYPTO*, volume 537 of *Lecture Notes in Computer Science*, pages 409–420. Springer, 1990. 41

[MeiPC04] W. Meier, E. Pasalic, and C. Carlet. Algebraic attacks and decomposition of Boolean functions. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 474–491. Springer, 2004. 5, 50, 80, 81, 97, 99, 100

[MeiS88] W. Meier and O. Staffelbach. Fast correlation attacks on certain stream ciphers. In C. Günther, editor, *EUROCRYPT*, volume 330 of *Lecture Notes in Computer Science*, pages 301–314. Springer, 1988. 43

[MeiS89] W. Meier and O. Staffelbach. Nonlinearity criteria for cryptographic functions. In *EUROCRYPT*, pages 549–562, 1989. 43, 113

[MeiS92] W. Meier and O. Staffelbach. Correlation properties of combiners with memory in stream ciphers. *Journal of Cryptology*, 5(1):67–86, 1992. 43

[MeiS94] W. Meier and O. Staffelbach. The self-shrinking generator. In A. De Santis, editor, *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 205–214. Springer, 1994. 35

[MihFI00] M. Mihaljevic, M. Fossorier, and H. Imai. A low-complexity and high-performance algorithm for the fast correlation attack. In Schneier [DBLP:conf/fse/2000], pages 196–212. 43

[MihFI01] M. Mihaljevic, M. Fossorier, and H. Imai. Fast correlation attack algorithm with list decoding and an application. In M. Matsui, editor, *FSE*, volume 2355 of *Lecture Notes in Computer Science*, pages 196–210. Springer, 2001. 43

[MihG90] M. Mihaljevic and J. Golic. A fast iterative algorithm for a shift register initial state reconstruction given the nosiy output sequence. In J. Seberry and J. Pieprzyk, editors, *AUSCRYPT*, volume 453 of *Lecture Notes in Computer Science*, pages 165–175. Springer, 1990. 43

[MihH02] M. Mihaljević and H. Imai. Cryptanalysis of Toyocrypt-HS1 stream cipher. *IEICE Transactions on Fundamentals*, E85-A:66–73, 2002. 134

[Pen96] W. T. Penzhorn. Correlation attacks on stream ciphers: Computing low-weight parity checks based on error-correcting codes. In D. Gollmann, editor, *Fast Software Encryption*, volume 1039 of *Lecture Notes in Computer Science*, pages 159–172. Springer, 1996. 43

[QuFL05] L. Qu, G. Feng, and C. Li. On the Boolean functions with maximum possible algebraic immunity : Construction and a lower bound of the count. Cryptology ePrint Archive, Report 2005/449, 2005. `http://eprint.iacr.org/`. 80, 113

[Rue85] R. Rueppel. Correlation immunity and the summation generator. In H. Williams, editor, *Crypto*, volume 218 of *Lecture Notes in Computer Science*, pages 260–272. Springer, 1985. 38, 43, 113

[Rue89] R. Rueppel. Security models and notions for stream ciphers. In H. Beker and F.Piper, editors, *Proceedings of 2nd IMA Conference on Cryptography and Coding*, pages 213–230. Oxford University Press, 1989. 11

[Rue92] R. Rueppel. Stream ciphers. In G. Simmons, editor, *Contemporary cryptology - The science of information integrity*, pages 65–134. IEEE Press, 1992. 11

[Saa02] M. Saarinen. A time-memory tradeoff attack against LILI-128. In J. Daemen and V. Rijmen, editors, *FSE*, volume 2365 of *Lecture Notes in Computer Science*, pages 231–236. Springer, 2002. 134

[Sch02] F. Schleer. Einsatz von OBDDs zur Kryptanalyse von Flusschiffren. Master's thesis, Universität Mannheim, 2002. 42

[Sch77] A. Schönhage. Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. *Acta Informatica*, 7:395–398, 1977. 149, 152, 172

[Sha48] C. Shannon. A mathematical theory of communication. *Bell Systems Techn. Journal*, 27:623–656, 1948. 10

[Sha49] C. Shannon. Communication theory of secrecy systems. *Bell Systems Techn. Journal*, 28:656–719, 1949. 10, 13, 59

[Sie84] T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions of Information Theory*, IT-30(5):776–780, 1984. 43, 113

[Sie85] T. Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Transactions of Information Theory*, C-34(1):81–85, 1985. 43

[SimDGM00] L. Simpson, E. Dawson, J. Golic, and W. Millan. LILI keystream generator. In D. Stinson and S. Tavares, editors, *Selected Areas in Cryptography*, volume 2012 of *Lecture Notes in Computer Science*, pages 248–261. Springer, 2000. 124

[SkoA02] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In B. Kaliski Jr., Ç. Koç, and C. Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12. Springer, 2002. 115

[Ste04] D. Stegemann. FBDD-basierte Kryptanalyse des A5/1-Schlüsselstromgenerators. Master's thesis, Universität Mannheim, 2004. 42

[Sti02] D. Stinson. *Cryptography - Theory and Practice*. 10, 13, 14

[Str69] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969. 66, 69

[Ver26] G. Vernam. Cipher printing telegraph system for secret wire and radio telegraphic communications. *Journal of American Institute of Electrical Engineers*, 45:109–115, 1926. 13

[Weg87] I. Wegener. *The complexity of Boolean functions*. J. Wiley & Sons, Inc., New York, NY, USA, 1987. 42, 102

[Wol86] S. Wolfram. Random sequence generation by cellular automata. *Advances in Applied Mathematics*, 7:123–169, 1986. 37

[XiaM88] G. Xiao and J. Massey. A spectral characterization of correlation-immune combining functions. *IEEE Transactions of Information Theory*, 34(3):569–571, 1988. 43

[Zen04] E. Zenner. *On cryptographic properties of LFSR-based pseudorandom generators*. PhD thesis, Universität Mannheim, 2004. 11, 41, 42

[Zen06] E. Zenner. Private communication, 2006. 72

[ZenH88] K. Zeng and M. Huang. On the linear syndrome method in cryptoanalysis. In S. Goldwasser, editor, *CRYPTO*, volume 403 of *Lecture Notes in Computer Science*, pages 469–478. Springer, 1988. 43

[ZenWL00] E. Zenner, R. Weis, and S. Lucks. Sicherheit des GSM-Verschlüsselungsstandards A5. *Datenschutz und Datensicherheit*, 24(7), 2000. 12

[ZenYR89] K. Zeng, C. Yang, and T. Rao. On the linear consistency test (LCT) in cryptanalysis with applications. In G. Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 164–174. Springer, 1989. 41, 43, 110

[Ziv91a] M. Zivkovic. On two probabilistic decoding algorithms for binary linear codes. *IEEE Transactions on Information Theory*, 37(6):1707–, 1991. 43

Index